T.C.

FIRAT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL ROBOTLARDA PROGRAMLANABİLİR KAPI DİZİLERİ ALANI KULLANILARAK GERÇEK ZAMANLI MODELLEME

Gökşen ÖZDEMİR

Tez Yöneticisi
Yrd. Doç. Dr. Hayrettin CAN

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ELAZIĞ, 2008

T.C.

FIRAT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL ROBOTLARDA PROGRAMLANABİLİR KAPI DİZİLERİ ALANI KULLANILARAK GERÇEK ZAMANLI MODELLEME

Gökşen ÖZDEMİR

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Bu tez, 09/01/2009 tarihinde aşağıda belirtilen jüri tarafından oybirliği ile başarılı olarak değerlendirilmiştir.

Danışman : Yrd.Doç.Dr. Hayrettin CAN

Üye : Prof. Dr. Erhan AKIN

Üye : Yrd. Doç. Dr. Ahmet ORHAN

Üye :

Üye :

Bu seminerin kabulü, Fen Bilimleri Enstitüsü Yönetim Kurulu' nun .... / ..... / ..... tarih ve ..... sayılı kararıyla onaylanmıştır.

TEŞEKKÜR

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

Tez çalışmam süresince gösterdiği büyük ilgi, destek ve her türlü yardımlarından dolayı Cuma Ali MANTAŞ'a sonsuz şükranlarımı sunarım.

Bu zamana kadarki eğitimim süresince desteklerini her an üzerimde hissettiğim sevgili aileme, gösterdikleri sabır ve verdikleri emeklerden dolayı minnettarlarımı sunarım.

İÇİNDEKİLER

Sayfa no

ŞEKİLLER LİSTESİ

Sayfa no

TABLOLAR LİSTESİ

Sayfa no

KISALTMALAR LİSTESİ

FPGA: Field Programmable Gate Array (Programlanabilir Kapı Dizileri Alanı)

CPLD: Complex Programmable Logic Devices (Karmaşık Programlanabilir Lojik Cihazlar)

ASIC: Application Specific Integrated Circuits (Uygulamaya Özgü Tümleşik Devreler)

PLD: Programmable Logic Devices (Programlanabilir Lojik Cihazlar)

ROM: Read Only Memory (Yalnızca Okunabilir Hafıza)

CLB: Configurable Logic Block (Konfigüre Edilebilir Lojik Blok)

LUT: Look-Up Table (Look-Up Tablosu)

PLL: Phase-Locked Loop (Faz Kapalı Döngüsü)

HDL: Hardware Description Language (Donanım Tanımlama Dili)

VHDL: VHSIC (Very High-Speed Integrated Circuit) Hardware Description Language (Çok Hızlı Tümleşik Devreler İçin Donanım Tanımlama Dili)

ÖZET

Yüksek Lisans Tezi

MOBİL ROBOTLARDA PROGRAMLANABİLİR KAPI DİZİLERİ ALANI KULLANILARAK GERÇEK ZAMANLI MODELLEME

Gökşen ÖZDEMİR

Fırat Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
2008, Sayfa : 48

Programlanabilir Kapı Dizileri Alanı (Filed Programable Gate Array, FPGA) yapılandırılabilir mantık blokları ile birlikte bu bloklar arasındaki değiştirilebilir ara bağlantılardan oluşan sayısal tümleşik devrelerdir. FPGA'lar hızlı işlem gerektiren gerçek zamanlı uygulamalarda ve paralel işlem gerektiren özel işlemler için kullanılır. Tipik bir FPGA altmış dörtten yüzbinlere kadar lojik blok ve flip-flop içerir. Kullanıcının tasarladığı devreye göre, FPGA üreticisi tarafından sağlanan bir yazılım sayesinde lojik bloklar ve aralarındaki bağlantılar programlanır.

Bu çalışmada, FPGA geliştirme kartı kullanılarak bir omni-directional robot modeli gerçekleştirilmektedir. Robot üç tane omni-directional tekerleğe sahiptir ve bu tekerlekler üzerinde serbestçe hareket eden ufak tekerlekler bulunmaktadır. Bu sayede robot her yöne hareket etme yeteneğine sahip olmaktadır.

Bu tez çalışmasında, ilk olarak omni-directional robotun ters kinematik ve dinamik modelleri Matlab'da oluşturulmuştur. Sonra farklı yörüngeler için robot simülasyonu gerçekleştirilmiştir. Daha sonra robot modelinin FPGA uygulaması gerçek zamanlı uygulamalar için kullanılan Altera'nın geliştirme kartı ve araçları ile gerçekleştirilmiştir. Son olarak, omni-directional robot modelinin gerçek zamanlı FPGA uygulamasını doğrulamak için sonuçlar karşılaştırılmıştır.

Anahtar Kelimeler : Gerçek zamanlı modelleme, Omni-directional robot, FPGA

ABSTRACT

Master Thesis

MODELING MOBILE ROBOTS BY USING FPGA (FIELD PROGRAMMABLE GATE ARRAY)

Gökşen ÖZDEMİR

Firat University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering
2008 , Page : 48

Field Programmable Gate Array (FPGA)s are integrated circuits that are consist of can be configured logic blocks and changable intermediate connections among those blocks. FPGA's are used in real time applications that need fast processing and for special processings that need parallel operation. A typical FPGA contains flip-flops and logic blocks from sixty four to hundreds of thousands. Depending on the circuit that designed by user, logic blocks and intermediate connections are programmed by a software provided from FPGA producer.

In this study, an omni-directional robot model implemented by using a FPGA development board. The robot has three omni-directional wheels on which free-turning rollers are placed. Therefore the robot has the ability to move any direction.

In this thesis, first, inverse kinematic and dynamic model of the omni-directional robot are developed in Matlab. Next, the simulation of the robot is realized for different trajectories. Then, the FPGA implementation of the robot model is performed using Altera chips and tools for real time applications. Finally, the results are compared to verify the real time implementation of the omni-directional robot model.

Key Words : Real time modelling, Omni-directional robot, FPGA

1. GİRİŞ

Tümleşik devreler, genel olarak sayısal ve analog işlem sistemleri olarak iki ana başlık altında toplanır. Sayısal işlem sistemleri, sayısal verileri işlemek için tasarlanmış, hızlı donanımlara ve bu donanımlara işlevsellik kazandıracak esnek yazılımlara ihtiyaç duyan yapılardır. Sayısal işlem sistemi oluşturmada, donanım ve yazılım tabanlı olmak üzere iki geleneksek yöntem uygulanır.

Donanım tabanlı yöntem : Sayısal verileri işlemek için ağırlıklı olarak özel tüm devreler (ASIC) kullanılır. Bu tip devreler özel bir fonksiyonu gerçekleştirmek amacıyla üretildiğinden, bu fonksiyonları etkin ve hızlı bir şekilde gerçeklerler. Ancak işlevleri sınırlıdır ve sadece ilgili oldukları uygulamaya yönelik üretilmişlerdir. ASIC devreler doğru ve hızlı sonuç vermesine rağmen çözüm ürettiği problemin çeşitli türevleri için kullanılamayacaklardır. Yeni problemler için yeni donanımlara ve yeni ASIC yapılara ihtiyaç duyulur. Bu da maliyet artışına ve zaman kaybına neden olur.

Yazılım tabanlı yöntem **:** Bu yöntemde tasarım değişikliklerine daha esnek olan mikroişlemciler kullanılır. Mikroişlemcinin koşturduğu yazılımlar değiştirilerek, hiçbir donanım değişikliğine gidilmeden tasarım ortamına yeni fonksiyonlar eklenebilir. Mikroişlemciler üzerinde koşan, yazılım uygulamaları aynı anda birçok işlemi yerine getirmek için tek bir işlemcinin genel kaynaklarını kullanırken, yavaş fakat esnek yazılımlar ile çalışırlar. Fakat sıradan bir uygulama için komutların bellekten okunması, onların yorumlanıp yerine getirilmesi, sistemin performansı ve hızını oldukça düşürmektedir.

Günümüzde sayısal işlem sistemlerinin kullandığı alandaki hızlı gelişmeler, üretim tamamlandıktan sonra da esnek genel amaçlı olacak şekilde tasarlanan işlem sistemlerini ortaya çıkarmıştır. Özellikle iyi tasarlanmış ve işlemci yükünü azaltan, paylaşan donanımlar, performans artışı için iyi bir çözüm olabilir. Bununla beraber işlevleri uygulama sırasında değiştirilebilen programlanabilir devre elemanlarının kullanımı da avantaj getirecektir. Bu devre elemanları ile gerçekleştirilen donanımlar, ASIC gibi devre elemanları ile yapılan klasik donanımlara göre daha işlevseldirler.

Tekrar düzenlenebilen işlem sistemleri (Reconfigurable Computing System) olarak da adlandırılan bu sistemler, esnek ve genel amaçlı yapıları sayesinde yeni bir üretim aşamasına ihtiyaç duymadan, değişen protokollere, sistem özelliklerine ve kullanıcı ihtiyaçlarına kısa sürede cevap verebilirler. Yine bu özellikleri sayesinde tekrar düzenlenebilen işlem sistemleri, donanım tabanlı sayısal işlem sistemlerine göre daha esnek ve yazılım tabanlı sayısal işlem sistemlerine göre daha hızlı sayısal tasarım ortamı oluşturarak, bu iki sistem arasındaki boşluğu doldururlar.

Tekrar düzenlenebilir sayısal işlem sistemlerinin ihtiyaç duyduğu esnek donanımlar, Programlanabilir Kapı Dizileri Alanı (Filed Programable Gate Array, FPGA) kullanılarak karşılanır. FPGA'ler yapılandırılabilir mantık blokları ile birlikte bu bloklar arasındaki değiştirilebilir ara bağlantılardan oluşan sayısal tümleşik devrelerdir. Özellikle SRAM (Statik Rastgele Erişimli Bellek) tabanlı FPGA'lar tekrar düzenlenebilirlik kabiliyetleri ve yüksek performanslı uygulamalardaki yeterlilikleri sayesinde, genel amaçlı sayısal donanımların tasarımında anahtar rol oynarlar. Tipik bir FPGA altmış dörtten yüzbinlere kadar lojik blok ve flip-flop içerir [9]. Kullanıcının tasarladığı devreye göre, FPGA üreticisi tarafından sağlanan bir yazılım sayesinde 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'lar şifreleme , filtreleme gibi hızlı işlem gerektiren uygulamalarda ve paralel işlem gerektiren özel işlemler için kullanılır.

Donanım tabanlı sistemlerin yazılım tabanlı sistemlere olan en önemli üstünlüğü hızdır. Hızın en kritik olduğu sistemlerden biri de gerçek zamanlı kontrol sistemleridir. Bu tez çalışmasında FPGA gerçek zamanlı robot modellemesinde kullanılmaktadır. FPGA kullanılarak gerçekleştirilen robot modeli ileriki çalışmalarda gerçek zamanlı bir kontrol sistemi tasarımında kullanılabilecektir.

Robot modellemesi yazılım ve donanımın birlikte kullanılması ile gerçekleştirilebilmektedir. Ancak yazılım algoritmalarının uzaması çalışma periyodu (execution cycle) zamanını arttırmakta ve bu durum modelleme performansını düşürmektedir. Bu tezde amaç yazılım ile gerçekleştirilen bir çok modülün FPGA kullanılarak donanımsal olarak gerçekleştirilmesidir.

.

2. PROGRAMLANABİLİR KAPI DİZİLERİ ALANI (FPGA)

2.1. Sayısal Tümdevreler

Sayısal tümdevreler genel olarak aşağıdaki şekilde sınıflandırılabilir.

- Standart Lojik
  - TTL (74xxx)
  - CMOS (4xxx)
- Programlanabilir Lojik
  - PLD (Programmable Logic Devices)
  - FPGA (Field Programmable Gate Array)
  - CPLD (Complex Programmable Logic Devices)
- ASIC (Application Specific Integrated Circuits)
  - Kapı Dizileri
  - Standart Hücre
- VLSI (Very Large Scale Integrated)
  - Mikroişlemciler & RAM

TTL veya CMOS ailesine mensup tümleşik entegreler, üretici firma tarafından belirlenen sabit bir fonksiyonu yerine getirirler ve kullanıma hazır halde satılırlar. Herhangi bir tasarımı gerçekleştirmek için bu entegrelerden farklı işlevsellikteki birkaç tanesinin birlikte kullanılması gerekmektedir.

Uygulamaya Özgü Tümleşik Devreler (ASICs), Karmaşık Programlanabilir Lojik Cihazlar(CPLDs) ve Programlanabilir Kapı Dizileri Alanı (FPGAs) ailelerine ait entegreler programlanabilir teknolojiler ile üretilmektedirler. Yani entegrelerin gerçekleştirecekleri fonksiyonlar kullanıcı tarafından belirlenir ve programlanır. Bunlar içerisinde uygulamaya özgü tümleşik devreler(ASICs) diğerlerinden farklı olarak ek bir programlama aşaması gerektirirler ve bu aşama üretici firma tarafından kullanıcının belirlediği şekilde gerçekleştirilir. CPLD ve FPGA entegrelerinin fonksiyonları ise sadece kullanıcı tarafından yapılan programlama işlemine bağlıdır.

İlk programlanabilir tümleşik devreler Programlanabilir Lojik Cihazlar(PLD) olarak takdim edilir. Programlanabilir lojik teknolojinin en sade halidir ve otuz yılı aşkın bir süredir kullanılmaktadır. Bu teknolojide lojik ifade en aza indirgenir ve çarpımların toplamı biçiminde yazılır.

Kişisel bilgisayarlarda kullanılan hafıza ve işlemci entegrelerinin üyesi olduğu Çok Büyük Ölçekli Entegre(VLSI) teknolojisi, pahalı bir geliştirme süreci gerektirir ve sadece sektördeki en büyük entegrelerin tasarımında kullanılır.

2.2. FPGA Teknolojisi ve Programlanması

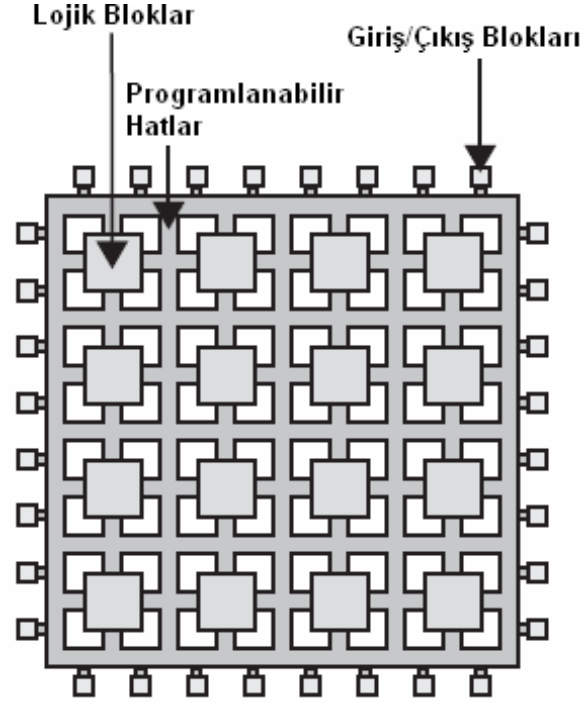
Programlanabilir kapı dizileri alanı, elektriksel olarak programlanabilir eleman ve arabirimlerden oluşan bir tüm devredir. Programlanabilir kapı dizileri alanı kısaca FPGA (field pogrammable gate array) olarak adlandırılır. Programlanabilir eleman ve arabirimler, AND, OR, XOR, NOT gibi temel lojik işlemlerini veya daha karmaşık olan dekoder, multiplekser gibi matematiksel işlemleri gerçekleştirmek amacıyla programlanabilir. FPGA yapısında bulunan arabirimler tasarımdaki bağlantılar göz önüne alınarak elektriksel olarak programlanabilir ve istenildiği kadar programlanabilme yapısına sahip olduğundan tasarımlarda büyük kolaylık sağlar.

FPGA'lar şifreleme , filtreleme gibi hızlı işlem gerektiren uygulamalarda ve paralel işlem gerektiren özel işlemler için kullanılır. İlk olarak 1980 ortalarında ara yapıştırıcı mantık ve kısıtlı veri işleme şeklinde kullanılmıştır. 2000'lerde ise; milyonlarca kapı ve buna ek olarak gömülü mikroişlemci çekirdekleri, yüksek hızlı I/O arayüzleri, gömülü RAM ve DSP öbekleri içeren yüksek performanslı modeller geliştirilmiştir.

Tipik bir FPGA altmış dörtten yüzbinlere kadar lojik blok ve flip-flop içerir. FPGA'lar temelde üç bloktan oluşur; lojik bloklar(CLB), giriş-çıkış blokları(I/O) ve bağlantı blokları. Lojik bloklar boolean fonksiyonlarının gerçekleştirildiği yapılardır. Giriş/Çıkış blokları FPGA'nın dış kenarlarında bulunur. Entegre devrenin paket bacakları ile iç bağlantılar arasındaki ilişkiyi oluşturur. FPGA'ya veri giriş çıkışını sağlarlar. Bağlantı blokları lojik bloklarla giriş-çıkış blokları arasındaki bağlantıyı sağlayan yapılardır. Bu yapılar, yollandırma kanalları ve programlanabilir anahtarlardan oluşur.

Pek çok FPGA yapısı programlanabilir eleman ve ara birimlere ek olarak hafıza birimleri de bulundururlar. Bu hafıza birimleri ayrık flip-flop yapılarından veya hafıza bloklarından oluşabilir.

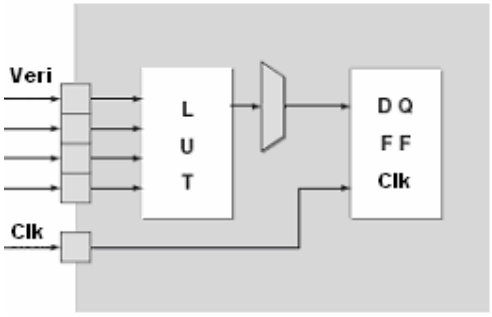
Şekil 2.1 'de genel FPGA mimarisi verilmiştir.



Şekil 2.1 Genel FPGA mimarisi

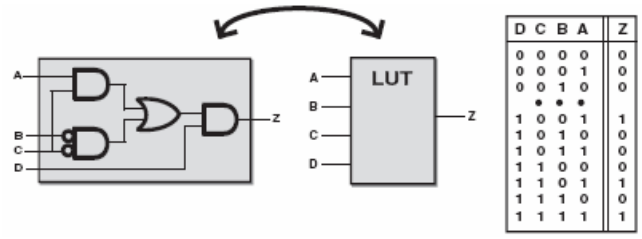
Şekil 2.1'de görüldüğü gibi lojik bloklar, entegre içerisinde matris biçiminde yerleştirilmiştir. Eğer gerçekleştirilecek lojik fonksiyon bir lojik bloğun içerdiği elemanlar ile mümkün değilse birden fazla lojik blok birbirlerine bağlanır. Lojik blokların birbirlerine bağlanmasını sağlayan programlanabilir bu yapıya ara bağlantı ağı denir [6].

Lojik bloklar genel olarak 5 girişten oluşular, bu girişlerden 4 ü LUT (look-up table) girişleri iken diğer giriş ise D tipi flip flop yapısının saat girişidir. Lojik bloklar, içinde bulunan LUT ve flip-flop sayesinde çıkış alır. Lojik bloğun çıkışını LUT'un giriş çıkış fonksiyonu belirlemektedir. Konfigüre edilebilir lojik blok yapısına ait genel blok şeması Şekil 2.2'de verildiği gibidir.



Şekil 2.2 Lojik blok yapısı

FPGA içerisinde sistem lojik bloklara bölünür ve bloklar içerisindeki LUT'lar tipik olarak 4 girişli olup girişine göre çıkış fonksiyonu oluştururlar. Şekil 2.3 'de örnek olarak bir LUT'un iç yapısı gösterilmiştir [5].



Şekil 2.3 Örnek bir LUT (look-up table)'ın iç yapısı

Günümüzde FPGA'ların üretim safhasında genel olarak kullanılan iki çeşit programlama teknolojisi bulunmaktadır; SRAM ve Antifuse. Her iki teknolojinin birbirilerine göre çeşitli avantaj ve dezavantajları bulunmaktadır. Bunun dışında Altera firmasının MAX ailesi FPGA'larda EPROM ve EEPROM teknolojileri de bulunmaktadır. Bir SRAM hücresi tekrar programlanabilir yapıdadır fakat aynı zamanda uçucudurlar, yani devrenin gücü kesildiğinde programlanmış hücredeki veriler kaybolur. Antifüse hücresindeki veri kalıcıdır fakat hücreyi tekrar programlama imkanı vermez [7].

Piyasada birkaç farklı FPGA üretici firması bulunmaktadır. Bunlardan bazıları; Actel, Altera, Anadigm, Atmel, Lattice Semiconductor, Leopard Logic, QuickLogic, Xilinx firmalarıdır. Özellikle Xilinx ve Altera firmalarının üretmiş olduğu FPGA'lar geniş kullanım alanı bulmaktadırlar. Bu firmalardan Xilinx'in Virtex ve Spartan aileleri ile Altera'nın Stratix ve Cyclon aileleri piyasanın tercih ettiği FPGA'lardır.

Bu tez çalışmasında, Altera firmasının üretmiş olduğu Stratix ailesinden Stratix II GX model FPGA kullanılmıştır. Kullandığımız FPGA ve geliştirme kartının teknik özellikleri tezin ileriki bölümlerinde verilecektir.

Kullanıcının tasarladığı devreye göre, FPGA üreticisi tarafından sağlanan bir yazılım sayesinde 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 yapısının nasıl davranacağının belirlenmesi amacıyla tasarımcı donanım tanımlama dillerini (HDL) veya şematik tasarım araçlarını kullanır. Donanım tanımlama dillerinin en çok bilinenleri VHDL ve Verilog-HDL dilleridir. Tasarımcı ilk olarak gerçekleştirmek istediği fonksiyona ait şematik veya HDL dillerinden birinde tasarımını gerçekleştirir ve gerekli simülasyonları yapar. Bu simülasyonlar sonucunda tasarım istenildiği gibi çalışıyorsa, devrenin son hali üretici firmanın sunmuş olduğu paralel kablolar yardımıyla bilgisayar kullanılarak FPGA üzerine yüklenir ve FPGA programlanabilir birimi istenildiği işi yapmak üzere programlanmış olur [3].

2.3. FPGA Modellemesinde IEEE-754 Kayan Noktalı Sayı Formatı Kullanımı

FPGA'larda, AND, OR, XOR, NOT gibi temel lojik işlemlerle birlikte daha karmaşık matematiksel işlemleri gerçekleştirmek mümkündür. Kayan noktalı sayı işlemleri için farklı sayı formatları kullanılabilmektedir. En yaygın olarak kullanılan format IEEE-754 kayan noktalı (floating point) sayı formatıdır. Bu çalışmada FPGA'da gerçekleştirilen kayan noktalı sayı işlemlerinde 32 bitlik IEEE-754 kayan noktalı sayı formatı kullanılmıştır. Dolayısıyla FPGA'da kayan noktalı sayı işlemleri için kullanılacak olan fonksiyonlara giriş olarak bu formata dönüştürülmüş sayısal değerlerin uygulanması gerekmekte ve aynı şekilde çıkış olarak da bu formatta sayısal değerler üretilmektedir.

IEEE-754 kayan noktalı sayı formatı günümüzde gerçek sayıları temsil etmek üzere kullanılan en yaygın gösterim şeklidir. Bilgisayarda gerçek sayıları temsil etmenin birkaç farklı yolu vardır. Örneğin sabit noktalı (fixed point) sayı gösterimi ile her sayı 2 integer sayının oranı şeklinde gösterilebilir. Sabit noktalı sayılarda sayının noktadan önceki ve sonraki kısımları için sabit uzunlukta yerler ayrılır. Bu gösterim pratik değildir ve bellekte fazla yer kaplar. Dolayısıyla işlem yapmak için uygun değildir.

IEEE-754 kayan noktalı sayı formatında ise gerçek sayılar bilimsel bir notasyonla gösterilir. IEEE-754 kayan noktalı sayı formatında daha az bit sayısı ile sayıları temsil edebilmek için normalize edilmiş form (normalized form) kullanılır. Böylece kayan noktalı sayılar sabit noktalı sayı formatına göre hafızada daha az yer kaplayarak tutulmuş olur.

Sabit noktalı sayı formatında temsil edilemeyen sonsuz, belirsiz gibi bazı özel değerler IEEE-754 kayan noktalı sayı formatında temsil edilebilmektedir.

IEEE-754 kayan noktalı sayı formatındaki sayılar 3 temel parçadan oluşur. Bunlar:

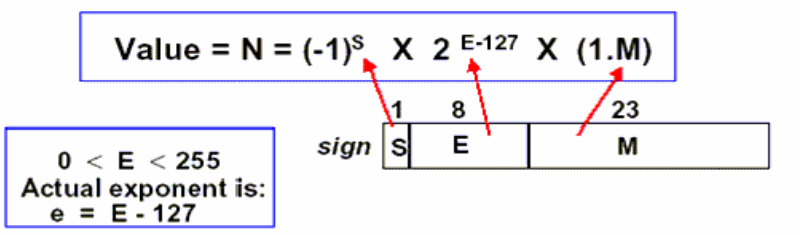
- Sign
- Exponent
- Mantissa

IEEE-754 de 32 bitlik ve 64 bitlik olmak üzere 2 tür gösterim mevcuttur. Her 2 gösterim için sign, exponent ve mantissanın durumları:

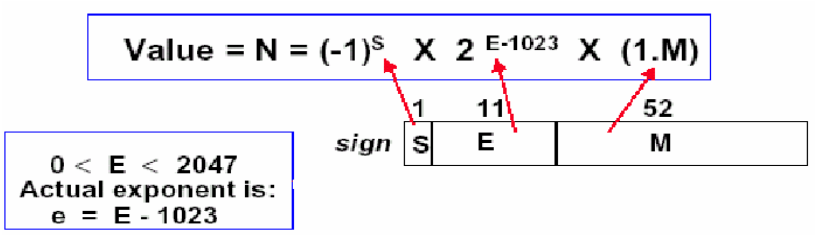
	Sign	Exponent	Fraction	Bias
32 bitlik	1 bit [0]	8 bit [1-8]	23 bit [9-31]	127
64 bitlik	1 bit [0]	11 bit [1-11]	52 bit [12-63]	1023

şeklindedir.

Şekil 2.4 ’de 32 bitlik IEEE-754 standardı, Şekil 2.5 ’de ise 64 bitlik IEEE-754 standardı tanımlanmıştır.



Şekil 2.4 32 bitlik IEEE-754 standardı



Şekil 2.5 64 bitlik IEEE-754 standardı

Sign Biti: Sign bitini elde etmek oldukça kolaydır.Pozitif sayılar için 0, negatif sayılar için 1 değerini alır.

Exponent: Exponent alanı hem negatif hem de pozitif üsleri temsil edebilmektedir. Bunu gerçekleştirmek için bir bias değeri gerçek üs değeriyle toplanıp exponent kısmı oluşturulur.

IEEE-754 32 bitlik gösterimi için bias değeri 127, 64 bitlik gösterim için de 1023 tür. Buna göre gerçek üssün 0 olması exponent alanında saklanan değerin 127 olacağı anlamına gelir.

Mantissa: Mantissa sayıyı ifade eden bitleri gösterir. Bu, sayının tam ve kesir (fraction) kısımlarını gösteren bitlerden oluşur.

2.3.1. Normalize Edilmiş Form

Daha az bit sayısı ile sayıları temsil edebilmek için normalize edilmiş form (normalized form) kullanılır. Bu temel olarak radix pointi 0 dan farklı ilk digitin sonuna koyar. Normalize edilmiş formda 5 sayısı 5.0×10^0 olarak gösterilir.

2 lik tabanda 0 dan farklı tek rakam 1 olduğu için normalize edilmiş formda sayının tam kısmı her zaman 1 olur ve bunun açık bir şekilde tutulmasına gerek kalmaz. Çünkü bu kısım tüm sayılar için aynı hale gelmiş olur. Farklı olan kesir bitleridir.

Örneğin 11110000 11001100 10101010 00000000 sayısı

$+1.1110000\ 11001100\ 10101010 \times 2^{31}$

şeklinde gösterilir.

Bu formatta belirgin 5 tür kayan noktalı (floating point) sayı gösterilemez. 32 bitlik gösterim için;

- $-(2-2^{-23}) \times 2^{127}$ den daha küçük negatif sayılar (*negative overflow*)
- -2^{-149} den daha büyük negatif sayılar (*negative underflow*)
- Sıfır
- 2^{-149} den daha küçük pozitif sayılar (*positive underflow*)
- $(2-2^{-23}) \times 2^{127}$ den daha büyük pozitif (*positive overflow*)

2.3.2. Özel Değerler

IEEE-754 'de exponent alanında tamamen 1 yada 0 saklanarak özel değerler elde edilir. Bunlar;

Sıfır: Exponent ve fraction alanlarının tamamen 0 dan oluşması sıfıra karşılık gelir.

Normalize Edilmemiş (Denormalized) Sayı: Eğer exponent alanının tüm bitleri 0 ve fraction alanı ise 0 dan farklıysa bu sayı normalize edilmemiş (denormalized) bir sayıdır. 32 bitlik gösterimde $(-1)^{s} \times 0.f \times 2^{126}$, 64 bitlik gösteride $(-1)^{s} \times 0.f \times 2^{1022}$ şeklinde tanımlıdır.

Sonsuz: + sonsuz ve – sonsuz tüm exponent bitlerinin 1 ve fraction bitlerinin de 0 olması şeklinde tanımlanır. Sign biti + ve – sonsuzu birbirinden ayırır.

Belirsizlik: Tüm exponent bitlerinin 1 ve fraction bitlerinin ise 0 dan farklı olması durumudur.

Tablo 2.1 'de IEEE-754 kayan noktalı sayı formatında özel değerlerin gösterimi verilmiştir.

Tablo 2.1 IEEE-754 özel sayı gösterimi

32 Bitlik Hassasiyet		**64 Bitlik Hassasiyet**		**Temsil Edilmekte Olduğu Sayı**
Exponent	**Mantissa**	**Exponent**	**Mantissa**	
0	0	0	0	**0**
0	sıfırdan farklı	0	sıfırdan farklı	**Normalize Edilmemiş (Denormalized) Sayı**
1 ile 254 arası	herhangi bir şey	1 ile 2046 arası	herhangi bir şey	**Kayan Noktalı (Floating Point) Sayı**
255	0	2047	0	**Sonsuz**
255	sıfırdan farklı	2047	sıfırdan farklı	**Belirsiz**

2.3.3. IEEE-754 Formatına Dönüşüm

Bu tez çalışmasında kayan noktalı sayı işlemlerinde 32 bitlik format kullanılmıştır. Aşağıda örnek bir format dönüşümü gösterilmektedir.

Örneğin onluk tabandaki "-2345.125_{10}" sayısının 32 bitlik IEEE-754 kayan noktalı (floating point) sayı formatına dönüşümü şu şekilde gerçekleştirilmektedir.

Öncelikle onluk tabandaki sayı ikilik tabana çevrilir. Bu durumda;

$-2345.125_{10} = -100100101001.001_2$

İkilik tabana çevrilen sayı yukarıda anlatıldığı üzere 1.M formuna yani normalize edilmiş forma dönüştürülür. Böylece

$-1.00100101001001 \times 2^{11}$

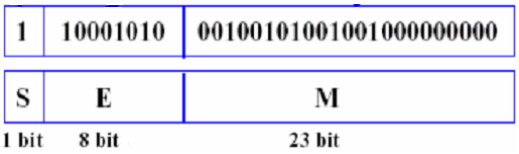
elde edilir.

Sayı negatif olduğu için IEEE-754 sayı formatındaki işaret biti S = 1 olarak belirlenir.

Üsse bias değeri (127) eklenerek 8 bitlik E (exponent) hesaplanır. E = e + 127 olduğundan $E = 11 + 127 = 138_{10} = 10001010_2$ olur.

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

Hesaplanan S, E ve M parçaları birleştirilerek sayının IEEE-754 sayı formatındaki 32 bitlik karşılığı elde edilir.



Adım adım gösterilen format dönüşümü ve bunun ters dönüşümü için matlab ortamında bazı hazır fonksiyonlar bulunmaktadır. Bu çalışmada FPGA'ın kayan noktalı sayı işlemlerinde kullanılan sayılar, matlab fonksiyonları ile IEEE-754 formatına dönüştürülmüş ve aynı formattaki çıkış değerlerinin de yine ilgili matlab fonksiyonları ile ters dönüşümü sağlanmıştır.

Kayan noktalı sayının 32 bitlik IEEE-754 kayan noktalı sayı formatına dönüşümü için kullanılan matlab fonksiyonu *num2hex(single(sayi))* şeklindedir.

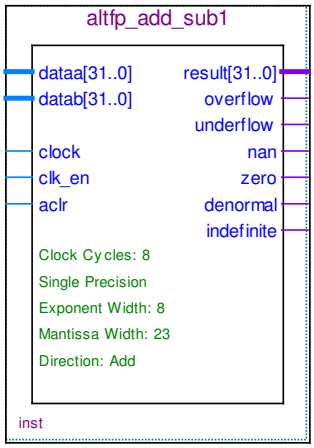
Ters dönüşüm için de *q=quantizer('single','nearest','saturate')* ve *hex2num(q,sayi)* matlab fonksiyonları kullanılır.

2.4. FPGA'da Kayan Noktalı Sayıların Aritmetiksel İşlemleri

Bu tez çalışmasında FPGA tasarımında kayan noktalı sayılar için IEEE-754 kayan noktalı sayı formatı kullanılmıştır. Bu yüzden gerekli tüm kayan noktalı sayı işlemleri bu formatın kurallarına göre gerçekleştirilmektedir. Aşağıda sırasıyla toplama, çıkarma, çarpma ve bölme aritmetiksel işlemleri için ilgili fonksiyon ve çalışma prensibi anlatılmaktadır. Aşağıda sıralanan kayan noktalı sayılardaki bu aritmetiksel işlemler, robot modeli FPGA ortamında gerçekleştirilirken kullanılmıştır.

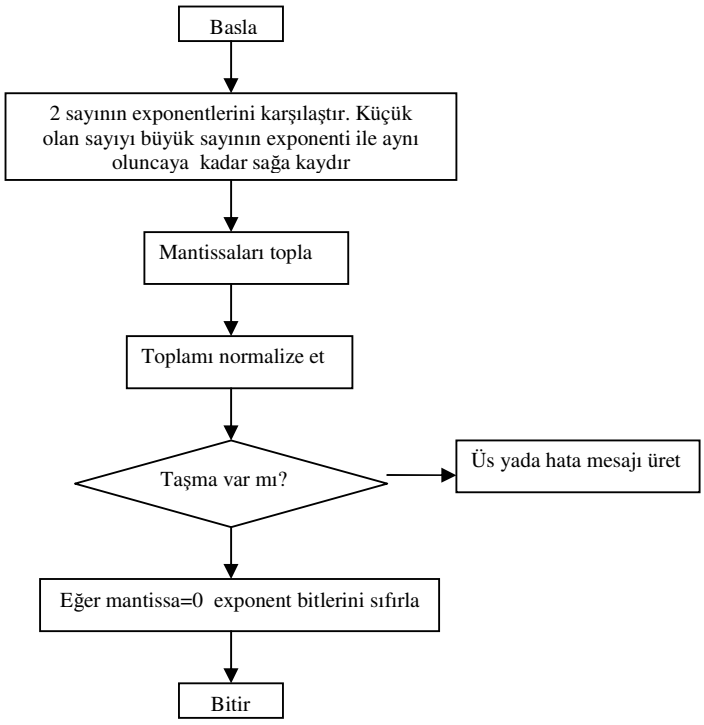
2.4.1. Toplama İşlemi

Şekil 2.6 'da 32 bitlik toplama ünitesi verilmiştir. data(a) ve data(b) 32 bitlik sayı girişleridir. Bu iki girişe verilen değerler toplanır ve 32 bitlik bir çıkış olan result çıkışına verilir. Taşma ve özel sayı değerlerinin oluşması durumları overflow, underflow, nan, zero, denormal, indefinite çıkışları kontrol edilerek takip edilebilir. Ayrıca ünitenin bir saat (clok) girişi bulunmakta olup isteğe bağlı olarak clock enable ve eşzamansız clear portu yerleştirilebilir. Yine isteğe bağlı olarak zero, denormal ve indefinite çıkışları kaldırılabilir.



Şekil 2.6 32 bitlik toplama ünitesi

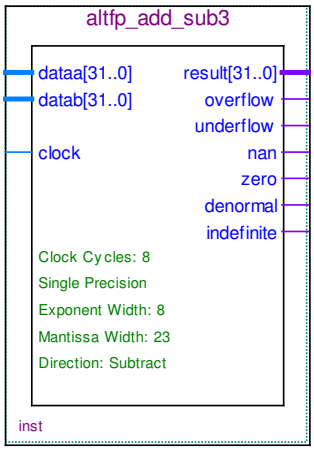
Şekil 2.7 'de toplama ünitesinin çalışma prensibi akış şeması olarak verilmiştir.



Şekil 2.7 Toplama ünitesinin çalışma prensibi

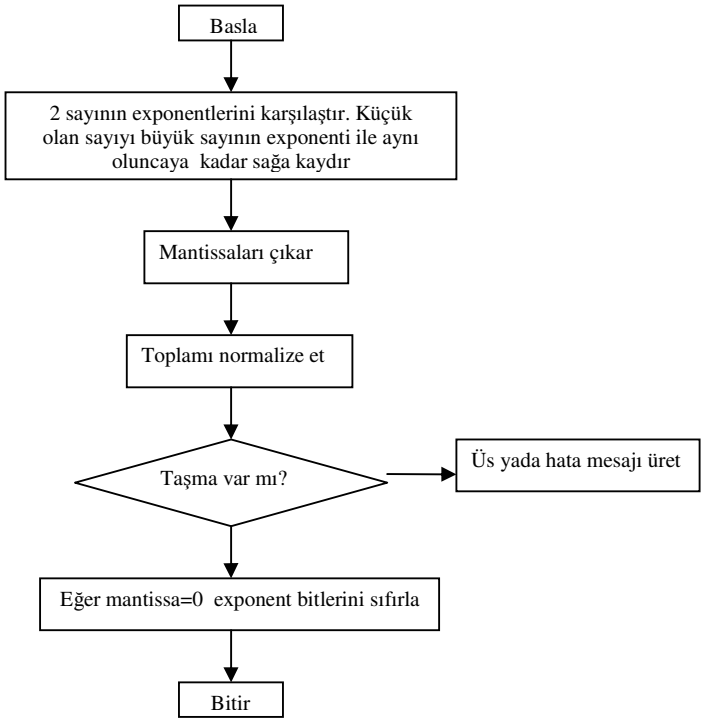
2.4.2. Çıkarma İşlemi

Şekil 2.8 'de 32 bitlik çıkarma ünitesi verilmiştir. data(a) ve data(b) 32 bitlik sayı girişleridir. Bu iki girişe verilen değerler çıkarılır ve 32 bitlik bir çıkış olan result çıkışına verilir. Taşma ve özel sayı değerlerinin oluşması durumları overflow, underflow, nan, zero, denormal, indefinite çıkışları kontrol edilerek takip edilebilir. Ayrıca ünitenin bir saat (clok) girişi bulunmakta olup isteğe bağlı olarak clock enable ve eşzamansız clear portu yerleştirilebilir.



Şekil 2.8 32 bitlik çıkarma ünitesi

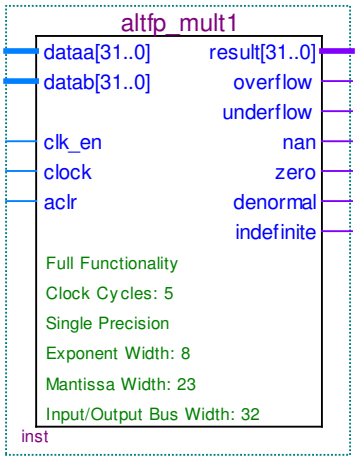
Şekil 2.9 ’da çıkarma ünitesinin çalışma prensibi akış şeması olarak verilmiştir.



Şekil 2.9 Çıkarma ünitesinin çalışma prensibi

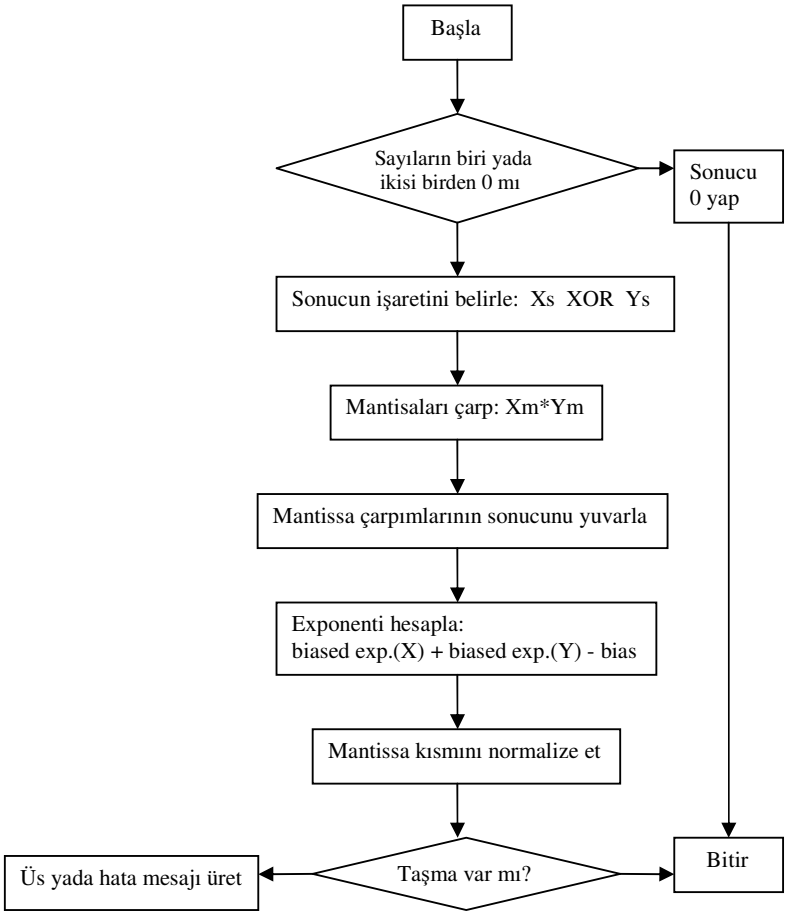
2.4.3. Çarpma İşlemi

Şekil 2.10 ’da 32 bitlik çarpma ünitesi verilmiştir. data(a) ve data(b) 32 bitlik sayı girişleridir. Bu iki girişe verilen değerler IEEE-754 kayan noktalı sayı formatına göre çarpılır ve sonuç 32 bitlik bir çıkış olan result çıkışına verilir. Taşma ve özel sayı değerlerinin oluşması durumları overflow, underflow, nan, zero, denormal, indefinite çıkışları kontrol edilerek takip edilebilir. Ayrıca ünitenin bir saat (clok) girişi bulunmakta olup isteğe bağlı olarak clock enable ve eşzamansız clear portu yerleştirilebilir. Yine isteğe bağlı olarak zero, denormal ve indefinite çıkışları kaldırılabilir.



Şekil 2.10 32 bitlik çarpma ünitesi

Şekil 2.11 'de çarpma ünitesinin çalışma prensibi akış şeması olarak verilmiştir.

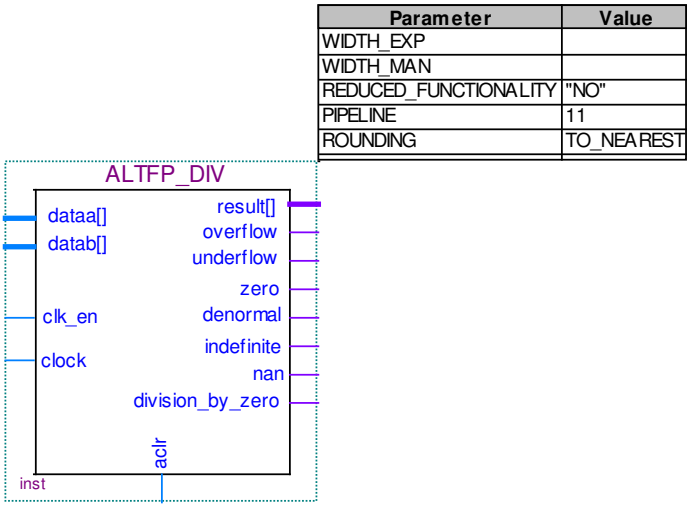


Şekil 2.11 Çarpma ünitesinin çalışma prensibi

2.4.4. Bölme İşlemi

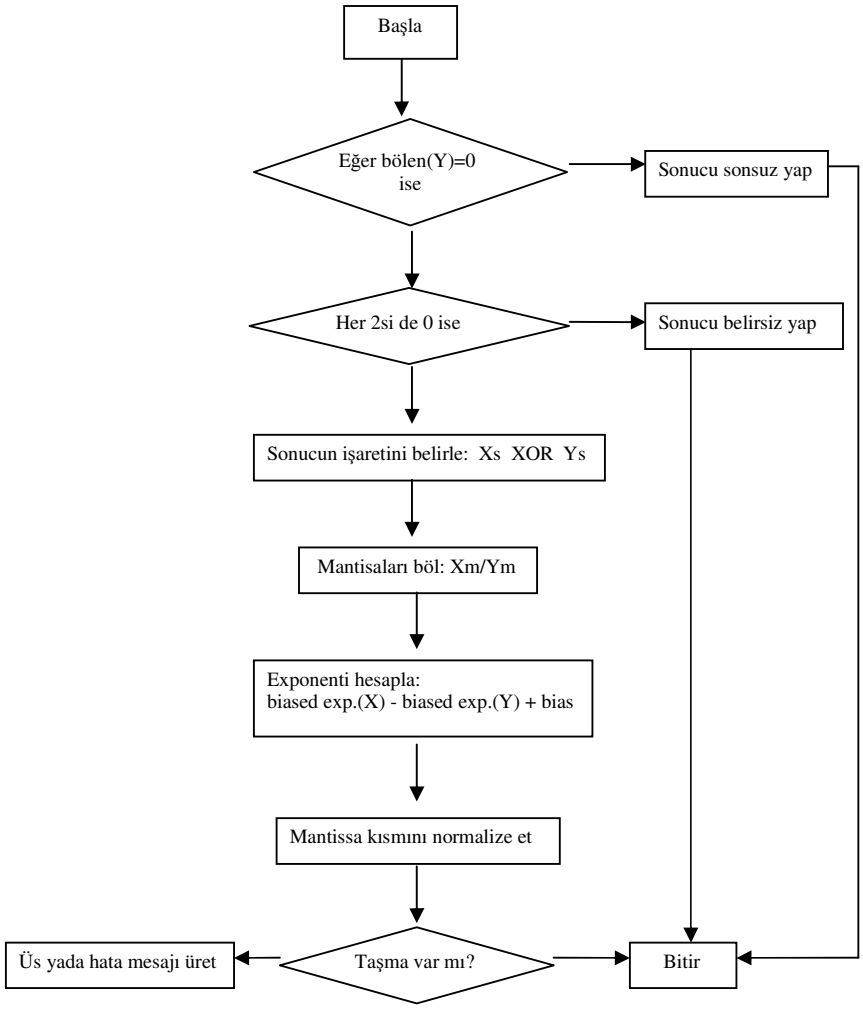
Şekil 2.12 ’de kayan noktalı sayıların bölme ünitesi verilmiştir. data(a) girişi bölünen ve data(b) girişi bölen sayı girişleridir. Bu iki girişe verilen değerler IEEE-754 kayan noktalı sayı formatına göre bölünür ve sonuç result çıkışına verilir. Taşma ve özel sayı değerlerinin oluşması durumları overflow, underflow, nan, zero, denormal, indefinite ve div is ion_by_zero çıkışları kontrol edilerek takip edilebilir. Ayrıca ünitenin bir saat (clok) girişi ve bir clock enable girişi bulunmaktadır.

Bölme ünitesinde ek olarak exponent ve mantissa alanlarının genişliğinin belirlenmesi, yuvarlama modunun kullanılması gibi işlemlerin seçimi için bir parametre-değer tablosu bulunmaktadır.



Şekil 2.12 Bölme ünitesi

Şekil 2.13 ’de bölme ünitesinin çalışma prensibi akış şeması olarak verilmiştir.



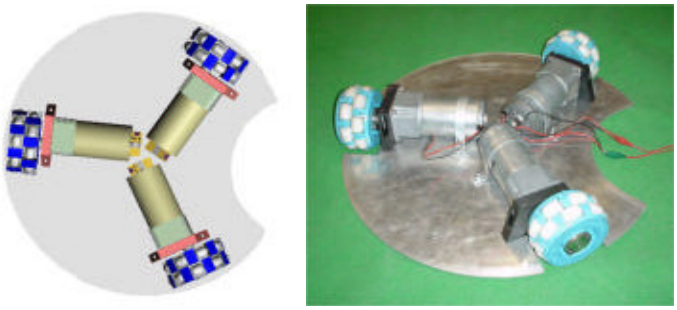
Şekil 2.13 Bölme ünitesinin çalışma prensibi

3. ÜÇ TEKERLEKLİ OMNI-DIRECTIONAL ROBOT VE MODELİ

Mobil robotlar için farklı şekillerde çalıştırılan ve yönetilen modeller mevcuttur. Kullanılacak bir robot için ilk olarak belirlenmesi gereken tekerlek sayısıdır. Çoğunlukla iki, üç ve dört tekerlek kullanılır. Her birinin birbirine göre avantaj ve dezavantajları vardır. İki tekerlekli robotun çok basit bir kontrolü vardır fakat manevra kabiliyeti düşüktür. Üç tekerlekli robotun idaresi ve kontrolü basittir fakat sınırlı çekiş gücüne sahiptir. Dört tekerlekli robotun mekaniği ve kontrolü daha komplekstir fakat daha yüksek çekiş gücüne sahiptir [10,11].

Literatürde omni-directional tekerlerin kullanıldığı robot uygulamalarına sıkça rastlanmaktadır. Yapılan bir çalışmada üç tekerlekli omni-directional bir robotun, optimum zamanda bir noktadan başka bir noktaya gidebilmesi için izlemesi gereken yörüngenin (trajectory) tespiti gerçekleştirilmiştir [12]. Bir başka çalışmada yine üç tekerlekli omni-directional bir robotta teker kaymaları dikkate alınarak robotun dinamik modeli oluşturulmuş ve simülasyonları gerçekleştirilmiştir. [13]. Mobil robotta Caster tip omni-directional tekerleklerin kullanıldığı bir başka çalışmada, robot modeli oluşturulup bir kontrol sistemi içerisinde kullanılmıştır [14].

Omni-directional tekerlerin kullanıldığı üç tekerli robot yüksek manevra kabiliyetine sahip ve kontrolü kolaydır. Omni-directional tekerlerde, tekerlerin her yönde serbestçe hareket etmesine izin veren ufak tekerlekler bulunmaktadır. Bunlar dış tekerle birlikte hareket eder. Bu yeni robot modelinde motorlar direk mekaniği basitleştiren omni tekerlere bağlanır. Üçüncü teker sayesinde çekiş gücündeki düşüş kısmen azaltılmıştır. Robotun mekanik yapısı Şekil 3.1'de gösterilmiştir.



Şekil 3.1 Üç tekerlekli omni-directional robotun mekanik yapısı

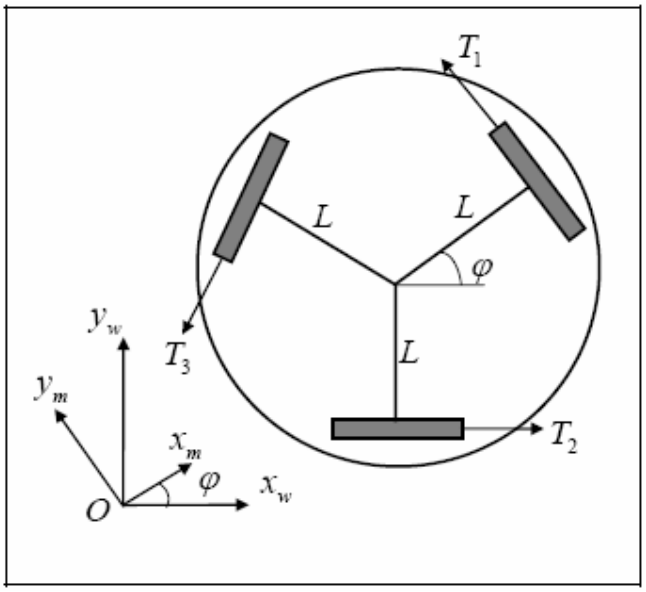
Soldaki şekilde gri daire robot platformunu göstermektedir. Şekil 3.1 'de görüleceği üzere her bir tekere bağlı birer motor bulunmaktadır. Omni tekerlerle birleştirilmiş olan üç motor, aralarında 120 derece açı ile eksenleri robot merkezinde kesişecek şekilde platform

üzerine monte edilmiştir. Bu durumda motor ve tekerlek aynı dönel merkeze sahiptir. Merkezde her motor eksenine kodlayıcı (encoder) bağlandığı görünmektedir [10].

Robot üzerinde DC motor kullanılmakta olup, motorla birlikte kullanılan redüktör vasıtası ile motorun devrinin düşürülüp daha yüksek moment vermesi sağlanmıştır. Kullanılan motorun matematiksel modeli detaylı olarak EK-2 'de verilmiştir.

3.1. Robot Platformunun Kinematiği ve Ters Kinematiği

Robotu temsil eden koordinatların platformun merkezine yerleştirildiği düşünülmektedir. Şekil 3.2 'de robotun koordinat sistemi gösterilmektedir [16,17].



Şekil 3.2 Robot için koordinat sistemleri

Şekil 3.2 'de görüldüğü üzere $x_w o y_w$ global koordinatları (global eksen), $x_m o y_m$ ise hareketli çerçeve koordinatlarını (lokal eksen) ifade etmektedir. Lokal eksen koordinatları, global eksen koordinatları ile aynı orijin noktasına sahiptir fakat lokal eksen koordinatları robot ile birlikte dönmektedir. x_m ekseni, T_1 çekme kuvvetine göre dik doğrultuda yer almaktadır ve φ lokal eksenle global eksen arasındaki açıyı temsil etmektedir.

Hareketli çerçevedeki 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 3.2 'de de gösterilmekte olan L, robot merkezi ile bir tekerlek arasındaki mesafedir.

${}^m F = [F_{mx} \quad F_{my} \quad T_Z]^T$ hareketli çerçevedeki kuvvetleri belirtmektedir ve ${}^t F = [T_1 \quad T_2 \quad T_3]^T$ tekerleklerin itme kuvvetlerini göstermektedir. Katsayı matrisi olan B matrisi şu şekilde ifade edilir:

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

Yukarıdaki ilişkilerden şu eşitlik yazılabilir:

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

Enerjinin korunumu prensibinden faydalanılarak aşağıdaki eşitlik elde edilebilir:

$${}^W F^T \bullet {}^W \dot{X} = {}^m F^T \bullet {}^m \dot{X} = {}^t F^T \bullet {}^t \dot{X} = {}^t F^T \bullet r\dot{q} \tag{3.5}$$

burada ${}^W F = [F_x \quad F_y \quad T_Z]^T$ global koordinat sistemindeki kuvvetlerdir, r tekerleklerin yarıçapı ve $\dot{q}$ tekerleklerin açısal hızlarıdır.

Şimdi robotun merkezine iliştirilmiş hareketli çerçevenin hızı şu şekilde yazılabilir:

$${}^m \dot{X} = (B^T)^{-1} r\dot{q} \tag{3.6}$$

Bu hızın global koordinat sistemindeki karşılığı bilinmek zorunda olduğu için, bir dönme matrisi kullanılmak zorundadır. Bu dönme matrisi koordinatların, hareketli robot çerçevesinden sabit global çerçeveye dönüştürülmesine olanak sağlamaktadır. Dönme matrisi şu şekilde verilir.

$${}^W_m R = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Burada ϕ Şekil 3.2 'de gösterilmekte olan lokal eksenle global eksen arasındaki açıyı temsil etmektedir.

Yukarıdaki dönme matrisi sayesinde hareketli çerçevenin hız ve ivmesi global çerçeveye göre düzenlenebilir. Eşitlikler aşağıda verilmektedir.

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

$${}^W \ddot{X} = {}^w_m \dot{R} \, (B^T)^{-1} r\dot{q} + {}^w_m \dot{R} \, (B^T)^{-1} r\ddot{q} \tag{3.8}$$

Eşitlik 3.7 robotun güncel pozisyonunu bulmak için kullanılacak olan ileri kinematik eşitliğidir. Bu eşitlik kodlayıcılar tarafından hesaplanmış olan güncel açısal hızları alacak ve onları robotun güncel hızlarına dönüştürecektir. Bu hızlar daha sonra robotun güncel pozisyonunu bulmak için kullanılabilir. Yani bir global 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 eşitlik aşağıdaki gibidir.

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

3. 2. Robotun Dinamik Modeli

Robotun hem bilgisayar hem de hardware simülasyonları çalışırken uygun şekilde temsil edilebilmesi için sistemin dinamik modelinin çıkartılmasına ihtiyaç duyulur. Robot için dinamik model çıkarma İkinci Newton Kuralı kullanılarak başlar [2].

$${}^{w}F = M\ {}^{w}\ddot{X} = {}^{w}_{m}R\ B\ {}^{t}F \tag{3.10}$$

Burada ${}^{w}\ddot{X}$, robotun X ve Y yönündeki ivmelerini ve ayrıca robotun kendi merkezi etrafında yaptığı dönme hareketinin ivmesini ihtiva eder. Bağıntıda verilen tüm hareket ve koordinatlar global çerçeveye aittir. $\mathbf{M}$ robotun ağırlık matrisidir [16-17].

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

Ağırlık matrisindeki m kütle, J ise robotun dönme ataletidir.

$$({}^{w}_{m}R)^{-1} = {}^{w}_{m}R^{T}$$

Eşitlik 3.8 , 3.10 ve yukarıda verilmiş olan dönme matrisinin ortogonal özelliği kullanılarak robotun çekme kuvveti şu şekilde elde edilir:

$${}^{t}F = (B)^{-1}\, {}^{w}_{m}R^{T}\, M \left[{}^{w}_{m}\dot{R}\, (B^{T})^{-1} r\dot{q}_{L} + {}^{w}_{m}R\, (B^{T})^{-1} r\ddot{q}_{L} \right] \tag{3.11}$$

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

$$\tau_{L} = J_{L}\ddot{q}_{L} + c_{L}\dot{q}_{L} + {}^{t}F\ r \tag{3.12}$$

$$\tau_{m} = J_{m}\ddot{q}_{m} + c_{m}\dot{q}_{m} + (1/n)\tau_{L} \tag{3.13}$$

$$\dot{q}_{L} = (1/n)\dot{q}_{m} \tag{3.14}$$

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

Eşitlik 3.11 ve 3.12 yerine konulduktan sonra Eşitlik 3.13 matris formuna çevrilerek robotun tüm simülasyonlarında kullanılan nihai moment (tork) eşitliği elde edilmiş olur [16].

$$\tau_M = k_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \ddot{q}_m + k_2 \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \ddot{q}_m + k_3 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot{q}_m + k_4 \dot{\phi} \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \dot{q}_m \tag{3.15}$$

Burada,

$$k_1 = J_m + \frac{J_L}{n^2} + \frac{(4m\,L^2 + J)\,r^2}{9L^2 n^2} \tag{3.16}$$

$$k_2 = \frac{(-2m\,L^2 + J)\,r^2}{9L^2 n^2} \tag{3.17.}$$

$$k_3 = c_m + \frac{c_L}{n^2} \tag{3.18}$$

$$k_4 = \frac{2\sqrt{3}\,mr^2}{9n^2} \tag{3.19}$$

$$\dot{\phi} = \frac{r(\dot{q}_1 + \dot{q}_2 + \dot{q}_3)}{3L} = \frac{r(\dot{q}_{m1} + \dot{q}_{m2} + \dot{q}_{m3})}{3nL} \tag{3.20}$$

Bir motor için aşağıdaki denklem verilebilir [17].

$$\tau_m = (E - k_E \dot{q}_m) k_{lr} k_M \tag{3.21}$$

Burada *E* motor giriş gerilimi, k_E motorun zıt *EMF* sabiti ve k_M motor moment sabitidir. k_{lr} , *1/R* ifadesine eşittir ki burada *R* motorun terminal direncidir. Motorun indüktansı, küçük olduğu için ve genellikle robot dinamiklerinde önemsenmediği için ihmal edilmiştir. Motorun matematiksel modeli detaylı olarak EK-2 'de verilmiştir.

Eşitlik 3.21 ile Eşitlik 3.15 birleştirilerek üç tekerlekli omni-directional robotun birleşmiş doğrusal olmayan hareket eşitliği (EOM) nin nihai ifadesi olan Eşitlik 3.22 elde edilir.

$$E = \frac{k_1}{k_{lr}k_M}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\ddot{q}_m + \frac{k_2}{k_{lr}k_M}\begin{bmatrix}0&1&1\\1&0&1\\1&1&0\end{bmatrix}\ddot{q}_m$$
$$+\left(\frac{k_3}{k_{lr}k_M}+k_E\right)\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\dot{q}_m + \frac{k_4\dot{\phi}}{k_{lr}k_M}\begin{bmatrix}0&1&-1\\-1&0&1\\1&-1&0\end{bmatrix}\dot{q}_m \qquad (3.22)$$

Doğrusal olmayan terimin etkisi robotun dönme hızı $\dot{\phi}$ ve $k_4/k_{lr}k_M$ katsayısı ile orantılıdır. Robot kendi üzerinde dönmeden hareket ettiği zaman Eşitlik 3.22 'deki doğrusal olmayan terim (dördüncü terim) sıfır olacaktır.

Tablo 3.1 Robotun parametreleri

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

Tablo3.1'de verilen robot ve motora ait parametre değerlerinin çoğu deneysel olarak elde edilmektedir. Bu tez çalışmasında, daha önce aynı yapıda benzer özellik gösteren bir robotun kullanıldığı bir tez çalışmasındaki deneysel olarak ölçülmüş olan robota ait parametre değerleri kullanılmıştır [17].

Tablo 3.1 'deki veriler Eşitlik 3.16, 3.17, 3.18 ve 3.19 'da yerine konularak Tablo 3.2'deki terimler elde edilir.

Tablo 3.2 Robot için EOM katsayıları

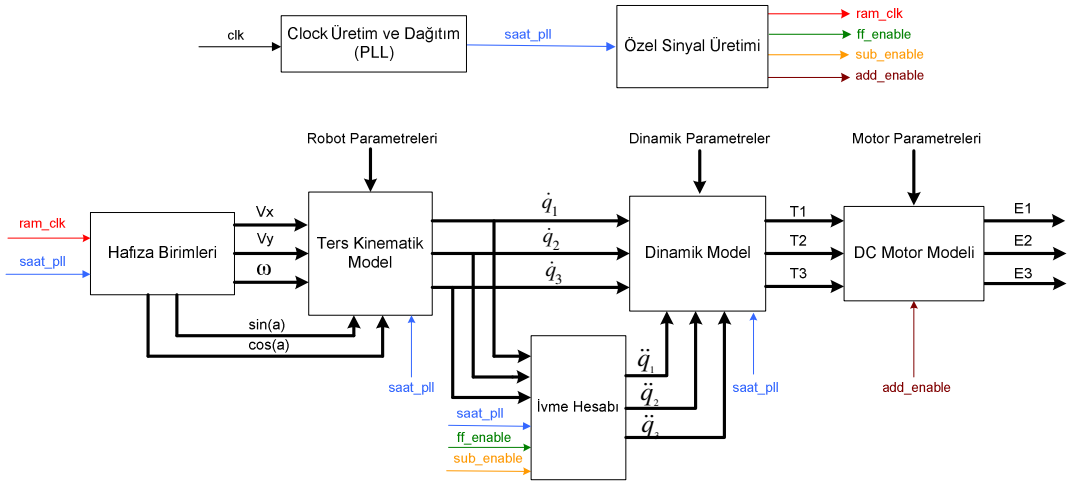
k_1	k_2	k_3	k_4
2.82e-6	-8.572e-7	5.02e-8	7.86e-7
$k_1/(k_{lr}k_M)$	$k_2/(k_{lr}k_M)$	$k_3/(k_{lr}k_M)+k_E$	$k_4/(k_{lr}k_M)$
1.6e-3	-4.81e-4	1.45e-2	4.41e-4

4. ROBOT MODELİNİN FPGA TASARIMI

Bölüm 3'te tanıtılan robotun ters kinematik ve dinamik modelleri Altera firmasının üretmiş olduğu Quartus II 8.1. FPGA programlama arayüzü kullanılarak yine aynı firmanın üretmiş olduğu Stratix II GX model FPGA üzerine tasarlanmış ve simüle edilmiştir. Kullanılmakta olan FPGA ve geliştirme kartına ilişkin teknik özellikler detaylı olarak EK-1 'de sunulmuştur.

Sistemi genel olarak ters kinematik ve dinamik model olmak üzere ikiye ayırmak mümkündür. Ters kinematik modelinde alt sisteme robotun X eksenindeki hızı, Y eksenindeki hızı ve kendi ekseni üzerinde açısal dönme hızı giriş olarak verilmekte ve çıkış olarak her bir tekerleğin bağlı olduğu motorun açısal hızı hesaplanır. Dinamik modelde ise motorların açısal dönme hızları ve ivmeleri alt sisteme giriş olarak uygulanır ve çıkışta bu hız ve ivmelere ulaşabilmek için sırasıyla motorlara uygulanması gereken moment ve gerilim değerleri hesap edilir.

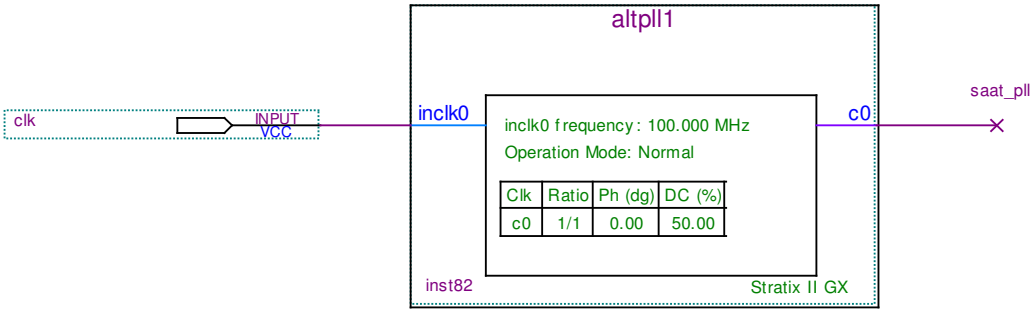
Şekil 4.1 'de robot modeli için gerçekleştirilen FPGA tasarımının blok diyagramı yer almaktadır.



Şekil 4.1 FPGA tasarımının blok diyagramı

4.1. Clock Üretim ve Dağıtım Alt Devresi

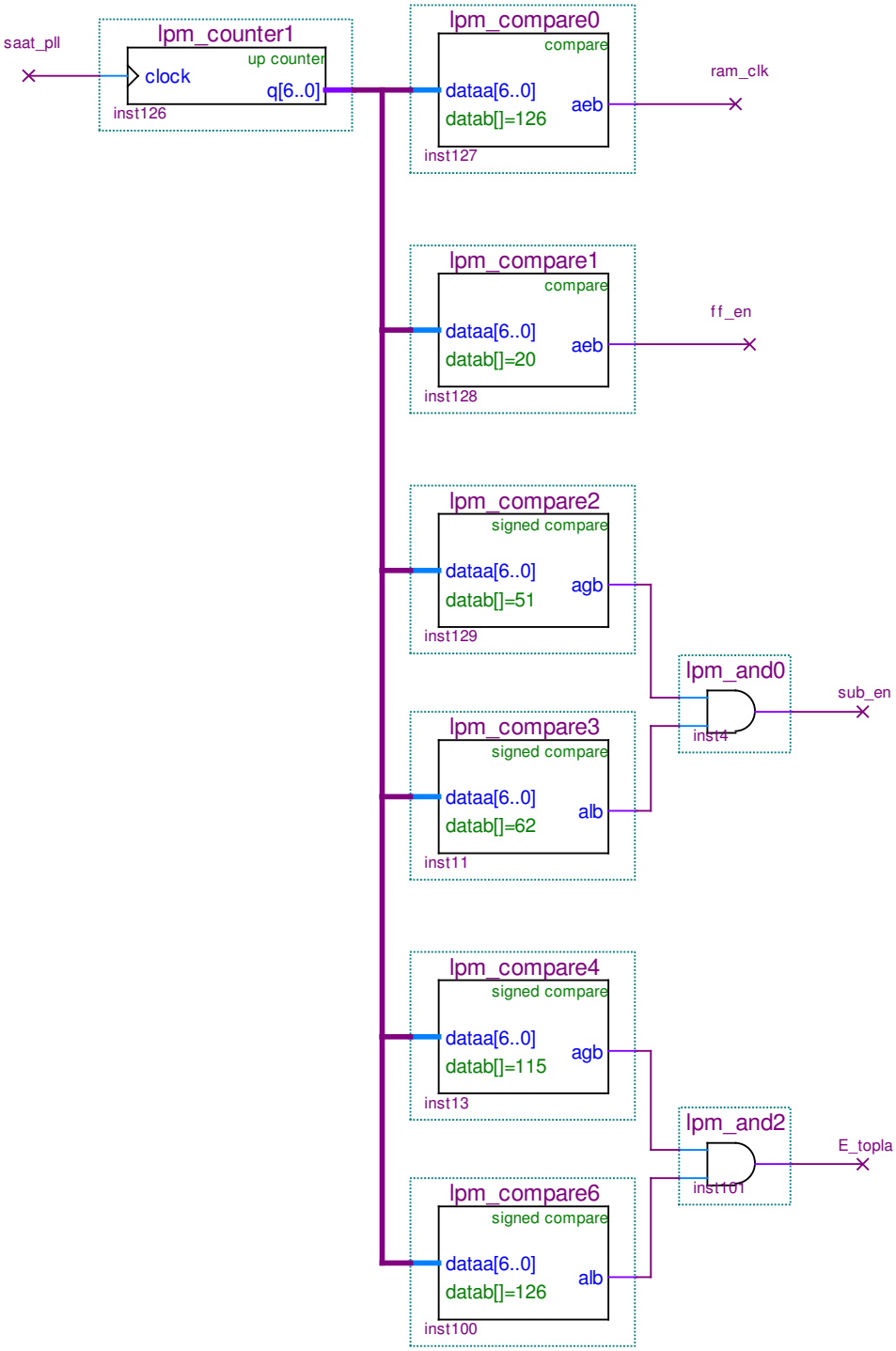
Şekil 4.2 'de gösterilen alt devre, bir FPGA bloğu olan altpll1 vasıtasıyla girişine uygulan clock işaretinden FPGA'ın her noktasında eş zamanlı olarak kullanılabilecek bir clock işareti üretir ve bu işareti yine herhangi bir kayma ya da gecikme olmaksızın her noktaya dağıtmayı sağlar [8]. Tasarımda 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 clock kaynağı

4.2. Özel Sinyal Oluşturma Alt Devresi

Projenin bazı toplama, çıkarma, flip-flop ve counter bloklarında farklı frekans ve görev periyotlu işaretlere ihtiyaç duyulmaktadır. Bunları üretmek için Şekil 4.3 'de gösterilmekte olan alt devre tasarlanmıştır. Sistemin clock işaretinden sayıcı ve karşılaştırıcılar kullanılarak istenen işaretler elde edilir. Üretilen çıkış işaretleri diğer alt devrelerin clock ve enable girişlerine uygulanmaktadır.

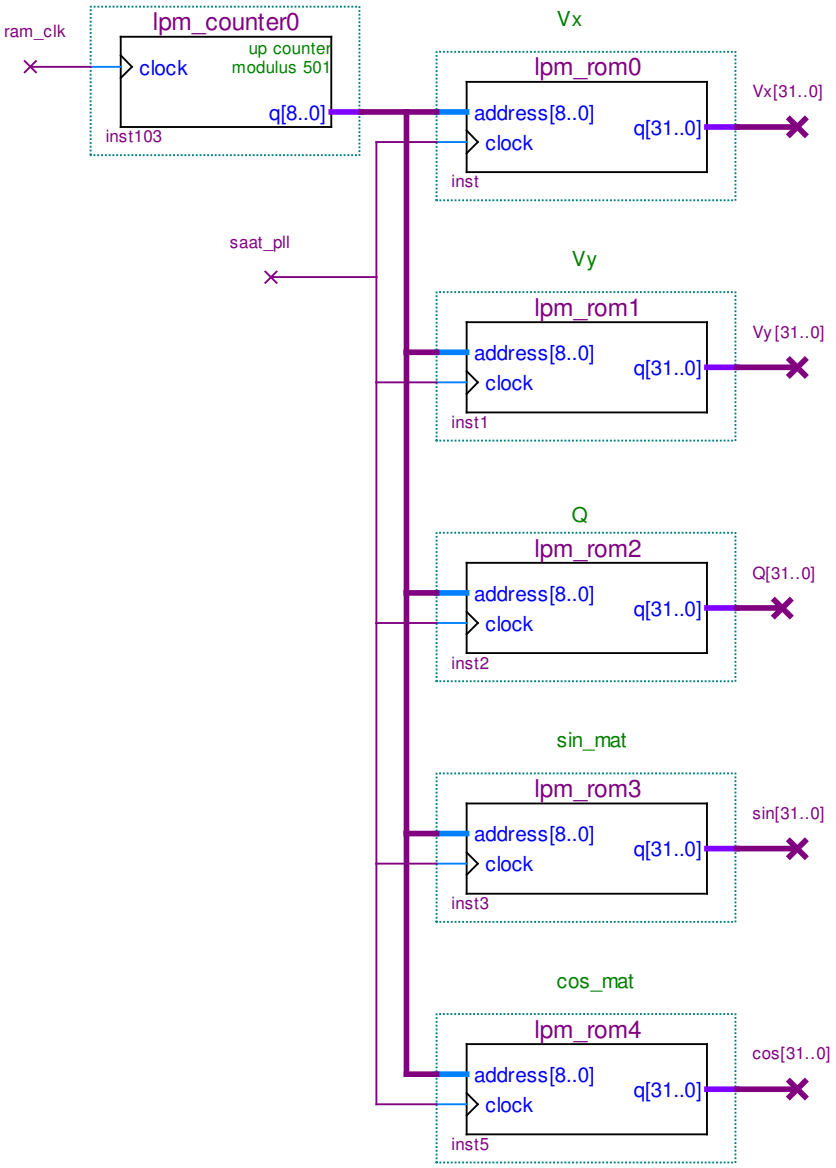


Şekil 4.3 Özel sinyal oluşturma alt devresi

4.3. Hafıza Yönetimi

Sistemin girişleri olan robotun x ekseni, y ekseni ve kendi ekseni etrafındaki dönme hızları, sistem çalıştırılmaya başlamadan önce ROM hafıza birimlerine kaydedilir. Ayrıca matematiksel denklemlerin çözümünde kullanılacak olan sinüs ve cosinüs tabloları da ROM hafıza birimlerinde tutulur. ROM'da bulunan veriler belirli zaman aralıklarıyla veri yoluna aktarılır.

Şekil 4.4 'de gösterilen hafıza yönetim alt birimi, sistemin bir çevirim süresi olan 128 sistem clock cycle 'ında bir ROM bölgesinin bir sonraki adresinde yer alan veriyi veri yoluna koymaktadır.



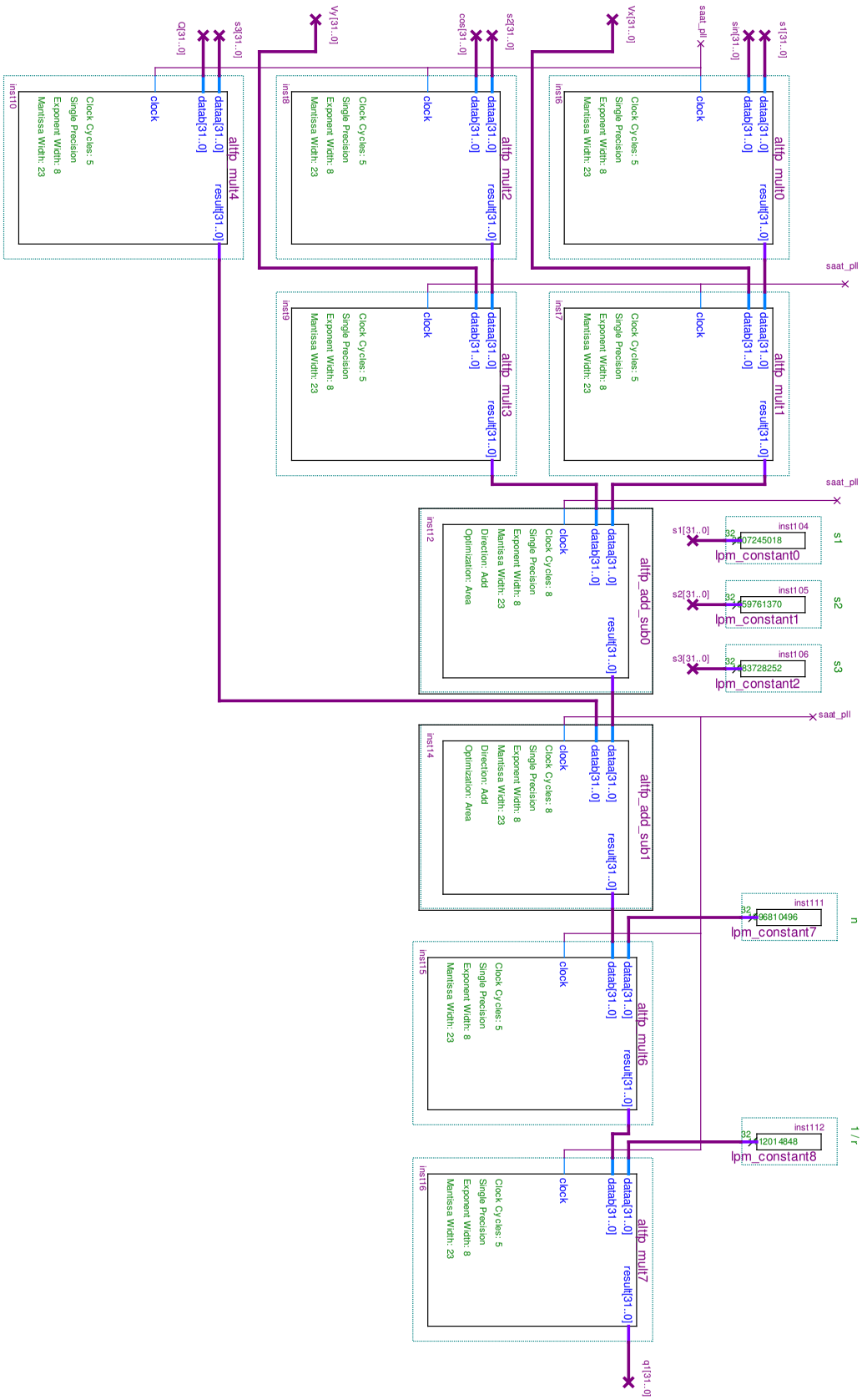
Şekil 4.4 Hafıza yönetim alt devresi

4.4. 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 dönme hızlarını hesaplamaktadır. Yani Bölüm 3.2 'de ayrıntıları verilmiş olan Eşitlik 3.9 'daki denklem gerçekleştirilmektedir.

Şekil 4.5 'de örnek olarak birinci motorun açısal dönme hızı olan q1'in elde edilişi gösterilmiştir. Eşitlik 3.9 'un açılımı sonucu q1 = (-sin(a)*s1*Vx + cos(a)*s2*Vy + s3*Q) * n/r şeklinde elde edilir. Burada Vx, robotun x eksenindeki hızını; Vy, robotun y eksenindeki hızını; Q, robotun kendi ekseni etrafındaki dönme hızını; a, robotun lokal ekseniyle global eksen arasındaki açı farkını; n, motorun dişli oranını ve r ise tekerleklerin yarıçapını temsil etmektedir. s1, s2 ve s3 değerleri ise sabit değerlerdir.

İkinci motorun ve üçüncü motorun açısal dönme hızları olan q2 ve q3 de benzer şekilde tasarlanmıştır.



Şekil 4.5 Birinci motora ait açısal dönme hız hesabı

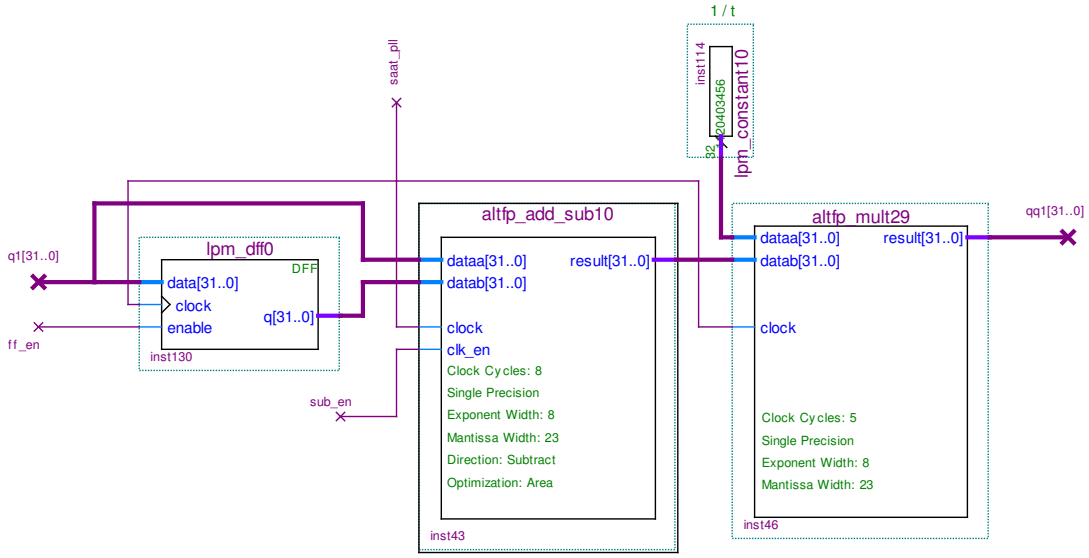
4.5. İvme Hesabı

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

qq1 = (q1(h) - q1(h-1)) / t Burada qq1 birinci motora ait ivmedir ve q1 ise açısal dönme hızıdır. t ise örnekleme süresidir.

Şekil 4.6 'da birinci motorun ivme hesabı için gerçekleştirilen FPGA tasarımı gösterilmiştir.

İkinci motorun ve üçüncü motorun ivmeleri olan qq2 ve qq3 de benzer şekilde tasarlanmıştır.

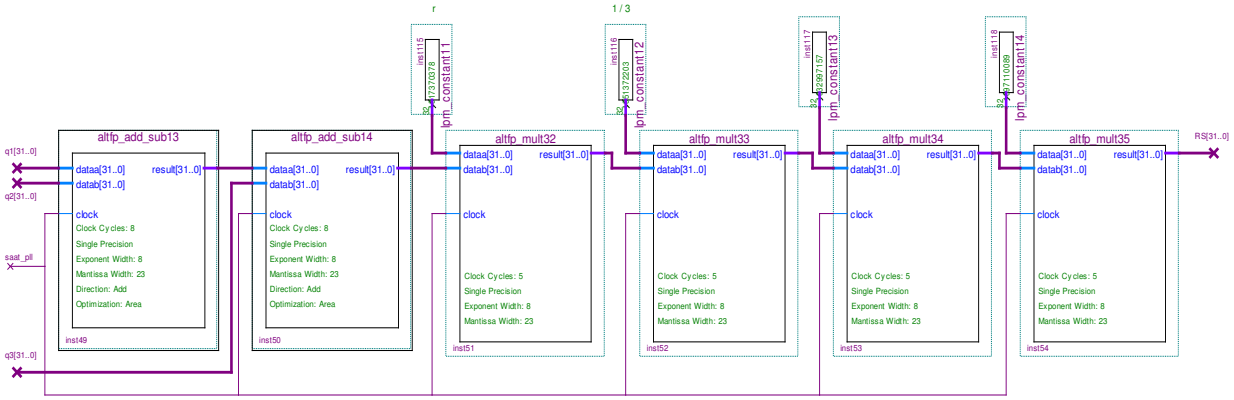


Şekil 4.6 Birinci motora ait ivme hesabı

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 gerekli formüller ve açıklamalar ayrıntılı olarak Bölüm 3.3 'te verilmiştir.

Şekil 4.7 , Eşitlik 3.20 'nin FPGA tasarımındaki karşılığıdır.

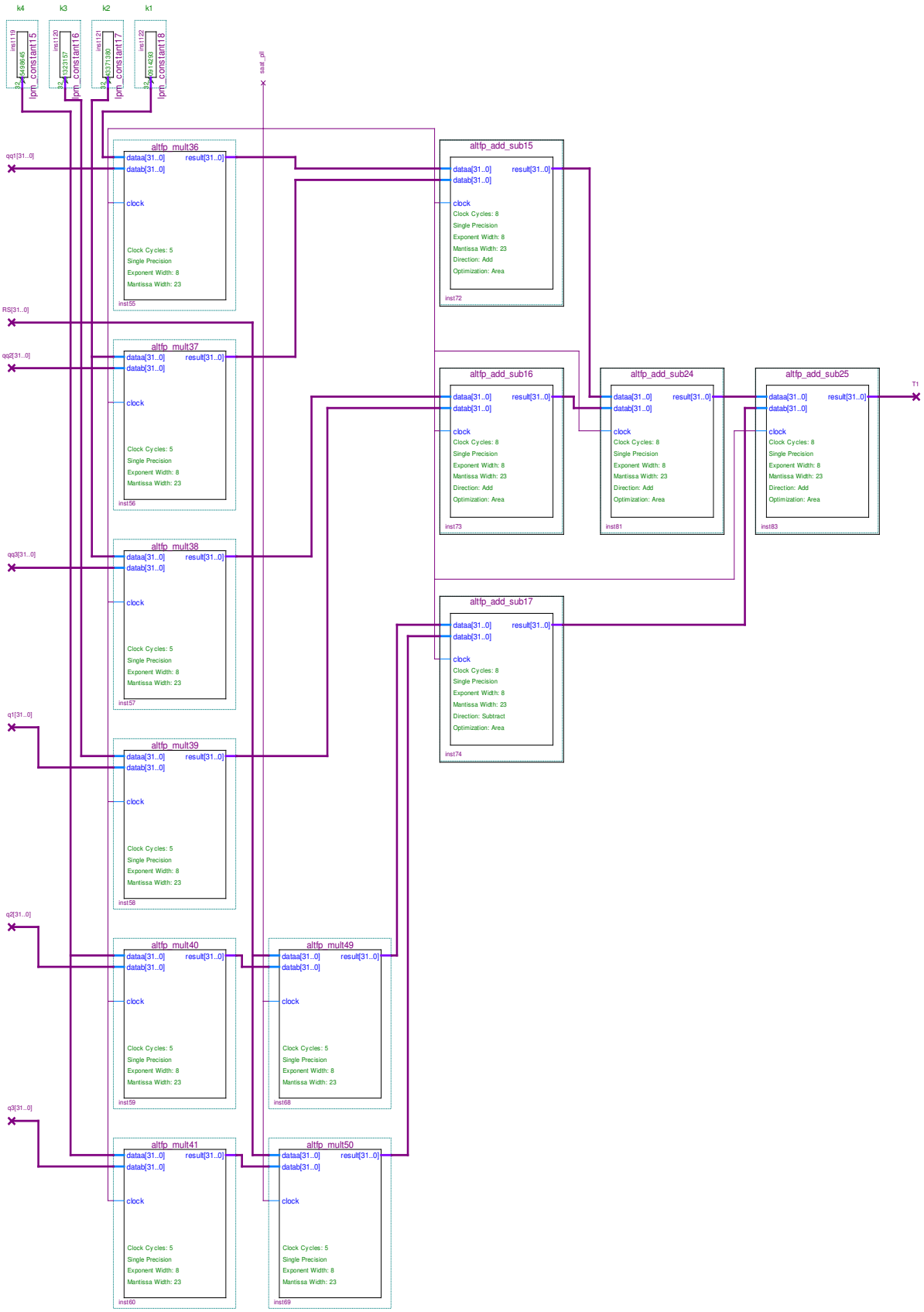


Şekil 4.7 Moment için RS (dönme hızı) hesabı

Eşitlik 3.15 'in açılımı gerçekleştirilince birinci motora ait moment formülü T1 = k1*qq1 + k2*qq2 + k2*qq3 + k3*q1 + k4*RS*q2 – k4*RS*q3 şeklinde elde edilir. Burada kullanılan k değerleri sabitleri ve Bölüm 3.3 de verilmiştir.

Şekil 4.8 'de örnek olarak birinci motora ait moment hesabı için gerçekleştirilen FPGA tasarımı gösterilmektedir.

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

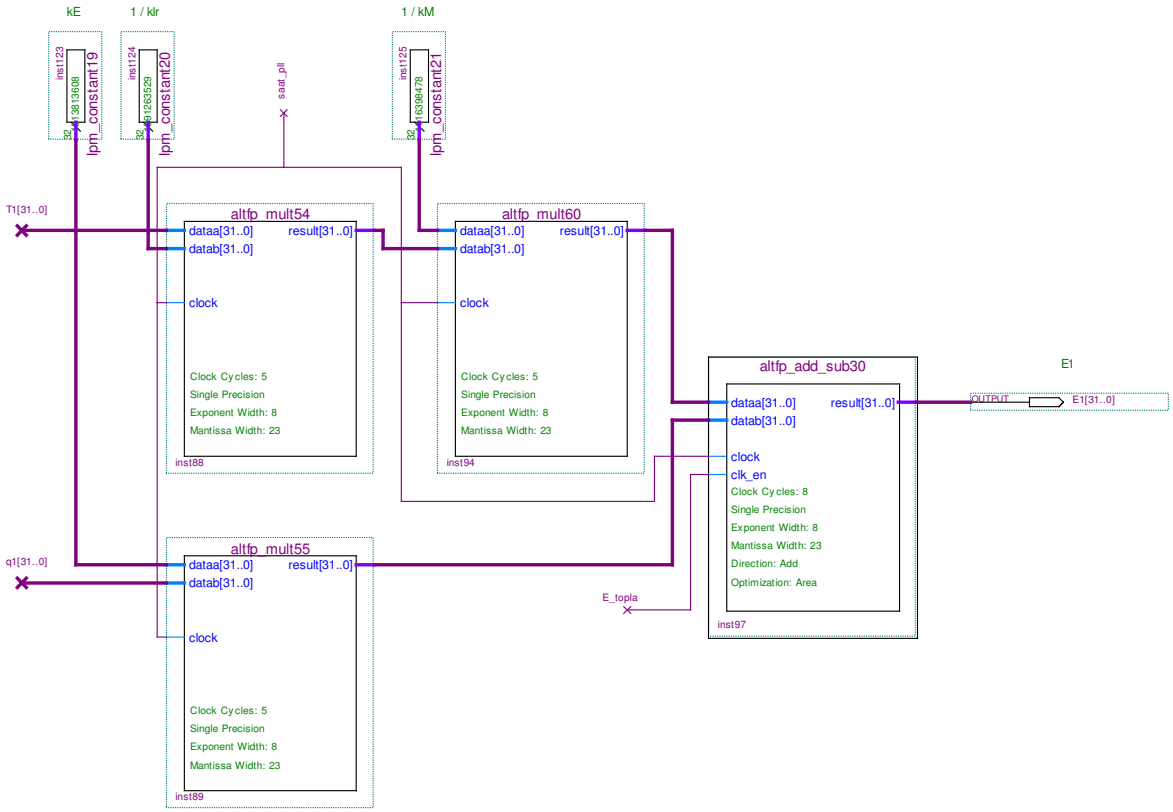


Şekil 4.8 Birinci motora ait moment hesabı

4.7. Gerilim Hesabı

Şekil 4.9 ‘da örnek olarak birinci motora ait gerilim değerinin hesabı için gerçekleştirilen FPGA tasarımı gösterilmektedir. Eşitlik 3.21 ‘de gerekli düzenlemeler yapılırsa birinci motora gerilim değeri olan E1 = 1/(klr*kM)*T1 + kE*q1 şeklinde elde edilir.

İkinci motorun ve üçüncü motorun gerilimleri olan E2 ve E3 de benzer şekilde tasarlanmıştır.



Şekil 4.9 Birinci motora ait gerilim hesabı

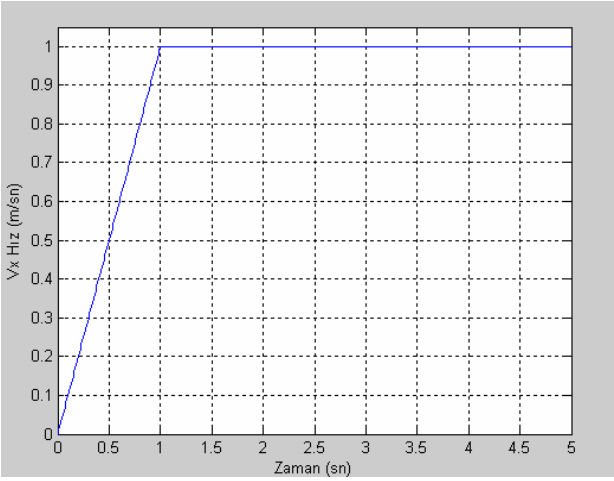
5. SİMÜLASYON SONUÇLARI

Bu bölümde robot modelinin farklı birkaç robot hareketi için matlab ve FPGA ortamında gerçekleştirilen simülasyon sonuçları verilmiştir. Robotun belirlenmiş doğrusal ve dairesel yollar için gerçekleştireceği hareketin matlab ve FPGA ortamlarında modellemesi yapılmıştır. Robotun ters kinematik modeliyle tekerlerin ve buna bağlı olarak motorların açısal dönme hızları hesaplanmıştır. Robotun dinamik modeliyle de hesaplanmış olan motor açısal dönme hızlarına göre sırasıyla motorların momentleri (tork) ve motorlara uygulanması gereken gerilimler hesaplanmıştır. Matlab ve FPGA sonuçları karşılaştırıldığında sonuçların birbiriyle örtüştüğü görülmüştür.

Belirtilen farklı robot hareketleri için robot modelinin simülasyon sonuçları aşağıda sunulmuştur.

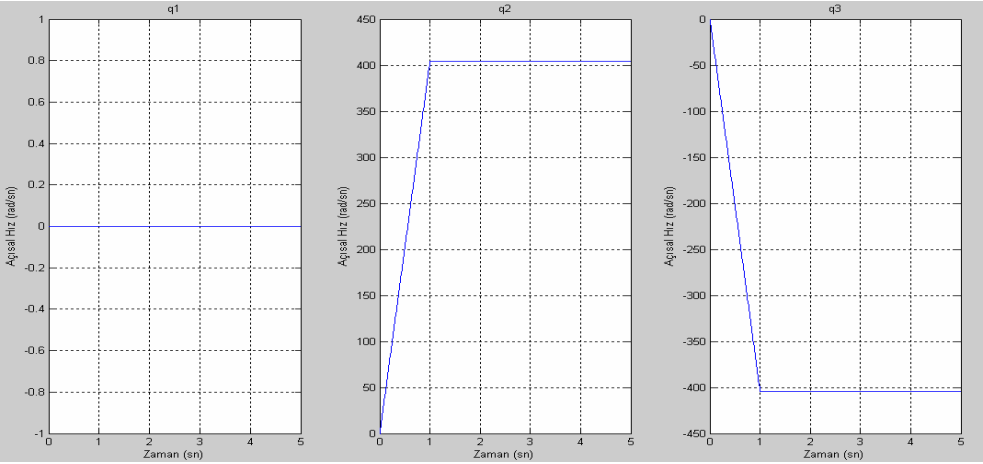
Uygulama 1 : Robotun Kendi Ekseni Etrafında Dönme Hareketi Yapmaksızın Doğrusal Bir Yol İzlemesi

X ekseni doğrultusunda doğrusal bir yol izlerken hız - zaman grafiği Şekil 5.1 'de gösterildiği gibi olan robotun her bir tekerleğe bağlı motorunun açısal hız – zaman, moment(tork) – zaman ve gerilim – zaman grafikleri modelin matlab ve FPGA simülasyonları ile elde edilmiştir. Robotun başlangıç anında global eksenle arasında açı farkı bulunmayacak şekilde konumlandığı varsayılmıştır.

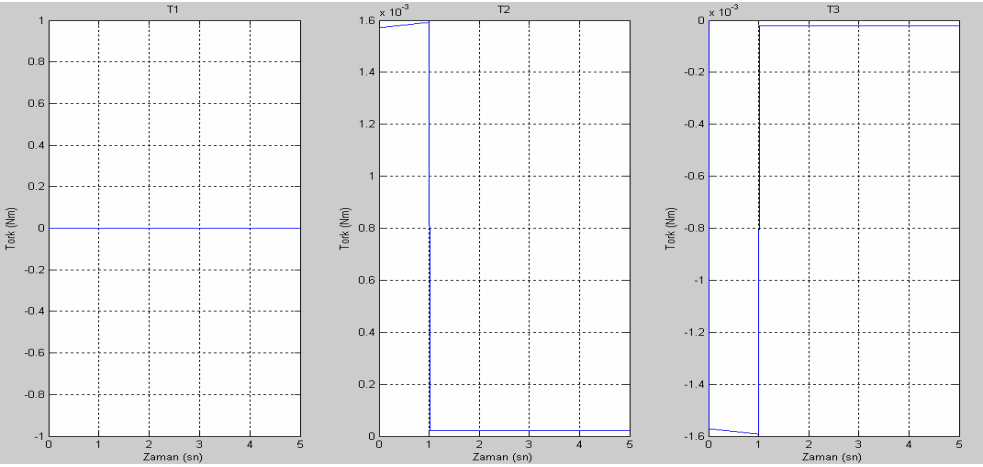


Şekil 5.1 Doğrusal hareket yapan robotun hız-zaman grafiği

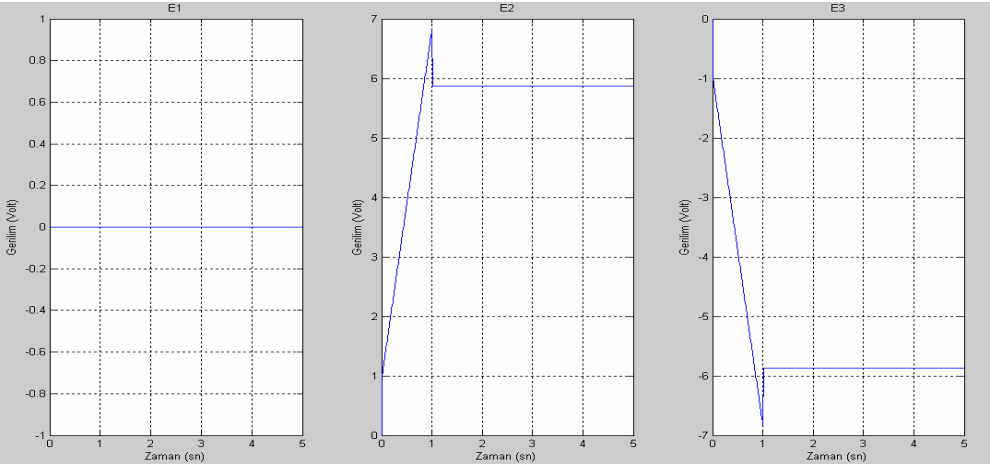
Uygulama 1 için matlab simülasyon sonuçları Şekil 5.2, Şekil 5.3 ve Şekil 5.4'de verilmiştir.



Şekil 5.2 Uygulama 1'in matlab simülasyonu sonucunda elde edilen motorlara ait açısal hız - zaman grafikleri



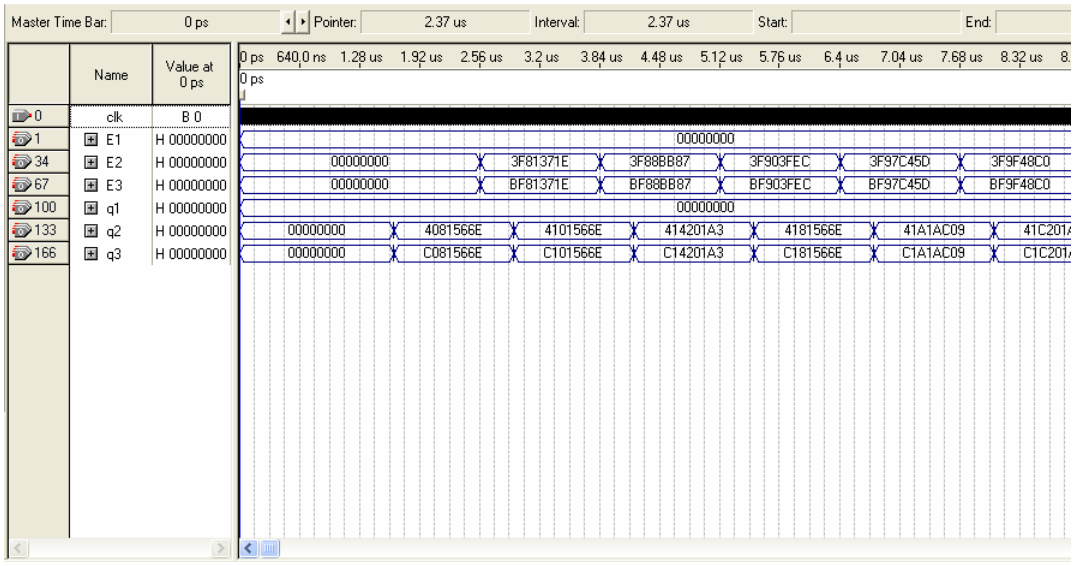
Şekil 5.3 Uygulama 1'in matlab simülasyonu sonucunda elde edilen motorlara ait moment - zaman grafikleri



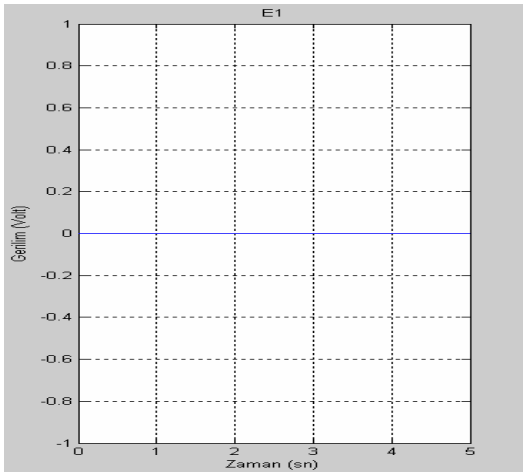
Şekil 5.4 Uygulama 1'in matlab simülasyonu sonucunda elde edilen motorlara ait gerilim - zaman grafikleri

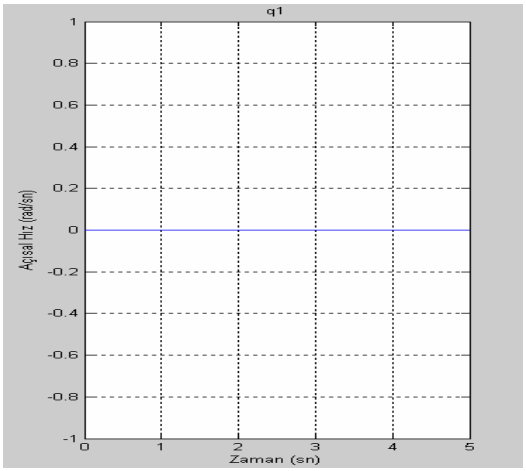
Robotun lokal ekseniyle global eksen arasında açı farkı bulunmadığı için robotun 1 nolu tekerleği x eksenine dik olarak konumlanmış olur. Robot hareket süresince kendi ekseni etrafında dönme hareketi yapmadığı için bu konumlanma tüm hareket boyunca değişmez. x ekseni doğrultusunda gerçekleştirilecek bu doğrusal hareket için 1 nolu tekerlek dönmez. Bu durumda 1 nolu tekerlek dönmediği için açısal dönme hızının sıfır olması ve açısal dönme hızına bağlı olarak moment ve gerilim değerlerinin de sıfır olması beklenmektedir. Simülasyon sonuçlarından Şekil 5.2, Şekil 5.3 ve Şekil 5.4 'de 1 nolu tekerleğe ait için açısal dönme hızı, moment ve gerilim değerlerinin sıfır olduğu görünmektedir.

Uygulama 1'in Quartus 8.1 FPGA tasarım geliştirme arayüzünde gerçekleştirilen simülasyon sonucu Şekil 5.5 'de gösterilmiştir. Burada clk, sistemin girişine uygulanan 100 MHz ve % 50 görev periyotlu clock işaretidir. q1, q2 ve q3 sırasıyla her bir motorun simülasyon sonunda hesaplanan açısal dönme hızlarıdır. E1, E2 ve E3 ise sırasıyla her bir motorun simülasyon sonunda hesaplanan gerilim değerleridir. Simülasyon sonucunda oluşan datalar Quartus ortamından alınarak Şekil 5.6 , Şekil 5.7 ve Şekil 5.8 'da görüldüğü üzere matlab ortamında çizdirilmiştir.



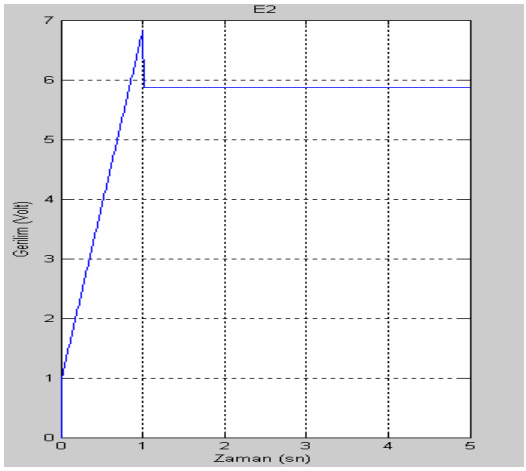
Şekil 5.5 Uygulama 1 'e ait FPGA simülasyon sonucu

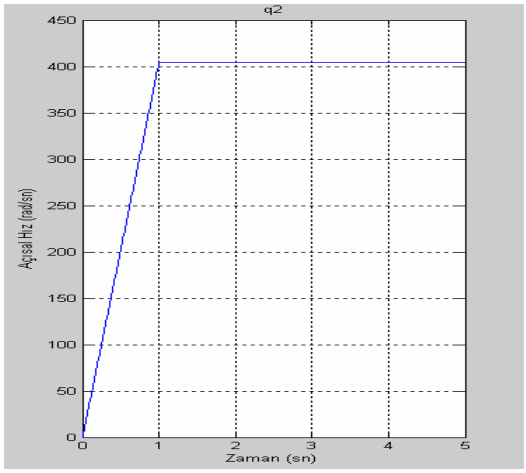




(a) **(b)**

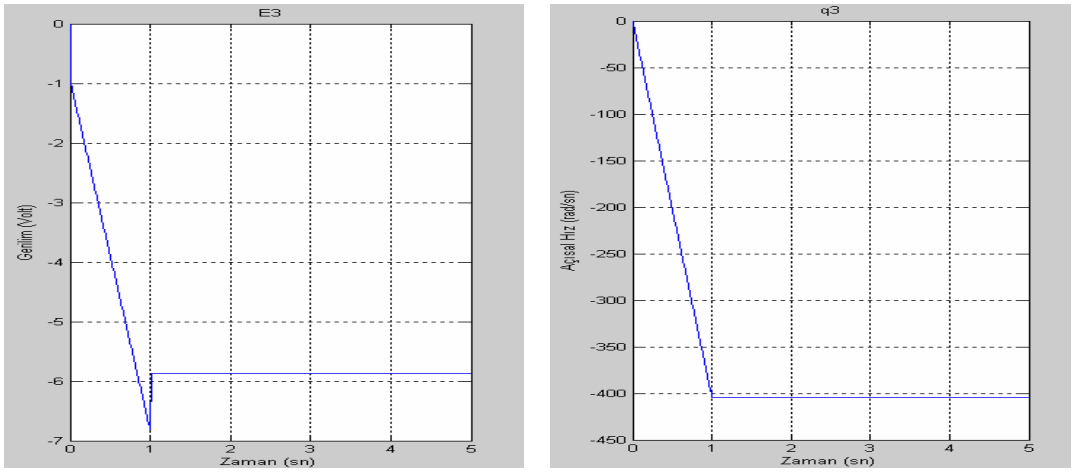
Şekil 5.6 Uygulama 1'in FPGA simülasyonu sonucunda elde edilen 1. motora ait **(a)** gerilim - zaman ; **(b)** açısal hız - zaman grafikleri





(a) **(b)**

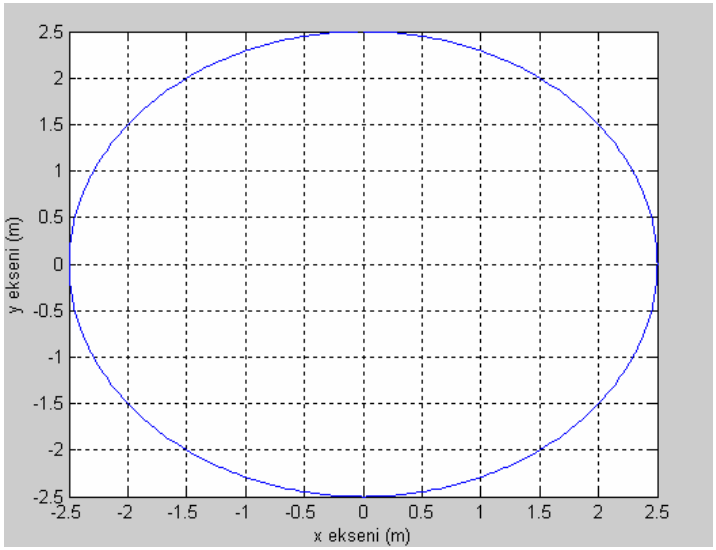
Şekil 5.7 Uygulama 1'in FPGA simülasyonu sonucunda elde edilen 2. motora ait **(a)** gerilim - zaman ; **(b)** açısal hız - zaman grafikleri



(a) **(b)**

Şekil 5.8 Uygulama 1'in FPGA simülasyonu sonucunda elde edilen 3. motora ait **(a)** gerilim - zaman ; **(b)** açısal hız - zaman grafikleri

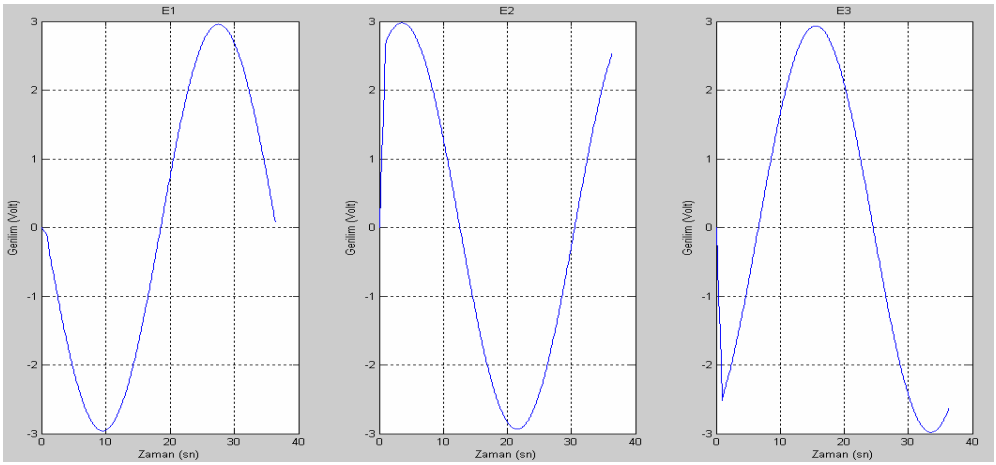
Uygulama 2 : Robotun Kendi Ekseni Etrafında Dönme Hareketi Yapmaksızın Dairesel Bir Yol İzlemesi



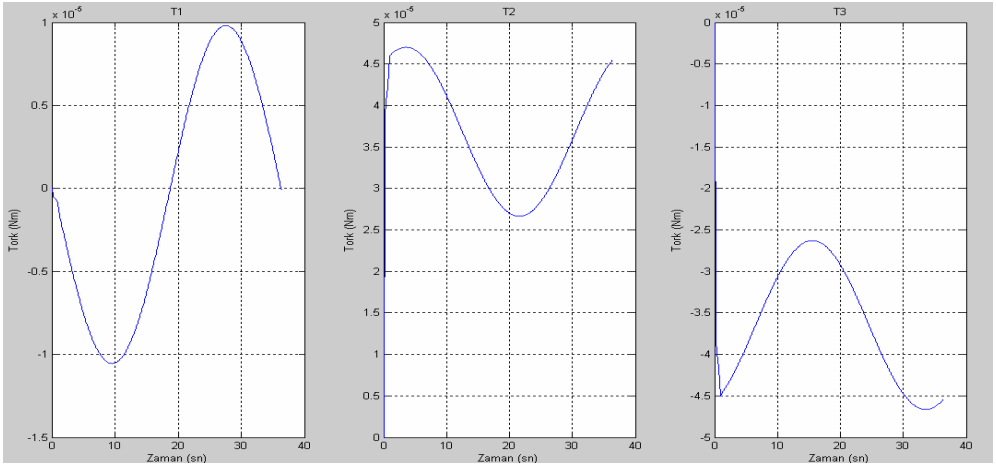
Şekil 5.9 Dairesel hareket yapan robotun izlediği yol

Başlangıç anında global eksenle arasında açı farkı olmayan robotun kendi üzerinde dönme hareketi yapmadan Şekil 5.9 'da gösterilen 2.5 m yarıçaplı dairesel bir pisti 36.4 saniyede alması için tekerleklerin bağlı olduğu motorların açısal dönme hızları ve bu hızları almak için motorlara uygulanması gereken gerilim değerleri, robot modelinin matlab ve FPGA simülasyonları sonucunda aşağıdaki gibi elde edilmiştir.

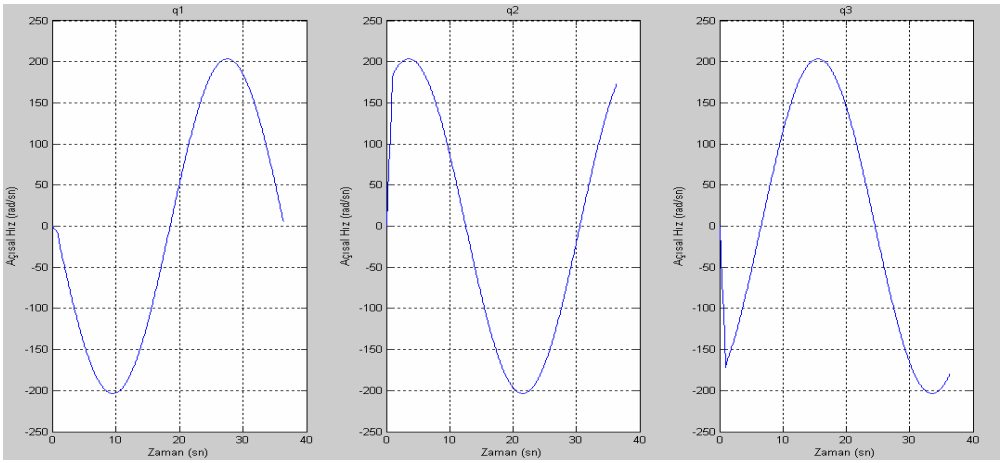
Uygulama 2 için matlab simülasyon sonuçları Şekil 5.10, Şekil 5.11 ve Şekil 5.12 ‘de verilmiştir.



Şekil 5.10 Uygulama 2’nin matlab simülasyonu sonucunda elde edilen motorlara ait açısal hız - zaman grafikleri

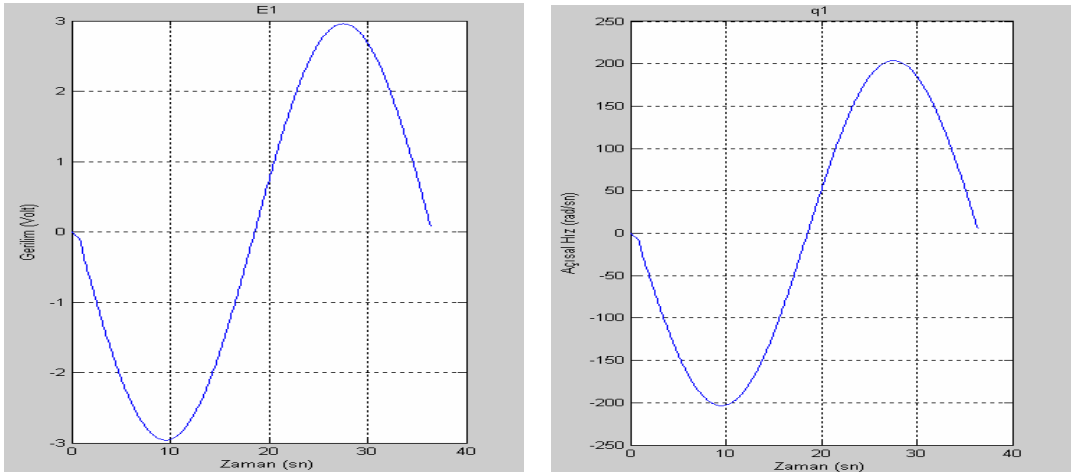


Şekil 5.11 Uygulama 2’nin matlab simülasyonu sonucunda elde edilen motorlara ait moment - zaman grafikleri



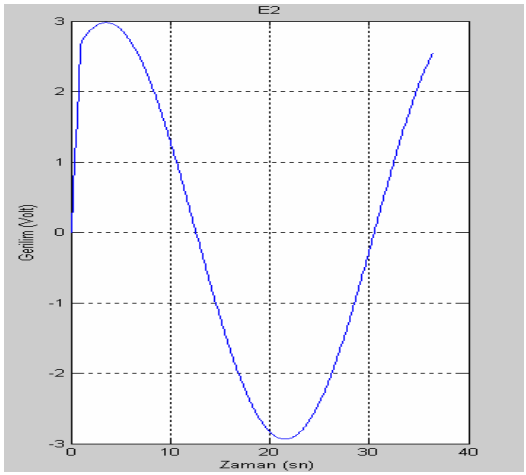
Şekil 5.12 Uygulama 2'nin matlab simülasyonu sonucunda elde edilen motorlara ait gerilim - zaman grafikleri

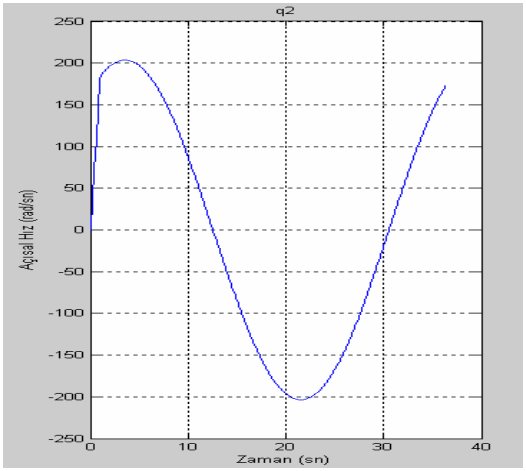
Uygulama 2 için FPGA simülasyon sonuçları Şekil 5.13, Şekil 5.14 ve Şekil 5.15 'de verilmiştir.



(a) **(b)**

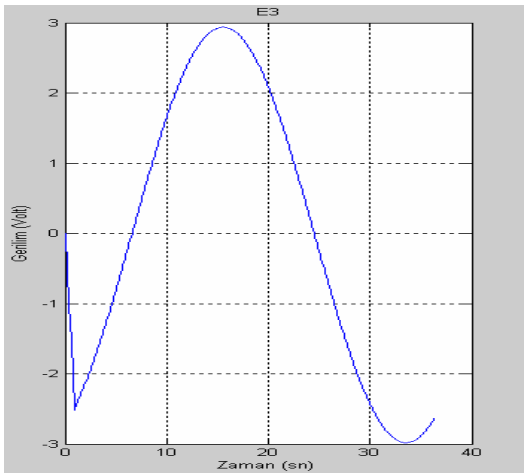
Şekil 5.13 Uygulama 2'nin FPGA simülasyonu sonucunda elde edilen 1. motora ait **(a)** gerilim - zaman ; **(b)** açısal hız - zaman grafikleri

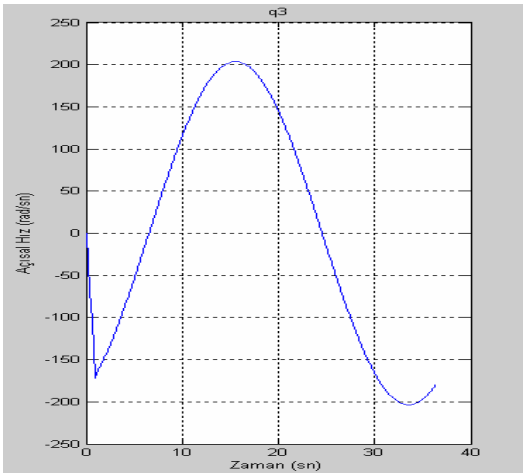




(a) **(b)**

Şekil 5.14 Uygulama 2'nin FPGA simülasyonu sonucunda elde edilen 2. motora ait **(a)** gerilim - zaman ; **(b)** açısal hız - zaman grafikleri





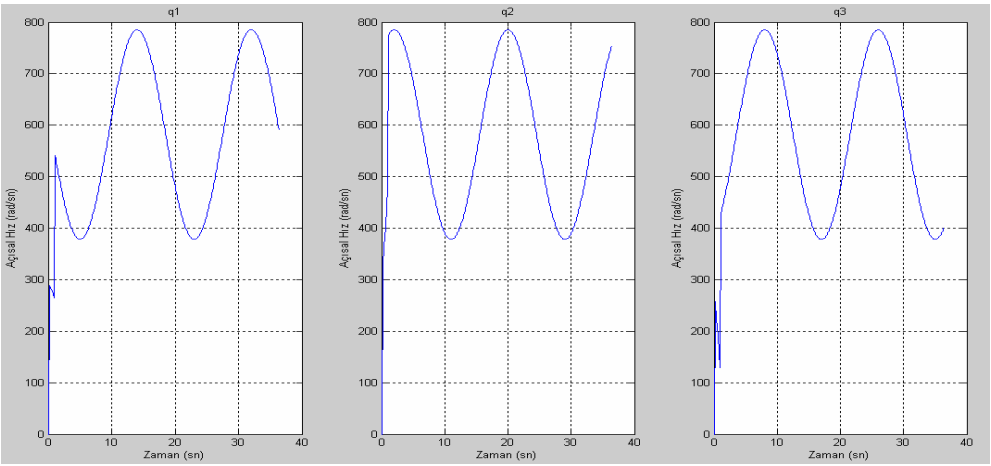
(a) **(b)**

Şekil 5.15 Uygulama 2'nin FPGA simülasyonu sonucunda elde edilen 3. motora ait **(a)** gerilim - zaman ; **(b)** açısal hız - zaman grafikleri

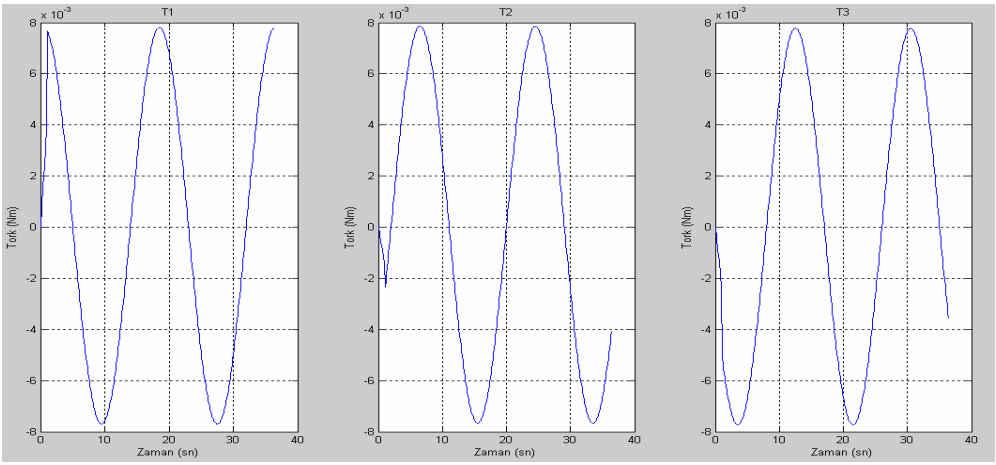
Uygulama 3 : Robotun Kendi Ekseni Etrafında Dönme Hareketi Yaparak Dairesel Bir Yol İzlemesi

Başlangıç anında global eksenle arasında açı farkı olmayan robotun her alınan 1 derecelik yolda kendi üzerinde 1 derece dönmesi şeklinde 36.4 saniyede gerçekleştirdiği dairesel hareket için elde edilen matlab ve FPGA simülasyon sonuçları aşağıda verilmiştir. İzlenen dairesel yol Şekil 5.9 'de gösterilen 2.5 m yarıçaplı dairesel bir pisttir. Yolun tamamlanması sonunda robot kendi üzerinde 360 derece dönmüş olacaktır.

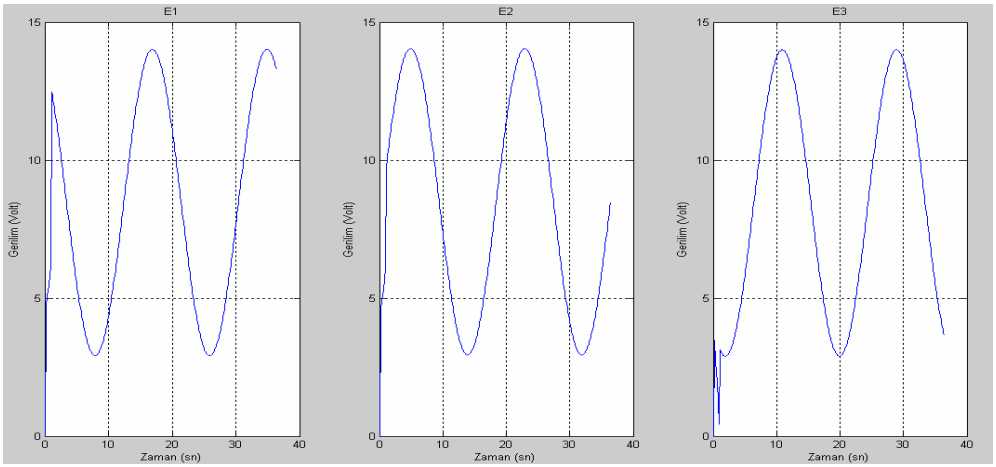
Uygulama 3 için matlab simülasyon sonuçları Şekil 5.16, Şekil 5.17 ve Şekil 5.18 'de verilmiştir.



Şekil 5.16 Uygulama 3'ün matlab simülasyonu sonucunda elde edilen motorlara ait açısal hız - zaman grafikleri

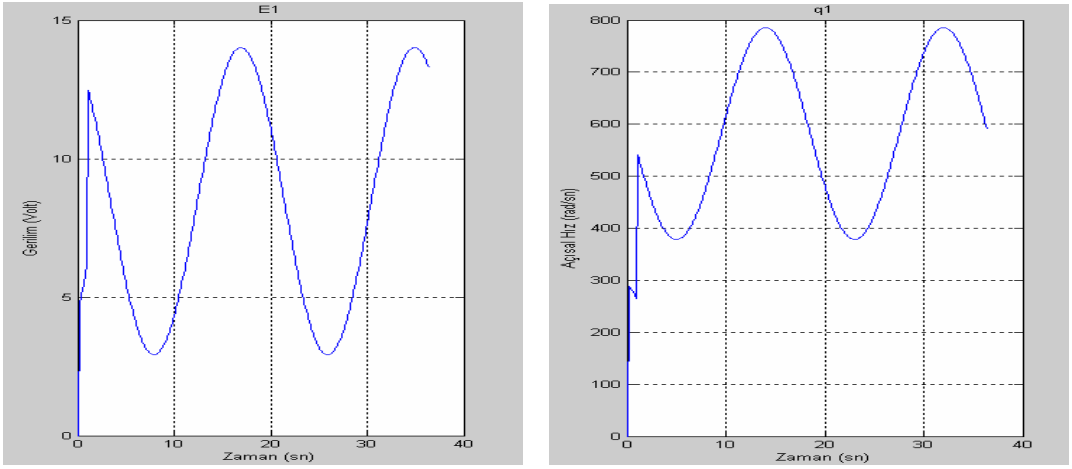


Şekil 5.17 Uygulama 3'ün matlab simülasyonu sonucunda elde edilen motorlara ait moment - zaman grafikleri



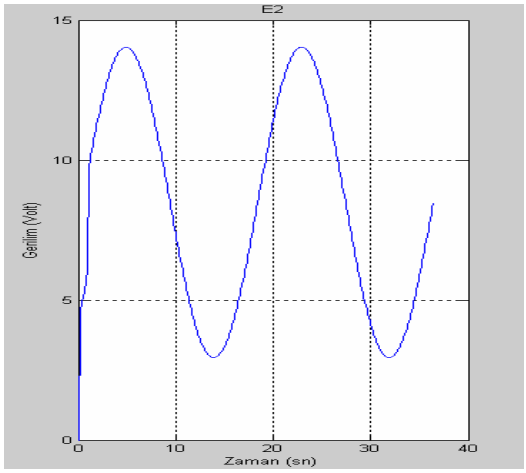
Şekil 5.18 Uygulama 3'ün matlab simülasyonu sonucunda elde edilen motorlara ait gerilim - zaman grafikleri

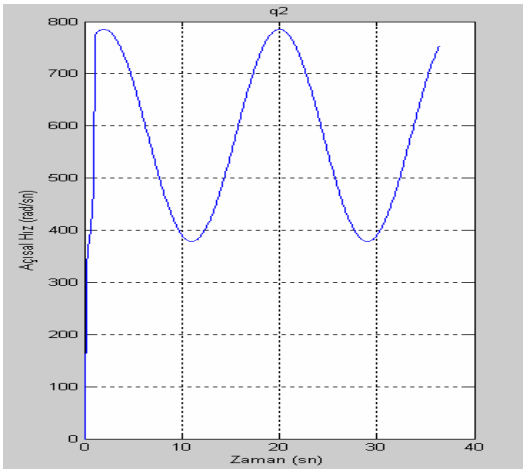
Uygulama 3 için FPGA simülasyon sonuçları Şekil 5.19, Şekil 5.20 ve Şekil 5.21 'de verilmiştir.



(a) **(b)**

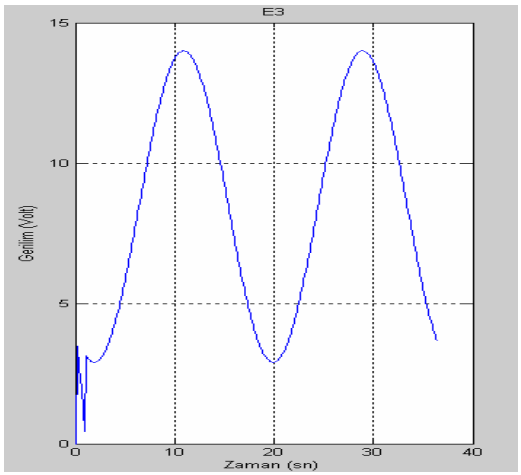
Şekil 5.19 Uygulama 3'ün FPGA simülasyonu sonucunda elde edilen 1. motora ait **(a)** gerilim - zaman; **(b)** açısal hız - zaman grafikleri

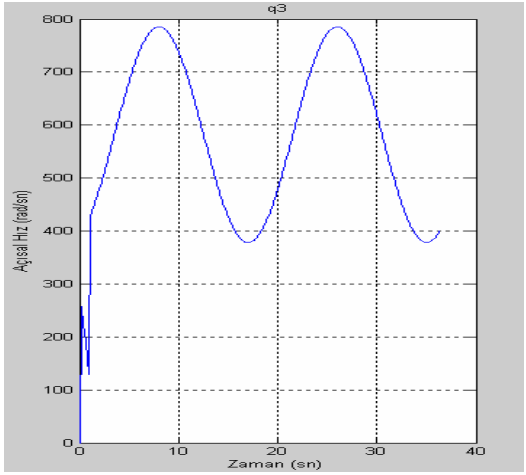




(a) **(b)**

Şekil 5.20 Uygulama 3'ün FPGA simülasyonu sonucunda elde edilen 2. motora ait **(a)** gerilim - zaman; **(b)** açısal hız - zaman grafikleri





(a) **(b)**

Şekil 5.21 Uygulama 3'ün FPGA simülasyonu sonucunda elde edilen 3. motora ait **(a)** gerilim - zaman; **(b)** açısal hız - zaman grafikleri

6. SONUÇ

Robotu temsil eden robot modeli çoğunlukla karmaşıktır ve hesaplanması zaman alır. Robot modeli yazılım ve donanım tabanlı olarak gerçekleştirilebilmektedir. Yazılım tabanlı yöntemlerde tasarım yazılımsal olarak gerçekleştirilir ve mikroişlemci üzerinde çalıştırılır. Mikroişlemcinin koşturduğu yazılımlar değiştirilerek, hiçbir donanım değişikliğine gidilmeden tasarım ortamına yeni fonksiyonlar eklenebilir. Bu, tasarıma esneklik kazandırır ancak sistem performansını ve hızını oldukça düşürür. Donanım tabanlı yöntemlerde ise sayısal verileri işlemek için ağırlıklı olarak özel tümdevreler kullanılır. İşlemler doğrudan donanımsal olarak gerçekleştirildiği için yüksek hız ve yüksek sistem performansı sağlarlar.

FPGA, elektriksel olarak programlanabilir eleman ve arabirimlerden oluşan bir tümdevredir. İçerdiği lojik ve giriş-çıkış blokları arasında programlanabilir hatlar vasıtasıyla donanımsal tasarım gerçekleştirmeyi sağlar. 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'lar hızlı işlem gerektiren uygulamalarda ve paralele işlem gerektiren özel işlemler için kullanılır. Yoğun işlem ve yüksek hız gereksinimi nedeniyle gerçek zamanlı robot modellemesinde veya bu modelin kullanılacağı bir kontrol sistemi tasarımında FPGA kullanımı gerçek zamanlı sistemin gerçekleştirilebilmesini mümkün kılar.

Bu tez çalışmasında FPGA kullanılarak gerçek zamanlı robot modellemesi gerçekleştirilmiştir. Modellemesi yapılan robot, yüksek manevra kabiliyetine sahip omni-directional robottur. Robotun ters kinematik ve dinamik modelleri ilk olarak matlab ortamında yazılımsal olarak tasarlanmış ve farklı giriş ve farklı parametre değerlerine göre sonuçlar alınarak yine matlab ortamında değerlendirilmiştir.

Aynı şekilde robotun ters kinematik ve dinamik modelleri Altera firmasının üretmiş olduğu Quartus II 8.1. FPGA programlama arayüzü kullanılarak Stratix II GX model FPGA'ı üzerine tasarlanmış ve çalıştırılmıştır. FPGA sonuçları doğruluk ve zaman açısından olmak üzere iki şekilde değerlendirilmiştir.

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

Zaman açısından da oldukça iyi sonuçlar elde edilmiştir. FPGA tasarımında robot modelinin gerçekleştirilmesi için 128 cycle'lık bir çevirime ihtiyaç duyulmaktadır. Robot modeli için gerçekleştirilen FPGA tasarımında sistem frekansı 100 MHz olduğundan model 1.28 µs'de gerçekleştirilmiştir.

Robot modelinin FPGA kullanılarak oldukça hızlı bir şekilde gerçekleştirilebilmesi, bu modelin ileriki çalışmalarda gerçek zamanlı kontrol sistemlerinde kullanılabilmesine imkan tanımaktadır.

KAYNAKLAR

1. Roland Siegwart and IIIah R. Nourbakhsh, 2004, Autonomous Mobile Robots, Massachusetts Institute of Technology

2. Mark W. Spong, Seth Hutchinson, M. Vidyasagar, 2006, Robot Modelling And Control, John Wiley & Sons Inc.

3. J.O. Hamblen, T.S. Hall and M.D. Furman, 2002, Rapid Prototyping of Digital Systems, Kluwer Academic Publishers

4. Dr. Zafer Bingül, Dr. Serdar Küçük, 2008, Robot Dinamiği ve Kontrolü, Birsen Yayınevi

5. Cofer R.C., Benjamin F. Harding, 2006, Rapid System Prototyping with FPGAs, Elsevier Inc.

6. Bob Zeidman, 2002, Designin With FPGAs And CPLDs, CMP Boks

7. Peter R. Wilson, 2007, Design Recipes for FPGAs, Newnes

8. Steve Kilts, 2007, Advanced FPGA Design, John Wiley & Sons Inc.

9. Pong P.Chu, 2008, FPGA Prototyping By VHDL Examples, John Wiley & Sons Inc.

10. F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, N. Pereira, Three Omni-Directional Wheels Control on a Mobile Robot

11. Carter B., Good M., Dorohoft M., Lew J., Williams R., Gallina P., Mechanical Design and Modeling of an Omni-directional RoboCup Player

12. Paritosh A. Kavathekar, Devin J. Balkcom, Matthew T. Mason, The Geometry of Time-Optimal Trajectories for an Omni-Directional Robot

13. Robert L. Williams, Brian E. Carter, Paolo Gallina, Giulio Rosati, March 2002, Dynamic Model with Slip for Wheeled Omni-Directional Robots, IEEE Transactions on Robotics and Automation, Vol. 18, No. 3, sf. 285-293

14. Jae Hoon Lee, Yuta S., Eiji Koyanagi, Byung-Ju Yi, 2005, Command System and Motion Control for Caster-type Omni-Directional Mobile Robot, IEEE/RSJ Internatioal Conference on Intelligent Robots and Systems

15. Dzhantimirov, Palis, Schmucker, Telesh, Zavgorodniy, HIL/DIL By Development of Six-Legged Robot Slair2

16. Timothy Paul Henning, August 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

17. Jianhua Wu, March 2005, Dynamic Path Planning of an Omni-Directional Robot in a Dynamic Environment, Ph.D. Thesis, The Fritz J. And Dolores H. Russ College of Engineering and Technology of Ohio University

18. Tan Kiong Howe, May 2003, Evaluation of the Transient Response of a DC Motor Using Matlab/Simulink, Department of Electrical Engineering University of Queensland

19. www.altera.com

20. Stratix II GX PCI Express Development Board Reference Manual

21. PCI Express Development Kit, Stratix II GX Edition Getting Started User Guide

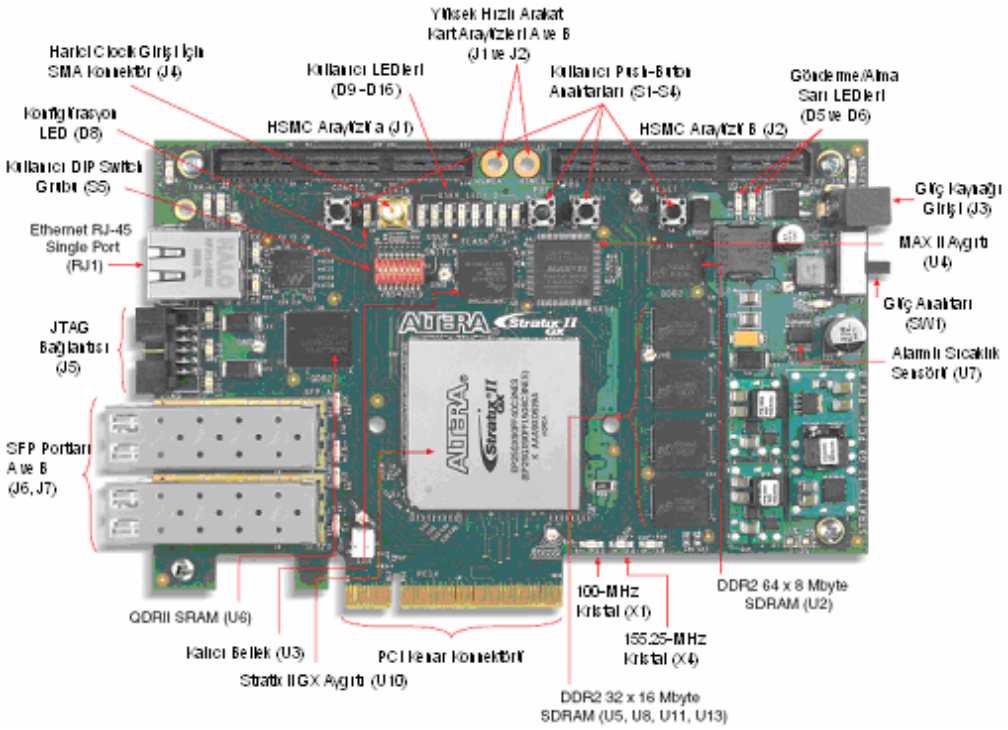
ÖZGEÇMİŞ

Gökşen ÖZDEMİR

1983	Elazığ'da doğdu.
1997 – 2001	Elazığ Balakgazi Lisesinden mezun oldu.
2001 – 2005	Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünden bölüm birincisi olarak mezun oldu.
2005 – …	Fırat Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında yüksek lisans tez çalışması başladı.
2007 – 2008	Özel bir endüstriyel otomasyon şirketinde çalıştı.
2008 – …	T.C. Adalet Bakanlığı Bilgi İşlem Dairesi Başkanlığında bilgisayar mühendisi olarak çalışmaya başladı. Halen bu kurumda çalışmaya devam etmektedir.

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 [20,21]. 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
  - Laptop tipi DC güç kaynağı

➢ 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
➢ 20 alıcı/verici kanalları
➢ 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 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:

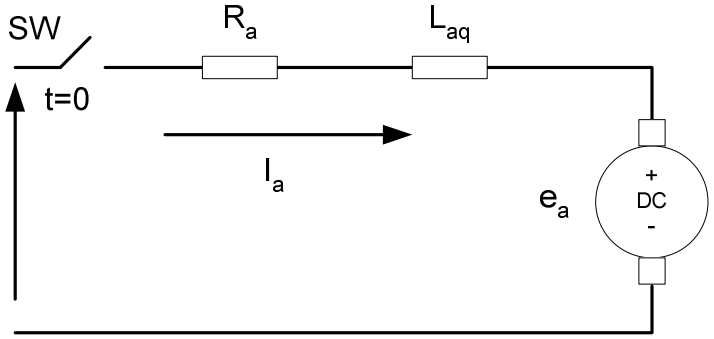
- Yüksek hız
- Düsük moment
- Ters yönde kullanım
- Sürekli hareket

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 [4].

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 (Vt) değiştirilmesiyle endüviden akan akım (Ia) değişir. Böylece akımla, motorda üretilen moment arasındaki ilişkiden moment kontrol edilir. Şekil EK-2.1 ’de DC motorun eşdeğer devresi gösterilmektedir.



Şekil EK-2.1 DC motorun eşdeğer devresi

Şekil EK-2.1 'de temsil edilen DC motora ait değişkenler şunlardır:

Vt = Endüvi gerilimi

Laq = Endüvi endüktansı

Ra = Endüvi direnci

ea = Zıt elektro motor kuvveti

Ia = Endüvi akımı

ωm = Motorun açısal hızı (raydan/sn)

T = Moment

TL = 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 [4].

Şekil EK-2.1'deki DC motorun eşdeğer devresi baz alınarak motorun matematiksel modeli şu şekilde elde edilir [18].

Manyetik doğrusallık varsayılarak temel motor eşitlikleri şöyle verilebilir:

$$T = K_f i_f i_a = K_m i_a \qquad \text{(EK-2.1)}$$

$$e_a = K_f i_f \omega_m = K_m \omega_m \qquad \text{(EK-2.2)}$$

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

Eşitlik EK-2.1 ve EK-2.2 'nin Laplace dönüşümleri sonucunda şu eşitlikler elde edilir:

$$T(s) = K_m i_a(s) \qquad \text{(EK-2.3)}$$

$$E_a = K_m \omega_m(s) \tag{EK-2.4}$$

t=0 anında SW switch'i kapatılırsa;

$$V_t = e_a + R_a i_a + L_{aq} \frac{di_a}{dt} \tag{EK-2.5}$$

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

$$V_t = K_m \omega_m + R_a i_a + L_{aq} \frac{di_a}{dt} \tag{EK-2.6}$$

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

$$V_t(s) = K_m \omega_m(s) + R_a I_a(s) + L_{aq} s I_a(s) \tag{EK-2.7}$$

veya

$$V_t(s) = K_m \omega_m(s) + I_a(s) R_a (1 + s\tau_a) \tag{EK-2.8}$$

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

Mekanik sistem için dinamik eşitlik;

$$T = K_m i_a = J \frac{d\omega_m}{dt} + B\omega_m + T_L \tag{EK-2.9}$$

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

Eşitlik EK-2.1 ve Eşitlik EK-2.9 'un Laplace dönüşümleri sonucu;

$$T(s) = K_m i_a(s) = J s\omega_m(s) + B\omega_m(s) + T_L(s) \tag{EK-2.10}$$

olur.

Eşitlik EK-2.10 ve EK-2.3 'den;

$$\omega_m(s) = \frac{T(s) - T_L(s)}{B(1 + s J/B)} = \frac{K_m I_a(s) - T_L(s)}{B(1 + s\tau_m)} \tag{EK-2.11}$$

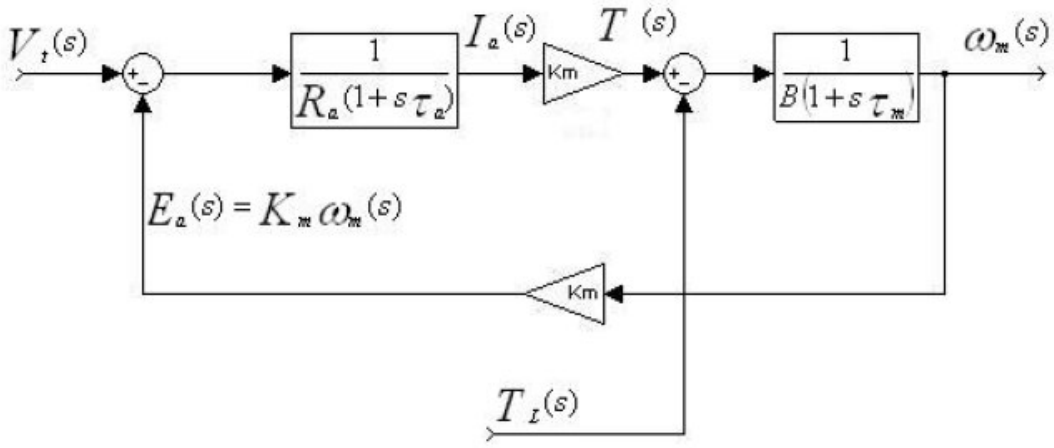
elde edilir.Burada $\tau_m = J/B$, sistemin mekaniksel zaman sabitidir.

Eşitlik EK-2.4 ve EK-2.8 'den

$$I_a(s) = \frac{V_t(s) - E_a(s)}{R_a(1 + s\tau_a)} = \frac{V_t(s) - K_m \omega_m(s)}{R_a(1 + s\tau_a)} \tag{EK-2.12}$$

elde edilir.

Eşitlik EK-2.10 ve EK-2.11 'in blok diyagram olarak gösterimi Şekil EK-2.2 'de verilmiştir.



Şekil EK-2.2 DC motorun matematiksel modelinin blok diyagramı