

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ



FPGA TABANLI GENEL AMAÇLI DİJİTAL İKİZ
SİSTEMİNİN GELİŞTİRİLMESİ VE ROBOT
SİMÜLATÖRÜ İLE BİRLİKTE KULLANILMASI

Mehmet Rıza SARAÇ

Doktora Tezi

MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI
Bilgisayar Sistemleri Bilim Dalı

ŞUBAT 2023

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Mekatronik Mühendisliği Anabilim Dalı

Doktora Tezi

FPGA TABANLI GENEL AMAÇLI DİJİTAL İKİZ
SİSTEMİNİN GELİŞTİRİLMESİ VE ROBOT
SİMÜLATÖRÜ İLE BİRLİKTE KULLANILMASI

Tez Yazarı

Mehmet Rıza SARAÇ

Danışman

Prof. Dr. Ömür AYDOĞMUŞ

ŞUBAT 2023

ELAZIĞ

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Mekatronik Mühendisliği Anabilim Dalı
Doktora Tezi

Başlığı:	FPGA Tabanlı Genel Amaçlı Dijital İkiz Sisteminin Geliştirilmesi ve Robot Simülatörü ile Birlikte Kullanılması
Yazarı:	Mehmet Rıza SARAÇ
İlk Teslim Tarihi:	06.01.2023
Savunma Tarihi:	14.02.2023

TEZ ONAYI

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına göre hazırlanan bu tez aşağıda imzaları bulunan jüri üyeleri tarafından değerlendirilmiş ve akademik dinleyicilere açık yapılan savunma sonucunda OYBİRLİĞİ ile kabul edilmiştir.

İmza

Danışman:	Prof. Dr. Ömür AYDOĞMUŞ Fırat Üniversitesi, Teknoloji Fakültesi	Onayladım
Başkan:	Prof. Dr. Muhammed Fatih TALU İnönü Üniversitesi, Mühendislik Fakültesi	Onayladım
Üye:	Prof. Dr. Ayşegül UÇAR Fırat Üniversitesi, Teknoloji Fakültesi	Onayladım
Üye:	Doç. Dr. Fatih ERTAM Fırat Üniversitesi, Teknoloji Fakültesi	Onayladım
Üye:	Dr. Öğr. Üyesi Mustafa Can BİNGÖL Burdur Mehmet Akif Ersoy Üniversitesi, Mühendislik ve Mimarlık Fakültesi	Onayladım

Bu tez, Enstitü Yönetim Kurulunun/...../2023 tarihli toplantısında tescillenmiştir.

Prof. Dr. Burhan ERGEN
Enstitü Müdürü

BEYAN

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım “FPGA Tabanlı Genel Amaçlı Dijital İkiz Sisteminin Geliştirilmesi ve Robot Simülatörü ile Birlikte Kullanılması” Başlıklı Doktora Tezimin içindeki bütün bilgilerin doğru olduğunu, bilgilerin üretilmesi ve sunulmasında bilimsel etik kurallarına uygun davrandığımı, kullandığım bütün kaynakları atıf yaparak belirttiğimi, maddi ve manevi desteği olan tüm kurum/kuruluş ve kişileri belirttiğimi, burada sunduğum veri ve bilgileri unvan almak amacıyla daha önce hiçbir şekilde kullanmadığımı beyan ederim.

14.02.2023

Mehmet Rıza SARAÇ

ÖNSÖZ

Gerçek zamanlı simülatörler, mevcut ürünlerin yeni sistemlerde kullanımında yeni ürünlerin tasarımında kullanılarak prototip üretim maliyetlerini ve test maliyetlerini düşürmektedir. Bu şekilde geliştirme süreçlerine destek olmakta, hasar verme ihtimali olan testlerde ürünler veya sistemlerin kayıpları ile oluşan maddi zayıata engel olmaktadır. Bunları yaparken mümkün olduğunca gerçek sistem tepkilerinin benzetimini gerçek zamanlı olarak gerçekleştirerek tasarım süreçlerine yardımcı olmaktadır.

Sonlu Eleman Analizi gibi modellemelerde olduğu kadar hassas ve detaylı modelleme yapılamasa bile geliştirme ve tasarım süreçlerinde zaman ve mali olarak önemli ölçüde esneklik ve faydalar sağlamaktadırlar.

Gerçek zamanlı simülatörlerin sayısal türleri analog türlerine nazaran kalibrasyon ve hata toleransı açılarından daha uygun seçenekler sunmaktadır. Standart olmayan işlemlerin gerçek zamanlı sayısal simülatörlerde programlanabilir yongalarda gerçekleştirilmesi, simülatörün esnekliğini artırır ve gerçek zaman hızına ulaşılmasını sağlar. Bu, hem model değişimlerinde hem de mevcut modelin parametre değişimlerinde önemlidir. Ancak, programlanabilir yongaların donanım tasarımı işleminin gerektirdiği uzmanlık ve deneyim çoğu zaman belli sağlayıcılar ve bu sağlayıcıların desteklediği belli yongalarla sınırlıdır, bu da kısıtlayıcı bir durum oluşturmaktadır.

Bu çalışma sırasında simülatör sağlayıcı şirketlerden bağımsız şekilde sistem modelleri doğrudan programlanabilir yongalar için sentezlenmiş, model parametre değişimleri gerçek zamanlı değişime olanak tanıyacak şekilde modele entegre edilmiştir.

Çalışmalarım süresince gerek sunduğu donanım ve yazılım desteği ve gerekse donanım geliştirme hususunda harcamak durumunda kaldığım süreç boyunca esirgemediği sabır ve anlayış için danışman hocam Sayın Prof. Dr. Ömür AYDOĞMUŞ'a teşekkürü bir borç bilirim.

Mehmet Rıza SARAÇ

ELAZIĞ, 2023

İÇİNDEKİLER

	Sayfa
ÖNSÖZ	iv
İÇİNDEKİLER	vi
ÖZET	vii
ABSTRACT	viii
ŞEKİLLER LİSTESİ	xi
TABLolar LİSTESİ	xii
EKLER LİSTESİ	xiii
SİMGELER VE KISALTMALAR	xiv
1. GİRİŞ	1
2. DONANIM GELİŞTİRME	7
2.1. Donanım Geliştirmenin Temelleri	7
2.1.1. Mantıksal Tasarım	7
2.1.2. Fiziksel Tasarım	9
2.1.3. Yonga Üretimi ve Paketlenmeleri	9
2.2. FPGA ile Donanım Geliştirme	10
2.2.1. FPGA Yongalarının Genel Yapısı	11
2.2.2. Donanım Tanımlama Dilleri	12
2.3. Donanım Geliştirme İşlemlerinde Zamanlama	14
2.3.1. Mantık Devrelerinde Gecikme	14
2.3.2. Kaydediciler için Temel Zamanlama Terimleri	15
2.4. Synopsys Tasarım Kısıtları	22
2.4.1. SDC Saat Tanımlama	23
2.5. Sentezleme ve Tasarlanan Donanımın FPGA Yongasına Aktarma	24
3. SİSTEM DONANIM ALTYAPISI	28
3.1. DE10-Nano Geliştirme Kartı	28
3.1.1. FPGA Yonga Özellikleri	31
3.1.2. ADC Entegre Özellikleri	35
3.1.3. GPIO Port Özellikleri	38
3.1.4. USB-Blaster II Bağlantı Arayüzü	38
3.2. Analog Giriş Çıkış Kartı	38
3.2.1. Analog Sinyal Giriş Devresi	38
3.2.2. Analog Sinyal Çıkış Devresi	39
3.3. Sürücü Kartı	41
3.4. Sabit Mıknatıslı Çekirdeksiz Doğru Akım Motoru	42
3.5. Arttırımlı Enkoder	43
3.6. Analog Discovery 2 ve Digital Discovery USB Osiloskop, Lojik Analizör ve Sinyal Jeneratörü Cihazları	44
4. FPGA MODÜL TASARIMLARI	46
4.1. Saat ve PLL Sistemi	46
4.2. Asenkron Resetleme Sistemi	47
4.3. ADC Modülü	47
4.4. Simülatör Modülleri	49
4.4.1. Modül Giriş ve Çıkış Prosedürleri	50

4.4.2. Kayar Nokta İşlem Birimleri	51
4.5. DAC Modülü	52
4.6. Harici Motor Veri Okuma Modülü	53
4.6.1. Sensör Okuma Sistemi	54
4.6.2. Açısal Hız Hesaplama Sistemi	55
5. SİMÜLATÖR TASARIMLARI	56
5.1. RL Devre SİMÜLATÖRÜ	57
5.1.1. Sistem Modeli	57
5.1.2. Bilgisayar SİMÜLASYONU	58
5.1.3. FPGA SİMÜLASYON MODÜLÜ	64
5.1.4. Karşılaştırmalı Deneyler	65
5.2. PMDC Motor SİMÜLATÖRÜ	67
5.2.1. Sistem Modeli	68
5.2.2. Bilgisayar SİMÜLASYONU	71
5.2.3. FPGA SİMÜLASYON MODÜLLERİ	72
5.2.4. Karşılaştırmalı Deneyler	76
5.3. Webots Robot SİMÜLATÖRÜ	78
5.3.1. SİMÜLATÖR KONFIGÜRASYONU	80
5.3.2. Webots ve FPGA SİMÜLATÖRLERİNİN BİRLİKTE ÇALIŞTIRMALARI	80
5.3.3. Deneyler	82
6. BULGULAR VE TARTIŞMA	91
7. SONUÇLAR	93
GELECEK ÇALIŞMALAR VE ÖNERİLER	94
KAYNAKLAR	95
EKLER	101
ÖZGEÇMİŞ	114

ÖZET

FPGA Tabanlı Genel Amaçlı Dijital İkiz Sisteminin Geliştirilmesi ve Robot Simülatörü ile Birlikte Kullanılması

Mehmet Rıza SARAÇ

Doktora Tezi

FIRAT ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

Mekatronik Mühendisliği Anabilim Dalı
Bilgisayar Sistemleri Programı
Şubat 2023, Sayfa: xv + 100

Endüstriyel uygulamaların birçoğu gerçekleştirilmeden önce, modellenerek, benzetim ortamında kontrol edilerek veya test edilerek incelenmektedir. Bu, kontrol sistemlerini tasarlamak, test etmek, süreç performansını optimize etmek veya operatörleri eğitmek için kullanılmaktadır. Genellikle, işletim sistemiyle çalışan bilgisayar tabanlı sistemlerle kolayca elde edilebilen bu benzetimler, yazılımsal bir bakış açısı sunar ve bu nedenle modellenmiş sistemin çalışma süresi ile benzetim süresi arasında büyük farklar oluşabilmektedir. Bu problemi çözmek için, gerçek zamanlı çalışabilen Gerçek Zamanlı Dijital Simülatör (Real Time Digital Simulator RTDS) veya dijital ikiz fikri ortaya çıkmıştır. Dijital sistemler, paralel çalışma kabiliyetine sahip olan Sahada Programlanabilir Kapı Dizileri (Field Programmable Gate Arrays FPGAs) gibi çözümlerle ihtiyacı tatmin edici bir şekilde karşılayabilmektedir.

Bu tez çalışmasında, sabit mıknatıslı doğru akım motoru modellenerek FPGA üzerinde gerçek zamanlı olarak çalıştırılmıştır. Aynı koşullar altında, hem gerçek motor hem de modellenmiş FPGA tabanlı dijital ikizi gerçek zamanlı olarak çalıştırılmıştır. Hem dijital ikiz motorun hemde gerçek fiziksel motorun armatür akımları ve motor hızları anlık olarak aynı osiloskop ekranında incelenmiştir.

Sonuçlar, dijital ikiz motor ile gerçek motor arasında yüksek doğrulukta benzer dinamiklerin elde edilebileceğini göstermiştir. Ayrıca PMDC motorların FPGA temelli dijital ikizi, robot simülasyon programı olan Webots uygulaması ile bir diferansiyel tahrikli mobil robotun motorlarının elektriksel tepkilerinin simülasyonu için ek simülatör olarak kullanılmıştır. Bu çalışmalar sonucunda motorun akım, hız ve moment değişkenleri farklı yük ve yörüngeler için elde edilmiştir. Önerilen sistemle gerçek zamanlı uygulama ortamlarının dijital ikizlerinin elde edilebileceği gösterilmiştir.

Anahtar Kelimeler: FPGA, RTDS, Dijital İkiz.

ABSTRACT

Development of an FPGA-Based General Purpose Digital Twin and Using as Co-Simulator With Robot Simulator

Mehmet Rıza SARAÇ

Ph.D. Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences

Department of Mechatronics Engineering

February 2023, Page: xv + 100

Many industrial applications are examined before being implemented through modeling, control in a simulated environment, or testing. This is used to design control systems, test them, optimize process performance, or train operators. These simulations are usually easily obtained using computer-based systems that work with an operating system and offer a software-based perspective, resulting in significant differences between the simulation time and the working time of the modeled system. The idea of a Real-Time Digital Simulator (RTDS) or digital twin, which can work in real-time, has emerged to solve this problem. Digital systems can meet this need satisfactorily with solutions such as parallel working capability of Field Programmable Gate Arrays (FPGAs).

In this thesis, a permanent magnet direct current motor is modeled and operated in real-time on an FPGA. Both the real motor and the modeled FPGA based digital twin are operated in real-time under the same conditions. Armature currents and motor speeds of both physical motor and its digital twin were observed simultaneously on the same oscilloscope screen.

The results showed that similar dynamics can be obtained with high accuracy between the digital twin motor and the real motor. In addition, FPGA-based digital twin of PMDC motor used as co-simulator with Webots application which is used to simulate robots, for electrical responses of motors of a differential driven mobile robot, and the current, speed, and moment variables were obtained for different loads and orbits in real-time. The proposed system demonstrates that digital twins of real-time application environments can be obtained.

Keywords: FPGA, RTDS, Digital Twin.

ŞEKİLLER LİSTESİ

	Sayfa
Şekil 2.1.	3 girişli Bir LUT yapısı ve SRAM hücreleri 11
Şekil 2.2.	Tasarımda kullanılan IP blokları listesi 13
Şekil 2.3.	İdeal yayılma gecikmesi 14
Şekil 2.4.	İdeal olmayan yayılma gecikmesi 15
Şekil 2.5.	D tipi kaydedicilerin veri girişi için zamanlama gereksinimleri 16
Şekil 2.6.	D tipi kaydedicilerin asenkron reset girişleri için zamanlama gereksinimleri . . . 17
Şekil 2.7.	Giriş sinyallerini senkron hale getirmek için kullanılan bir eş-zamanlama çifti . . 20
Şekil 2.8.	Sentezleme sonrasında RTL gösterimde görünen bir veri hattı eş-zamanlama çifti 20
Şekil 2.9.	Asenkron reset sinyalinin bitişi için senkronlayıcı devre 21
Şekil 2.10.	Sentezleme sonrasında RTL gösterimde görünen bir reset hattı eş-zamanlama çifti 22
Şekil 2.11.	Quartus Prime yazılımı Timing Analyzer modülü saat tanımlama ekranı 23
Şekil 2.12.	Quartus Prime yazılımı Timing Analyzer modülü üretilmiş saat tanımlama ekranı 24
Şekil 2.13.	Sentezleme sonrasında sistem için oluşturulan MTBF raporu 25
Şekil 2.14.	Sentezleme sonrasında zamanlama raporlarında görülen saat bölgeleri arası geçişler 26
Şekil 2.15.	Sentezleme süreci adımları ve işlem süreleri 26
Şekil 2.16.	Sentezleme süreci sonunda oluşturulan rapor 27
Şekil 3.1.	DE10-Nano geliştirme kartı ön yüzü 28
Şekil 3.2.	DE10-Nano geliştirme kartı arka yüzü 29
Şekil 3.3.	DE10-Nano geliştirme kartı blok yapısı 29
Şekil 3.4.	DE10-Nano geliştirme kartı blok yapısı 2 30
Şekil 3.5.	DE10 saat işareti üreticisi 30
Şekil 3.6.	FPGA yongası isim kodlaması 31
Şekil 3.7.	Cyclone V yongalarında bulunan global saat dağıtım hat ve bölgeleri 33
Şekil 3.8.	Cyclone V yongalarında bulunan ALM yapıları 34
Şekil 3.9.	Cyclone V yongalarında bulunan ALM yapısı detayı 34
Şekil 3.10.	Cyclone V yongalarında bulunan DSP modül yapısı 35
Şekil 3.11.	ADC entegresi bağlantı soketi 36
Şekil 3.12.	ADC entegresi blok şeması 37
Şekil 3.13.	ADC entegresi SPI arayüzü zamanlama çizelgesi 37
Şekil 3.14.	Analog giriş sisteminin giriş sinyallerini ölçeklemesi 39
Şekil 3.15.	Analog giriş sinyallerinin ölçeklenmesinde kullanılan devre şeması 39
Şekil 3.16.	DAC entegresi fonksiyonel blok diyagramı 40
Şekil 3.17.	DAC entegresi PI arayüzü zamanlama çizelgesi 40
Şekil 3.18.	Sürücü kartı 42
Şekil 3.19.	AMT10 serisi enkoder ve motora bağlantısı 43
Şekil 3.20.	AMT10 serisi enkoder zamanlama çizelgesi 44
Şekil 3.21.	Analog Discovery 2 ve Digital Discovery USB tipi osiloskop, lojik analizör 44
Şekil 4.1.	ADC entegresinin okunmasını sağlayan modülün durum makinesi 48
Şekil 4.2.	Kayar nokta bölme işlemi IP bloğu için konfigürasyon pencereleri 1 51
Şekil 4.3.	Kayar nokta bölme işlemi IP bloğu için konfigürasyon pencereleri 2 52
Şekil 4.4.	DAC kontrol modülü ve girişini kontrol eden veri senkronizasyon modülü 53

Şekil 4.5.	Harici motor için hız ve konum hesaplama modülü	53
Şekil 4.6.	Harici motor için enkoder verisi işleme modülü	54
Şekil 5.1.	Blok diyagramlarından nümerik hesaplama yapmak için gerekli dönüşümler . .	56
Şekil 5.2.	Basit RL devresi şeması	57
Şekil 5.3.	Qucs simülasyon programı arayüzü ve oluşturulan devre	58
Şekil 5.4.	Qucs programı RL devresi simülasyon sonuçları	59
Şekil 5.5.	OpenModelica simülasyon programı arayüzü ve oluşturulan sistem modeli . . .	59
Şekil 5.6.	OpenModelica programında kullanılan Modelica simülasyon dili	60
Şekil 5.7.	OpenModelica simülasyon sonuçlarının gösterimi	60
Şekil 5.8.	Octave ile denklem çözüm yöntemi kullanılarak elde edilen sonuçlarının gösterimi	62
Şekil 5.9.	RL devresinin ilk durum değerini de içeren blok diyagramı	62
Şekil 5.10.	RL devresinin model yaklaşımı ile simülasyon tasarımı için blok diyagramı . .	63
Şekil 5.11.	Octave ile modelin doğrudan kodlanmasıyla elde edilen sonuçlarının gösterimi .	63
Şekil 5.12.	RL devresi denklem çözücü model blok şeması	65
Şekil 5.13.	RL simülatörü ve fiziksel devre karşılaştırması, kare dalga girişi	66
Şekil 5.14.	RL simülatörü ve fiziksel devre karşılaştırması, sinüs sinyali ve ek harmonik . .	66
Şekil 5.15.	RL simülatörü ve fiziksel devre giriş ve çıkış sinyalleri karşılaştırması (sinyal gecikmesi)	67
Şekil 5.16.	Sabit mıknatıslı doğru akım motoru modeli	68
Şekil 5.17.	PMDC motor sistem blok diyagramı	69
Şekil 5.18.	PMDC motor sistem blok diyagramı eklenen ilk durum blokları	69
Şekil 5.19.	FPGA sistemine aktarılacak şekli ile PMDC motor sistem blok diyagramı . . .	70
Şekil 5.20.	Octave ile elde edilen PMDC motor simülasyon sonuçları	72
Şekil 5.21.	PMDC motor simülatör ve fiziksel sistem blok şeması	73
Şekil 5.22.	PMDC motor simülatörü parametre belirleme kısmı blok diyagramı	74
Şekil 5.23.	PMDC motor simülatörü blok diyagramı	75
Şekil 5.24.	PMDC motor simülatörü deney düzeneği	76
Şekil 5.25.	Fiziksel PMDC motor ve tasarlanmış gerçek zamanlı simülatörün karşılaştırılması	77
Şekil 5.26.	PMDC motor simülatörü ve fiziksel motor sistemi arasındaki farklı anlık gecikme durumları	77
Şekil 5.27.	PMDC motor simülatörü ve fiziksel motor sistemlerinin kare dalga gürültü gerilimi ile çalıştırılarak karşılaştırılması	78
Şekil 5.28.	Webots robot simülatörü kullanıcı arayüzü	79
Şekil 5.29.	RTDS sistemin Webots simülatörü ile birlikte çalıştırılması	80
Şekil 5.30.	Motor gerilimi kontrol sistemi	81
Şekil 5.31.	Webots simülatörü ile birlikte simülasyon (doğrusal-1)	83
Şekil 5.32.	Webots simülatörü ile birlikte simülasyon (doğrusal-2)	84
Şekil 5.33.	Webots simülatörü ile birlikte simülasyon (doğrusal-3)	84
Şekil 5.34.	Webots simülatörü ile birlikte simülasyon (doğrusal-4)	85
Şekil 5.35.	Webots simülatörü ile birlikte simülasyon, dairesel-1	86
Şekil 5.36.	Webots simülatörü ile birlikte simülasyon, dairesel-2	86
Şekil 5.37.	Webots simülatörü ile birlikte simülasyon, dairesel-3	87
Şekil 5.38.	Webots simülatörü ile birlikte simülasyon, 8-1	88
Şekil 5.39.	Webots simülatörü ile birlikte simülasyon, 8-2	88

Şekil 5.40. Webots simülatörü ile birlikte simülasyon, 8-3	89
Şekil 5.41. Webots simülatörü ile birlikte simülasyon, 8-4	89
Şekil 5.42. Webots simülatörü ile birlikte simülasyon, 8-5	90



TABLÖLAR LİSTESİ

	Sayfa
Tablo 3.1. Intel Cyclone V SE 5CSEBA6U23I7NDK donanım kaynakları.	32
Tablo 3.2. Linear Technology LTC2308 ADC entegresi özellikleri	36
Tablo 3.3. Maxon F2140-934 motoru özellikleri	43
Tablo 5.1. OpenModelica programı ile basit RL devresi simülasyonu aşamaları boyunca harcanan süreler.	61
Tablo 5.2. PMDC motor simülatör modülü ve parametre hazırlama modülü kaynak kullanımı kıyaslaması.	75
Tablo 5.3. PMDC motor simülatörü FPGA kaynak kullanımı bilgileri	75

EKLER LİSTESİ

	Sayfa
Ek- 1:	Setup Slack (Kurma Payı) Hesaplama şekli 101
Ek- 2:	Hold Slack (Tutma Payı) Hesaplama şekli 102
Ek- 3:	İntel Quartus Prime Lite Donanım Geliştirme Platformu 103
Ek- 4:	İntel Quartus Prime Programı Pin Planlama Arayüzü 104
Ek- 5:	Signal Tap Logic Analyzer Arayüzü 105
Ek- 6:	İntel Quartus Prime Programı Timing Analyzer Modülü 106
Ek- 7:	Cyclone-V entegresiALM ünitesi..... 107
Ek- 8:	Analog Çıkış Devresi 108
Ek- 9:	Quartus Prime Yazılımı Chip Planner Modülü 109
Ek- 10:	PMDC Motor Simülatörünün FPGA Yongasında Yüzey Yerleşimi 110
Ek- 11:	PMDC Motor Simülatörü FPGA Modülleri RTL Görünümü 111
Ek- 12:	Webots Robot Simülatörü ile Birlikte Çalıştırılan Simülatörün RTL Görünümü . 112
Ek- 13:	Analog Giriş / Çıkış Kartı 113

SİMGELER VE KISALTMALAR

Simgeler

t_{db}	:	Girişe göre çıkış sinyali değişimin başlangıç zamanı
t_{ds}	:	Girişe göre çıkış sinyalinde ki değişimin bitiş zamanı
t_{su}	:	Setup time (kurma zamanı)
t_{hd}	:	Hold time (tutma zamanı)
t_{pd}	:	Propagation Delay (Yayılma gecikmesi)
t_{cq}	:	Kaydedicinin saat işaretinin aktif kenarından itibaren veriyi çıkışa aktarması için gereken süre
t_r	:	Recovery time (toparlanma zamanı)
t_m	:	Removal time (giderme zamanı)

Kısaltmalar

AHDL	:	Altera Hardware Description Language (Altera Donanım Tanımlama Dili)
ALM	:	Adaptive Logic Module (Adaptif Mantık Birimi)
ASIC	:	Application Specific Integrated Circuit (Uygulamaya Özel Entegre Devre)
ASSP	:	Application Specific Standard Product (Uygulamaya Özel Standart Ürün)
BGA	:	Ball Grid Array (İletken Bilyalardan Oluşan bağlantı ızgara dizgesi)
CPU	:	Central Processing Unit (Merkezi İşlem Birimi)
CDC	:	Clock Domain Crossing (Saat etki bölgeleri arası geçiş)
CMOS	:	Complementary MOS (Eşlenik MOS Yapısı)
CPLD	:	Complex Programmable Logic Device (Karmaşık Programlanabilen Mantık Devresi)
CPR	:	Counts Per Revolution (Devir başına düşen sayım)
DC	:	Direct Current (Doğru Akım)
DLL	:	Delay Locked Loop (Gecikme Kilitli Döngü)
DRC	:	Design Rule Check (Tasarım kurallarına uygunluk kontrolü)
DSP	:	Digital Signal Processing (Sayısal İşaret İşleme)
DT	:	Digital Twin (Dijital İkiz)
DTA	:	Dynamic Timing Analysis (Dinamik Zamanlama Analizi)
ECO	:	Engineering Change Order (Mühendislik Değişiklik Talimatı)
EPLD	:	Erasable Programable Logic Device (Silinebilir Programlanabilen Mantık Devresi)
FPGA	:	Field Programmable Gate Array (Sahada Programlanabilen Kapı Dizisi)
GAL	:	Generic Array Logic (Genel Mantık Dizisi)
GPU	:	Graphics Processing Unit (Görüntü İşlem Birimi)
HDVL	:	Hardware Description and Verification Language (Donanım Tanımlama ve Doğrulama Dili)
HDL	:	Hardware Description Language (Donanım Tanımlama Dili)
HIL	:	Hardware in Loop (Döngüde donanım ile simülasyon)
HLS	:	High Level Synthesis (Yüksek seviyeli diller ile Sentezleme)
HVL	:	Hardware Verification Language (Donanım Doğrulama Dili)
LVS	:	Layout versus Schematic (Tasarım yerleşiminin şematik ile karşılaştırılması)
LAB	:	Logic Array Block (Mantık Dizisi Bloğu)

LASER	: Light Amplification by Stimulated Emission of Radiation (Işığın uyarılmış radyasyon salınımı ile kuvvetlendirilmesi)
LIDAR	: Laser Imaging, Detection and Ranging (Lazerle görüntüleme, tespit ve mesafe ölçümü)
LC	: Logic Cell (Mantık Hücresi)
LE	: Logic Element (Mantık Elemanı)
LUT	: Look Up Table (Taramalı Tablo)
MTBF	: Mean Time Between Failures (Arızalar Arası Ortalama Süre)
MLAB	: Memory LAB (Hafıza olarak tasarlanmış LAB)
MOS	: Metal Oxide Semiconductor (Metal-Oksit Yarıiletken)
MCM	: Multi Chip Modules (Çoklu Yonga Modülleri)
MUX	: Multiplexer (Çoğullayıcı)
MAC	: Multiply-Accumulate (Çarp Biriktir)
OVM	: Open Verification Methodology (Açık Doğrulama Metodolojisi)
PCIe	: Peripheral Component Interconnect Express (Çevre Birimleri Hızlı Bağlantı Veriyolu)
PMDC	: Permanent Magnet DC (Sabit Mıknatıslı DC)
PAL	: Programmable Array Logic (Programlanabilen Mantık Dizisi (sabit OR dizilimli))
PLL	: Phase Locked Loop (Faz Kilitli Döngü)
PLA	: Programmable Logic Array (Programlanabilen Mantık Dizisi (Programlanabilir OR dizilimli))
PLD	: Programmable Logic Devices (Programlanabilen Mantık Devresi)
PPR	: Pulses Per Revolution (Devir başına düşen sinyal)
RAM	: Random Access Memory (Rastgele Erişimli Bellek)
ROM	: Read Only Memory (Salt Okunur Bellek)
RTDS	: Real-Time Digital Simulators (Gerçek Zamanlı Sayısal Simülatör)
RISC	: Reduced Instruction Set Computer (İndirgenmiş Komut Takımlı Bilgisayar)
SPI	: Serial Peripheral Interface (Seri Çevresel Arabirim Bağlantısı)
SDF	: Standard Delay Format (Standart Gecikme Biçimi)
SRAM	: Static RAM (Durağan RAM)
STA	: Static Timing Analysis (Durağan Zamanlama Analizi)
SDC	: Synopsys Design Constraints (Synopsys Tasarım Kısıtları)
SoC	: System on Chip (Yonga Üzeri Sistem)
Tcl	: Tool Command Language (Aygıt Komut Dili)
UBGA	: Ultra FineLine BGA (Oldukça sık aralıklı BGA)
UVM	: Universal Verification Methodology (Evrensel Doğrulama Metodolojisi)
VHDL	: VHSIC HDL Very High-Speed Integrated Circuit HDL (Çok Yüksek Hızlı Entegre Devre Donanım Tanımlama Dili)

1. GİRİŞ

Simülátörler yıllardan beridir mühendislik sistemlerinin planlama, tasarım ve eğitim faaliyetlerinde yoğun olarak kullanılmaktadır [1]. Tasarım sürecinde bir sistemin unsurlarının yüksek maliyeti, testler sırasında maruz kalabileceği yıpratıcı süreçler veya tehlike arz eden durumlarda gerçek sistem veya belli kısımlarının yerine simülátör kullanmak kaçınılmaz bir gerekliliktir. Ayrıca işleyişin durdurulması veya kesintiye uğramasının göze alınamayacağı yapılarda, örnek olarak bir yonga üretim fabrikasının karmaşık yapısı sebebiyle fabrikanın iç etkileşimleri sadece simülasyonlar sayesinde ele alınabilmektedir [2].

Simülátör kelime anlamı olarak bir sistemin özelliklerinin veya işleyişinin, bir başka sistemin kullanımı veya işleyişi ile temsil edilmesidir [3]. Yukarıda bahsedilen sebepler gibi çeşitli problemlerden dolayı bir sistemin tamamının veya belli bir kısmının yerine simülátör kullanılarak işleyiş veya özelliklerin incelenmesi sağlanmaktadır.

Tarihte bu gereklilikler sebebiyle oluşturulmuş ilk sistemler incelendiğinde karşımıza gök cisimlerinin konum ve durumlarının hesaplanmaları için tasarlanmış makineler çıkmaktadır. Bu makineler ile ilgili eldeki ilk kayıt bir Roma devlet adamı olan *Marcus Tullius Cicero*'nun "*De Re Publica*" isimli eseridir. Eserde bu makinelerden iki tane bulunduğu ifade edilmekte ve bazı özellikleri tarif edilmektedir [4]. Buna karşın bu bilgi ancak 1900'lü yılların başında *Antikitera* mekanizmasının bulunması ile teyid edilebilmiştir. Yapılan incelemeler sonrasında Antikitera mekanizmasının yan tarafındaki bir kolun çevrilmesi vasıtası çalıştırılan mekanik bir takvim olduğu anlaşılmıştır [5]. Bu mekanik takvim, güneş ve ayın durum ve konumları ile birlikte o tarihte bilinen beş gezegen için konum göstergeleri bulundurmaktadır [6]. Antik dönemlerde imal edilmiş bu aygıt tarihte simülátör tanımına uyan ilk *Analog Bilgisayar* olarak tanımlanmaktadır [7, 8].

Sonraki dönemlerde analog mekanik bilgisayarların farklı örnekleri görülmeye başlanmış ve zamanla elektromekanik sistemlere doğru evrilmiştir [9]. Uzay çalışmalarının ilk başlarında Sovyet Soyuz uzay araçlarında mekanik ve elektromekanik bilgisayarlar çeşitli görevler için kullanılmıştır. Gelişim sürecinin sonraki aşamalarında tamamen elektronik analog bilgisayarlar kullanıma sunulmuştur. Elektronik analog bilgisayarlar eğitim, endüstriyel ve askeri amaçlarla simülátör olarak yoğun kullanım alanı bulmuştur.

Analog sistemlerdeki hassas kalibrasyon ihtiyacı ve bunun tekraren yapılmasının gerekliliği, kuantizasyon hatalarına karşın dijital sistemlerin kullanımına yol açmıştır. Dijital sistemlerin yaygınlaşması ve buna programlama vasıtası ile kullanılabilen bilgisayarlarda eklenince simülátör endüstrisinde dijital sistemler egemen olmaya başlamıştır. Özellikle 20.yy'ın son çeyreğinde bilgisayar endüstrisindeki muazzam gelişmeler, kullanıcıların daha üstün özellikli simülátörlere daha uygun ücretlerle ulaşabilmelerine olanak tanımıştır.

Simülasyon işlemleri, kullanılan teknik, çalıştırılan modelin kapsamı ve kapasitesi ile işleyiş hızı açısından çok farklı çeşitlerde uygulanabilmektedir. Örnek olarak sonlu elemanlar analizinde oldukça detaylı hesaplamalar için çok fazla işlem yükü gerekmektedir. Bunu yaparken sistemin statik bir durumunu veya dinamik bir sistemin belli bir süre zarfında incelenmek istenen özellik değişimlerini hesaplayabilmek için uzun süre işlem yapma ihtiyacı duymaktadır. Buna karşın diğer simülasyon işlemleri sonlu elemanlar analizi kadar kaynak açlığı çekmeyen daha farklı ihtiyaçlar için kullanılmaktadır.

Simülasyonda kullanılan sistem modeli ve hesaplamada kullanacak bilgisayar sisteminin kapasitesine bağlı olarak simülasyon değerleri gerçek fiziksel sistemle aynı zaman sürecinde gerçek zamanlı (*real-time*) gerçekleştirilebileceği gibi gerçek sistemden daha yavaş veya daha hızlı olarak da *off-line* çalıştırılabilmektedir [1]. Dijital simülatörler modelledikleri sistemin durumlarını ardışık adımlar şeklinde hesaplamaktadırlar. Bu adımların her birisi bir önceki adımda oluşan sonuç değerlerini kendisi için sınır değerler olarak kullanmaktadır. Giriş sinyalleri ise simülasyon adım aralıkları boyunca basamak fonksiyonu şeklinde ele alınmaktadır [10].

Simülatörde kullanılan modeller hesaplama sırasında sabit veya değişken zaman aralıklı çözüm teknikleri kullanabilmektedirler. Yüksek frekanslı sistemler ve lineer olmayan sistemlerin modellenmeleri sırasında değişken zaman aralıklı hesaplama metotları yoğun olarak kullanılmaktadır [11]. Değişken zaman aralıklı hesaplama yöntemleri, gerçek zamanlı çalışan simülatörler için uygun olmadığından bu sistemlerde sabit zaman aralıklı hesaplamalar kullanılmaktadır. Özellikle elektriksel devrelerin dijital modellerinin çıkarılması için endüktans ve kapasitans elemanlarının modellenmesi için Dommel tarafından geliştirilen yöntem yaygın şekilde kullanılmaktadır [12].

İmalat ve enerji endüstrisi başta olmak üzere askeri ve eğitim amaçlı sistemler, kontrol sistemlerinin tasarımı ve video oyunlarında gerçek zamanlı simülatörler yaygın şekilde kullanılmaktadır. Video oyunlarındaki simülasyon hızının kullanıcıyı rahatsız etmeyecek düzeyde olması yeterli olduğundan bilgisayardaki belli donanım birimleri ve yazılım kısımlarının iyileştirilmesi veya komple bir platform olarak ele alınmaları yoluna başvurulmaktadır. Endüstriyel ve askeri ihtiyaçların karşılanması sırasında simülatörün hızı farklı gerçek zamanlı sistemlerle birlikte çalışabilmek adına çok daha önem arz etmektedir. Bu tür ihtiyaçların giderilmesi için “Gerçek Zamanlı Dijital Simülatörler” (Real-Time Digital Simulator RTDS) geliştirilmiştir. RTDS üniteleri programlanabilir bilgisayarlar kullanılarak yapılabildikleri gibi özel amaçlı donanımlar kullanılarak da imal edilebilmektedir. Özellikle sadece belli bir sistemin simülasyonu için kullanılacak RTDS üniteleri fiyat ve performans açısından gömülü sistem (*eRTDS*) olarak üretilebilmektedir. Gerçek zamanlı sayısal simülatörler halihazırda güç sistemleri [13], mikro şebekeler [14], arıza tespiti [15], kontrol algoritmalarının geliştirilmesi [16] ve elektrik motorlarının simülasyonu [17] gibi bazı alanlarda yaygın şekilde kullanılmaktadırlar.

Özellikle yüksek maliyetli ve tehlikeli sistem senaryolarının RTDS ile test edilmesi oldukça önemlidir. Elektrik motorları ile ilgili olarak çalışırken zaman, iş güvenliği, ve

maliyet konularının düşünülmesi gerekmektedir. Elektrik motorlarının modelleri elektriksel ve mekanik kısımlar barındırdığından dolayı gerçek zamanlı simülasyonları için yüksek hızlı dijital sistemler kullanmak gerekmektedir.

RTDS kullanılarak yapılan çalışmalar arasında yüksek voltajlı enerji nakil hatları üzerinde yapılan çalışmalar [18, 19] ve bu hatlar üzerindeki rölelerin gerçek zamanlı simülasyonu için yapılan çalışmalar gösterilebilir [20]. Özel olarak üretilmiş RTDS birimleri haricinde PC kümeleri ile oluşturulan RTDS sistemi ile yapılan güç sistemleri simülasyonu çalışmaları da bulunmaktadır [21]. PC kümelerinin RTDS yapıları olarak kullanılmaları özel olarak üretilen RTDS ünitelerinin satın alınmalarının yerine mevcut bilgisayarların bu amaçla kullanılmaları için PC kümeleri şeklinde organize edilmeleri maliyet açısından avantajlar sağlamıştır. Güç sistemlerinin analizinde bu yöntem kullanılarak hem PC kümeleri hemde bu kümelerin ağ şeklinde birbirine bağlanması ile dağıtık yapıda simülatörler ile simülasyon çalışmaları yapılmıştır [22, 23]. OPAL-RT tarafından dağıtık süper bilgisayar birimleri ve giriş çıkış işlemleri için FPGA (Field Programmable Gate Array) “Sahada Programlanabilen Kapı Dizileri” kullanılarak yapılan çalışmada simülasyon işlemlerinin haricinde de kullanım alanı olan bilgisayarlar kullanılarak yüksek performanslı RTDS üniteleri oluşturulmuştur [24].

Simülasyon sistemlerinde gerçek sisteme ait bazı sensör yapılarının modellenmesi oldukça zor ve mevcut modelin karmaşıklığını arttıran bir unsurdur. Bu bazen simülasyonun gerçek zamanlı yapılmasını engelleyecek düzeyde bir karmaşıklık getirdiği gibi çoğu zaman bilinmeyen parametreleri sebebiyle modelleme bile sağlıklı şekilde yapılamamaktadır. Gerçek zamanlı simülasyonlarda bu problemleri aşabilmek adına “HIL” (Hardware in Loop) yöntemi geliştirilmiştir. Bu yöntemin temel fikri modellenmesi zor sensör yapıları gibi birimlerin imkan varsa doğrudan kendisini kullanmak veya yerlerine geçebilecek RTDS yapıları kullanmaktır [25].

HIL yöntemi sivil sektörde özellikle otomotiv endüstrisinde kullanılmaktadır. Buna örnek olarak turbo şarjlı bir dizel motorun kontrol sisteminin geliştirilmesi için oluşturulan HIL sistemi gösterilebilir [26]. HIL sistemler özellikle askeri uygulamalarda yoğun olarak kullanılmışlardır. Bu askeri uygulamalara; aktif füze simülasyonu [27], balistik füze savunma programı için önleme sistemi [28], füze komutanlığı için simülatör [29], aktif ve pasif milimetre dalga boyu ve kızılötesi sensörlü silah sistemleri için simülatör [30], çift mod kızılötesi ve radar simülatör birimleri [31], simülatörler için W-bandı radar noktasal hedef modelleri [32], aviyonik simülatörü [33] ve kızıl ötesi sahne projeksiyonlu, çifte kızıl ötesi sensörlü ve LIDAR (Laser Imaging Detection and Ranging) lazerle görüntüleme, tespit ve mesafe ölçüm sistemli uzaydan önleme sistemleri simülatörleri [34], örnekler olarak gösterilebilir. HIL sistemlerinin Eglin hava üstündeki 20 yıllık kullanımından dersler edinilmiştir [35]. Yine donanma test ve değerlendirme çalışmalarına HIL sistemleri ile katkı sağlanmıştır. [36]. Hera hedef roketleri programına HIL kullanılarak katkı sağlanmıştır [37]. Füze güdüm ve kontrol sistemlerinin gelişimine katkı sağlamıştır [38].

RTDS oluşturulurken sinyal işlemede μP , ASIC (Application Specific Integrated

Circuit “Uygulamaya Özel Entegre Devre”), DSP (Digital Signal Processing “Dijital Sinyal İşleme”) ve FPGA birimlerinden faydalanılabilmektedir. Bu sinyal işleyici birimlerin kendilerine göre farklı avantaj ve dez avantajları bulunmaktadır [39]. PC bazlı simülatörler gibi μP kullanılan RTDS yapılarını esneklik olarak daha fazla seçenek sunabilmektedir. ASIC, DSP veya FPGA kullanan RTDS yapıları esneklikten ödün verilerek hız ve performansa ağırlık verilen yapılardır. Bazı işlemler μP ile gerçek zamanda yapılamıyorsa aynı işlem ASIC, DSP veya FPGA birimleri kullanılarak hızlandırılabilir. FPGA yüksek hızlı cihaz elde etmek söz konusu olduğunda esneklikleri ve hızları dolayısıyla en çok tercih edilen cihazlardan biridir. RTDS yapılarında kullanılan FPGA birimleri donanımsal olarak sundukları esneklik ile tasarımcıları cezbetmektedir. FPGA entegreleri görüntü ve ses işleme, dijital modülasyon ve de-modülasyon, veri işleme, otomasyon sistemleri, iletişim sistemleri, hafızalar ve çevre birimleri arayüz oluşturma, robotik, savunma, havacılık ve uzay çalışma alanları ile medikal uygulamalarda geniş kullanım alanı bulan yapılardır [40]. Ayrıca FPGA ile RTDS oluşturma için OPAL-RT ve National Instruments gibi firmalar hazır kullanılabilecek yazılım ve donanım paketleri sunmaktadır.

Elektromanyetik geçici durum simülasyonunun yoğun işlem yükünü FPGA yongasına aktararak simülasyon hızının gerçek zamana yaklaştırıldığı çalışma FPGA kullanılarak simülasyon hızının artırılması çalışmasına bir örnektir [41]. Aynı ekip sonrasında FPGA tabanlı bir RTDS ünitesi oluşturmuştur [42]. FPGA tabanlı gerçek zamanlı donanımsal simülatörler daha sonra elektrik makineleri için de oluşturulmuş ve gerçek zamanlı olmayan ticari simülatör sonuçları ile kıyaslama yapılmıştır [43]. Elektrik makineleri için benzer bir simülatör OPAL-RT ürünleri aracılığı ile oluşturulup kontrolör tasarım ve testleri için kullanılmıştır [44]. Benzer şekilde sabit mıknatıslı senkron makinenin sonlu elemanlar analizi sonuçlarından elde edilen endüktans matrisi kullanılarak RT-LAB platformu yardımı ile oluşturulan RTDS sayesinde $1,5 \mu s$ gecikme değerine kadar ulaşılabilmiştir [44, 45]. Yine elektromanyetik geçici durum simülasyonunun gerçek zamanlı yapılabilmesi için oluşturulmuş ve 46 bitlik kayar nokta veri hatları kullanılan FPGA tabanlı RTDS uygulamasında fiyat performans olarak geleneksel PC bazlı sistemlerden daha ulaşılabilir bir simülatör oluşturulabilmiştir [46].

Fotovoltaik panelleri için güneş ışığının geliş açısı ve sıcaklık gibi parametrelerinin fiziksel modellerle kontrolünün zorluklarından dolayı FPGA tabanlı RTDS üniteleri kullanılarak panel simülatörü tasarlanmıştır [47]. FPGA tabanlı RTDS uygulamalarının avantajlarını Mahmoud Matar’ın çalışmasında görmek mümkündür. Bu çalışmada FPGA tabanlı simülatörün gerek ticari ürünler ve gerek dağıtık yapıya sahip kurumsal simülatörlere göre avantajları ortaya konulmuştur [48]. Benzer çalışma gerçek zamanlı güç elektroniği çevirici yapıları için de oluşturulmuştur [49]. Elektrik makineleri simülasyonu için FPGA tabanlı HIL sistemi oluşturulan başka bir çalışmada Modelica kütüphanelerinden çevrilerek oluşturulmuş yapı blokları ve NIOS çekirdeği kullanılarak elde edilmiş bir RTDS sistemin performansı incelenmiştir [50]. Güç elektroniği çeviricileri hakkında yapılan farklı çalışmalarda ADC (Associate Discrete Circuit) yaklaşımı

kullanılarak H köprüsü için modellemeler yapılmıştır [51, 52].

FPGA tabanlı RTDS yapılarına farklı örnekler olarak elektromanyetik geçici rejim analizi için frekansa bağlı faz domeyni iletim hattı modeli [53], veya geniş ölçekli güç sistemlerinin simülasyonu için FPGA tabanlı yardımcı simülasyon donanımı önerisi gösterilebilir [54]. 2018 yılında yapılan bir başka çalışmada elektromekanik geçici durum simülasyonu sisteminin FPGA tabanlı sayısal çözümleyici uygulaması yapılmıştır [55].

FPGA tabanlı gömülü RTDS uygulamaları hakkında Mohammed DAGBAGI tarafından yapılan çalışmada elektrik sistemlerinin simülasyonu için RT-LAB kullanılarak simülatör oluşturulmuş ve bu simülatör ile farklı sinyal bit uzunluklarında sistemin kararlılık durumundaki değişimler irdelenmiştir [56]. Farklı bir çalışmada FPGA tabanlı RTDS kullanılarak şebekeye bağlı PV sistemlerin kontrolü incelenmiştir [57]. Simülatörlerde kompleks sistemlerin modellenmesi sırasında modelin alt sistem bloklarına ayrılması sebebiyle bu bloklarda oluşan gecikmelerin bertaraf edilmesi için çalışmalar yapılmıştır [58]. Güncel yapılmış bir çalışmada HLS (High Level Synthesis) kullanılarak simülatör için yazılan C kodları donanım tanımlama diline çevrilerek donanım tasarlanmış ve modüler çok seviyeli çevirici için sensörsüz kontrolü için eRTDS oluşturulmuştur [59].

RTDS platformları ayrıca DT (Digital Twin) “Dijital İkiz” sistemleri olarak tanımlanmaktadır. Dijital ikiz kavramı ilk olarak 1997 yılında Hernández L. A. ve Hernández S. tarafından kullanılmıştır [60]. Buna ek olarak Michigan Üniversitesinden Dr. Grieves 2003 yılında bir fiziksel ürün için ilk kez bir dijital eşdeğerini oluşturan kişi olmuştur [61]. Nasa 2010 yılında yazdığı bir rapor ile bu kavramın yayılmasında “Dijital İkiz” terimini kullanarak katkıda bulunmuştur [62, 63]. Bu kavram ile ilgili çalışmalar 2011 yılından sonra hızla artmaya devam etmektedir. Gartner danışmanlık şirketinin araştırmalarına göre dijital ikiz kavramı endüstrideki ilk 10 stratejik teknoloji arasında olduğunu göstermiştir. Tahminler bu teknolojinin pazar payının 2023 yılına doğru 16 milyar dolara ulaşarak %35 ’ten daha fazla büyüyeceğini öngörmektedir [64, 65, 66].

Bu çalışmada amacımız hali hazırda RTDS birimleri veya bu birimleri oluşturacak yazılım ve donanım paketlerini sunan firmalara alternatif olarak doğrudan donanım geliştirme teknikleri kullanarak RTDS uygulamalarının yaygınlaşmasına bir ölçüde katkı sağlamaktır. Sistemlerin modellenmesi kısmında *Dommel* metodu yerine sistemin transfer fonksiyonunun doğrudan kullanımı amaçlanmaktadır. Sistemin sinyal işleme kısımlarında tek duyarlıklılık sayı gösterimi esas alınarak tam sayı tipi yerine daha hassas bir simülasyon için çaba sarf edilecektir.

Tezin Organizasyonu

Tezin ikinci bölümü bilgisayar sistemlerinde donanım geliştirme konusu ve bunun genel olarak FPGA üzerinde uygulanması hakkında olacaktır. Bu kısım asıl olarak RTDS tasarımında öncü firmaların piyasadaki mevcut donanım ve yazılım paketlerinin yerine araştırmacıların kendi sistemlerini tasarlarlarken genelde ihmal ettikleri bir konudur.

Üçüncü bölümde FPGA yongası haricinde eldeki mevcut donanım yapısı ve özellikleri

izah edilecektir. Bu bölümdeki donanım özellik ve kısıtları ışığında FPGA için donanım geliştirme işlemi uygulanacaktır.

Dördüncü bölümde Mevcut donanım yapısı ile etkileşime girecek FPGA modüllerinin tasarımı ve RTDS sistemi için gereken genel donanım modülleri izah edilecektir. Bu modüller dışında araştırmacılar yalnızca uygulamak istedikleri simülatörle ilgili hesaplama yapan modüllere yoğunlaşarak gereken tasarımlarını yapabileceklerdir.

Beşinci bölümde FPGA tabanlı RTDS için kullanılan modelleme blokları tanıtılacak ve bu donanım modüllerinin gerçek sistemle karşılaştırmalı testleri ile etkinlikleri incelenecektir. Tezin sonraki kısımlarında alınan deney verileri tartışılacak ve bu verilerden sonuçlar çıkarılarak sonraki çalışmalar için önerilerde bulunulacaktır.



2. DONANIM GELİŞTİRME

Bilgisayar sistemlerinde ürün geliştirme genel olarak yazılım ve donanım geliştirme olarak kabaca iki temel farklı kulvardan oluşmaktadır. Bu iki alan belirli konularda birbirlerinden oldukça farklı geliştirme süreçlerini içermektedir. Özellikle donanım geliştirme süreci günümüzde belirli firma ve grupların çalıştığı kendine has bir alan haline gelmiştir. Özellikle yonga tasarımı ve üretimi yüksek maliyet, uzmanlık ve tecrübe gerektirdiği için dünya genelinde bu alanda hizmet veren oldukça kısıtlı sayıda firma bulunmaktadır. Yonga imalatının kısıtlayıcı ve maliyetli süreçleri sayesinde neredeyse dünyanın belirli köşelerinde sıkışıp kalan donanım üretimi tesisleri astronomik ilk yatırım maliyetleri sebebiyle hızlıca yaygınlaşamamaktadır. Merkezi işlem birimi (Central Processing Unit CPU) veya grafik işlem birimi (Graphics Processing Unit GPU) gibi yongaların tasarımından üretimine ve paketlenmesine kadar geçen süreci tek başına yürütebilen intel® firması hariç diğer tüm firmalar bu sürecin belli kısımlarına odaklanarak yonga üretiminde farklı firmalar ile birlikte çalışmak zorunda kalmaktadır.

2.1. Donanım Geliştirmenin Temelleri

Donanım geliştirme süreci yazılım geliştirme sürecinde bulunan ön yüz ve arka yüz geliştirme terimlerine benzer şekilde mantıksal tasarım ve fiziksel tasarım kısımlarından oluşmaktadır [67].

2.1.1. Mantıksal Tasarım

Mantıksal tasarım aşamasında yonga mimarisi üzerinde durularak gerçekleştirilecek fonksiyonlar, kullanılacak güç ve kabaca zamanlamalar üzerinde kararlar alınır. Yonganın gerçekleştireceği işlev ve fonksiyonlar (Hardware Description Language HDL) donanım tanımlama dillerinden faydalanılarak mantık devrelerine ve bloklara aktarılır. Donanım tanımlama işlemi için en yaygın olarak;

- VHDL; VHSIC HDL Very High-Speed Integrated Circuit HDL (Çok yüksek hızlı entegre devre donanım tanımlama dili)
- Verilog; Verilog HDL

dilleri kullanılmaktadır.

VHDL donanım tanımlama dili Birleşik Devletler Savunma Bakanlığı tarafından geliştirilmiştir. Ülkenin savunma ihalelerinde kullanımı zorunlu tutulduğu için askeri uygulamalarda yaygın olarak kullanım alanı bulan bir dildir [68].

Verilog® donanım tanımlama dili 1983/84 yıllarında tescilli bir ürün olarak simülasyon ve doğrulama işlemlerinde kullanılmak üzere geliştirilmiştir [69]. Donanım tanımlama dilleri arasında en yaygın kullanılan dil olmasının yanında *SystemVerilog* diline de öncülük etmiştir. Bu diller dışında farklı donanım tanımlama dilleri de bulunmakla

birlikte kullanımları yaygınlık kazanmamıştır. Bununla birlikte FPGA üreticilerinden Altera firması kendi donanım tanımlama dili olarak AHDL (Altera HDL) donanım tanımlama dilini oluşturmuş ve uygulama kütüphanelerinde kullanmıştır. Daha sonradan intel® firması tarafından satın alınan firmanın Quartus donanım geliştirme platformu ile yapılan projelerde kullanılan fikri mülkiyet (Intellectual Property IP) bloklarında çoğu kod kısımları halen AHDL dilinde kalmaya devam etmektedir. Verilog diline benzer yapısı ve donanıma daha yakın bir dil olmasıyla bilinmektedir.

Kullanılan donanım tanımlama dilinden sentezleme yardımı ile sistem, mantık kapıları düzeyine kadar detaylandırılır. Mantık kapılarına kadar detaylandırılmış sistemin fonksiyon testlerinin yapılabilmesi için tasarıma test sinyallerinin uygulanabileceği ek girişler eklenir ve bir bakıma test için tasarım yeniden düzenlenir. Test için yeniden düzenlenen sisteme fonksiyon testleri uygulanarak sistemin istenen fonksiyonları yerine getirip getirmediği incelenmektedir. Bu amaçla OVM (Open Verification Methodology) ve UVM (Universal Verification Methodology) gibi doğrulama metotları kullanılmaktadır.

Donanım tanımlanması kısmında kullanılan HDL donanım tanımlama dillerine benzer olarak HVL (Hardware Verification Language) donanım doğrulama dilleri kullanılmaktadır. *SystemVerilog* dili HDVL (Hardware Description and Verification Language) donanım tanımlama ve doğrulama dili gibi kullanılan, gerek HDL gerekse HVL olarak işlev görebilen ve donanım doğrulama süreçlerinde endüstride yaygın olarak kullanılan bir dildir.

Tasarımın doğru çalıştığı anlaşıldıktan sonra devrenin belirlenen bir frekansta çalışıp çalışmayacağını incelemek için zamanlama analizi yapılır. Zamanlama analizleri iki tipte uygulanmaktadır;

- DTA (Dynamic Timing Analysis) Dinamik zamanlama analizi
- STA (Static Timing Analysis) Statik zamanlama analizi

DTA dinamik yapısı itibarı ile sisteme farklı test vektörleri (giriş sinyal grupları) uygulayarak sistemin çalışmasını incelemektedir. Gerektirdiği test sayısı ve test vektörü sayısı sebebiyle kullanımı oldukça zor bir test olduğundan kısıtlı sayıda uygulamalarda kullanılmaktadır.

STA temelde giriş vektörlerine bağlı olmadan incelediği sistemdeki her bir veri ve saat yolunun olabilecek en kötü senaryo durumuna göre zamanlamalarını incelemektedir [70]. İnceleme hızı ve performansı yüksek olduğundan zamanlama analizi uygulamalarında yoğun olarak kullanılmaktadır.

Zamanlama analizi, yonga tasarım süreçlerinin hem mantıksal hem de fiziksel kısımlarının neredeyse her aşamasında gerek direktifler vermek gerekse kısıtlara uyumu tespit etmek amaçlı olarak yapılmaktadır. Mantıksal tasarım aşamasında zamanlama analizleri ideal saat işaretleri kullanılarak ancak belli bir hassasiyetle yapılabilmektedir. Tasarım sürecinin bitmesi ve üretici firmaya gerekli dosyaların gönderilmesi için mutlaka zamanlama analizi süreçlerinden onay almalıdır.

2.1.2. Fiziksel Tasarım

Fiziksel tasarım sürecine ulaşmış bir tasarımda mantıksal fonksiyonlar gerektiği gibi çalıştıkları bilinmektedir. Ayrıca bunu yapacak bloklar arasındaki zamanlama kısıtlarında sistemin çalışması için gereken sayısal süre değerleri bilinmektedir. Bu bilgiler yardımı ile sistemin yonga üzerinde yerleşimi ve ara bağlantı yolları belirlenmektedir. Oluşan yerleşim ve bağlantı yollarına bakılarak LVS (Layout versus Schematic) ve DRC (Design Rule Check) testleri ile hem yerleştirilen sistemin şematik ile uyumu hem de tasarım kurallarına uyup uymadığı incelenir. Bu testlerden sonra blokların yerleşimi, barındırdıkları giriş ve çıkış devre katları ile bağlantı yolları belirlenmektedir. Bu oluşan bilgiler ışığında daha kesin rakamlar kullanılarak hesaplanan gecikmiş gerçek saat işaretleri hesaba katılarak zamanlama analizi yeniden yapılmaktadır. Sistemin çalışabilirliği onay aldıktan sonra tasarım üreticiye gönderilmeye hazır hale getirilir.

Geliştirme ve tasarım süreçlerinin farklı aşamalarında ECO (Engineering Change Order) direktifleri ile tasarıma müdahale edilebilmektedir. Ancak, tape-out işlemi olarak bilinen tasarım bilgi ve fotomaskeleri yonga üretim tesisine gönderildikten sonra ECO uygulamak oldukça zahmetli ve hatalara açık bir işlem olduğundan oldukça sınırlı şekilde yapılabilmektedir.

2.1.3. Yonga Üretimi ve Paketlenmeleri

Fabless firmalar olarak bilinen şirketler yonga üretiminin tasarım aşamasını gerçekleştirerek, kalan diğer aşamaları farklı firmalar üzerinden yaptırmaktadırlar. Tasarım aşamasında sistemin doğrulama işlemlerini hizmet alımı ile yapan firmalar da bulunmaktadır. Bunun dışında üretimin fiziksel kısmını gerçekleştiren *Foundry* firmalar tasarım dosyalarını kullanarak silikon plaka disk “wafer” yarı iletken malzemeler üzerinde yongaları imal etmektedirler. Üretilen yongaların testleri ve paketlenme işlemleri de farklı firmalar tarafından yapılabilmektedir.

Günümüzde yonga çipleri tek başlarına paketlenirken *interposer* olarak tabir edilen temel yongaları üzerinde birden fazla yonganın tek modül olarak paketlenmesi daha ileri paketleme teknolojileri kullanılarak MCM (Multi Chip Modules) birimleri de oluşturulabilmektedirler. Özellikle alan sıkıntısı duyulan akıllı saatler ve bazı mobil cihazlarda bu paketleme teknikleri sayesinde çok daha etkili bir alan kullanım imkanı sunulmaktadır. Ancak bu paketleme tekniğinde *interposer* yongaları üzerine yerleştirilecek yongaların hassas hizalanma gereksinimi ancak sayılı firmalar tarafından sağlanabilmektedir. *Interposer* yongaları hem aktif hem de pasif olarak imal edilebilmekte ve 2D, 2.5D veya 3D teknolojiler kullanılarak üzerilerine yongalar yerleştirilebilmektedir. Xilinx firması tarafından tasarlanan Virtex-7 T serisi XC7V2000T gibi bazı büyük ölçekli FPGA modülleri bu şekilde üretilmektedirler.

Çoklu yongaların tek *interposer* ünitesi üzerinde birleştirilebilme imkanı sayesinde aynı işlevleri gerçekleştirecek boyutta tek bir yonga üretiminden kaçınılabilmektedir. Büyük boyutlu yongaların üretimi sırasında wafer üzerlerinde oluşabilecek hataların bir yongayı

işlevsiz hale getirme olasılığı daha da artmaktadır. Aynı işlevlerin daha küçük yongalara bölünmesi ve imalattan sonra birlikte paketlenmeleri, yonga üretiminde hatalı yongalar sebebi ile kaybedilen wafer yüzeyini azaltmakta ve ekonomik olarak avantaj sağlamaktadır.

2.2. FPGA ile Donanım Geliştirme

Donanım geliştirme ve üretimi süreçlerinin zorlukları ve özellikle üretim sürecine girmiş bir ürün için ECO uygulamanın zorlukları hatta imkansızlıkları sebebiyle üretildikten sonra programlanabilen yonga türlerine ihtiyaç duyulmuştur. Bu amaçla farklı ölçekli ihtiyaçlar için kullanılan yapılar aşağıda listelenmiştir.

- ROM (Read Only Memory)
- PLD (Programmable Logic Devices)
 - PLA (Programmable Logic Array)
 - PAL (Programmable Array Logic)
 - GAL (Generic Array Logic)
 - EPLD (Erasable Programmable Logic Device)
 - CPLD (Complex Programmable Logic Device)
- FPGA (Field Programmable Gate Array)

Bu yapılardan CPLD ve FPGA yongalarının kullanımları diğerlerini gölgeleyerek baskın hale gelmiştir. CPLD yapıları aslında PLD yapılarının birleştirilmesinden oluşturulmuş yongalar olarak daha ziyade bileşimli mantık yapıları gerçekleştirmek amacıyla kullanılmaktadır. FPGA yongaları ile LUT (Look Up Table) yapılarını temel alan ve flip-flop barındıran yapılar olarak hem bileşimli hem de sıralı çalışan mantık yapıları oluşturmak için kullanılmaktadır.

Günümüzde en yaygın şekilde kullanımı süren yapılar FPGA yongalarıdır. Hem donanım tasarımında sınama amacıyla hem de son üründe kullanılmak üzere en yoğun kullanımı bulunan programlanabilir mantık yonga tipi olarak karşımıza çıkmaktadır.

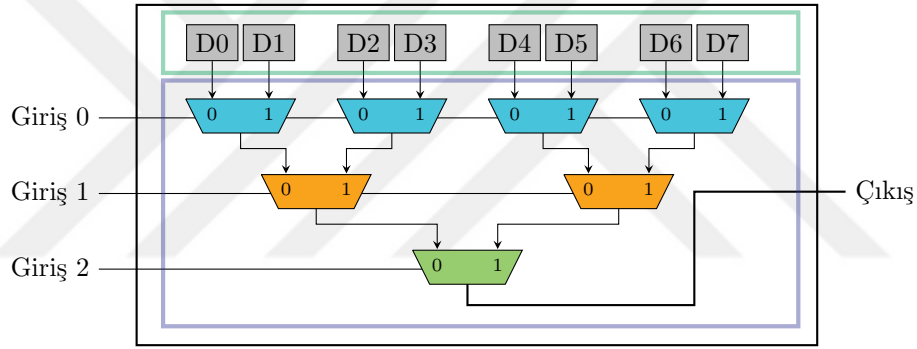
FPGA tasarımları her ne kadar ASIC (Application Specific Integrated Circuit), ASSP (Application Specific Standard Product) veya CPU sistemlerinin hızlarına ve enerji verimliliğine ulaşamamaları bile gerek tasarım sırasında sundukları esneklik gerek üretim hattında bile ECO uygulanabilmesini sağlamaktadır. Ayrıca ASIC yonga üretimini mali olarak karşılayamayacak projeler için donanım tasarım ve uygulama imkanı sunmaktadır.

Genel kullanım alanı bulan fonksiyonlar için düzenlenen uluslararası yarışmalarda tasarımcılar FPGA üzerinde en etkili donanım tasarımları için yarışmakta ve ödül alan tasarımların ASIC veya ASSP yongalarına aktarımı yapılabilmektedir. Bu hali ile şirketler çalışanları dışındaki geliştirici kitlelerinden de donanım tasarımı konusunda faydalanabilmektedirler.

Sağladıkları imkanlara rağmen FPGA ile donanım tasarımı halen klasik anlamdaki donanım tasarımı sürecine belli ek olanaklar ve kısıtlamalar ile dahildirler. Geliştiriciler tasarlayacakları donanımları HDL ile kodlayarak zamanlama analizlerini yapmak ve sonrasında da FPGA yongasına aktararak denemek durumundadırlar.

2.2.1. FPGA Yongalarının Genel Yapısı

FPGA yongaları CPLD yongalarından farklı olarak LUT yapılarını kullanırlar, LUT yapılarında hafıza birimi olarak SRAM (Static RAM) hücreleri kullanılmaktadır. Bu hücreler birimin yapılandırılacağı zaman dışarıdan erişilerek programlanabildiği gibi bazı modellerde flash hafıza bulunur ve yapılandırma bilgileri buradan alınmaktadır. LUT yapıları ile devre tasarlandığında devrenin çıkış ifadesine göre MUX (Multiplexer) girişlerine uygulanan sinyaller ile çıkışa bağlanacak SRAM hücresi için veri girilmektedir. Şekil 2.1.'de görülen 3 girişli LUT yapısında SRAM hücreleri MUX yapıları ile girişe uygulanan sinyale göre çıkışa aktarılmaktadır.



Şekil 2.1. 3 girişli Bir LUT yapısı ve SRAM hücreleri

Yapılandırma bilgileri LUT yapıları içine kaydedilecek veriyi ve FPGA yongası içindeki hücrelerin birbirleri ile kuracakları bağlantıları ve varsa bazı sabit donanım kısımları için konfigürasyon bilgisi içermektedir. LUT yapılarında giriş terminal sayısı arttıkça iç yapıları da üssel olarak artmaktadır. Bu durum sebebiyle FPGA yapılarında giriş sinyalleri 6, 5 ve 4 girişli sabit LUT hücrelerine farklı şekillerde dağıtılarak fonksiyon kısıtlamasına gidilmektedir. Bu şekilde birim içi eleman sayısı kabul edilebilir düzeylerde tutulmaya çalışılmaktadır.

FPGA yongalarının temel birimleri farklı tescilli isimler ile anılmaktadır. Örneğin Altera firması temel hücrelerini LE (Logic Element) adlandırırken Xilinx firması kendi temel hücre yapılarını LC (Logic Cell) olarak adlandırmaktadır. Bu isimlendirmeler temel alınsa bile tasarımlarda etkinlik sağlanabilmesi için firmalar normal birimlerinden farklı özelliklerde birimlerde kullanılmaktadırlar.

Temel birimler giriş kısımlarında LUT yapıları barındırmakla birlikte iç yapılarında D tipi yazmaçlar da barındırmaktadırlar. Bunların yanı sıra temel birimlerde toplama birimleri

de bulunmakta ve yan taraflarındaki birimlerle bağlanarak elde zinciri oluşturmak için ek giriş ve çıkış terminalleri barındırmaktadırlar.

Temel birimler dışında ayrıca farklı sayı ve çeşitte LUT, aritmetik toplama ve kaydediciler kullanılarak ALM (Adaptive Logic Module) gibi daha gelişmiş modüllerde kullanılmaktadır. Cyclone V entegrelerinde 10 ALM yapısı birleştirilerek LAB (Logic Array Block) gibi daha üst mantık birimlerini oluşturmaktadırlar.

Bahsedilen temel mantık birimleri ve grupları dışında yonga içinde ayrıca DSP (Digital Signal Processing) blokları ve iç yapıda kullanılabilecek şekilde dağıtılmış SRAM blokları da bulunmaktadır.

İç yapı blokları dışında FPGA yongalarında bulunan ve belli amaçlar için özelleşmiş bloklar da bulunmaktadır. Özelleşmiş bloklardan en yaygın kullanılanları;

- PLL (Phase Locked Loop) Faz kilitli döngü
- DLL (Delay Locked Loop) Gecikme kilitli döngü
- Harici hafıza arayüzleri
- Dahili ADC modülleri
- Yüksek hızlı bağlantı arayüzleri
- RISC mimarisinde sabit donanımsal işlemciler

şeklinde sıralanabilmektedir. Sık kullanılan bu bloklardan farklı olarak bazı üreticiler FPGA yongalarına farklı endüstri ve teknik uygulamalarda özelleşmiş ek donanım birimleri ile donatmaktadırlar. FPGA yongaları ayrıca giriş ve çıkışlar için farklı gerilimler ile beslenebilen portlar da bulundurmaktadırlar. Bu giriş çıkış terminalleri sayesinde aynı FPGA yongası 2.8 V, 3.3V v.b. gibi farklı gerilimlerde çalışan çevre entegre ve donanımlara doğrudan bağlanabilmektedir.

2.2.2. Donanım Tanımlama Dilleri

Donanım tasarımı günümüzde çizim yolu ile tasarlanamayacak derecede karmaşık ve hacimli bir duruma gelmiştir. Moore, yasası ile kabaca her iki senede yonga başına transistör sayısının ikiye katlanacağına dair bir gözlemde bulunmuştur [71]. Katlanarak artan transistör sayısı ile birlikte mühendis başına düşen sorumlu olunan transistör sayısı da artmaktadır. Tasarımcı mühendis başına artan bu transistör sayısı klasik tasarım yöntemlerinin dışında donanım tasarımında donanım tanımlama dillerinin kullanımını zorunlu kılmaktadır.

Daha önce Bölüm 2.1.1.'de bahsedilen dillerden, sivil endüstride en çok kullanılan donanım tanımlama dili Verilog olarak karşımıza çıkmaktadır. Günümüzde donanım tanımlama dilleri lisans düzeyinde üniversitelerde sayısal elektronik derslerinde gösterilmektedir.

Endüstride çalışırken yazılan bir Verilog kodunun mümkün olduğunca taşınabilir ve yeniden farklı parametreler ile kullanılabilir olması talep edilmektedir. Bu amaçla günümüzde Verilog modülleri yazılırken iç yapının uygun şekilde düzenlenebileceği *Parametre* değerleri ile birlikte kodlanmaktadır. Parametre kodları ilgili Verilog kodu her çağrıldığında yeniden düzenlenebilmekte, bu sayede işlevler, giriş çıkış veri boyutları ana kodlama bölümlerine dokunulmaksızın kontrol edilebilmektedir.

Parametre değerleri dışında Verilog modül kodlamaları içinde kalan kısımda *localparam* adlandırması ile bölgesel parametrelerde kullanılmaktadır. Bölgesel parametreler kod blokları çağrılırken değiştirilemeyen ancak modül içinden değiştirilebilen parametrelerdir.

C dilinde ön-derleyici bildirelerine benzer şekilde Verilog kodlamasında *sabit fonksiyonlar* olarak adlandırılan fonksiyonlar sayesinde donanım tasarımı kodları dışında sentezleme işlemlerinden önce yapılması istenen ön hesaplama işlemleri yapılabilmektedir. Bunlara bir örnek olarak modül girişine uygulanabilecek en büyük sayıya karşılık modül girişindeki veri hattının bit sayısının hesaplanması için gerekli logaritma işlemleri ve diğer çeşitli işlemler gösterilebilmektedir.

Tasarımcının kendi yazdığı kodlamalar dışında yongada bulunan çeşitli donanım bölümlerinin kullanımını sağlayan IP kütüphaneleri de kontrollerini yazılı oldukları donanım tanımlama dillerinde parametrik olarak yapmaktadırlar. Bazı IP blokları yazıldıkları dönemde kullanılan tanımlama dilindeki kodları halen barındırmaktadırlar. Buna bir örnek olarak Quartus Prime ile kullanılan bazı IP blokların kodlarının AHDL dilinde olmasıdır.

FPGA geliştirme sırasında tasarımcılar istedikleri dilleri birlikte kullanabildikleri gibi şematik dosyaları kullanılarak da sistemi tasarlayabilmektedirler. Bir tasarım dosyası farklı dillerdeki kod bloklarını kullanabilmektedir. Ancak her bir kaynak kodu parçasının yalnızca tek bir dilde olması gerekmektedir. Farklı diller kullanmak bir avantaj gibi görünmekle birlikte kodlama işlemlerinde hatalar yapmaya da sebep olabilmektedir.

Project Navigator						
IP Components						
	Entity	IP Component	Version	Supported Device Families	IP File	Vendor
✓	clk_pll	PLL Intel FPGA IP v18.1	18.1	Cyclone V	clk_pll/clk_pll.qip	Altera
✓	cnvrt_fx2fp_vs	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fx2fp_vs.qip	Altera
✓	cnvrt_fx2fp_tl	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fx2fp_tl.qip	Altera
✓	cnvrt_fx2fp_ra	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fx2fp_ra.qip	Altera
✓	cnvrt_fx2fp_la	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fx2fp_la.qip	Altera
✓	cnvrt_fx2fp_bm	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fx2fp_bm.qip	Altera
✓	cnvrt_fx2fp_jm	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fx2fp_jm.qip	Altera
✓	cnvrt_fx2fp_km	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fx2fp_km.qip	Altera
✓	div_flt	ALTFP_DIV	18.1	Cyclone V	smltr/ftlpt/div_flt.qip	Altera
✓	mlt_flt	ALTFP_MULT	18.1	Cyclone V	smltr/ftlpt/mlt_flt.qip	Altera
✓	add_sub_flt	ALTFP_ADD_SUB	18.1	Cyclone V	smltr/ftlpt/add_sub_flt.qip	Altera
✓	cnvrt_fp2fx_vs	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fp2fx_vs.qip	Altera
✓	cnvrt_fp2fx_i	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fp2fx_i.qip	Altera
✓	cnvrt_fp2fx_w	ALTFP_CONVERT	18.1	Cyclone V	smltr/ftlpt/cnvrt_fp2fx_w.qip	Altera
✓	lpm_div_int	LPM_DIVIDE	18.1	Cyclone V	extdcm/lpm_div_int.qip	Altera

Şekil 2.2. Tasarımda kullanılan IP blokları listesi

Donanım tasarımı sırasında sağlayıcıdan temin edilerek kullanılan IP blokları tasarımı büyük ölçüde kolaylaştırmaktadır. Tasarımımızda da üretici firmanın sağladığı PLL yapılandırma blokları ve kayar-nokta işlemleri için hazırlanan IP bloklarından ve dönüştürücü bloklardan kullanıldığı Şekil 2.2.'de görülmektedir.

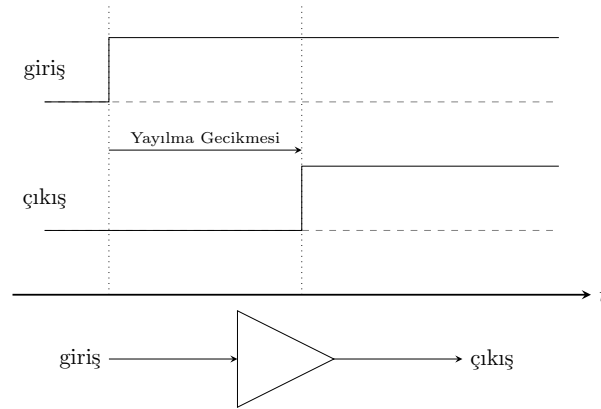
IP blokları, FPGA yongasının değişmesi veya geliştirme platformunun farklı bir versiyonunda sentezlenmesi gerektiğinde yeniden ele alınarak gözden geçirilmesi gerekmektedir. Intel Quartus Prime geliştirme platformu, IP blokları barındıran projelerin üst versiyon yazılımlarda açılması durumunda otomatik olarak çalışan bir IP güncelleyici yazılımı bulundurmaktadır.

2.3. Donanım Geliştirme İşlemlerinde Zamanlama

Zamanlama konusu, donanım geliştirme işlemlerinde donanım tanımlama ile birlikte en temel ve önemli iki konuyu oluşturmaktadır. Donanımı tanımlamak kadar onun düzgün şekilde çalışmasını sağlamak da tasarım işleminin temel işlevidir. Zamanlama analizinde endüstride hakimiyeti olan teknik, statik (durağan) zamanlama analizi olarak karşımıza çıkmaktadır.

2.3.1. Mantık Devrelerinde Gecikme

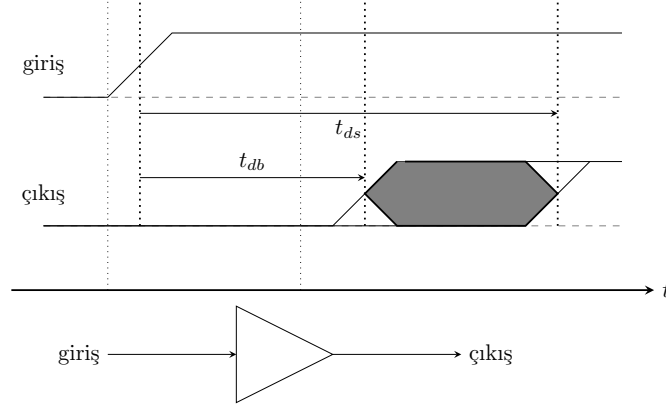
Yayılma gecikmesi mantık devrelerinde zamanlama konusunda karşımıza ilk çıkan terim olmaktadır. Şekil 2.3.'te görüldüğü gibi bir devrenin girişine uygulanan sinyalin sonucunun devrenin çıkışına ulaşması için gereken süre yayılma gecikmesi olarak anılmaktadır.



Şekil 2.3. İdeal yayılma gecikmesi

Elbette giriş ve çıkış için sürücü gücü, hat uzunluğu ve hat üzerindeki yüklenmeler göz önünde bulundurulduğunda sinyallerin yükselme ve düşüş hızları da belirginleşmektedir. Bu ek gecikme değerlerinin yanında incelenen devrenin çıkışında önceki sinyal değerinin kaybolmaya başlayıp çıkışın değişmeye başladığı zaman t_{db} ve

devre çıkışındaki değişim sürecinin bitip çıkışta kararlı bir sinyal olduğu ilk zaman t_{ds} Şekil 2.4.'ten görülebilmektedir.



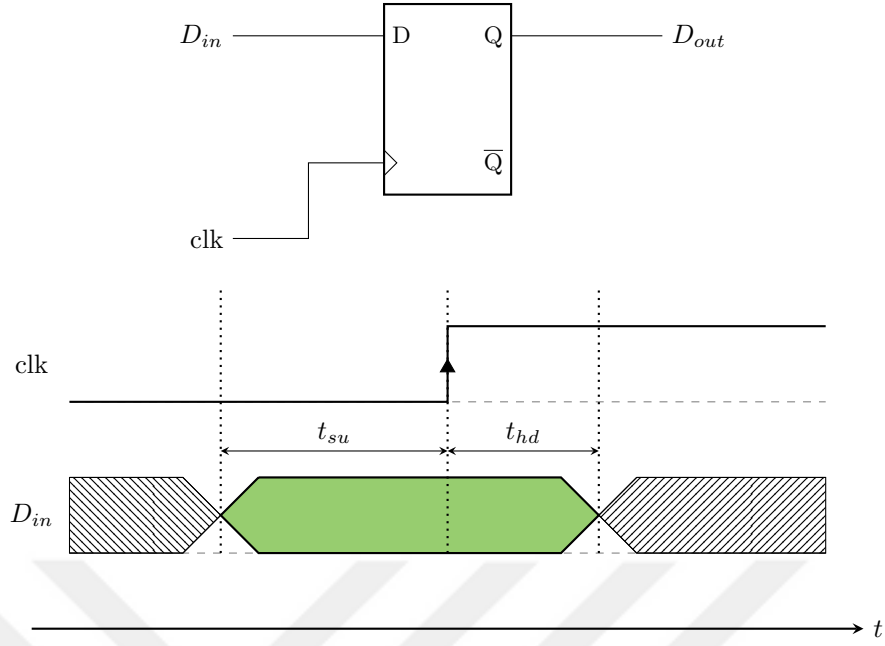
Şekil 2.4. İdeal olmayan yayılma gecikmesi

Üretici firmalar sağladıkları kütüphanelerde mantık birimlerinin gecikme değerlerini giriş sinyalinin yükselme hızı ve çıkış katının kapasitif yükünü baz alan tablolar ile sunmaktadırlar. Giriş sinyalinin değişim hızının artması veya çıkış kapasitif yükünün azalması devrenin hızını arttırmakta tersi durumlar ise azaltmaktadır. Zamanlama hesaplamalarında elemanların gecikmeleri SDF (Standard Delay Format) dosya biçimi ile kaydedilerek kullanılmaktadır.

2.3.2. Kaydediciler için Temel Zamanlama Terimleri

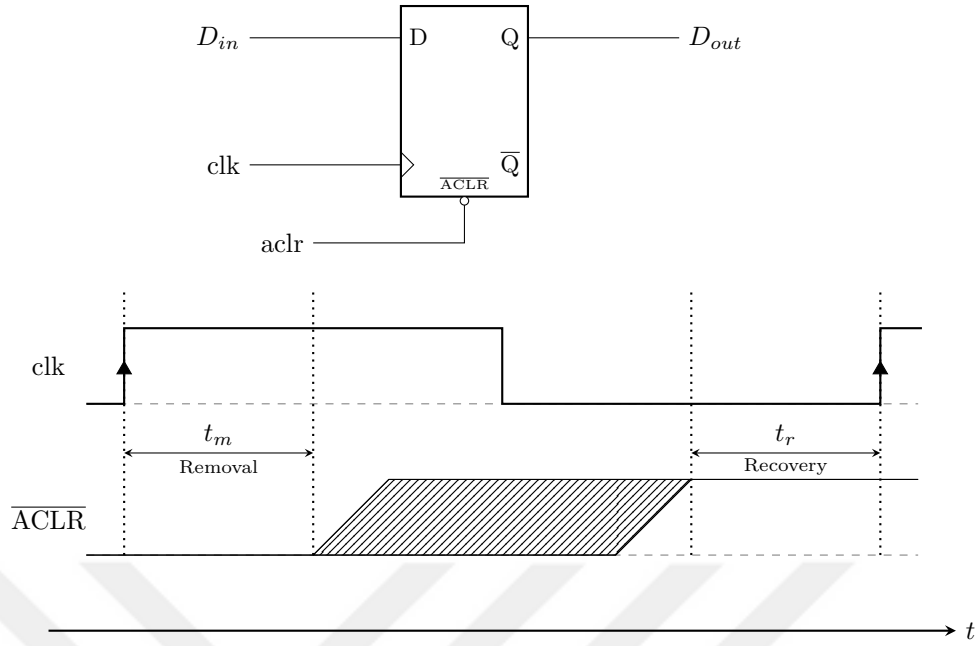
Donanım tasarımlarında çok sık şekilde kullanılan D tipi kaydedicilerin giriş ve çıkışlarında beklenen veya talep edilen sinyal gecikme ve stabil kalma zamanları zamanlama hesaplamalarının önemli bir bölümünü oluşturmaktadır.

D tipi kaydedicilerin giriş sinyalini veri girişinden alıp çıkışa aktardıkları saat işaretinin aktif kenarı ve bu işaretin belli bir süre öncesi ve sonrasında giriş sinyalinin stabil olması gerekmektedir. Şekil 2.5.'te gösterildiği üzere, D tipi kaydedicinin saat işaretinin aktif kenarından t_{su} (*setup time*, kurma zamanı) süresi kadar öncesinden başlayarak giriş sinyalinin sabit uygulanıyor olması gerekmektedir. Benzer durum aktif saat işaretinin t_h (*hold time*, tutma zamanı) süresi kadar sonrasına dek geçerlidir. Bu zaman bölgesi içerisinde giriş sinyalinin sabit olmaması kaydedicinin istenen çıkış değerine ulaşamaması ile sonuçlanabilmektedir. Bu durum mantık devresinin lojik 0 ve lojik 1 değerleri için kabul edilen gerilim değerleri arasında bir gerilimde kalarak *metastability* olarak adlandırılan yarı kararlı bir durumda kalması ile sonuçlanmaktadır. Böylesi bir durumda donanımın bu kaydedici çıkışından bilgi kullanan kısımlarının kararlılığını olumsuz yönde etkileyip devrenin istenen işlevi gerçekleştirmesine engel olacaktır.



Şekil 2.5. D tipi kaydedicilerin veri girişi için zamanlama gereksinimleri

Herhangi bir kaydedici için t_{su} süresi veya t_{hd} sürelerinin ilgili oldukları saat kenarı aynı veya farklı olabileceği gibi değerlerden birisi eksi işaretli de olabilmektedir. Negatif işaretli zaman değeri t_{su} süresi için verilmiş ise, bu kaydedicinin aktif saat işaretinden verilen negatif değer kadar sonrasında veri kabulüne başlayacağı anlamına gelmektedir. Benzer şekilde negatif işaretli t_{hd} süresi ise kaydedicinin aktif saat işaretinden belirtilen süre öncesinden itibaren veri kabulünü sonlandırdığı belirtilmektedir.



Şekil 2.6. D tipi kaydedicilerin asenkron reset girişleri için zamanlama gereksinimleri

Veri giriş bacağındaki zamanlama gereksinimlerine benzer şekilde, şayet kullanılıyorsa asenkron resetleme girişleri için de Şekil 2.6.'da görüldüğü gibi zamanlama gereksinimleri bulunmaktadır. Kaydedicinin reset durumundan sağlıklı şekilde kurtulmuş olması için saat işaretinin aktif kenarından t_r (*recovery time* “toparlanma zamanı”) kadar süre önceden reset işaretinin kalkmış olması gerekmektedir. Senkron olmayan resetleme girişinin sağlıklı resetleme yapabilmiş olması için uygulandığı süre boyunca oluşan tüm aktif saat kenarlarının etkisini gidermiş olması gerekmektedir. Reset sinyali süresi içinde kalan en son aktif saat kenarının etkisinin sağlıklı şekilde giderilebilmiş olması için ilgili aktif saat kenarından t_m (*removal time* “giderme zamanı”) süre sonraya kadar resetleme sinyalinin aktif olması gerekmektedir.

Sistemin doğru çalışabilmesi için resetleme sinyalinin de senkron olması gerekmektedir. Asenkron resetleme sinyalinin kullanılması durumunda; resetleme başlangıcı önemsiz olurken reset sinyalinin sonlanma kısmının senkron olması gerekmektedir. Aksi durumda resetleme işleminin kendisi sistemin düzgün çalışmasına engel olmaktadır.

Tasarımın herhangi bir kısmındaki iki kaydedici arasındaki devrenin gereken zamanda işlemlerini bitirip oluşturduğu değerlerin sonraki kaydedicide uygun zamanda mandallandığını hesaplayarak göstermemiz gerekmektedir. Bir FPGA yongasında iç yapı blokları sabit olduklarında devrelerin gerektirdiği işlem zamanı verileri geliştirme platformu yazılımında zaten bulunmaktadır. FPGA içinde kalan tasarım kısıtlamaları şayet kaydedici saatleri aynı ise doğrudan kontrol edilebilmektedir.

İki kaydedici arasındaki zamanlama aşağıda belirtilen iki farklı zaman payının

hesaplanması ile yapılmaktadır.

- Setup Slack (Kurma Payı)
- Hold Slack (Tutma Payı)

Denklem 2. 1’de görülen kurma payı *Setup Slack* hesaplaması verinin gönderildiği kaydediciden başlayarak iki kaydedici arasındaki fonksiyon bloklarının sürelerini de hesaba katarak verinin ikinci kaydedici veri girişine ulaşmaya kadar yolda geçireceği zamanın en yüksek değerini hesaplar ve verinin ulaşma zamanını bulmaktadır. Ayrıca ikinci kaydedici için bir sonraki saat işareti zamanını ve veriyi alacak kaydedici için gerekli t_{su} değerini de kullanarak ikinci kaydedici için verinin talep edilmeye başlanacağı en düşük zamanı hesaplamaktadır. Devrenin kurma zaman payı, verinin ikinci kaydedici tarafından talep edilmeye başlandığı, zamandan verinin data girişine ulaşma zamanının çıkartılması ile elde edilmektedir. Kurma payı hesaplamasının daha detaylı grafiksel gösterimi için EK-1 incelenebilir. Kurma payının sıfırdan büyük olması gerekmektedir. Sıfırdan küçük değerlerin oluşması durumunda ya saat hızı düşürülmeli ya da kaydediciler arası fonksiyonlar indirgenmeli veya zamansal olarak dağıtılmaları gerekmektedir.

Kurma payı hesaplamalarında gönderen kaydedicideki birinci saat işareti ve alıcı kaydedici için bu saat işaretinin hemen ardından gelen aktif kenar hesaplamada kullanılır.

$$\text{Setup Slack} = \text{Data Required}|_{\min} - \text{Data Arrival}|_{\max} \quad (2. 1)$$

$$\text{Setup Slack} = (t_{clk2}|_{\min} - t_{su}) - (t_{clk1}|_{\max} + t_{cq}|_{\max} + t_{pd}|_{\max}) \quad (2. 2)$$

- t_{clk2} ikinci aktif saat kenarının alıcı kaydediciye ulaşacağı zaman
- t_{su} ikinci kaydedici için kurma zamanı (saat işaretinden önce verinin hazır bulunması gereken süre)
- t_{clk1} ilk aktif saat kenarının gönderici kaydediciye ulaşacağı zaman
- t_{cq} gönderen kayıtcı için aktif saat kenarının saat girişine ulaşmasından, verinin Q çıkışına ulaşacağı süre
- t_{pd} iki kaydedici arasındaki devreler için yayılma gecikmesi

Formülasyonda görülen max değerleri sinyal hattındaki değişimin bitip kararlı değerine ulaştığı anı, min değerleri ise sinyal hattında bir önceki işaretin değişmeye başladığı anı belirtmektedir.

Denklem 2. 3’te görülen tutma payı *Hold Slack* hesaplamasında alıcı kaydedicinin veriyi sorunsuz şekilde almayı bitirdiği hesaplamalarla doğrulanmaktadır. Alıcı kaydedicide

ikinci saat işaretinde veriyi almaya başlarken aynı zamanda ilk kaydediciye de saat işareti ulaşmakta ve gönderen kaydedicinin çıkışındaki bilgi değişmeye başlamaktadır. Bu değişimin ikinci kaydediciye ulaşmadan önce ikinci kaydedicinin veriyi almayı bitirmesi gerekmektedir. Aksi durumda ara kararlılık durumu oluşmaktadır. Tutma payı hesaplamasının daha detaylı grafiksel gösterimi için EK-2 incelenebilir.

Tutma payında sıfırdan küçük değerler oluşması durumunda iki kaydedici arasındaki yol uzatılmakta ya da ikinci kaydedicinin saat girişi erkene alınmaya çalışılmaktadır. Farklı bir teknik ise ilk kaydedicinin saat işaretini geciktirmektir. Böylece tutma payının değerinin pozitif değere ulaşması sağlanmakta ancak bu eylem kurma payını da etkileyeceğinden dikkatli olunmalıdır

$$\text{Hold Slack} = \text{Data Arrival}|_{\min} - \text{Data Required}|_{\max} \quad (2.3)$$

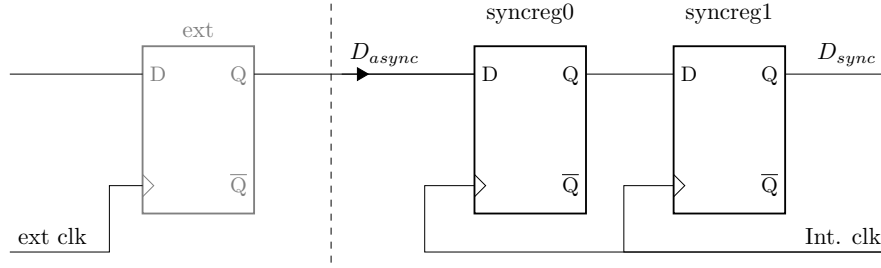
$$\text{Hold Slack} = (t_{clk1}|_{\min} + t_{cq}|_{\min} + t_{pd}|_{\min}) - (t_{clk2}|_{\max} + t_h) \quad (2.4)$$

Setup Slack hesaplamasında veri hatlarındaki yayılma gecikmesinin maksimum değeri kullanıldığı için bu hesaplama *max* hesaplama, hold slack hesaplamasına ise benzer sebeple *min* hesaplama isimleri de verilmektedir.

Senkron çalışan resetleme sinyalleri için Recovery Check ve Removal Check denen hesaplamalar ile zamanlama kontrolü yapılmaktadır.

Kullandıkları saat işaretlerinin birbiri ile senkronizasyonun olmadığı farklı saat bölgeleri arası geçişlerde veya saat işaretinin bulunmadığı veya yine senkronizasyonun bulunmadığı harici sinyal girişleri için Setup Slack ve Hold Slack hesaplamaları yapılamadığı gibi senkron olmayan resetleme girişleri için de Recovery Check ve Removal Check hesaplamaları yapılamamaktadır.

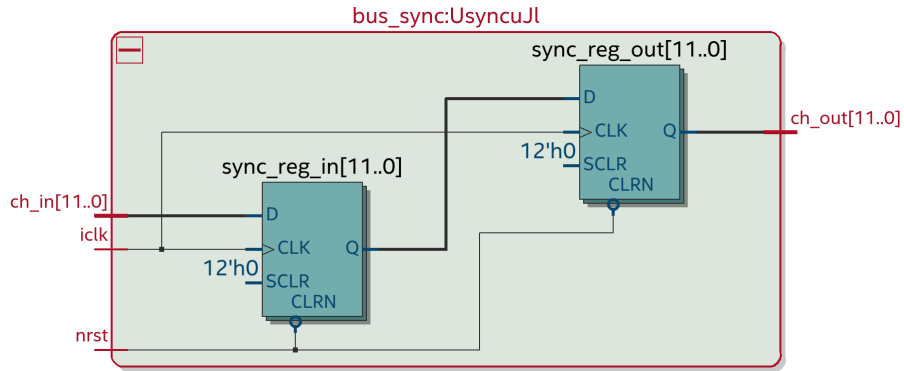
Böyle durumlarda ardışık kaydediciler kullanılarak sinyal girişi blokta kullanılan saat ile senkronize edilmeye çalışılmaktadır [72]. Aşağıda Şekil 2.7.'de ikili senkronlama dizisi gösterilmektedir. Dışardan gelen veya aynı yonga içinde bile olsa senkron olmayan bir saatin kullanıldığı devre bölgesinden gelen sinyali devrenin kullandığı aynı saatle senkronlamak için kullanılan sistemdir. Sinyallerin alınacağı devrenin kullandığı saat işareti ile çalıştırılan iki ardışık kaydedici kullanarak gelen sinyali senkron hale getirmeye çalışılmaktadır. Girilen bölgedeki saat hızı yükseldikçe ardışık bağlanan kaydedici sayısı arttırılmak durumundadır.



Şekil 2.7. Giriş sinyallerini senkron hale getirmek için kullanılan bir eşzamanlama çifti

Şekil 2.7.'de gösterilen devrenin yalnızca donanım tanımlama dilinde tasarlanması senkronlayıcı çiftin *fitter* işlemi sırasında donanımda hatalı konumlandırılmasına ve işlevini yerine getirememesine sebep olmaktadır. Bu sebeple sonraki alt bölümde değinilecek olan zamanlama tasarım kısıtları kullanılarak bu devredeki kaydedicilerin donanımda mümkün olduğunda birbirine yakın konumlandırılması sağlanmalıdır.

Uygun kısıtlar konulmadığı zaman, iki kaydedici arasında herhangi bir devre ve dolayısıyla gecikmeye sebep olacak bir şey tanımlanmadığı için hesaplama yapıldığında bu yapı uzak noktalarda bile konumlandırıldığında zamanlama açısından sorun çıkarmayan bir yapı olarak görülmektedir. Bu durum *fitter* tarafından kaydedicilerin birbirinden uzak noktalarda konumlandırılacaklardır ve bu uzak noktalar arasındaki hattın kaynaklanan gecikme sebebiyle devre senkronlama işlevi göstermeyecektir.

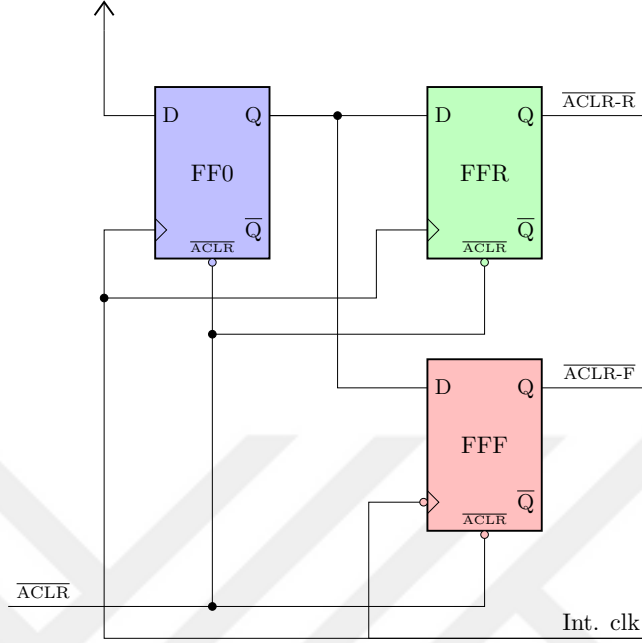


Şekil 2.8. Sentezleme sonrasında RTL gösterimde görünen bir veri hattı eş-zamanlama çifti

Şekil 2.8.'de görülen RTL gösteriminde mevcut tasarımlarımızdan birinde kullanılan 12 bitlik senkronlayıcı çift görülmektedir. Zamanlama tasarım kısıtları uygun ayarlanmadığında RTL gösteriminde bu fark anlaşılamamaktadır. Problem ancak devrenin çalışmasında problemler ile gözlenebilmektedir.

Tıpkı diğer asenkron giriş sinyalleri gibi asenkron resetlemelerde resetlemenin bitiş anı için senkronlayıcı devreler kullanılarak devre ile eş-zamanlı hale getirilmeye çalışılmaktadır. Resetleme sinyalleri ayrıca kullanılan saatin hangi kenarına

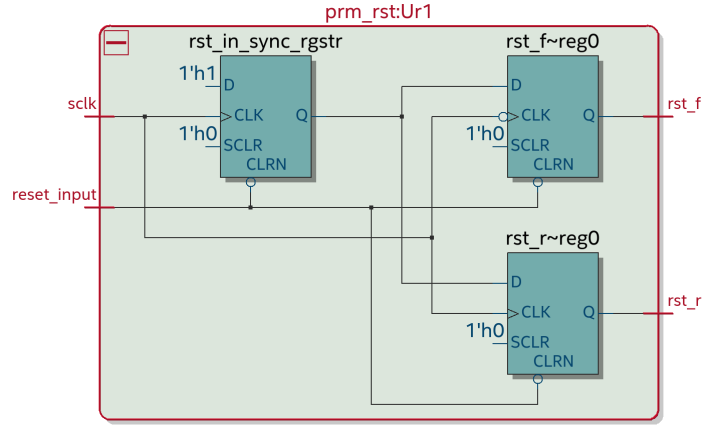
hizalanacaklarına göre kurulan devreler ile gerek düşen kenar gerekse yükselen kenara hizalanarak devrede kullanılabilmektedirler.



Şekil 2.9. Asenkron reset sinyalinin bitişi için senkronlayıcı devre

Şekil 2.9.'dan görülebildiği gibi asenkron resetleme sinyali, en başta 1 değerine bağlı bir kaydedicinin bulunduğu senkronlayıcı çiftler kullanılarak sinyalin bitiş anı senkronize edilir. Şekilde 2 adet ikinci senkronlayıcı bulunmaktadır. Bu kaydedicilerden birinin çıkışı saatin düşen kenarına hizalı reset üretirken diğeri çıkan kenara hizalı reset üretmektedir. Bu devrede zaman kısıtları çok daha önem kazanmaktadır, çünkü devrede çıkan kenarla tetiklenen giriş kaydedicisi ile çıkıştaki kaydedicilerden düşen kenarı kullanan kaydedici arasında saat periyodunun sadece yarısı kadar süre bulunmaktadır.

Şekil 2.10.'da RTL görünümünden bakıldığında görülebilen bir reset sinyali senkronlama devresi bulunmaktadır. Resetleme girişinde üçlü senkronlayıcı kullanılacaksa iki kaydedici aynı saat kenarı ile önde olmalı, üçüncü ve son aşamada gerekirse saatin hem düşen hem de çıkan kenarları için ayrı ayrı kaydediciler kullanılarak toplamda 4 kaydedici kullanılmalıdır.



Şekil 2.10. Sentezleme sonrasında RTL gösterimde görünen bir reset hattı eş-zamanlama çifti

2.4. Synopsys Tasarım Kısıtları

Donanım geliştirme endüstrisinde tasarımın tanımlanması işlemi gibi kısıtlamaların girilmesi de hayati öneme sahiptir. Kısıtlamalar olmadan fonksiyon modülleri istenen işlevleri yerine getirebilecek şekilde yapılandırılmamaktadırlar.

Donanım tasarımında en önemli kısıtlama konuları aşağıda sıralanmaktadır.

- Zamanlama
- Güç
- Alan (yonga yüzey kullanımı)

Bu sayılan konular tasarım sürecinde birbirlerinden bağımsız olmayan ve gerektiğinde birbirlerine ödünler vererek ilerleyebilen kısıtlardır. Sayılan kısıtlama konuları FPGA yongaları söz konusu olduğunda belli hususlar zaten FPGA yongasının tasarımından dolayı değiştirilmesi ya söz konusu olamamakta ya da mevcut yapı üzerinde kısıtlamanın uygulanabilen şekli ile uygulanmaktadır. Örnek olarak FPGA yongasında bir çıkış katının totem direğinin özellikleri değiştirilemez ancak istenirse çıkış yelpazesi (*fan-out*) arttırmak için ek bir hücrenin çıkışı sistem çıkışına eklenerek çıkış özellikleri iyileştirilebilmektedir.

Zamanlama kısıtlarının bir bölümü üretim söz konusu olmadığı için yine kısıtlı uygulanabilmekte veya bazı komut opsiyonlarının işlevi bulunmamaktadır. Buna bir örnek olarak iletim hatlarının geçiş güzergahları değiştirilemediği için sinyaller arası etkileşimleri kontrol eden bazı komutların FPGA uygulamalarında kullanımı bulunmamaktadır. Elbette bu durum yonga üretim için tasarlanırken düşünüldüğü için kendi tasarımlarımızı geliştirirken üzerinde durmamıza gerek kalmamaktadır.

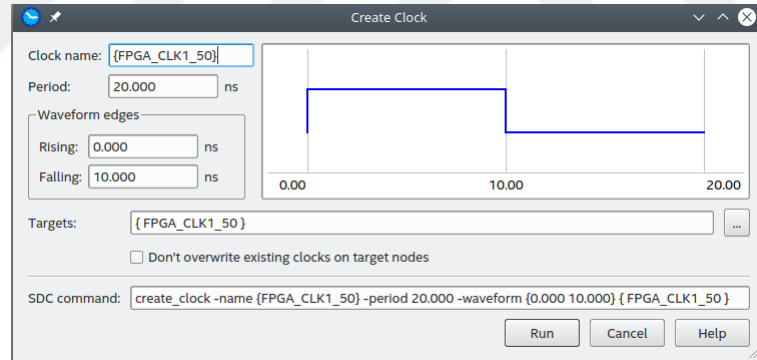
Endüstride kısıtlama tanımlarının bildirimi için defakto standart haline gelen Synopsys® firmasının tasarım kısıtları formatı kullanılmaktadır. SDC (Synopsys Design Constraints) olarak bilinen bu kısıtlar endüstride genel kabul ve uygulama alanı

bulmuştur. Firmalar bu kısıtlara kendi sistemleri için belli makrolar ekleyerek kullanıcılarına sunmaktadırlar. Firma tasarım kısıtlarının girilmesi için Tcl (Tool Command Language) dilini kullanmaktadır. Tcl programlama dili gerek gömülü sistem endüstrisinde gerekse de donanım geliştirme endüstrisinde yaygın şekilde kullanılan üst seviye bir yorumlanan dildir. FPGA tasarım platformları Tcl komutlarını doğrudan kabul edecek konsollar bulundurmakta ve henüz bir ara yüzü bulunmayan işlemler halen bu platformlarda Tcl komut satırından yapılmaktadır.

Intel Quartus Prime donanım geliştirme platformu Timing Analyzer isminde bir uygulama sunmaktadır. EK-6'da arayüzü görülebilen program SDC dosyasını okuyarak zamanlama hesaplamaları yapabilmektedir. Program ayrıca zamanlama kısıtlarının oluşturulması için formlar kullanmaktadır.

2.4.1. SDC Saat Tanımlama

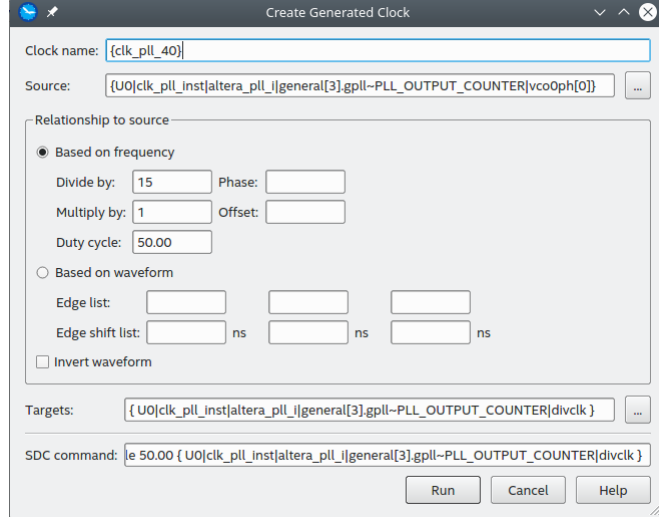
SDC komutlarının ilki ve en önemlisi saat tanımlamasında kullanılan `create_clock` komutudur. Daha önceden gördüğümüz gibi saat işaretleri hesaplamaların tamamında esas hareket noktası olarak kullanılmaktadır. Bu sebeple saat tanımlamaları da SDC komutlarında önemli bir yer tutmaktadır. Şekil 2.11.'de görülen formdan saat oluşturmak için gereken SDC komutunu oluşturulabilmektedir.



Şekil 2.11. Quartus Prime yazılımı Timing Analyzer modülü saat tanımlama ekranı

Saat tanımlaması gerçek saat işaretleri için zaten yapılmak durumundadır. Ancak üretilmiş saat işaretlerinin de tanımlanması gerekmektedir.

Üretilmiş saatler bir saat işaretinden PLL blokları yardımı ile üretilen saatler ise bu işlemi Altera firması `derive_pll_clocks` makro komutu yardımı ile yapabilmektedir. Ancak Bu komut kullanıldığında saatlere otomatik isim vermekte ve bu sebeple saatlerin gruplandırılması gibi işlemlerde problemlere sebep olmaktadır. Firma bu sebeple ilk sentezlemenin ardından programın mesajlar kısmından bu makronun ürettiği mesajın kopyalanarak SDC dosyasına manuel şekilde yazılması ve makronun daha sonrası için pasifleştirilmesini önermektedir.



Şekil 2.12. Quartus Prime yazılımı Timing Analyzer modülü üretilmiş saat tanımlama ekranı

Timing Analyzer programı üretilmiş saatler Şekil 2.12.'de görüldüğü gibi tanımlama için ayrıca bir form sunmaktadır. Bu form yardımıyla üretilen saatin kaynak durumundaki saatinin seçilmesi, parametrelerin girilmesi gibi kısımlarda yazılımsal destek sağlayabilmektedir. Gerçekte kodların manuel yazılması çok daha kolay olabileceksen, tasarımdaki hiyerarşi sebebiyle bir nesnenin adreslenebilmesi için gereken yol tarifi manuel kod girişlerini sıkıntılı hale getirmektedir. Bu sebeple geliştirme platformu ile sunulmuş formları kullanmanın faydaları bulunmaktadır.

Gerçek ve üretilmiş saatler haricinde ayrıca sanal saat kullanımları da bulunmaktadır. Özellikle giriş ve çıkışların SDC kısıtları girilirken sanal saat oluşturma'nın faydaları bulunmaktadır.

Saat kısıtlarının girilmesinin ardından, tanımlanan saatler için gecikme değerleri, saat işaretindeki düzensizlikler ve asenkron saat grupları gibi kısıtlar da tanımlanabilmektedir.

Giriş ve çıkış terminallerinde yine zaman kısıtlarının girilmesi gerekmektedir. Bu amaçla hem her giriş ve çıkış terminal grupları için sanal saatler tanımlamak gerekmektedir. Tanımlanan bu sanal saatlere göre minimum ve maksimum gecikme değerlerini girmek gerçek saat ile tanımlama yapmaktan daha avantajlı olmaktadır. Belli durumlar için ilgili girişi `set_false_path` komutu ile zaman hesaplarından muaf tutarak bu girişteki aralarında iletim hattı dışında eleman olmayan kaydedicilerin senkronizasyon çiftleri olarak tanınmalarını sağlamak farklı bir seçenek olarak elimizde bulunmaktadır. Tasarımımızda giriş ve çıkış terminalleri `set_false_path` ile tanımlandığı için girişlerdeki senkronizasyon çiftlerinin uygun yerleşimi sağlanmıştır.

2.5. Sentezleme ve Tasarlanan Donanımın FPGA Yongasına Aktarma

Geliştirme platformları öncelikle tanımlama dilleri kullanılarak sisteme girilen

yapıları inceler ve mümkün olduğunca yapıları basitleştirerek uygulanacak devrenin eleman sayısı bazında *fitter* öncesi son durumunu belirlemektedirler. Bu aşamadan sonra tanımlanan kısıtlara göre devre için yerleşim ve kaynak kullanımı yapılmaya çalışılmaktadır. Bu aşama *fitter* aşaması olarak bilinmektedir. Bir tasarımın kullanacağı yapıların sayısı FPGA kaynaklarından daha az olmasına rağmen *fitter* aşamasında yerleşim ve konumlandırma konusunda darboğazlar yaşanabilmekte ve tasarım FPGA yongasına sığdırılamayabilmektedir.

Fitter işleminde kullanılacak kaynakların konumları ve ara bağlantılar seçildikten sonra ilgili LUT hücrelerinde bulunan SRAM verileri ve ara bağlantılar için gerekli bilgiler bir SOF (SRAM Object File) veya POF (Programmer Object File) dosyasına kaydedilmektedir. Bu dosyalar kullanılarak tasarlanan donanım doğrudan FPGA yongasına veya yonganın seri programlayıcı entegresine gömülmek üzere hazırlanmış olur.

Tasarlanan mantıksal devrenin optimizasyonu ve *fitter* işlemleri yoğun işlem ve hafıza kaynakları tüketebilen uygulamalar olduğundan özellikle üst seviye yongalar için donanım tasarımı yapılacağı zaman sistemde 128 GB kadar bellek bulunması önerilmektedir.

Tasarlanan donanımın tanımlama ve kısıtlama bilgileri doğru şekilde girilmesi durumunda sistem sentezlendikten sonra MTBF (Mean Time Between Failures) Hatalar arası ortalama zaman değeri hesaplanabilmektedir. Bu hesaplama temelde en kötü durumdaki senkronlama çifti baz alınarak yapılan bir hesaplamadır.

Simülatörlerimiz için tasarladığımız sistemlerin MTBF değerleri düzeltmelerimizin ardından Şekil 2.13.'te görülebileceği gibi bir milyar yıllık değere ulaşmıştır. Zaten zaman kısıtlarının uygulanması ve gerekli düzenlemelerin yapılmasının ardından giriş ve çıkış portlarındaki temassızlıklar haricinde sistemlerimizde hiç problem görülmemiştir.

Slow 1100mV 100C Model Metastability Summary
Worst-Case MTBF of Design is 1e+09 years or 3.15e+16 seconds. Typical MTBF of Design is 1e+09 years or 3.15e+16 seconds.
Number of Synchronizer Chains Found: 152 Shortest Synchronizer Chain: 1 Registers Fraction of Chains for which MTBFs Could Not be Calculated: 0.079 Worst Case Available Settling Time: 9.113 ns
Worst-Case MTBF values are calculated based on the worst-case silicon characteristics, with worst-case operating conditions. - Under worst-case conditions, an increase of 100ps in available settling time will increase MTBF values by a factor of 8.0 Typical MTBF values are calculated based on the nominal silicon characteristics, at nominal operating conditions. - Under typical conditions, an increase of 100ps in available settling time will increase MTBF values by a factor of 17.8

Şekil 2.13. Sentezleme sonrasında sistem için oluşturulan MTBF raporu

Sentezleme sonrasında sistem ile ilgili edinilen raporlardan birisi olan *Setup Transfers* raporunda kaydediciden kaydediciye tanımlanmış tüm yollar gerek uygulanan saatlere göre ve gerekse aktif saat kenarlarına göre Şekil 2.14.'te görüldüğü üzere özet halinde görülebilmektedir. İlgili raporda *From Clock* ve *To Clock* kısımları saat işaretlerinin uygulandığı gönderici ve alıcı kaydedici saatlerini göstermektedir. Raporun

sağ sütunlarında bulunan tanımlamalar R ve F harfleri ile tanımlanmış olup bunlar *Rising* ve *Falling* olarak çıkan ve düşen kenarlar olarak görülmektedir. Bu şekilde örnek olarak RR sütunu gönderici ve alıcı kaydedicilerin saatin yükselen kenarlarında çalışanları göstermektedir.

Setup Transfers						
	From Clock	To Clock	RR Paths	FR Paths	RF Paths	FF Paths
1	altera_reserved_tck	altera_reserved_tck	9152	57	56	0
2	clk_pll_40	clk_pll_40	1844	0	574	873
3	clk_pll_40	clk_pll_50	false path	false path	0	0
4	clk_pll_50	clk_pll_50	336061	0	0	0
5	clk_pll_100	clk_pll_100	25	0	0	0
6	FPGA_CLK1_50	FPGA_CLK1_50	1	0	0	0

Şekil 2.14. Sentezleme sonrasında zamanlama raporlarında görülen saat bölgeleri arası geçişler

Bir sentezleme işleminin aşamalarında harcanan süreler Şekil 2.15.'te gösterilen kısımdan incelenebilir. Şekilden görüldüğü üzere sistemin tanımlanması ve optimizasyonu için yaklaşık iki dakika harcanırken *fitter* aşamasında bu zamanın 3 katı zaman harcanmıştır.

Tasks			Compilation			
	Task	Time				
✓	▶ Compile Design	00:09:49				
✓	▶ Analysis & Synthesis	00:01:46				
✓	▶ Fitter (Place & Route)	00:06:39				
✓	▶ Assembler (Generate programming files)	00:00:16				
✓	▶ Timing Analysis	00:00:47				
	▶ EDA Netlist Writer					
	▶ Edit Settings					
	▶ Program Device (Open Programmer)					

Şekil 2.15. Sentezleme süreci adımları ve işlem süreleri

Her sentezleme işleminin sonunda Şekil 2.16.'da gösterilen özet rapor ekranı gösterilmektedir. Bu özet raporda genel anlamda FPGA yongasındaki kaynak kullanımları özetlenmektedir. Örnek olarak iki motor ve bir parametre belirleme modülünün bulunduğu tasarımımda %66 oranında DSP bloklarını kullanmıştır. Bu şekilde ALM modüllerinin sadece %34 lük kısmı kullanılmıştır.

Table of Contents	Flow Summary																																						
- Flow Summary - Flow Settings - Flow Non-Default Global - Flow Elapsed Time - Flow OS Summary - Flow Log - Analysis & Synthesis - Fitter - Flow Messages - Flow Suppressed Message - Assembler - Timing Analyzer	<<Filter>> <table> <tr> <td>Flow Status</td><td>Successful - Sat Dec 24 15:16:39 2022</td></tr> <tr> <td>Quartus Prime Version</td><td>18.1.0 Build 625 09/12/2018 SJ Lite Edition</td></tr> <tr> <td>Revision Name</td><td>smltr_dcmt</td></tr> <tr> <td>Top-level Entity Name</td><td>smltr_dcmt</td></tr> <tr> <td>Family</td><td>Cyclone V</td></tr> <tr> <td>Device</td><td>5CSEBA6U23I7DK</td></tr> <tr> <td>Timing Models</td><td>Final</td></tr> <tr> <td>Logic utilization (in ALMs)</td><td>14,329 / 41,910 (34 %)</td></tr> <tr> <td>Total registers</td><td>25612</td></tr> <tr> <td>Total pins</td><td>93 / 314 (30 %)</td></tr> <tr> <td>Total virtual pins</td><td>0</td></tr> <tr> <td>Total block memory bits</td><td>57,030 / 5,662,720 (1 %)</td></tr> <tr> <td>Total DSP Blocks</td><td>74 / 112 (66 %)</td></tr> <tr> <td>Total HSSI RX PCSs</td><td>0</td></tr> <tr> <td>Total HSSI PMA RX Deserializers</td><td>0</td></tr> <tr> <td>Total HSSI TX PCSs</td><td>0</td></tr> <tr> <td>Total HSSI PMA TX Serializers</td><td>0</td></tr> <tr> <td>Total PLLs</td><td>1 / 6 (17 %)</td></tr> <tr> <td>Total DLLs</td><td>0 / 4 (0 %)</td></tr> </table>	Flow Status	Successful - Sat Dec 24 15:16:39 2022	Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition	Revision Name	smltr_dcmt	Top-level Entity Name	smltr_dcmt	Family	Cyclone V	Device	5CSEBA6U23I7DK	Timing Models	Final	Logic utilization (in ALMs)	14,329 / 41,910 (34 %)	Total registers	25612	Total pins	93 / 314 (30 %)	Total virtual pins	0	Total block memory bits	57,030 / 5,662,720 (1 %)	Total DSP Blocks	74 / 112 (66 %)	Total HSSI RX PCSs	0	Total HSSI PMA RX Deserializers	0	Total HSSI TX PCSs	0	Total HSSI PMA TX Serializers	0	Total PLLs	1 / 6 (17 %)	Total DLLs	0 / 4 (0 %)
Flow Status	Successful - Sat Dec 24 15:16:39 2022																																						
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition																																						
Revision Name	smltr_dcmt																																						
Top-level Entity Name	smltr_dcmt																																						
Family	Cyclone V																																						
Device	5CSEBA6U23I7DK																																						
Timing Models	Final																																						
Logic utilization (in ALMs)	14,329 / 41,910 (34 %)																																						
Total registers	25612																																						
Total pins	93 / 314 (30 %)																																						
Total virtual pins	0																																						
Total block memory bits	57,030 / 5,662,720 (1 %)																																						
Total DSP Blocks	74 / 112 (66 %)																																						
Total HSSI RX PCSs	0																																						
Total HSSI PMA RX Deserializers	0																																						
Total HSSI TX PCSs	0																																						
Total HSSI PMA TX Serializers	0																																						
Total PLLs	1 / 6 (17 %)																																						
Total DLLs	0 / 4 (0 %)																																						

Şekil 2.16. Sentezleme süreci sonunda oluşturulan rapor

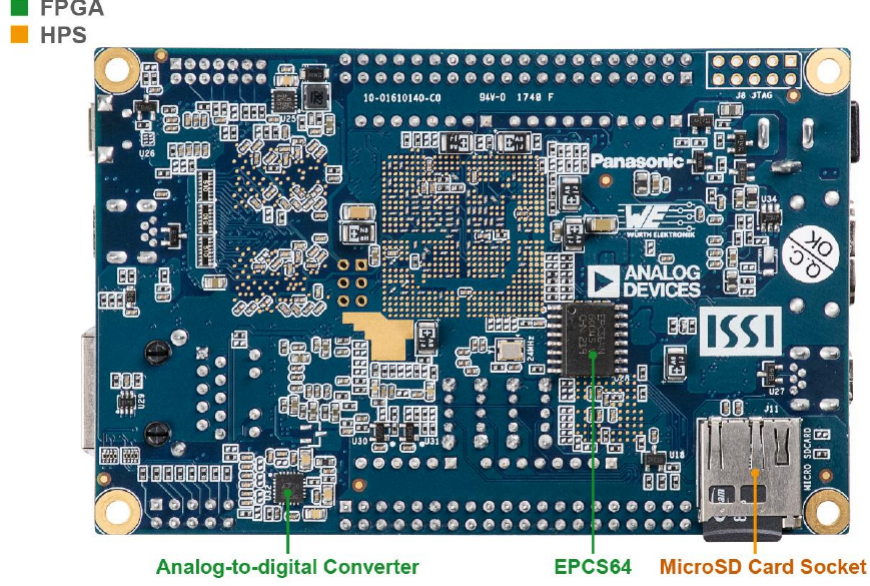
Raporda kırmızı renkte görülen Timing Analyzer kısmında devrenin işleyişinde soruna sebep olmayan hataların bildirimi yapılmıştır. Tasarımcıların işlerinden birisi de zamanlama hata ve raporlarını değerlendirip hangilerinin düzeltileceğini belirlemektir. STA analizi kabaca zamanlama değerleri ile yapılan dört işlem ile oluşmaktadır. Bu bağlamda incelemeye konu olan devrenin kullanımında probleme sebep olmasa bile çıkan tüm tanımlama eksiklikleri ve hesaplama problemleri iletilmektedir.

Örnek olarak tasarımıımızda saat olmadığı halde sistemde hücrelerin saat girişlerinde görülen bir işaret için uyarı oluşturulmaktadır. Ayrıca FPGA yongasını kullanılmayan yığınla giriş çıkış terminalleri için hata mesajı oluşmaktadır.

Zamanlama hesaplamaları ayrıca farklı *köşe* değerleri için yapıldığından birinde olmasa bile bir diğerinde sorun olabilmektedir. Örnek olarak farklı sıcaklık ve farklı gerilim senaryoları için bir tasarım sorun oluşturabilmektedir.

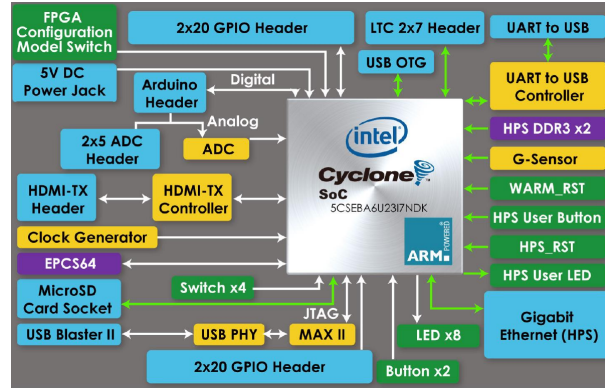
FPGA geliştirme ortamları zamanlama kısıtlarına ek olarak tasarlanan modüllerin yonga üzerindeki konumlandırılmalarının kontrolünü de sağlamaktadır. Bu işlem için EK-9'da görülen "Chip Planner" uygulaması kullanılmaktadır. Çalışmalarımız sonunda biz bu yönde herhangi bir kısıtlamaya ihtiyaç duymadığımız için *fitter* programı tasarımı sadece saat ve giriş çıkış port kısıtlarına göre konumlandırma yapmıştır. Tasarlanan sistemin yonga yüzeyi konumlandırması EK-10'da görülebilmektedir.

DE10-Nano-SoC geliştirme kartının üst yüzünde görülen donanım kısımları bağlı oldukları FPGA kısımları için farklı renklerde kodlanmış halde Şekil 3.1.'de görülmektedir. Geliştirme kartının alt yüzünde FPGA programlanabilir mantık kısımlarına bağlı olan ADC entegresi ve FPGA yongasının seri programlayıcı entegresi Şekil 3.2.'de görülmektedir.



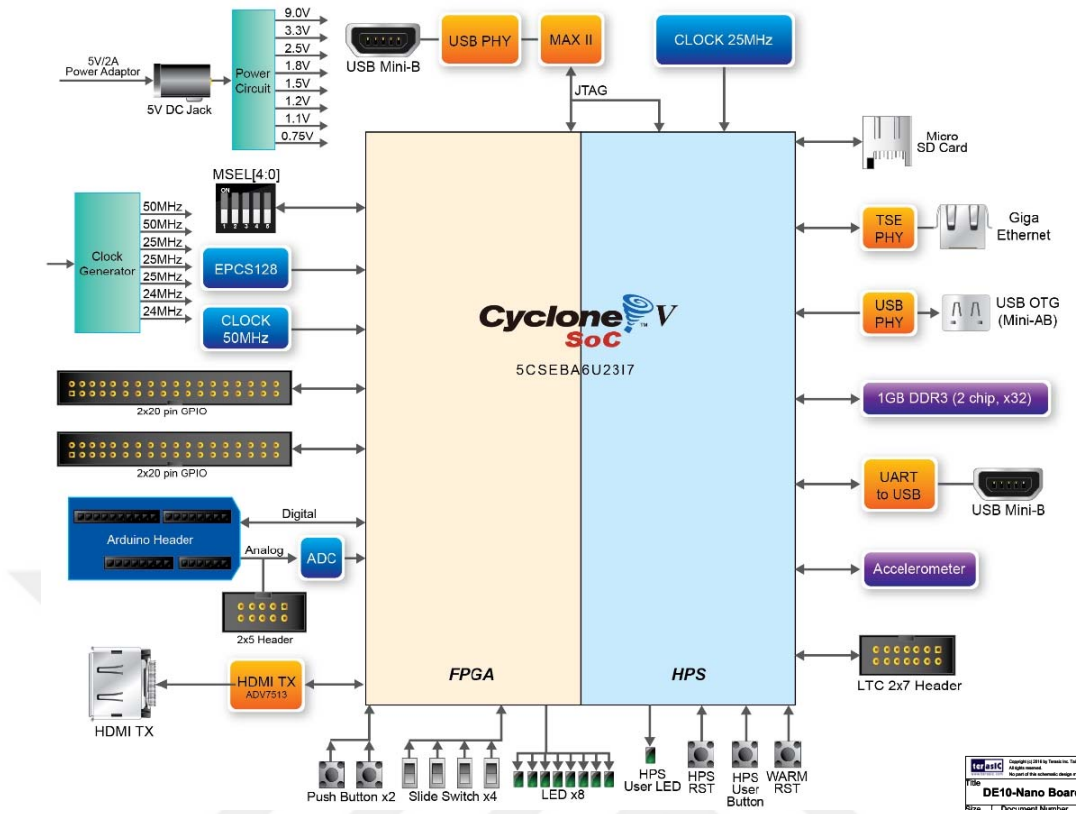
Şekil 3.2. DE10-Nano geliştirme kartı arka yüzü

Geliştirme kartının Şekil 3.3.'te görülen blok diyagramında beyaz oklar ile bağlantısı görülen bloklar HPS sistemi dışında FPGA yongasına bağlı birimlerdir.



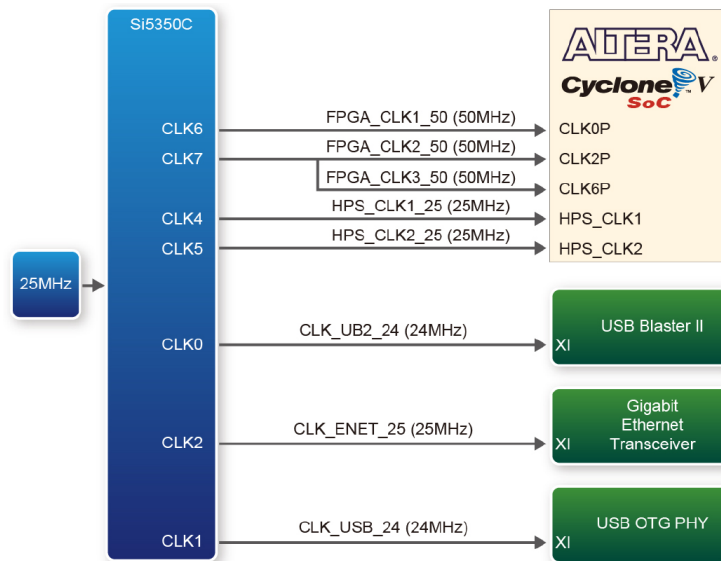
Şekil 3.3. DE10-Nano geliştirme kartı blok yapısı

HPS bağlantıları ayrımının daha kolay yapıldığı Şekil 3.4.'te görülen çizimde ADC entegresi, GPIO terminalleri, butonlar ve ledlerin entegrenin FPGA kısımlarına bağlı olduğu görülmektedir. Bu sayede entegrenin konfigürasyonu için bırakılan seçme anahtarlarından HPS sistemi kapatılarak entegrenin fazla ısınması engellenmiştir.



Şekil 3.4. DE10-Nano geliştirme kartı blok yapısı 2

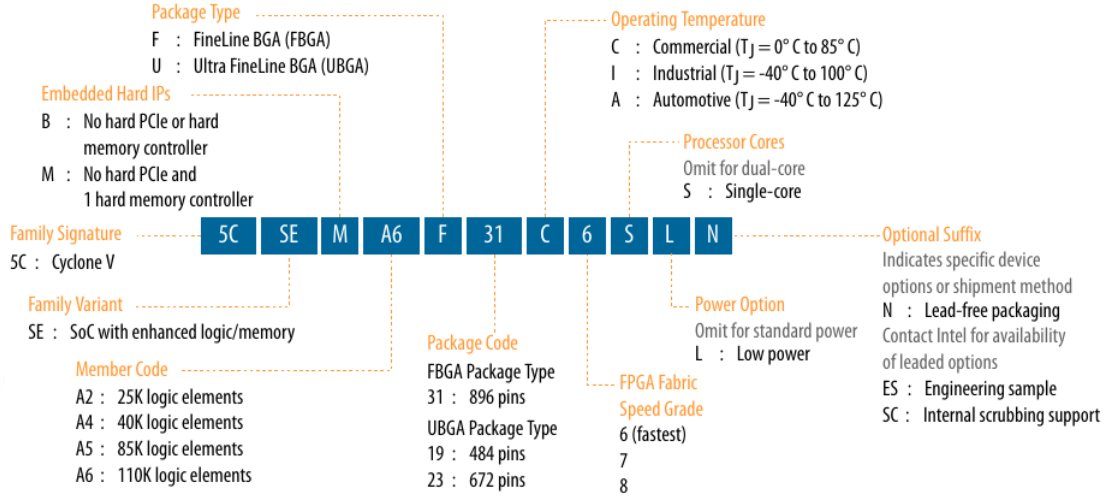
Kartın saat üreticinin bağlantılarının görüldüğü Şekil 3.5.'ten anlaşılacağı üzere 25 MHz saat işaretinden saat üretici sayesinde 50 MHz saat işareti üretilerek FPGA entegresinin kullanımı için üç terminalden bağlantı yapılmıştır.



Şekil 3.5. DE10 saat işareti üreticisi

3.1.1. FPGA Yonga Özellikleri

Intel (Altera) FPGA yongalarının temel özellikleri Şekil 3.6.'da görüldüğü üzere isimlendirmelerinde kodlanmaktadır.



Şekil 3.6. FPGA yongası isim kodlaması.

Kart üzerindeki yonganın kodlaması 5CSEBA6U23I7 olarak okunduğundan yonganın temel özellikleri için;

- 5C Cyclone V Serisi bir entegre olduğunu
- SE SoC grubu iyileştirilmiş bellek ve mantık birimleri içerdiğini
- B herhangi bir gömülü PCIe veya hafıza kontrolcüsü içermediğini
- A6 110 bine yakın LE içerdiğini
- U UBGA yonga paketlemesine sahip olduğunu
- 23 672 pin UBGA paket yapısında olduğunu
- I Endüstriyel sınıf entegre olduğunu
- 7 FPGA yapısının hız sınıfının 7 olduğunu

yongamızın isim kodlamasından anlayabiliyoruz ancak yonganın fiziksel kaynaklarının detaylarını üretici firma dokümanlarından öğrenebiliyoruz [73]. FPGA yongasının kaynaklarının daha detaylı bir listesi Tablo 3.1.'den incelene bilmektedir.

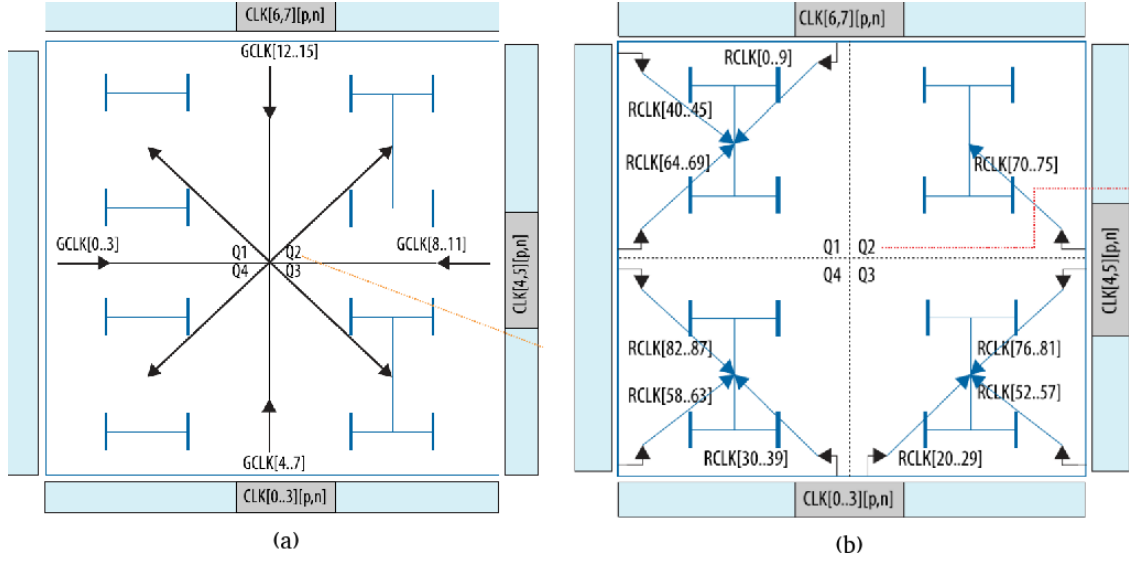
Tablo 3.1. Intel Cyclone V SE 5CSEBA6U23I7NDK donanım kaynakları.

Kaynaklar	İzahat	Değer
LEs (K)	Logic Elements	110
ALMs	Adaptive Logic Modules	41910
Reg	Registers	116036
MLAB (Kb)	Memory	621
M10K (Kb)	Memory	5570
DSP	DSP Blocks	112
18x18 Mult.	18x18 Multipliers	224
PLL	Phase Locked Loop (Fractional)	6
DLL	Delay Locked Loop	4
Global Clock	Clock Networks	16
GPIO	General Purpose I/O	288

Farklı saat hızlarında çalışan bloklar yonga içerisinde farklı saat dağıtım bölgelerine ihtiyaç duymaktadırlar. Donanım geliştirme işlemlerinde veri hatları ve saat hatları birbirlerinden farklıdırlar. Veri hatları normal hatlardır ve hat üzerinde normal tampon birimleri kullanılmaktadır.

Saat hatlarında normal tampon devreleri kullanılamaz çünkü normal tampon devrelerinin çıkış totem direklerinin CMOS yapısında **p** ve **n** kanallı FET'ler eş büyüklüklerdedirler. Bu özellik sinyalin düşen kenarının normal hızda, yükselen kenarının ise daha düşük eğimde olmasına sebep olmaktadır. Saat sinyalleri hat üzerinde defalarca tamponlama işlemine maruz kaldığından ve fan-out değerleri de yüksek olduğundan birkaç tampon sonrasında saat işaretinin pozitif kısımlarının genişlikleri kabul edilebilir sınırların altına düşmektedir.

Saat hatlarındaki tampon devrelerinde p-kanallı transistörün alanı n-kanallı olandan en az 2 kat daha büyüktür ve bu sinyalin yükselen kenarının da düşen kenar kadar hızlı olmasını sağlar. Bu sebeplerden dolayı saat işaretinin dağıtımında sadece saat tamponları kullanılır, bu gereklilik de farklı saat kullanan blok sayılarının ve konumlandırılmalarının kısıtlanmasına sebep olmaktadır. Ayrıca saat dağıtım hatları Setup Slack ve Hold Slack hesaplamalarında gönderici ve alıcı kaydedicilerin saatlerinin birbirlerine eş olabilmesi için olabildiğince homojen şekilde dağıtım yapılarına sahiptirler. Şekil 3.7.'de Cyclone V yongalarının global ve bölgesel saat dağıtım bölgeleri ve yönleri gösterilmektedir.



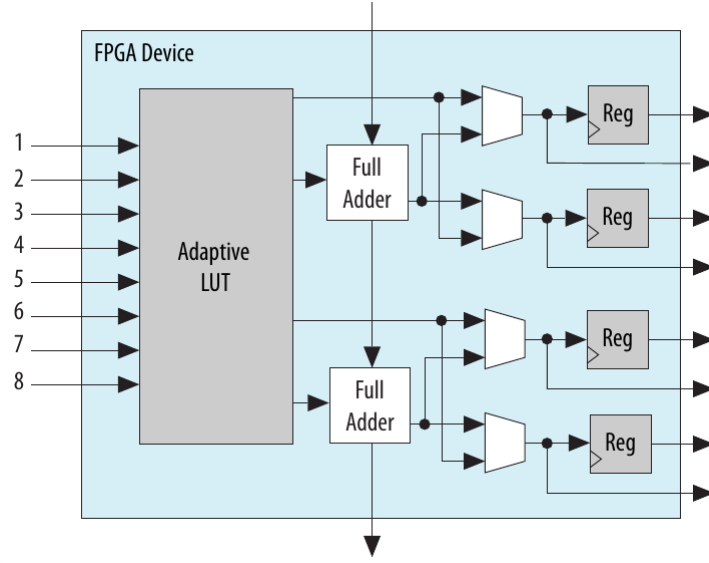
Şekil 3.7. Cyclone V yongalarında bulunan global saat dağıtım hat ve bölgeleri a) Global saat dağıtım hatları ve bölgeleri b) Bölgesel saat dağıtım hatları ve bölgeleri

Farklı saatler kullanan kaydediciler arası veri hattı geçişlerinin bu gösterilen bölgeler arasında seyahat etmesi gerektiğinden *fitter* işlemi blokları konumlandırırken daha fazla kaynak kullanmak durumunda kalmaktadır.

Saat bölgeleri arası geçişler CDC (Clock Domain Crossing) olarak adlandırılmakta ve özel dikkat gerektirmektedir. Tasarımımızda saat bölgesi geçişlerinde eş-zamanlayıcı bloklar kullanılmıştır. Bazı saatler arası geçişleri asenkron olmaya zorlamak için SDC komutları ile saatlerin gruplandırılması işlemi kullanılmıştır.

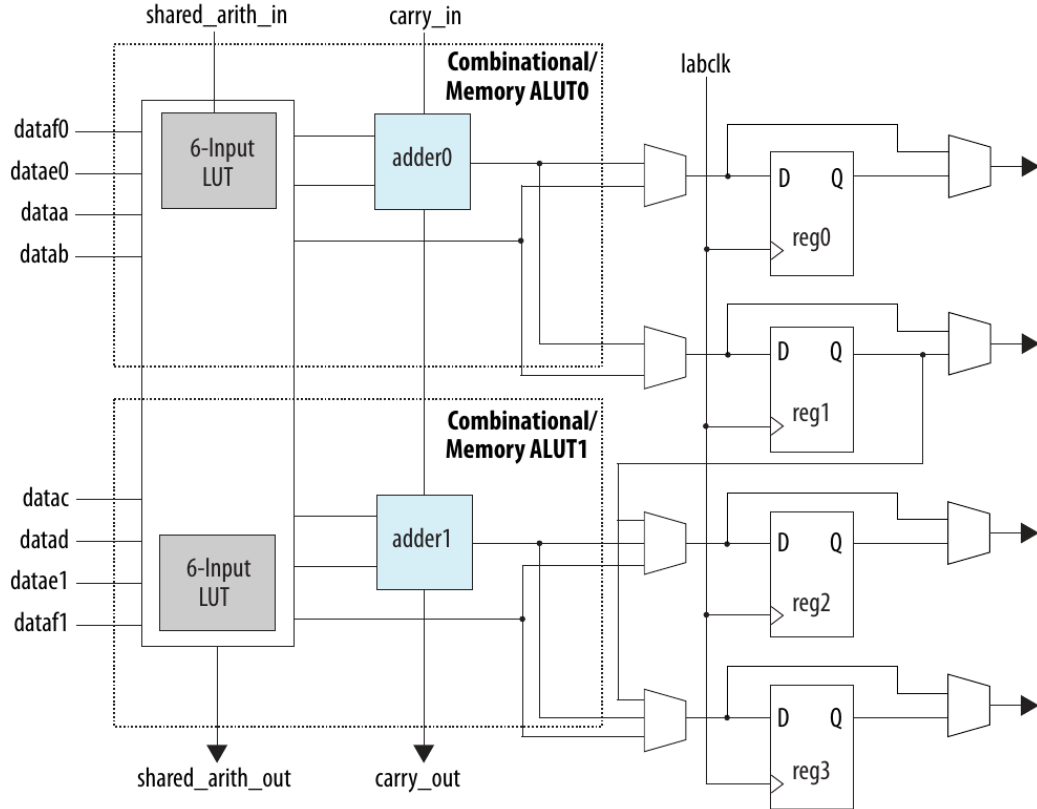
FPGA yongaları ile yapılan donanım geliştirme çalışmalarında sadece temel mantık birimlerinin kullanımından kaynaklanan darboğazların aşılması için firmalar normal mantık birimlerinden daha karmaşık yapıdaki özel birimleri de yongalarına dahil ederek geliştirme işlemlerini kolaylaştırmaktadırlar. Bu gelişmiş yapılar hem mantık birimlerinin kullanımının hem de ara bağlantılar için tahsis edilen hatların kullanımının azaltılmasını sağlamaktadırlar.

Şekil 3.8.'de gösterilen diyagramda Cyclone V yongalarında kullanılan ALM ünitelerinin genel blok şeması görülmektedir.



Şekil 3.8. Cyclone V yongalarında bulunan ALM yapıları

Şekil 3.9.'da ALM ünitelerinin daha detaylı bir gösterimi yapılmış olup burada LUT yapısının nasıl parçalı yapıda olduğu görülebilmektedir. Daha detaylı bir çizim için EK-7 incelenebilir.

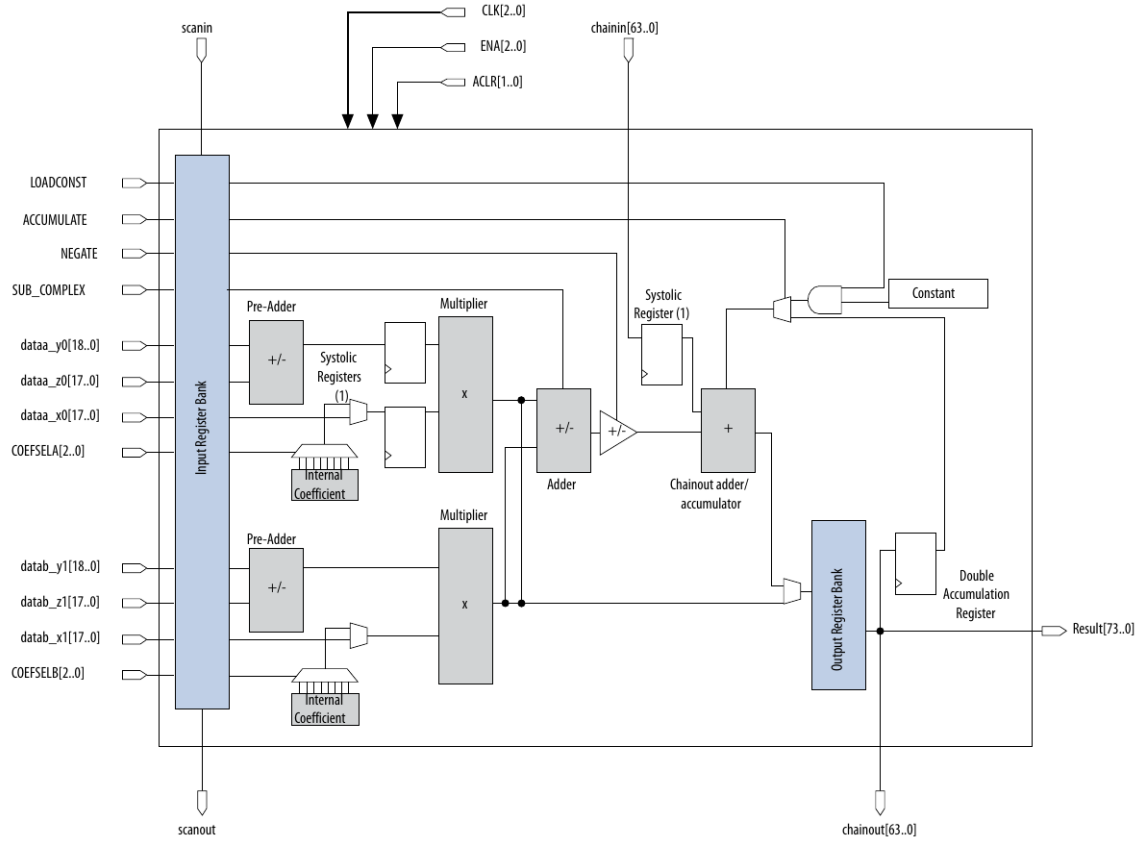


Şekil 3.9. Cyclone V yongalarında bulunan ALM yapısı detayı

FPGA yongalarında genel ve özel mantık birimleri haricinde ayrıca DSP hücreleri de bulunmaktadır. DSP hücreleri temelde donanımsal çarpma ünitelerini barındırmaktadırlar. Tasarımlarımızda kullandığımız kayar nokta modülleri sebebiyle sentezleme raporlarımızdan DSP kullanım oranının %66 değerinde olduğu görülmektedir.

Şekil 3.10.'da Cyclone V entegrelerinde bulunan DSP yapısı görülmektedir. Yapı incelendiğinde DSP modülünün MAC (Multiply -Accumulate) işlemlerine ne kadar uygun olduğu görülebilmektedir. Ayrıca geniş veri hatlarının bulunduğu işlemler için DSP modüllerinin paralele bağlantılarda kullanılmak üzere ek terminalleri de görülebilmektedir.

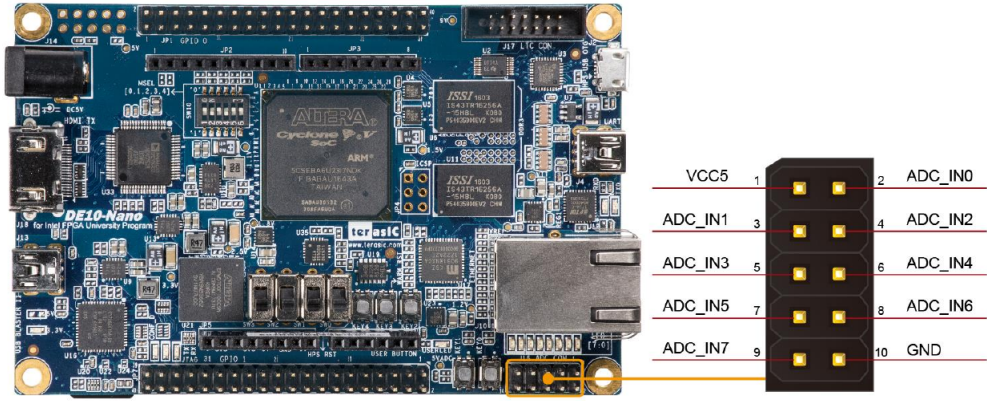
IP modülleri ayarlanırken DSP hücrelerinin kullanılıp kullanılmayacağı seçilebilmektedir. Ancak kullanımın engellenmesi durumunda aynı işlevin yerine getirilmesi için oldukça yüksek sayıda mantık biriminin kullanımını gerektirmektedir. Mevcut tasarımlarımızda DSP birimleri olmadan tasarımın yongaya sığmasının ihtimali bulunmamaktadır.



Şekil 3.10. Cyclone V yongalarında bulunan DSP modül yapısı

3.1.2. ADC Entegre Özellikleri

DE10-Nano-SoC kartında kullanılan FPGA yongasında gömülü ADC fonksiyonu bulunmadığından bu özellik karta LTC2308 ADC entegresi ile ilave edilmiştir. LTC2308 entegresinin kart üzerindeki bağlantıları Şekil 3.11.'den görülebilmektedir.



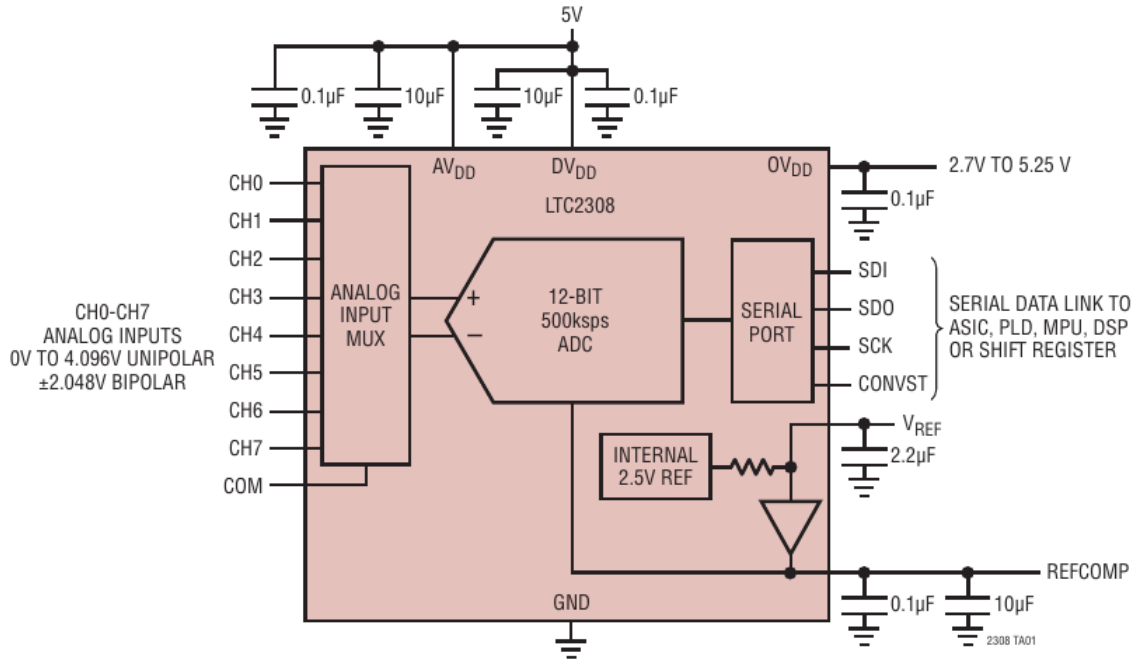
Şekil 3.11. ADC entegresi bağlantı soketi

ADC entegresinin genel özellikleri Tablo 3.2.'de genel olarak aktarılmaktadır.

Tablo 3.2. Linear Technology LTC2308 ADC entegresi özellikleri

Özellik	İzahat	Değer
Çözünürlük	Veri yolu genişliği	12 bit
Hız	Örnekleme hızı	500 ksps
Reg	Registers	116036
Arayüz bağlantısı		4-telli SPI
Arayüz tipi	Bağlantı arayüzü	SPI/MICROWIRE
Kanal Sayısı		8
Arayüz hızı (Max)	iletişim hızı	40 MHz

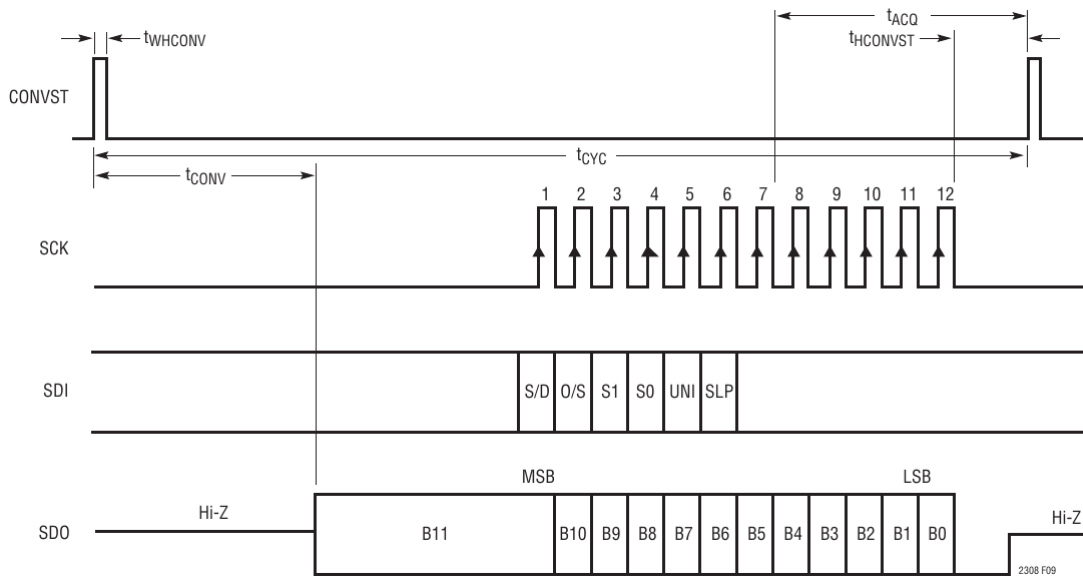
Entegrenin iç yapısının blok şeması Şekil 3.12. incelendiğinde, entegrede dahili referans üretici olduğu görülmektedir.



Şekil 3.12. ADC entegresi blok şeması

Donanım tasarımında kullanılacak olan arayüz zamanlama çizelgesi Şekil 3.13.'te görülebilmektedir. Bu çizelge entegrenin en yüksek hızda çalışması için kullanılan şemadır ve ADC çevrim komutu burada hızlıca verilip kapatılmaktadır.

Şekil 3.13.'te görülen zamanlama diyagramı kullanılarak FPGA için bir iletişim modülü tasarlanmış olup en yüksek saat hızı olan 40 MHz değerini kullanmaktadır.



Şekil 3.13. ADC entegresi SPI arayüzü zamanlama çizelgesi

3.1.3. GPIO Port Özellikleri

Geliştirme kartında kullanılan GPIO terminali ek bir donanıma ihtiyaç duyulmadan sadece dışı uclu kablolar ile çevre donanımlara bağlanmayı olanaklı kılmaktadır. GPIO portlarından iki adet bulunmaktadır. Bu portlardan birisi analog arayüz kartı ile iletişim için kullanılmıştır. Diğer terminal ilk başlarda geliştirilen donanımda fonksiyon testleri yapmak ve hata bulmak amacı ile Digital Discovery lojik analizörünün bağlanması için kullanılmış sonrasında testlerde kullanılan harici motorun enkoder okumaları için kullanılmıştır.

GPIO portlarının çalışma gerilimleri bu geliştirme kartı için seçilememektedir. GPIO port hatları üzerinde koruma devresi bulunmadığından bağlantıların dikkatli şekilde ve tek gerilim standardına bağlı yapılması gerekmektedir.

Geliştirme kartlarında FPGA yongası zaten montajlı olduğundan giriş çıkış pinleri bellidir ve bir dosya ile birlikte kutu içeriğinde bulunmaktadır. Tasarımda baskı devre kartı da tasarlanacaksa giriş çıkış pinlerinin tayini gerekmektedir. Bu işlem için Quartus Prime platformunun EK-4'te görülen "Pin Planner" isimli uygulaması kullanılabilmektedir.

3.1.4. USB-Blaster II Bağlantı Arayüzü

Tasarlanan donanımın FPGA yongasına aktarılması için Altera firması USB-Blaster II arayüzünü geliştirmiş ve geliştirme kartlarında kullanmıştır. USB-Blaster II arayüzü SOF ve POF dosyalarının aktarımının yanı sıra hata ayıklama ve gözlem amacı ile Quartus platformu ile kullanılabilmektedir.

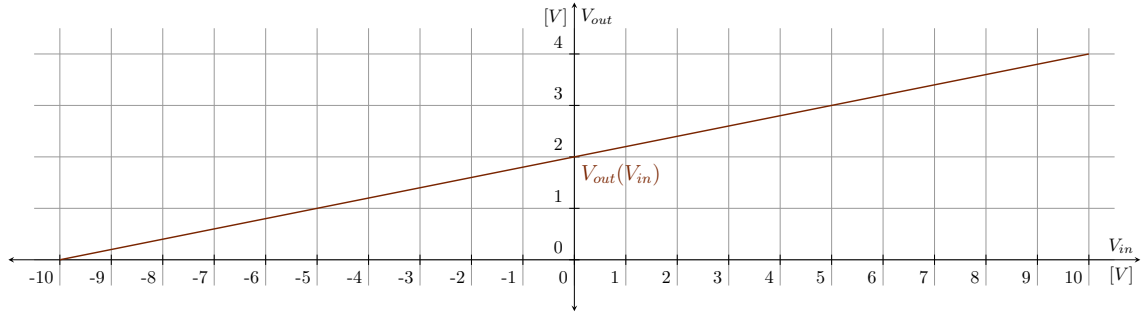
Signal Tap Logic Analyzer yazılımında bu arayüz üzerinden sinyalleri okuyarak bilgisayar üzerinden donanımın belli ölçüde izlenmesini sağlamaktadır. Kolay konfigürasyonu ve pratik kullanımı sebebiyle Digital Discovery analizörü ile yaptığımız işlemleri bu yazılım ile kolayca yapabilmekteyiz.

3.2. Analog Giriş Çıkış Kartı

EK-13'te görülen analog giriş çıkış kartı, simülatöre dışarıdan analog sinyal girilebilmesi için yapılmış olup hem analog sinyali kendisi üretebilmekte hem de simülasyonlarda harici sinyal jeneratörlerinden uygulanan gerilimin ADC entegresinin kabul edeceği gerilim aralığına çekilmesi için kullanılmıştır.

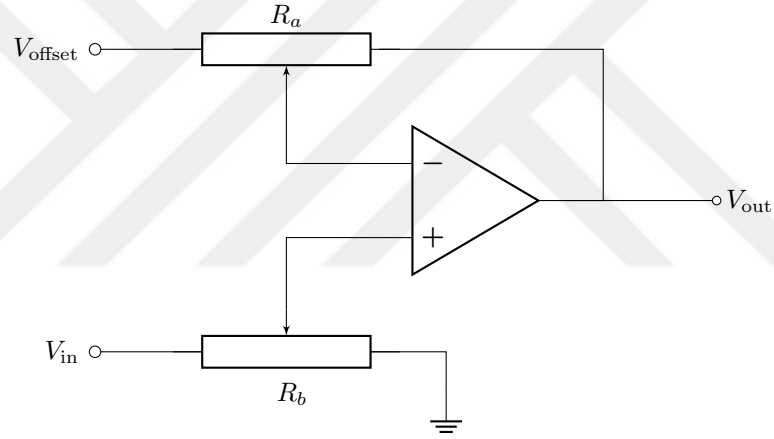
3.2.1. Analog Sinyal Giriş Devresi

Karşılaştırmalı testler sırasında PMDC motor ve RL devresine uygulanan gerilim $\pm 10V$ civarında bir gerilim olduğu halde bu gerilimin ölçeklenerek 0-4 V aralığına ölçeklenmesi gerekmektedir. Bu ölçekleme işlemi analog giriş çıkış kartının giriş sinyali ölçekleme devresinde yapılmakta olup, ölçekleme Şekil 3.14.'te görselleştirilmiştir.



Şekil 3.14. Analog giriş sisteminin giriş sinyallerini ölçeklemesi

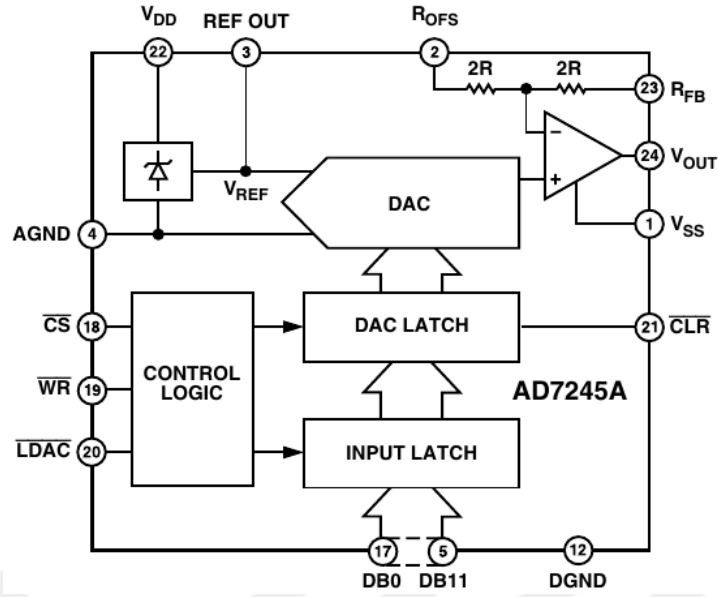
Analog giriş sinyali ölçekleme devresi bu işlemi hem kazanç ayarı hem de offset ekleyerek yapmaktadır. İlgili ölçekleme devresinin şematik çizim Şekil 3.15.'ten incelenebilmektedir.



Şekil 3.15. Analog giriş sinyallerinin ölçeklenmesinde kullanılan devre şeması

3.2.2. Analog Sinyal Çıkış Devresi

DE10-Nano geliştirme kartında DAC sistemi bulunmadığı için bu işlemin analog giriş çıkış kartında yapılması gerekmektedir. Bu işlem için MX7245JN entegresi kullanılmıştır. Bu entegre 12 bitlik DAC sistemi barındıran ve paralel iletişim yolu üzerinden bağlantısı yapılarak kontrol edilebilen bir entegredir. Sinyal çıkışının oturma zamanı yaklaşık $5 \mu s$ olarak belirtilmiş olan entegrenin blok diyagramı Şekil 3.16.'da görülmektedir.

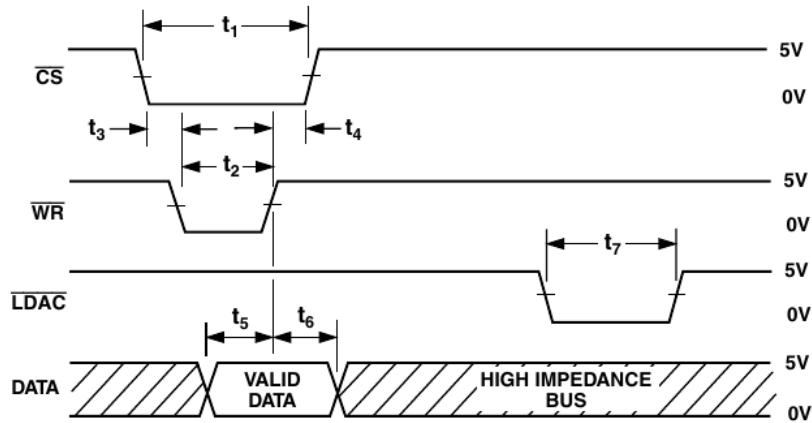


Şekil 3.16. DAC entegresi fonksiyonel blok diyagramı

DAC entegresinin çıkış katı doğrudan ölçüm cihazlarında sorun yarattığı için EK-8’de gösterilen analog devre ile çıkış dışarıya aktarılmıştır. Analog giriş çıkış kartında DAC entegreleri gerek simetrik gerekse de tek kaynaktan beslenebilmekte ve çıkış devresi de buna göre anahtarlar yardımı ile değiştirilebilmektedir. Entegre mevcut hali ile $\pm 12V$ simetrik besleme ile beslenmiştir. Entegre çıkışı $\pm 5V$ sinyal üretecek şekilde ayarlanmıştır. DAC entegresinin çıkış sinyali LM318N yüksek hızlı op-amp entegresi ile işlenerek $\pm 10V$ aralığında sinyal gözlemlenecek şekilde ayarlama yapılmıştır.

Entegrenin paralel haberleşme protokolü Şekil 3.17.’de gösterilmiş olup haberleşme için tasarlanan FPGA modülü bu protokolü kullanacak şekilde tasarlanmıştır.

Analog giriş çıkış kartında bahsedilen DAC sistemlerinden 8 tanesi için devre kurulmuş olup 5 tane için sistem çalıştırılmıştır.



Şekil 3.17. DAC entegresi PI arayüzü zamanlama çizelgesi

3.3. Sürücü Kartı

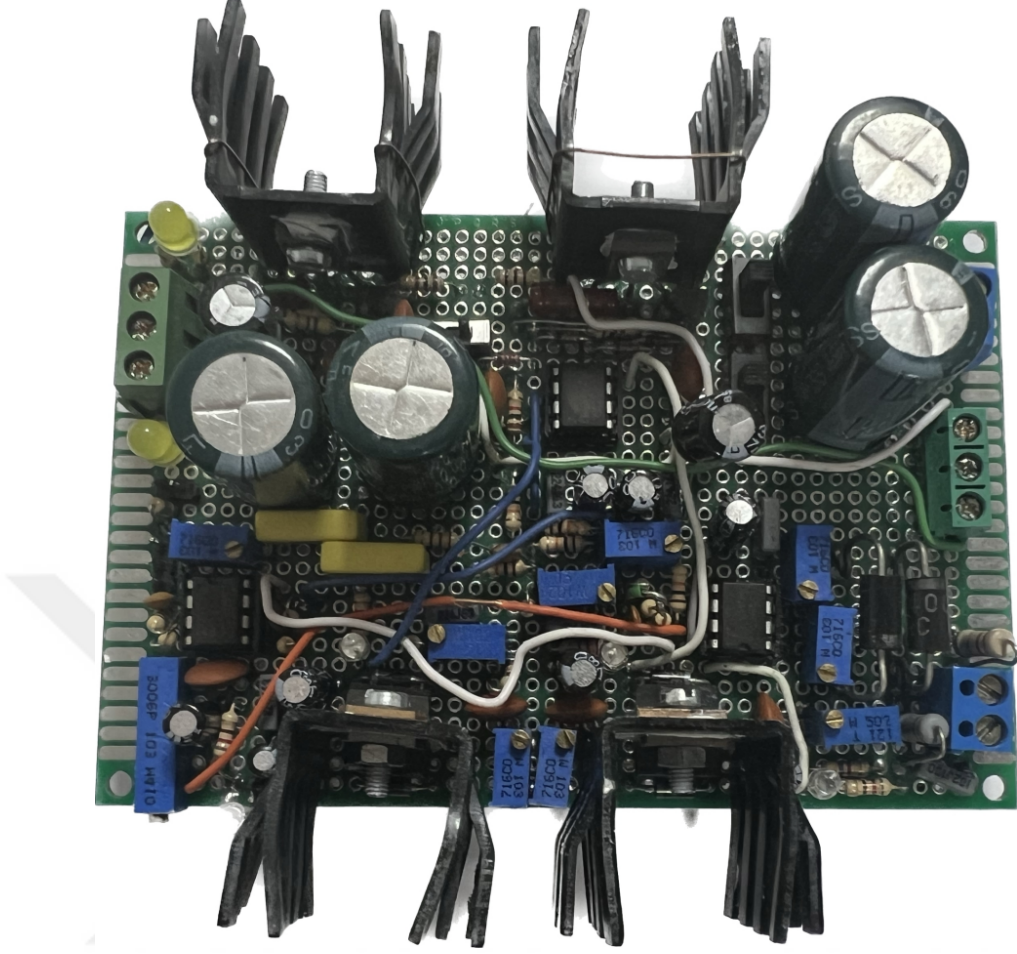
Simulatörün gerçek sistemlerle karşılaştırılabilmesi için gerçek RL ve PMCD motor sistemlerinin farklı gerilim sinyallerine maruz bırakılması ve bu uygulanan gerilimin varsa üzerindeki bozulmalar ile birlikte tamponlanarak analog giriş çıkış kartına aktarılması gerekmektedir.

Bu işlem için Analog Discovery 2 ünitesinin sinyal çıkışı kullanılmıştır. Bu ünitenin sinyal çıkış değeri $\pm 5V$ olduğundan $\pm 10V$ değerine çıkartmak için önce bir kazanç devresi ile yükseltilmektedir. Sonrada yüksek güçlü bir totem çıkışa sahip AB tipi kuvvetlendirici ile motora veya RL devresine gerilim uygulanmaktadır.

Uygulanan gerilimde bozulma olma ihtimali olduğundan bu bozulma ile birlikte gerilim örneklenerek ayrı bir hat üzerinden analog giriş çıkış kartına iletilmektedir. Ayrıca sürücü kartı çıkışta oluşan akımı şönt direnç üzerinden ölçüp kuvvetlendirerek ölçülen akım değerini ayrı bir terminalden izlenmek üzere dışarı aktarır.

Devrenin gerilim kuvvetlendirme ve ölçüm kısımları $\pm 12V$ simetrik besleme ile çalışmaktadır. Buna karşın çıkış totem direği $\pm 30V$ ile beslenmektedir. Bunun sebebi PMDC motorlara kare dalga gerilimi uygulanırken hızlıca çekilen demeraj akımının besleme hattını bozmasıdır. Bunu engellemek ve totem çıkış katında transistörler üzerinde oluşan gerilim düşümlerine rağmen çıkışta gerekli voltaj değerini koruyabilmek için çıkış katı ayrı ve daha yüksek bir gerilim ile beslenmiştir.

Sürücü kartı Şekil 3.18.'de gösterilmektedir. Totem çıkışın sürücü transistörleri 3'lü darlington yapıdadır ve burada darlingtonun son kademesinde kullanılan transistörler görünmemektedir. Bu transistörler takılı oldukları soğutucu üzerinden karta kablolar ile bağlanmaktadır.



Şekil 3.18. Sürücü kartı

Kartın sol tarafında $\pm 12V$ klemensli besleme girişi ve yanında pinli bağlantı uçları ile uygulanacak gerilimin alınıp ölçülen uygulanmış gerilimin ve ölçülen akımın dışarı aktarıldığı terminaller bulunmaktadır. Sağ tarafta ise $\pm 33V$ besleme girişi, totem katı güç transistörleri bağlantı terminali ve yükün bağlandığı terminal bulunmaktadır.

Denemeler sırasında motora kare dalga uygulanırken bile totem çıkışında çok az bozulma gözlemlenmiştir.

3.4. Sabit Mıknatıslı Çekirdeksiz Doğru Akım Motoru

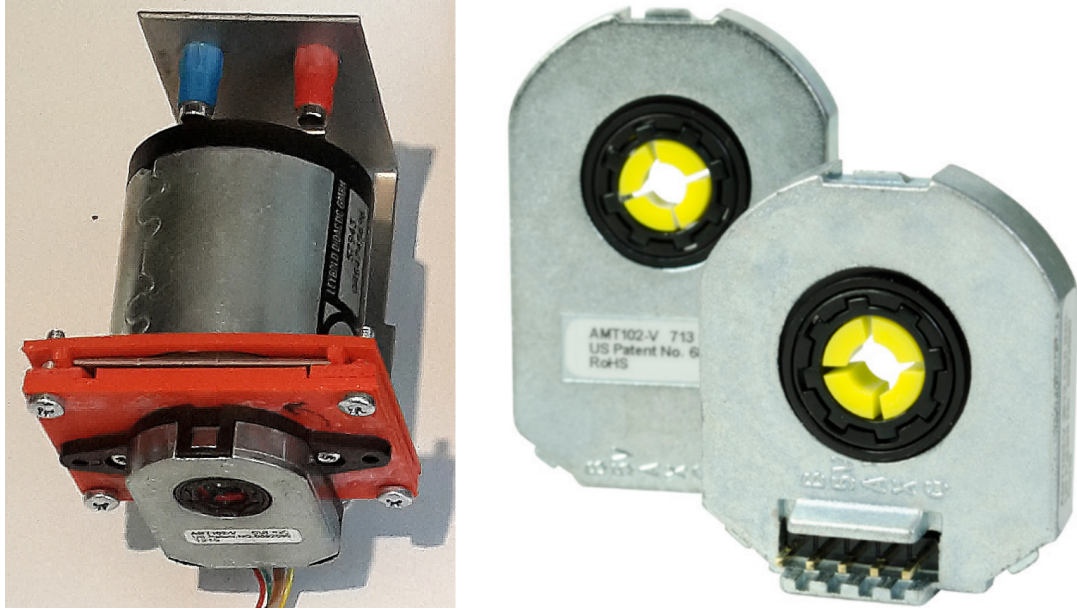
Simülasyonunda kullanılan Maxon F2140-934 çekirdeksiz PMDC motorunun özellikleri Tablo 3.3.'te listelenmiştir.

Tablo 3.3. Maxon F2140-934 motoru özellikleri

Elektriksel Parametreler	Değer	Mekanik Parametreler	Değer
Anma voltajı (V)	12	Tork sabiti (mNm/A)	27.8
Demeraj akımı (A)	1.23	Hız sabiti (rpm/V)	344
Boştaki akımı (mA)	12.3	Mekanik zaman sabiti (ms)	31.9
Anma akım (sürekli) (A)	0.508	Rotor ataleti (gcm ²)	25.3
Verimlilik (max.) %	82	Boştaki hızı (rpm)	4090
Terminal direnci (Ω)	9.73	Anma hızı (rpm)	2430
Terminal endüktansı (mH)	1.27	Anma torku (sürekli) (mNm)	13.9
		Kavrama troku (mNm)	34.2

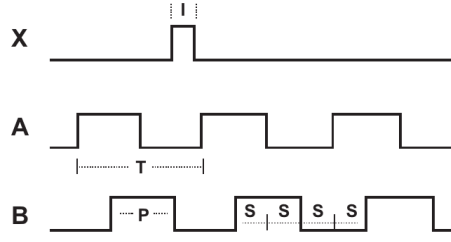
3.5. Arttırımlı Enkoder

Maxon F2140-934 motorunun ön kısmına monte edilebilen AMT102 serisi arttırımlı enkoder index sinyal çıkışı ve 90° faz farkına iki sinyal çıkışı ile standart arayüze sahip bir sensördür. Enkoder yapısı ve motora bağlandığı hali ile Şekil 3.19.'dan incelenebilmektedir.



Şekil 3.19. AMT10 serisi enkoder ve motora bağlantısı

Enkoderin Şekil 3.20.'de görülen çıkış sinyal şekline göre sensörün CPR (Counts Per Revolution) ve PPR (Pulses Per Revolution) okuyabilen ve bunu RPM veya rad/s cinsinden hız değerine dönüştüren donanım modülü FPGA için tasarlanmıştır.

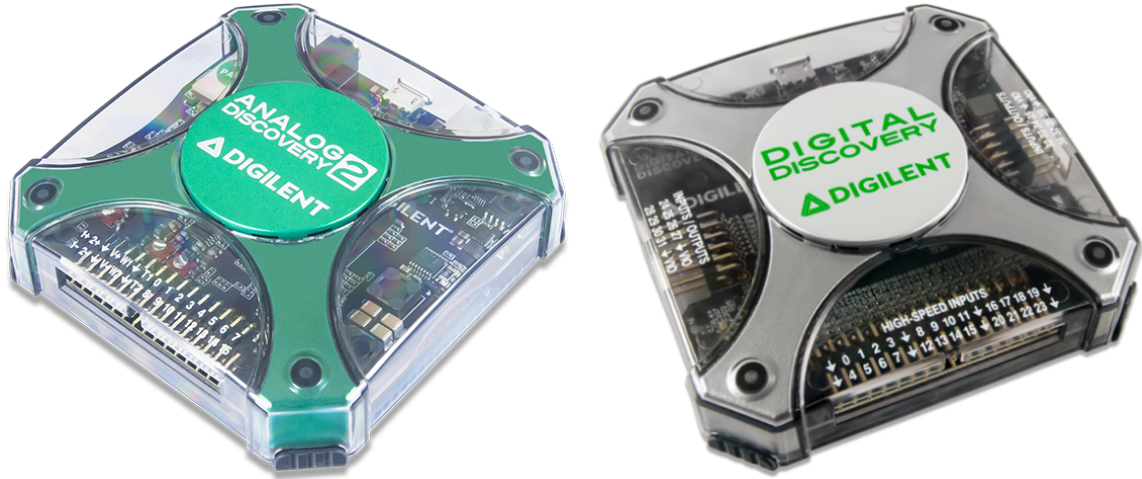


Şekil 3.20. AMT10 serisi enkoder zamanlama çizelgesi

3.6. Analog Discovery 2 ve Digital Discovery USB Osiloskop, Lojik Analizör ve Sinyal Jeneratörü Cihazları

Analog Discovery 2 cihazı FPGA tabanlı bir USB ekipman olup; osiloskop, analizör, sinyal jeneratörü, voltmetre, lojik analizör ve patern jeneratörü olarak çalışabilen çok yönlü bir laboratuvar cihazıdır.

Şekil 3.21.'de gösterilen Analog Discovery 2 cihazı USB arayüzü üzerinden bilgisayara bağlanarak *WaveForms* isimli yazılımı üzerinden kullanılabilmektedir [74, 75].



Şekil 3.21. Analog Discovery 2 ve Digital Discovery USB tipi osiloskop, lojik analizör

Cihaz ayrıca *libdwf* kütüphanesi üzerinden C kodları ile veya aynı kütüphaneyi kullanan *pydwf* isimli kütüphane ile Python üzerinden çağrılarak kullanılabilmektedir.

100 Msps örnekleme değerine sahip olduğundan bu örnekleme kapasitesini konfigürasyon menüsünü kullanarak fonksiyonları arasında nasıl dağıtılacağını seçerek gerekirse bir fonksiyonun çok daha hızlı çalışması sağlanabilmektedir.

Pydwf kütüphanesi üzerinden birden fazla Analog Discovery 2 cihazı seri numaraları belirtilmek sureti ile aynı zamanda kullanılabilmektedir.

Yine FPGA tabanlı bir USB ekipman olan Digital Discovery cihazı analog fonksiyonları olmamasına karşın Analog Discovery 2 cihazına göre çok daha yüksek olan

800 Msps deęerlerinde rnekleme yaparak daha yksek hızlarda lojik analizr olarak kullanılabilir. zellikle ilk ařamalarda FPGA sisteminin geliřtirme ařamasında hatalarını bulabilmek iin bu tez alıřmasında kullanılmıřtır. Yksek hızlarda alıřabilmesi iin yksek hızlı baęlantı adaptr ve kabloları ile birlikte kullanmak gerekmektedir.

řekil 3.21.'de gsterilen Digital Discovery cihazı da USB arayz zerinden bilgisayara baęlanarak *WaveForms* isimli yazılım zerinden kullanılabilir [76].

Digital Discovery cihazı dıřında EK-5'te gsterilen ve bir lojik analizr sisteminin FPGA modl řeklinde yongaya gmlerek alıřtırabilen Signal Tap Logic Analyzer programı, ok daha etkili bir řekilde kullanılabilir.



4. FPGA MODÜL TASARIMLARI

Simülatör tasarımlarından önce FPGA yongasına bağlanan çevre birimlerinin sağlıklı ve doğru çalışabilmeleri için gerekli arayüz niteliğindeki modüllerin hazırlanması gerekmektedir. Tasarlanacak modüller daha sonra tasarıma eklenecek simülatör modülleri için veriyi kararlı şekilde sunmalıdır. Üretilen veriyi yine gerektiği şekilde dış birimlere aktaracak arayüz modülleri oluşturulmalıdır. Modüllerin iletişim kuracakları çevre donanımına uyumlu saat frekansında çalıştırılmaları ve bu saat işaretlerine uygun şekilde resetlenip uygun zamanda da reset durumundan çıkabilmelidirler.

4.1. Saat ve PLL Sistemi

Saat ve PLL sistemlerinin tasarımı ve ayarlanmalarında çevre donanımların çalışma frekansı da göz önüne alınmaktadır. Ayrıca, FPGA yongasında bulunan DSP birimleri ile işlem yapabilmek için yüksek frekanslı saat işaretlerine de ihtiyaç duyulmaktadır.

Geliştirme setinde kullanılan FPGA yongasının hız sınıfı değeri 7 ile belirtilmiştir. Bu değer ilgili yongada bulunan bir makro hücrenin sinyal giriş ve çıkış terminalleri arasında en kötü durum için ölçülebilecek yayılma gecikmesini ifade etmektedir [77]. Verilen değer kullanılarak 7 ns yaklaşık yayılma gecikmesi için hesaplama yapıldığında karşımıza 143 MHz gibi bir azami saat hızı çıkmaktadır. Bu değerın hesaplamalar için net bir değer olmadığına dikkat etmek gerekmektedir. Gerçek sınır frekans değeri, tasarımın sentezlenmesi sonucunda, azami çalışma frekansı raporunda elde edilmektedir. PLL ünitesinden üretilecek saat işaretlerinin 143 MHz değerini aşmaması, ancak işlemlerin örnekleme zamanları arasında bitirilebilmesi için yeterince yüksek olması gerekmektedir.

Sistemde kullanılan ADC yongası veri sayfasında en yüksek iletişim frekansının 40 MHz olduğu bildirilmiştir. Sisteme giren mevcut saat işareti hali hazırda 50 MHz değerindedir. Bu hali ile PLL modülünden üretilecek saat işaretleri için frekans değerlerini belirlemeye çalıştığımızda üretilecek ilk iki saat işareti 40 ve 50 MHz hızlarındaki saat işaretleri olacaktır. Bu iki saat işareti haricinde gerektiğinde kullanılabilecek üzere 143 MHz değerini aşmamak üzere iki saat işareti daha üretilecektir. Bunlardan ilki, kolay şekilde 50 MHz değerinin 2 ile çarpımından üretilebilen 100 MHz değeri ve yine 143 MHz değerinden düşük olan 120 MHz değerleri PLL yapısı kullanarak üretilecek ek saat işaretleri olacaktır. Bu saat işaretlerini üretmek amacıyla, Quartus Prime Lite platformunda PLL ayarlama için sunulan IP modülü kullanılmış ve ilgili tasarım dosyaları oluşturulmuştur.

İlk sentezleme aşamasında `derive_pll_clocks` komutu aktif bırakılmış olup, oluşturulan saat işaretleri ile ilgili mesajlar kopyalanarak kolay kullanılabilir saat isimleri ile SDC kısıtları oluşturularak `derive_pll_clocks` komutu pasif hale alınmıştır.

Modüller arası geçişlerde senkronizasyon çeşitlerinin tanınabilmesi için mevcut bütün saatler `set_clock_groups` komutu ve `-asynchronous` bayrağı ile asenkron olarak tanımlanmışlardır.

4.2. Asenkron Resetleme Sistemi

Asenkron resetleme sisteminde, yongaya giren reset işareti ilk aşamada dışarıdan alınan 50 MHz saat işareti kullanılarak senkronlanmıştır. Bununla birlikte her modül farklı saat işaretini kullanmaya ihtiyaç duyduğundan, modül bazında alınan resetleme sinyali modülün içerisinde kullanılan saat işareti ile çalışan eşzamanlayıcı devreler vasıtasıyla yeniden senkronlama işlemine tâbi tutulmuştur.

Asenkron eeset sinyalinin senkronlama işlemi yapılırken ilgili saat işaretinin hem yükselen hem de düşen kenarlarına senkronlanmış halde iki adet reset sinyali üretilmiştir.

4.3. ADC Modülü

ADC modülü bağlandığı çevre birimi ile seri iletişim kurduğu için modüllerin tasarımı sırasında en çok dikkat gerektiren modül olmuştur.

Şekil 3.13.'te gösterilen zamanlama diyagramı dikkate alındığında modül saatin yükselen ve çıkan kenarlarında çalışacak kısımlara sahip olmalıdır.

Modül parametrelerle kontrol edilebilecek şekilde tasarlanmıştır. Öncelik tanınmış kanalların örnekleme hızının arttırılması amacıyla asimetrik okuma yapılmaktadır. Modül başka bir modülden çağrılırken öncelikli kanal sayısı ve normal kanal sayıları belirtilerek konfigürasyonu otomatik yapılacak şekilde kodlanmıştır.

ADC entegresi 40 MHz de çalışırken bir analog okuma işlemini $2\mu s$ içinde bitirebilmektedir. Çevrimin bitmesinin ardından SPI arayüzü için saat işaretini kabul etmektedir. SPI arayüzü okuduğu değeri seri olarak iletirken aynı zamanda veri girişinden bir sonraki okuma için konfigürasyon bilgisini de almaktadır.

Toplam 8 kanalı olan entegre için kanallar sırası ile;

1. V_S Giriş gerilimi
2. T_L Yük torku
3. R_a Motor direnci
4. L_a Motor endüktansı
5. B_L Yük viskoz sürtünme katsayısı
6. J_L Yük atalet momenti
7. K_M Tork katsayısı
8. K_B Hız katsayısı

olacak şekilde sıralanmıştır. Bu kanallardan V_S ve T_L sinyalleri, motor modelinin giriş sinyalleri olarak kullanıldığından öncelikli kanallar olarak görülmektedir. Diğer giriş kanallarındaki veri değiştiğinde ise bir anlamda modellenen motor veya bağlı bulunduğu mekanik yük değişmiş gibi model parametreleri güncellenmektedir.

Modülün uyguladığı okuma işleminde sırası ile ana okuma çevrimi ve yardımcı okuma çevriminin beraber çalıştığı bir sistem kullanılmaktadır.

Ana okuma çevrimi ADC entegresinin SPI arayüzü ile birlikte çalışan modülün kullandığı asıl çevrimdir. Bu çevrimde öncelikli kanal sayısı ve bir adet normal kanal olacak şekilde bir döngü içinde okuma işlemi yapılır.

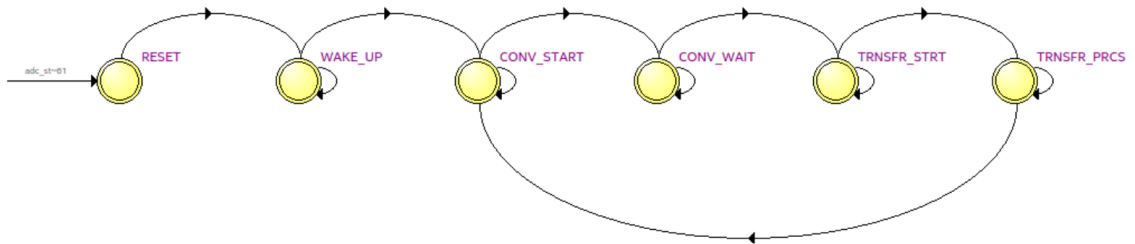
Okuma çevrimi sürekli sıradaki kanalı okuyarak ilerler, sıra öncelikli kanallardan sonra normal kanala gelince ana çevrim dışındaki yardımcı çevrimden bir kanal okunur ve yardımcı çevrim bir adım ilerletilir. Ana çevrim tekrar öncelikli kanalları okur. Daha sonra yardımcı çevrimin normal kanalı okuduğunda yardımcı çevrim bir sonraki kanalın bilgilerini vermiş olur.

Normal okuma yapılmış olsaydı 8 kanal için her biri $2\mu s$ değerinden $16\mu s$ olan bir toplam okuma çevrimi süresi tüm kanallara uygulanmış olurdu. Bu şekilde motor modelinde temel girişler; V_S ve T_L sinyallerinin örnekleme hızı $16\mu s$ olarak kalırdı. Ancak tasarımı yapılan modül (öncelikli kanal sayısı + bir normal kanal) şeklinde çalıştığı ve bu örnekte normal kanal sayısı iki olduğu için toplamda $6\mu s$ olan bir okuma hızında V_S ve T_L sinyallerini okuyabilmektedir. Bu şekilde öncelikli kanalların sinyallerinin örnekleme frekansı 2,6 kat arttırılabilmektedir. Bu işlemin yan etkisi normal kanalların her birinin ana okuma çevriminde sırayla okunmalarıdır. Buradaki örnekte normal kanal sayısı 6 adet olduğundan ve ana çevrim süresi $6\mu s$ olduğundan normal bir kanalın örnekleme süresi $36\mu s$ değerine yükselmektedir. Bu okuma yöntemi normal kanalların örnekleme hızını 2,25 kat azaltmaktadır. Bu asimetrik okuma sayesinde öncelikli addedilen kanalların örnekleme frekansı arttırılarak sistemin yanıt süresi iyileştirilmeye çalışılmıştır.

İki motorun simüle edildiği modellerde öncelikli kanal sayısı 4 olarak atanmakta geri kalan 4 kanal normal olarak kalmaktadır. Bu durum için öncelikli kanalların örnekleme hızı $10\mu s$ olurken normal kanalların örnekleme hızı $40\mu s$ değerine çıkmaktadır.

Elbette öncelikli kanal sayısının arttırılması sağlanan avantajın azalmasına sebep olmaktadır. Ancak son örnekte bile sistem örnekleme hızını %62 oranında iyileştirebilmektedir.

Normal kanallar temsil ettikleri değerlere bakılarak hızlı örneklenmesine gerek olmayan kanallardır.



Şekil 4.1. ADC entegresinin okunmasını sağlayan modülün durum makinesi

Şekil 4.1.'de gösterilen diyagramda ADC entegrasyonunun okunması için kullanılan modülün durum makinesi görülmektedir. Burada resetleme işaretinin bitiminin ardından entegrenin ihtiyacı olan toparlanma süresi kadar beklenmekte, sonrasında normal okuma çevrimine girilmektedir. Normal okuma çevrimi analog sinyalin örneklenmesi ile başlamaktadır. Analog örnekleme işlemi sonrasında entegrenin kendi iç sisteminde bulunan saat ile işlettiği analog-dijital çevrim süreci beklenmektedir. Çevrim sürecinin bitiminde modül okunan değerin ilgili olduğu kanal adresini ve bir sonraki analog-dijital çevrim için entegreye aktarılacak konfigürasyon verisini oluştururken kendi içinde bir sonraki çevrim için gerekli hazırlıkları yaptığı transfer başlangıç adımını icra etmektedir. Transferin başlatılması işleminden sonra veri aktarım işlemi gerçekleştirilmektedir. Veri transferinin bitimine müteakip, sıradaki okunacak analog girişin değeri örneklenerek çevrim devam ettirilmektedir.

4.4. Simülatör Modülleri

Simülatör modüllerinin durumları genel olarak daha sonra değinilecek bir konu olmasına karşın bu bölümde simülatör modüllerinde kullanılan kayar nokta hesaplama modüllerine ve simülatör modüllerinin genel çalışma sistemine değinilecektir.

Simülasyonun yapıldığı modüllere değerler işaretli tam sayı olarak veya işareti var ise bile işaretli tam sayı yerine bir ofset değeri ile kaydırılmış şekli ile yine işaretli tam sayı olarak gelmektedirler. Bu sebeple sayı tipinin çevrildiği modüllerin dikkatle konfigürasyonu gereklidir.

Modülün hesaplama ile bulduğu değerler dışarıya aktarılırken işaretli tam sayı değerleri olarak ve kullanılacak bit sayısı kadar bir hassasiyetle aktarılmaktadırlar.

Gelen değerler önce kayar nokta değerlere dönüştürülmektedir. Bu dönüşümden sonra işlemler 32 bit genişliğinde ve IEEE-754 kayar nokta standardında çalışan modüller yardımı ile gerçekleştirilmektedir.

Verilog donanım tanımlama dilinde temelde 4 işlem için operatörler mevcuttur ve gerektiğinde çarpma ve bölme işlemleri sentezleme işleminde modüllere dönüştürülerek, toplama ve çıkarma işlemleri ise doğrudan icra edilmektedirler. Ancak hesaplama kayar nokta standardı ile yapıldığında işlemlerin artık Verilog işlem operatörleri yerine IP kütüphanesi modüllerine bağlantılar şeklinde yapılması gerekmektedir. İşlem için operand değerlerin veri hatlarının yapılacak işleme uygun modülün girişlerine bağlantılanması ve sonucun çıkış portundan uygun genişlikte veri hattı ile alınması gerekmektedir.

Bu bağlantıların yapılmasının ardından işlem modülü girişlerinde operand değerlerin mevcut olduğundan emin olunmalı ve işlem modülüne, modül için belirtilen sayıda saat işareti uygulamak ve sonrasında oluşan sonuç değerini korumak gerekmektedir. Pratik olarak bu işlem saat işareti yerine saat izin işareti girişinden uygulanan sinyal ile yapılmaktadır.

Simülatör içerisinde çalıştırılan kayar nokta modüllerinin saat izin girişlerine gerekli sinyalleri vermek üzere ilgili simülatör modülünün tamamındaki kayar nokta modülleri için gerekli sinyalleri oluşturan bir global kontrol modülü tasarlanıp çalıştırılmaktadır.

Simülatörde herhangi bir değişiklik yapılacağı zaman hem kayar nokta modülleri hem de verilerin taşındığı hatların değiştirilmesi gerekmektedir. Aynı zamanda işlem sırasını kontrol eden global kontrol ünitesi, işlem sıralamasında değişikliğin yapıldığı notadan itibaren yeniden düzenlenmeye gereksinim duymaktadır.

Simülatör içerisinde oluşan değerlerden bir sonraki hesaplama adımında kullanılacak olan değerler için değerın üretildiğı modülün saat izninin kesilmesi yeterlidir. Ancak bir sonraki hesaplama çevriminde kullanılacak değerlerin uygun genişlikteki kaydedicilerde saklanması ve bu kaydedicinin de global kontrol birimi tarafından kontrol edilmesi gerekmektedir.

4.4.1. Modül Giriş ve Çıkış Prosedürleri

Simülatör ve simülatör için parametre değerlerinin oluşturulduğu modüllerde kayar nokta olmayan veri giriş ve çıkışlarında izlenen işlem dizisi bulunmaktadır.

Modül Giriş Prosedürleri

Modüle giriş yapan tüm kayar nokta dışı değerler işaretli tamsayı olarak ele alınmaktadır.

Veri girişı işlem akışı sırasıyla;

- *Giriş Senkronizasyonu*; Bu adımda modüle giriş yapan veri hattı modül işlem saatine senkronlandırılır. Bu adımda giriş ve çıkış işaretli tam sayı tipindedir.
- *Giriş Tamponlama Birimi*; Bu adımda senkronlanmış işaret modülün örnekleme sinyali ile birlikte mandallanmaktadır. Ayrıca bu adımda işaretli olduğu halde bir offset değeri bulunduran girişler bir negatif offset değeri kullanılarak ikinin tümleyeni standardında işaretli tam sayıya dönüştürölmektedirler. Giriş ve çıkış bit sayısı değişmemektedir.
- *Giriş Kayar Nokta Dönüşümü*; Bu adımda veri işaretli bir tam sayı formundadır ve IP kütüphanesinden dönüşüm için konfigüre edilmiş bir dönüştürücü vasıtası ile kayar nokta yapısına dönüştürölmektedir. Bu adım global kontrol birimi tarafından kontrol edilen ilk işlem basamağıdır ve çıkış hattı 32 bit değerindedir. Dönüşüm yaparken IP bloğunun ayarlarına göre bir ölçekleme işlemi de yapılmış olur.
- *Giriş Ölçekleme Adımı*; Bu adımda girişı yapılan kayar nokta değeri ADC girişinde kabul edilen mühendislik as veya üs katlarına göre ölçeklendirilmektedir.
- *Giriş Değer Seçimi*; Giriş kısmında okunan değeri ve sabit değeri olmak üzere iki tip değeri alınmaktadır. Bu değerler seçme anahtarları girişlerine göre toplama işlemine tabi tutularak tek bir işlem değeri elde edilmektedir.

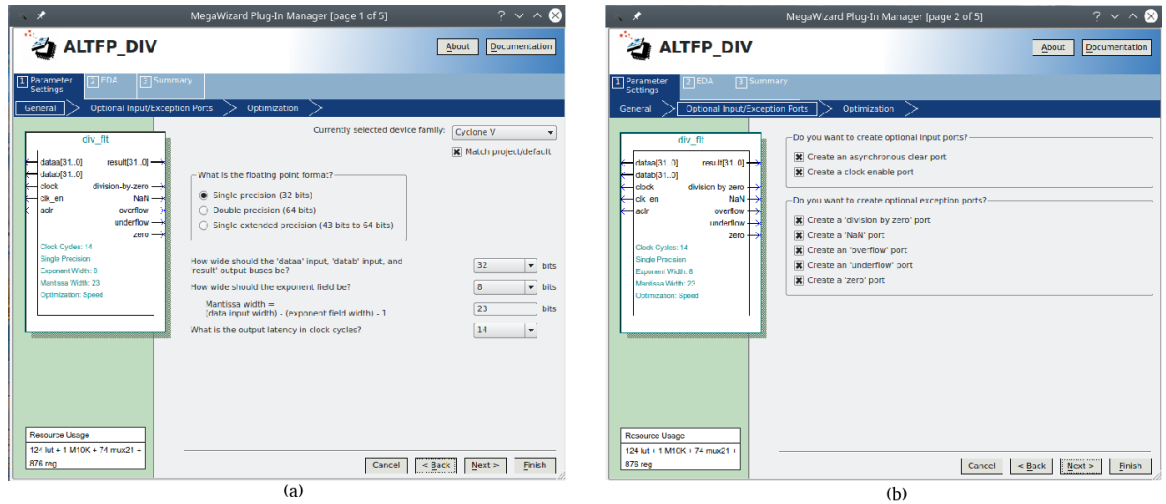
Modül Çıkış Prosedürleri

Modülden çıkan ve simülatör sonuçlarından sayılan tüm veri hatları için uygulanan çıkış prosedürleri sırası aşağıda verilmiştir.

- *Çıkış Sabit Nokta Dönüşümü*; Bu adımda kayar nokta tipinde giriş yapan değerler sabit noktalı gösterim tipinde değerlere dönüştürülmektedirler. Dönüşüm IP kütüphanesinden kullanılarak ilgili veri için ayarlanan bir modüldür tarafından yapılmaktadır. Veri için oluşabilecek değer aralığına göre sabit noktanın yeri ve varsa işaret biti eklenerek çıkışta sabit nokta gösterim tipine geçilir. Bu adımda IP bloğunun ayarları ile ölçekleme yapılmış olmaktadır. Bu adımda çıkış veri boyutu DAC birimi veri genişliğindedir.
- *Çıkış Tamponlama Birimi*; Bu adımda alınan sabit nokta gösterim tipine sahip veri sayet işaretli yapıda ise işaret biti terslenir ve oluşan değer bir offset değeri kadar kaydırılarak işaretli tam sayı gösterim tipine dönüştürülmektedir. Bu adımın ardından veri DAC birimine aktarılmaya hazır hale getirilmektedir.

4.4.2. Kayar Nokta İşlem Birimleri

Kayar nokta işlem birimleri Quartus Prime geliştirme platformunda IP katalog kısmından bulunarak çağrılıp konfigüre edilir ve bu konfigürasyonla birlikte kullanılacak modül dosyaları istenen yere kaydedilir. Kaydedilmiş bu modül gerektiği sayıda kullanılabilir ancak aynı işlem farklı parametreler ile konfigüre edilmiş bir modülde işlenmesi gerekiyor ise o modülün de oluşturulması gerekmektedir.



Şekil 4.2. Kayar nokta bölme işlemi IP bloğu için a) genel konfigürasyon penceresi b) çıkış portlarının ayarlanması

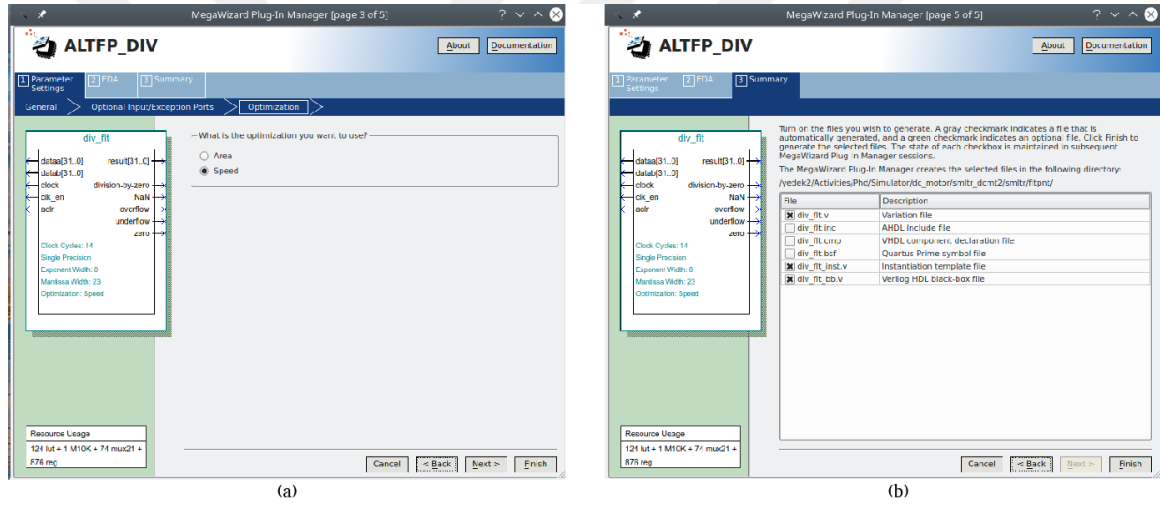
Burada örnek olarak bir kayar nokta bölme modülünün oluşturulması incelenecektir. EK-3'te görülen uygulama penceresinde sağ tarafta bulunan IP katalog kısmındaki ağaç

yapısından seçilen blok için konfigürasyon penceresi açılacaktır. Bu işlemler ardından oluşturulan dosyaların projeye dahil edilmesi gerekmektedir.

Şekil 4.2.'de a) kısmında görülen konfigürasyon penceresi, konfigürasyon aşamaları ilerledikçe sonraki forma geçiş yapmaktadır. Ayrıca konfigürasyon bilgisi oluştukça bloğun kullanacağı kaynak miktarı ve çeşidi de sol alt kısmında izlenebilmektedir. Bu pencerede modülün kaç bitlik olacağı ayarlanmaktadır.

Gerektiğinde 32 veya 64 bit olan standart genişlikler dışında da modüller konfigüre edilebilmektedir. Bu sayede fazla kaynak harcamadan arzu edilen duyarlılığa ulaşılabilmektedir.

Şekil 4.2.'de b) kısmında görülen konfigürasyon penceresinde modüle eklenecek yardımcı sinyal giriş ve çıkışları ayarlanmaktadır. Özellikle hata durumlarının anlaşılması için ilgili çıkışın mevcut olması gerekmektedir. Aksi halde sadece veri çıkışına bakılacak olursa problemin anlaşılması büyük çaba gerektirmektedir.



Şekil 4.3. Kayar nokta bölme işlemi IP bloğu için a) modül optimizasyon tipinin seçimi b) modül için üretilecek dosyaların seçilmesi

Şekil 4.3.'te a) kısmında görülen pencerede ise modüle uygulanacak optimizasyon tipi belirtilmektedir. Bura yapılacak ödünleşme ile sistem kaynak kullanımı ve hızı arasında gerekli seçim yapılmalıdır.

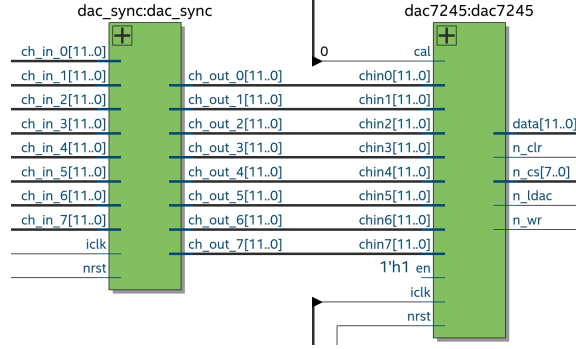
Şekil 4.3.'te b) kısmında görülen son pencerede ise oluşturulan modül için üretilecek dosyalar seçilmektedir. Burada sistemde kullanılan donanım tanımlama diline uygun dosyalar seçilerek projeye kullanıma sunulmaktadır.

4.5. DAC Modülü

DAC entegreleri paralel arayüz bağlantı tipini kullanarak FPGA yongasına bağlanmaktadır. DAC entegrelerini kontrol eden donanım modülü de parametrik olarak değişebilecek şekilde tasarlanmıştır. parametre değeri verilerek 8 e kadar istenen sayıda

aynı tip DAC entegresi aynı veri yolu üzerinden kontrol edilebilmektedir. Farklı bir entegre için ise modülün **localparam** değerleri uygun seviyelere getirilmelidir.

Modül entegre seçme pini ($\overline{CS}$) hariç diğer pinleri ortak kullanılmaktadır. ($\overline{CS}$) pini her entegre için ayrı çıkartılmaktadır.



Şekil 4.4. DAC kontrol modülü ve girişini kontrol eden veri senkronizasyon modülü

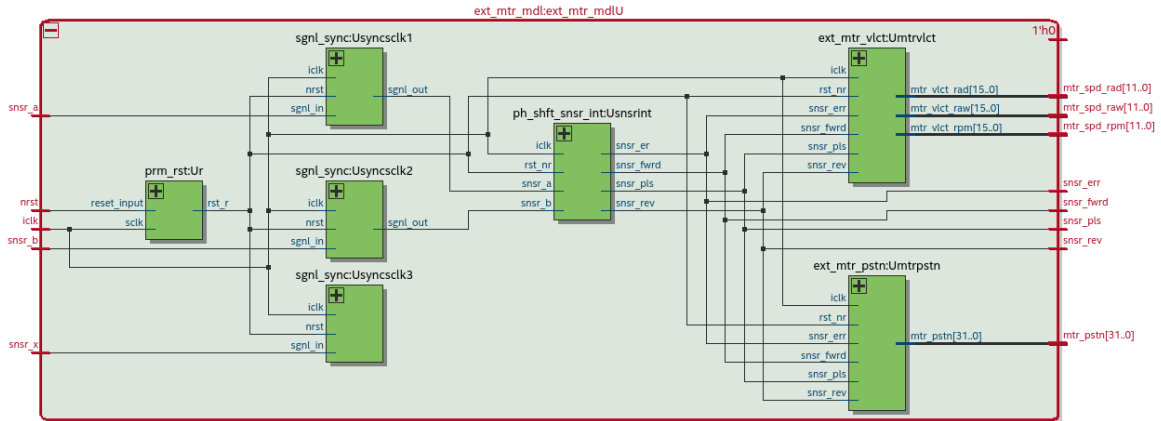
Modül 50 MHz saat işareti ile çalıştırılmaktadır ve Şekil 3.17.'de gösterilmiş olan zamanlamalara uyacak şekilde sıra ile giriş kanallarındaki verileri ilgili DAC entegresinin tamponuna yükleyerek çıkışlarına izin vermektedir.

Modül çalıştığı saat işareti ve çevre birimi olarak verinin üretildiği modüller ile senkron çalışmadığı için bağlanan veri hatlarının tümünün senkronizasyonu için Şekil 4.4.'te görülen ayrı bir modül ile birlikte kullanılmaktadır.

Bu tarz modüller donanım tasarımlarında farklı saat bölgeleri arası geçişleri sağlayan senkronizasyon yapılarıdır.

4.6. Harici Motor Veri Okuma Modülü

Motor simülatörünün karşılaştırmalı deneyleri yapılırken motora bağlanmış arttırımlı enkoder sinyallerini okuyarak işleyen modül Şekil 4.5.'ten incelenebilir.



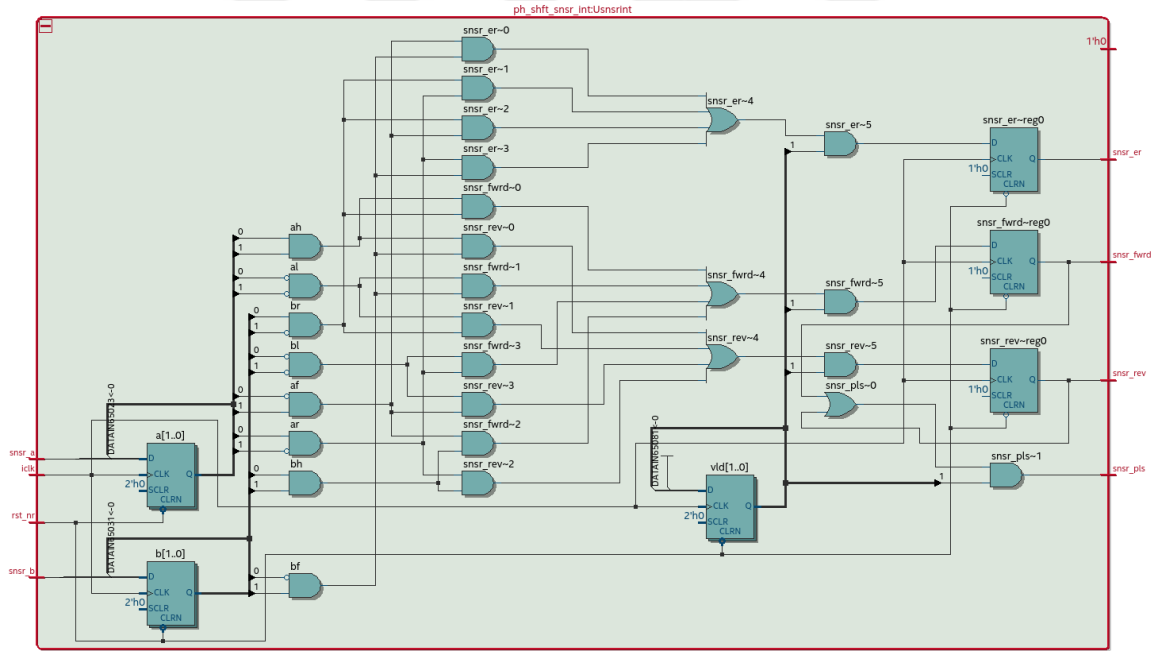
Şekil 4.5. Harici motor için hız ve konum hesaplama modülü

Şekilde sol tarafta bulunan modüller gelen sinyallerin senkronizasyonu için kullanılan modüllerdir, ortadaki modül senkronlanmış enkoder sinyallerini okuyarak motorun dönüş yönünü ve CPR tipinde dönüş miktarını vermektedir. Şeklin solunda kalan modüller ise oluşan bilgilerden motorun hızını ve pozisyonunu oluşturan modüllerdir.

Motorun ölçülen CPR cinsinden hız değeri matematiksel işlemler ile rpm ve rad/s değerlerine dönüştürülmektedir. Hesaplama şekli olarak motorun PPR darbeleri arasında kalan saat sinyalini sayarak ölçüm yapmaktadır. Hassasiyetin yüksek olması için 100 MHz saat işareti ile işlenmiştir. Motor dönüş yönünde değişme gördüğünde hemen önce hız verisini resetlemekte ardından yeniden saat darbelerini saymaktadır. Bu izah edilen çalışma şekli sebebiyle hız bilgisinin oluşması için ard arda aynı yönlü en az iki PPR sinyali gerekmektedir. Enkoderin hassasiyeti tur başına 2048 PPR olduğundan bu gereksinimin problem oluşturma ihtimali bulunmamaktadır.

4.6.1. Sensör Okuma Sistemi

Enkoder sensörünü okuyan modül Şekil 4.6.'da görülmektedir. Modül PPR değerini esas alacak şekilde tasarlanmış olup, dönüş palsi ve yönünü bilgi olarak dışarı aktarmakta ve sensörlerin vermemesi gereken bilgiler görüldüğünde hata çıkışı üretmektedir.



Şekil 4.6. Harici motor için enkoder verisi işleme modülü

Her sensör fazının mevcut ve önceki durumlarını karşılaştırarak CPR düzeyinde dönüş darbesi ve yönü bilgisi oluşturmaktadır.

4.6.2. Açısal Hız Hesaplama Sistemi

Açısal hız değerini ölçen modül CPR işaretleri arasında oluşan 100 MHz saat darbelerini saymakta ve matematiksel işlemlerden sonra farklı katsayılarla çarparak ayrı kanallardan hem rpm cinsinden hem de rad/s cinsinden hız verisini dışarı aktarmaktadır. Modül farklı hassasiyette enkoderler ile kullanılabilir diye modül içerisinde `localparam` değişkenleri ile kontrol edilebilmektedir. Modülün çağrıldığı yerde farklı saat frekanslarında çalışabilmesi için çalışma saat frekansı çağrıldığı yerde parametre olarak girilebilmektedir.



5. SİMÜLATÖR TASARIMLARI

Simülasyon için yapılan donanımsal hazırlıklardan sonra geriye simülatörümüzün matematiksel modeline göre sistemin donanıma aktarılması işlemi kalmaktadır. Bu işlem iki farklı yolla yapılabilmektedir.

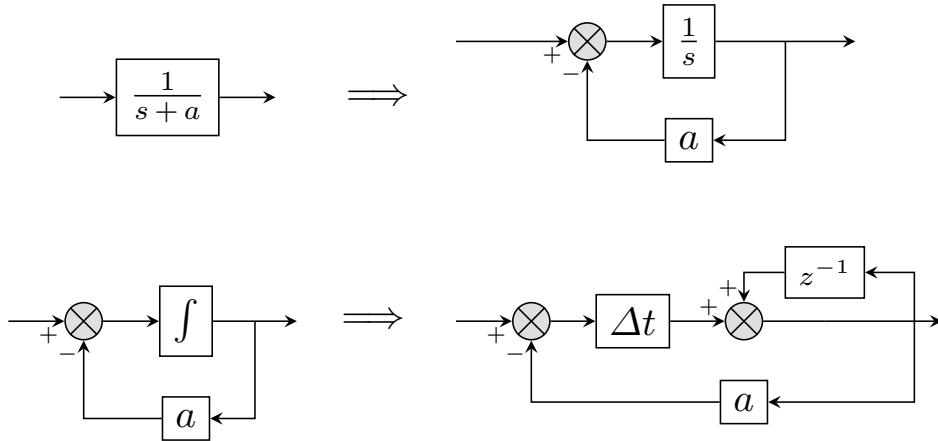
Birinci yöntemde fiziksel model incelenerek çıkış ifadesi için denklem çözümü oluşturulmaktadır. Bu işlem için giriş fonksiyonu sabit şekilde örnekleme zamanları arasında basamak giriş fonksiyonu gibi düşünülmektedir.

$$Y(s) = U(s) \cdot G(s) \quad \rightarrow \quad Y(s) = \frac{1}{s} \cdot G(s) \quad (5.1)$$

$$y(t) = \mathcal{L}^{-1} \left\{ \frac{1}{s} \cdot G(s) \right\} \quad (5.2)$$

Basamak giriş ifadesi sistemin transfer fonksiyonu ile çarpılarak Denklem 5. 1'de gösterildiği gibi çıkış ifadesi bulunur ve bu ifadenin ters Laplace dönüşümü kullanılarak Denklem 5. 2 ile zaman bölgesinde yazılmaktadır. Bu ifade işlem adımlarına dönüştürülerek adım adım donanımsal olarak gerçekleştirilmektedir.

İkinci bir yöntem olarak modelin transfer fonksiyonu doğrudan donanıma gömülerek yapılmaktadır. Bu işlem için sistemin blok diyagramı Şekil 5.1.'de betimlenen dönüşümler kullanılarak donanımsal olarak çalıştırılabilecek forma dönüştürülür ve ardından hesaplama adımları donanımda adım adım gerçekleştirilmektedir.



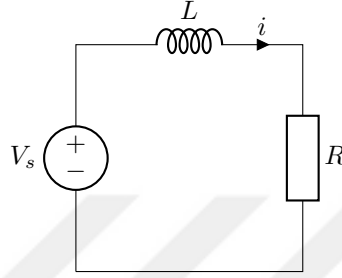
Şekil 5.1. Blok diyagramlarından nümerik hesaplama yapmak için gerekli dönüşümler

5.1. RL Devre Simülatörü

İlk simülatör tasarımı basit şekilde bir RL devresi için yapılmıştır.

5.1.1. Sistem Modeli

Elektrik makinaları modellerinde endüktans ve direnç yoğun şekilde kullanıldığı için simülatör tasarımlarına RL devresi ile başlanmıştır.



Şekil 5.2. Basit RL devresi şeması

Şekil 5.2.'de gösterilen devredeki sistemin diferansiyel denklemi Denklem Grubu 5. 3'te gösterildiği şekilde Laplace dönüşümü ile yazılıp çözülmektedir.

$$\begin{aligned} V_s(t) &= R \cdot i(t) + L \cdot \frac{d}{dt}i(t) \\ V(s) &= R \cdot I(s) + L[s \cdot I(s) - i(0^-)] \\ V(s) &= I(s)(sL + R) - L \cdot i(0^-) \\ I(s) &= \frac{1}{sL + R}V(s) + \frac{L}{sL + R}i(0^-) \end{aligned} \quad (5.3)$$

Simülasyon adımları arasında kaynak gerilimi, değeri V olan basamak fonksiyonu şeklinde düşünülmektedir. Laplace formunda giriş voltaj ifadesi de yazılarak çıkış ifadesi Denklem 5. 4'te gösterilen hale dönüştürülmektedir.

$$V(s) = \frac{V}{s} \quad \rightarrow \quad I(s) = \frac{V}{R} \left[\frac{1}{s} - \frac{1}{s + \frac{R}{L}} \right] + \frac{1}{s + \frac{R}{L}} i(0^-) \quad (5.4)$$

Denklem 5. 4'ün ters Laplace dönüşümü yardımıyla Denklem 5. 5'te gösterilen zaman bölgesi formuna dönüşmektedir.

$$I(t) = \frac{V}{R} \left[1 - e^{-\frac{R}{L}t} \right] + i(0^-) e^{-\frac{R}{L}t} \quad (5.5)$$

Elde edilen Denklem 5. 5 ifadesinde t değeri için simülasyon adım süresi olan Δt değeri kullanılarak Denklem 5. 6'da görülen formu ile donanımsal olarak gerçekleştirilebilmektedir.

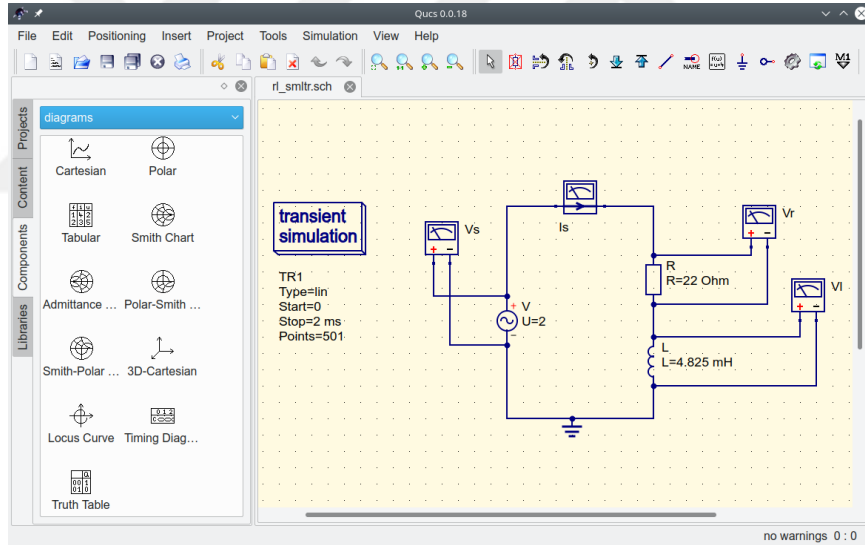
$$I(t) = \frac{V}{R} \left[1 - e^{-\frac{R}{L} \Delta t} \right] + i(0^-) e^{-\frac{R}{L} \Delta t} \quad (5. 6)$$

5.1.2. Bilgisayar Simülasyonu

Bilgisayar simülasyonlarında Qucs, OpenModelica ve GNU Octave programları kullanılmıştır. Örnek simülasyonlarda her üç program için aynı devre parametreleri ile birlikte $4\mu s$ lik adım aralıkları ile devrenin 2ms süre boyunca çalışması simüle edilmiştir.

Qucs Programı ile Simülasyon

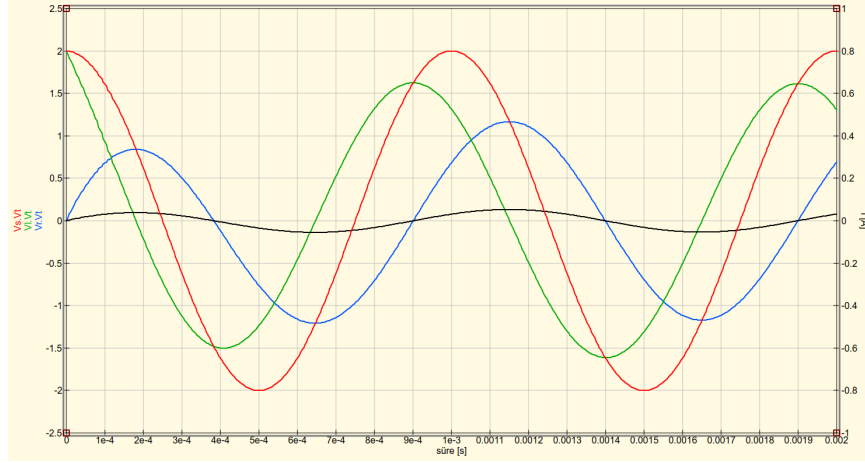
Qucs Quite Universal Circuit Simulator kelimelerinin baş harflerinden oluşmuş bir isimdir. Simülasyon programın Şekil 5.3.'te görülen arayüzden devre ve simülasyon parametrelerinin girilmesi ile yapılabilmektedir.



Şekil 5.3. Qucs simülasyon programı arayüzü ve oluşturulan devre

Qucs devreleri modellemek için tasarlandığından elektro mekanik elemanların modelleme işleminde SPICE modellerinin sağlanması gerekmektedir. Yapılacak simülasyon için devre şeması, simülasyonun modülü ve parametre girişinin ardından kaydedilen değerler çizim ekranında farklı diyagram tipleri seçilerek çizilebilmektedir.

Program arka planda kendi simülasyon motorunu kullansa bile ayarlanırsa SPICE programını da kullanabilmektedir.

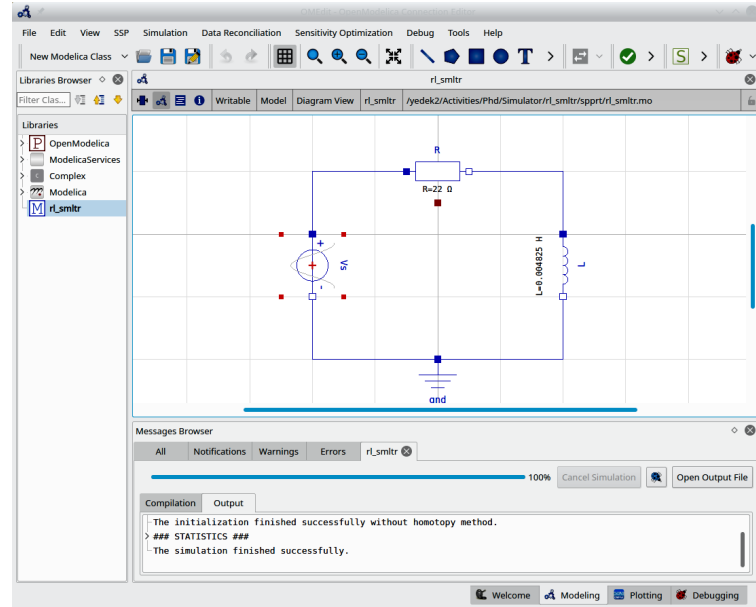


Şekil 5.4. Qucs programı RL devresi simülasyon sonuçları

Şekil 5.4.'te görülen simülasyon sonuçları devrede $V_S(t)$, $V_L(t)$, $V_R(t)$ ve $I(t)$ değerlerinin grafiğini göstermektedir.

OpenModelica Programı ile Simülasyonu

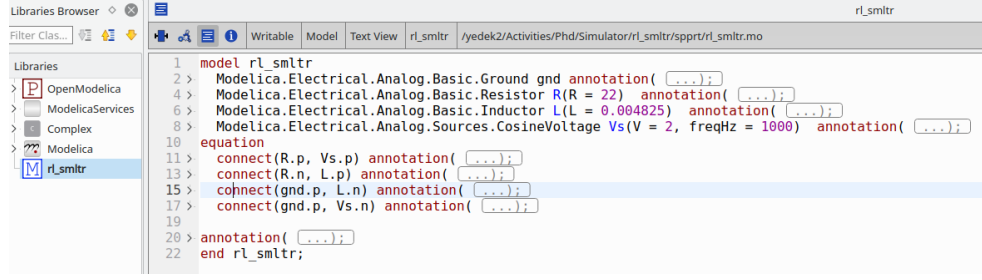
OpenModelica programı nesne tabanlı bildirimsel yapıdaki modelleme dillerinden Modelica temel alınarak geliştirilmiş bir açık kaynak kodlu modelleme ve simülasyon ortamıdır.



Şekil 5.5. OpenModelica simülasyon programı arayüzü ve oluşturulan sistem modeli

Modellemede eleman veya bileşen bazlı modelleme olarak bilinen yaklaşımla karmaşık yapıdaki sistemleri modelleyebilmektedir. Programın Şekil 5.5.'te gösterilmiş

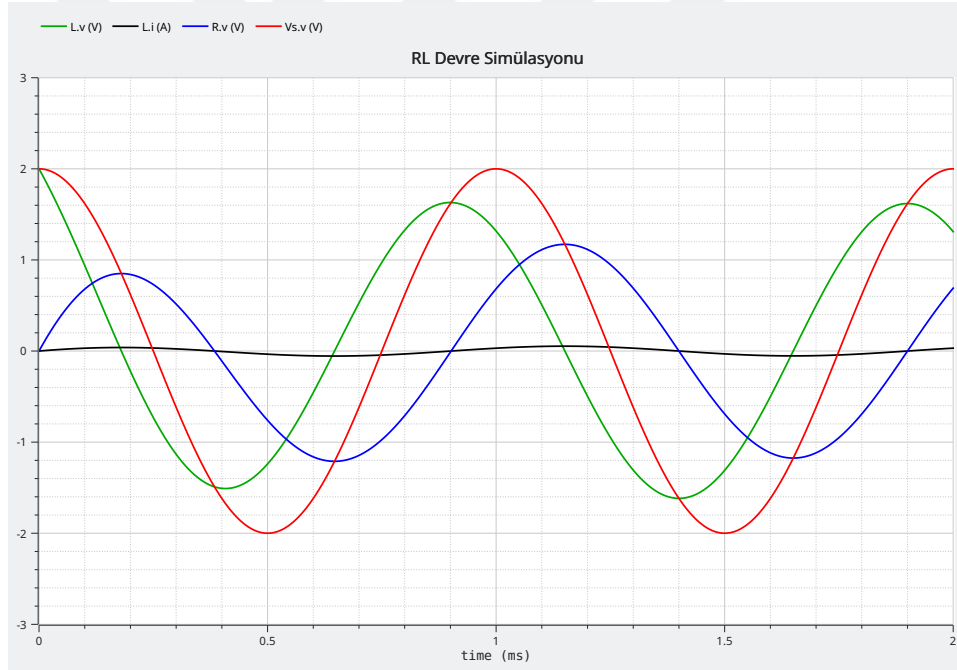
modelleme arayüzünde devrenin simülasyon için yapılan sistem modeli görülmektedir.



```
1 model rl_smltr
2 > Modelica.Electrical.Analog.Basic.Ground gnd annotation( ...);
4 > Modelica.Electrical.Analog.Basic.Resistor R(R = 22) annotation( ...);
6 > Modelica.Electrical.Analog.Basic.Inductor L(L = 0.004825) annotation( ...);
8 > Modelica.Electrical.Analog.Sources.CosineVoltage Vs(V = 2, freqHz = 1000) annotation( ...);
10 equation
11 > connect(R.p, Vs.p) annotation( ...);
13 > connect(R.n, L.p) annotation( ...);
15 > connect(gnd.p, L.n) annotation( ...);
17 > connect(gnd.p, Vs.n) annotation( ...);
19
20 > annotation( ...);
22 end rl_smltr;
```

Şekil 5.6. OpenModelica programında kullanılan Modelica simülasyon dili

Modelde kullanılan elemanlar Modelica dilinde kodlanmış fonksiyonlar halinde Şekil 5.6.'da görülen model dosyasında tutulmaktadır. İstenirse bu dosya ile model değiştirilerek de simülasyonlar yapılabilir.



Şekil 5.7. OpenModelica simülasyon sonuçlarının gösterimi

Sistem şemasının oluşturulmasının ardından simülasyon parametreleri ayarlanarak simülasyon gerçekleştirilir sonrasında çizim modülünden istenen değişkenler seçilerek Şekil 5.7.'de görüldüğü şekilde sonuçlar çizdirilir..

Tablo 5.1. OpenModelica programı ile basit RL devresi simülasyonu aşamaları boyunca harcanan süreler.

Simülasyon Aşaması	Yüzdelik dilim	Süre [ms]
init.xml dosyasının okunması		1.83852
info.xml dosyasının okunması		0.157879
Ön ayarlar	% 3.3	0.218573
ayarlar	% 1.3	0.088002
Adımlar	% 0.1	0.00937
Çözümleyici	% 8.4	0.560408
Çıkış dosyası oluşturma	% 5.6	0.369841
İşlem yorumlama	% 4.4	0.290692
Ön işlem	% 2.0	0.130205
Simülasyon	% 75.0	4.99195
Toplam	% 100.0	6.65904

OpenModelica programı simülasyonla ilgili tuttuğu kayıt dosyalarında simülasyon aşamalarında geçen süreyi kaydetmektedir. Basit bir RL devresi simülasyonunda harcanan zaman Tablo 5.1.'den incelene bilmektedir. Tablo incelendiğinde sadece simülasyon için harcanan süreye yaklaşık 5ms değeri ile gerçek zamanının 2,5 kat üzerine çıkmaktadır.

GNU Octave Programı ile Simülasyonu

GNU Octave programı, MathWorks firmasının Matlab numerik hesaplama programına benzer şekilde sayısal hesaplama yapmak amacıyla açık kaynak olarak geliştirilmiştir. Program girdi dosyası olarak Matlab programı gibi .m betik dosyaları kullanacak şekilde oluşturulmuş ve büyük ölçüde kodlama konusunda benzerlikler göstermektedirler.

Farklı işlerde kullanılmak üzere paket dosyaları bulunmakta ve bunlar ücretsiz olarak indirilerek kullanılabilir.

Bu tez çalışmalarında genel olarak kullanılan simülasyon programı Octave olmuştur. Bunun sebebi donanımsal simülatörde yapılacak sayısal işlemlerin, öncesinde yazılımsal olarak bu programda denenebilmesidir. Denemelerde zaman zaman belli hataların ayıklanabilmesi için LibreOffice Calc programı da beraberce kullanılmıştır.

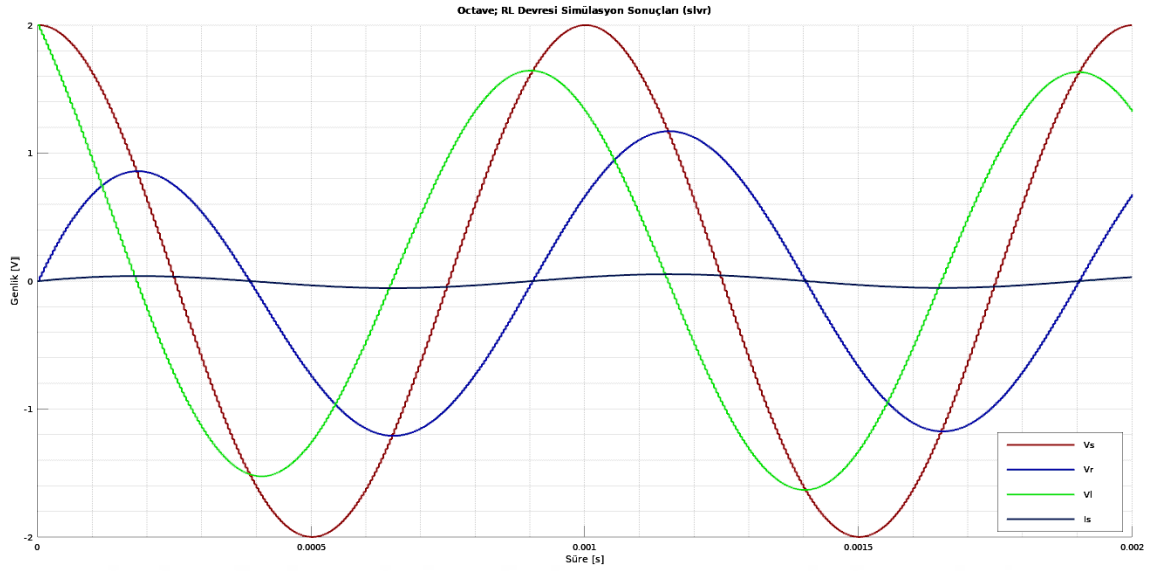
Donanımsal simülatör tasarlamak için kullandığımız iki yöntemin RL devresi için kodlamaları yapılarak Octave programında test edilmiş ve aynı sonuçlar alınmıştır.

İlk olarak RL devresinin denklem çözümü yardımı ile simülasyonu için kodlama yapılmıştır. Denklem 5. 6'da belirtilen ifade aşağıdaki listede bulunan kod parçası şeklinde kodlanarak çalıştırılmıştır. Sonuçlar Şekil 5.8.'de gösterilmektedir.

```

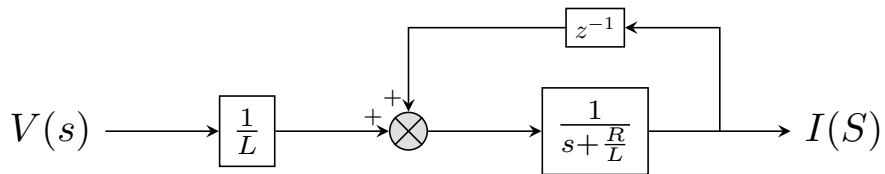
srvl1 = e^( (-R/L)*dt );
srvl2 = (1-srvl1)/R;
t0 = clock ();
for j=2:divide(t_end+dt,dt)
    i(j) = v(j-1)*srvl2 + i(j-1)*srvl1 ;
end for
vr = R * i ;
vl = v - vr ;
elapsed_time = etime (clock (), t0);

```



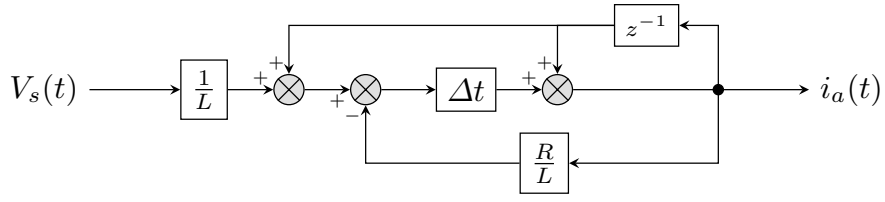
Şekil 5.8. Octave ile denklem çözüm yöntemi kullanılarak elde edilen sonuçlarının gösterimi

RL devresinin ayrıca blok diyagramından faydalanılarak modelin doğrudan kodlanması yöntemi kullanılarak ikinci bir simülasyon yapılmıştır. Bu işlem için RL devresinin ilk durum değerini de içeren Şekil 5.9.'da görülen blok diyagramından yola çıkılmıştır.



Şekil 5.9. RL devresinin ilk durum değerini de içeren blok diyagramı

Şekil 5.1.'de gösterilen dönüşümler kullanılarak blok diyagramı Şekil 5.10.'da gösterilen forma getirilmiştir. Bu blok diyagramından faydalanılarak Denklem 5. 7 elde edilmiştir.



Şekil 5.10. RL devresinin model yaklaşımı ile simülasyon tasarımı için blok diyagramı

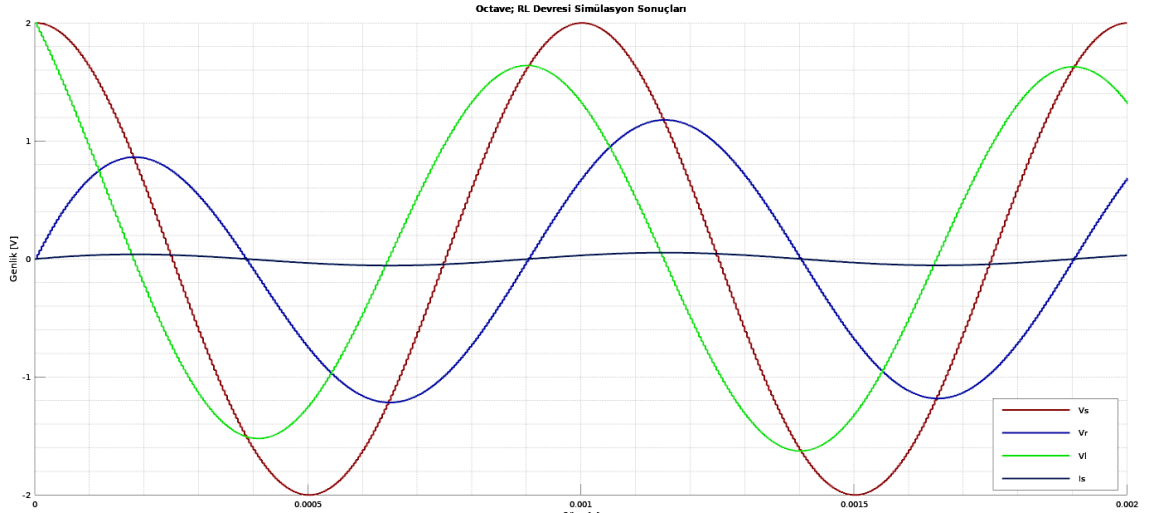
$$I(t) = I(t - \Delta t) + \Delta t \cdot \left[\left(\frac{V}{L} + I(t - \Delta t) \right) - \frac{R}{L} \cdot I(t - \Delta t) \right] \quad (5.7)$$

Elde edilen denklemden yola çıkılarak Octave kodları aşağıda gösterildiği şekli ile değiştirilmiştir.

```

srvl1 = dt/L;
srvl2 = 1 + dt - (dt*R/L);
t0 = clock ();
for j=2:divide(t_end+dt,dt)
    i(j) = srvl2 * i(j-1) + srvl1 * v(j-1) ;
end for
vr = R * i ;
vl = v - vr ;
elapsed_time = etime (clock (), t0);

```



Şekil 5.11. Octave ile modelin doğrudan kodlanmasıyla elde edilen sonuçlarının gösterimi

Yeni kodlar ile yapılan simülasyonun Şekil 5.11.'de gösterdiği gibi sonuçların aynı olduğunu ve gerektiğinde modelin kendisinin de doğrudan donanıma gerekli dönüşümler yardımıyla gömülebileceği anlaşılmaktadır.

Her iki yaklaşım için çözüm sürelerine bakıldığında ki sürelerin sayımı sadece akım ve gerilimlerin hesaplanması adımlarını kapsamaktadır, model bazlı yaklaşımın daha hızlı olduğu görülmektedir. Birinci yaklaşımla oluşturduğumuz betik kısmı 19,646 saniye sürmüştür halbuki ikinci yaklaşımda süre 6,622 saniye değerine düşmüştür. Elbette burada hesaplama yapılacak adımların ve işlem türü ve sayısının önemi büyüktür, ayrıca bilgisayar simülasyonlarında simülatör programı işletim sisteminin zaman paylaşımı çalışma mantığı ile sistem kaynaklarını sadece belli zamanlarda kullanabildiğini unutmamak gerekmektedir.

5.1.3. FPGA Simülasyon Modülü

RL devresinin donanımsal simülatörü için sistem çıkışının denkleminin çözümü ile simülatör tasarımı kullanılmıştır. Bu yaklaşım RL devresi gibi birinci dereceden sistemlerde rahatlıkla kullanılabilir.

Tasarlanan donanım modülleri klasik donanım simülatörlerinden farklı şekilde gerek giriş değişkeni işaretini gerekse de sistemin R ve L değişkenlerini de gerçek zamanlı girdi şeklinde kabul etmektedir. Ayrıca Δt değeri de sisteme girdi olarak kabul edilmekte ve sistemde sabit değer olarak gömülmemektedir.

Bu hali ile simülatör normal donanımsal simülatörler gibi çalışabilmenin yanında sistem parametrelerinin gerçek zamanlı değişimine de olanak vermektedir. Bu örnekte RL devresinde direnç ve endüktans elemanları yerine potansiyometre ve değişken endüktans kullanılmış gibi düşünülebilir.

Bu şekilde tasarım yaparak gerçek sistemlerle karşılaştırma yaparken hassas olmayan eleman değerleri ölçümünden kaynaklanabilecek farklılıkları çalışma anında düzeltebilmek amaçlanmıştır.

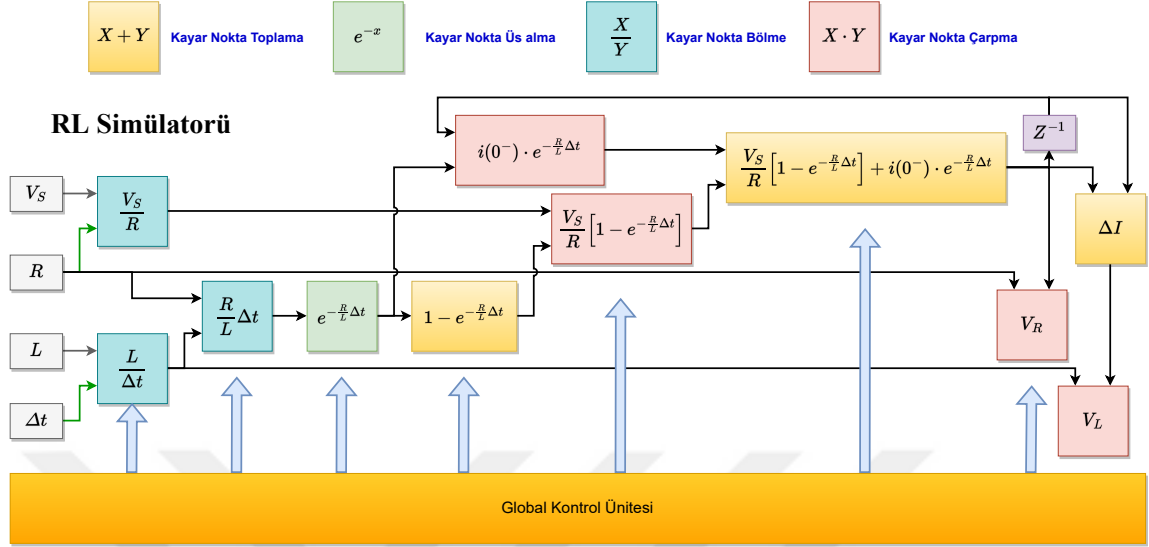
Simülatörün bu şekilde tasarlanmasının dezavantajı, bilgisayar simülasyonlarında kullanılan betiklerden görülen girdilerin sabit katsayı ile çarpımının avantajlarından vazgeçilmiştir. Dolayısıyla simülatör hem normal simülatör işlemlerini yapacak hem de simülatör için gereken katsayıların diğer girdiler ile oluşturulması işlemini de anlık olarak yapacaktır.

Bu yaklaşım simülasyon adım süresini uzatmaktadır ancak sistemin asıl yavaş kalan elemanı ADC sistemi olduğu için problem olmamaktadır. ADC çevrim hızı $2\mu s$ olduğu ve ADC modülünün öncelikli kanal sayısı bir olarak ayarlandığı için örnekleme süresi $4\mu s$ olarak kullanılmaktadır.

Sadece gerekli adımlar işletilirse 50MHz saat işaretinde işlem yaklaşık $1\mu s$ sürmektedir. Bu sebeple sistemde diğer parametrelerin de birlikte hesaplanması zaman açısından bir sorun oluşturmamaktadır. Yonga kullanımı olarak bakıldığında problem bulunmakla birlikte yaklaşımın sağladığı avantajlar tercih edilmiştir.

Şekil 5.12.'de RL simülatörü için tasarlanan FPGA modülünün blok şeması görülmektedir. Şemadan bakıldığında işlem basamakları adım adım takip edilebilmektedir. Daha önceden izah edildiği için buradaki şemada asıl modülün içinde bulunan

mandallama, senkronizasyon ve kayar nokta çevrim gibi ön ve art işlem modülleri gösterilmemiştir.



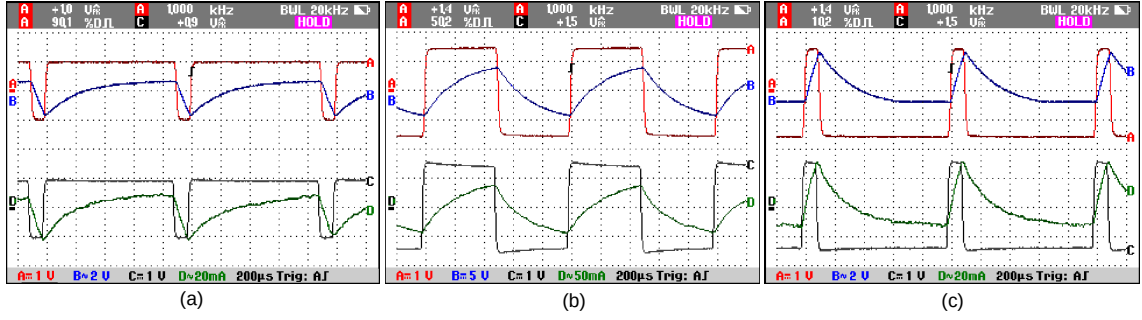
Şekil 5.12. RL devresi denklem çözücü model blok şeması

Modül 50 MHz saat işareti kullanmaktadır. Simülasyon adımını 162 saat darbesinde, yaklaşık $3,24\mu s$ süresinde bitirmektedir. ADC entegresi kapasitesi örnekleme zamanını $4\mu s$ civarına getirdiğinden kalan sürede değerler çıkışta mandallanmış şekilde sonraki simülasyon adımına kadar beklemektedirler. Elbette normal bir simülatör sistemi ile RL devresinin tek bir sinyal girişi olacağından işlemler kısıtlanarak sistem $2\mu s$ adımlar için ayarlanabilecektir.

5.1.4. Karşılaştırmalı Deneyler

Donanımsal simülatör deneylerde gerçek fiziksel devre ile birlikte çalıştırılarak tepkileri karşılaştırılmıştır. Denemelerde her iki sistem için direnç değeri $R = 22\Omega$, endüktans değeri ise $L = 4,825mH$ olarak mevcut fiziksel devre elemanlarına uygun olacak şekilde kullanılmıştır.

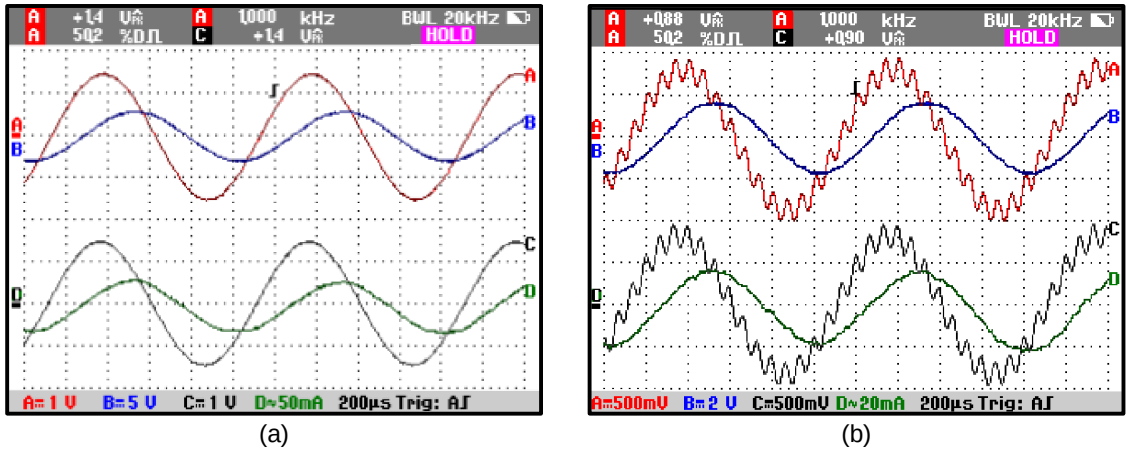
Akım değeri voltaj olarak çıkışa verildiği için ölçekleme ayarları yapılmıştır.



Şekil 5.13. RL simülâtörü ve fiziksel devre karşılaştırması, kare dalga girişi; a) %90 görev periyodu, b) %50 görev periyodu, c) %10 görev periyodu için

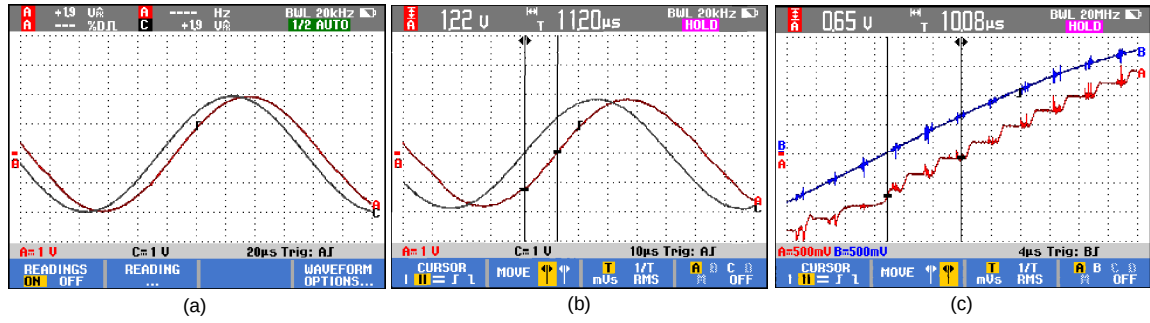
Bu denemelerden kare dalga giriş için yapılan testlerin sonuçları Şekil 5.13.'te görülebilmektedir. Sinyaller sırasıyla; A-Kanalı (Kırmızı) simülâtör giriş voltajı, B-Kanalı (Mavi) simülâtör akım çıkışı, C-Kanalı (Siyah) gerçek devre giriş voltajı, D-Kanalı (Yeşil) gerçek devre akımı olarak osiloskoptan kaydedilmiştir. Kare dalga uygulaması farklı görev süreleri için test edilerek sonuçların benzerliği görülmüştür.

Kare dalga deneyinde fiziksel devreyi besleyen sürücünün çıkış gücünün yetersiz kalması problemi dışında bir problem yaşanmamıştır.



Şekil 5.14. RL simülâtörü ve fiziksel devre karşılaştırması, sinüs sinyali ve ek harmonik; a) 1KHz sinüs işareti , b) 1KHz sinüs işareti ve 15KHz yüksek frekanslı bileşen eklenmiş hali

Şekil 5.14.'te görülebilen sinüs sinyali uygulamasında gerçek sistemle benzer şekilde akım gerilime göre gecikmiştir. Ana sinüs bileşeni üzerine eklenen yüksek frekanstaki bileşenin akım cevabında nasıl filtrelendiği görülebilmektedir.



Şekil 5.15. RL simülatörü ve fiziksel devre giriş ve çıkış sinyalleri karşılaştırması (sinyal gecikmesi); a) 5KHz sinüs karşılaştırma, b) 10KHz sinüs karşılaştırma, c) 10KHz sinüs karşılaştırma detay

Şekil 5.15.'te görülebilen sinüs sinyali uygulamasında gerçekleşen gecikme görülebilmektedir. Daha ayrıntılı olan Şekil 5.15. (b)'de gerçek sinyal ve simülatör çıkışı arasında $11,2\mu s$ gecikme görülmektedir. Sinyallere daha yakından bakıldığından simülatörün basamaklı tepki sinyali Şekil 5.15. (c)'de görülebilmektedir, ayrıca bu detay düzeyinde simülatör örnekleme zamanı olan $4\mu s$ adımlar seçilebilmektedir.

Donanımsal simülatörün bir sinyali almasının ardından bir ADC okuma periyodu geçmektedir. Sonraki okuma periyodunun başında simülatöre ulaşan sayısal değer bir okuma periyodu kadar işlem görmektedir. İkinci okuma periyodunun sonunda sinyal çıkış modülüne iletilmekte ve bundan sonrası için DAC entegresi hızına ve DAC entegresi çıkış oturma zamanına bağlı olarak bir süre daha kaybedilmektedir. Şekillerden de anlaşılabacağı gibi arada geçen bu toplam süre okuma, hesaplama ve çıkışa aktarma işlemleri boyunca harcanan süreye yaklaşık olarak uymaktadır.

5.2. PMDC Motor Simülatörü

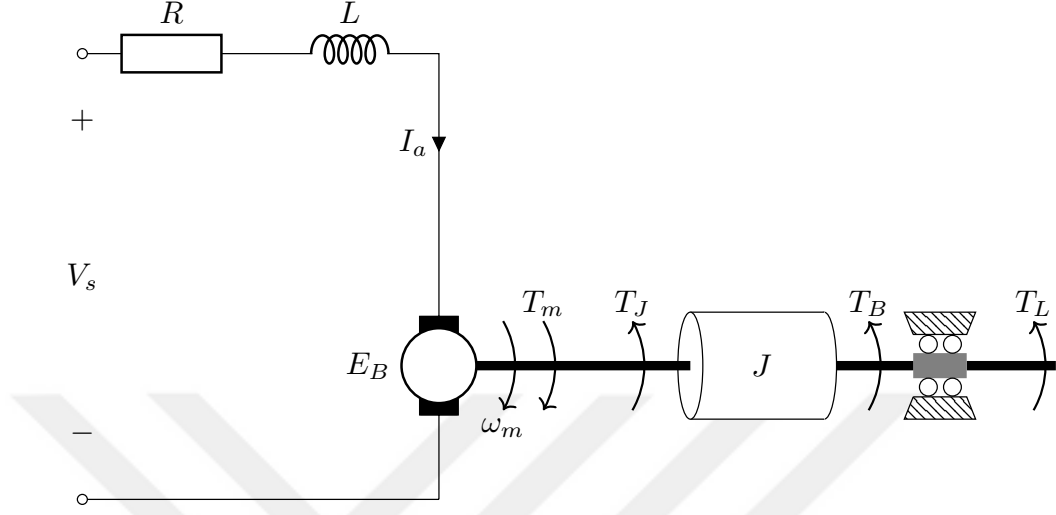
İkinci Simülatör tasarımı olarak çokça uygulaması olan PMDC motor simülatörüdür. Bu simülatör için Bölüm laboratuvarında bulunan ve teknik detayları dokümanlarda bulunabilen bir PMDC motor olan Maxon F2140-934 çekirdeksiz tipteki motor seçilmiştir.

Bir PMDC motor ile ilgili denklemlerde kullanılan bütün parametre ve giriş değerleri ADC entegresinin 8 kanalına dağıtılarak tamamı modelde aktif olarak kullanılmıştır. Giriş kanallarından uygulanan değerler donanımsal olarak simülatöre gömülü varsayılan değerlere eklenerek simülasyonda kullanılacak şekilde tasarım yapılmıştır. Seçme anahtarları yardımı ile gömülü varsayılan değerler ve ADC ile okunan değerler ayrı ayrı seçilerek toplama ünitesine aktarılıp simülatör sistemine uygulanabilmektedir.

Bu hali ile simülatör çalışma anında parametrelerdeki değer artışlarını gerçek zamanlı olarak tepkisine yansıtmaktadır. Parametre değerinin düşürülmesi gerekiyorsa ya modele girilen sabit değerler düşürülmeli ya da ek olarak giriş kanalından gelen veriyi kayar nokta sayı tipine çeviren çeviricilerde gerekli değişiklikler yapılmalıdır.

5.2.1. Sistem Modeli

Kalsik PMDC motor modeli Şekil 5.16.'da gösterilmektedir.



Şekil 5.16. Sabit mıknatıslı doğru akım motoru modeli

$$V_s(t) = L \frac{di_a(t)}{dt} + Ri_a(t) + E_b(t) \quad (5.8)$$

$$T_e(t) = J \frac{d\omega(t)}{dt} + B\omega(t) - T_i(t) \quad (5.9)$$

Denklem 5. 8 ve Denklem 5. 9 sistemin elektriksel ve mekanik eşdeğer devrelerinin diferansiyel denklemlerini ifade etmektedir.

$$E_b(t) = K_b \cdot \omega(t) \quad , \quad T_e(t) = K_m \cdot i_a(t) \quad (5.10)$$

Denklem 5. 10 eşdeğer devreler arası etkileşimleri ifade etmektedir.

$$V_s(s) = (Ls + R) \cdot I(s) - L \cdot i(0^-) + K_b \cdot W(s) \quad (5.11)$$

$$T_e(s) = (Js + B) \cdot W(s) - J \cdot \omega(0^-) + K_m \cdot I(s) \quad (5.12)$$

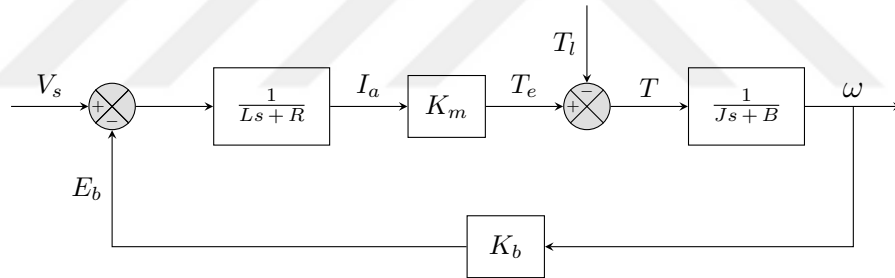
Denklem 5. 11 ve Denklem 5. 12 ise sistemin elektriksel ve mekanik eşdeğer devrelerinin denklemlerinin Laplace dönüşümlerini ifade etmektedir.

Sistem için çıkış denklemlerinin çözümü metodu kullanılacak olursa her iki çıkış değişkeni için sistemin transfer fonksiyonlarının basamak girişi ile çarpılması ve ardından çözümleri gerekmektedir. Sistem ikinci dereceden bir sistem olduğundan çözüm sırasında karşımıza ζ değerinin alabileceği değerlere göre farklı sistem tipleri için farklı çözümler çıkacaktır. Donanımsal simülatörde bu, modül seçmek veya dönüştürmeyi gerektirecektir ki seçilecek modüllerin sistemde bulunması veya dönüşecek modüllerin dönüşme işlemine uygun yapılması gerekecektir.

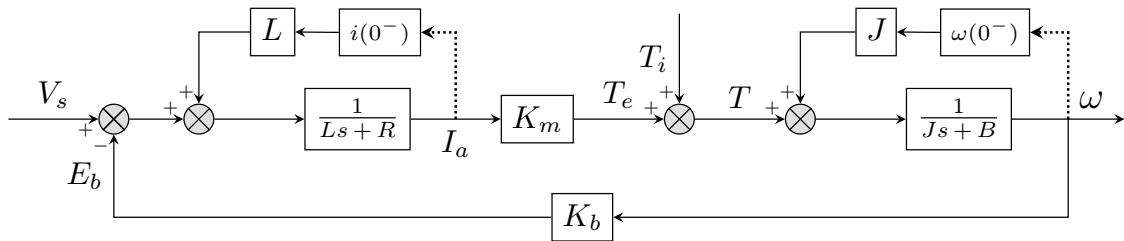
Ayrıca modellemede sistemin enerji depolayan birimleri kadar sistem denkleminde ek terim bulunacaktır. Motor örneğimizde enerji depolayan iki eleman L_a ve J_m sayesinde $i(0)$ ve $\omega(0)$ değerleri ikinci dereceden terimleri ile birlikte hem akım hem de açısal hız denklemlerinde belireceklerdir.

Bu sebeplerden ikinci veya daha yüksek mertebeden sistemlerin donanımsal simülatörlerini tasarlarken modeli doğrudan donanıma gömmek en kolay ve uygulanabilir yol olarak karşımıza çıkmaktadır.

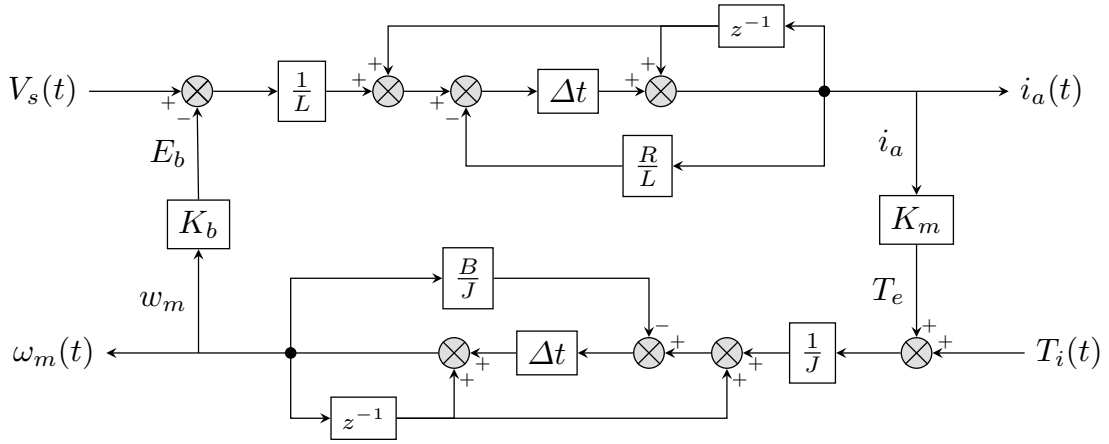
Denklem 5. 11 ve Denklem 5. 12 kullanılarak Şekil 5.17.'de görülen PMDC motor modeline ulaşılabilmektedir.



Şekil 5.17. PMDC motor sistem blok diyagramı



Şekil 5.18. PMDC motor sistem blok diyagramı eklenen ilk durum blokları



Şekil 5.19. FPGA sistemine aktarılabilecek şekli ile PMDC motor sistem blok diyagramı

Şekil 5.18.'de görülen diyagram, Şekil 5.17.'de görülen şemaya sistemdeki başlangıç değerlerinin eklenmesinden ibarettir. Şekil 5.1.'de tasvir edilen dönüştürme işlemleri kullanılarak, Şekil 5.19.'da görülen ve simülasyon tasarımlarında kullanılabilecek blok gösterimine erişilmektedir.

Şekil 5.19.'da görülen blok şemasından yararlanarak armatür akımı için Denklem 5. 13 ve açısal hız için Denklem 5. 14 elde edilmektedir.

$$i(t) = i(t - \Delta t) + \Delta t \left\{ \left[i(t - \Delta t) + \frac{1}{L} (V_s(t) - K_b \omega(t)) \right] - \frac{R}{L} i(t) \right\} \quad (5. 13)$$

$$\omega(t) = \omega(t - \Delta t) + \Delta t \left\{ \left[\omega(t - \Delta t) + \frac{1}{J} (T_i(t) + K_m i(t)) \right] - \frac{B}{J} \omega(t) \right\} \quad (5. 14)$$

Denklem 5. 13 ve Denklem 5. 14 birlikte kullanılarak akım ve açısal hız değerleri için sırası ile Denklem 5. 15 ve Denklem 5. 16 elde edilmektedir.

$$i(t) = \frac{(1 + \frac{B\Delta t}{J}) \frac{\Delta t}{L} V(t) - \frac{\Delta t^2 K_b}{JL} T_i(t)}{(1 + \Delta t \frac{R}{L})(1 + \Delta t \frac{B}{J}) + \Delta t^2 \frac{K_m K_b}{JL}} + \frac{(1 + \Delta t) [(1 + \Delta t \frac{B}{J}) i(t - \Delta t) - \frac{\Delta t K_b}{L} \omega(t - \Delta t)]}{(1 + \Delta t \frac{R}{L})(1 + \Delta t \frac{B}{J}) + \Delta t^2 \frac{K_m K_b}{JL}} \quad (5. 15)$$

$$\begin{aligned}\omega(t) = & \frac{(1 + \frac{R\Delta t}{L})\frac{\Delta t}{J}T_i(t) + \frac{\Delta t^2 K_m}{JL}V(t)}{(1 + \Delta t\frac{R}{L})(1 + \Delta t\frac{B}{J}) + \Delta t^2\frac{K_m K_b}{JL}} \\ & + \frac{(1 + \Delta t)[(1 + \Delta t\frac{R}{L})\omega(t - \Delta t) + \frac{\Delta t K_m}{J}i(t - \Delta t)]}{(1 + \Delta t\frac{R}{L})(1 + \Delta t\frac{B}{J}) + \Delta t^2\frac{K_m K_b}{JL}}\end{aligned}\quad (5.16)$$

Bu denklemlerde ortak olan payda kısmının gösterim kolaylığı bakımından Denklem 5. 17’de olduğu gibi “G” olarak yazılması denklemlerin matris formunda yazımını kolaylaştırmaktadır.

$$G = (1 + \Delta t\frac{R}{L})(1 + \Delta t\frac{B}{J}) + \Delta t^2\frac{K_m K_b}{JL} \quad (5.17)$$

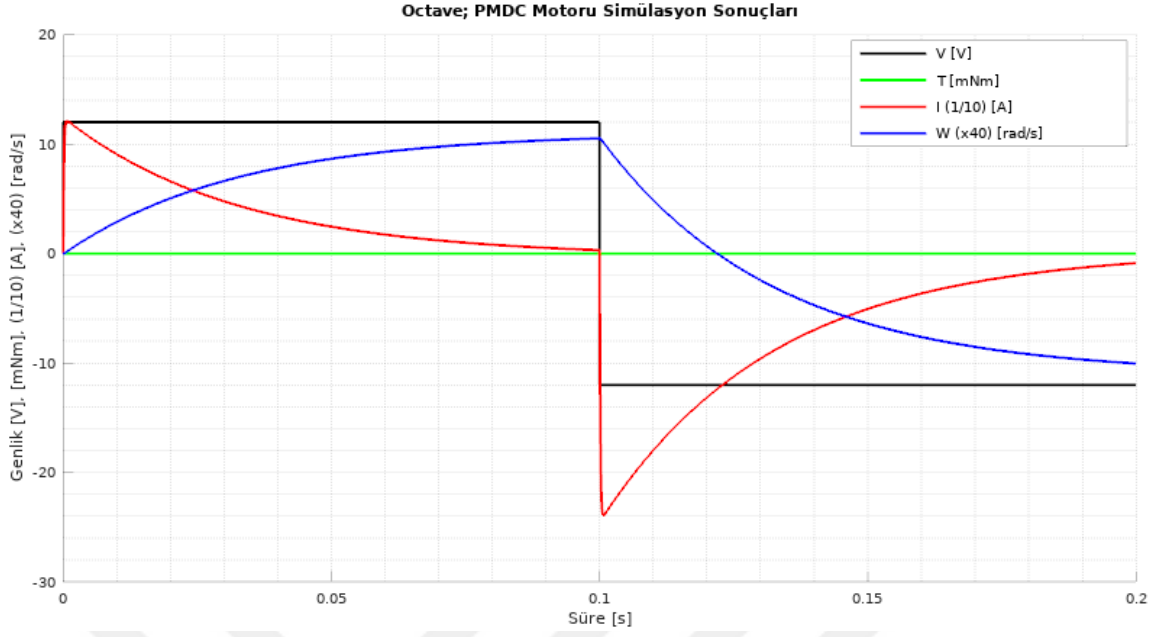
Denklem 5. 18 kullanılarak donanımsal simülatör için işlemler ve işlem sıraları belirlenir ve tasarım gerçekleştirilir.

$$\begin{aligned}\begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix} = & \frac{1}{G} \begin{bmatrix} (1 + B\frac{\Delta t}{J})\frac{\Delta t}{L} & K_b\frac{\Delta t^2}{JL} \\ K_m\frac{\Delta t^2}{JL} & (1 + R\frac{\Delta t}{L})\frac{\Delta t}{J} \end{bmatrix} \begin{bmatrix} V_s(t) \\ T_i(t) \end{bmatrix} \\ & + \frac{1 + \Delta t}{G} \begin{bmatrix} 1 + B\frac{\Delta t}{J} & -K_b\frac{\Delta t}{L} \\ K_m\frac{\Delta t}{J} & 1 + R\frac{\Delta t}{L} \end{bmatrix} \begin{bmatrix} i(t - \Delta t) \\ \omega(t - \Delta t) \end{bmatrix}\end{aligned}\quad (5.18)$$

5.2.2. Bilgisayar Simülasyonu

Denklem 5. 18 kullanılarak oluşturulan betik dosyası Octave ile çalıştırılmıştır. PMDC motor için 0,2 s süreli modelleme 1,9 s süre boyunca sürmüştür.

Şekil 5.20.’de görülen sonuçlarda eksenler tek olduğundan değerler belli katsayılarla çarpılarak çizilmiştir. Sinyallerin daha okunaklı olması için akım 10 ile çarpılmış, açısal hızın ise 40’a bölümü çizdirilmiştir. Motorun çektiği demeraj akımı ve açısal hızın yükselişleri görülebilmektedir.



Şekil 5.20. Octave ile elde edilen PMDC motor simülasyon sonuçları

5.2.3. FPGA Simülasyon Modülleri

PMDC motor simülatör tasarımında RL devre modelinden farklı olarak simülatör iki farklı modüle ayrılmıştır. Motorun simülasyonunu yapan modül giriş değişkenleri olarak sadece giriş voltajı V_S , ve yük torku T_L değerlerini almaktadır. Hesaplama için gerekli katsayılar ayrı bir parametre modülünde üretilerek her simülasyon adımı başında motor simülasyon modülüne aktarılmaktadır.

Şekil 5.21.'de görülen blok diyagramında, parametre hesaplama modülü her örnekleme periyodu sonunda kullanılacak çarpım parametrelerini ADC modülünden okunan armatür direnci R_a , armatür endüktansı L_a , viskoz sürtünme katsayısı B_m , atalet momenti J_m , ve motor katsayıları olan K_m ve K_b değerlerini alarak bu değerlerden gerekli parametreleri üreterek motor simülasyon modülüne aktarmaktadır.

Bu şekilde simülatör giriş voltajı V_S , ve yük torku T_L değişkenlerini çıkışa bir simülasyon adımı sonrasında, R_a , L_a , B_m , J_m , K_m ve K_b değerlerindeki değişimi ise iki adım sonra çıkışa yansıtabilmektedir.

Motor simülasyonunda ADC sistemi girişlerden ikisi; V_S ve T_L değerlerini öncelikli kanal olarak okumakta, diğerlerini ise normal okumaktadır. İki öncelikli kanal ve normal kanallarla beraber etkin örnekleme zamanı $6\mu s$ olarak sağlanmaktadır.

Simülatör ayrıca karşılaştırma için bağlanacak harici fiziksel motora monte edilmiş arttırımlı enkoder sinyallerini okuyarak hızı hesaplayacak modülü de içermektedir.

Simülatör çıkış değerleri olarak motor akımı i_m ve simülasyon motorunun şaft hızını ω_s vermekte ayrıca uygulanan gerilimin kontrol için bir örneğini de çıkış kanallarından birine yansıtmaktadır. Bu sinyallere ek olarak, bağlanmış harici motorun enkoder bilgisinden

The diagram illustrates the hardware and software components of a PMDC motor control system. It is organized into three main functional areas:

- PC (Personal Computer):** Acts as the central control unit, connected to the USB Blaster II and the Analog Discovery 2.
- Analog Discovery 2:** A multi-channel oscilloscope and signal generator used for monitoring and controlling the motor's electrical signals.
- Cyclone V FPGA (Field-Programmable Gate Array):** The core processing unit, containing several key blocks:
 - Signal Tap Logic Analizör:** For real-time signal monitoring.
 - Reset Sys.:** For system reset functionality.
 - PLL (Phase-Locked Loop):** For clock signal generation.
 - Parametre Hesaplama:** For parameter calculation.
 - PMDc Motor Simülâtörü:** A software model of the motor for simulation.
 - ADC (Analog-to-Digital Converter):** For converting analog signals to digital.
 - ADC SPI:** Interface for the ADC.
 - DAC PI (Digital-to-Analog Converter with Proportional-Integral control):** For generating control signals.
 - Enkoder Ara yüzü:** Interface for the motor's position feedback.

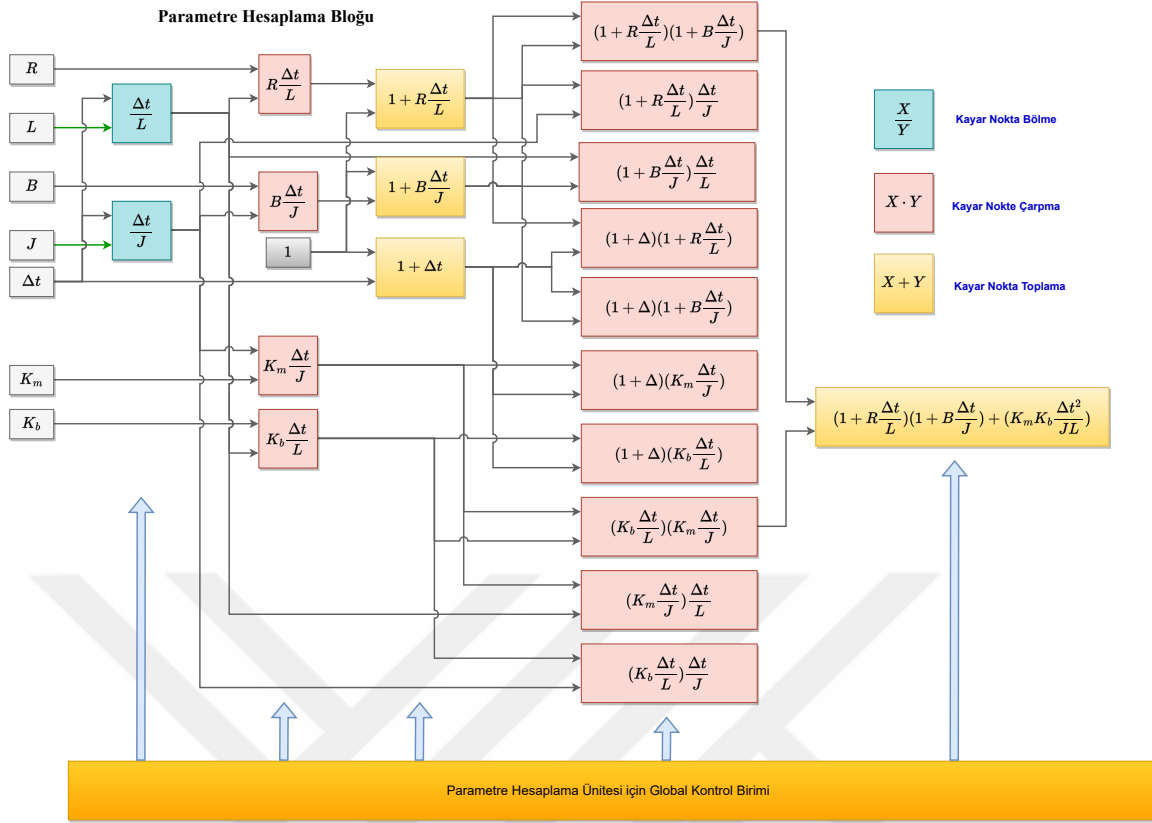
The system's operation involves the following signal flow:

- The **PC** sends commands to the **USB Blaster II**, which interfaces with the **Cyclone V FPGA**.
- The **Analog Discovery 2** measures the motor's current (I_a) and provides feedback to the **Cyclone V FPGA**.
- The **Cyclone V FPGA** processes the feedback and generates control signals for the **PMDc Motor** and the **Analog Discovery 2**.
- The **PMDc Motor** is connected to a power source and a feedback encoder.

The diagram also shows the motor's electrical characteristics, including the current I_a , the supply voltage V_s , and the motor's speed $\omega_m(t)$ in rpm.

Simülasyon Parametre Belirleme Modülü

Motor modülü ile birlikte aynı saat işaretini ve aynı örnekleme işaretini kullanmakta ve motor modülüne paralel olarak çalışarak parametre değerlerini bir sonraki simülasyon adımı için motor modülü parametre girişlerinde hazır bulundurmaktadır. Her simülasyon çevrimi boyunca parametre değerlerinin mandallanması işlemini de kendisi yapmaktadır.



Şekil 5.22. PMDC motor simülörü parametre belirleme kısmı blok diyagramı

Motor Simülasyon Modülü

PMDC motor simülasyon modülü Şekil 5.23.'te takip edilebileceği üzere, öncelikli ADC kanallarının bilgilerini ve kullanılacak parametre değerlerini dışarıdan almakta, kendi içerisinde kaydettiği sistem ilk değerleri de kullanarak bir adım sonrasının çıkış değerlerini üreterek hem DAC modülüne aktarmakta hem de bir sonraki simülasyon adımının ilk değerleri için kendisi de kaydetmektedir.

Simülasyon modülü toplama ve çıkarma işlemlerini yaparken hassasiyetin mümkün olduğunca kaybolmayacağı bir işlem sırası takip etmektedir. FPGA simülasyon modülleri bilgisayar tabanlı simülörlerin kullandığı 64 bit genişliğindeki çift duyarlılıktaki değerler yerine 32 genişliğindeki tek duyarlılık sayıları kullanarak hesaplamalar yapmaktadır. Bu sebeple işlem önceliği ve farklı büyüklüklerdeki sayıların beraber işlenmesi konularında daha dikkatli olunması gerekmektedir.

Simülörün kaynak kullanım oranlarına bakıldığında parametre hazırlama modülünün motor simülasyon modülünden daha fazla kaynak kullandığı Tablo 5.2.'de görülebilmektedir. Ancak yapılan deneylerde motorun yüklenmesi ile ilgili öngörülmesi zor değerlerin hesaplanarak simülöre gömülmesi işleminden çok daha kolay bir şekilde motor ve simülör tepkilerinin eşitlenmesinde parametre hesaplama modülünün büyük faydaları görülmüştür. Her seferinde değerlerin hesaplanarak yeniden sentezleme yapmak yerine

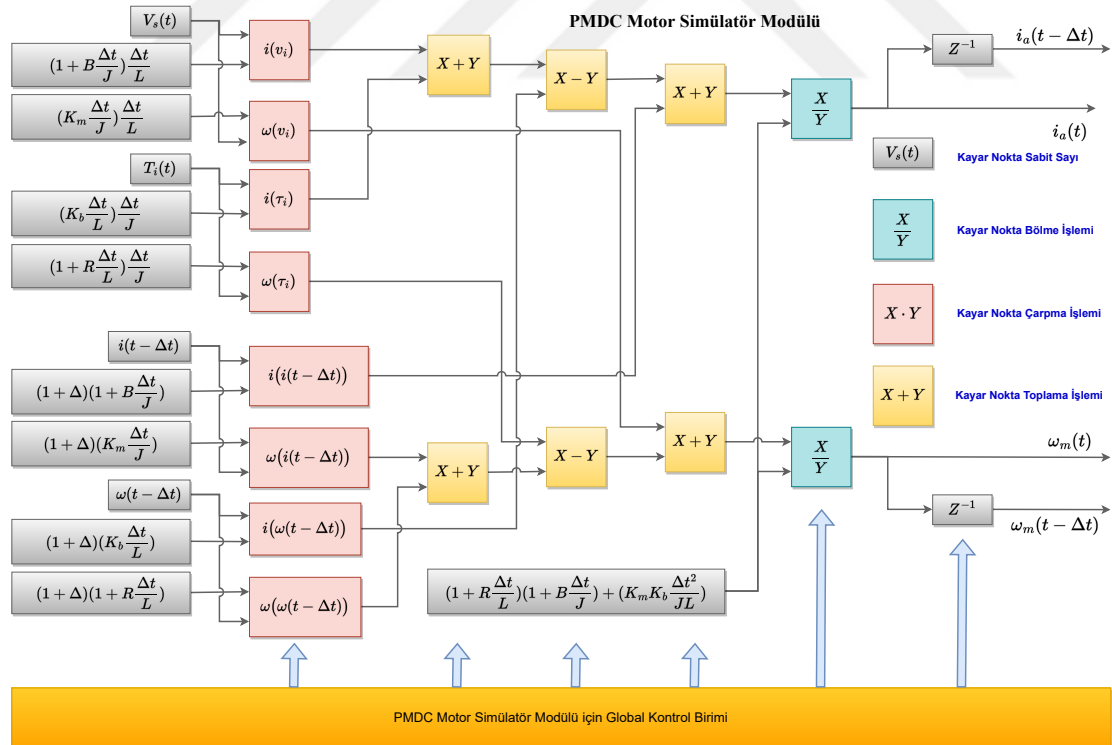
Tablo 5.2. PMDC motor simülatör modülü ve parametre hazırlama modülü kaynak kullanımı kıyaslaması.

Modül	Blok Hafıza	M10Ks	DSPs
Parametre Modülü	18896	83	32
Simülatör Modülü	10568	55	21

Tablo 5.3. PMDC motor simülatörü FPGA kaynak kullanımı bilgileri

Kaynak İsmi	Kaynak Kullanımı	FPGA Mevcut Miktar	Kullanım Oranı
ALMs	10496	41910	%25
Blok Hafıza	29464	5662720	%0.52
M10Ks	138	553	%25
DSP Blokları	53	112	%47
PLL	1	6	%25

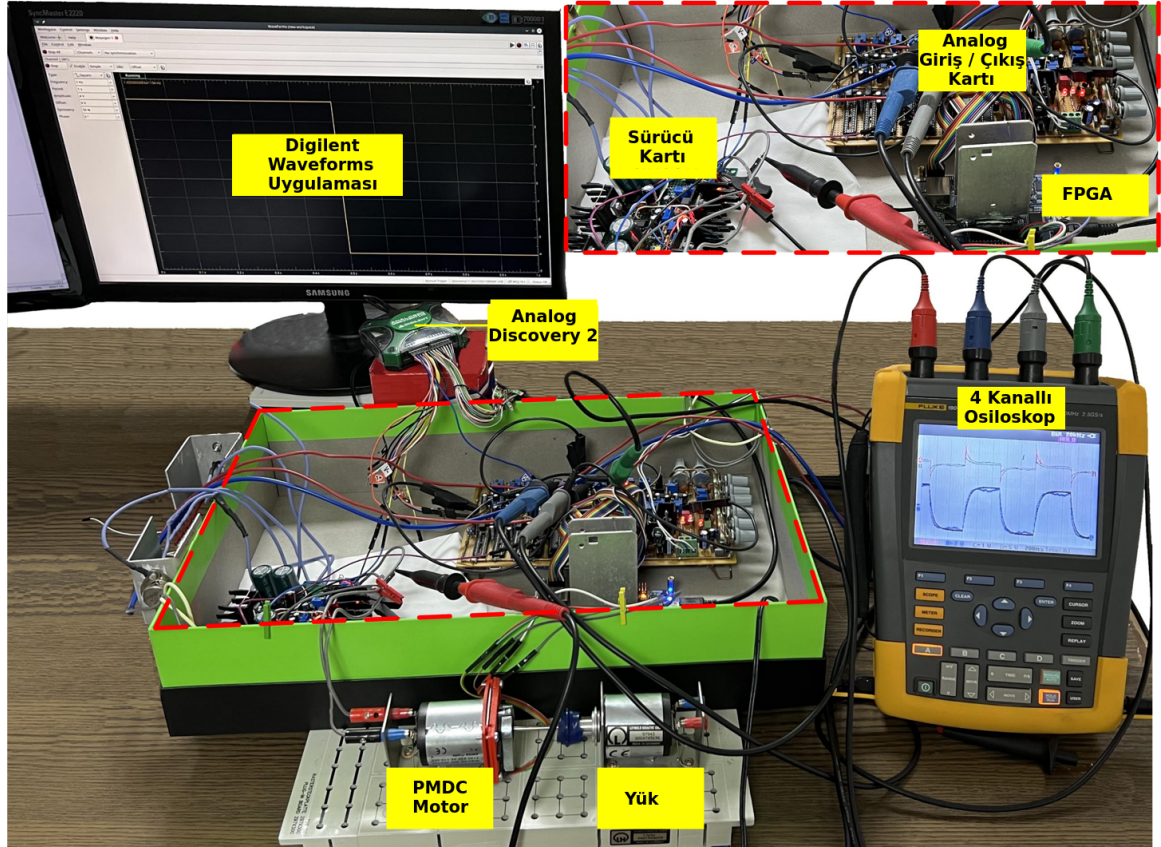
osiloskop ekranından bakarken canlı şekilde motor ve yüklenme değerlerinin ayarlanması sayesinde donanımsal model gerçek sisteme çok daha hızlı şekilde eşitlenebilmiştir. Simülatör tasarımının toplam kaynak kullanımı Tablo 5.3.'te görülmektedir. Simülatör devresinin sentezleme sonrası RTL görünümü EK-11'de incelenebilir.



Şekil 5.23. PMDC motor simülatörü blok diyagramı

5.2.4. Karşılaştırmalı Deneyler

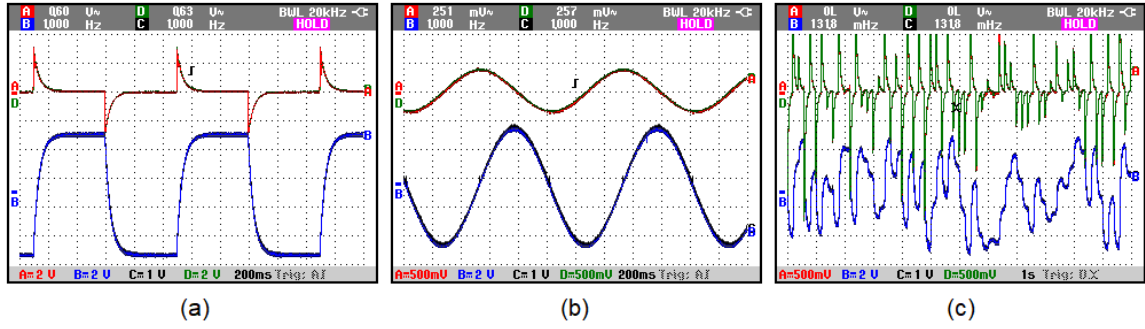
Şekil 5.24.'te görülen deney düzeneği kullanılarak, tasarlanan gerçek zamanlı dijital PMDC motor simülatörü ile gerçek fiziksel motor birlikte çalıştırılarak karşılaştırmalı deneyler yapılmıştır. Gerekğinde fiziksel motoru yüklemek için aynı tip motordan bir yük sistemi oluşturularak birbirine akuple edilmiştir. Deney düzeneğinde çeşitli gerilim dalga şekilleri Digilent Waveforms uygulaması kullanılarak Analog Discovery 2 cihazı üzerinden oluşturularak sistemlere uygulanmıştır.



Şekil 5.24. PMDC motor simülatörü deney düzeneği

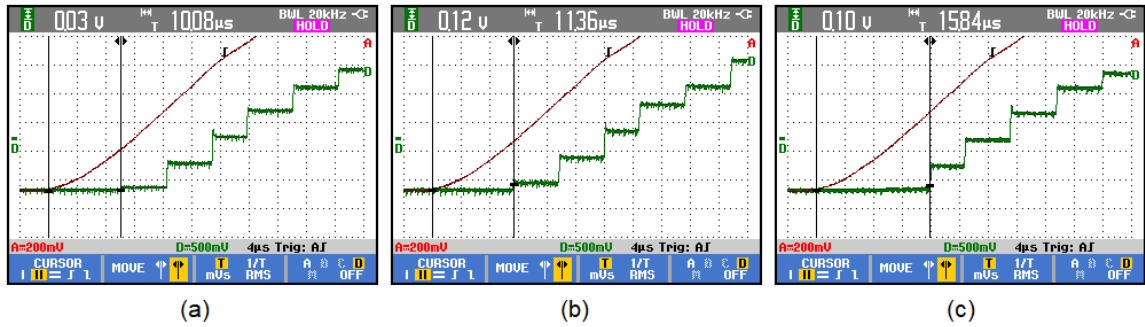
Kullanılan osiloskop 4 kanallı olduğundan sistemlerin giriş sinyali hariç, sadece çıkış sinyalleri olan açısal hız (ω) ve armatür akımı (I_a) olmak üzere her sistem için ikişer parametreden toplam 4 kanal izlenmiştir.

Maxon F2140-934 motorunun parametreleri dokümanında bulunmakla birlikte, şafta takılan enkoder için yüklenme verileri bulunmadığı için simülatör ve motor benzer tepkiler verinceye kadar motor parametre girişi potansiyometreler ile kalibre edilmiştir. Ayrıca analog çıkış devrelerinde gerekli kazanç ve offset ayarları yapıldıktan sonra aşağıda görülen şekillerde çıkışlar karşılaştırmalı olarak incelenmiştir.



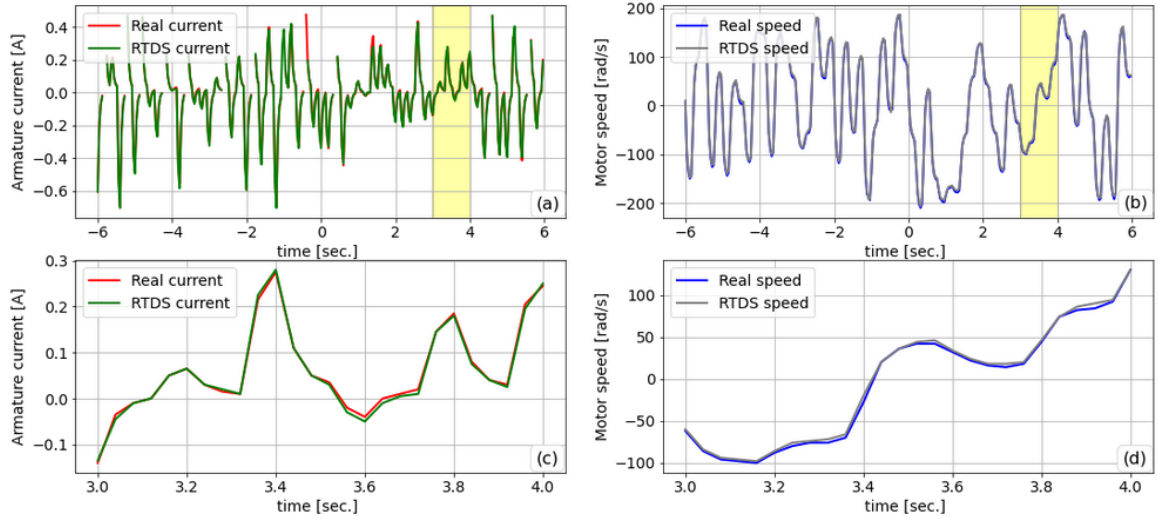
Şekil 5.25. Fiziksel PMDC motor ve tasarlanmış gerçek zamanlı simülâtörün karşılaştırılması ; a) 1 Hz kare dalga giriş gerilimi uygulanması, b) 1 Hz sinüs dalga giriş gerilimi uygulanması, c) Kare dalga formunda gürültü işaretinin giriş gerilimi olarak kullanılması

Şekil 5.25. (a) ekranında gerçek ve dijital ikiz motorlara 1 Hz V_{p-p} 8 volt kare dalga gerilim uygulanmış ve sistemlerin çektikleri akım ile açısal hız değerleri görülmüştür. Şekillerde A kanalı gerçek motor akımını, D kanalı ise simülâtör akımını göstermekte, B kanalı gerçek motorun açısal hızını, C kanalı ise dijital ikiz motorun açısal hızını göstermektedir. A ve D kanallarında her kare 2 volt olup voltaj başına 0,5 A ölçülmektedir, B kanalında 100 rad/s için 2 V/kare, C kanalında ise 100 rad/s için 1 V/kare ölçülmektedir. Şekil 5.25. (b)'de iki motora da sinüs sinyali uygulanmış olup sistemler arası farkların neredeyse fark edilemez olduğu görülmüştür. Şekil 5.25. (c)'de ise sistemlere kare dalga gürültü uygulanmış ve yine sistemlerde benzer tepkiler gözlemlenmiştir.



Şekil 5.26. PMDC motor simülâtörü ve fiziksel motor sistemi arasındaki farklı anlık gecikme durumları; a) 10.08 μ s gecikme, b) 11.36 μ s gecikme, c) 15.84 μ s gecikme

Şekil 5.26.'da görülen osiloskop görüntülerinde sistemlerin cevapları arasındaki farkı incelemek için zaman ekseninde yeterince büyütme yapılmıştır. Şekil 5.26. a, b ve c anlık durumları için simülâtörün gecikme değerinin 10 ~16 μ s arasında olduğu gözlemlenmiştir.



Şekil 5.27. PMDC motor simülatörü ve fiziksel motor sistemlerinin kare dalga gürültü gerilimi ile çalıştırılarak karşılaştırılması; a) Armatür akımlarının karşılaştırılması, b) Açısal hızların karşılaştırılması, c) Armatür akımlarının karşılaştırılması (detay), d) Açısal hızların karşılaştırılması (detay)

Şekil 5.27.'de görülen osiloskop görüntülerinde ise Analog Discovery 2 ünitesi ile oluşturulan kare dalga gürültü ile sistemler birbirleri ile mukayese edilmiş ve benzerliğin yüksek olduğu görülmüştür.

Bu deney sonuçlarının ardından donanımsal PMDC motor simülatörü tasarımıımızın gerçek sistemlerle oldukça yakın sonuçlar verdiği ve farklı motor uygulamalarında da kullanılabileceğini söylemek mümkündür.

Fiziki olarak iki aynı tip motor birlikte aynı koşullarda test edilse bile yine sonuçlardaki gibi küçük farkların oluşabileceğini söylemek mümkündür. Bu nedenle tezde edilen sonuçlar oldukça tatminkar olarak değerlendirilebilir.

5.3. Webots Robot Simülatörü

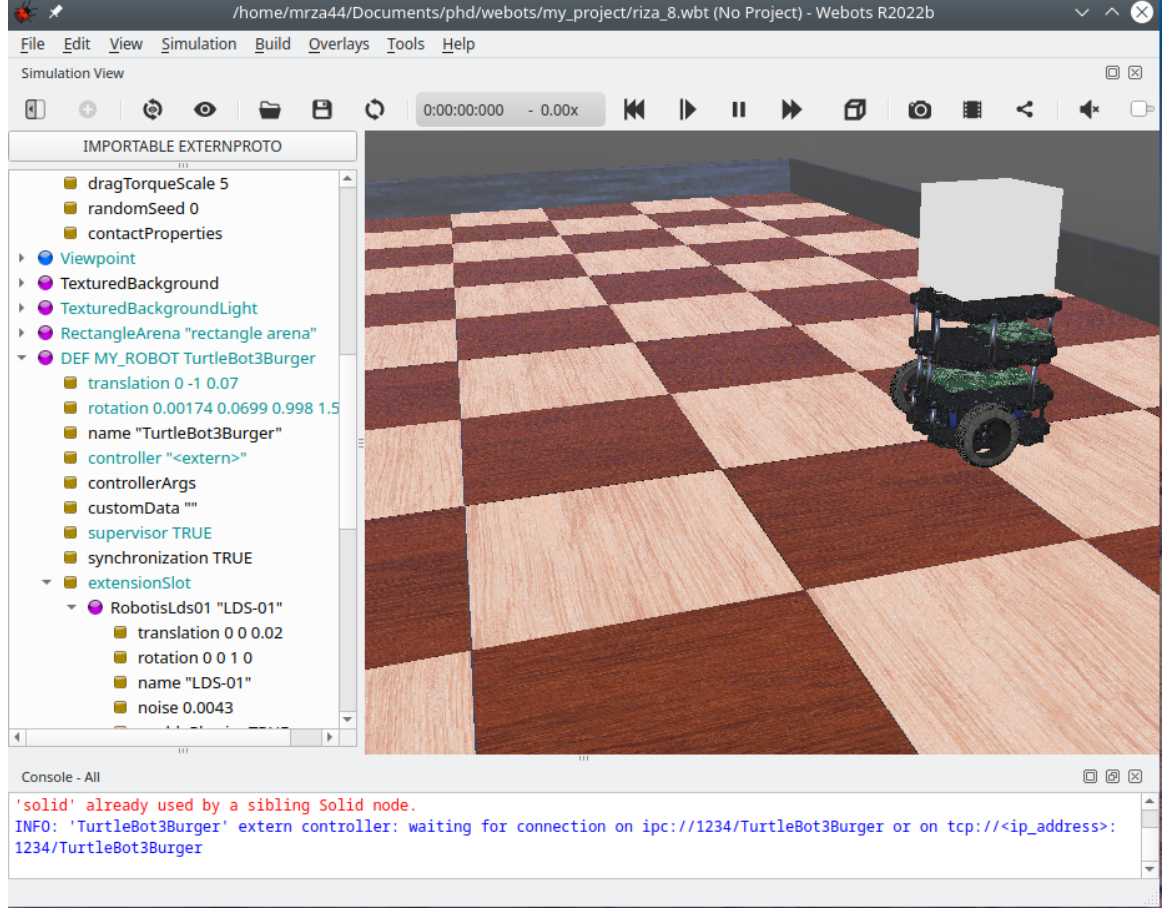
Günümüzde birçok robot simülatörü bulunmaktadır. Robot simülatörleri arasında popüler olan Webots, Gazebo ve CoppeliaSim simülatörleri incelenmiş ve Webots robot simülatörünün kaynak kullanımı açısından bu tez için daha uygun olacağına karar verilmiştir.

Webots robot simülatörü Cyberbotics firması tarafından geliştirilmiş sonradan kaynak kodları açık hale getirilmiştir. Simülasyon sırasında işlemci yükünün Gazebo gibi simülatörlerden daha düşük olması gibi bir avantajı bulunmaktadır.

Program ayrıca kullanıcıların C, C++, Python, Java dillerinde yazılmış programları tarafından kütüphaneler vasıtası ile kullanılabilmektedir. Bununla birlikte ROS ve Matlab gibi programlar ile de kullanılabilmektedir. Ancak robot simülatörleri sadece fiziksel bir sistem üzerine kurulmuş Fizik Motoru ile hesaplamalar sunmaktadır. Bu tez çalışmasının amaçlarından biri ise robot simülatörlerinin eksik olan elektriksel kısmının FPGA yardımı ile gerçek zamanlı olarak çalıştırılabileceğini göstermektir.

Program düşük yüklü modeller ve çok kısa olmayan simülasyon adım süreleri için gerçek zaman değerlerine yaklaşan değerlerde çalışabilmektedir. Kullandığımız hali ile modelimizde sadece bir robot, boş bir ortam ve robot üzerinde ağırlık bulunmaktadır.

Şekil 5.28.'de programın kullanıcı arayüzü görülmektedir.



Şekil 5.28. Webots robot simülatörü kullanıcı arayüzü

Amacımız belli şartlar için gerçek zamana yakın simülasyon değerlerine sahip Webots simülatörünü FPGA üzerinde tasarlanmış gerçek zamanlı donanımsal simülatörümüz ile birlikte kullanmaktır. Mevcut hali ile Webots simülatörü fizik modellemesini yaparken mekanik modellemeyi gerçekleştirmektedir. Programda bulunan motor fonksiyonları gerçek bir servo motorun mekanik ve elektriksel benzetiminden daha ziyade robotun hareketleri için gerekli mekanik olayları simüle etmektedir.

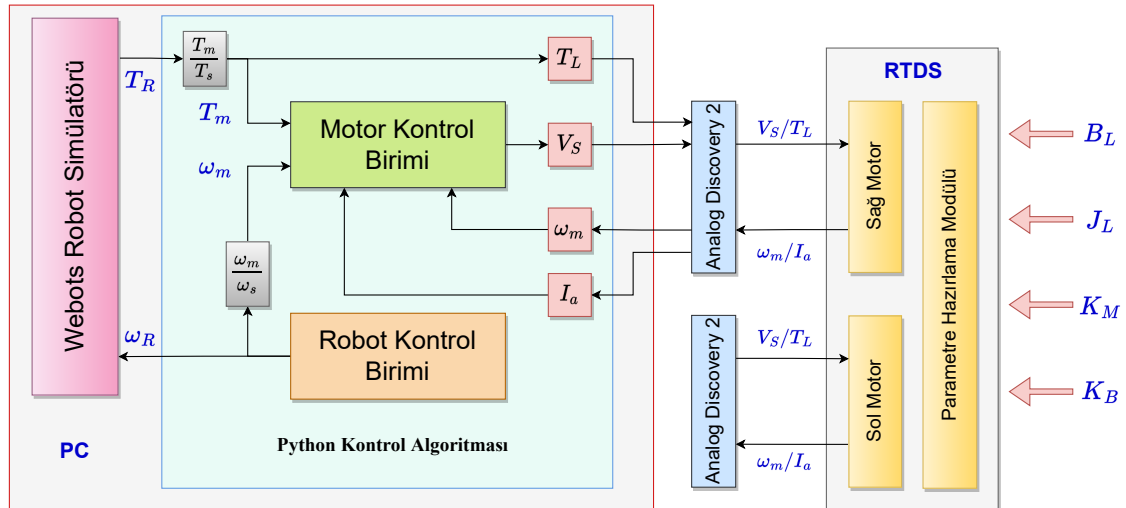
Webots simülatöründe diferansiyel tekerlekli *TurtleBot3Burger* robotu düz yüzeyli bir çalışma ortamında gerektiğinde üzerine ağırlık eklenerek simüle edilmiştir. Python arayüzü kullanılarak Webots simülatörü gerçek zamanlı simülatör ile birlikte çalıştırılmış ve Webots programında bulunmayan, motorun elektriksel davranışları gerçek zamanlı simülatörü üzerinden analiz edilmiştir.

5.3.1. Simülör Konfigürasyonu

Simülörün gerçek zamana yaklaşabilmesi için simülasyonun görsel kısmı kapatılmıştır. Sadece tek bir robot ve düz bir yüzey kullanılmıştır. Simülasyon adım süresi 32 ms olarak ayarlanmıştır. Simülasyon sırasında bilgisayarın yüksüz olması sağlanmıştır. Simülör programının işletim sisteminde önceliği artırılmışsa bile (niceness değeri düşürülmüş) bu pek bir avantaj sağlamamıştır. Webots simülörü çalışma sırasında her 4 simülasyon adımı için bir kontrol adımı gerçekleştirmektedir. Bu sebeple 32 ms olarak ayarlanan simülasyon adımı süresi ile 128 ms değerinde bir kontrol adım süresi oluşmaktadır. Robotun kontrolünde kullanılan programın, kontrol adım süresini baz alarak çalışması gerekmektedir. Tasarlanan donanımsal simülörün sentezleme sonrası RTL görünümü EK-12’de görülmektedir.

5.3.2. Webots ve FPGA Simülörlerinin Birlikte Çalıştırmaları

Bu çalışmada robotun fiziksel hesaplamalarını gerçekleştirmesi için Webots simülörü ve motorların elektriksel simülasyonlarını gerçekleştirmesi için FPGA tabanlı RTDS birlikte kullanılmıştır. Simülörlerin birlikte çalıştırılabilmesi için merkezde Python kodlamasının olduğu bir program tarafından her iki simülöre de komut gönderilmiştir. Python programında bir alt program robotun yapacağı hareket için motorlara uygulanacak açısal hız bilgisini oluşturmaktadır. Kontrol programı robotun sağ ve sol motorları için donanımsal simülörde oluşturulmuş iki adet motor modülüne, pydwf kütüphanesi vasıtası ile kullandığı Analog Discovery 2 cihazları üzerinden sinyal gönderip almaktadır.



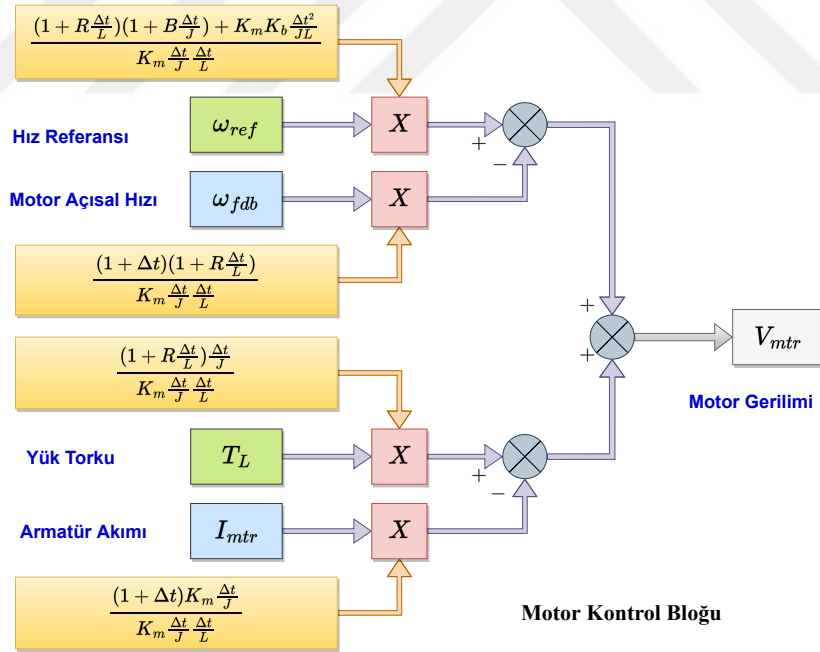
Şekil 5.29. RTDS sistemin Webots simülörü ile birlikte çalıştırılması

Şekil 5.29.’da görüldüğü gibi robot kontrol algoritmasının ürettiği açısal hız değerleri Webots programına gönderilmektedir. Webots simülöründen gelen tork bilgisi ve açısal

hız referansı, gerçek zamanlı simülatörden alınan armatür akımı ve açısıl hız geri dönüş bilgileri ile birlikte kullanılarak dijital motorlar için besleme gerilimi değeri oluşturulmaktadır. Bu gerilim değerinin hesaplanması için Denklem 5. 16'da $T_i(t)$ giriş torku yerine $T_L(t)$ yük torku ifadesi kullanılarak Denklem 5. 19'da gösterilen eşitlik üretilmiştir. Bu eşitlikten faydalanılarak motorlara uygulanacak besleme gerilim değeri üretilirken Δt için 128 ms değeri kullanılmalıdır. Ancak Δt değeri simülasyon süresince sürekli değişik değerler alabildiği için bir değişken gibi ölçülerek hesaplamalarda kullanılmıştır.

$$V(t) = \frac{(1 + R\frac{\Delta t}{L})(1 + B\frac{\Delta t}{J}) + K_m K_b \frac{\Delta t^2}{JL}}{K_m \frac{\Delta t^2}{JL}} \omega(t) + \frac{(1 + R\frac{\Delta t}{L})\frac{\Delta t}{J}}{K_m \frac{\Delta t^2}{JL}} T_i(t) - \frac{(1 + \Delta t)}{K_m \frac{\Delta t^2}{JL}} \left[(1 + R\frac{\Delta t}{L})\omega(t - \Delta t) - K_m \frac{\Delta t}{J} i(t - \Delta t) \right] \quad (5. 19)$$

Denklem 5. 19'da gösterilen hesaplamalar Şekil 5.30.'da gösterilen motor gerilimi kontrol sisteminde gerçekleştirilerek motorlara uygulanacak besleme gerilim değerleri oluşturulmaktadır.



Şekil 5.30. Motor gerilimi kontrol sistemi

Oluşan gerilim değeri yük torku değeri ile birlikte Analog Discovery 2 cihazları üzerinden gerçek zamanlı simülatöre uygulanmaktadır. Gerçek zamanlı simülatör aldığı bilgilere göre motorları çalıştırmakta, oluşan açısıl hız ve armatür akımı sinyallerini

Python ile yazılmış kontrol programına geri göndermektedir. Kontrol programı ayrıca oluşan değerleri deney sonuçları olarak kaydetmektedir.

5.3.3. Deneyler

Kurulan düzenele temelde 3 farklı hareket ile düzlemsel yüzeyin farklı eğimleri için robot üzerindeki ağırlığın farklı değerleri kullanılarak deneyler yapılmıştır.

Robot için kullanılan hareket yörüngeleri;

- Doğrusal hareket
- Dairesel hareket
- Sonsuzluk sembolünü çizerek yol alma

olacak şekilde ve motorlara farklı yüklenilmeye sebep olacak şekilde eğim ve ek ağırlık değerleri ile çalışılmıştır.

Robotun üzerinde Hareket ettiği düzlem yüzeyin belirlenen dört doğrultusu yönünde 5 farklı eğim uygulanmıştır;

- $y = 0$ doğrusu etrafında $-0.1, -0.05, 0, 0.05, 0.1$ [rad] açı değerlerinde
- $x = 0$ doğrusu etrafında $-0.1, -0.05, 0, 0.05, 0.1$ [rad] açı değerlerinde
- $y = x$ doğrusu etrafında $-0.1, -0.05, 0, 0.05, 0.1$ [rad] açı değerlerinde
- $y = -x$ doğrusu etrafında $-0.1, -0.05, 0, 0.05, 0.1$ [rad] açı değerlerinde

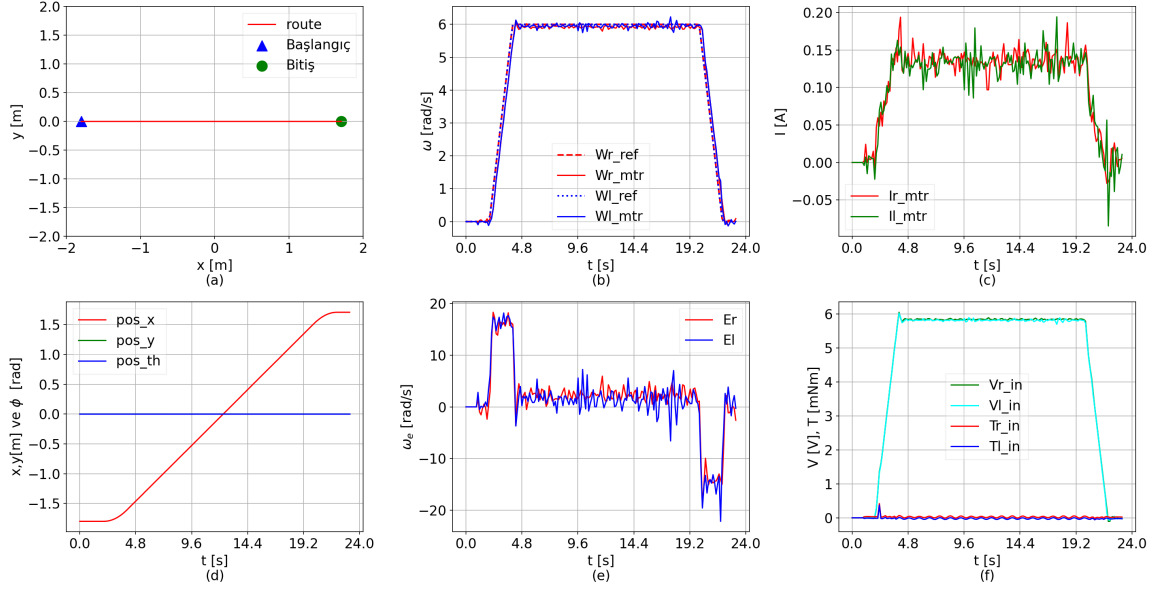
yüzeye eğim uygulanarak yokuş iniş ve çıkışları ile yanlamasına eğim gibi durumlar oluşturulmuştur.

Robot üzerine bir ağırlık konarak ağırlığın değişimi ile motorların tepkileri incelenmiştir. Belli değerlerde motorların açısal hız ve uyguladıkları torklar sınır değerlerine yaklaşınca Webots programındaki kontrolör sinyalinde bozulmalar gözlenmiştir. Eğim belirtilen doğrultudaki vektöre göre saatin tersi yönü pozitif kabul edilerek işaretlendirilmiştir. Çalışmalarda araç x ekseninde ileri doğru harekete başlamaktadır.

Doğrusal Hareket Sonuçları

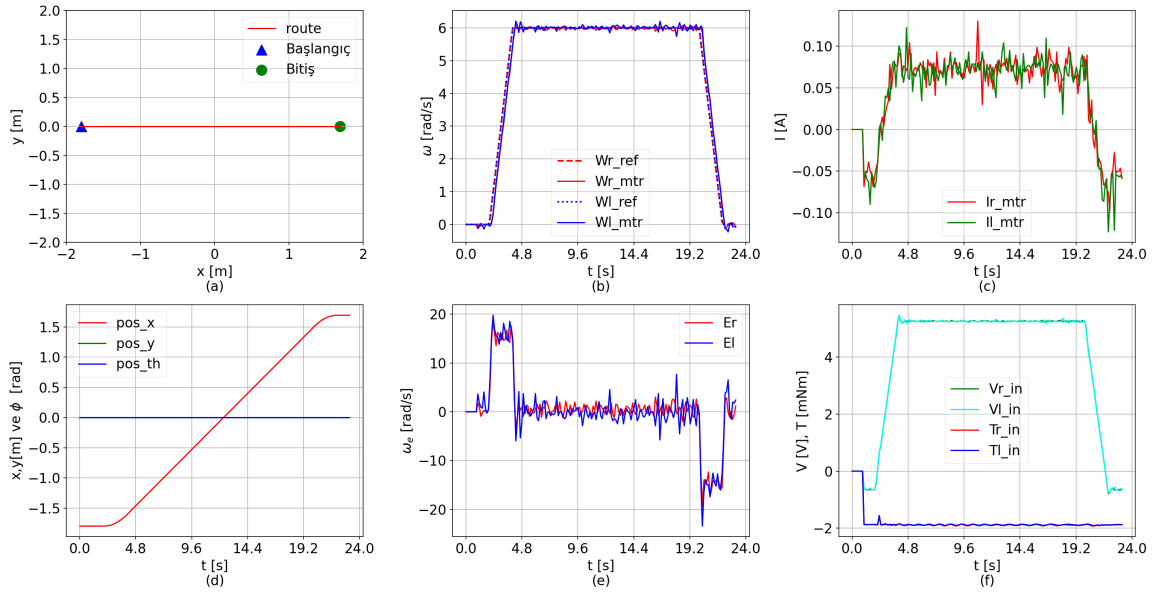
Bu çalışmada robotun hareketi ivmelenme, sabit hızda ilerleme ve frenleme olarak 3 farklı bölgede incelenmiştir. Karşılaştırmalarda kolaylık sağlaması için tüm sonuç grafiklerinin (a) kısımları robotun xy eksenlerindeki hareket yörüngelerini gösterecek şekilde düzenlenmiştir. XY ekseninde gözlemlenen hareketin yanısıra robotun x ve y eksenlerindeki pozisyon değerleri ile yönelim açısı ϕ değerinin zamana göre değişimi (d) grafiğinde gösterilmiştir. Benzer şekilde (b) grafikleri hem sağ hemde sol tekerlekler için açısal hız referansları ve RTDS tarafından oluşan açısal hız değerleri gösterilmiştir. Açısal

hız referans değerleri ve RTDS ile oluşan değerler arasındaki farklılıklar şekillerde (e) grafiklerinde gösterilmiştir. RTDS simülatörlerinin armatür akımları şekillerde (c) grafiklerinde gösterilmiştir. Şekillerde (f) grafiklerinde webots simülatörünün oluşturduğu tork değerleri ve motor kontrol birimlerinde hesaplanan motor besleme gerilimi değerleri görülmektedir.



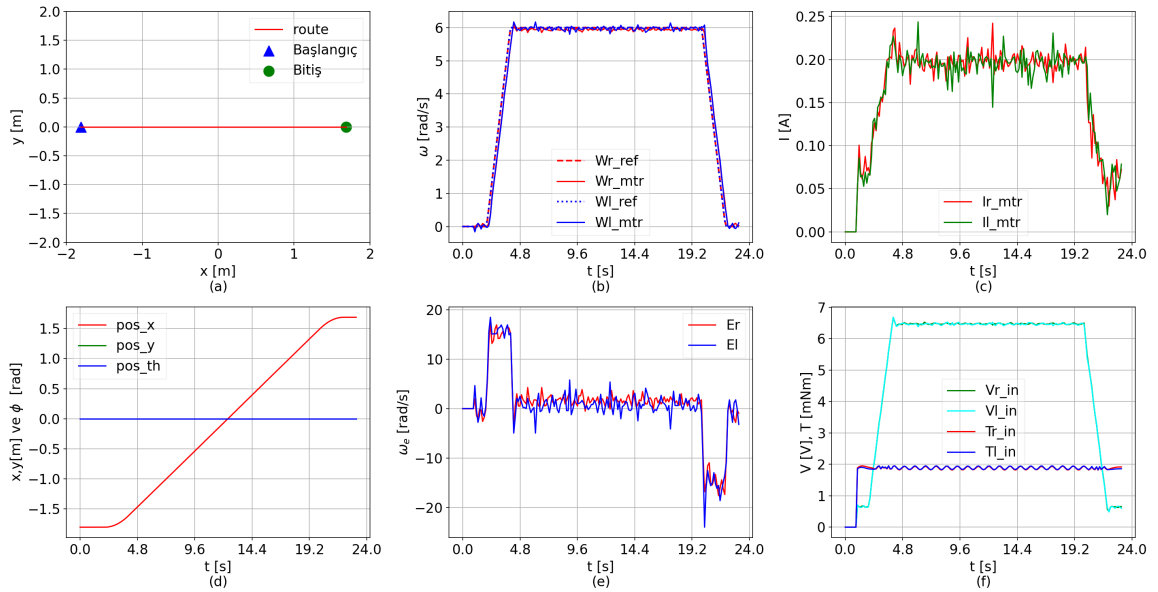
Şekil 5.31. Webots simülatörü ile birlikte simülasyon (doğrusal-1); a) robotun XY düzlemindeki hareket yörüngesi, b) robotun sağ ve sol tekerlekleri için açısal hız referansları ve simülasyon değerlerinin karşılaştırılması, c) sağ ve sol motorların armatür akımlarının grafikleri, d) robotun pozisyon ve yönelim açısının zaman göre değişimi, e) robotun sağ ve sol tekerleklerdeki açısal hız referansları ile oluşan simülasyon değerlerinin karşılaştırmalı gösterimi, f) Webots simülatörünün sağ ve sol tekerlekler için hesapladığı tork değerleri ve motor kontrol birimlerinde hesaplanan besleme gerilimi değerleri

Şekil 5.31.'de görülen deney sonuçları; eğimsiz bir düzlemde 2 kg ağırlık taşıyan robota aittir. Tork değerinin düşüklüğü ve birbirine benzer değerler aldığı görülmektedir. Bu sebeple akım değerleri de benzer değerlere ulaşmaktadır. Hata değerlerinin hızlanma ve yavaşlama sırasında aldıkları yüksek değerler Webots simülatörünün kontrol çevriminin 130 ms civarında olmasından kaynaklanmaktadır. Simülatör kontrol çevrimi süresi düşürülmeye çalışıldığında simülatör gerçek zaman hızına ulaşmada güçlük yaşamaktadır. Deneyler sırasında Webots simülatörü ve dijital ikiz sistemlerin süreleri arasındaki fark %1 değerine yakın tutulmaya çalışılmıştır.



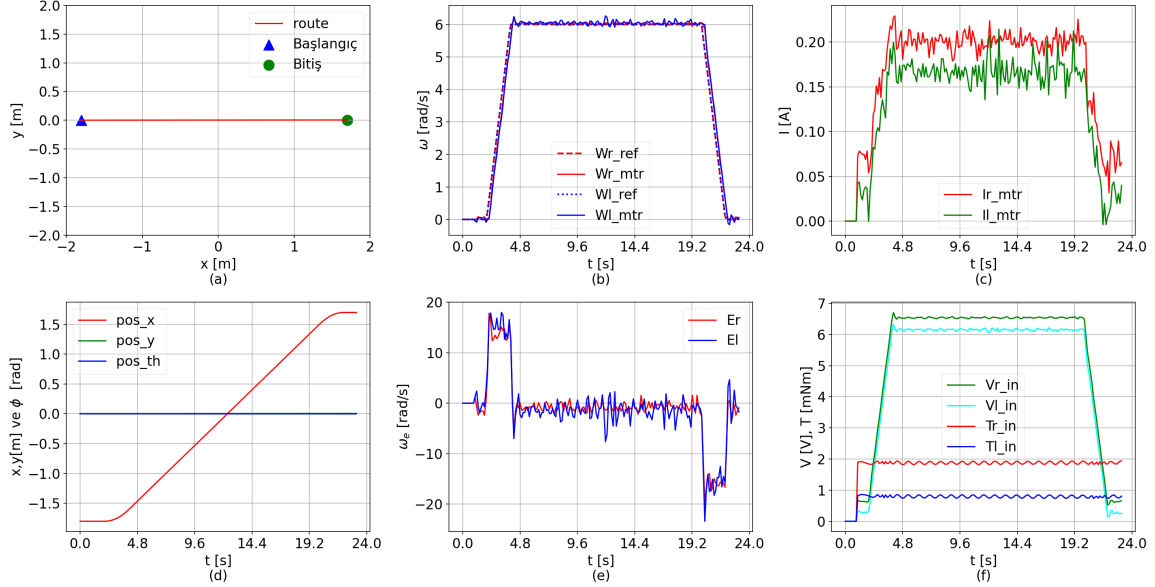
Şekil 5.32. Webots simülâtörü ile birlikte simülasyon (doğrusal-2)

Şekil 5.32.'de görülen deney sonuçları; $x = 0$ ekseninde 0,1 rad eğimli yüzeyde 2 kg ağırlık taşıyan robota aittir. Robot yokuş inerken tork negatif çıkmakta ve motor armatür akımlarının düşük hız değerlerinde negatif olduğu görülmektedir. Ayrıca gerilim ve akım değerlerinin önceki deneyde görülen seviyelerden daha düşük olduğu da görülmektedir. Hata sinyallerinde hızlanma ve yavaşlama bölgelerinde oluşan yüksek değerler Webots simülâtörünün 130 ms değerlerindeki kontrol adım süresinden kaynaklanmaktadır.



Şekil 5.33. Webots simülâtörü ile birlikte simülasyon (doğrusal-3)

Şekil 5.33.'te görülen deney sonuçları; $x = 0$ ekseninde $-0,1$ rad eğimli yüzeyde 2 kg ağırlık taşıyan robota aittir. Robot yokuş çıkarken tork yüksek çıkmakta ve motora uygulanan gerilim ve armatür akımlarının yüksek değerleri gözlemlenebilmektedir.



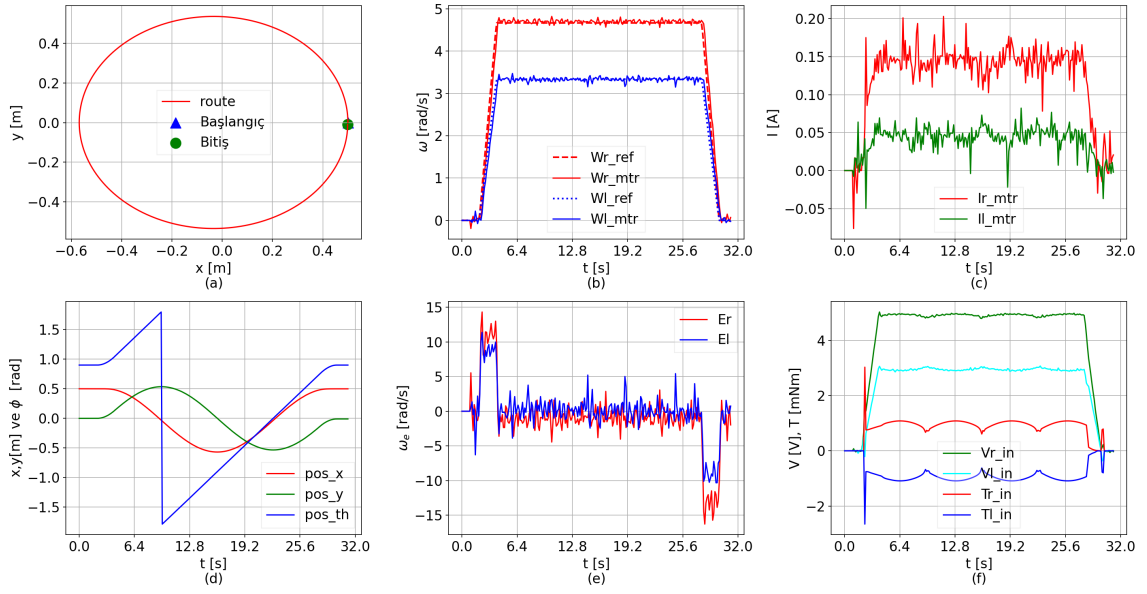
Şekil 5.34. Webots simülatörü ile birlikte simülasyon (doğrusal-4)

Şekil 5.34.'te görülen deney sonuçları; $y = x$ ekseninde $-0,1$ rad eğimli yüzeyde 2 kg ağırlık taşıyan robota aittir. Robot hareket eksenine dik olarak eğimli yüzeyde hareket ederken torklar farklılaşmakta, aşağıda kalan tekerlek için tork değeri daha yüksek oluşmaktadır. Bu durum motorlara uygulanan gerilim değerine ve akıma da yansımaktadır.

Dairesel Hareket için Sonuçlar

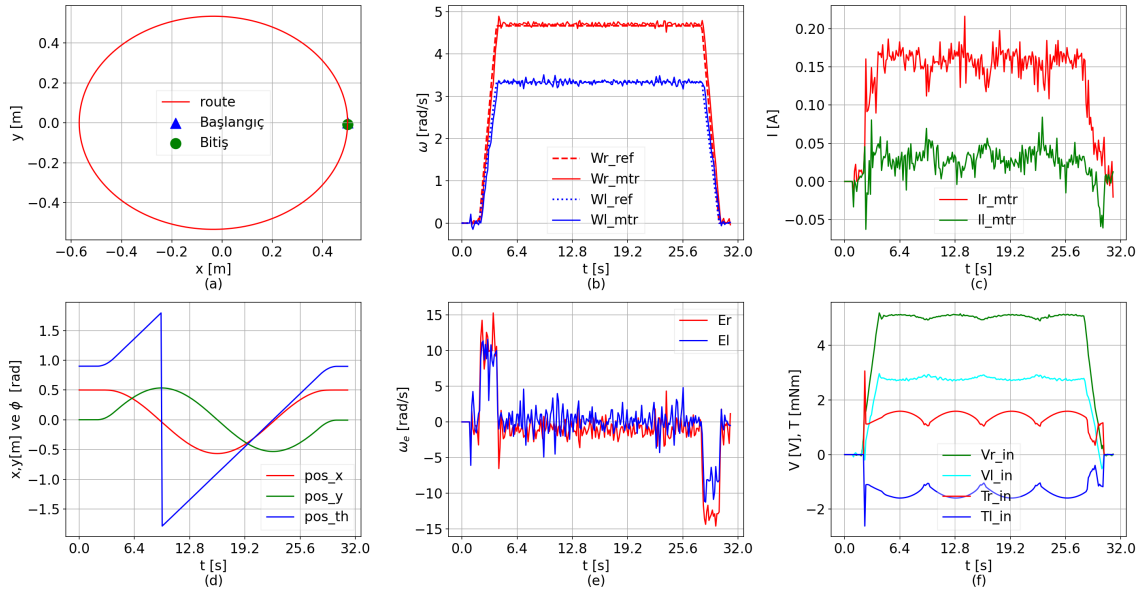
Deneyin bu kısmında robot düzgün doğrusal görüme üzerinde sola dönerek bir halka çizmektedir. Yol üzerinde önce ivmelenerek sabit bir hıza çıkmakta bir süre bu hızı korumakta sonrada yavaşlayıp durmaktadır.

Robot hareket sırasında sabit sola dönüş yaparak çember çizmekte ve bu sırada farklı yüzey eğimleri için yapılan deneylerde simülatörlerde oluşan değerler aşağıdaki şekillerde görülmektedir. Robot sola döndüğü için dış tekerlekte oluşan fazladan tork ve hız motor besleme gerilimi ve armatür akımlarından gözlemlenebilmektedir.



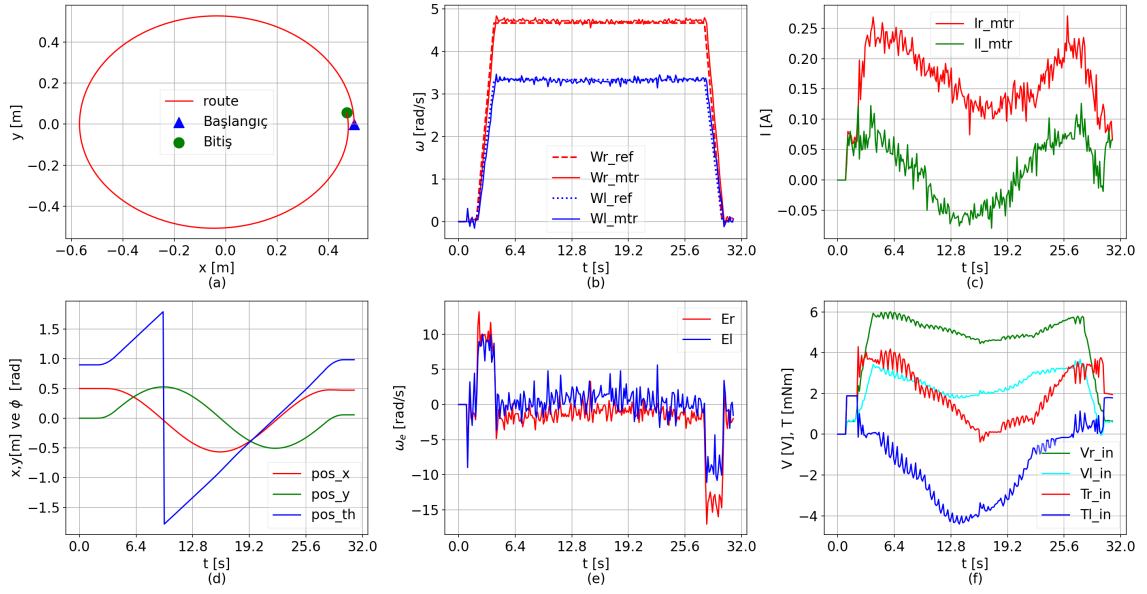
Şekil 5.35. Webots simütörü ile birlikte simülasyon, dairesel-1

Şekil 5.35.'te görülen deney sonuçları; eğimsiz bir yüzeyde sadece kendi ağırlığını taşıyan ve daire çizen bir robota aittir. Robot daire çizerken dairenin iç ve dış tarafları için farklı tork ve akım değerleri görülebilmektedir.



Şekil 5.36. Webots simütörü ile birlikte simülasyon, dairesel-2

Şekil 5.36.'da görülen deney sonuçları; eğimsiz bir yüzeyde 2 kg ağırlık taşıyan ve daire çizen bir robota aittir. Robot üstündeki ağırlıkla daire çizerken dairenin iç ve dış tarafları için farklı tork ve akım değerleri görülebilmektedir.

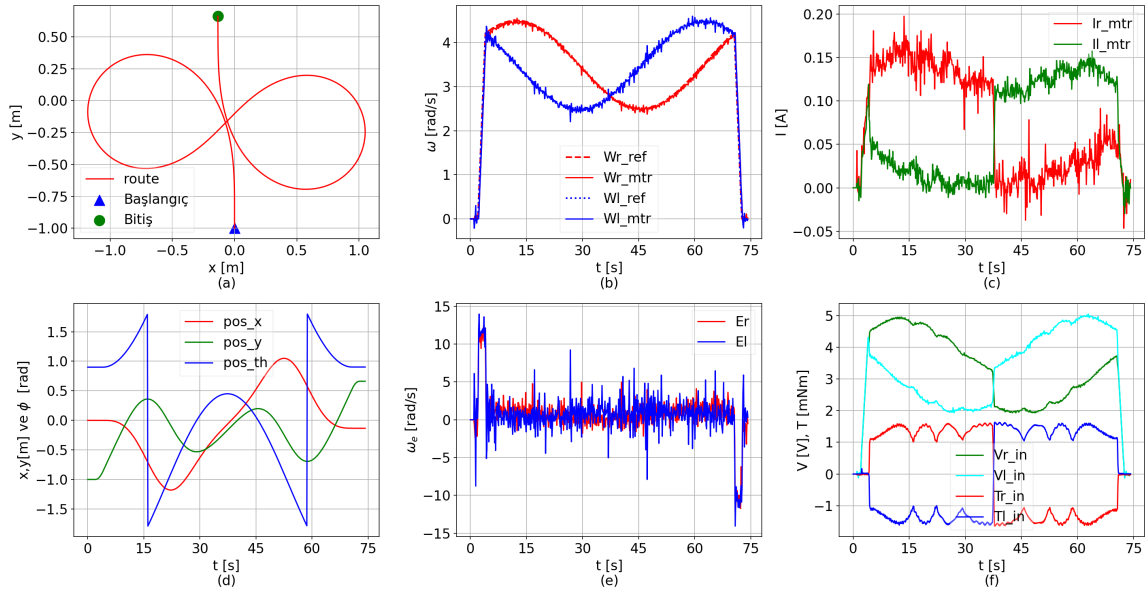


Şekil 5.37. Webots simütörü ile birlikte simülasyon, dairesel-3

Şekil 5.37.'de görülen deney sonuçları; $y = 0$ doğrultusunda 0.1 rad eğimli bir yüzeyde 2 kg ağırlık taşıyan ve daire çizen bir robota aittir. Robot üstündeki ağırlıkla daire çizerken eğimli yüzey için tork ve akım değerlerinin değişimi görülebilmektedir.

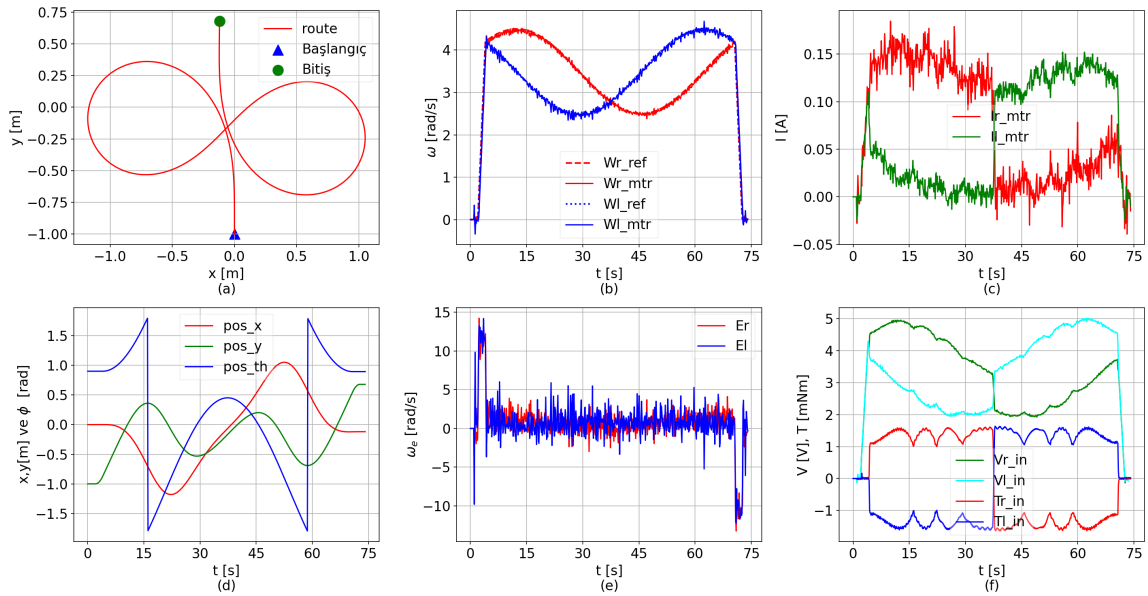
8 Şeklinde Yola Alma Hareketi Sonuçları

Deneyin bu kısmında robot 8 şeklinde hareket etmekte harekete 8 şeklinin sol ortasından girip önce sol sonra sağ tarafta sekiz hareketinin dairelerini çizerek başladığı doğrultudaki yöne geri dönüp durmaktadır. Yol üzerinde önce ivmelenerek sabit bir hıza çıkmakta sonrasında hareketin gerektirdiği hız profilini izleyerek sonrasında yavaşlayıp durmaktadır. Hareketin oluşturulabilmesi için sağ ve sol motorlara uygulanan hız değerlerinde sabit hız bileşeni ile birlikte sağ ve sol için aralarında 90° faz farkı bulunan sinüzoidal hız bileşenleri eklenmiştir. Hareketin ilk hızlanma ve son yavaşlama aşamalarında sinüzoidal hız bileşenleri bulunmamaktadır.



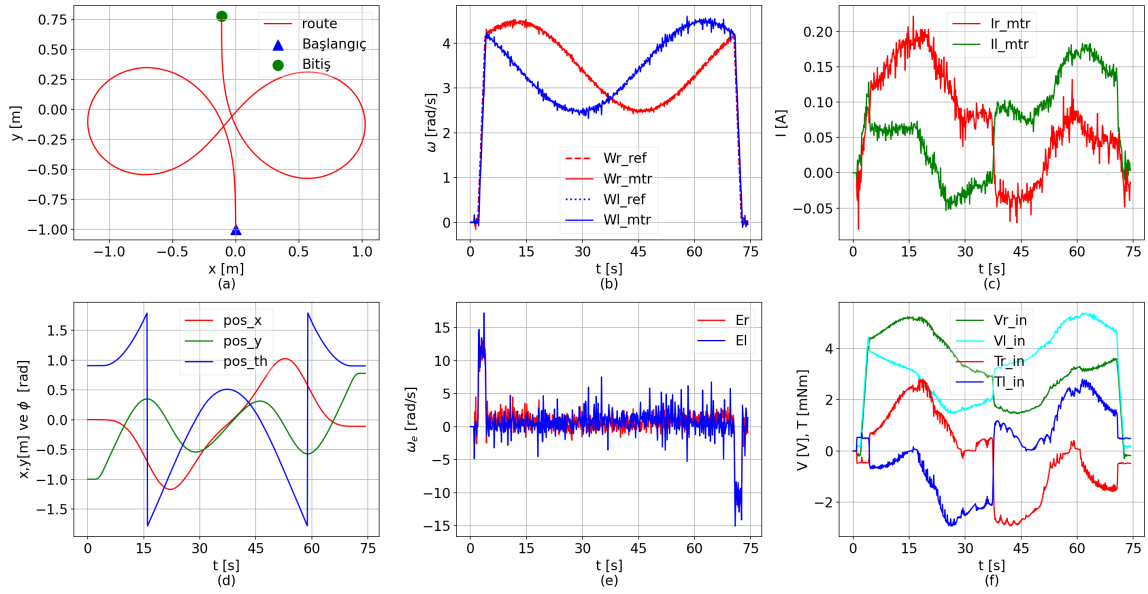
Şekil 5.38. Webots simütörü ile birlikte simülasyon, 8-1

Şekil 5.38.'de görülen deney sonuçları; eğimsiz bir yüzeyde sadece kendi ağırlığını taşıyan ve 8 çizerek yol alan bir robota aittir. Hareketin birbirine zıt dönüş yönlerdeki kısımlarında sağ ve sol motorlar için farklı tork ve akım değerleri gözlemlenebilmektedir.



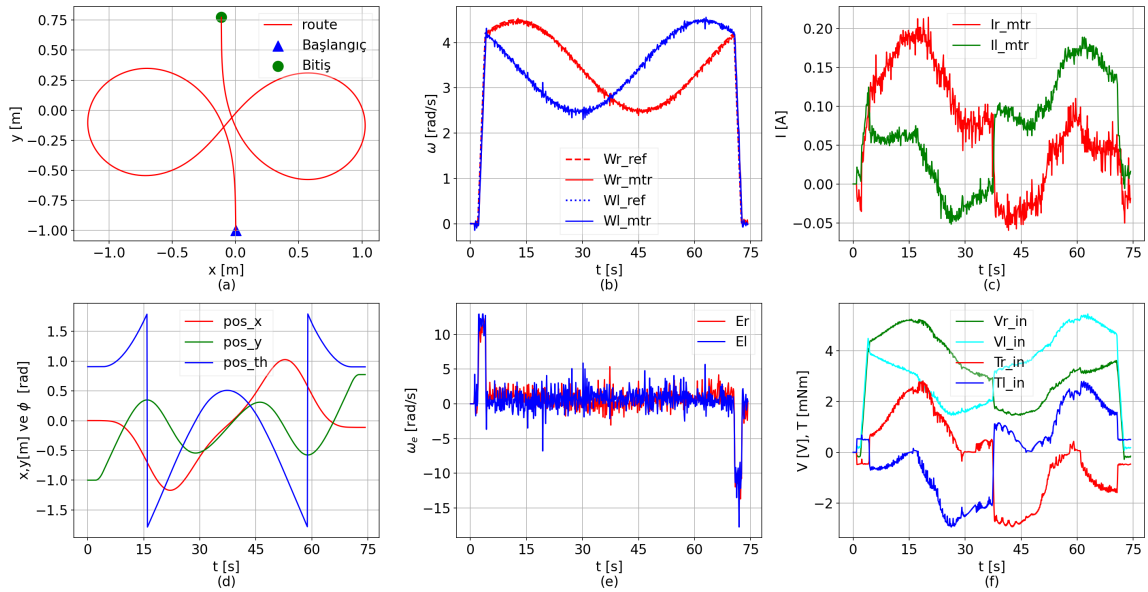
Şekil 5.39. Webots simütörü ile birlikte simülasyon, 8-2

Şekil 5.39.'da görülen deney sonuçları; eğimsiz bir yüzeyde 1 kg ağırlık taşıyan ve 8 çizerek yol alan bir robota aittir. Hareketin yapıldığı düzlemde eğim olmadığı için hareketin ek ağırlıkla yapılan kısmı birbirine oldukça benzer oluşmaktadır.



Şekil 5.40. Webots simütörü ile birlikte simülasyon, 8-3

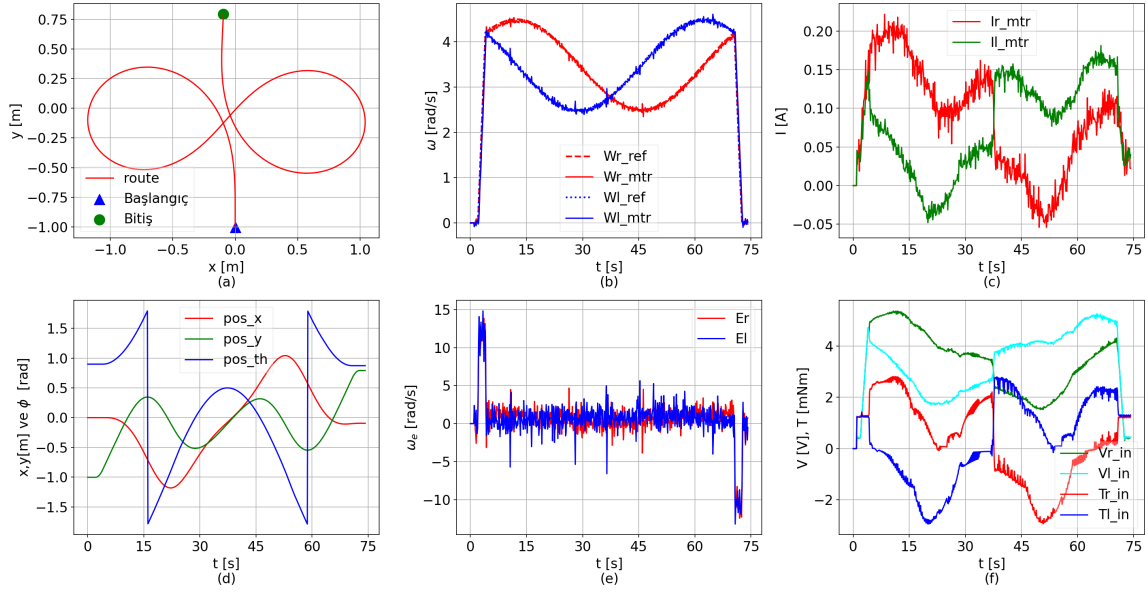
Şekil 5.40.'ta görülen deney sonuçları; $x = 0$ ekseninde 0,1 rad eğimli yüzeyde sadece kendi ağırlığını taşıyan ve 8 çizerek yol alan bir robota aittir. Hareket dilen yüzeyin eğimli olmasından dolayı 8 hareketinin yokuş çıkılan kısımları ve yokuş inilen kısımları arasındaki tork farklılıkları ve bunun motor besleme gerilimlerine ve armatür akımına yansımaları görülebilmektedir.



Şekil 5.41. Webots simütörü ile birlikte simülasyon, 8-4

Şekil 5.41.'de görülen deney sonuçları; $x = 0$ ekseninde 0,1 rad eğimli yüzeyde 1 kg

ağırlık taşıyan ve 8 çizerek yol alan bir robota aittir.



Şekil 5.42. Webots simülatörü ile birlikte simülasyon, 8-5

Şekil 5.42.'de görülen deney sonuçları; $y = 0$ ekseninde 0,1 rad eğimli yüzeyde sadece kendi ağırlığını taşıyan ve 8 çizerek yol alan bir robota aittir. Eğimin uygulandığı vektörün değişimi sonucu yörünge üzerindeki yokuş iniş ve çıkış kısımları değiştiğinden tork sinyali ve beraberinde motor gerilim ve armatür akımlarının şekilleri değişmiştir.

Hareket yüzeyinin eğimli olmasından kaynaklanan yokuş iniş ve çıkış kısımları motor yük torku değerlerini etkilediği gibi robotun dönüş yönü itibarı ile hızlanan ve yavaşlayan motorlar arasındaki hız farkından kaynaklanan tork farklılıkları da en az tırmanma ve iniş bölgeleri kadar önemli görülmektedir. Robotun dönüş yönü itibarı ile dönüş hareketinde iç kısımda kalan motorun tork ve akım değerlerinin negatif değerler alabildiği deney sonuçlarından görülebilmektedir. Hareketin ilgili bölgelerindeki negatif değerli tork değerine sahip motorun çalışma bölgesi değişmekte ve motor jeneratör moduna girmektedir.

6. BULGULAR VE TARTIŞMA

Bu tez çalışmasında; RL devre simülatörü, PMDC motor simülatörü ve Webots simülatörü olmak üzere 3 farklı simülatör çalışması gerçekleştirilmiştir. Gerçek zamanlı bir simülatörün çalışma doğruluğunu gösterebilmek için en temel yöntem olarak direnç ve endüktanstan oluşan seri bir devrenin test edilmesi düşünülmüştür. Gerçek bir seri RL devresi ile FPGA yongasında oluşturulan dijital ikizi aynı gerilim kaynağından beslenerek akım ve gerilim ölçümleri anlık olarak aynı osiloskop ekranından ölçülmüştür. Bu sayede hem gerçek RL sonuçları hem de dijital ikizin sonuçları kıyaslanarak dijital sistemin tepki hızı ve doğruluğu test edilmiştir. Bu aşamada gerçek sistem ile dijital sistemin uyuşmasını sağlayabilmek için gerekli kalibrasyon ayarlarının yapılması gerekmiştir. Çalışmalar sonucunda dar boğaz olarak analog dijital dönüşüm çevrim hızı olduğu tespit edilmiştir. ADC çevrimi ana çevrim grubu hızı $4\mu s$ olduğundan dolayı diğer tüm işlemlerin bu süreden önce yapılması gerekmektedir. Kullanılan FPGA 50 MHz saat hızına sahip olduğundan dolayı işlemler yaklaşık $1\mu s$ sürede tamamlanabilmektedir. Deneysel ve simülasyon çalışmalarında $R = 22\Omega$, endüktans değeri ise $L = 4,825mH$ olarak kullanılmıştır. Akım ve gerilim sonuçları kare dalga, sinüzoidal ve harmonik bindirilmiş sinüzoidal dalga şekilleri için ayrı ayrı analiz edilmiştir. Sonuçlar incelendiğinde dijital ikizin gerçek devre ile aynı tepkileri gerçek zamanlı olarak verdiği gösterilmiştir.

İkinci çalışmada; Maxon firmasının F2140-934 ürün kodlu bir PMDC motoru modellenerek FPGA ortamında dijital ikizi oluşturulmuştur. RL çalışmasından farklı olarak bu çalışmada parametre modülü ve motor simülasyon modülü olmak üzere iki farklı modüle ihtiyaç duyulmuştur. Her bir işlem adımında parametre modülü gerekli parametreleri üreterek motor simülasyon modülüne aktarmıştır. Motorun dijital ikizi için dışarıdan besleme gerilimi ve yük momenti girilmiştir. İki öncelikli kanal ve normal kanallarla beraber etkin örnekleme zamanı $6\mu s$ olarak sağlanmıştır. FPGA simülasyon modüllerinde 32 bit genişliğindeki tek duyarlıklılı sayıları kullanarak hesaplamalar yaptığından dolayı işlem önceliği ve farklı büyüklüklerdeki sayıların beraber işlenmesi aşamasında dikkatli olunması gerekmektedir. PMDC motorun FPGA ortamındaki dijital ikizi için kullanılan FPGA kaynaklarından ALMs için %25, blok hafıza için %0,52, M10Ks için %25, DSP blokları için %47 ve PLL için %25 kaynak kullanımı gerçekleşmiştir. Bir önceki çalışmaya benzer şekilde hem gerçek motor hem de motorun dijital ikizi aynı çalışma koşulları altında çalışmaları sağlanmıştır. Her iki sistemin çıkışları anlık olarak aynı osiloskop ekranına aktarılmış ve sonuçlar analiz edilmiştir. Testler sırasında referans sinyaller kare dalga, sinüzoidal dalga ve rastgele kare dalga sinyal olarak ayarlanmıştır. Motorun dijital ikizinin gerçek motor ile benzer dinamikler sergilediği elde edilen sonuçlar ile ispatlanmıştır. Oluşturulan simülatörün gecikme süresinin $10 - 16\mu s$ arasında değiştiği gözlemlenmiştir. Elde edilen FPGA tabanlı simülatörün yeterli performans gösterdiğini söylemek mümkündür. Bu simülatör ile herhangi bir motor modellenerek gerçek zamanlı olarak herhangi bir deneysel düzeneğe ihtiyaç duymadan dijital olarak çalıştırılmasının

mümkün olacağı gösterilmiştir. Özellikle kontrol algoritması geliştirmek için gerçek bir motorun kullanılması hem zaman hem de maddi açıdan bazı zorluklar oluşturmamasından dolayı böyle bir yaklaşımla birçok avantaj sağlanabilmektedir.

Üçüncü çalışmada; elde edilen simülatörün robotik uygulamalarda da kullanılabileceğini göstermek için farklı bir çalışma sunulmuştur. Günümüzde robotların simülasyonu için birçok robot simülatör programları bulunmaktadır. Ancak bu programlar sadece kinematik ve dinamik hesaplamalar üzerine inşa edildikleri için elektriksel analiz yapmaya müsait değildir. Bu eksikliği gidermek için bu tez çalışmasında oluşturulan dijital motorun mevcut bir robot simülatörüne entegre edilebileceği gösterilmiştir. Bu sayede, robot simülatörlerinin elektriksel olarak gerçek zamanlı analiz yapabilmesi sağlanabilmektedir. Bu çalışmada açık kaynaklı bir robot simülatörü olarak Webots tercih edilmiştir. Webots ortamında popüler mobil robotlardan biri olan Turtlebot3 kullanılmıştır. Webots kullanılarak robotun tekerlek moment değerleri ölçülmüş ve anlık olarak geliştirilen FPGA tabanlı simülatöre aktarılmıştır. Burada FPGA simülatörün görevi gerçek zamanlı olarak robotun iki motorunun akım ve hız bilgilerini hesapladıktan sonra kapalı çevrim kontrolüne göndermektir. Böylece robot simülasyonunda olmayan elektriksel hesaplamalar anlık olarak gerçekleştirilmiştir. Webots 32 ms döngü süresine ihtiyaç duymaktadır. Bu süre geliştirilen FPGA tabanlı simülatör programından yaklaşık 2000 kat daha fazladır. Bu nedenle geliştirilen dijital ikiz herhangi bir robot simülasyon sisteminde kullanılabilir.

7. SONUÇLAR

Simülâtör sistemlerinin ve özel anlamda gerek zamanlı simülâtör sistemlerinin gerekliliğı tartışma götürmeyen bir gerçek olarak endüstride durmaktadır. Yapılan bu çalışma ile çalışma anında parametre değışimine izin veren gerçek zamanlı farklı simülâtör tasarımları gerçekleştirilmiştir. Bu yaklaşım simülâtörün iş yükünü ve hacmini arttırsa da çalışma anında yüklenme parametrelerinin ayarlanabilmesi gibi kayda değer kazanımları beraberinde getirmektedir.

FPGA yongaları dijital sistem elemanı olarak “kök hücre” benzeri esnekliğı ile neredeyse çoğı alanda kendine yer edinmiş bir teknoloji olarak bu çalışmada da iş yükünü üstüne almıştır. Karşılaşılan olumsuzlukların büyük kısmının kullanıcıdan kaynaklanan problemler olduğunu zaman ilerledikçe gördüğümüz bu teknoloji, ya doğrudan kullanılacak ya da ücreti mukabilinde kullanılacak şekilde gittikçe yaygınlaşacaktır.

Sistem tasarımlarında ve tesislerde tehlikeden, zaman kaybından ve maliyetten kaçınmanın muhtemelen en önemli yollarından birisi RTDS kullanımı olarak görünmektedir. FPGA yapıları RTDS üreticilerinin de üstünde ağırlıkla durdukları bir seçenektir. Yeniden ayarlanabilen yapıları ile çalışmamızda da olduğu gibi farklı RTDS yapılarının aynı donanımda çalıştırılması maliyet yükünün azalmasına ve daha fazla kişi ve sektörün bu olanaktan faydalanabilmesine zemin hazırlamıştır.

Bu tezde tasarlanan sistemler önemli ölçüde başarılı sonuçlar vermişlerdir. FPGA ile sistem tasarlamayı ve özellikle zamanlama analizlerini öğrenirken harcadığımız zaman, tüm hedeflerimize ulaşmamızı engellediyse bile önümüzde açtığı kapılar zahmetlere değecek nitelikte olduğu açıktır.

Bir sistemin denenmesi gerektiğinde belki de kullanımının uygun olmayacağı anlaşılabilecek donanımların numunelerinin tedariğine veya harcanmasına, zaman kayıplarına ve tehlikeli durumlarla uğraşmayı gerektirmeden sistemin çalışır durumda performansını ve muhtemel hata durumları ile birlikte hataya verilen cevapları görebilmek büyük bir olanaktır.

Sonuç olarak bu tez çalışması; herhangi bir sistemin dijital ikizinin oluşturularak gerçek zamanlı ortamda test edilmesine olanak sağlayacak FPGA tabanlı sistem geliştirilmesine bir kılavuz olarak kullanılabilecektir.

GELECEK ÇALIŞMALAR VE ÖNERİLER

Çalışma yaptığımız alana ilgisi olan geliştiricilerin FPGA ile donanım geliştirmesini öğrenme yükünü omuzlamaları ilk tavsiyemizdir. Özellikle sorunsuz çalışan sistemler tasarlamak ardından ne olacak diye bakmaya gerek duymamak için bu çok önemli bir kazanım olacaktır.

Tasarımlarımızda basit bir PMDC motor ile ortalama hacimdeki bir FPGA yongasının DSP blok kullanım oranı büyük FPGA yongaları ve özel geliştirme kartlarına duyulan ihtiyacı açıklamaktadır. Düşük kapasitelerde FPGA yongaları ile model simülasyonu yapmaya kalkışmamak gerekmektedir. Ancak şayet büyük bir FPGA ile gerçekleştirildiğinde kullanım raporlarında yeterince düşük rakamlar görülüyorsa küçük boyutlu FPGA yongaları kullanılmalıdır.

Simülatör tasarımlarında sistem modeli oluşturma ve sistemi FPGA yongasına aktarma işlemlerinin birbirinden habersiz kalmayacak şekilde ayrılmaları avantajlar doğuracaktır. Sistem modelleme işi ve donanım geliştirme işlemleri gerçek anlamda birbirlerinden farklı şeylerdir.

FPGA tasarımlarında yonga içinde işlemci kullanımının avantajları gelecek nesil RTDS birimleri için umut vadetmektedir. Bu anlamda FPGA içinde işlemci veya işlemciye yardımcı bir FPGA ünitesinin olanakları, yapılacak tasarımlarla zorlu işleri başarabilecek bir ekip gibi durmaktadır. Bu işlemlerin başlangıç noktası olan bilgisayar donanımları bilgisi işe başlarken büyük avantajlar sağlayacaktır.

KAYNAKLAR

- [1] Jean Bélanger, P Venne, and Jean-Nicolas Paquin. (2010). The what, where and why of real-time simulation. *Planet Rt*, 1(1):25–29.
- [2] J. Banks. (2010). *Discrete-event System Simulation*. Prentice-Hall international series in industrial and systems engineering. Pearson. ISBN 9780138150372. URL <https://books.google.com.tr/books?id=LNV9PgAACAAJ>.
- [3] Anne H. Soukhanov, editor. (1992). *The American Heritage Dictionary of the English Language*. Houghton Mifflin. ISBN 9780395448953. URL <https://books.google.com.tr/books?id=WWYNAQAAMAAJ>.
- [4] M.T. Cicero, J.E.G. Zetzel, and P.E. Easterling. (1995). *Cicero: De Re Publica: Selections*. Cambridge Greek and Latin Classics. Cambridge University Press. ISBN 9780521348966. URL <https://books.google.com.tr/books?id=mNvL-eJMUgAC>.
- [5] Derek Solla de Price. (1974). Gears from the greeks. the antikythera mechanism: a calendar computer from ca. 80 bc. *Transactions of the American Philosophical Society*, pages 1–70.
- [6] Tony Freeth, Yanis Bitsakis, Xenophon Moussas, John H Seiradakis, Agamemnon Tselikas, Helen Mangou, Mary Zafeiropoulou, Roger Hadland, David Bate, Andrew Ramsey, et al. (2006). Decoding the ancient greek astronomical calculator known as the antikythera mechanism. *Nature*, 444(7119):587–591.
- [7] JH Seiradakis and MG Edmunds. (2018). Our current knowledge of the antikythera mechanism. *Nature Astronomy*, 2(1):35–42.
- [8] Tony Freeth, David Higgon, Aris Dacanalis, Lindsay MacDonald, Myrto Georgakopoulou, and Adam Wojcik. (2021). A model of the cosmos in the ancient greek antikythera mechanism. *Scientific reports*, 11(1):1–15.
- [9] Larry Owens. (1986). Vannevar bush and the differential analyzer: The text and context of an early computer. *Technology and Culture*, 27(1):63–95. ISSN 0040165X, 10973729. URL <http://www.jstor.org/stable/3104945>.
- [10] J. V. Wait. (1967). State-space methods for designing digital simulations of continuous fixed linear systems. *IEEE Transactions on Electronic Computers*, EC-16(3):351–355, June. ISSN 0367-7508. doi: 10.1109/PGEC.1967.264693.
- [11] J.J. Sanchez-Gasca, R. D’Aquila, W.W. Price, and J.J. Paserba. (1995). Variable time step, implicit integration for extended-term power system dynamic simulation. In *Proceedings of Power Industry Computer Applications Conference*, pages 183–189. IEEE, May. doi: 10.1109/PICA.1995.515182.
- [12] Hermann W. Dommel. (1969). Digital computer solution of electromagnetic transients in single-and multiphase networks. *IEEE Transactions on Power Apparatus and Systems*, PAS-88(4):388–399, April. ISSN 0018-9510. doi: 10.1109/TPAS.1969.292459.
- [13] Kevin Kawal, Qiteng Hong, Subhadeep Paladhi, Di Liu, Panagiotis N Papadopoulos, Steven Blair, and Campbell Booth. (2022). Vulnerability assessment of line current differential protection in converter-dominated power systems. In *IET 16th International Conference on Developments in Power System Protection*, volume 2022, pages 1–6, March. doi: 10.1049/icp.2022.0967.
- [14] Alejandro Avendaño. (2022). Microgrid controller evaluation using real-time digital simulation. In *2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5. IEEE, April. doi: 10.1109/ISGT50606.2022.9817558.

- [15] Mojtaba Afshar, Mehrdad Majidi, Oveis Asgari Gashteroodkhani, and Mehdi Etezadi Amoli. (2022). High impedance fault detection in a practical platform using a real-time-digital simulator. In *2022 IEEE Texas Power and Energy Conference (TPEC)*, pages 1–6. IEEE, February. doi: 10.1109/TPEC54980.2022.9750700.
- [16] Hua Li, Keqilao Meng, Yufei Peng, and Xiaoyan Li. (2022). Control strategy for seamless switching of virtual synchronous generators based on secondary frequency and voltage regulation. *Alexandria Engineering Journal*, 61(12):10477–10484. ISSN 1110-0168. doi: 10.1016/j.aej.2022.03.057. URL <https://www.sciencedirect.com/science/article/pii/S1110016822002320>.
- [17] AB Dehkordi and TL Maguire. (2021). A multi-star synchronous machine model for real-time digital simulation and its applications. *Electric Power Systems Research*, 197:107312. ISSN 0378-7796. doi: 10.1016/j.epsr.2021.107312. URL <https://www.sciencedirect.com/science/article/pii/S0378779621002935>.
- [18] H. Duchen, M. Lagerkvist, R. Kuffel, and R.P. Wierckx. (1995). Hvdc simulation and control system testing using a real-time digital simulator (rtds). In *ICDS '95. First International Conference on Digital Power System Simulators*, pages 213–. IEEE, April. doi: 10.1109/ICDS.1995.492832.
- [19] R. Kuffel, J. Giesbrecht, T. Maguire, R.P. Wierckx, and P. McLaren. (1995). Rtds-a fully digital power system simulator operating in real time. In *Proceedings 1995 International Conference on Energy Management and Power Delivery EMPD '95*, volume 2, pages 498–503 vol.2. IEEE, Nov. doi: 10.1109/EMPD.1995.500778.
- [20] P.G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt. (1992). A real time digital simulator for testing relays. *IEEE Transactions on Power Delivery*, 7(1):207–213, Jan. ISSN 1937-4208. doi: 10.1109/61.108909.
- [21] Yasushi Fujimoto. (1999). Real-time power system simulator on a pc cluster. *Proc. of the IPST'99*.
- [22] Takaya Yamamoto, Hiroshi Enomoto, Kenichi Kuroda, Yoshiyuki Kono, and Hisao Taoka. (2004). Real-time power system simulator using pc cluster. *IEEJ Transactions on Power and Energy*, 124(5):733–740, January.
- [23] J.A. Hollman and J.R. Marti. (2003). Real time network simulation with pc-cluster. *IEEE Transactions on Power Systems*, 18(2):563–569, May. ISSN 1558-0679. doi: 10.1109/TPWRS.2002.804917.
- [24] Jean Bélanger, Vincent Lapointe, Christian Dufour, and Loic Schoen. (2007). emegasim: An open high-performance distributed real-time power grid simulator. architecture and specification. In *Proceedings of the International Conference on Power Systems (ICPS 2007)*, pages 12–24. Citeseer.
- [25] M. Bacic. (2005). On hardware-in-the-loop simulation. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 3194–3198. IEEE, Dec. doi: 10.1109/CDC.2005.1582653.
- [26] R. Isermann, J. Schaffnit, and S. Sinsel. (1999). Hardware-in-the-loop simulation for the design and testing of engine-control systems. *Control Engineering Practice*, 7(5):643–653. ISSN 0967-0661. doi: 10.1016/S0967-0661(98)00205-6. URL <https://www.sciencedirect.com/science/article/pii/S0967066198002056>.
- [27] Mitchell E. Sisle and Edward D. McCarthy. (1982). Hardware-in-the-loop simulation for an active missile. *SIMULATION*, 39(5):159–167. doi: 10.1177/003754978203900503. URL <https://doi.org/10.1177/003754978203900503>.

- [28] Chet A DeCesaris Jr, Paul J Millner, and Jon Forst. (1996). Role of interceptor hardware-in-the-loop testing in ballistic missile defense programs. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing*, volume 2741, pages 2–13. SPIE.
- [29] John S Cole Jr and Alexander C Jolly. (1996). Hardware-in-the-loop simulation at the us army missile command. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing*, volume 2741, pages 14–19. SPIE.
- [30] William J Sholes, Elizabeth A Waites, and David W Linden. (1996). Hardware-in-the-loop simulation for a weapon system with active millimeter-wave, passive millimeter-wave, and infrared sensors. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing*, volume 2741, pages 354–359. SPIE.
- [31] Thomas E O’Bannon and Scott A Gearhart. (1996). Dual-mode infrared and radar hardware-in-the-loop test assets at the johns hopkins university applied physics laboratory. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing*, volume 2741, pages 332–346. SPIE.
- [32] Annie V Saylor, Douglas P Barr, Bobby G Woodruff, and Larry T Lowe. (1996). W-band point-scatterer target models for hwil simulations. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing*, volume 2741, pages 304–315. SPIE.
- [33] Minoru Shibata and Haruo Notoh. (1996). Avionics simulator. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing*, volume 2741, pages 366–375. SPIE.
- [34] Larry K Harbor. (1996). Space-based interceptor hardware-in-the-loop simulation featuring ir scene projection and dual-sensor ir and lidar signal injection. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing*, volume 2741, pages 439–446. SPIE.
- [35] Marion S Skalka, William L Gash, Robert Lee Murrer Jr, and Richard A Woodard. (1996). Twenty years of hardware-in-the-loop simulation at eglin air force base, florida: lessons learned. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing*, volume 2741, pages 20–32. SPIE.
- [36] Michelle Bailey and Julie Doerr. (1996). Contributions of hardware-in-the-loop simulations to navy test and evaluation. In *Technologies for synthetic environments: Hardware-in-the-loop testing*, volume 2741, pages 33–43. SPIE.
- [37] Michael Marcin Jr. (1996). Use of hardware-in-the-loop (hwil) testing from early software development through final system test to mitigate risk and to assure mission success for the hera targets program. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing*, volume 2741, pages 409–421. SPIE.
- [38] Hirofumi Eguchi and Tadashi Yamashita. (2000). Benefits of hwil simulation to develop guidance and control systems for missiles. In *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing V*, volume 4027, pages 66–73. SPIE.
- [39] Randall C Restle. (2000). Choosing between dsps, fpgas, μ ps, and asics to implement digital signal processing. In *Conference proceedings of ICSPAT: DSP world, Autumn*.
- [40] Omur Aydogmus and Gullu Boztas. (2019). Implementation of an fpga-based motor control with low resource utilization. In *2019 4th International Conference on Power Electronics and their Applications (ICPEA)*, pages 1–5. doi: 10.1109/ICPEA1.2019.8911149.
- [41] M. Matar, M. Abdel-Rahman, and A. Soliman. (2004). Developing an fpga coprocessor for real-time simulation of power systems. In *International Conference on Electrical, Electronic and Computer Engineering, 2004. ICEEC ’04.*, pages 791–794. IEEE, Sep. doi: 10.1109/ICEEC.2004.1374598.

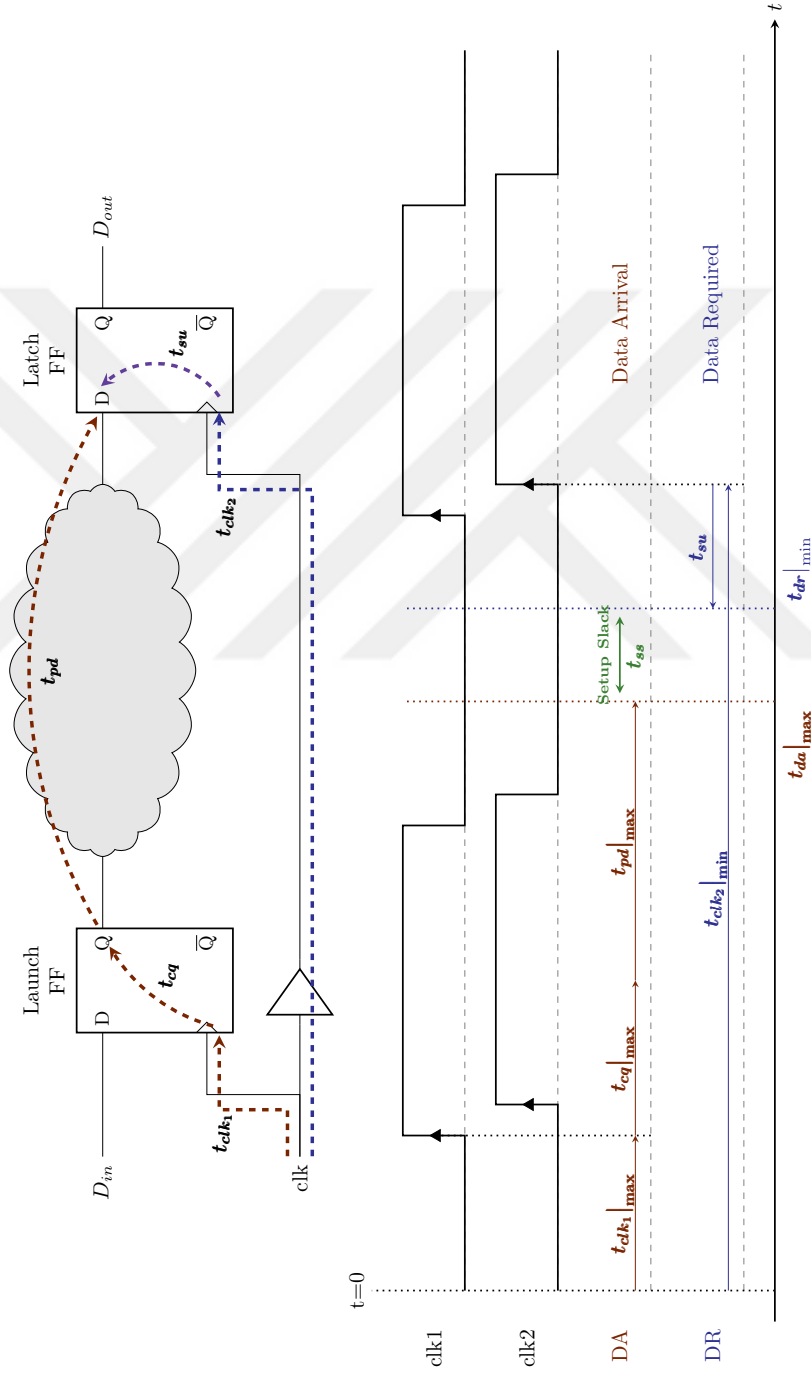
- [42] Mahmoud Matar, Mohamed Abdel-Rahman, and Abdel-Mohaimen Soliman. (2005). Fpga-based real-time digital simulation. In *Int. Conf. on Power Systems Transients*.
- [43] K. Jayalakshmi and V. Ramanarayanan. (2006). Real-time simulation of electrical machines on fpga platform. In *2006 India International Conference on Power Electronics*, pages 259–263. IEEE. doi: 10.1109/IICPE.2006.4685378.
- [44] Christian Dufour, Simon Abourida, Jean Belanger, and Vincent Lapointe. (2006). Real-time simulation of permanent magnet motor drive on fpga chip for high-bandwidth controller tests and validation. In *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*, pages 4581–4586. IEEE, Nov. doi: 10.1109/IECON.2006.347676.
- [45] Christian Dufour, Jean Belanger, Simon Abourida, and Vincent Lapointe. (2007). Real-time simulation of finite-element analysis permanent magnet synchronous machine drives on a fpga card. In *2007 European Conference on Power Electronics and Applications*, pages 1–10. IEEE, Sep. doi: 10.1109/EPE.2007.4417435.
- [46] Yuan Chen and Venkata Dinavahi. (2009). Fpga-based real-time emtp. *IEEE Transactions on Power Delivery*, 24(2):892–902, April. ISSN 1937-4208. doi: 10.1109/TPWRD.2008.923392.
- [47] Eftichios Koutroulis, Kostas Kalaitzakis, and Vasileios Tzitzilonis. (2009). Development of an fpga-based system for real-time simulation of photovoltaic modules. *Microelectronics Journal*, 40(7):1094–1102. ISSN 0026-2692. doi: 10.1016/j.mejo.2008.05.014. URL <https://www.sciencedirect.com/science/article/pii/S002626920800236X>. Mixed-Technology Testing Rapid System Prototyping.
- [48] Mahmoud M. A. M. Bayoumi. (2009). *An FPGA-Based Real-Time Simulator for the Analysis of Electromagnetic Transients in Electrical Power Systems*. PhD thesis. URL <https://www.proquest.com/dissertations-theses/fpga-based-real-time-simulator-analysis/docview/1353186129/se-2>. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-03-04.
- [49] Mahmoud Matar and Reza Iravani. (2009). Fpga implementation of the power electronic converter model for real-time simulation of electromagnetic transients. *IEEE Transactions on Power Delivery*, 25(2):852–860, April. ISSN 1937-4208. doi: 10.1109/TPWRD.2009.2033603.
- [50] S. Usenmez, R.A. Dilan, M. Dolen, and A.B. Koku. (2009). Real-time hardware-in-the-loop simulation of electrical machine systems using fpgas. In *2009 International Conference on Electrical Machines and Systems*, pages 1–6. IEEE, Nov. doi: 10.1109/ICEMS.2009.5382811.
- [51] M. Dagbagi, L. Idkhajine, E. Monmasson, and I. Slama-Belkhodja. (2012). Fpga implementation of power electronic converter real-time model. In *International Symposium on Power Electronics Power Electronics, Electrical Drives, Automation and Motion*, pages 658–663. IEEE, June. doi: 10.1109/SPEEDAM.2012.6264543.
- [52] Morteza Rezaei Larijani, Mohammad-Reza Zolghadri, and Mahmoud Shahbazi. (2016). Design and implementation of an fpga-based real-time simulator for h-bridge converter. In *2016 7th power electronics and drive systems technologies conference (PEDSTC)*, pages 504–510. IEEE.
- [53] Jiadai Liu, Yuan Chen, Hui Ding, and Yi Zhang. (2021). Development of phase domain frequency-dependent transmission line model on fpga for real-time digital simulator. *Electric Power Systems Research*, 197:107305. ISSN 0378-7796. doi: 10.1016/j.epsr.2021.107305. URL <https://www.sciencedirect.com/science/article/pii/S0378779621002868>.
- [54] Conghuan Yang, Ying Xue, Xiao-Ping Zhang, Yi Zhang, and Yuan Chen. (2018). Real-time fpga-rtds co-simulator for power systems. *IEEE Access*, 6:44917–44926. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2862893.

- [55] Bingda Zhang, Yang Wang, Sijia Tu, and Zhao Jin. (2018). Fpga-based real-time digital solver for electro-mechanical transient simulation. *Energies*, 11(10):2650. ISSN 1996-1073. doi: 10.3390/en11102650. URL <https://www.mdpi.com/1996-1073/11/10/2650>.
- [56] M Dagbagi. (2015). *FPGA-based embedded real time simulation of electrical systems*. PhD thesis, Ph. D. Thesis, CY Cergy-Paris University, Cergy. URL <http://www.theses.fr/2015CERG0808>.
- [57] N. Chettibi and A. Mellit. (2014). Fpga-based real time simulation and control of grid-connected photovoltaic systems. *Simulation Modelling Practice and Theory*, 43:34–53. ISSN 1569-190X. doi: 10.1016/j.simpat.2014.01.004. URL <https://www.sciencedirect.com/science/article/pii/S1569190X14000057>.
- [58] Chen Liu, Rui Ma, Hao Bai, Franck Gechter, and Fei Gao. (2018). A new approach for fpga-based real-time simulation of power electronic system with no simulation latency in subsystem partitioning. *International Journal of Electrical Power & Energy Systems*, 99: 650–658. ISSN 0142-0615. doi: 10.1016/j.ijepes.2018.01.053. URL <https://www.sciencedirect.com/science/article/pii/S014206151732149X>.
- [59] Daniel Tormo, Ricardo Vidal-Albalade, Lahoucine Idkhajine, Eric Monmasson, and Ramon Blasco-Gimenez. (2022). Embedded real-time simulator for sensorless control of modular multi-level converters. *Electronics*, 11(5). ISSN 2079-9292. doi: 10.3390/electronics11050719. URL <https://www.mdpi.com/2079-9292/11/5/719>.
- [60] LA Hernández and S Hernández. (1997). Application of digital 3d models on urban planning and highway design. *WIT Transactions on The Built Environment*, 33.
- [61] Michael Grieves. (2014). Digital twin: manufacturing excellence through virtual factory replication. *White paper*, 1(2014):1–7.
- [62] M Shafto, M Conroy, R Doyle, E Gleassgen, C Kemp, J LeMoigne, and L Wang. (2010). Draft modelling, simulation, information technology & processing roadmap—technology area 11. *Washington, DC: National Aeronautics and Space Administration*, 11.
- [63] CK Lo, CH Chen, and Ray Y Zhong. (2021). A review of digital twin in product design and development. *Advanced Engineering Informatics*, 48:101297. ISSN 1474-0346. doi: 10.1016/j.aei.2021.101297. URL <https://www.sciencedirect.com/science/article/pii/S1474034621000513>.
- [64] Abdulmotaleb El Saddik. (2018). Digital twins: The convergence of multimedia technologies. *IEEE MultiMedia*, 25(2):87–92, Apr. ISSN 1941-0166. doi: 10.1109/MMUL.2018.023121167.
- [65] Yuqian Lu, Chao Liu, I Kevin, Kai Wang, Huiyue Huang, and Xun Xu. (2020). Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and Computer-Integrated Manufacturing*, 61:101837.
- [66] Qinglin Qi, Fei Tao, Ying Zuo, and Dongming Zhao. (2018). Digital twin service towards smart manufacturing. *Procedia Cirp*, 72:237–242. ISSN 2212-8271. doi: 10.1016/j.procir.2018.03.103. URL <https://www.sciencedirect.com/science/article/pii/S2212827118302580>. 51st CIRP Conference on Manufacturing Systems.
- [67] Sridhar Gangadharan and Sanjay Churiwala. (2013). *Constraining Designs for Synthesis and Timing Analysis A Practical Guide to Synopsys Design Constraints*. Springer. ISBN 978-1-4614-3268-5.
- [68] David R Coelho. (2012). *The VHDL handbook*. Springer Science & Business Media.
- [69] Donald Thomas and Philip Moorby. (2008). *The Verilog® hardware description language*. Springer Science & Business Media,.

- [70] Rakesh Chadha and J Bhasker. (2009). *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer. ISBN 978-0-387-93819-6.
- [71] Gordon E. Moore. (2006). Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, 11(3):33–35. doi: 10.1109/N-SSC.2006.4785860.
- [72] *Design Guidelines and Timing Closure Techniques for HardCopy ASICs*, July 2010.
- [73] *Cyclone V Device Overview*, May 2018.
- [74] Analog discovery 2 reference manual. URL <https://digilent.com/reference/test-and-measurement/analog-discovery-2/reference-manual>.
- [75] Waveforms reference manual. URL <https://digilent.com/reference/software/waveforms/waveforms-3/reference-manual>.
- [76] Digital discovery reference manual. URL <https://digilent.com/reference/test-and-measurement/digital-discovery/reference-manual>.
- [77] Justin S. J. Wong. (2011). *Delay Measurements and Self Characterisation on FPGAs*. PhD thesis, Imperial College London, October.

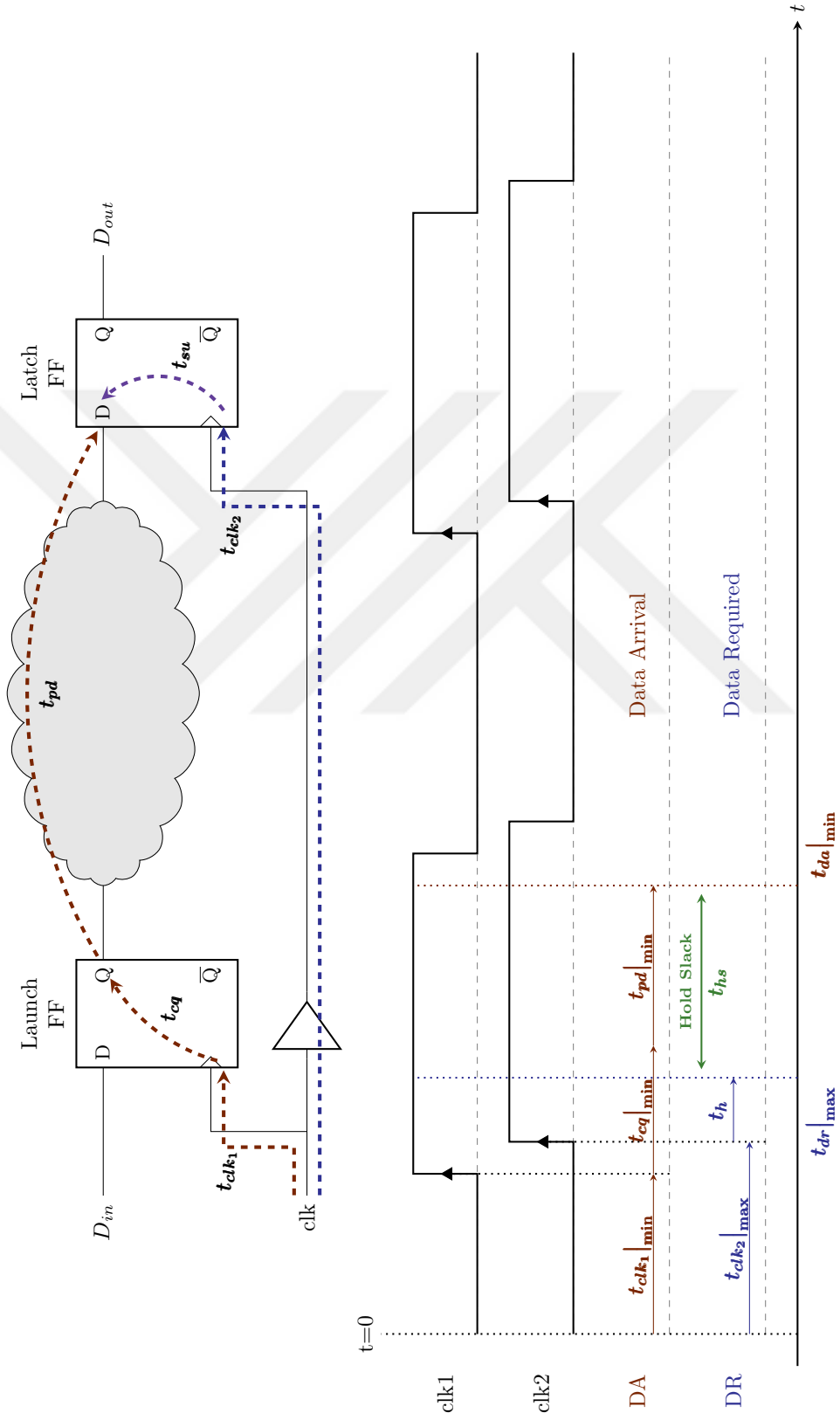
EKLER

EK-1: SETUP SLACK (KURMA PAYI) HESAPLAMA ŞEKLİ



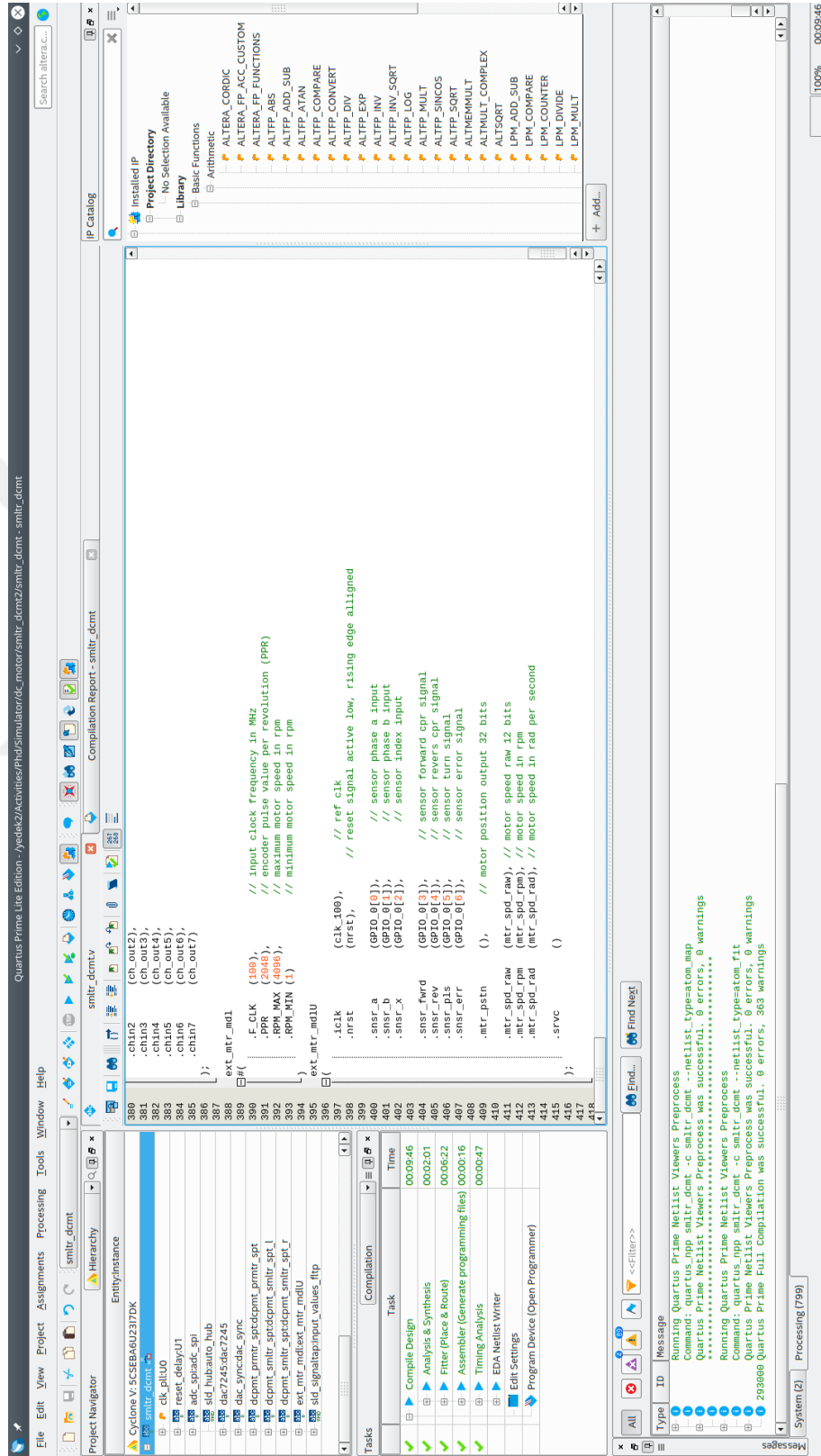
Şekil 7.1. Setup Slack (Kurma Payı) hesaplama şekli

EK-2: HOLD SLACK (TUTMA PAYI) HESAPLAMA ŞEKLİ

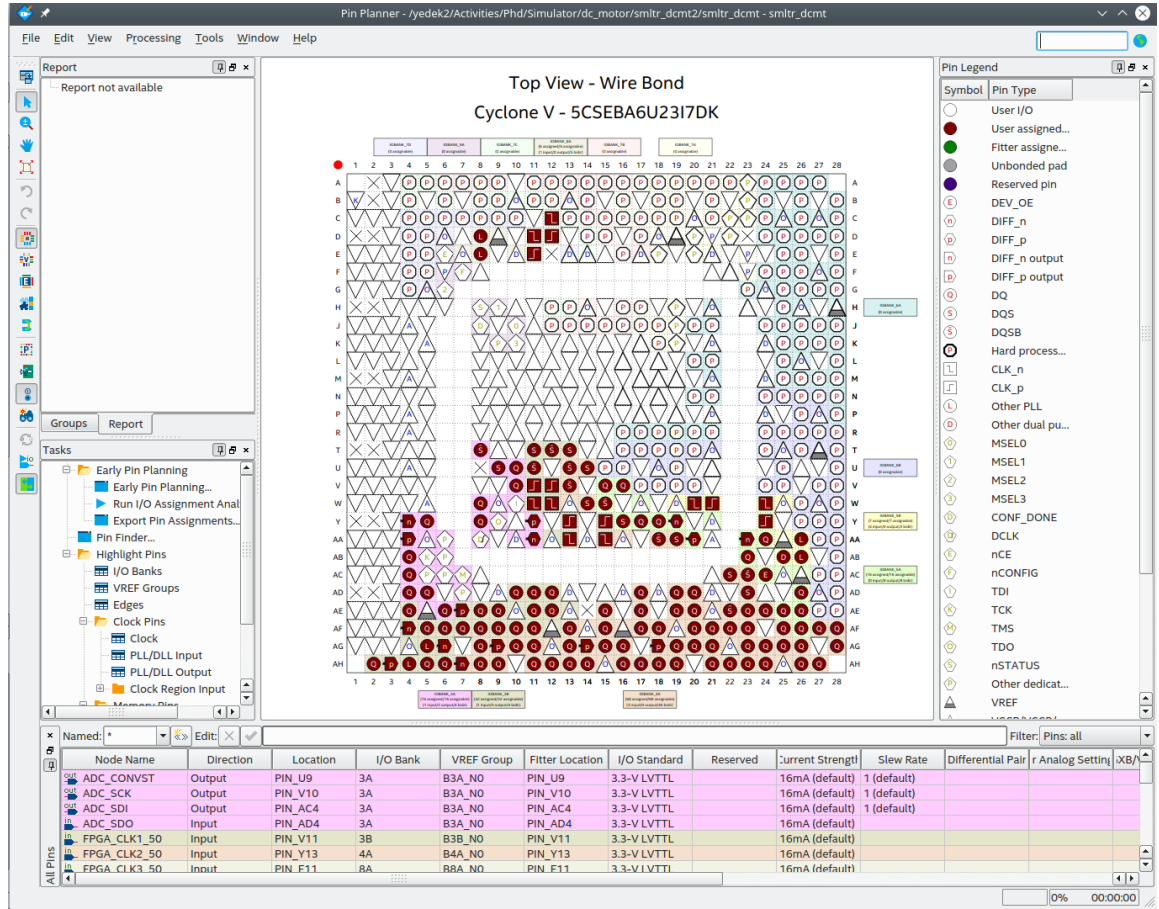


Şekil 7.2. Hold Slack (Tutma Payı) hesaplama şekli

EK-3: İNTEL QUARTUS PRIME LITE DONANIM GELİŞTİRME PLATFORMU



EK-4: İNTEL QUARTUS PRIME PROGRAMI PIN PLANLAMA ARAYÜZÜ



Şekil 7.4. Intel Quartus Prime programı pin planlama arayüzü

EK-5: SIGNAL TAP LOGIC ANALYZER ARAYÜZÜ

Signal Tap Logic Analyzer - /yedek2/Activities/Phd/Simulator/dc_motor/smltr_dcmt - [spt1.stp]*

File Edit View Project Processing Tools Window Help

Instance Manager: Acquisition in progress

Instance	Status	Enabled	LEs: 3481	Memory: 17152	Small: 0/0	Large: 0/0
auto_signaltap_0	Not running	☐	0 cells	0 bits	0 blocks	0 blocks
input_values_fltip	Waiting for trigger	☒	3481 cells	17152 bits	0 blocks	0 blocks

JTAG Chain Configuration: JTAG ready

Hardware: DE-Soc [1-5]

Device: @2: 5CSEBA6((ES)/5CSEBA6/..(0x02D0D0DD))

SOF Manager: k2/Activities/Phd/Simulator/dc_motor/smltr_dcmt2/smltr_dcmt.sof

log: Trig @ 2023/01/01 10:19:41

Type	Alias	Name	8	9	10	11	12	13	14	15	16
cho											
Vs_left			-14.75	-14.75	-14.75	-14.75	-14.75	-14.75	-14.75	-14.75	-14.75
T_left			-14.703125	-14.703125	-14.703125	-14.703125	-14.703125	-14.703125	-14.703125	-14.703125	-14.703125
I_left			-0.747329593	-0.747285604	-0.747244656	-0.747149408	-0.747099267	-0.747062385	-0.746980309	-0.746900309	-0.746820309
W_left			-254.061493	-254.061462	-254.061432	-254.061402	-254.061372	-254.061342	-254.061312	-254.061282	-254.061252
Vs_right			-14.75	-14.75	-14.75	-14.75	-14.75	-14.75	-14.75	-14.75	-14.75
T_right			-14.703125	-14.703125	-14.703125	-14.703125	-14.703125	-14.703125	-14.703125	-14.703125	-14.703125
I_right			-0.74750787	-0.747452718	-0.747407562	-0.747362406	-0.747317250	-0.747272094	-0.747226938	-0.747181782	-0.747136626
W_right			-254.055908	-254.055878	-254.055848	-254.055818	-254.055788	-254.055758	-254.055728	-254.055698	-254.055668

click to insert time bar

Hierarchy Display:

- smltr_dcmt
 - adc_spladc_spl
 - dcprnt_smltr_sptdcpm...
 - cnvrt_fx2fp_tcnvrt...
 - cnvrt_fp2fx_tcnvrt...
 - cnvrt_fx2fp_vscnvrt...
 - cnvrt_fp2fx_vscnvrt...
 - cnvrt_fx2fp_tcnvrt...
 - cnvrt_fp2fx_tcnvrt...
 - cnvrt_fx2fp_vscnvrt...
 - cnvrt_fp2fx_vscnvrt...
 - cnvrt_fx2fp_tcnvrt...
 - cnvrt_fp2fx_tcnvrt...
 - cnvrt_fx2fp_vscnvrt...
 - cnvrt_fp2fx_vscnvrt...

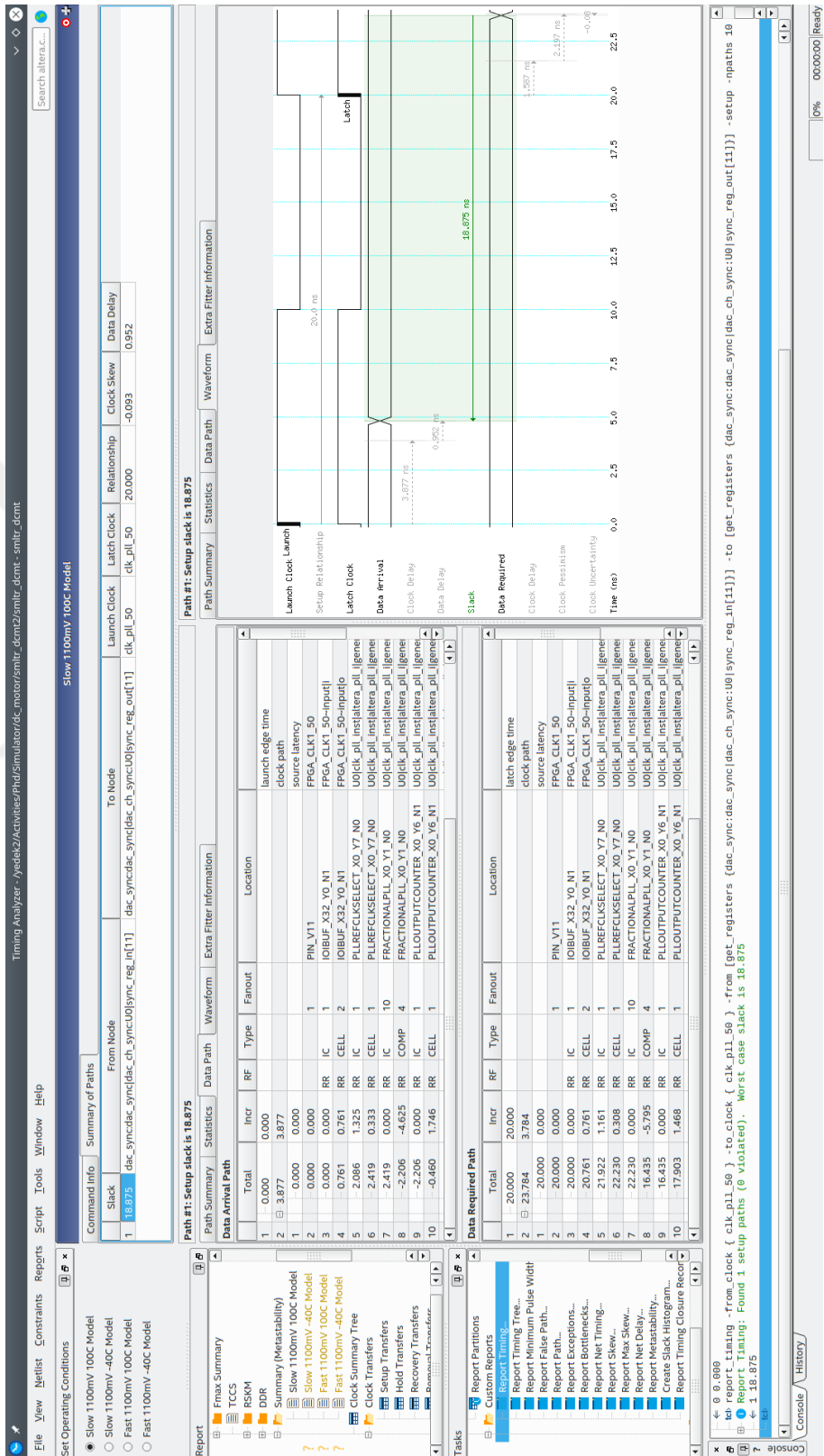
auto_signaltap_0

input_values_fltip

0% 00:00:00

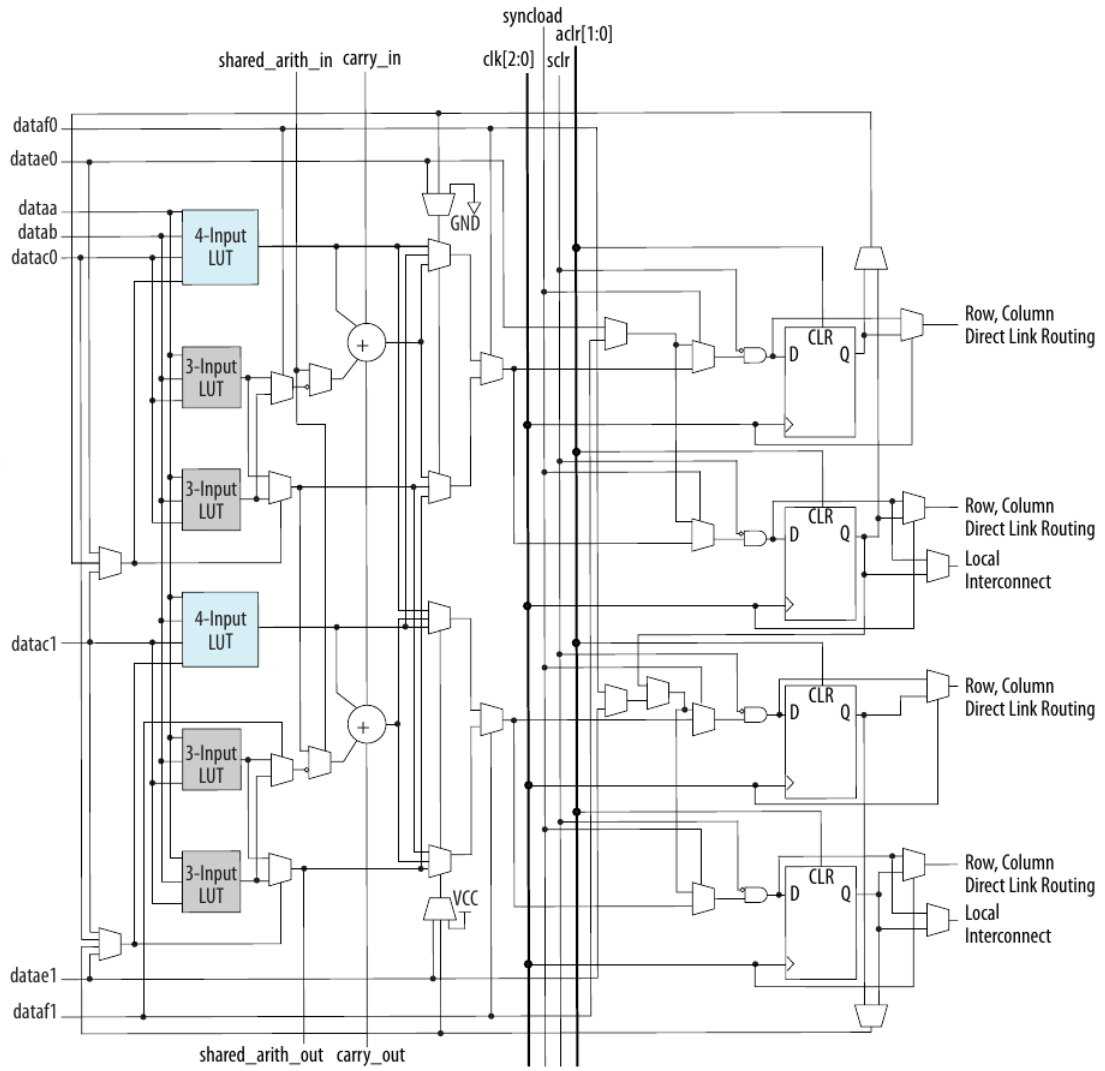
Şekil 7.5. Intel Quartus Prime programı Signal Tap Logic Analyzer arayüzü

EK-6: İNTEL QUARTUS PRIME PROGRAMI TIMING ANALYZER MODÜLÜ



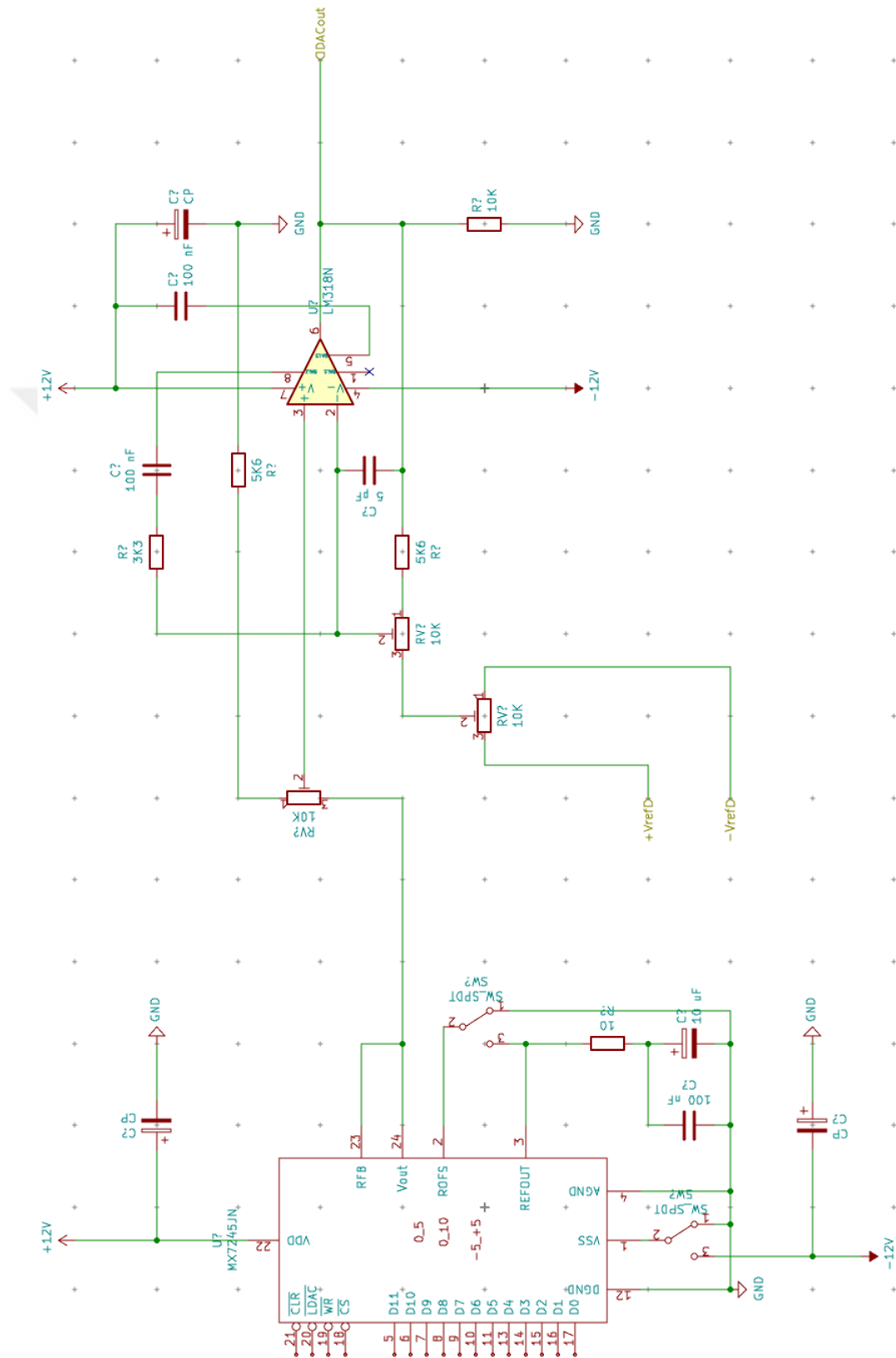
Şekil 7.6. Intel Quartus Prime programı Timing Analyzer modülü

EK-7: CYCLONE-V ENTEGRESİALM ÜNİTESİ



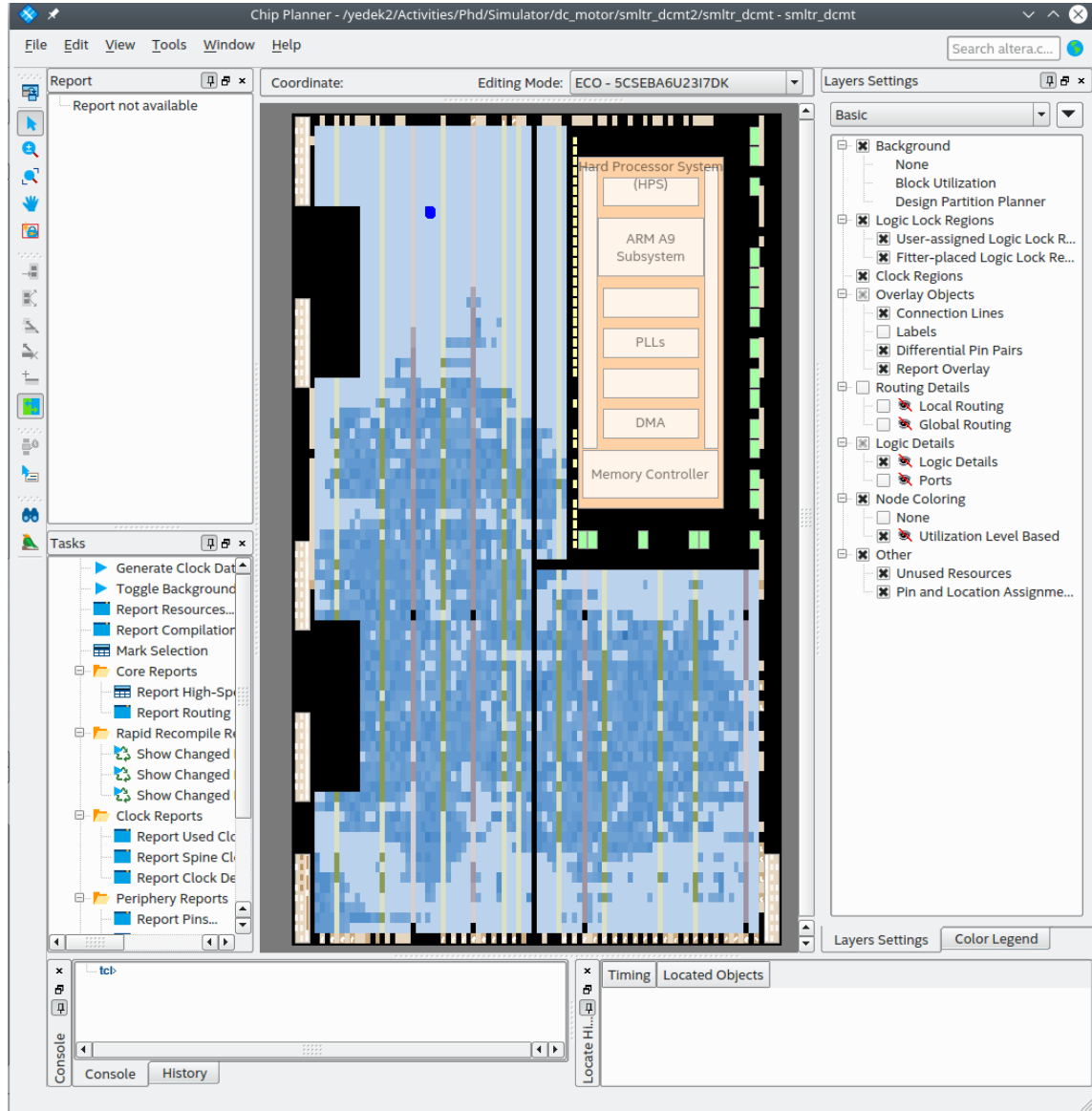
Şekil 7.7. Cyclone V entegresi ALM ünitesi detayı

EK-8: ANALOG ÇIKIŞ DEVRESİ



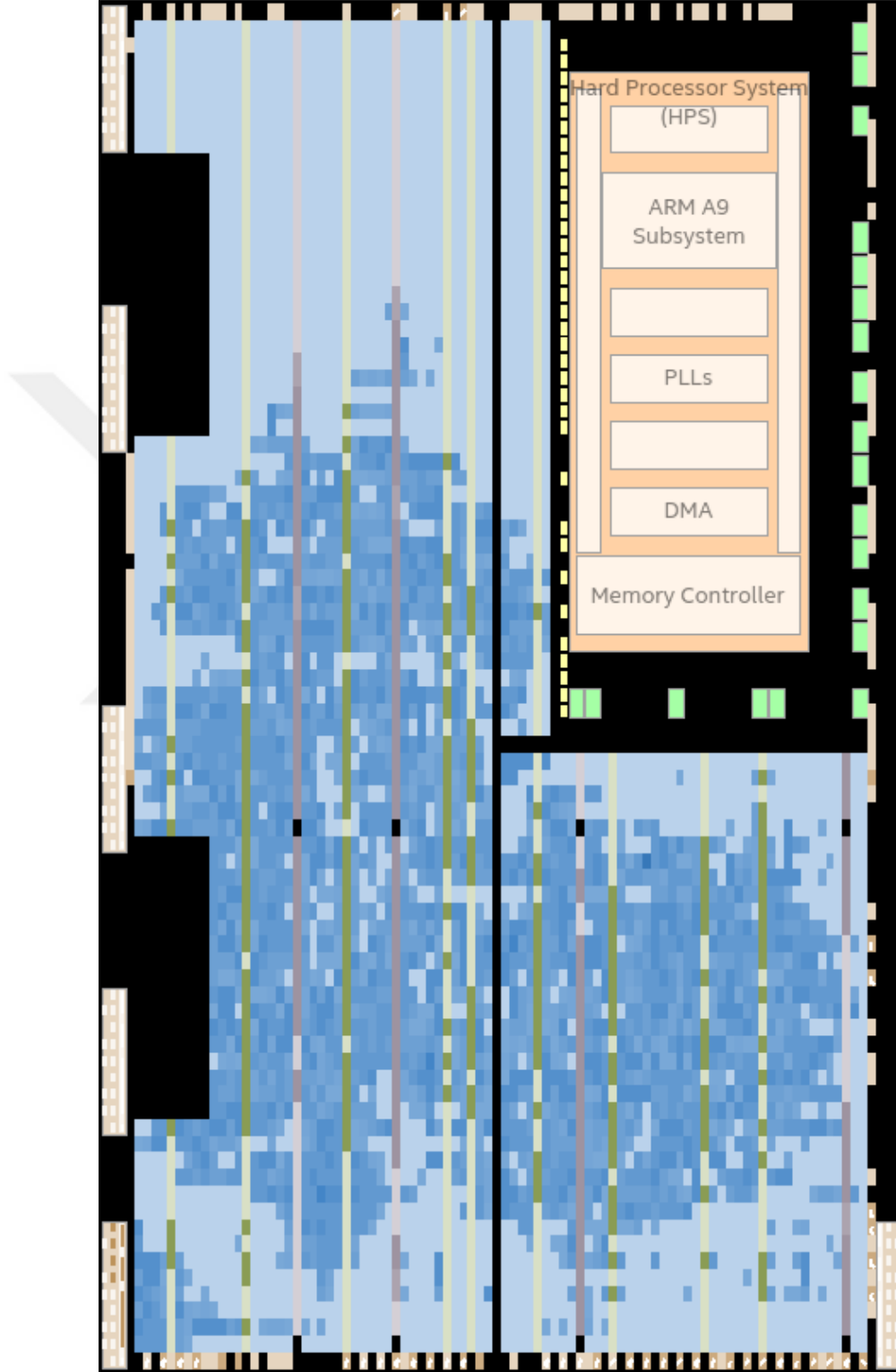
Şekil 7.8. Analog çıkış devresi

EK-9: QUARTUS PRIME YAZILIMI CHIP PLANNER MODÜLÜ

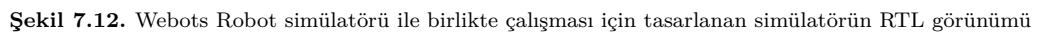


Şekil 7.9. Quartus Prime yazılımı Chip Planner modülü

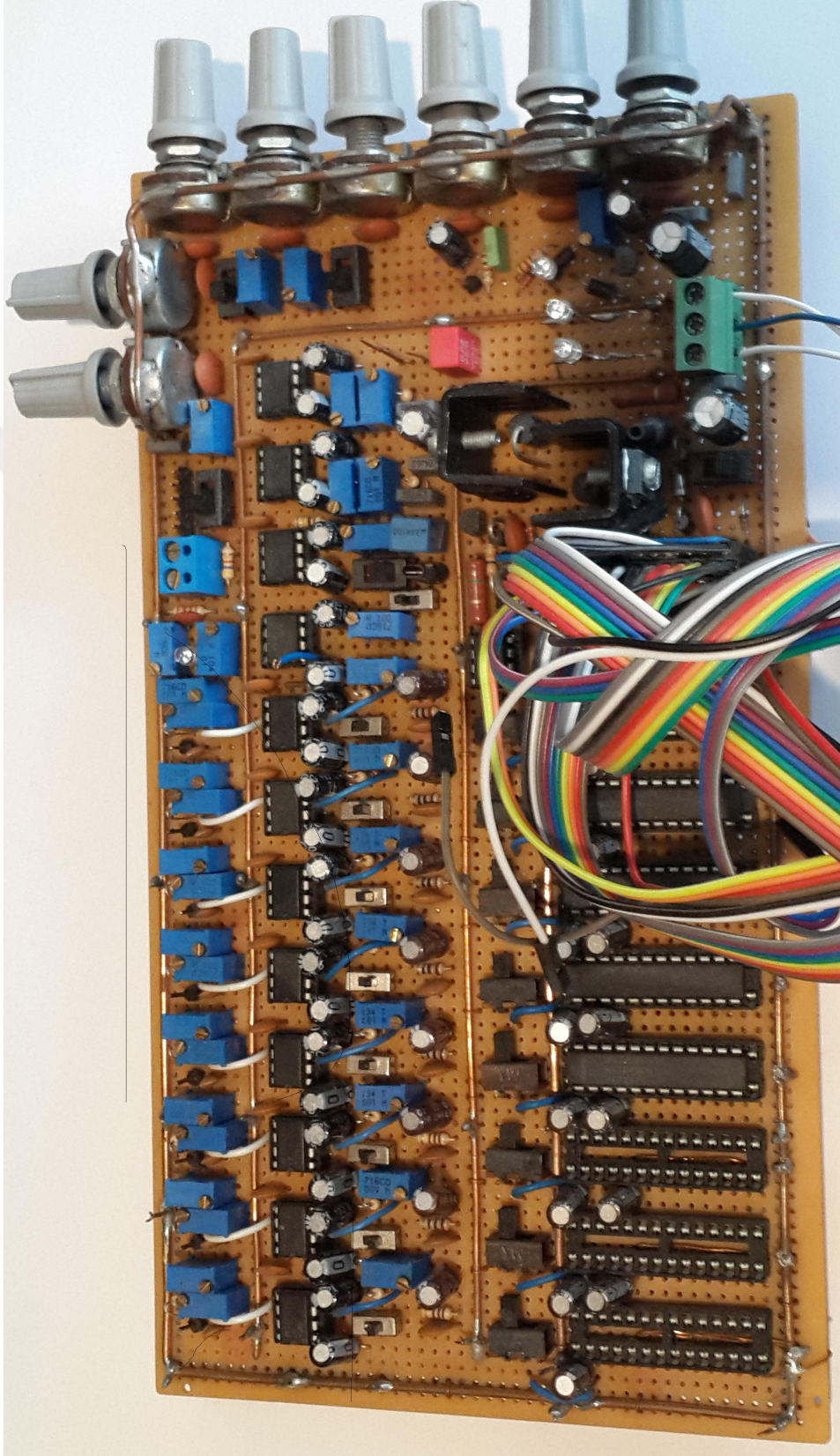
EK-10: PMDC MOTOR SİMÜLATÖRÜNÜN FPGA YONGASINDA YÜZEY YERLEŞİMİ



Şekil 7.10. PMDC Motor SİMÜLATÖRÜNÜN FPGA YONGASINDA YÜZEY Yerleşimi

[illegible]

EK-13: ANALOG GİRİŞ / ÇIKIŞ KARTI



Şekil 7.13. Analog giriş / çıkış kartı

ÖZGEÇMİŞ

Mehmet Rıza SARAÇ

[Redacted]

[Redacted]

[Redacted]

[Redacted]

[Redacted]

[Redacted]

[Redacted]

[Redacted]

[Redacted]

[Redacted]

AKADEMİK FALİYETLER

Bildiriler :

1. Sarac M. R., AYDOGMUS O. (2022). Design and Implementation of an FPGA Based Digital Twin for an Electric Motor, Online Sunum ICCES - 2022 Springer Hindistan