T.C.
BOZOK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Yüksek Lisans Tezi

FPGA TABANLI MİKROBİLGİSAYAR MİMARİSİ KULLANILARAK DC MOTOR SÜRÜCÜ TASARIMI VE UYGULAMASI

Emre ÖLMEZ

Tez Danışmanı

Yrd. Doç. Dr. Halit ÖZTEKİN

Yozgat 2012

T.C.
BOZOK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Yüksek Lisans Tezi

FPGA TABANLI MİKROBİLGİSAYAR MİMARİSİ KULLANILARAK DC MOTOR SÜRÜCÜ TASARIMI VE UYGULAMASI

Emre ÖLMEZ

Tez Danışmanı

Yrd. Doç. Dr. Halit ÖZTEKİN

Bu çalışma, Bozok Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından 2012FBE/T34 kodu ile desteklenmiştir.

Yozgat 2012

T.C.
BOZOK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEZ ONAYI

Enstitümüzün Mekatronik Mühendisliği Anabilim Dalı 70111710017 numaralı öğrencisi Emre ÖLMEZ'in hazırladığı "**FPGA Tabanlı Mikrobilgisayar Mimarisi Kullanılarak DC Motor Sürücü Tasarımı Ve Uygulaması**" başlıklı YÜKSEK LİSANS tezi ile ilgili TEZ SAVUNMA SINAVI, Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği uyarınca 21/12/2012 Cuma günü saat 14:00'te yapılmış, tezin onayına OY BİRLİĞİYLE karar verilmiştir.

Başkan : Prof. Dr. Feyzullah TEMURTAŞ

Üye : Yrd. Doç. Dr. Halit ÖZTEKİN (Danışman)

Üye : Yrd. Doç. Dr. Orhan ER

ONAY:

Bu tezin kabulü, Enstitü Yönetim Kurulu'nun ...10.../...01.../20.13.. tarih ve ...01... sayılı kararı ile onaylanmıştır.

...10.../...01.../20.13...

Enstitü Müdürü
Doç. Dr. Hidayet ÇETİN

İÇİNDEKİLER

Sayfa

FPGA TABANLI MİKROBİLGİSAYAR MİMARİSİ KULLANILARAK DC MOTOR SÜRÜCÜ TASARIMI VE UYGULAMASI

Emre ÖLMEZ

Bozok Üniversitesi
Fen Bilimleri Enstitüsü
Mekatronik Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

2012; Sayfa: 91

Tez Danışmanı: Yrd. Doç. Dr. Halit ÖZTEKİN

ÖZET

Bu çalışmada, eğitimsel amaçlı bir fırçasız DC (doğru akım) motor sürücü tasarımı yapıldı. Tasarımın kontrolör kısmı FPGA tabanlı eğitimsel amaçlı bir mikrobilgisayar olan BZK.SAU.FPGA ile gerçekleştirildi. BZK.SAU.FPGA'in kontrol yönündeki eksiklikleri tespit edilerek, çevresel birimlerden veri alıp gönderebilmek için standart paralel port şeklinde bir haberleşme arabirimi oluşturuldu. Motor için gerekli elektriksel gücü sağlayan sürücü devresi, 3 faz köprüsü şeklinde mosfetler kullanılarak tasarlandı. Motor kontrol işlemi hem BZK.SAU.FPGA mikrobilgisayarına program kodu yazılarak hem de alternatif olarak tamamen lojik seviyede tasarlanan donanımsal bir kontrolör ile gerçekleştirildi. Çalışmanın sonucunda hazırlanan tasarımın eğitimsel amaçlı kullanılabileceği ve ayrıca mikroişlemci kontrollü endüstriyel bir fırçasız DC motor sürücünün temellerini oluşturduğu görüldü.

Anahtar Kelimeler: BZK.SAU.FPGA, Eğitimsel Amaçlı Mikrobilgisayar, Fırçasız DC Motor, Fırçasız DC Motor Sürücü, FPGA Tabanlı Mikrobilgisayar, Motor Kontrol, Paralel Port, Üç Faz Köprüsü.

DC MOTOR DRIVER DESIGN AND APPLICATION USING WITH FPGA BASED MICROCOMPUTER ARCHITECTURE

Emre ÖLMEZ

Bozok University
Graduate School of Natural and Applied Sciences
Department of Mechatronic Engineering
Master of Science Thesis

2012; Page: 91

Thesis Supervisor: Assoc. Prof. Dr. Halit ÖZTEKİN

ABSTRACT

In this study a brushless DC (direct current) motor driver was designed for educational purpose. Controller part of the design was implemented with FPGA based educational purpose microcomputer BZK.SAU.FPGA. Determining the lacks of the BZK.SAU.FPGA in controlling, a communication interface as the form of standard parallel port was designed for controlling peripheral devices. The circuit supplying the necessary electrical power for the motor was designed with the mosfets in the form of three phase bridges. Motor control operations were implemented by writing program code for BZK.SAU.FPGA microcomputer and also, as an alternative, by a hardware controller designed at absolutely logic level. As the result of the study, it was seen that the prepared design can be used as educational purpose and moreover it creates the basis of a microprocessor controlled industrial brushless DC motor driver.

Keywords: BZK.SAU.FPGA, Educational Purpose Microcomputer, Brushless DC Motor, Brushless DC Motor Driver, FPGA Based Microcomputer, Motor Control, Parallel Port, Three Phase Bridge.

TEŞEKKÜR

Değerli hocalarım Prof.Dr. Feyzullah TEMURTAŞ'a, Yrd.Doç.Dr. Halit ÖZTEKİN'e ve Yrd.Doç.Dr. Orhan ER'e, ayrıca çalışmalarımıza 110E069 nolu proje kapsamında verdiği destekten dolayı TÜBİTAK'a ve 2012FBE/T34 nolu proje kapsamında verdiği destekten dolayı Bozok Üniversitesi Bilimsel Araştırma Projeleri Birimine teşekkürlerimi sunarım.

Ayrıca haklarını hiçbir zaman ödeyemeyeceğim sevgili anne ve babama, gösterdikleri anlayıştan dolayı sevgili eşime ve oğluma sonsuz teşekkür ederim.

TABLOLAR LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

KISALTMALAR LİSTESİ

DC	:	Doğru Akım (Direct Current)
FPGA	:	Sahada Programlanabilir Kapı Dizini (Field Programmable Gate Arrays)
PLD	:	Programlanabilir Lojik Cihaz (Programmable Logic Device)
PLA	:	Programlanabilir Lojik Dizi (Programmable Logic Array)
FPLA	:	Sahada Programlanabilir Lojik Dizi (Field PLA)
PAL	:	Programlanabilir Dizi Lojik (Programmable Array Logic)
GAL	:	Genel Dizi Lojik (General Array Lojik)
SPLD	:	Basit PLD (Simple PLD)
CPLD	:	Kompleks PLD (Complex PLD)
ASIC	:	Uygulamaya Özel Tümleşik Devre (Application-Specific Integrated Circuit)
CISC	:	Karmaşık Komut Seti (Complex Instruction Set Computer)
BLDC	:	Fırçasız DC (Brushless DC)
IEEE	:	Elektrik Elektronik Mühendisleri Enstitüsü (Institute of Electrical and Electronics Engineers)

1. GİRİŞ

FPGA'ler günümüzde kullanım alanları giderek artan cihazlar olarak karşımıza çıkmaktadır. Akademik amaçlar ile veya küçük bütçeli projelerde FPGA, özel amaçlı tümdevre geliştirmek isteyen tasarımcılar için son derece uygun ve ekonomik bir çözümdür. Yeniden yapılandırılabilir donanım birimleri olan FPGA'ler kullanıcının makul bir süre içerisinde tasarım yapmasına izin verip, tasarımın sonuçlarının anında görebilmesine imkan sağlarlar. FPGA'lerin bu artıları simülatif ortamda yapılan eğitimsel çalışmaların FPGA ortamında da kolaylıkla yapılabilmesine olanak sağlar. Simülatif ortamın ideal şartları yerine, FPGA'ler gerçek dünya şartları altında gözlenebilen, elle tutulabilen, çalıştırılabilen eğitimsel amaçlı tasarımlara olanak sağlar.

Fırçasız Doğru Akım (DC) motorları yüksek moment/akım ve yüksek moment/eylemsizlik oranına sahiptir. Bu motorların, sağlam yapı, yüksek verim ve yüksek güvenirlik gibi üstünlüklerinin yanı sıra sürücü maliyetlerinin düşük olması ve bakım gereksinimlerinin konvansiyonel motorlara göre daha düşük olması nedeniyle kullanım alanları giderek artmaktadır. Fırçasız DC motorlar günümüzde, endüstriyel uygulamaların çoğunda, özellikle otomotiv sektörü, uzay teknolojileri, bilgisayar teknolojileri, tıp elektroniği, askeri alanlar ve robotik uygulamalarında sıklıkla kullanılırlar.

Bu çalışmada, eğitimsel amaçlı bir Fırçasız DC Motor sürücü devresi ve kontrolörü tasarlandı. Tasarımın kontrolör kısmı FPGA tabanlı eğitimsel amaçlı bir mikrobilgisayar olan BZK.SAU.FPGA ile gerçekleştirildi. BZK.SAU.FPGA'in kontrol yönündeki eksiklikleri tespit edilerek giriş/çıkış arabirimleri oluşturularak çevresel birimlerden veri alıp gönderebilecek hale getirildi. Tasarlanan giriş/çıkış arabirimi paralel port şeklinde oluşturuldu. Mikrobilgisayardan aldığı sinyallere göre motora güç aktarımını sağlayacak olan sürücü devresi ise 3 faz inverter köprüsü şeklinde mosfetler kullanılarak tasarlandı. Sürücü devrenin tasarımında kullanılan transistör ve diğer devre elemanlarının piyasadan kolaylıkla temin edilebilen türden olmasına özen gösterildi. BZK.SAU.FPGA'in komutları kullanılarak yazılan kontrol

programıyla da motor kontrol işlemi gerçekleştirilmiş oldu. Son olarak ta mikrobilgisayara kod yazmak yerine tamamen lojik seviyede alternatif donanımsal bir kontrolör tasarımı yapıldı.

Bu tez çalışmasında temel amaç eğitimsel olduğu için, tasarımda kullanılan elemanların her birinin içyapısı incelenebilir, kullanıcı tarafından müdahale edilebilir ve geliştirilebilir olmasına özen gösterildi. Çalışmada kullanılan eğitimsel amaçlı bir mikrobilgisayar olan BZK.SAU.FPGA açık kaynak kodlu olması ve modüler yapısı sayesinde kullanıcıların çalışan bir bilgisayar mimarisine müdahale etmesine olanak sağladığı için tercih edildi. BZK.SAU.FPGA için paralel port arabiriminin tasarımı ile mikrobilgisayarın çevresel aygıtlarla haberleşme yönündeki eksikliği kapatılırken, bir bilgisayarın çevresel birimlerden nasıl veri alıp gönderdiği bu esnada bilgisayar kaynaklarının nasıl kullanıldığı gibi konularda temel prensipler uygulamalı olarak verilmeye çalışıldı. Lojik seviyede tasarlanan alternatif tasarım ile de tasarımcıların bir mikroişlemci veya mikrobilgisayara kod yazarak kontrol işlemi yapması ile donanımsal bir tasarım yaparak aynı işlemin yapılması işlemlerinin kıyaslaması sağlanmış oldu.

Bu çalışmada ana hedef eğitimsel amaç olsa da geliştirilen tasarımın aynı zamanda mikroişlemci kontrollü endüstriyel bir fırçasız DC motor sürücünün temellerini oluşturmaktadır. Bu yönüyle çalışma pratik olarak ta Fırçasız DC Motor sürme uygulamalarında kullanılabilecek bir klavuz niteliğindedir.

1.1. Literatür Taraması

Literatürde FPGA ortamında yapılmış çeşitli fırçasız DC motor kontrol uygulamaları mevcuttur. Bu uygulamaların bir kısmı aşağıda sıralanmıştır.

Anand Sathyan ve arkadaşları fırçasız DC motor uygulamalarının maliyetini düşürmek amacıyla düşük maliyetli bir dijital PWM kontrolör tasarladı. Bu kontrolörün basit yapısından dolayı düşük maliyetli uygulamaya özel entegre devre (ASIC- Application Specific Integrated Circuit) şeklinde uygulanabilme potansiyeli vardır. Bu kontrolörün deneysel doğrulaması FPGA ortamında yapılmıştır [1-2].

Bogdan Alecsa ve Alexandru Onea Xilinx firmasının düşük maliyetli Spartan-3E'sini kullanarak, FPGA ile düşük maliyetli bir fırçasız DC motor hız kontrol uygulaması gerçekleştirdi [3].

Cheng-Tsung Lin ve arkadaşları, FPGA ile 4 anahtarlı 3 faz fırçasız DC motor sürücü için pozisyon sensörsüz kontrol uygulaması gerçekleştirdi. Bu uygulamada pozisyon sensörü kullanılmaması ve anahtar sayısının azaltılması ile kontrol maliyetleri düşürülmüş oldu [4]. Bir diğer 4 anahtarlı 3 faz sürücü devre ile FPGA kullanarak sensörsüz fırçasız DC motor sürme uygulamasını M.Shanmugapriya ve Prawin Angel Michael gerçekleştirdi [5].

Ming-Fa Tsai ve arkadaşları ise Matlab/Simulink ve ModelSim uygulamalarını kullanarak fırçasız DC motor ve sürücü devresini bilgisayar ortamında modelledi. Simülasyon ortamında denenerek üretilen kod FPGA'e yüklenerek kontrol uygulaması gerçekleştirildi [6].

Bu çalışmada ise eğitimsel amaçlı bir mikrobilgisayar olan BZK.SAU.FPGA ile eğitimsel amaçlı bir fırçasız DC motor sürücü uygulaması gerçekleştirildi. Bu çalışmada kullanılan mikrobilgisayar BZK.SAU ismiyle ilk kez 2009 yılında eğitimsel amaçlı bir mikrobilgisayar simülatörü [7] olarak tasarlanmıştır. 2010 yılında ise BZK.SAU.FPGA10.0 versiyonuyla simülatif ortamdan yeniden yapılandırılabilir donanımlardan olan FPGA geliştirme ortamına taşınmıştır [8]. Taşınma işleminin ardından kullanıcın sisteme müdahil olması amacı doğrultusunda, mikro bilgisayar mimarisine modülerlik özelliği katılmıştır [9-10]. Kullanıcının programlarını kaydedip değişiklik yapabilmesi amacıyla mimariye flash bellek kontrolü de eklenmiştir [11]. Mimarinin FPGA ortamına taşınması, modülerlik kazandırılması ve flash bellek desteği ile BZK.SAU.FPGA'in eğitimsel amaçlı kullanım yönü oldukça güçlendirilmiştir.

Bu dokümanın ikinci bölümde FPGA'ler hakkında genel bilgi verilmiş, üçüncü bölümde ise ALTERA firmasına ait FPGA'leri programlamak için kullanılan QUARTUS yazılımının kullanımı anlatılmış ve basit bir uygulama örneği

gösterilmiştir. Dördüncü bölümde BZK.SAU.FPGA hakkında bilgi verilip beşinci bölümde Fırçasız DC Motorlar ve sürüş teknikleri hakkında bilgi verilmiştir. Altıncı bölümde IEEE 1284'e göre paralel portun yapısı anlatılmış yedinci ve son bölümde ise BZK.SAU.FPGA için paralel port arabiriminin ve sürücü devrenin tasarlanması ile motor sürme kontrol kodunun yazılması ve alternatif bir tasarım olarak ta tamamen lojik seviyede tasarlanmış donanımsal bir kontrol yapısı anlatılmıştır.

2. FPGA (Sahada Programlanabilir Kapı Dizileri)

Bu bölümde FPGA mimarisinin anlaşılmasına yönelik teorik bilgiler verilmiştir.

2.1. Programlanabilir Lojik

Dijital sistem dizaynında yazılım ile donanım arasında bir ayrım yapmak giderek daha zor bir hale geliyor. Donanım mühendisleri artık tasarlayacakları yeni tasarımların büyük bir kısmını VHDL veya Verilog gibi programlama dillerini kullanarak tasarlıyorlar. Bu diller CPLD ve FPGA gibi programlanabilir cihazlara yüklenecek devrenin tanımlanmasını sağlarlar.

Günümüzde programlanabilir lojik cihazların sayısının ve üreticilerinin oldukça artması ve aralarındaki şiddetli rekabet dolayısıyla cihaz fiyatlarının düşmesi bu cihazları daha ulaşılabilir hale getirmiştir. Çevresel aygıtları ve dahili işlemci çekirdeğini tek bir çip içerisinde barındırabilmesinden dolayı birçok sistem tasarımcısı günümüzde bu cihazları tercih etmektedir. Programlanabilir lojik cihazlar temel olarak üç ayrı mimari gruba ayrılabilir.

- SPLD (Basit Programlanabilir Lojik Aygıt)
- CPLD (Karmaşık Programlanabilir Lojik Aygıt)
- FPGA (Sahada Programlanabilir Kapı Dizileri)

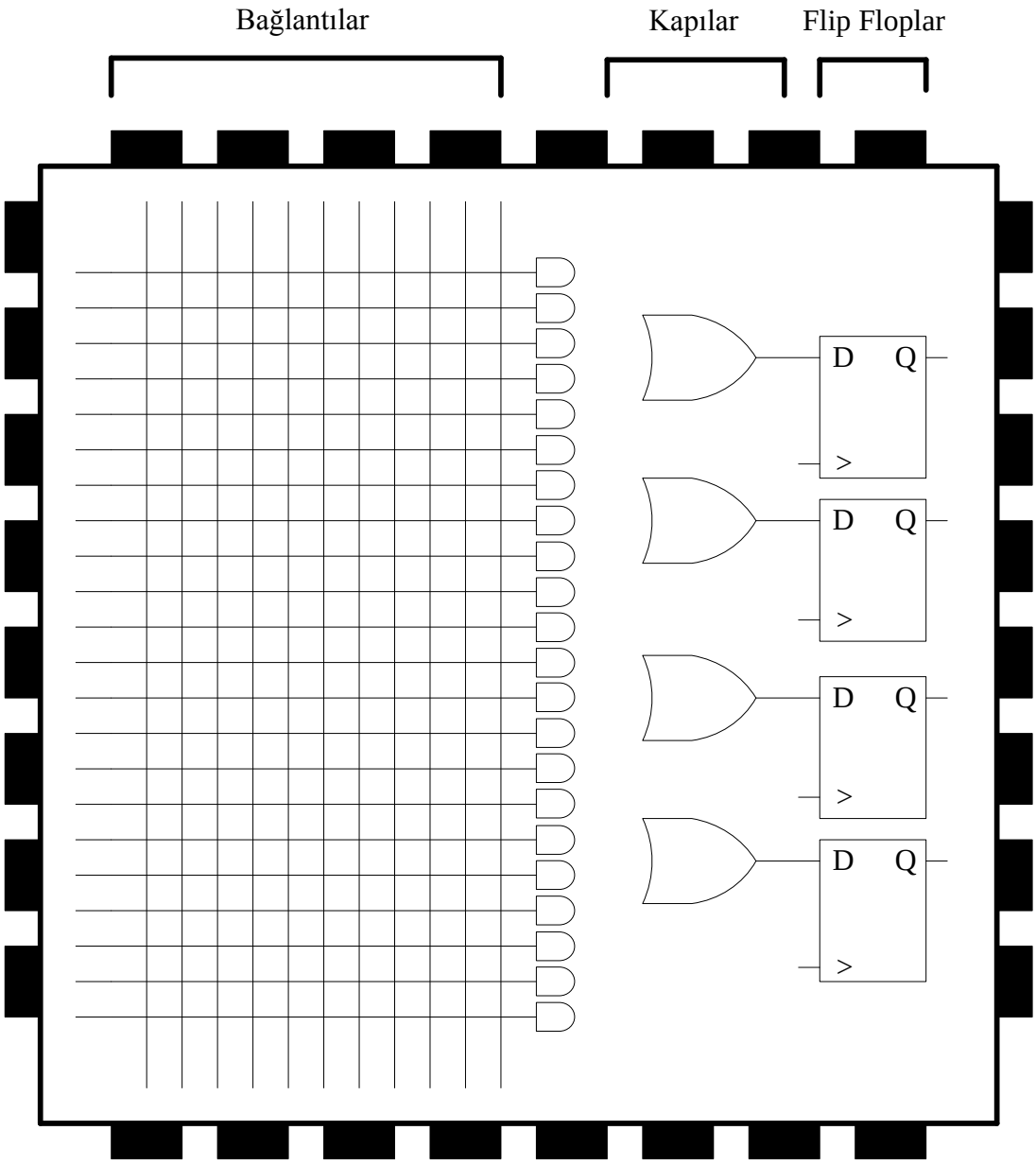
2.1.1. SPLD (Basit Programlanabilir Lojik Aygıt)

SPLD'ler programlanabilir lojik aygıtlar arasında en basit, kapasite bakımından en küçük ve en ucuz olanıdır. SPLD'ler birbirleri arasındaki bağlantılar kullanıcı tarafından konfigüre edilebilen diziler şeklinde uzanmış lojik kapılardan oluşur [7]. SPLD terimi aşağıdaki cihaz tiplerini kapsar:

- PLA (Programlanabilir Lojik Dizi): Bu cihaz programlanabilir AND ve OR düzlemlerine sahiptir.
- FPLA (Sahada Programlanabilir Lojik Dizi): PLA ile aynıdır fakat silinip tekrar programlanabilir.

- PAL (Programlanabilir Dizi Lojik): Programlanabilir AND düzlemine sabit OR düzlemine sahiptir.
- GAL (Genel Dizi Lojik): PAL ile aynı lojik özelliklere sahiptir fakat silinip tekrar programlanabilir.

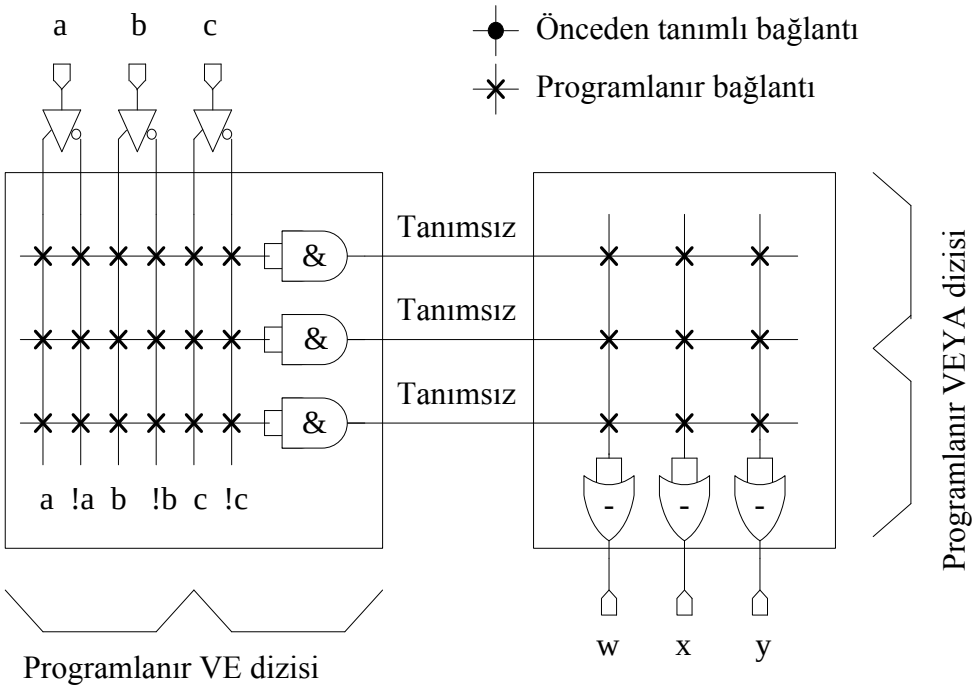
Şekil 2.1. SPLD'nin genel yapısını göstermektedir. İki hat arasındaki bağlantı linki SPLD türüne göre önceden tanımlanmış veya programlanmış olabilir.



Şekil 2.1. Genel SPLD Yapısı

2.1.1.1. PLA (Programlanabilir Lojik Dizi)

SPLD'lerin konfigürasyon bakımından en esnek olanlarıdır. Kullanıcı tarafından programlanabilir AND kapısı dizileri ve OR kapısı dizilerinden oluşurlar. PLA yapısı AND düzlemindeki bütün girişlerin çarpım terimleri şeklinde konfigüre edilebilmesine olanak sağlar. Aynı zamanda OR düzlemindeki bütün çıkışları AND düzlemindeki bütün çıkışların toplamını verebilecek şekilde konfigüre edilebilir. Bu yapı lojik fonksiyonları çarpımların toplamı şeklinde gerçekleştirmeye olanak sağlar. PLA'ler birçok çarpım teriminin birkaç çıkış tarafından kullanılması yönüyle özellikle geniş dizaynlar için kullanışlıdır. PLA'lerin dezavantajı ise yüksek üretim maliyetleri ve düşük hızlarıdır. PLA'lerin iki seviye olan programlanabilir bağlantıları düşük hızlarının sebebidir. Çünkü sinyalin programlanmış bağlantıdan geçiş hızı önceden tanımlı bağlantıya göre daha düşüktür [12].

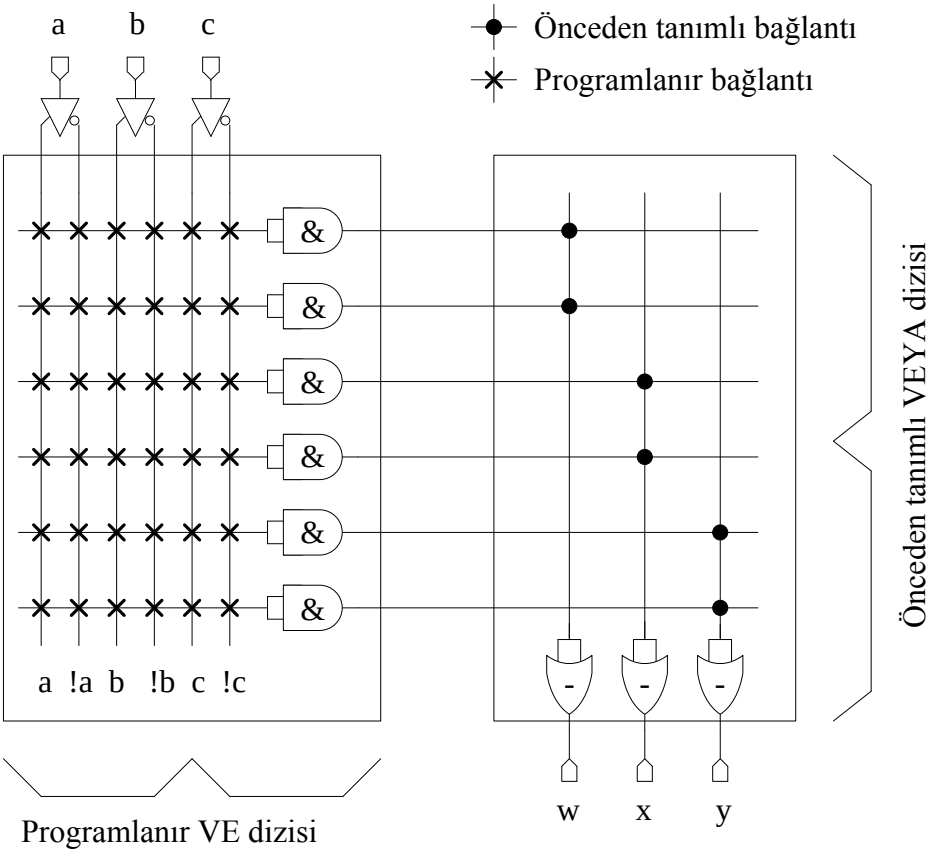


Şekil 2.2. PLA Yapısı [13]

2.1.1.2. PAL (Programlanabilir Dizi Lojik)

PAL'ler bir adet programlanabilir diziye sahip olduklarından PLA'lerden daha hızlıdırlar. Fakat bu durum OR lanacak çarpım terimlerinin sayısını kısıtlar. Bundan

dolayı farklı boyutta OR kapısı dizilerine ve farklı sayıda giriş ve çıkışlara sahip çeşitli modeller üretilmiştir. Aynı zamanda birçok PAL kaydedicili çıkışlara sahiptir. Çıkışların sıradaki clock sinyaline kadar flip - floplarda depolanabilmesi sıralı dizaynlar gerçekleştirilebilmesi için gereklidir. PAL yapısı Şekil 2.3.'de görülmektedir.

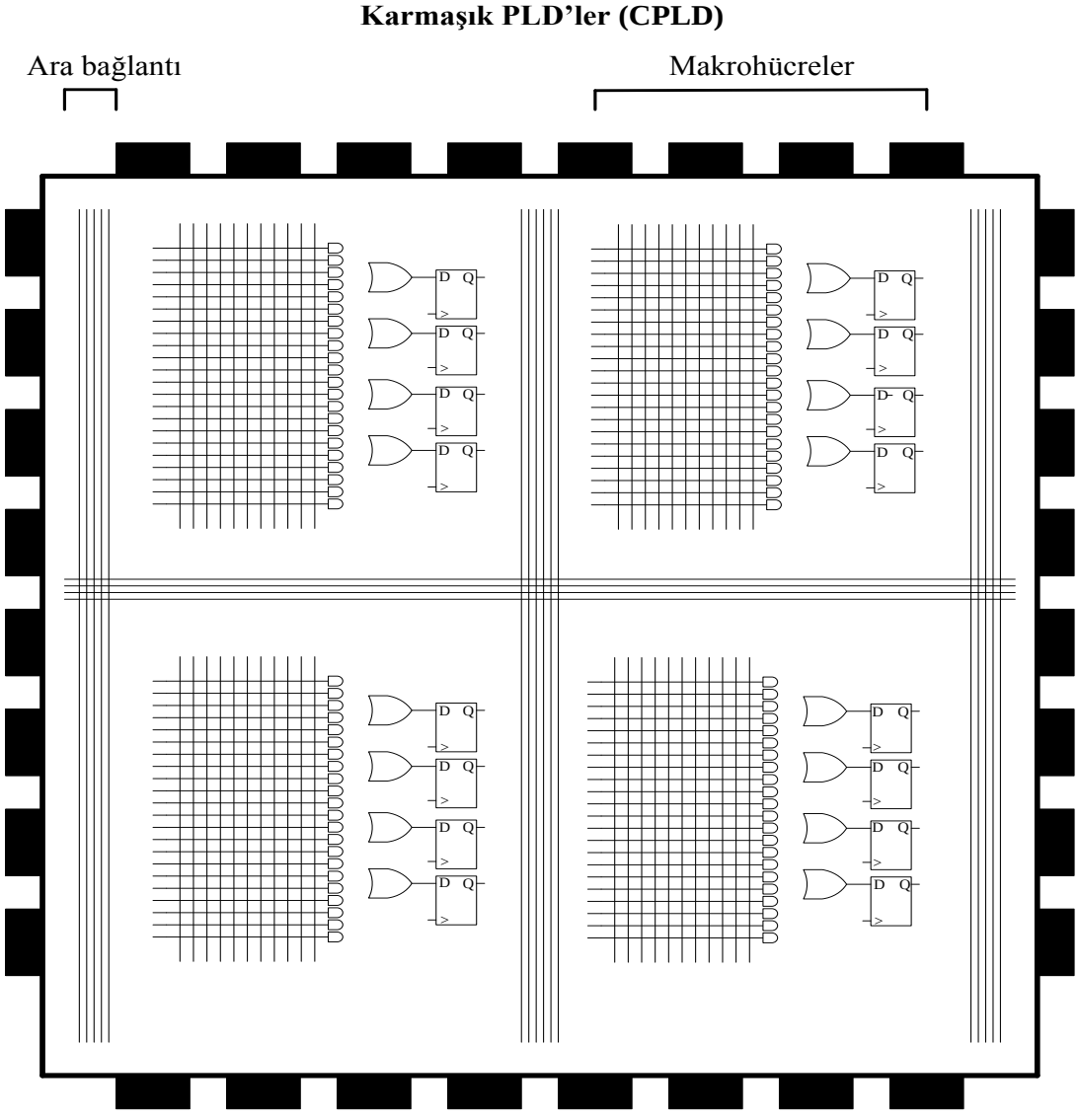


Şekil 2.3. PAL Yapısı

SPLD'ler sıklıkla adres kod çözücü olarak kullanılırlar. Yerini aldıkları 7400 serisi ürünlerine göre birtakım üstünlükleri vardır. İlk olarak tek çip olmaları dolayısıyla devre üzerinde daha az alan kaplarlar, güç tüketimleri daha azdır ve kablolama işlemi daha kolaydır. Bir diğer avantaj ise çip içerisinde değişiklik yapmak daha esnektir ve board üzerinde herhangi bir değişiklik yapmayı gerektirmez. Birçok SPLD tipi sigortalar veya kalıcı bellekleri kullanarak (EPROM, EEPROM veya Flash Memory gibi) lojik dizayn işlevselliğini saklarlar.

2.1.2. CPLD (Karmaşık Programlanabilir Lojik Aygıt)

Karmaşıklıkları ve yoğunlukları bakımından SPLD ve FPGA'ler arasında yer alırlar. SPLD'lere göre anlamlı bir şekilde daha büyük tasarımları yapmaya izin verirken FPGA'lerden daha az lojik sağlarlar. Her biri 8'den 500'e kadar makro hücre içeren birkaç lojik bloktan oluşurlar. Çoğunlukla pratik amaçlar için CPLD'ler tek çip içerisinde toplanmış çoklu SPLD'ler olarak düşünülebilir. Bir CPLD'nin tipik yapısı Şekil 2.4.'de görülmektedir.



Şekil 2.4. CPLD Yapısı

Bu lojik dizi blokların her biri bir SPLD'nin eşitidir. Tabi ki gerçek bir CPLD'de lojik blokların sayısı daha fazla veya az olabilir. Bu lojik dizi blokların her biri sıradan bir SPLD gibi makro hücreler ve bağlantılardan oluşur. Daha geniş boyuttaki CPLD'ler daha fazla lojik eşitliğe ve daha karmaşık dizaynlar yapmaya izin verirler. CPLD'lerin büyük bir kısmı karmaşık dizaynlar yapmaya izin verirler. Bunlar çarpımların toplamı şeklinde kombinasyonel lojik fonksiyonlar ve opsiyonel flip – floplar içeren makro hücrelerden oluşurlar.

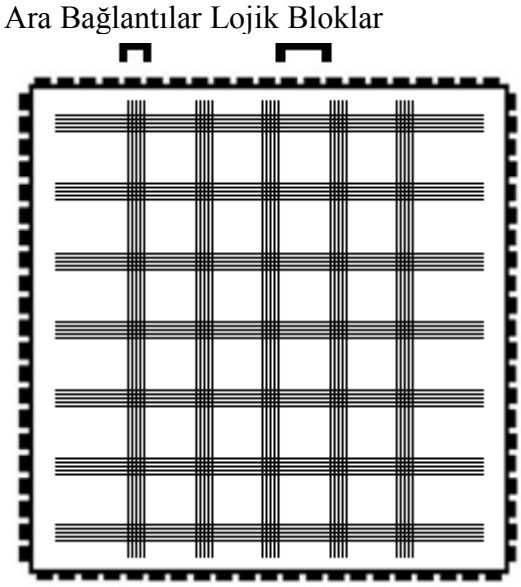
CPLD'lerin tahmin edilebilir zamanlama karakteristikleri kritik ve yüksek performans kontrol uygulamaları için onları ideal yapar. Tipik olarak CPLD'ler FPGA ve diğer programlanabilir lojik cihazlara göre daha kısa ve daha tahmin edilebilir gecikme sürelerine sahiptir. Daha ucuz olmaları ve daha düşük güç gereksinimleri ile CPLD'ler maliyet gözetilen, pil ile çalışan taşınabilir uygulamalarda sıklıkla kullanılırlar. Aynı zamanda adres kod çözme gibi basit uygulamalarda da kullanılırlar. Kapasiteleri SPLD'lerden daha yüksek olduğu için potansiyel kullanım alanları daha çeşitlidir. Hala adres kod çözme gibi basit uygulamalarda kullanılsalar da yüksek performans kontrol uygulamalarında veya kompleks sonlu durum makinelerin de daha çok kullanılırlar. Genel olarak CPLD'ler yüksek performans gerekli olduğunda FPGA'lere tercih edilirler. Bunun sebebi ise daha az esnek olan iç mimarisi, daha kısa ve tahmin edilebilir nano saniyeler mertebesinde olan gecikme sürelerine sahip olmalarıdır [12].

2.1.3. FPGA (Sahada Programlanabilir Kapı Dizileri)

FPGA'ler aralarında elektriksel programlanabilir ara bağlantı bulunan 2 boyutlu lojik blok ve flip – flop dizilerinden oluşur. Bu elektriksel programlanabilir ara bağlantılar FPGA'ler ile geleneksel entegreler arasındaki farkı oluşturur. Geleneksel entegre devrelerde lojik bloklar arası bağlantılar fabrikasyon teknolojisiyle metal ara bağlantı şeklinde yapılır.

FPGA'ler herhangi bir donanım tasarımını uygulamak için kullanılabilir. En genel kullanım alanının ise sistem prototipi hazırlamak olduğunu söyleyebiliriz. Ortaya çıkacak son üründe FPGA'in kullanılıp kullanılmayacağı ise geliştirme zamanı ve

maliyeti arasındaki denge ile nihai cihazın maliyetine bağlantıdır. Şekil 2.5. genel FPGA yapısını göstermektedir.



Şekil 2.5. Genel FPGA Yapısı

FPGA mimarisi aslında üç anahtar parçadan oluşmaktadır. Bunlar lojik bloklar, ara bağlantı ve I/O bloklarıdır. I/O bloklar yapının etrafını bir daire şeklinde saracak biçimde şekillendirilmiştir. Bu blokların her biri FPGA paketinin dışındaki I/O pinlerine input, output ve çift yönlü olarak erişimi sağlarlar. Mimaride I/O bloklarının iç kısmında ise dikdörtgen şeklinde diziler halinde lojik bloklar yer alır. Lojik bloklarla lojik bloklar ve lojik bloklarla I/O bloklar arasındaki bağlantılar ise programlanabilir ara bağlantılardır.

FPGA içerisindeki lojik blokların büyüklüğü bir PLD içerisindeki makro hücrelere göre eşit veya daha büyük, karmaşıklık olarak ise aynı veya daha karmaşıktır. Yinede bu lojik blokların büyüklüğü tüm PLD'lere göre daha büyük değildir. Örneğin CPLD'lere göre daha küçüktür. Aslında FPGA'lerin içerisindeki lojik bloklar birkaç çift lojik kapı ve bir look-up tablosu ile bir flip – floptan başka bir şey değildir. Bu extra flip – floplardan dolayı FPGA mimarisi CPLD'lere nazaran daha esnektir. Bu durum FPGA'leri pipelined ve ağır kayıt uygulamalarında daha iyi duruma getirir. Aynı zamanda FPGA'ler işlemci artı yazılım çözümlerinin kullanıldığı

uygulamalarda sıklıkla kullanılırlar. Özellikle giriş veri akımlarının hızlı bir şekilde işlenmesi gerektiği durumlarda tercih edilirler. Son olarak FPGA'ler kuzenleri olan CPLD'lere aynı alanda daha fazla kapıya sahip olması ve daha ucuz olmaları dolayısıyla tercih edilirler.

2.2. FPGA Mimarisi

FPGA'ler birçok farklı firma tarafından üretilirler. Her üretici firma mimarileri, programlama teknikleri, güç tüketimleri, lojik kapı sayıları, I/O sayıları vs. yönünden farklı özelliklere sahip çeşitli cihaz ailelerini müşterilerine sunarlar. Her ne kadar üreticiler kullanıcılara bellekler, kod çözücüler gömülü çarpıcılar gibi ekstra özellikler sunsalar da her bir model aslında programlanabilir arabağlantı, I/O bloklar ve lojik bloklar'dan oluşan genel yapının varyasyonları şeklindedir.

2.2.1. Programlama Teknolojileri

Programlanabilir cihazlar programlanabilir ve birkez programlanabilir olarak iki ana kategoriye ayrılabilir. FPGA cihazlarını programlamak için ise genel olarak dört teknoloji kullanılır. Aşağıda Tablo 2.1.'de bu teknolojiler kısaca özetlenmiştir.

Tablo 2.1. Programlama Teknolojileri

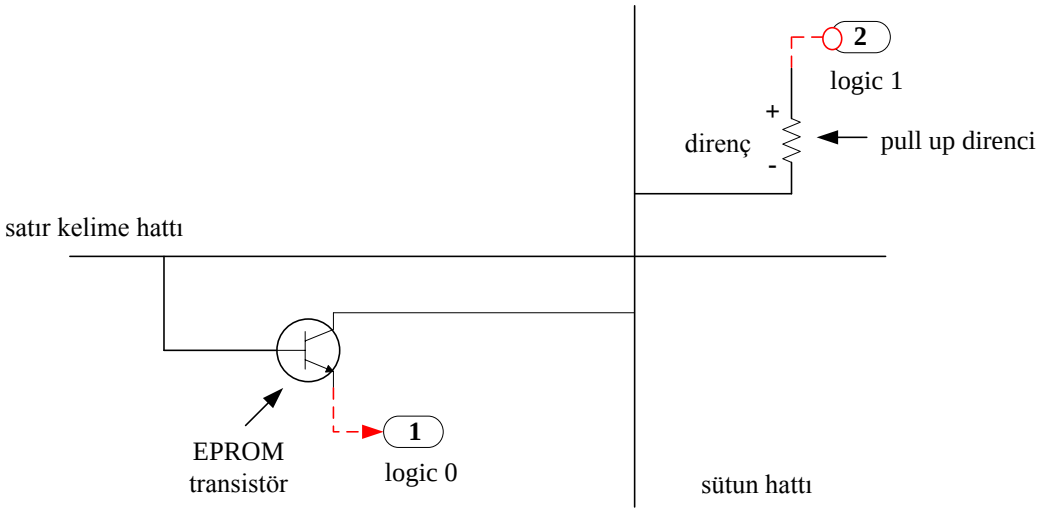
Programlama Teknolojisi	**Açıklaması**
EPROM	EPROM tabanlı belleklerin teknolojisiyle aynıdır. Kalıcıdır, elektrik kesintisinde içeriği silinmez.
EEPROM	EEPROM tabanlı belleklerin teknolojisiyle aynıdır. Kalıcıdır fakat yeniden programlanabilir.
Anti-Fuse	Konfigürasyon kalıcıdır. Bir kez programlanabilir.
SRAM	Kalıcı değildir. Elektrik kesildiğinde bellek silinir tekrar yüklenmesi gerekir.

2.2.1.1. EPROM (Erasable Programmable Read Only Memory)

EPROM lar silinip programlanabilir yalnızca okunur bellek olarak tanımlanabilirler. Standart MOS transistör yapısı üzerine temellendirilmiş olup ilave olarak oksit tabakaları arasında izole edilmiş serbest bir kapıya (floating gate) sahiptir.

Programlanmamış durumda serbest kapı (floating gate) şarj olmamış durumdadır. EPROM'u programlamak için gate ve drain uçları arasına yaklaşık 12 voltluk bir gerilim uygulanır. Bu şarj transistörün sabit olarak açık olmasını sağlar. Serbest kapı üzerindeki uyarılmış elektronlar ince oksit tabakanın diğer tarafına doğru itilirler ve negatif bir şarj oluştururlar. Bu programlama gerilimi kesilse dahi normal koşullar altında EPROM bu şarj durumunu yaklaşık 20 yıl boyunca korur.

Değişken kapı üzerinde depolanan şarj transistorün normal çalışmasını engeller. Biz bu karakteristiği Şekil 2.6.'da ki gibi bir bellek hücresini şekillendirmede kullanabiliriz.



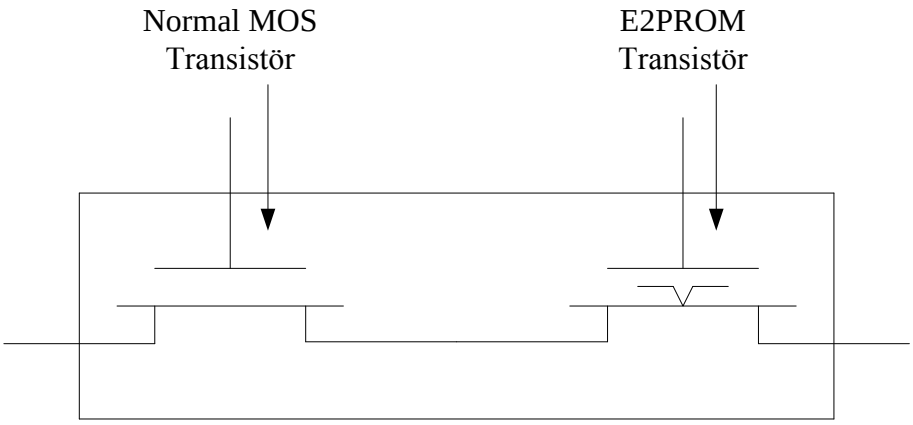
Şekil 2.6. EPROM Bellek Hücresi [14]

Programlanmamış durumda transistör normal çalışmasını sürdüreceği için satır kelime hattına uygulanan pozitif sinyalde satıra bağlı olan sütundaki programlanmamış transistörlerden lojik 0 değeri okunur. Programlanmış durumda transistör normal çalışmasını sürdüremeyeceği için okunan değer lojik 1 dir.

EPROM'un silinmesi serbest kapı hücrelerinin yükünün boşaltılmasıyla olur. Bunun için bir ultra-viole ışık kaynağına ihtiyaç vardır. EPROM'lar üzerinde içerisine UV ışığın girmesine izin veren bir QUARTZ pencereye sahiptir. Cihaz programlandıktan sonra bu pencere yanlışlıkla silinmeyi önlemek amacıyla yapışkan bir bantla kapatılır. Cihazı silmek için ilk olarak cihaz devreden ayrılır. Quartz penceresi açılır ve yüksek yoğunluklu bir UV kaynağı ile birlikte kapalı bir kutu içerisine konulur.

2.2.1.2. EEPROM (Electrically Erasable Programmable Read Only Memory)

Yapıları EPROM'larınkine benzerdir. Farkları elektriksel olarak silinebilmesidir. Şekil 2.7.'de görüldüğü gibi EPROM bellek hücresine ikinci bir transistör ilavesi ile oluşturulmuştur. Bu transistör hücreyi elektriksel olarak silmeye yarar.



Şekil 2.7. EEPROM Bellek Hücresi [14]

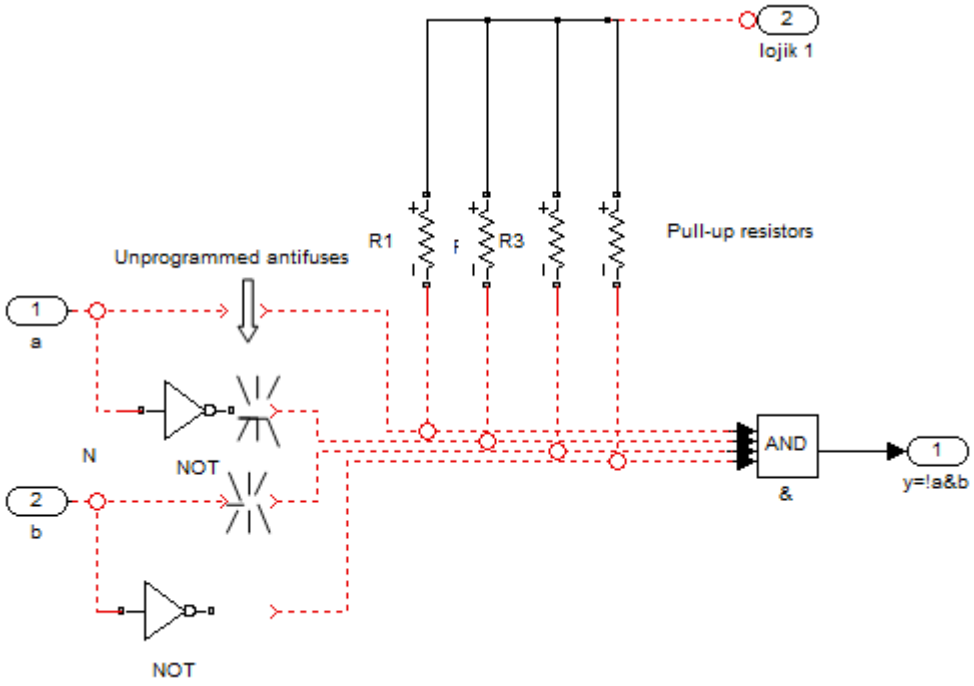
EEPROM'larda;

- Çipin tekrar yazılabilmesi için taşınmasına gerek yoktur.
- Bir kısım bilginin değiştirilmesi için tüm çipin yeniden yazılmasına gerek yoktur.
- Silebilmek için ilave özel donanımlara gerek yoktur.

2.2.1.3. Anti-Fuse (Karşıt Sigorta)

Karşıt sigorta teknolojisinde her yapılandırılabilir yolun programlanabilir bir bağlantısı vardır. Programlanmamış durumda karşıt sigorta yüksek empedans gösterir. Bu durumda karşıt sigorta iki metal parçayı birbirinden yalıtan bir izolatör

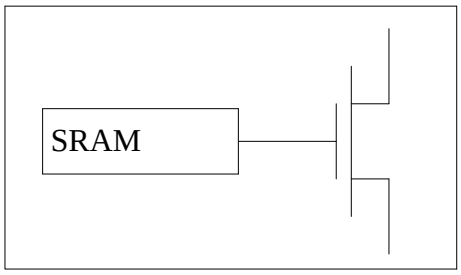
görevi görür. Yüksek voltaj ve akım darbelerinin uygulanmasıyla bu yalıtkan slikon kısım iletken polislikona dönüşür. Bu oluşturulan bağlantı ise geri alınamaz bir bağlantıdır. Yani Anti-Fuse teknolojisi bir kez programlanabilir bir teknolojidir.



Şekil 2.8. Karşıt Sigorta Teknolojisi (Anti-Fuse) [13]

2.2.1.4. SRAM

Yarı iletken RAM'lerin iki çeşidi bulunmaktadır: DRAM (Dinamik RAM) ve SRAM (Statik RAM). Dinamik RAM'de her hücre bir transistor ve kapasiteden oluşur. Bu bellek çeşidi dinamik olmasından dolayı içerisinde yük taşıyorsa belirli aralıklarla yükü tazelenmelidir. Bu sebeple DRAM teknolojisi programlanabilir lojik cihazlar için tercih edilmez. Statik RAM'de ise hücreye işlenen bir bilgi değiştirilene kadar veya güç kesilene kadar kaybolmaz. Tüm hücreler kontrol transistörünü süren SRAM depolama birimlerinden oluşur. Depolama biriminin içeriğine göre transistor açık veya kapalı konumda olacaktır. Bu teknolojinin dezavantajlarından bir tanesi 4 veya 6 transistörden oluşmalarıdır. Dezavantajlarından bir diğeri ise güç kesildiğinde tüm verilerin kaybolmasıdır. Tüm bunlara rağmen hızlı ve yeniden programlanabilir olmaları büyük bir avantaj sağlamaktadır.



Şekil 2.9. . SRAM Tabanlı Programlanabilir Hücre [14]

2.2.1.5. Programlama Teknolojileri Özeti

Programlama tekniği olarak Manyetik RAM ve Flash teknolojileri gibi yeni teknolojiler geliştirilse de bu bölümde anlatılan dört programlama teknolojisi programlanabilir lojik devreler için temel teşkil etmektedir. EPROM ve EEPROM teknolojisi SPLD leri programlamak için sıklıkla kullanılmaktadır. EEPROM bazı FPGA ler için kullanılmaktadır. Anti-Fuse ve SRAM ise FPGA ler için yaygın olarak kullanılmaktadır. Tablo 2.2.'de üç FPGA programlama tekniği bazı özellikleri bakımından kıyaslanmıştır [15].

Tablo 2.2. Programlama Teknolojilerinin Kıyaslanması

Özellik	**EEPROM**	**Anti-Fuse**	**SRAM**
Tekrar Programlanabilme	Var	Yok	Var
Tekrar Programlama Hızı	Orta		Hızlı
Kalıcılık	Var	Var	Yok
Prototipleme	İyi	Kötü	İyi
Güvenlik	İyi	İyi	Zayıf
Konfigürasyon Hücresi	Orta	Küçük	Geniş
Güç Tüketimi	Orta	Düşük	Orta

SRAM teknolojisi kullanan bazı FPGA aileleri:

- Altera Stratix II ve Cyclone II aileleri
- Atmel AT6000 ve AT40K aileleri
- Lattice LatticeEC ve LatticeECP aileleri
- Xilinx Spartan-3 ve Virtex-4 aileleri

Anti-Fuse teknolojisi kullanan bazı FPGA aileleri:

- Actel SX ve Axcelerator aileleri
- Quicklogic Eclipse II ailesi

Flash teknolojisi kullanan bazı FPGA aileleri:

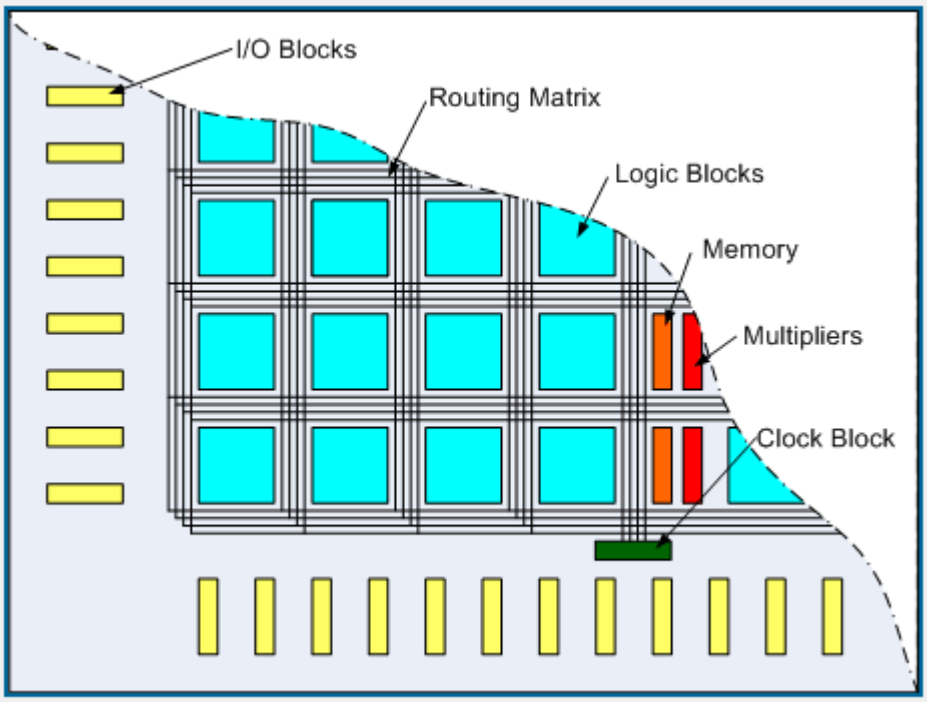
- Actel ProASIC ailesi

Hibrit Flash/SRAM teknolojisi kullanan bazı FPGA aileleri:

- LatticeXP ailesi

2.2.2. FPGA İç Yapısı

FPGA'ler farklı firmalar tarafından birçok farklı cihaz ailesi olarak üretilirler. Bu cihaz ailelerinden bazıları üreticiler arasındaki rekabetinde etkisiyle özel uygulama alanları için çok iyi tasarlanmışlardır. Aralarındaki farklılıklara rağmen dizayn mimarisi ve özellikleri bakımından benzerdirler. FPGA'ler temel olarak üç ana elemandan meydana gelmektedirler. Bunlar programlanabilir ara bağlantılar, lojik bloklar ve I/O bloklardır. Şekil 2.10. bu temel bileşenleri ve daha gelişmiş özellikleri göstermektedir. Bu bölümde temel bileşenlerden olan lojik bloklar ve I/O bloklardan kısaca bahsedilecek gelişmiş FPGA özelliklerine değinilmeyecektir.

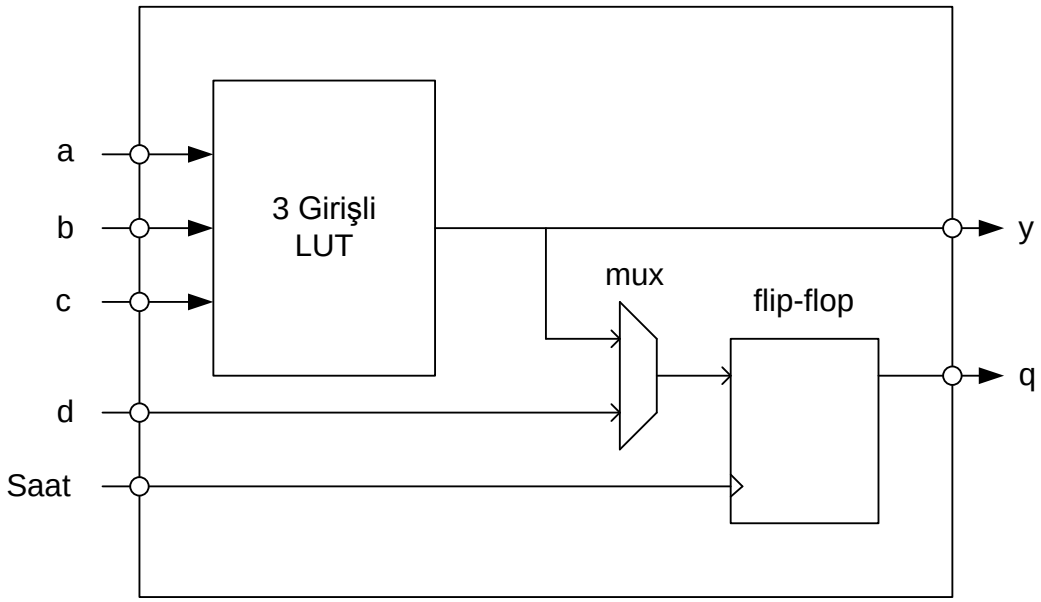


Şekil 2.10. FPGA Mimarisi [15]

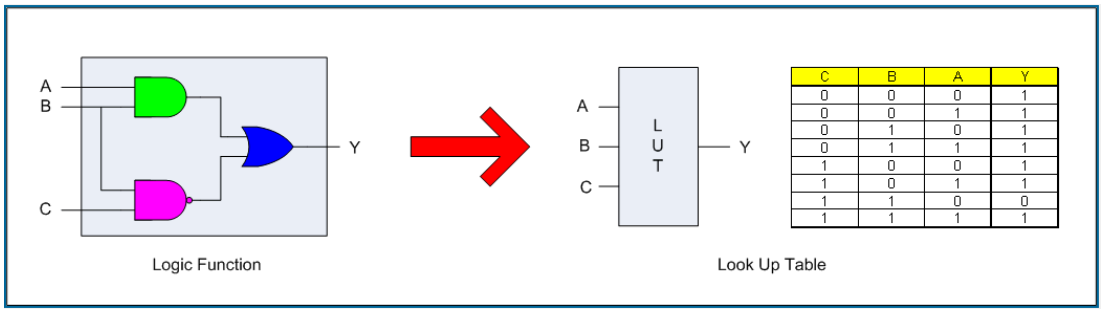
2.2.2.1. Lojik Bloklar

FPGA lojik blokları çeşitli üreticiler arasında farklı mimarilere sahip olarak üretilirler. Hatta bu farklılık aynı üreticinin farklı cihaz ailelerinde bile görülebilir. Tipik bir lojik blok içeriği bir veya daha fazla look-up tablosu, flip floplar, elde biti ve sinyal yönlendirme mux'undan oluşur. Basitleştirilmiş bir lojik blok Şekil 2.11.'de görülmektedir. Bu lojik blok 3 girişli bir look-up tablosu (LUT), bir mux ve bir flip-floptan oluşur.

Bir look-up tablosu N sayıda girişe sahip bir Boolean Fonksiyonunu icra eder. Şekil 2.12. bir boolean fonksiyonu uygulamasını lojik kapılar ve look-up tablosu şeklinde göstermiştir.

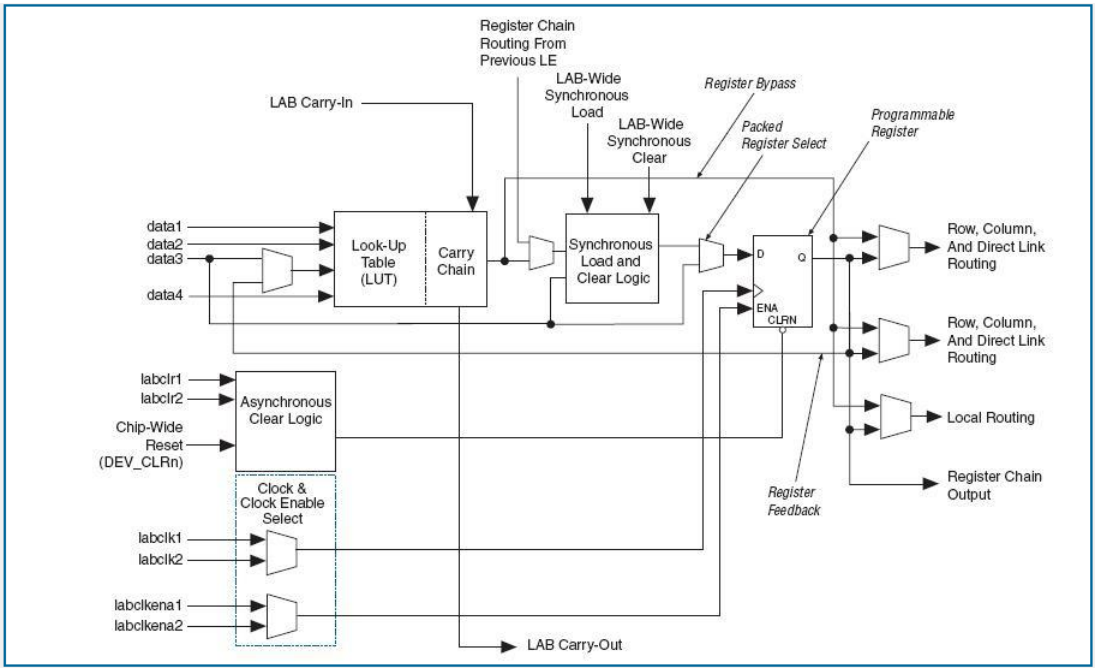


Şekil 2.11. Basitleştirilmiş Lojik Blok [15]



Şekil 2.12. Boolean Fonksiyonu Lojik ve LUT [15]

Look-Up tablosunun çıkışları doğrudan veya bir flip-flop üzerinden lojik bloğun dışarısına çıkarlar. Lojik bloklar daha büyük yapıları oluşturmak için birlikte gruplanabilirler. Şekil 2.13.'de Cyclone II ailesine ait bir lojik blok görülmektedir.



Şekil 2.13. Cyclone II Lojik Blok [15]

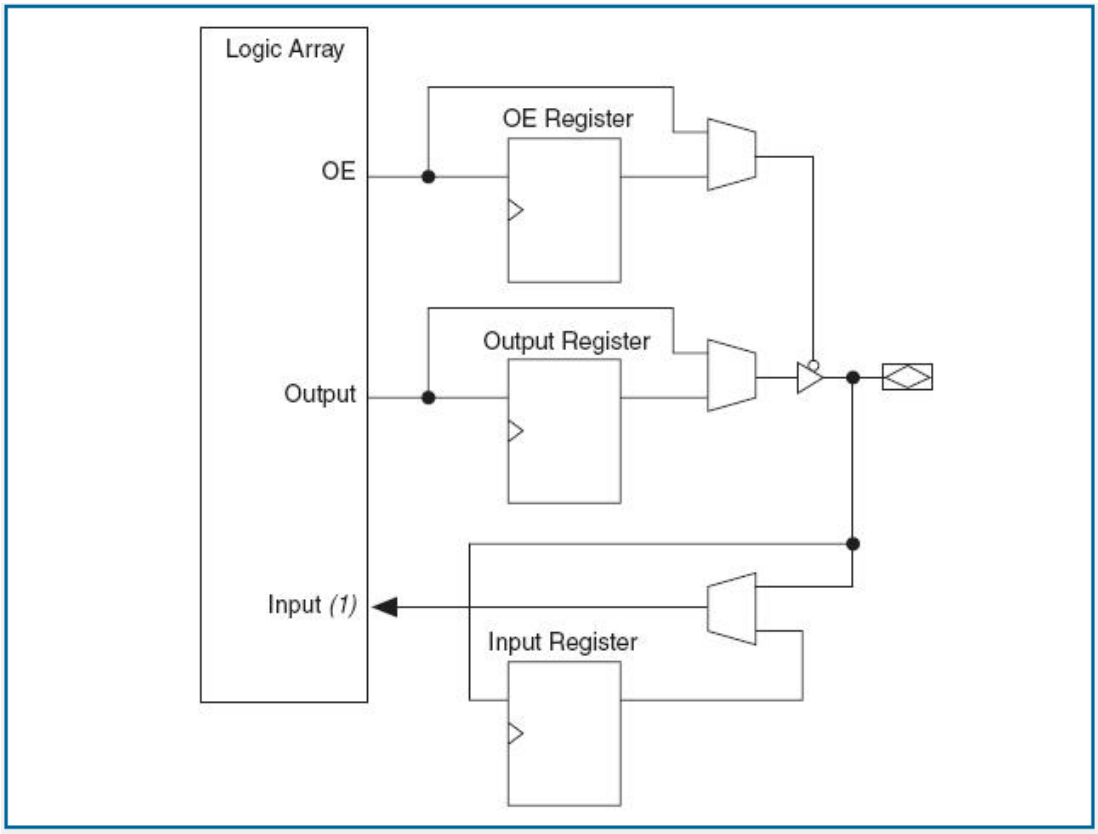
2.2.2.2. Input/Output Blokları

Cyclone II I/O blok yapısı Şekil 2.14.'de görülmektedir. I/O bloklar aşağıda sıralanan özellikleri sağlarlar.

- Diferansiyel ve tek uçlu I/O standardı
- 3.3-V, 64 ve 32 bit, 66 ve 33 Mhz PCI uyumluluğu
- JTAG ve BST sağlama
- Output gücü kontrolü
- Konfigürasyon boyunca zayıf pull-up dirençleri
- Üç konumlu tamponlar (Tri-state buffers)
- Bus-hold devre sistemi
- Kullanıcı modunda programlanabilir pull*up dirençleri
- Programlanabilir input, output gecikmeleri
- Open-drain çıkışları
- DQ ve DQS I/O pinleri

- VREF pinleri

Cyclone II I/O pinleri çift yönlü olup kullanıcı tarafından input, output veya çift yönlü olarak tanımlanıp kullanılabilir.



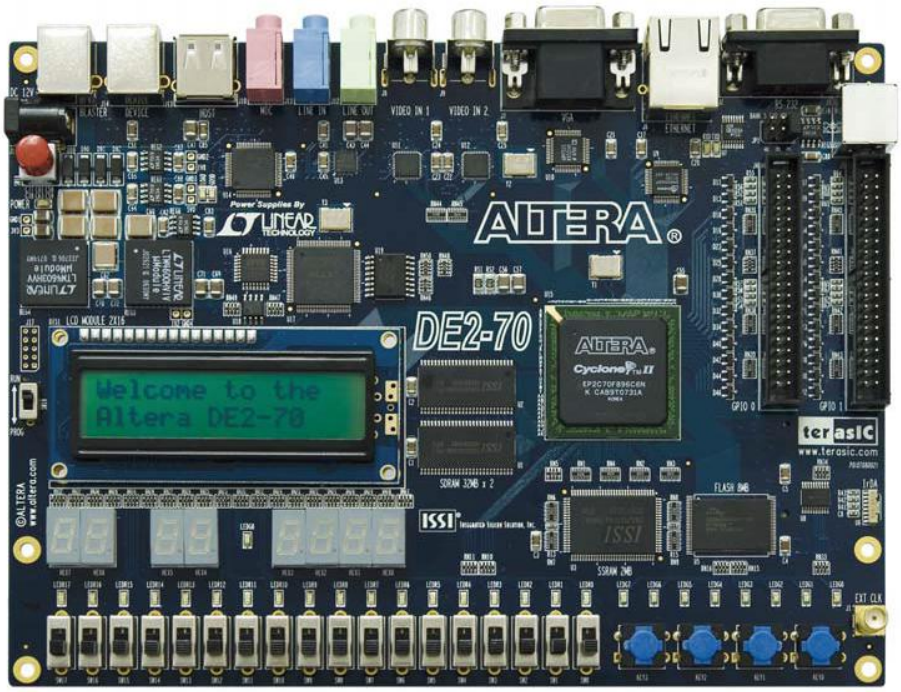
Şekil 2.14. Cyclone II I/O Blok [15]

2.3. FPGA Geliştirme Ortamı

Bu bölümde tez çalışmasında kullanılan Altera DE2-70 FPGA geliştirme ortamı hakkında kısaca bilgi verilecektir. DE2-70 board üzerinde Cyclone II serisi FPGA bulunur. DE2-70 board üzerindeki donanımlar aşağıda sıralanmıştır. Daha detaylı bilgi için bkz. Altera DE2-70 User Manual.

- Altera Cyclone® II 2C70 FPGA
- Altera Serial Configuration device - EPCS16
- USB Blaster (on board) programlama ve kullanıcı API kontrolü için
- 2-Mbyte SSRAM

- İki adet 32-Mbyte SDRAM
- 8-Mbyte Flash memory
- SD Card socket
- 4 adet pushbutton
- 18 adet toggle switch
- 18 adet kırmızı LED
- 9 adet yeşil LED
- 50-MHz ve 28.63-MHz osilator
- 24-bit CD-quality audio CODEC line-in, line-out, ve microphone-in jacklarına sahip
- VGA-out konnektörlü VGA DAC
- 2 TV Decoder (NTSC/PAL/SECAM) ve TV-in connector
- 10/100 Ethernet Controller
- USB Host/Slave Controller
- RS-232 transceiver ve 9-pin connector
- PS/2 mouse/keyboard connector
- IrDA transceiver
- 1 SMA connector
- 2 adet diyot korumalı 40-pin Expansion Header



Şekil 2.15. Altera DE2-70 Board [16]

3. QUARTUS II CAD KULLANIMI

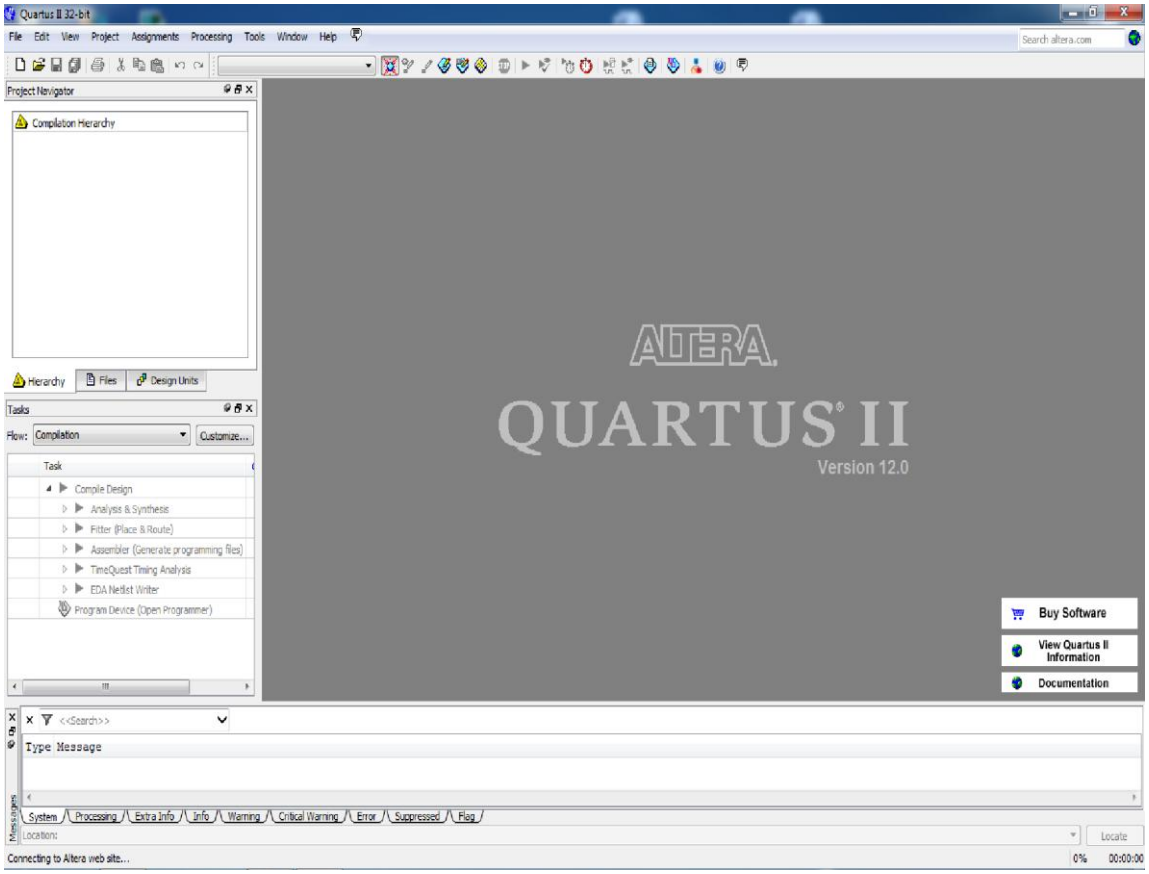
Bu bölümde Quartus II CAD programı kullanımı anlatılacaktır.

3.1. QUARTUS'a Giriş

Quartus II karmaşık bir CAD sistemidir. Birçok ticari CAD araçlarının devamlı olarak geliştirilip ve güncelleştirildiği gibi, Quartus II de bir dizi sürümden geçti. Burada Quartus II 12.0sp2 olarak bilinen sürüm kullanıldı [17]. Burada Quartus II yazılım paketinden basit bir şekilde bahsedeceğiz.

Bu bölümde Quartus II kullanılarak lojik devre tasarımını göstereceğiz. Bu dökümanda, kullanılan bilgisayarlarda Quartus II kurulu olduğu varsayılmaktadır. Quartus II kurulum aşamaları yazılımla birlikte sunulmaktadır. Quartus II programı farklı birçok bilgisayar sisteminde çalışmaktadır. Burada bilgisayarda bir Microsoft İşletim Sistemi (Windows NT, Windows 2000, veya Windows XP) olduğu varsayılmaktadır. Her ne kadar Quartus II desteklenen bilgisayar sistemlerinde benzer şekilde çalışsa da küçük farklılıklar bulunmaktadır. Microsoft Windows İşletim sistemi kullanmayan kullanıcı bu bölümden kaynaklanan bazı küçük farklılıklarla karşılaşabilir. Olası farklılıklara örnek olarak; bilgisayarın dosya sistemindeki dosyaların konumu ve yazılımın sunulduğu pencerenin doğru görüntüsüdür. Bütün bu farklılıklar önemli değildir ve kullanıcının dokümanı takip etmesini önlemez.

Bu doküman bilgisayarda mevcut işletim sisteminin nasıl kullanıldığını içermez. Kullanıcının; programları çalıştırma, farenin kullanımı, pencerelerin taşınması, pencerelerin yeniden boyutlandırılması, pencerelerin simge durumuna küçültülmesi ve ekran kaplattırılması, dizin(klasör) ve dosya oluşturulması ve bunun gibi işlemlerin nasıl yapıldığını bildiğini varsayıyoruz. İlerlemeden önce bu işlemlere aşina olmayan kullanıcının bilgisayarın işletim sisteminin nasıl kullanıldığının öğrenmesi gerekmektedir.



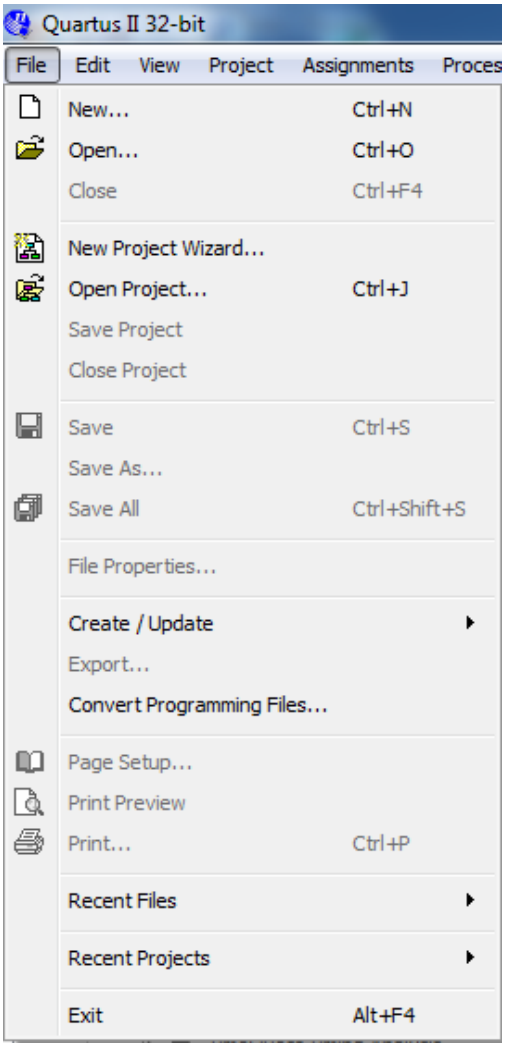
Şekil 3.1. Quartus II Ana Penceresi

3.1.1. Başlarken

Quartus II de tasarlanan lojik devresi ya da alt devrelerin her birine proje denir. Yazılım bir seferde bir projenin çalışmasına izin verir ve bütün bilgileri dosya sisteminde bir dizin içerisinde bulundurur (dosya sistemindeki bir konum için geleneksel olarak dizin ifadesini kullanacağız fakat Microsoft Windows klasör ifadesini kullanır) . Yeni bir lojiksel devre tasarımına başlarken yapılması gereken ilk işlem dosyaların kaydedilmesi için bir dizin oluşturmaktır. Quartus II kurulumunun bir parçası olarak, birkaç örnek proje qdesigns isimli bir dizin içerisine yerleştirilir. Burada tasarım dosyalarını saklamak için, ornek1 dizinini kullanacağız. Dizinin ismi ve konumu önemli olmadığı için kullanıcı her hangi bir dizin kullanabilir.

Quartus II yazılımına başlayalım. Şekil 3.1.'de gösterilen ekrana benzer bir ekran göreceksiniz. Bu ekran Quartus II de kullanıcının fare ile seçtiği bütün özelliklere erişimi sağlayan bir çok pencereden meydana gelir. Quartus II yazılımının sağladığı

çoğu komuta, başlık çubuğunun aşağısında bir dizi menü kullanılarak erişilebilir. Örneğin, Şekil 3.1. için File isimli menüde farenin sol tuşu seçilerek Şekil 3.2.' de gösterilen menü açılır. Exit öğesine farenin sol tuşu seçilerek Quartus II programından çıkılır. Genel olarak fare ile bir şeyler seçilecekse, farenin sol tuşu kullanılır. Bu yüzden hangi tuşun kullanılması gerektiğini belirtmeyeceğiz. Farenin sağ tuşunun kullanılmasını gerektiren bazı durumlarda ise bu durum açık olarak belirtilecektir. Bazı komutlar için, art ardına iki ya da daha fazla menüye erişim gerekir. Menü1|Menü2 |Item konvansiyonu kullanıcının istenilen komutu seçmesi önce Menü1 üzerinde farenin sol tuşu ile seçilmesini ve sonra Menü2 üzerinde bu menüyü seçilmesini ve sonra Item üzerinde Menü2 'nin seçilmesini gerektirir. Örneğin, File | Exit Quartus II yazılımından çıkmak için fareyi kullanır. Bir çok Quartus II komutları araç çubuğunda ilgili simgeler ile gösterilir. Mevcut araç çubuğu listesini görmek için Tools|Customize |Customize Toolbar seçeneğini seçin. Araç çubuğu açıldığında, fare ile taşınabilir ve simgeler bir araç çubuğundan diğer araç çubuğuna sürüklenebilir. Bir simge ile ilgili Quartus II komutunu görmek için, simgenin üzerine farenin kursörünü konumlandırın ve komut ismini görüntüleyen araç ipucu görüntülenir.



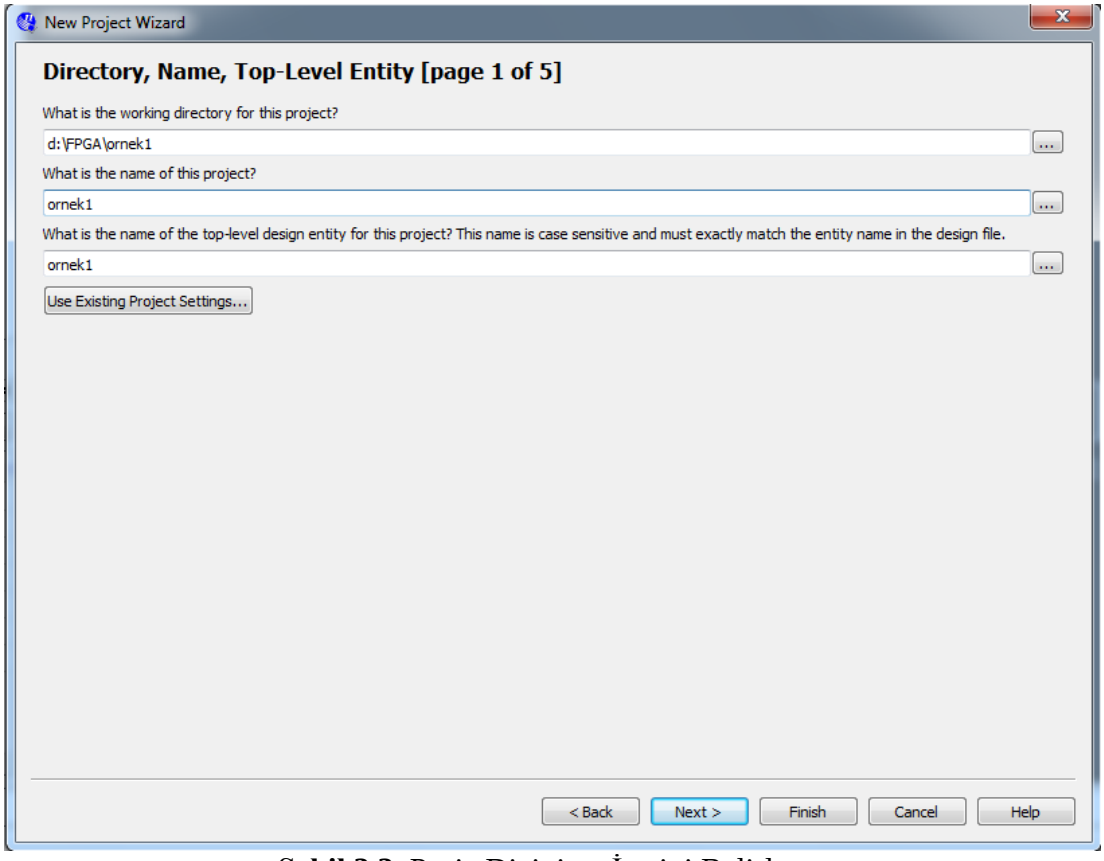
Şekil 3.2. Dosya Menüsünün Görünümü

3.1.2. Quartus II Çevrimiçi Yardım/Destek

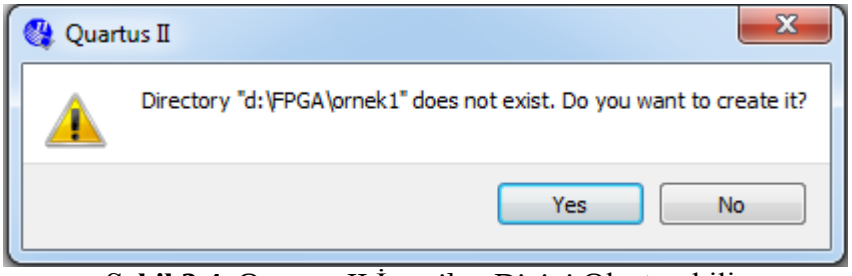
Quartus II, yazılımın kullanım sürecinde ortaya çıkabilecek sorunların çoğunu çözümleyen kapsamlı çevrim içi dökümantasyon bulundurur. Bu dökümantasyona Help penceresindeki menüden de ulaşılabilir. Sağlanan dökümantasyonun kapsamı hakkında fikir edinmek için, kullanıcının Help başlığını gözden geçirmesi faydalı olacaktır. Kullanıcı kelimlerin girilebileceği iletişim kutusunu açan Help | Search seçeneği ile Help başlığı aracılığıyla hızlı aramalar yapabilir.

3.2. Yeni Bir Proje Oluşturma

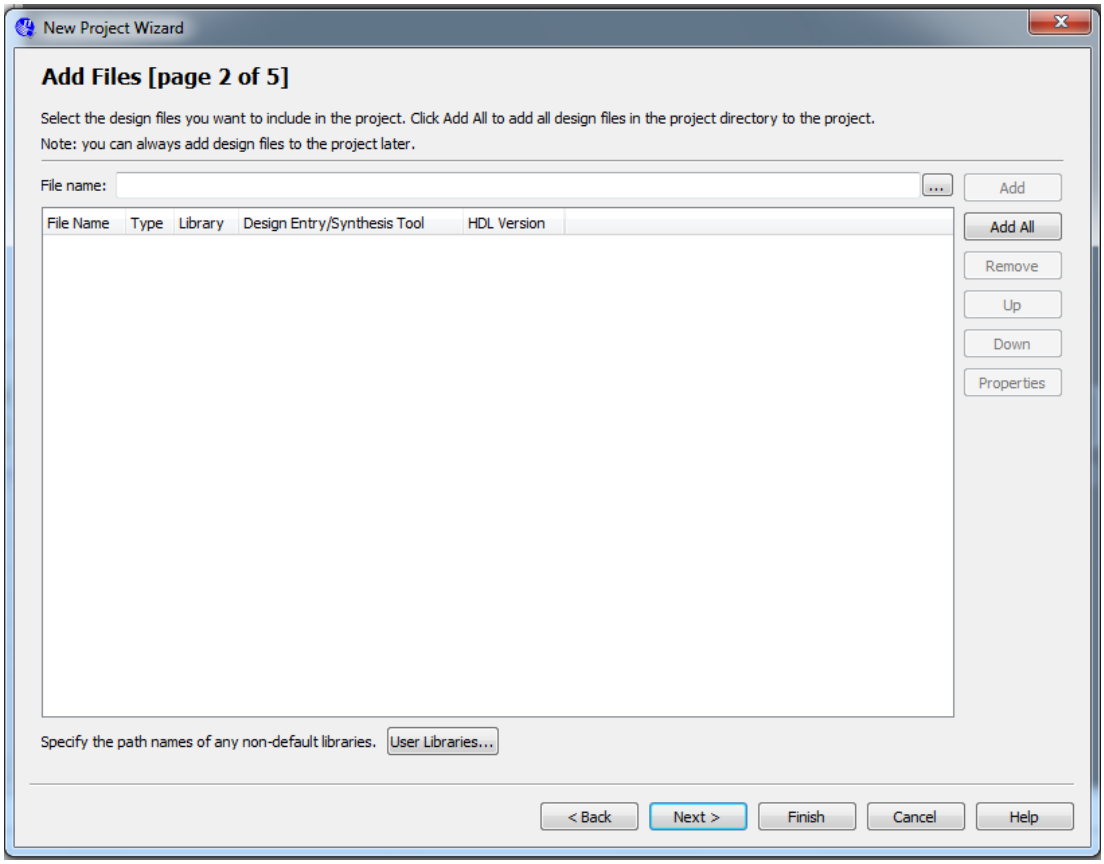
Yeni bir tasarım projesinde çalışmak için ilk olarak yeni bir design project(tasarım projesi) oluşturmalıyız. Quartus II, sihirbaz desteği ile tasarımcının işini kolaylaştırır. Sihirbaz yardımı penceresine ulaşmak için, File | New Project Wizard seçeneğini seçin. İlk olarak karşımıza çıkacak olan Introduction penceresini geçmek için next seçeneğini seçin. Bu pencerede alttaki kutucukta yer alan "Don't show me this introduction again" seçeneğini işaretlerseniz daha sonraki yeni proje oluşturma işlemlerinde bu pencere karşımıza çıkmayacaktır. 3.3. şeklinde gösterilen pencere için çalışma dizinini d:\FPGA\ornek1, proje isminide ornek1 olarak ayarlayın, Next (İleri) seçeneğini seçin. Projenin tercihe göre dizin ismi ile aynı isimde de olabilecek bir ismi olmalıdır. Quartus II, d:\FPGA\ornek1 dizini henüz oluşturulmadıysa, istenilen dizinin oluşturulup oluşturulmadığını Şekil 3.4.'de görüntülenen açılan kutu ile sorar. Şekil 3.5.'de gösterildiği gibi pencereleri göstermesi için Şekil 3.4.'de Yes(Evet) seçeneğini seçin. Bu pencerede, tasarımcı (varsa) projede kullanacağı daha önce oluşturulmuş dosyaları belirleyebilir. Daha önce oluşturulmuş kullanılacak dosya yoksa Next (İleri) seçeneğini seçin.



Şekil 3.3. Proje Dizini ve İsmini Belirleme

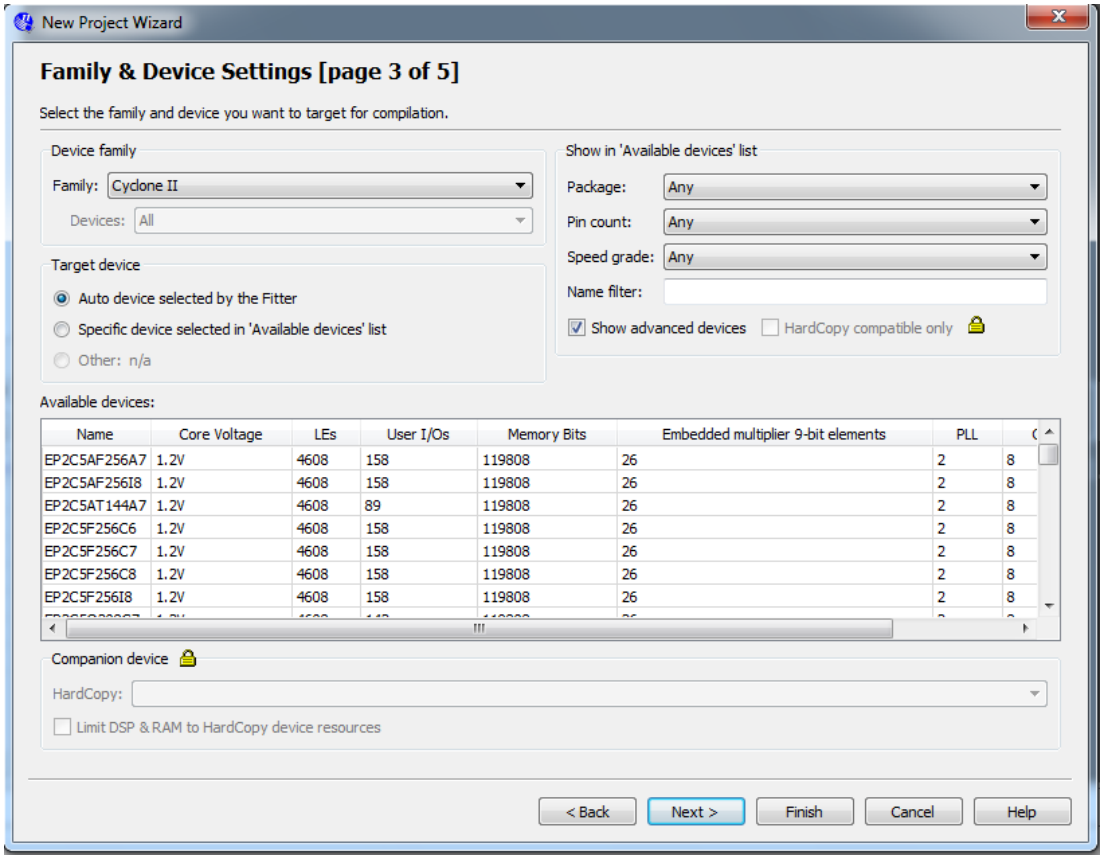


Şekil 3.4. Quartus II İstenilen Dizini Oluşturabilir



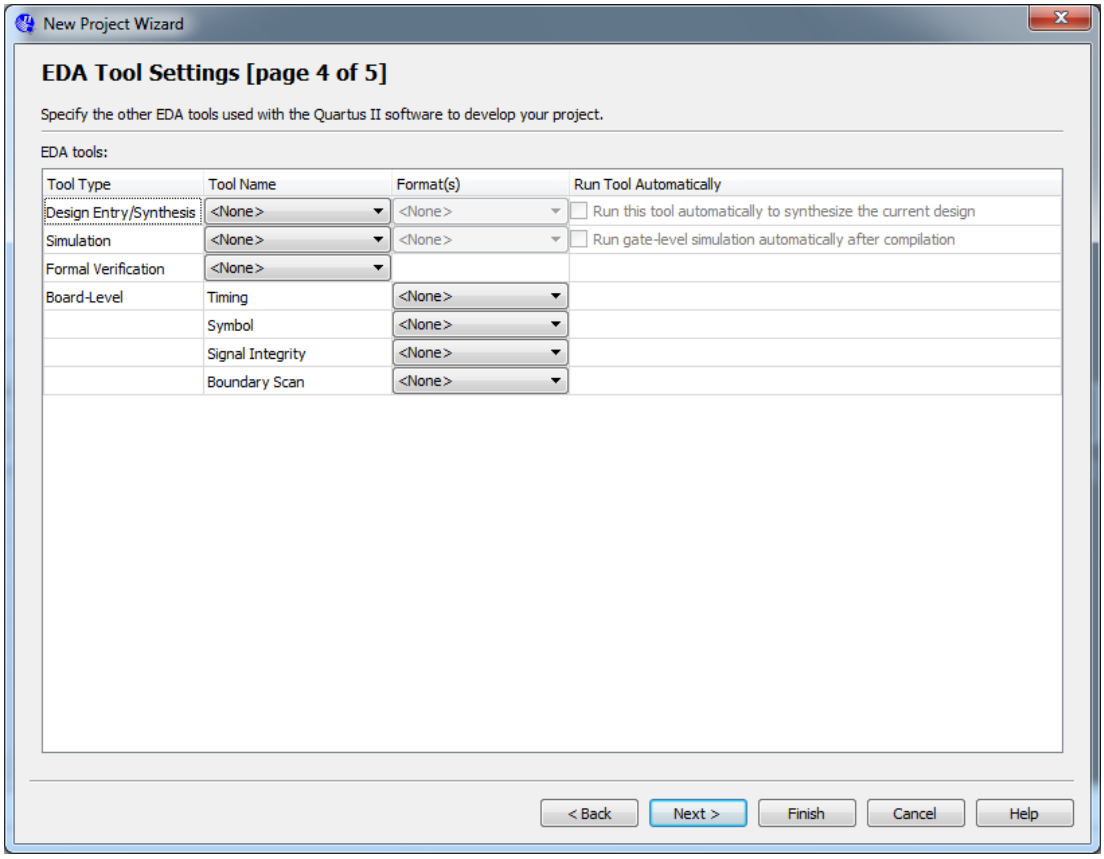
Şekil 3.5. Tasarım Dosyalarının Eklendiği Pencere

Şekil 3.6.'da tasarımcıya, tasarlanan devrenin uygulandığı aygıt türünü belirlemesi için izin verilen pencere gösterilmektedir. Burada aygıtın seçimi önemli değildir. Altera DE2-70 board(kart) üzerinde kullanılan FPGA türlerinden Cyclone II aygıt ailesini seçelim. Özel bir aygıtın seçilmesi gerekmemektedir, ' Available Devices (Mevcut Araçlar)' listesinden Fitter ile otomatik aygıt' seçeneğini seçin.



Şekil 3.6. Aygıt Ailesinin/Türünün Belirlenmesi

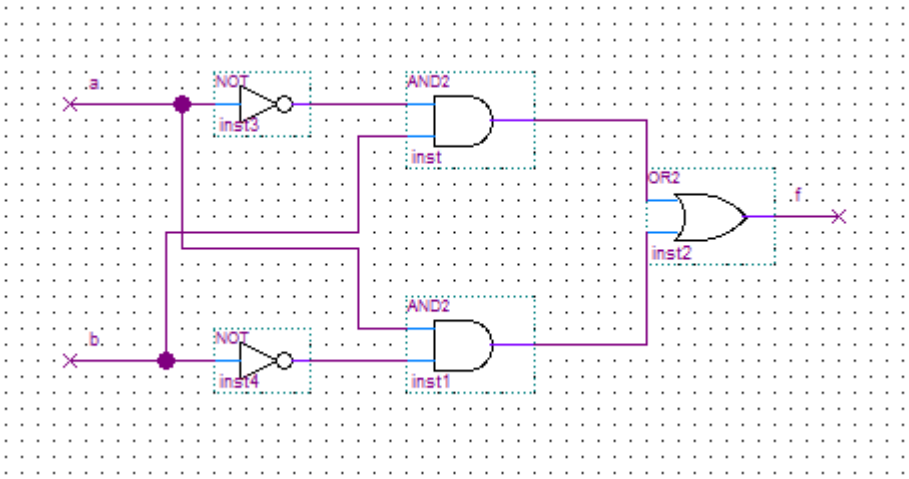
Next(İleri) seçeneğiyle Şekil 3.7.'de gösterilen pencereye ulaşacaksınız. Burada, kullanılan üçüncü parti CAD araçları(Quartus II yazılımının parçası olmayan) belirtilir. Burada , bilgisayar destekli tasarımda kullanılmak üzere geliştirilmiş yazılım programlarından bahsedilirken CAD araçları terimi kullanıldı. Elektronik Tasarım Otomasyonu olarak bilinen EDA araçları için kullanılan bir diğer yazılım terimidir. Bu terim Altera haricindeki diğer şirketlerin geliştirdiği ve piyasaya sürdüğü üçüncü parti araçlardan bahseden Quartus II iletilerinde karşınıza çıkar. Quartus II yazılımına duyduğumuz güven nedeniyle,diğer araçları tercih etmiyoruz. 3.1 şeklinde gösterilen Quartus II ekranına dönmek için ornek1 tasarımını yeni bir proje olarak kaydettikten sonra Finish(Bitir) butonunu seçin.



Şekil 3.7. Diğer EDA Araçlarının Belirlenmesi

3.3. Şematik Görüntü Kullanılarak Tasarıma Giriş

Burada, Quartus II de block düzenleyici olarak isimlendirilen şematik(devre) görüntü araçlarının kullanılması gösteriliyor. Basit bir örnekle ifade edilecek olursak, f=a'b+ab' lojiksel fonksiyonu için devre çizelim. F için devre şeması Şekil 3.8.'de türetilmiştir. f için doğruluk tablosu, Tablo 3.1.'de gösterildiği gibidir.



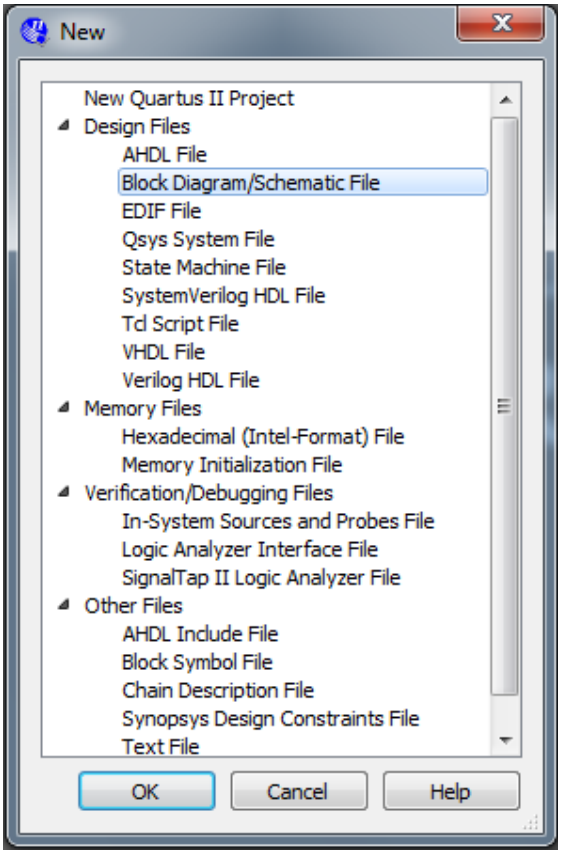
Şekil 3.8. Lojik Fonksiyon

Tablo 3.1. Lojik Fonksiyonun Doğruluk Tablosu

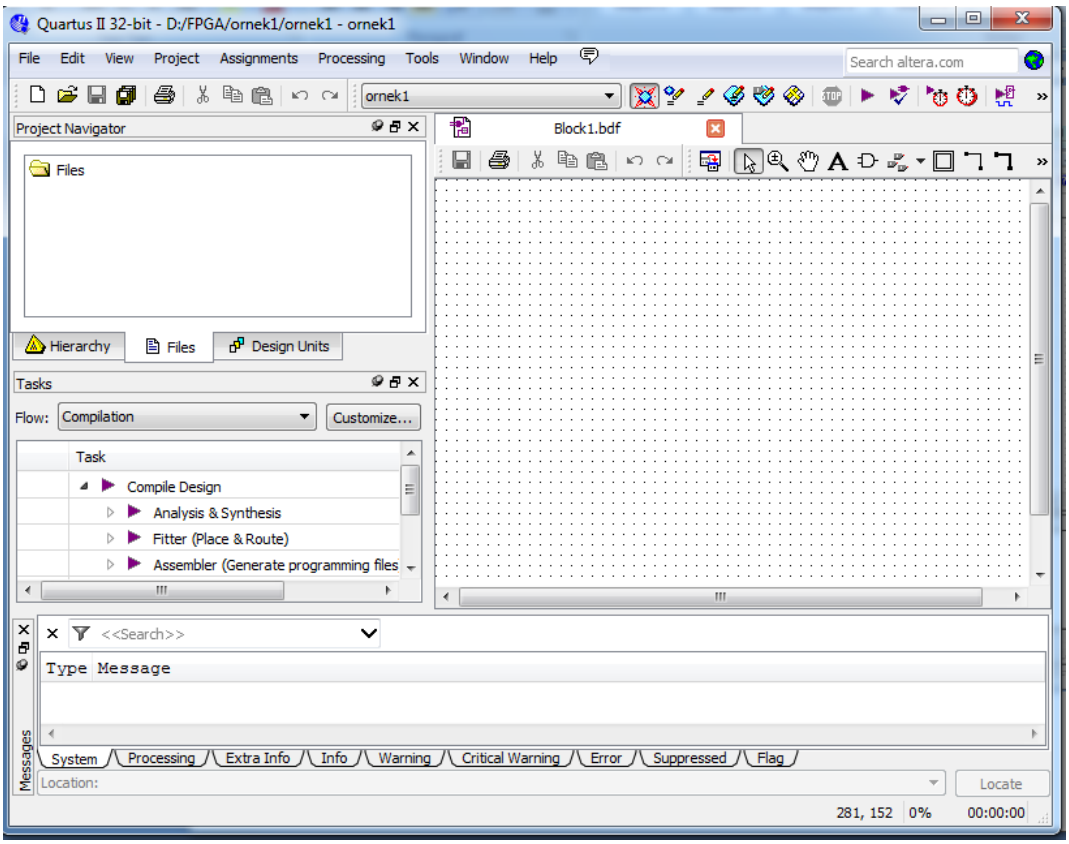
a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

3.3.1. Blok Düzenleyici Kullanımı

İlk aşama devre çizimidir. Quartus II ekranında File/New seçeneğini seçin. Şekil 3.9.'da gösterildiği gibi tasarımcıya oluşturacağı dosya türünü seçmesine izin veren bir pencereyle karşılaşacaksınız. Olası dosya çeşitleri, devreleri, Verilog kodlamalarını ve VHDL ve AHDL(Alteraya özgü HDL) gibi donanım tanımlayıcı dil dosyalarını içerir. EDIF (Electronic Design Interface Format - Elektronik Tasarım Arayüz Formatı) olarak isimlendirilen standart formatta devreyi gösteren dosyanın oluşturulması için üçüncü parti sentez aracı kullanılması da mümkündür. EDIF standardı EDA araçları arasında bilgi değişimi için uygun düzenek sağlar. Bu bölümde şematik giriş yaklaşımını örneklendirmek için Block Diagram/Schematic File seçeneğini seçin. Bu seçim, Şekil 3.10.'un sağında gösterildiği gibi block editör ekranını açar. Burada devre çizimi ile, istenilen blok diyagramı dosyası hazırlanır.



Şekil 3.9. Tasarım Dosyasının Türünün Seçilmesi

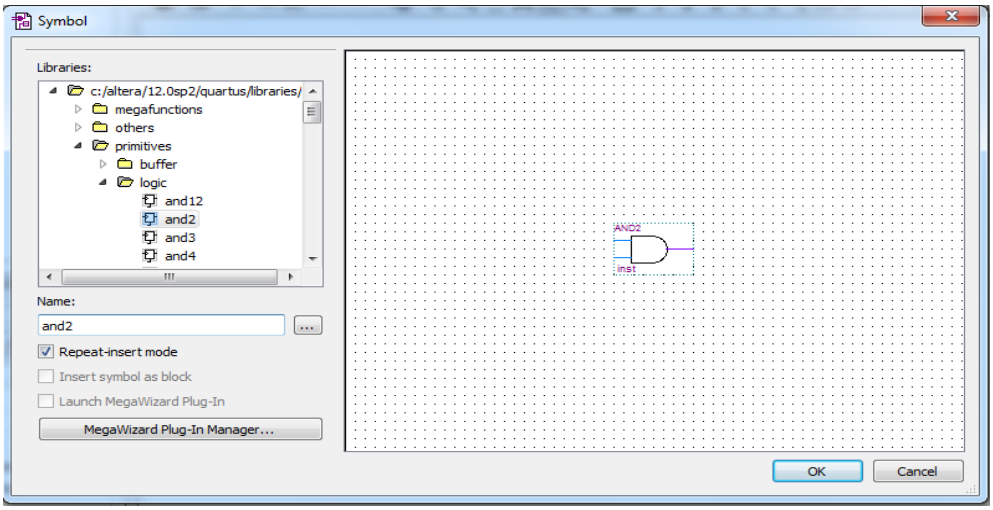


Şekil 3.10. Blok Editör Düzenleyici

3.3.1.1. Kullanılan Lojik Kapısı Sembolleri

Block editörü devreye dışarıdan eklenebilecek devre elemanlarını içeren çeşitli kütüphaneler bulundurur. Örneğimiz için, temel lojik kapılarını içeren primitives olarak isimlendirilen bir kütüphaneyi kullanacağız. Bu kütüphaneye Block Editörü penceresi içinde Şekil 3.11.'de gösterilen pencerenin açılması için boş alanda çift tıklanılarak erişilebilir (Diğer yöntem ise araç çubuğu üzerinde AND kapı sembolü seçilerek bu pencerenin açılmasıdır). Şekilde görüldüğü gibi, Quartus II ile gelen Libraries etiketi çeşitli kütüphaneleri listeler. Listeyi genişletmek için, c:/altera/12.0sp2/quartus/libraries yanındaki küçük + sembolünü seçin, sonra primitives yanındaki küçük + sembolünü seçin, ve son olarak logic yanındaki küçük + sembolünü seçin. Devreye eklemek için and2 sembolüne çift tıklayın (alternatif olarak and2 sembolünü seçip daha sonra OK iletisini onaylayabilirsiniz). Block Editör penceresinde iki girişli AND kapısı oluşur. Fare yardımı ile sembolü devrede bulunacağı yere taşıyın ve yerleştirin.

Fare yardımıyla devrede istenen semboller seçilebilir. Farenin işlecini devrede bulunan AND kapısının üzerine konumlandırın ve fare yardımıyla seçin. Sembol vurgulu renktedir. Sembolü taşımak için, farenin tuşuna basılı tutarak seçin ve sembolü taşımak istediğiniz yere fareyi sürükleyin. Grafik sembollerini daha kolay konumlandırmak için, Block Editör çalışma alanında klavuz ızgarası(rehber gridler) View | Show Guidelines seçimi ile görüntülenebilir.

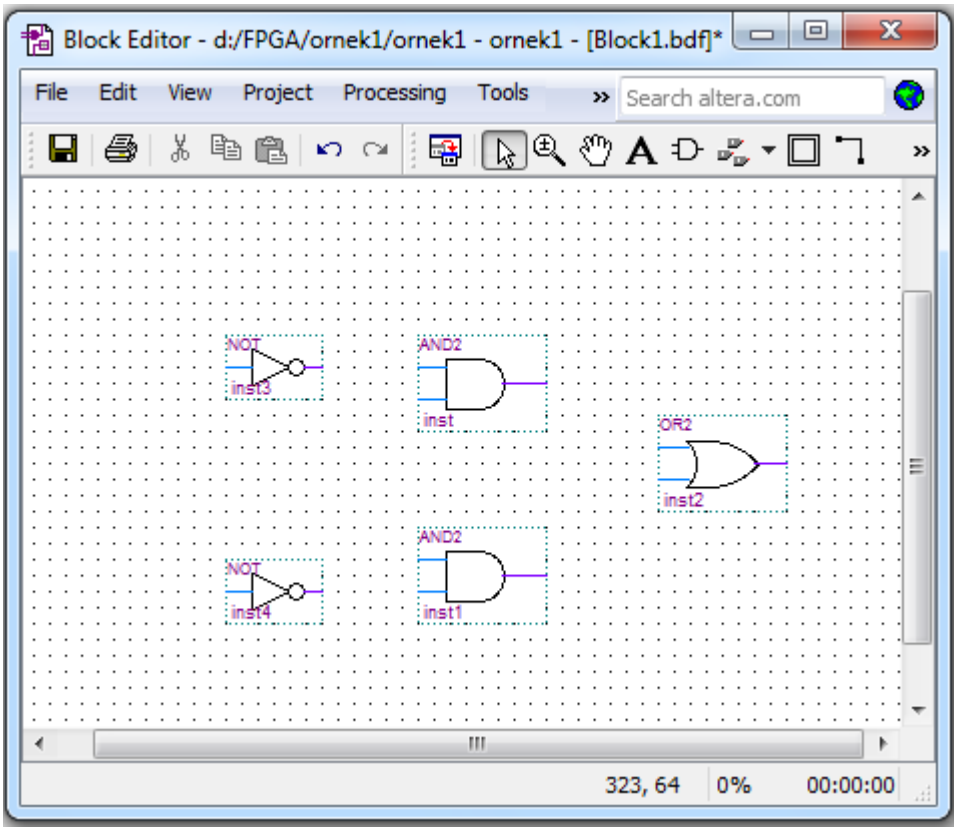


Şekil 3.11. Lojik Sembollerin Seçilmesi

f lojik foksiyonu ikinci bir iki girişli AND kapısı, bir iki girişli OR kapısı, ve bir NOT kapısına gereksinim duyar. Bu kapıları devreye eklemek için aşağıda belirtilen aşamaları uygulayın.

Fare işaretçisini eklenilen AND kapı sembolünün üzerine konumlandırın. Klavyeden kontrol (Ctrl) tuşuna basılı tutarak, AND kapı sembolünü fare yardımıyla seçin ve sürükleyin. Block Editör AND kapı sembolünün ikinci bir örneğini otomatik olarak ekler. Devre elemanının bir kopyasını oluşturan bu işlem devrede aynı elemandan birden fazla kullanıcağı zaman yaygın olarak kullanılır. Diğer alternatif yaklaşım ise, yukarıda açıklandığı gibi her bir örnek sembolü primitives kütüphanesi yardımıyla eklemektir.

OR kapısı sembolünü eklemek için, primitives kütüphanesine gelerek Block Editör penceresinde boş alanda fareyi iki kere tıklayın. or2 olarak isimlendirilen sembolü bulmak için kapı listesini aşağı yukarı kaydırarak kaydırma çubuğunu kullanın. Bu sembolü devreye ekleyin. Aynı işlemleri uygulayarak NOT kapısını devreye ekleyin. Devredeki semboller, yukarıda açıklandığı gibi fare yardımıyla seçilerek taşınabilir ve sürüklenebilir. Fare yardımıyla bir seferde birden fazla sembol seçilebilir ve taşınabilir. Seçilen semboller içlerinden herhangi biri hareket ettirilerek taşınabilir. Bu işlemleri uygulayabilirsiniz. Devrenin Şekil 3.12.'de gösterilene benzer şekilde görünmesi için devreyi hizalayın.



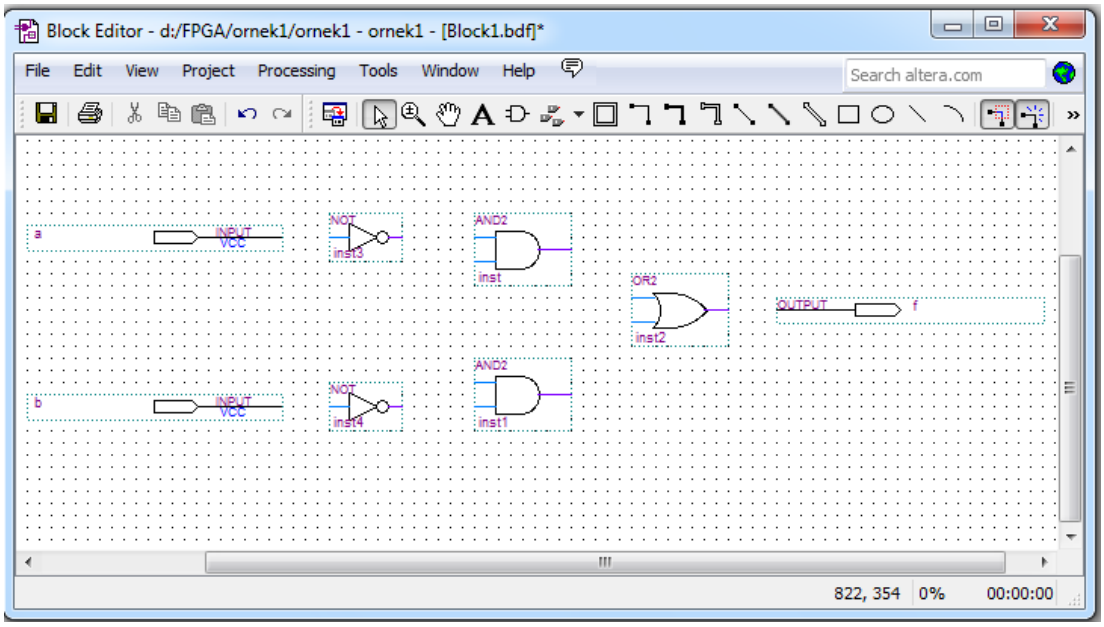
Şekil 3.12. Eklenen Kapı Sembolleri

3.3.1.2. Giriş Sembolleri ve Çıkış Sembolleri Ekleme

Lojik kapı sembolerinin kaydedilmesi için, devrenin giriş çıkış portlarında sembollerin gösterilmesi gereklidir. Tekrar primitives kütüphanesini açın. Pinlere ulaşılana kadar kapıları aşağı/yukarı kaydırın. Devreye input(giriş) olarak isimlendirilen sembolü ekleyin. Bir tane daha input(giriş) sembolü ekleyin. Devrenin çıkışını göstermek için, primitives kütüphanesini açın ve output(çıkış) olarak isimlendirilen sembolü devreye ekleyin. Şekil 3.13.'de görüldüğü gibi olacak şekilde sembolleri hizalayın.

3.3.1.3. Giriş Sembollerini ve Çıkış Sembollerini İsimlendirme

Şematikte giriş pin sembolü üzerinde sol üst köşesinde gösterilen pin adını işaretleyin ve fare yardımıyla iki sefer tıklayın. Yeni pin ismi girilerek pin adı seçilir. Giriş pinlerini sırasıyla a ve b olarak isimlendirin. Son olarak da, çıkış pinini f olarak isimlendirin.



Şekil 3.13. Kapılar ve Pinlerin Düzenlenmesi

.

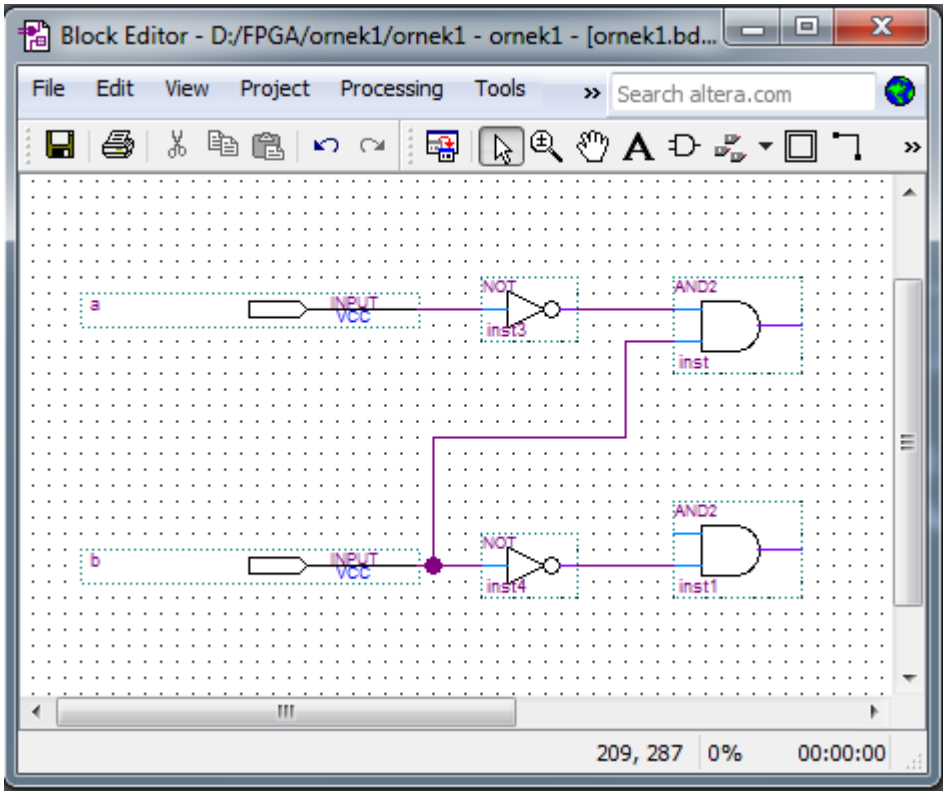
3.3.1.4. Hat (wires) ile Nod (nodes) Bağlantısı

Sonraki adım, devrede sembolleri birbirine bağlamak için hatlar çizmektir. Yatay araç çubuğunda büyük bir ok ucu gibi görünen simgeyi seçin. Bu simge Selection Tool (Seçim Aracı) olarak isimlendirilir ve Block Editörü ekranda bir sembolü seçerek veya semboleri birbirine bağlamak için hat çizim modları arasında sembollerin otomatik olarak değiştirilmesine imkan tanır. Farenin gösterdiği yere bağlı olarak uygun mod seçilir.

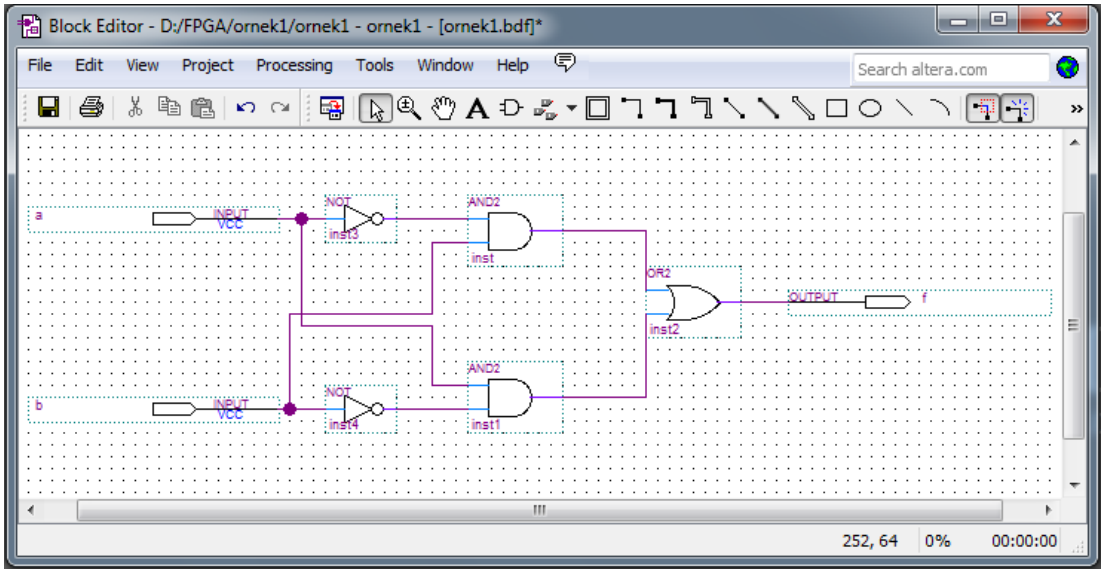
Fare işaretçisini a giriş sembolünün üzerine taşıyın. Sağ tarafta sembol üzerinde herhangi bir işaretleme dışında fare işaretçisi çapraz ok uçları gibi görünür. Farenin düğmesinin basılı durumda olması sembolün seçili olduğunu gösterir. a giriş sembolünün sağ tarafında pinstub olarak isimlendirilen küçük hat üzerine fare işaretçisini taşıyın. Fare işaretçisi, devrede bir diğer konuma pinstub bağlanması için hat çizimine izin veren artı imlecine dönüşür. Bir devrede iki veya daha fazla bağlantı node (düğüm) olarak isimlendirlilir. Bu isim node(düğüm) teriminin hatlar ile birbirine bağlı devredeki işaretçilerin sayısı olarak ifade edildiği elektrik terminolojisinden türetilmiştir.

Aşağıda gösterildiği gibi, devrenin üzerine AND kapısına a giriş sembolü bağlayın. Fare a sembolü üzerinde pinstubu işaret ederken, farenin düğmesine tıklayarak basılı tutun. Çizilen hat AND kapısının girişindeki pinstuba ulaşıncaya kadar fareyi sağ tarafa sürükleyin, sonra düğmeyi serbest bırakın. Böylece, iki pinstub birbirine bağlı ve devrede tek nod(düğüm) bulunmaktadır.

b giriş sembolü üzerinde pinstub üzerinden AND kapısı üzerine diğer girişe hat çizmek için aynı işlemleri uygulayın. Sonra, pinstub üzerinden AND kapısına b bağlantısına ulaşana kadar NOT kapısının girişine yukarı doğrultuda bir hat çizin. Farenin düğmesini serbest bırakın ve bağlantı noktasının otomatik olarak çizildiğini gözlemleyin. b giriş sembolüne karşılık üç adet pinstub, AND kapısı girişi ve NOT kapısı girişi devrede tek nod(düğüm) ile gösterilir. Şekil 3.14. bağlantıların yapıldığı devre bölümünün büyütülmüş görünümünü gösterir. Ekranda görüntülenen devre bölümünü artırmak veya azaltmak için, araç çubuğunda büyüteçe benzeyen simgeyi kullanın.



Şekil 3.14. Devrenin Genişletilmiş Görünümü



Şekil 3.15. Tamamlanmış Devre

Devreyi tamamlamak için, NOT kapısının çıkışını daha düşük seviyedeki AND kapısına bağlayın ve AND kapısına b giriş sembolünü bağlayın. İki AND kapısının çıkışlarını OR kapısına bağlayın ve OR kapısını da f çıkış sembolüne bağlayın. Sembolleri bağlarken herhangi bir hata yapıldığında, hatalı hatlar fare yardımıyla seçilebilir ve sonra Delete(sil) tuşuna basılarak veya Edit|Delete seçimi ile kaldırılabilir. Tamamlanmış devre Şekil 3.15.'de gösterilmektedir. File|Save As seçimini kullanarak devreyi ornek1 ismi ile kaydedin. Dosyanın ornek1.bdf ismi ile kaydedildiğine dikkat edin.

Devre tasarımını kapılardan birini seçip hareket ettirerek yeniden düzenlemeyi deneyin. Bütün bağlantı hatlarının otomatik olarak ayarlandığı kapı sembolünü taşıyın. Quartus II'nin öntanımlı olarak etkinleştirilmiş rubberbanding olarak isimlendirilen bir özelliği ile, Selection Tool (Seçim aracı) seçildiğinde meydana gelir. Araç çubuğunda köşesinde küçük onay işaretleri olan L - şeklinde bir hat gibi görünen bir rubberbanding simgesi bulunmaktadır. Bu simgenin rubberbanding kullanımını göstermek için vurgulandığına dikkat edin. Simgeyi kapatın ve kapıyı bu özelliğin etkisini görmek için taşıyın.

Örnek devremiz oldukça sade olmakla birlikte, karmaşık devreler çizmeye gerek kalmadan devredeki bütün hatları çizmek de kolaydır. Bununla birlikte, daha büyük

devrelerde bağlantılı olması gereken bazı nodlar(düğüm) uzak düştüğü durumlarda bu nod(düğüm)lar arasına hat çizmek kullanışlı değildir. Buna benzer durumlarda, hat çizimi yerine nod(düğüm)lar etiketleme yolu ile bağlantılanır. Daha ayrıntılı açıklama için Help bölümünü inceleyin.

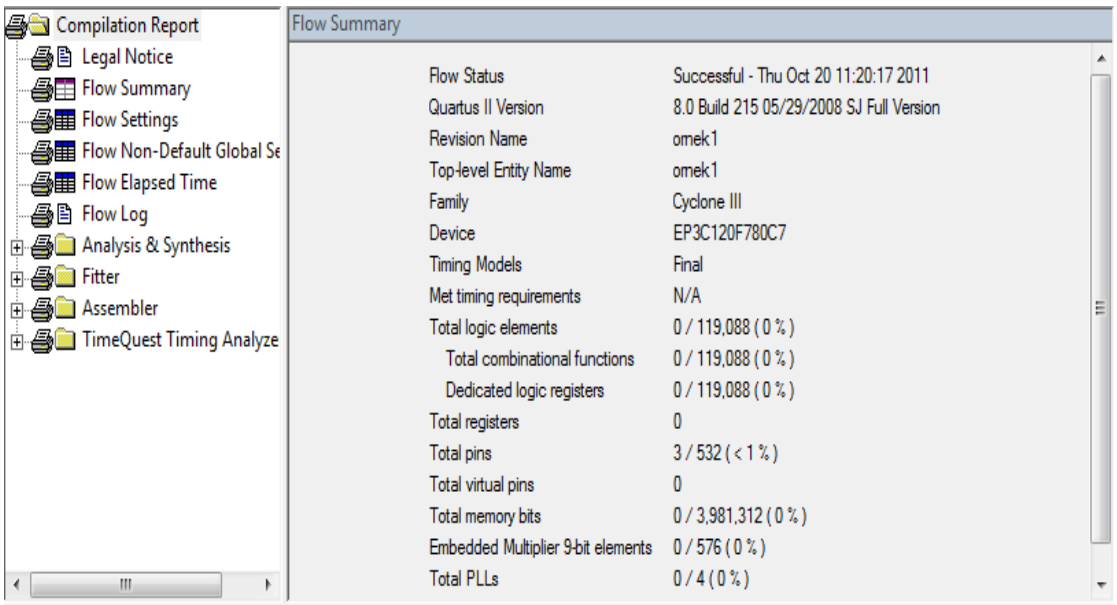
3.3.2. Derleyici Kullanımı

Quartus II içinde bulunan CAD araçları bir dizi modüllere ayrılır. Analysis & Synthesis modülü Quartus II yazılımında sentezleme adımlarını yürütür. Lojik elemanlardan temel çip içerisinde her bir elemanın doğrudan uygulandığı devre üretimini yapar. Fitter modülü sentez yöntemiyle üretilen elemanların her birinin çip üzerindeki tam konumunu belirler.

Quartus II modülleri Compiler(Derleyici) olarak isimlendirilen uygulama programı tarafından kontrol edilir. Derleyici bir seferde bir modül çalıştırmak için kullanılabilir, veya sırayla birden fazla modül çalıştırılabilir. Quartus II kullanıcı arayüzünde Derleyiciye erişim için birden fazla yöntem bulunmaktadır. Derleyiciye erişmenin yaygın yöntemlerinden bir diğeri, Processing | Start seçimini kullanmaktır. Processing | Start | Start Analysis & Synthesis , Synthesis modülünü çalıştıran komuttur. Processing | Start | Start Analysis & Elaboration komutu kullanılarak Synthesis modülü bölümü çağrılabilir. Bu komut yalnıza Synthesis modülünün tasarım projesinde söz dizimi hatalarının kontrol edildiği ve projede sunulan yardımcı tasarım isimlerinin tanımlandığı başlangıç bölümlerini çalıştırır. Processing | Start Compilation komutu ise sırasıyla Analysis & Synthesis, Fitter, Assembler ve TimeQuest Timing analyzer komutlarını ardarda çalıştırır. Bu komut için araç çubuğunda da mor üçgene benzer bir simge bulunmaktadır.

CAD araçlarını kullanmanın etkili bir yolu da tasarım sürecinin her hangi bir aşamasında sadece gerekli modülleri çalıştırmaktır. Bu yaklaşım kullanışlıdır çünkü bazı CAD araçları büyük çapta tasarım projelerini tamamlamak için uzun zaman gerektirebilir. Bu dökümanın amacı, şematiğimizin fonksiyonel tasarımını uygulamaktır. Bu işlemi yürütmek için sadece Synthesis çıkışı gerekli olduğundan, sadece Synthesis modülünü çalıştıracağız.

Processing | Start | Start Analysis & Synthesis komutunu seçin, bu komutu araç çubuğunda karşılayan simgeyi kullanın, veya Ctrl-k kısayolunu kullanın. Derleme devam ederken, ilerleme Quartus II ekranında sağ alt köşede ve sol tarafta Status yardımcı penceresinde rapor edilir (eğer pencere açılmazsa, View | Utility Windows | Status seçimiyle pencereye erişilebilir). Başarılı (veya başarısız) derleme açılır kutu ile gösterilir. OK seçeneği ile bu iletiyi onayı, derleme raporunun kontrolu Şekil 3.16.'da gösterilir.(eğer rapor açılmazsa, araç çubuğunda uygun araç kullanılarak Compiler Tool(Derleyici Aracı) üzerinde Report seçimi veya Processing | Compilation Report ile erişilebilir). Rapor özeti Cyclone II FPGA içinde küçük tasarımımızın sadece dört pin ve bir lojik eleman kullandığını gösterir.



Şekil 3.16. Derleme Raporu Özeti

Derleme raporu tasarımcıya, ilgisini çekebilecek bir çok bilgi sunar. Örneğin, sentezlenmiş lojik ifade biçiminde ayrıntılı uygulama derleyici raporunda Analysis & Synthesis yanındaki küçük + sembolü ve Equations seçilerek görülebilir. Devremizin uygulanması için Quartus II yazılımının kullandığı eşitlik:

f = a!b +ab!

Raporda AND için &, OR için # ve NOT için ! ifadeleri kullanılır. Derleme raporu istendiğinde Processing | Compilation Report seçimiyle veya mavi çip üzerinde beyaz sayfaya benzeyen ilgili araç çubuğu seçilerek açılabilir.

3.3.2.1. Hatalar

Quartus II derleme sürecinde mesajları, mesaj penceresinde görüntüler. Bu pencere Şekil 3.1.'de gösterildi gibi Quartus II ekranının alt kısmındadır. Şematik doğru olarak çizilirse, mesaj olarak 'Hata bulunamadı ve derleme süreci başarıyla sonuçlandı' ifadesiyle karşılaşılacaktır.

Hata yapıldığında ne olacağını görmek için, b girişini alttaki AND kapısında bağlayan hattı kaldırın ve değiştirilmiş şematiği derleyin. Bu durumda, derleme başarılı değildir ve iki hata mesajı görüntülenir. İlk olarak 'Söz konusu AND kapısının kaynağının bulunamadığı' tasarımcıya iletilir. İkinci olarak 'Hata bulundu' uyarısı tasarımcıya iletilir. Büyük bir devre söz konusu iken, her hangi bir hatanın konumunu bulmak zor olabilir. Quartus II yazılımı, hata mesajı üzerine iki sefer tıklandığında kullanıcıya yardımcı olur ki ilgili konum(bizim örneğimizde AND kapısı için) vurgulanır. Kaldırılmış hattı yeniden bağlayın ve doğrulanmış devreyi yeniden derleyin.

3.3.3. Tasarım Devresi Simülasyonu

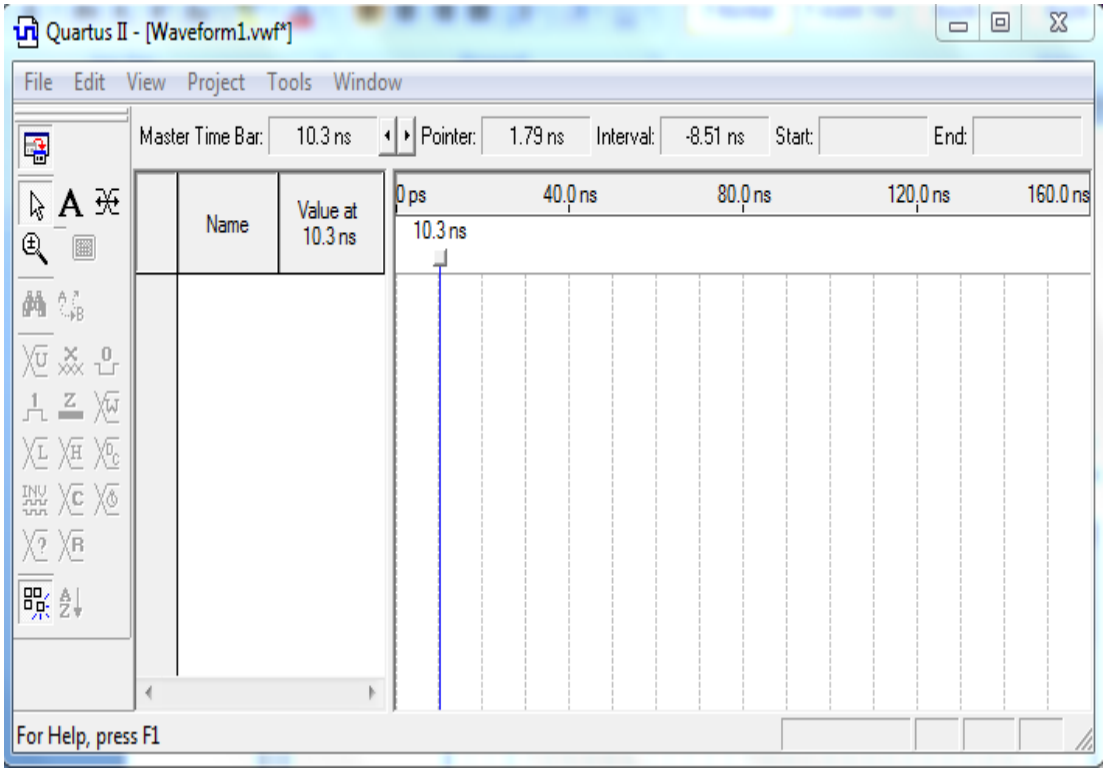
Quartus II yazılımında tasarlanan devrenin davranışını simüle etmeye yardımcı simülasyon araçları bulunmaktadır. Devre simüle edilmeden önce, giriş sinyallerini göstermek için test vectors olarak isimlendirilen dalga şekillerini oluşturmak gereklidir. Test vectors çizmek için Quartus II Waveform Editörünü kullanacağız.

3.3.3.1. Waveform Editörü Kullanımı

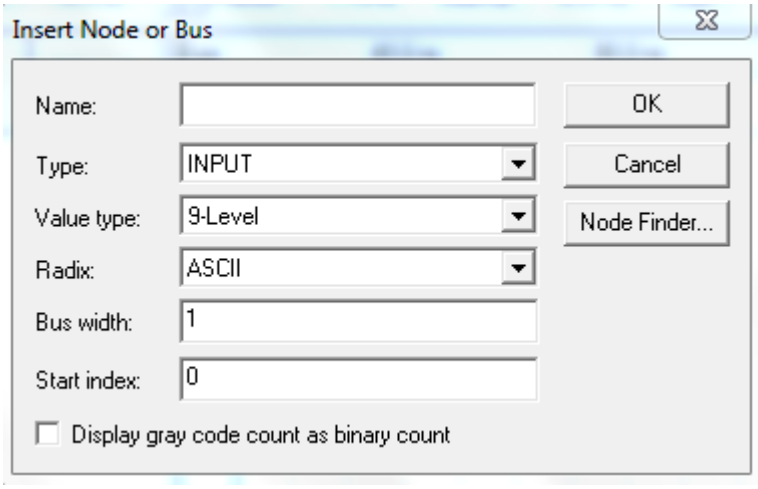
File | New seçimi ile Şekil 3.9. ile gösterilen pencereden Waveform Editörünü açın. Vector Waveform File seçin, OK düğmesi ile onaylayın.

Waveform Editör Penceresi Şekil 3.17.'de gösterildiği gibidir. İstenen simülasyonu Edit | End Time seçimi ile açılan iletişim kutusuna 160 ns girildikten sonra 0-160 ns olarak düzenleyin. 0-160 ns bütün simülasyon aralığını pencerede görüntülemek için View | Fit in Window seçimini uygulayın. Pencerenizi maksimum boyutuna ayarlamak isteyebilirsiniz.

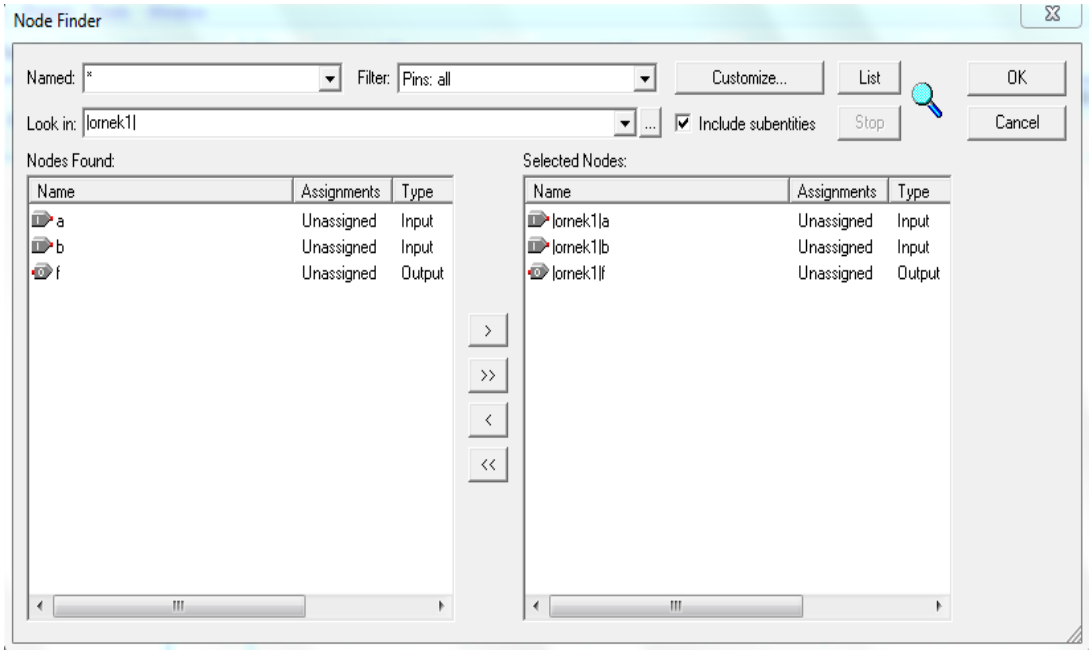
Sonrasında, simüle edilecek devrenin giriş ve çıkış nod(düğüm)ları kapsama dahil etmek istiyoruz. Bu işlem, Node Finder uygulaması kullanılarak yapılır. Şekil 3.18.'de gösterilen pencereyi açmak için Edit | Insert Node or Bus seçimini uygulayın. Name kutusu içine sinyal(pin)in ismi yazılabilir fakat Şekil 3.19'da gösterilen pencereyi açmak için Node Finder uygulamasının seçilmesi daha kullanışlıdır. Node Finder uygulamasında ne çeşit nod(düğüm)ların bulunduğunu göstermek için kullanılan bir filtre bulunmaktadır. Giriş ve çıkış pinleriyle ilgilendiğimiz için,filtreyi Pins :all durumuna göre ayarlayın. Giriş ve çıkış nod(düğüm)larını bulmak için List düğmesini seçin.



Şekil 3.17. Waveform Editör Penceresi



Şekil 3.18. Nod Ekleme veya Bus İletişimi

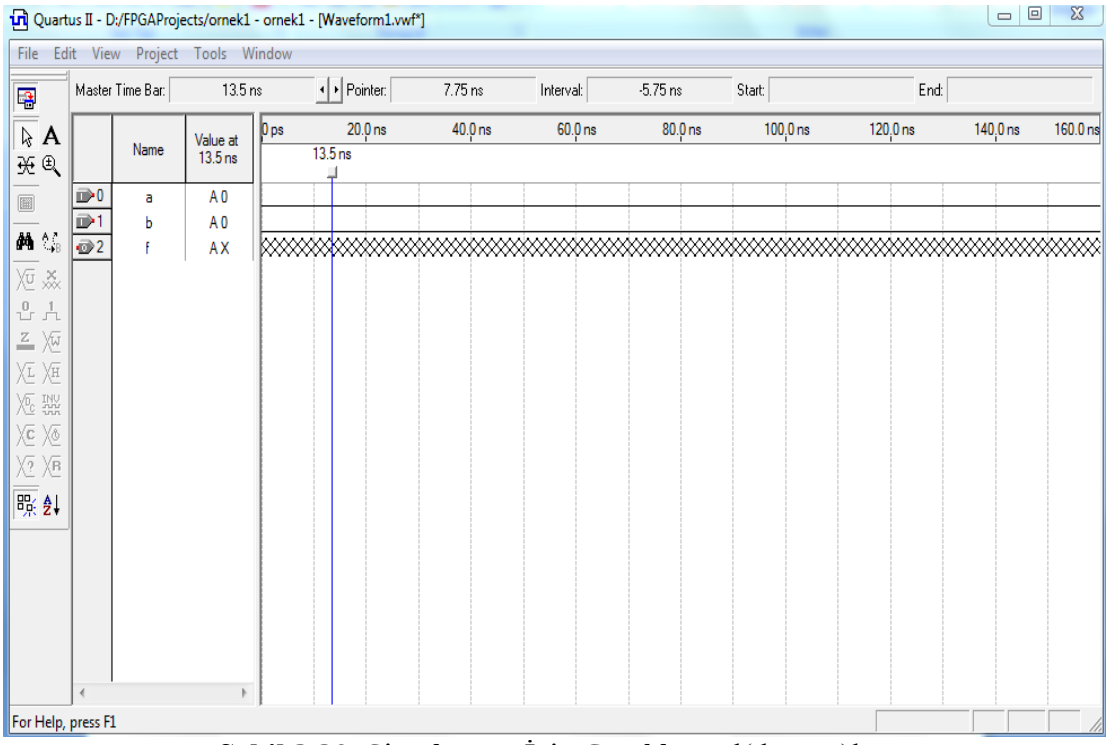


Şekil 3.19. Node Finder Penceresi

Node Finder, pencerenin sol tarafında f, a ve b nod(düğüm)larını görüntüler. a nod(düğüm)unu seçin ve > işareti ile şeklin sağ tarafındaki Selected Nodes bölümüne ekleyin. Aynı işlemi f ve b için de yapın. NodeFinder penceresini kapatmak için OK düğmesini onaylayın, ve sonra ekrana gelecek olan Şekil 3.18'de gösterilen pencerede OK düğmesini onaylayın. Böylece, Şekil 3.20.'de gösterildiği gibi Waveform Editör penceresinin tam görünümü görüntülenir. Şekil 3.20.'de gösterildiği gibi nod(düğüm)ları aynı sırada seçmediyseniz, nod(düğüm)ları yeniden sıralayabilrsiniz. Waveform Editör penceresinde dalga biçimini aşağı ya da yukarı

taşımak için, nod(düğüm) ismini (Name sütununda) seçin ve farenin düğmesini serbest bırakın. Dalga biçimi, seçimi bölümü vurgulu olarak gösterir. Dalga biçimini yeniden seçin ve dalga biçimini Waveform Editöründa aşağı yada yukarı sürükleyin.

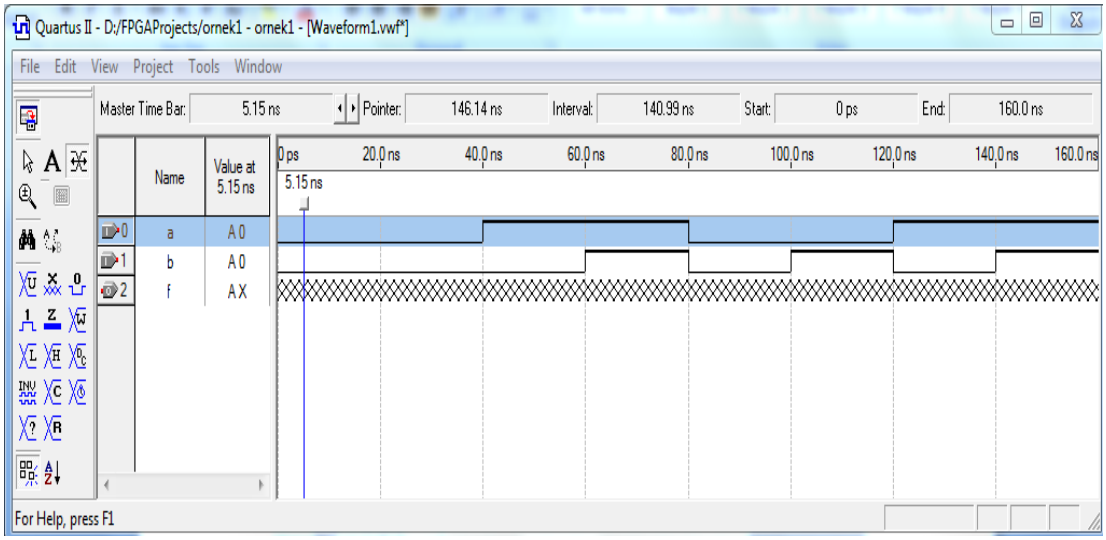
Simülasyon süresince giriş sinyalleri için kullanılacak lojik değerleri belirleyeceğiz. f çıkışındaki lojik değerler simülatör tarafından otomatik olarak üretilir. İstenilen dalga biçimlerini çizimini kolaylaştırmak için, Quartus II yazılımı dikey kılavuz çizgilerini görüntüler(varsayılan ayar) ve hat(View | Snap to Grid seçilerek çalıştırılabilen) çizimi için kolay çizim özelliği sağlar. Ayrıca üstü işaretlenerek taşınabilen ve yatay olarak sürüklenebilen, dikey aralıksız hattı inceleyin. Dalga biçimleri dikey araç çubuğu üzerindeki büyük ok başına benzeyen simge seçilerek etkinleştirilen seçim aracı kullanılarak çizilebilir.



Şekil 3.20. Simülasyon İçin Gerekli Nod(düğüm)lar

Büyük bir devrenin davranışını simule etmek için, yeterli sayıda giriş değerlendirmelerin kullanılması ve beklenilen çıkış değerlerinin gözlenmesi gereklidir. Olası giriş değerlendirmelerinin sayısı çok fazla olabilir, dolayısıyla bu giriş değerlendirmeleri için görece küçük(fakat tipik) örneklerin seçilmesi gereklidir.

Devremiz oldukça küçük olduğu için a ve b girişlerinin olası sekiz değerlendirmelerinin uygulanması ile tam olarak simule edilebilir. Her 20 ns için yeni bir değerlendirme uygulayalım. Başlangıçta, tüm girişler sıfır olsun.20 ns anında, b'den 1e gitmek istiyoruz. b seçimi; sinyali vurgular ve seçili dalga biçimini şekillendirmemize imkan veren dikey araç çubuğu etkinleştirir. Bu araç çubuğu, signal to 0, 1, unknown (X), high impedance (Z), don't care (DC), and inverting its existing value (INV) gibi seçeneklerin ayarlanmasını sağlar. Bu arada karma model ile gösterilen bir bilinmeyen değere sahip f çıkışının görüntüsünü inceleyin. Belirli bir zaman aralığı, aralık başlangıcında dalga biçimi üzerinde fare yardımıyla seçilir ve sonunda sürüklenir; seçili aralık vurgulanır. b için 20-40 ns zaman aralığını seçin ve sinyali 1 olarak ayarlayın. Benzer biçimde, b için sinyal 1 ve 60-80 ns,100-120 ns ve 140-160 ns olarak ayarlayın. Ardından, a için sinyal 1 ve 40-80 ns ve 120-160 ns olacak şekilde ayarlayın. Şekil 3.21.'de gösterilen görüntüyü elde etmek için kalan seçimleri tamamlayın.



Şekil 3.21. Bütün Test-Vectors

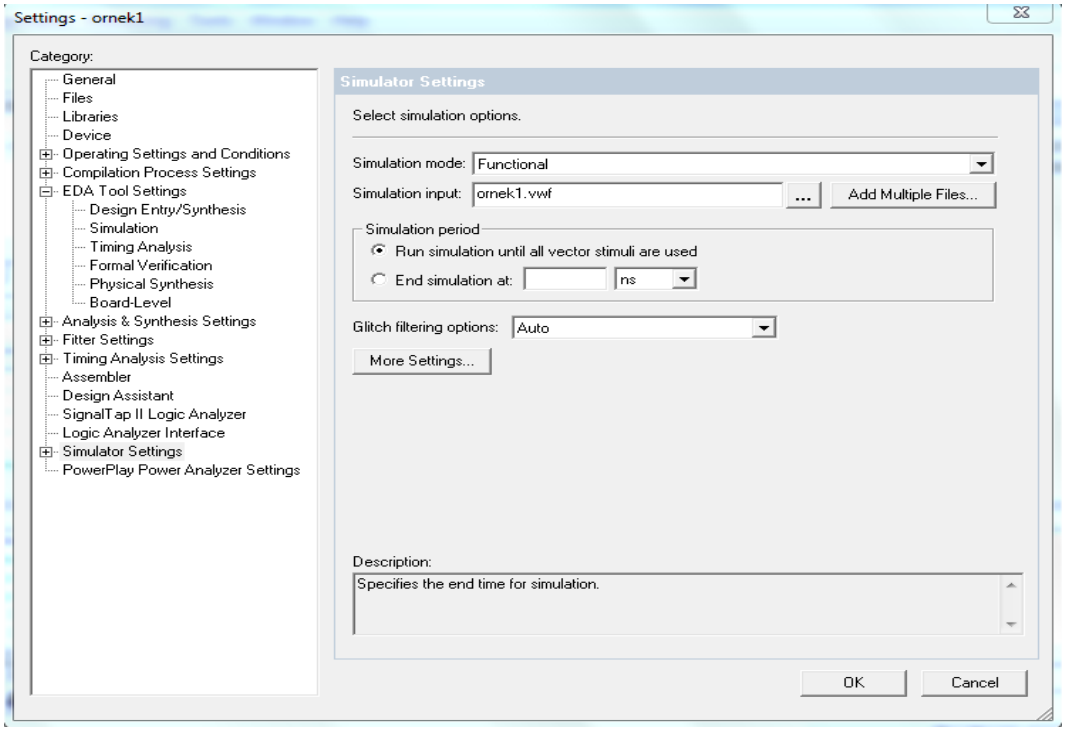
Giriş dalga biçimleri değişimi için uygun mekanizma, dalga biçimi düzenleme aracı ile sağlanır. Bu uygulamanın sol ve sağı işaret eden iki oka benzeyen simgesi dikey araç çubuğunda bulunmaktadır. Dalga biçiminin 0 (1) olduğu yerde fare bazı zaman aralıkları üzerine sürüklendiğinde, dalga biçimi 1(0) olarak değişecektir.

3.3.3.2. Simülasyonun Yürütülmesi

Bir devre iki şekilde simule edilebilir. En kolay yöntem, lojik elemanların ve ara bağlantı hatlarının mükemmel olduğunu, böylece devre içinde sinyallerin yayılmasında hiçbir gecikme olmadan yayıldığını varsaymaktır. Bu durum fonksiyonel simülasyon olarak isimlendirilir. Daha karmaşık bir alternatif ise, zamanlama simülasyonuyla sonuçlanan tüm yayılım gecikmelerini hesaba katmaktır. Genellikle, fonksiyonel simülasyon tasarlanan bir devrenin fonksiyonel doğruluğunu kanıtlamak için kullanılır. Bu, daha az zaman alır çünkü simülasyon devreyi tanımlayan lojik ifadeler kullanılarak kolay bir şekilde yürütülebilir. Bu dökümanda yalnız fonksiyonel simülasyon kullanılacaktır.

Fonksiyonel simülasyonu yürütmek için, Settings penceresini açmak için Assignments |Settings seçimini uygulayın. Pencerenin sol tarafında, Şekil 3.22.'deki pencereyi görüntülemek için Simulator seçeneğini ve simülasyon modunu da Functional olarak seçin. Simülatörün kurulumunu tamamlamak için, Processing |Generate Functional Simulation Netlist komutunu seçin. Quartus II simülatörü test girişlerini alır ve example_schematic.vwf dosyasında tanımlı çıkışları üretir. Bir simülasyon çalışması, Processing |Start Simulation seçimi ile veya araç çubuğu üzerinde altındaki kare dalga ile beraber mavi üçgene benzeyen kısayol simgesi kullanılarak başlatılabilir. Simülasyonun bitiminde, Quartus II yazılımı, simülasyonun başarıyla tamamlandığını bildirir ve bir simülasyon raporu görüntüler.

Şematik görüntüsü kullanılarak tasarıma giriş bölümünü tamamlamış bulunmaktayız. Hazırda açık bulunan projeyi kapatmak için File|Close Project seçimini uygulayın.



Şekil 3.22. Simülasyon Modunun Belirlenmesi

3.3.4. Quartus II Windows

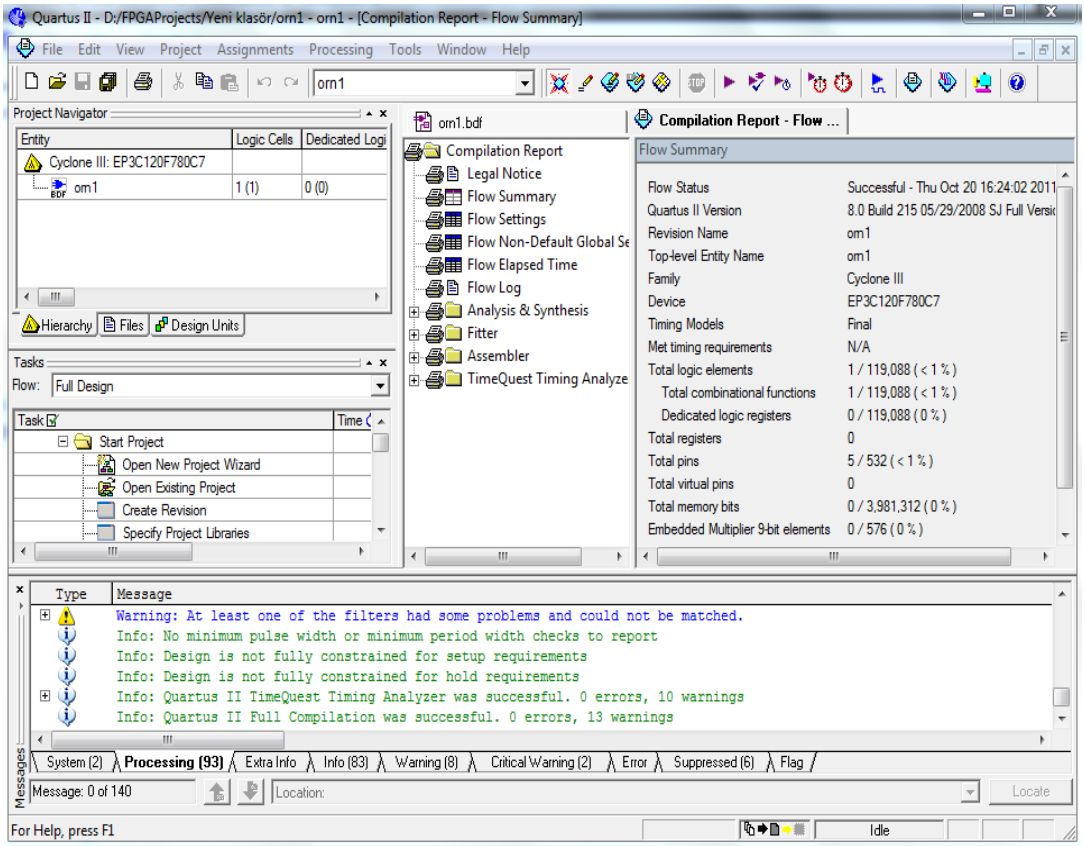
Quartus II görüntüsü, ekranda değişik yerlere konumlandırılabilen, boyutu değiştirilebilen veya kapatılabilen bir çok yardımcı pencere içerir.

Quartus II ekranının sağ tarafında bulunan geniş alan, değişik amaçlar için kullanılabilir. Göründüğü üzere, Block Editör, Text Editör ve Waveform Editör tarafından kullanılabiliyor. Ayrıca, bu alan derleme ve simülasyon sonuçlarını sörüntülemek için de kullanılabilir.

Yardımcı pencere, başlık çubuğu yardımıyla sürüklenip taşınabilir, pencere sınırları kullanılarak yeniden boyutlandırılabilir veya sağ köşedeki X simgesi yardımıyla kapatılabilir. Belirli bir yardımcı pencere, View |Utility Windows seçimi ile açılabilir.

Quartus II yazılımında, Quartus II araçlarında kullanılan komutlar bağlama duyarlıdır(context sensitive). Örneğin, Text Editör kullanılırken, Edit menüsü

Waveform gibi diğer araçlar kullanılırken bulundurulan komutlardan farklı komutlar bulundurur.



Şekil 3.23. Quartus II Görünümü

4. BZK.SAU MİKROBİLGİSAYAR MİMARİSİ

Eğitimsel amaçlı bir mikrobilgisayar olan BZK.SAU 2009 yılında yüksek lisans tez çalışması [18] olarak hazırlanmıştır. 2012 yılında ise bir doktora tezi [19] kapsamında tasarım simülatif ortamdan yeniden yapılandırılabilir donanımlardan olan FPGA geliştirme ortamına taşınmıştır. Taşınma işleminin ardından kullanıcın sisteme müdahil olması amacı doğrultusunda, mikro bilgisayar mimarisine modülerlik özelliği katılmıştır. Oluşturulan bu özgün donanım üzerine sıfırdan bir işletim sisteminin nasıl oluşturulacağı konusunda çalışmalar yapılmıştır. BZK.SAU.FPGA sistemine ait tasarım özellikleri Tablo 4.1'de verilmiştir.

Tablo 4.1. BZK.SAU.FPGA Sistem Özellikleri

Özellik	Açıklama
Sistem Adı	BZK.SAU.FPGA
Sistemin İnşa Edildiği Donanım	Altera DE2 ve CycloneIII FPGA kartları
Çıkış Birimi	VGA Monitör (640×480)
Ekran Alanı	40 sütun×24 satır(320×384)
Giriş Birimi	PS/2 klavye
Sistem Tanımlama Dili	Şematik(Donanımsal)
İşlemci Mimarisi	Von-Neumann(SISD mimarisi)
İşlemci Tipi	16-bit
Adres Yolu	16-bit
Veri Yolu	16-bit
Sistem Kaydedicileri	10 adet(Giriş ve çıkış kaydedicileri 8-bit diğerleri 16-bit genişliğinde)
Ana Bellek	64 KB – 16 bit
İkincil Bellek	Flash Bellek(4 MB) – 8 bit
Bellek Kelime Yerleşim Düzeni	Big-Endian
Komut Mimarisi	CISC

Komut Seti	Fonksiyonel, Kontrol, Transfer, Giriş-Çıkış ve Yığın Komutları(59 komut)
Komut Yapısı	16 bit(15-12. bitler adresleme modu, 11-0. bitler opcode alanı)
Komut İşleme Metodu	None Pipeline
Adresleme Mod Çeşidi	6(İvedi, direkt, dolaylı, indeks, göreceli ve doğal)
Kontrol Birim Yapısı	Donanımsal
ALU Birimi	16-bit(Sadece tamsayılar)
Sayı Sistemi	2'ye Tümleme
İşletim Sistemi	Tek Kullanıcı-Tek Görev
Dosya Sistemi	FAT
Assembly Dili	BZK.SAU Assembly programlama dili

Tasarımı yapılan bilgisayar mimarisinde 640×480 piksel çözünürlüğüne sahip VGA tipinde monitör kullanılmıştır. Monitör donanımı sadece metin ve tek renk(kırmızı) kipinde çalışan bir donanım olup, monitör ekranındaki her bir karakter 8×16 piksel ile temsil edilmektedir. Bellek boyutunun yetersiz oluşu nedeniyle monitör ekran alanının 320×384 piksel ebatlarındaki bölümü kullanıldı. Bu alan ABC80 [20], Apple I ve Apple II [21] gibi sistemlerde kullanılan 40 sütun ve 24 satırdan oluşan bir alana tekabül etmektedir. Başka bir deyişle monitör ekran alanında toplam 896 karakterin görüntülenmesi mümkün olmaktadır.

FPGA geliştirme kartları üzerinde tasarlanan sistem, temel olarak ikiye ayrılan ve yazılımsal ve donanımsal sistem tanımlama dilleri olarak adlandırılan diller kullanılarak tasarlanır. Bu tasarımda güdülen politikanın eğitimsel amaçlı olması, tasarlanan sistemin detaylarının öğrenciler tarafından incelenebilmesine olanak tanımasını gerektirir. Bu nedenden ötürü ve literatürde yer alan eğitimsel amaçlı bilgisayar mimarisi [22] tasarımlarında da kullanılan donanımsal tanımlama dili şematik tasarım, bu mimarinin temel sistem tanımlama dili olmasında etken olmuştur.

Literatürde eğitimsel amaçlı tasarlanan işlemcilerin [23,24] adres ve veri yolu genişliği genelde 16-bit olarak karşımıza çıkmaktadır. Ayrıca işlemci adres ve veri yolu tasarımlarının genişliği 16-bit'in üzerinde tasarlandığında yapının karmaşıklığı artacağından ve dolayısıyla öğrenimi zorlaştıracağından eğitimsel açıdan uygun değildir. Projede tasarımı yapılan bilgisayar mimarisinin adres ve veri yolu genişliği hem literatürde yapılan tasarımlar göz önüne alındığında hem de eğitimsel açıdan adres ve veri yolu tasarımının öğrenciler tarafından optimum sürede kavranmasını sağlayacak şekilde 16-bit genişliğinde tasarlanmıştır. Sistemde kullanılan kaydedicilerin genişliği 16-bit olduğundan sistem veri yolu 8-bit olarak tasarlanması durumunda verilerin yol üzerindeki aktarım işlemi için geçen sürenin uzamasına neden olacaktır. Adres yolu 16-bit olarak tasarlandığından dolayı, bu yolun adresleyebileceği maksimum bellek alanı olan 64 KB büyüklüğünde ana bellek oluşturulmuştur.

Projede kullanılan FPGA geliştirme kartlarının üzerinde yer alan flash bellek 4MB kapasiteye sahiptir. Mimarinin kullanıcı programlarını kaydedebileceği mevcut kaynaklar içinde tek birim flash bellek olduğundan, ikinci bellek birimi olarak flash belleğin seçilmesinde etkili olmuştur. Hem ana bellek olsun hem de ikincil bellek birimi olsun bu birimlerde bellek kelime yerleşim düzeni big-endian yapısında yerleştirilmiştir. Bu yapının seçilmesindeki amaç mimaride kullanılan verilerin günlük hayatta kullanılan verilerde kullanılan düzende olmasından kaynaklanmaktadır. Başka bir deyişle günlük hayatta kullanılan verilerin en anlamlı kısmı en solda, en düşük anlamlı kısmı ise en sağda yer almaktadır.

Bir bilgisayar mimarisinin tam olup olmaması sahip olduğu komut kümesine göre şekillenmektedir [25]. Eğer komut kümesi, aritmetik ve mantıksal işlemleri yapabilen, bellek ve kaydediciler üzerinde veri akışını sağlayabilen, işlemciyi kontrol edebilen ve giriş-çıkış birimleri ile alışveriş yapabilen komutlara sahip olması durumunda bir bilgisayar mimarisi "tam" özellikli bir yapıya kavuşmuş olur. Bundan dolayı projede kullanılan komutlar hem tam bir bilgisayarda olması gereken temel komutlardan oluşmakta hem de eğitimsel açıdan komutların işleyişini kavramada öğrenme sürecini geciktirmeyecek tarzdadır. Ayrıca kullanılan komutların

işlemcideki süreçleri birbirinden farklı olduğundan bilgisayar mimarisi CISC yapıdadır. Mimaride kullanılan kaydedici birim sayısı 10 adet olup, tam özellikli bir bilgisayar mimarisinde olması gereken 16-bit genişliğinde kaydediciler seçilmiştir. Eğitimsel amaçlı bilgisayar mimarilerinde yaygın olarak kullanılan 6 adet temel adresleme modu çeşidi kullanılmaktadır. Seçilen komut yapısı, kullanılan komut ve adresleme modlarına göre oluşturulmuş olup 15 ve 12. bitler arası adresleme modunu temsil ederken; 11 ve 0. bitler arası komutun işlem kodunu temsil etmektedir. Komutların kontrolünde programlama bilgisini gerektirecek mikro program yapısı yerine eğitimsel açıdan öğrenme sürecini optimum seviyeye çekecek olan donanımsal kontrol yapısının kullanılması tercih edilmiştir.

Bilgisayar Mimarisinde bulunan aritmetik ve lojik birimi mevcut haliyle 16-bit tamsayılar üzerinde işlem yapabilmektedir. Negatif sayılarını gösterimi için literatürde sıkça kullanılan 2'ye tümleme mantığı kullanılmıştır.

Tasarlanan işletim sistemi projenin temel amacı olan eğitimsel yönü nedeniyle basit seviyede oluşturulmuştur. Bundan dolayı tasarlanan işletim sistemi tek kullanıcı ve tek görev yapısında bir işletim sistemidir.

5. FIRÇASIZ DC MOTORLAR

Bu bölümde fırçasız DC motorların genel yapısı, sürücü devresi ve sürüş tekniği anlatılacaktır.

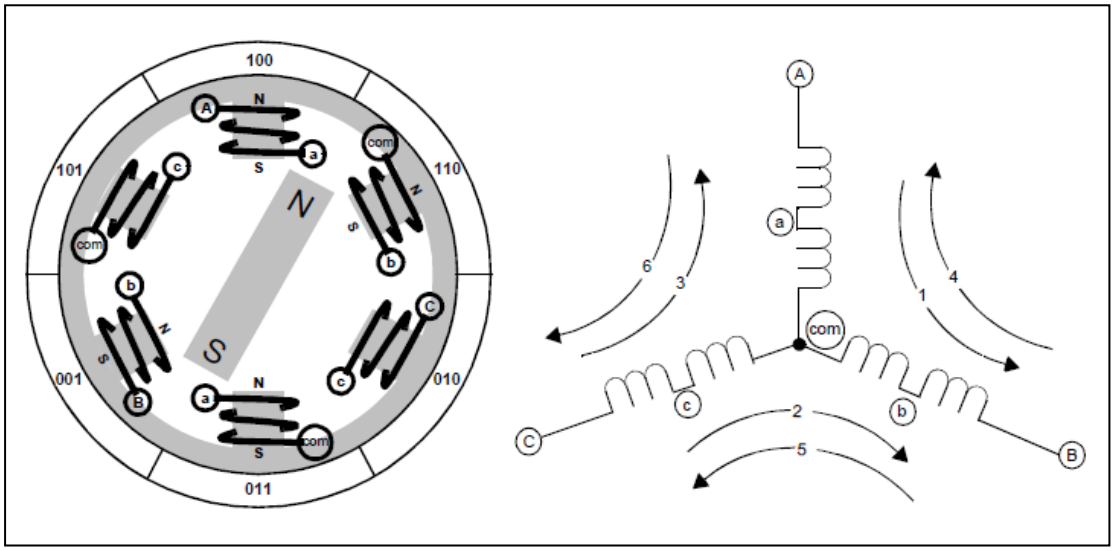
5.1. Giriş

Fırçasız DC motorlar yüksek verimleri ve düşük bakım gereksinimleri ile endüstride kullanım alanları giderek artan cihazlar olarak karşımıza çıkmaktadır. Doğrudan kar amaçlı olan evsel ve ticari uygulamalarda (örnek olarak buzdolabı, klima sistemleri gibi) düşük maliyetleri nedeniyle düşük verimli ve daha yüksek bakım gereksinimli konvansiyonel motorlar kullanılsa da yüksek verim, hassasiyet ve güvenilirlik gerektiren uygulamalarda (otomotiv sektörü, uzay teknolojileri, bilgisayar teknolojileri, tıp elektroniği, askeri alanlar, robotik uygulamalar gibi) fırçasız DC motorlar daha tercih edilir ürünlerdir. Genel olarak üstünlüklerini yüksek verime sahip olmaları, fırçasız yapıları sebebi ile ark oluşturmamaları ve bakım gerektirmemeleri, küçük boyutta yüksek moment üretebilmeleri, yüksek hızda ve sessiz çalışması, kolay soğutulabilmesi olarak sıralayabiliriz. Her ne kadar yüksek maliyeti bir dezavantaj teşkil etse düşük maliyetli sürücülerle sürülebilmesi de bir avantaj sayılabilir.

5.2. Fırçasız DC Motor Yapısı

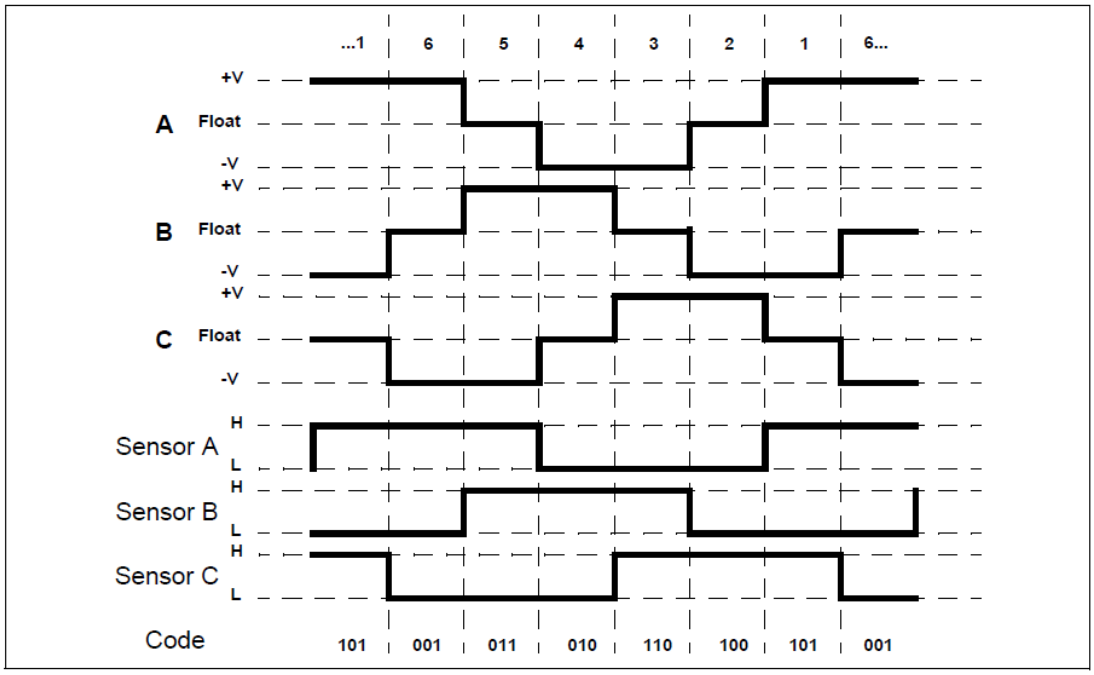
Genel olarak Fırçasız DC motorlar sabit mıknatıslı bir rotor ve tel sargılı stator kutuplarından oluşur. Tel sargılı stator kutuplarında indüklenen döner manyetik alan ile sabit mıknatıslı rotor arasındaki manyetik çekim kuvveti ile elektriksel enerji mekanik enerjiye çevrilir. Şekil 5.1.'de bir fırçasız DC motorun basitleştirilmiş şekli görülmektedir. Bu örnekte 3 adet elektromanyetik devre ortak bir noktada birleştirilmiştir. Her elektromanyetik devre merkezinden ikiye bölünmüştür, böylece sabit mıknatıslı rotorun indüklenen manyetik alan ortasında hareket etmesine izin verilir. Birçok fırçasız DC motorun bağlantı şekli 3 faz yıldız bağlantı şeklindedir. Bu bağlantı şeklindeki motorların sürülmesi aynı anda 2 fazın enerjilendirilmesiyle olur. Şekil 5.1.'deki akım yolları için ilgili motor uçlarına uygulanması gereken

voltajlar Şekil 5.2.'de gösterilmiştir. Örneğin Şekil 5.1.'deki 1 numaralı yol için motorun A ucuna pozitif voltaj, B ucuna negatif voltaj verildiğini Şekil 5.2.'den takip edebiliriz. 2 numaralı yol için de B nin negatif C nin ise pozitif voltaja karşılık geldiğini tablolardan takip edebiliriz. 1 den 6 ya kadar numaralandırılmış olan her bir kısmın aslında 60 derecelik bir motor adımını karşıladığını söyleyebiliriz. 1. durumdan 2. duruma geçilmesiyle motor saat yönünde 60 derecelik hareket yapacaktır [26].



Şekil 5.1. Basitleştirilmiş Fırçasız DC Motor Şekli [26]

Aslında fırçasız DC motor komütasyonunu rotorun pozisyonunu hissederek en fazla torku oluşturacak şekilde motor sargılarını enerjilendirmek şeklinde özetleyebiliriz. Rotor pozisyonunu en kolay şekilde hissetmek ise bir pozisyon sensörü yardımıyla olur. Bir çok fırçasız DC motor üreticisi motor üzerinde 3 elementli bir Hall Effect pozisyon sensörünü sağlar. Sensör üzerindeki her bir eleman 360 derecelik bir tam devrin yarısında yüksek seviye (high level), diğer yarısında ise düşük seviye (low level) gerilim üretirler. Her bir 60 derecelik periyotta ise sensör çıkışları değişerek rotor pozisyon bilgisi okunur. Motor sürmek için sargı uçlarına uygulanması gereken, sensör çıkışlarına karşılık gelen sargı voltajı değerleri Şekil 5.2.'de gösterilmiştir.



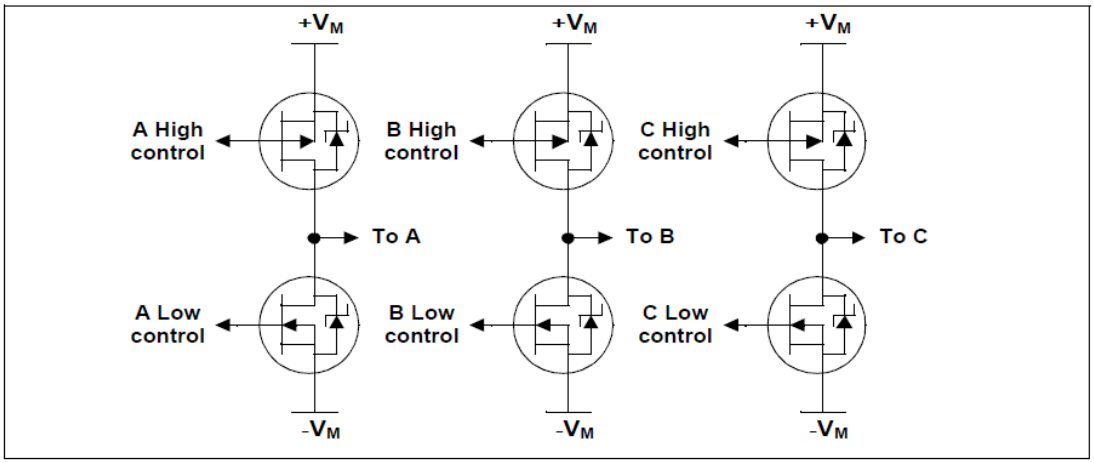
Şekil 5.2. Sensör Çıkışlarına Karşılık Gelen Sargı Gerilimleri [26]

Şekil 5.2.'nin üst kısmındaki numaralar Şekil 5.1.'deki faz akımlarına karşılık gelmektedir ve 6 aşamalı sürüşün her bir fazını göstermektedir. Her bir fazda pozisyon sensörü birbirinden farklı 6 değişik kod üretir. Pozisyon sensörünün ürettiği kodlar Şekil 5.1.'de motor şeklinin üzerinde gösterilmiştir. Burada rotorun kuzey kutbu sensörün üreteceği kodu göstermektedir. Her bir numara sensörlerin lojik seviyesini göstermektedir, en yüksek anlamlı bit C sensörünü en düşük anlamlı bit ise A sensörünü karşılamaktadır.

Her bir sürüş fazı motor terminal uçlarının bir tanesine pozitif voltaj, bir tanesine negatif voltaj verilmesi ve diğer ucun ise boşta bırakılmasıyla gerçekleşir. Basitleştirilmiş bir sürücü devresi ise Şekil 5.3.'de gösterilmiştir. Bu devre her bir motor terminal ucuna yüksek seviye ve düşük seviye gerilim uygulanmasına ve serbest bırakılmasına imkan sağlar.

Üç faz köprüsü şeklindeki devre yapısında transistörler iletime veya kesime geçirilerek Şekil 5.2.'deki sensör verisine karşılık gelen sargı gerilimleri motora uygulanır. Örnek olarak Şekil 5.2.'deki birinci adım için sensörden gelen 101 verisine karşı motorun A sargısına pozitif voltaj B sargısına negatif voltaj C sargısının ise boşta bırakılması gerekmektedir. Bunun için devremizde A high

control ve B low control transistörlerinin iletimde diğer transistörlerin ise kesimde olması gerekmektedir. Böylece Şekil 5.1.'de görülen motorun A sargısından B sargısına doğru bir akım akışı sağlanmış olur. Her bir adım için pozisyon sensörü verilerine göre transistör durumları Tablo 5.1.'de gösterilmiştir.



Şekil 5.3. Üç Faz Köprüsü [26]

Bu tip bir devrede dikkat edilmesi gereken nokta aynı sütunda olan transistörlerin aynı anda iletimde olmaması gereğidir. Transistör girişlerinde pull-up ve pull-down dirençleri kullanılarak transistörlerin daha hızlı kesime gitmeleri sağlanabilir. Transistörlerin kesime gitme süreleri iletime geçme sürelerinden daha uzun olduğu için fazlar arasındaki geçişlerde ölü zaman gecikmesi tabir edilen yeterli bir gecikme süresine ihtiyaç vardır.

Ölü zaman gecikmesini uygulamadığımız takdirde fazlar arası geçişte ilk adımın transistörleri tıkamaya geçmeden ikinci adımın transistörleri iletime girecektir. Bu durumda aynı anda iki sargı gurubu birden enerjilendirilmiş olup transistörlerde ısınma ve yanmalar meydana gelecek, motorda ise ısınma ve sarsıntı oluşacaktır. Ölü zaman gecikmesiyle 1. adımın transistörlerinin tam olarak kesime gitmesinden sonra 2. adımın transistörlerinin iletime geçmesi sağlanarak emniyetli ve sarsıntısız bir sürüş sağlanmış olur.

Tablo 5.1. Sensör Çıkışlarına Göre Sürücü Devre Girişleri [26]

Pin	RE2	RE1	RE0	RC5	RC4	RC3	RC2	RC1	RC0
Phase	Sensor C	Sensor B	Sensor A	C High Drive	C Low Drive	B High Drive	B Low Drive	A High Drive	A Low Drive
1	1	0	1	0	0	0	1	1	0
2	1	0	0	1	0	0	1	0	0
3	1	1	0	1	0	0	0	0	1
4	0	1	0	0	0	1	0	0	1
5	0	1	1	0	1	1	0	0	0
6	0	0	1	0	1	0	0	1	0

6. PARALEL PORT

Bu bölümde IEEE Std 1284'e göre paralel portun yapısı anlatılacaktır.

6.1. Giriş

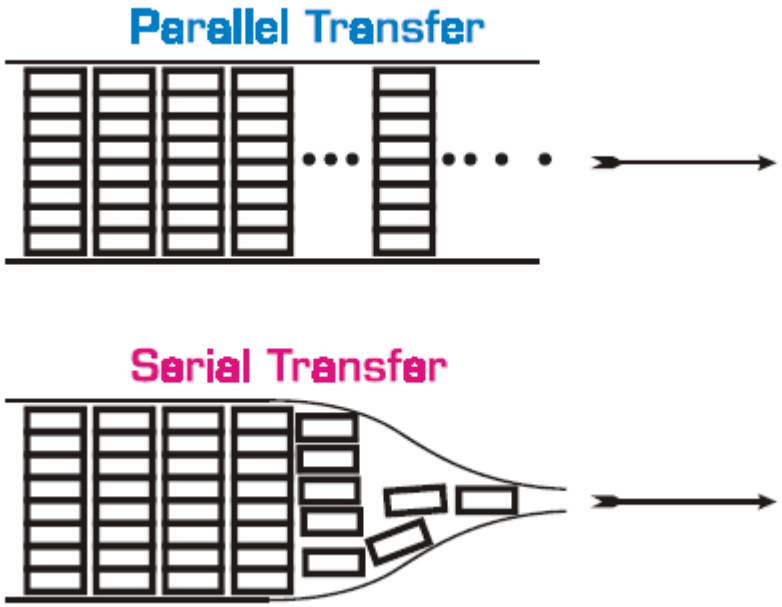
PC paralel portları daha hızlı veri gönderme isteğinin bir sonucu olarak ortaya çıkmıştır. Paralel port orijinalinde bir yazıcı bağlantı noktası olarak tasarlandı ve uzun yıllar tipik yazıcılara veri göndermek için kullanıldı. Bu data genellikle ASCII text şeklinde idi. Paralel port printer'ların yavaş ve basit olduğu dönemlerde beklediği rağbeti görmedi. Printer'lardaki gelişmeyle birlikte daha fazla hız ve kontrol gereksinimleri paralel portlara olan rağbeti arttırdı. Yazıcılara veri göndermekten başka paralel portlar harici veri depolama cihazları, çift yönlü veri alma gönderme ve kontrol uygulamaları için de kullanılırlar.

Zaman içerisindeki gereksinimlere bağlı olarak da çeşitli paralel port işletme modları ortaya çıktı. Bu modların her biri paralel portun hızını ve yeteneğini artırdı.

6.2. Paralel Port Nedir

Paralel portlar birim zamanda 1 byte(8-bit) veri gönderebilen veya alabilen kişisel bilgisayar arayüzleridir. Seri portlarla kıyaslandığında seri portlar birim zamanda yalnızca 1-bit veri gönderebilirler.

Paralel portların özellikleri IEEE'ın (Institute of Electrical and Electronic Engineers) IEEE Std 1284-1994 nolu standardınca belirlenmiştir. Temel olarak IEEE-1284 şu anda kullanmakta olduğumuz paralel iletimle alakalı standartları barındırır. IEEE-1284 bir çevresel birimle (genellikle bir yazıcı) bir sunucu arasındaki (genellikle bir bilgisayar) asenkron, fully interlocked ve çift yönlü haberleşmeyi tanımlar.



Şekil 6.1. Paralel ve Seri Transfer Karşılaştırılması [27]

Asenkron Haberleşme tipinde, bir haberleşme olayının zamanlaması bir diğer olaya tepki olarak oluşur. Haberleşme akışı içerisindeki olaylar clock olarak ayarlanmış özel bir zamanlama ile gerçekleştirilirler. Genel haberleşme zamanlaması bir clock sinyali tarafından dikte edilmez.

Fully Interlocked haberleşmede gönderilen her kontrol sinyaline karşılık bir kabul sinyali alınır. Sunucu cihaz veriyi sadece alıcı cihaz almaya hazır olduğu zaman gönderir.

Son olarak çift yönlü haberleşmede ise veri aktarımı iki yönlüdür. Data transferi sunucudan çevresel aygıta doğru olduğu gibi, çevresel aygıttan sunucuya doğruda yapılabilir.

6.3. Paralel Port Modları

Asenkron, fully interlocked ve çift yönlü haberleşmeyi gerçekleştirebilmek için çeşitli paralel port modları oluşturulmuştur. IEEE-1284'te aşağıdaki beş adet paralel port modu tanımlanmıştır.

6.3.1. Compatibility Mode (Uyumluluk Modu)

Compatibility mod bir ileri kanal modudur, bu modda data yalnızca PC den çevresel birim aygıtına doğru taşınır. Bu modda birim zamanda 8 bitlik (1 byte) data taşınır. Bu mod paralel portların default(varsayılan) modudur. IEEE-1284'e göre bir paralel port en azından compatibility mode işlemlerini gerçekleştirebilmelidir.

6.3.2. Nibble Mode (Ters Mod)

Nibble mod bir ters kanal modudur, bu modda data yalnızca çevresel birim aygıtından PC'ye doğru taşınır. Bu modda PC'ye birim zamanda bir "nibble" yani 4 bitlik bir veri taşınır. Nibble'lar PC'ye ulaştığı zaman byte formatına çevrilmesi gerekir bu işlem veri aktarım hızında yavaşlamalara neden olur. IEEE uyumluluğu için bir paralel port en azından nibble mod ve compatibility mod özelliklerini sağlamalıdır.

6.3.3. Byte Mode

Byte mod bir diğer ters kanal modudur. Bu mod Nibble mode üzerine geliştirilmiştir ve birim zamanda PC'ye tam 1 Byte'lık veri taşınmasına olanak sağlar.

6.3.4. Enhanced Parallel Port (EPP) Mode (Yükseltilmiş Mod)

EPP modu ileri ve ters yönlü çalışabilir ve her iki yöne de tam 1 byte'lık veri taşır. EPP modun Compatibility, Nibble ve Byte Mode'a göre en büyük üstünlüğü hızıdır. Tek bir ISA bus cycle'ında el sıkışma protokolü için gerekli olan sürede dahil olmak üzere 1 byte'lık veriyi okur veya yazar. Daha önceki modlarda 1 byte'lık veri transferi için 4 cycle gerekiyordu. EPP mode aynı zamanda data iletim yönünü çok hızlı değiştirebildiği için paralel porttan çalışan veri depolama cihazları için çok iyi bir seçimdir. EPP mode içerisinde gerektiğinde kullanılabilecek compatibility, nibble ve byte mode'larını barındırır.

6.3.5. Extended Capabilities Port (ECP) Mode (Yetenekleri Artırılmış Mod)

ECP Mode paralel veri transferine başka bir gelişmişlik ekler. Bu modda EPP gibi ISA bus cycle'ında 1 byte'lık veriyi okur veya yazar. Bu modun artısı tamponlara (buffer) sahip olup, DMA (belleğe doğrudan erişim) ve veri sıkıştırmayı sağlamasıdır. Bu mod diğer bütün modları içerisinde barındırır.

Byte, EPP ve ECP modlarının her biri yazılım ve donanım desteği gerektirir dolayısıyla uygulamaları diğer iki temel mod kadar kolay değildir. Diğer taraftan gelişmiş modların en büyük avantajları ise hızlarıdır.

6.4. Paralel Port Tipleri

Orijinal PC paralel portu compatibility ve nibble modda çalışır. Günümüzde bu tip paralel portlara Standart paralel port veya SPP denir. Bir sonraki paralel port versiyonu ise IBM PS/2 ile geldi. Bu portta ilave olarak byte mod kullanıldı ve "Simple bi-directional (basit çift yönlü)" veya "PS/2-type (PS/2-tip)" olarak adlandırıldı. Enhanced Parallel Port (EPP) ve Extended Capabilities Port (ECP) lar ise isimlerinden de anlaşılacağı gibi aynı ada sahip paralel port modlarını destekler.

6.5. Paralel Port Adresleri

Paralel port adresleri sayısal sıralamaya göre atanırlar. Eğer bir sistemde birden fazla paralel porttan bahsedecek olursak bu portların her biri "LPT" ile başlayan geleneksel isimlerini alarak LPT1, LPT2 vs. olarak isimlendirilirler.

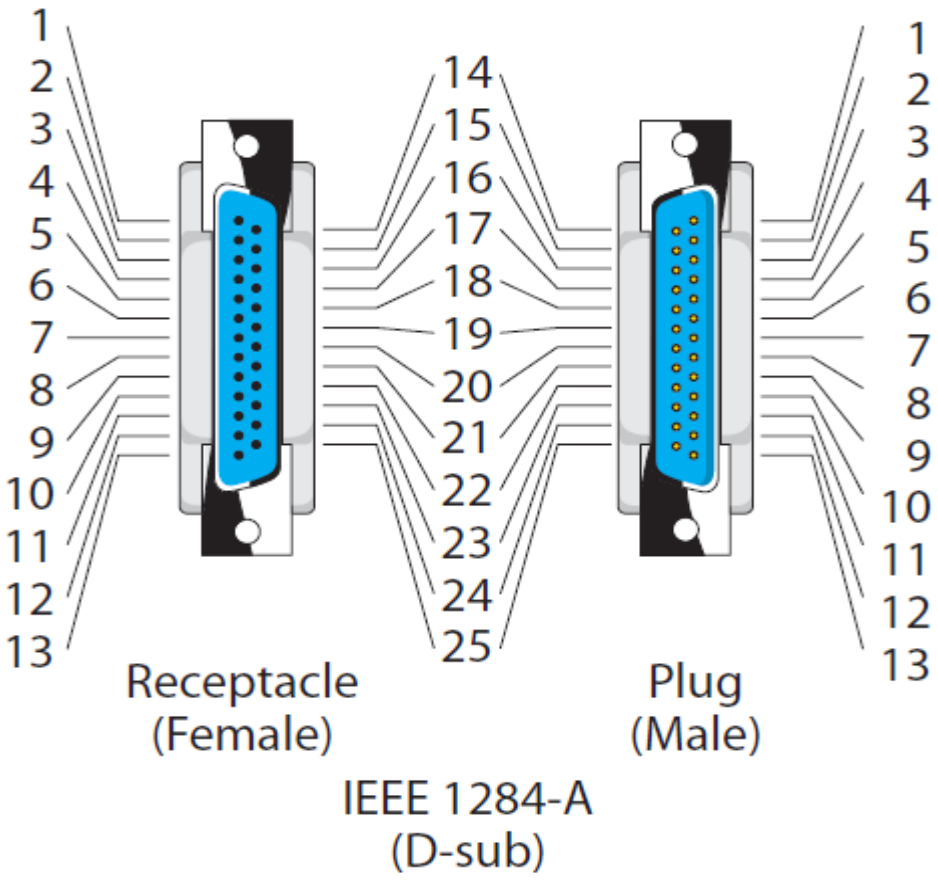
Standart paralel port ardışık devam eden 3 adet adres kullanır. Bunların ilki data portuna, ikincisi durum portuna, üçüncüsü ise kontrol portuna aittir. Aşağıda Tablo 6.1.'de standart paralel porta ait adresler görülmektedir.

Tablo 6.1. Standart Paralel Port Adresleri

	Veri Portu (Data)	**Durum Portu (Status)**	**Kontrol Portu (Control)**
LPT1	3BCh	3BDh	3BEh
LPT2	378h	379h	37Ah
LPT3	278h	279h	27Ah

6.6. Paralel Port Konnektörleri

IEEE 1284 PC'lerde ve çevresel birimlerde görülen üç tip konnektör ve bunlara ait pin çıkışlarını listeler. Geleneksel D-sub 25 pin konnektör IEEE 1284-A olarak isimlendirilir. Geleneksel Centronics konnektör ise IEEE 1284-B olarak isimlendirilir. Son olarak en küçük ve yeni olan Centronics'e benzeyen konektör ise IEEE 1284-C olarak isimlendirilir. Aşağıda Şekil 6.2.'de IEEE 1284-A'nın yapısı görülmektedir. Tablo 6.2.'de ise IEEE 1284-A nın pin atamaları görülmektedir.



Şekil 6.2. IEEE 1284-A Paralel Port Şekli [27]

Tablo 6.2. IEEE 1284-A Pin Atamaları [27]

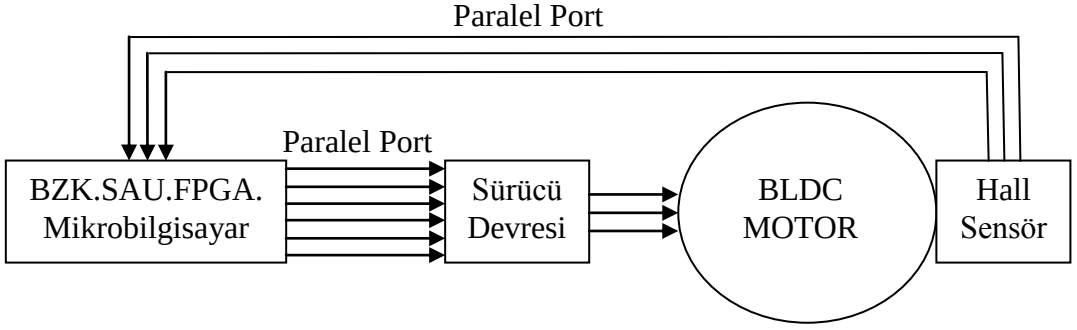
Pin	**Compatibility Mode**	**Nibble/Byte Mode**	**ECP Mode**	**EPP Mode**	**Signal Function**	**Signal Source**
1	nStrobe (L)	HostClk (V)	HostClk (C)	nWrite (L)	Control	Host
2	Data 1 (V)[b]	Data 1 (V)[b]	Data 1 (V)[b]	AD1 (V)[b]	Data	Either[c]
3	Data 2 (V)[b]	Data 2 (V)[b]	Data 2 (V)[b]	AD2 (V)[b]	Data	Either[c]
4	Data 3 (V)[b]	Data 3 (V)[b]	Data 3 (V)[b]	AD3 (V)[b]	Data	Either[c]
5	Data 4 (V)[b]	Data 4 (V)[b]	Data 4 (V)[b]	AD4 (V)[b]	Data	Either[c]
6	Data 5 (V)[b]	Data 5 (V)[b]	Data 5 (V)[b]	AD5 (V)[b]	Data	Either[c]
7	Data 6 (V)[b]	Data 6 (V)[b]	Data 6 (V)[b]	AD6 (V)[b]	Data	Either[c]
8	Data 7 (V)[b]	Data 7 (V)[b]	Data 7 (V)[b]	AD7 (V)[b]	Data	Either[c]
9	Data 8 (V)[b]	Data 8 (V)[b]	Data 8 (V)[b]	AD8 (V)[b]	Data	Either[c]
10	nAck (L)	PtrClk (V)	PeriphClk (C)	Intr (H)	Control	Peripheral
11	Busy (H)	PtrBusy (V)	PeriphAck (V)	nWait (L)	Control	Peripheral
12	PError (H)	AckData (V)	nAckReverse (L)	Userdefined1 (M)	Control	Peripheral
13	Select (H)	Xflag (V)	Xflag (V)	Userdefined3 (M)	Control	Peripheral
14	nAutoFd (L or V[d])	HostBusy(V)	HostAck (V or C[e])	nDStrb (L)	Control	Host
15	nFault (L)	nDataAvail (V)	nPeriphRequest (L)	Userdefined2 (M)	Control	Peripheral
16	nInit (L)	nInit (H)	nInit (L)	nReverseRequest (L)	Control	Host
17	nSelectIn (L)	IEEE1284 Active (V)	IEEE1284 Active (V)	nAStrb (L)	Control	Host
18	Signal ground for Pin 1[f]				Ground	
19	Signal ground for Pin 2 and 3[f]				Ground	
20	Signal ground for Pin 4 and 5[f]				Ground	
21	Signal ground for Pin 6 and 7[f]				Ground	
22	Signal ground for Pin 8 and 9[f]				Ground	
23	Signal ground for Pin 11 and 15[f]				Ground	
24	Signal ground for Pin 10,12 and 13[f]				Ground	
25	Signal ground for Pin 14,16 and 17[f]				Ground	

a. Sinyalin aktif olduğu durum parantez içerisinde gösterilmiştir: L=low, H=high,V= çeşitli low veya high, C= closed loop (kapalı çevrim) veya handshaking (el sıkışma), M=manufacturer-specific (üreticiye özel)
b. Data bitlerinden 1 en düşük anlamlı 8 ise en yüksek anlamlı bittir.
c. "Either" data sinyallerinin çift yönlü olduğunu gösterir.
d. Farklı çevre birimleri tarafından farklı yorumlanır.
e. İleri yönde bu bit bilginin komutmu verimi olduğunu gösterir. Ters yönde ise PeriphClk (Pin 10) ile birlikte handshake için kullanılır.
f. Ground pinlerinin bütün modlarda isimleri aynıdır.

7. PARALEL PORT ARABİRİMİ, SÜRÜCÜ DEVRE ve KONTROLÖR TASARIMI

Bu bölümde FPGA tabanlı mikrobilgisayar mimarisi (BZK.SAU.FPGA) kullanılarak tasarlanan eğitimsel amaçlı bir fırçasız DC motor (BLDC) sürücü uygulaması anlatılacaktır. Gerçekleştirilen sürücü sistemin blok şeması Şekil 7.1.'de gösterilmiştir. Motor kontrol işlemi BLDC motorun hall sensöründen okunan 3-bitlik pozisyon bilgisine göre, enfazla torku üretecek şekilde motoru döndürmek için gerekli sargı uçları enerjilendirilerek gerçekleştirilmiştir. Motor için gerekli sargı gerilimleri üreten sürücü devresi 3 faz inverter köprüsü şeklinde mosfetlerle tasarlanmıştır. Hall sensörden pozisyon bilgisini alarak değerlendiren mikrobilgisayar sürücü devreye gerekli gerilimleri üretebilmesi için tetikleme sinyallerini gönderir. Mikrobilgisayarın pozisyon sensörü ve sürücü devresi ile haberleşmesi mikrobilgisayar için tasarlanan paralel port arabirimi ile gerçekleştirilmiştir.

Bu bölümün ilk kısmında BZK.SAU.FPGA'in çevresel birimlerle haberleşmesini sağlayacak olan paralel port arabiriminin tasarımı anlatılacaktır. İkinci kısımda fırçasız DC motor için tasarlanan sürücü devre anlatılacaktır. Üçüncü kısımda ise BZK.SAU.FPGA. komut setiyle yazılmış olan motor sürme programı anlatılıp son kısımda alternatif olarak tamamen lojik seviyede tasarlanan donanımsal bir kontrolör tasarımı anlatılacaktır.



Şekil 7.1. Gerçekleştirilen Fırçasız DC Motor Sürücü Sistemi Blok Şeması

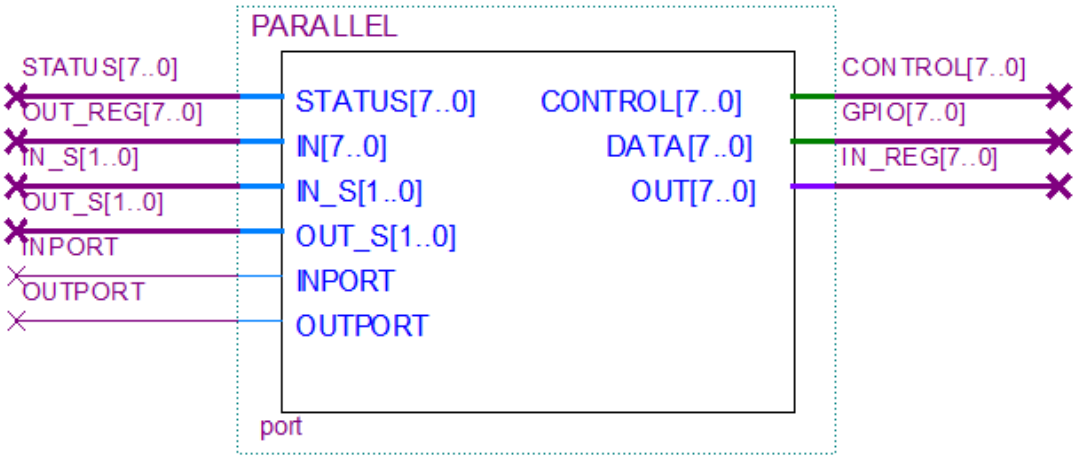
7.1. Paralel Port Arabirimi

Bu çalışmada fırçasız DC motorun hall sensörünün ürettiği veri ve sürücü devresine gönderilmesi gereken veri paralel yapıda olduğundan, kontrol arabirimi olarak paralel port tercih edildi. Tasarımda belirli bir standart sağlanabilmesi için 6. bölümde anlatılan standart paralel port yapısına uygun bir tasarım gerçekleştirildi.

Paralel port, bilgisayar kasasının arkasında bulunan 25 pinden oluşan D şeklindeki konnektördür. Orijinalinde bir yazıcı bağlantı noktası olarak tasarlanan paralel port yazıcılara veri göndermekten başka harici veri depolama cihazları, çift yönlü veri alma gönderme ve kontrol uygulamaları için de kullanılır. Standart bir paralel port data, status ve control portlarından meydana gelir. Bunlardan data portu 8 bit okuma/yazma, status portu 5 bit okuma, control portu ise 4 bit okuma/yazma işlemlerine izin verir. Tasarımda status ve control portları 8'er bit gözükse de status portunun 6.,7. ve 8. bitleri ile control portunun 5., 6., 7., ve 8. bitlerinin çevresel birimlerle bağlantısı olmayıp donanımsal olarak lojik 0'a çekilmiştir.

BZK.SAU.FPGA çevresel birimlere erişim için tasarlanmış 8-bitlik iki adet kaydediciye (register) sahiptir. Bunlardan giriş kaydedicisi (input register) çevresel birimlerden veri almak için, çıkış kaydedicisi (output register) çevresel birimlere veri göndermek için kullanılır. Bu kaydedicilere erişimi sağlayan mikrobilgisayar komutları ise IN ve OUT komutlarıdır. IN komutu input register'daki 8-bitlik veriyi alarak 16 bitlik akümülatörün en düşük anlamlı 8 bitine yazar. OUT komutu ise akümülatördeki 16 bitlik verinin en düşük anlamlı 8 bitini output register'a yazar.

Bu çalışmada tasarlanan arabirim çevresel birimlerden gelen verinin IN komutuyla birlikte input register'ına ulaşmasını ve akümülatöre yazılmasını, çevresel birimlere gönderilecek olan verinin ise OUT komutuyla birlikte akümülatörden alınıp output register üzerinden çıkış pinlerine aktarılmasını sağlar. Tasarlanan arabirimin blok hali Şekil 7.2.' de gösterilmektedir. Tablo 7.1.'de ise tasarlanan arabirimin bağlantı pinlerinin açıklaması verilmektedir.



Şekil 7.2. Paralel Port Modülü

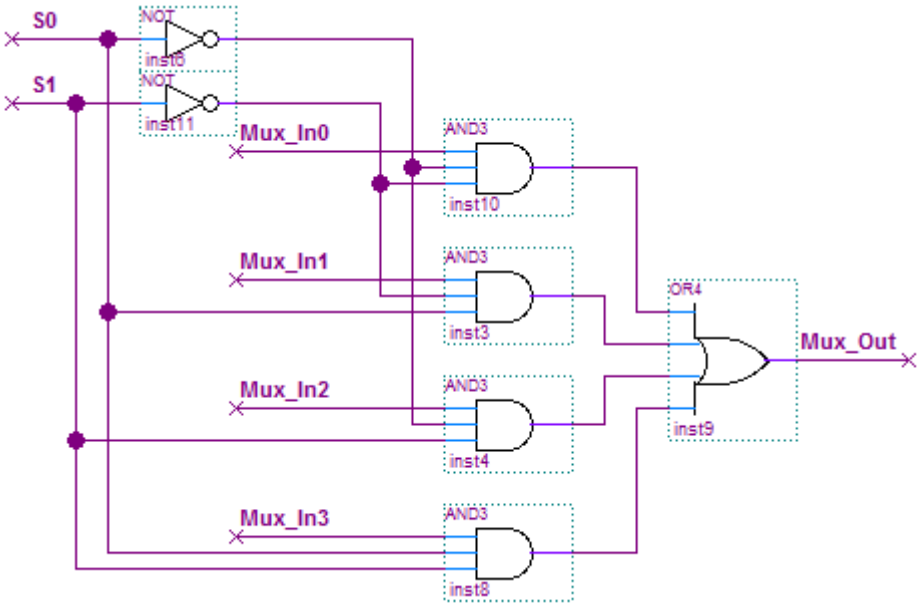
Tablo 7.1. Paralel Port Modülü Pin Tanımlamaları

Pin Adı	**Tipi**	**Açıklaması**
STATUS[7..0]	Input	Status portundan gelen veri
OUT_REG[7..0]	Input	Output registerdan gelen veri
IN_S[1..0]	Input	Hangi porttan okuma yapılacağı seçilir. “00” ise Data portu, “01” ise status portu, “10” ise control portu okunur
OUT_S[1..0]	Input	Hangi porta yazılacağı seçilir. “00” ise Data portuna, “10” ise control portuna yazılır
INPORT	Input	Portun okuma yapmasını sağlayan giriş
OUTPORT	Input	Portun yazma yapmasını sağlayan giriş
CONTROL[7..0]	In/Out	Control portundan gelen/giden veri
GPIO[7..0]	In/Out	Data portundan gelen/giden veri
IN_REG[7..0]	Out	Input register’a giden veri

Burada çevresel birimlerden okuma yapabilmek için IN_S[1] ve IN_S[0] girişleri ile hangi porttan okuma yapılacağı seçilip INPORT girişine lojik 1 verisi uygulanır. Yazma işlemi için ise OUT_S[1] ve OUT_S[0] girişleri ile hangi porta yazma yapılacağı seçilip OUTPORT girişine lojik 1 verisi uygulanır. Burada INPORT ve OUTPORT girişlerine IN ve OUT komutlarının son adımında lojik 1 verisi gönderilerek porttaki veri alınır veya porta veri gönderilir.

Tasarlanan Paralel port modülü 8 adet 4x1 Mux ve 8 adet 1x4 DeMux'tan oluşmaktadır. Modülün çalışma şeklini anlamak için 4x1 Mux ve 1x4 Demux'tan kısaca bahsedelim.

Multiplexer'lar (Çoklayıcı) birden fazla sayısal veri kaynağından birini seçerek o kaynağı çıktı olarak tek bir kanala ileten sistemlerdirler [28]. S0 ve S1 seçim uçlarına uygulanacak veriye göre giriş kanallarından bir tanesi çıkışa aktarılır. Şekil 7.3. bir 4x1 Mux'un yapısını, Tablo 7.2. ise doğruluk tablosunu göstermektedir.

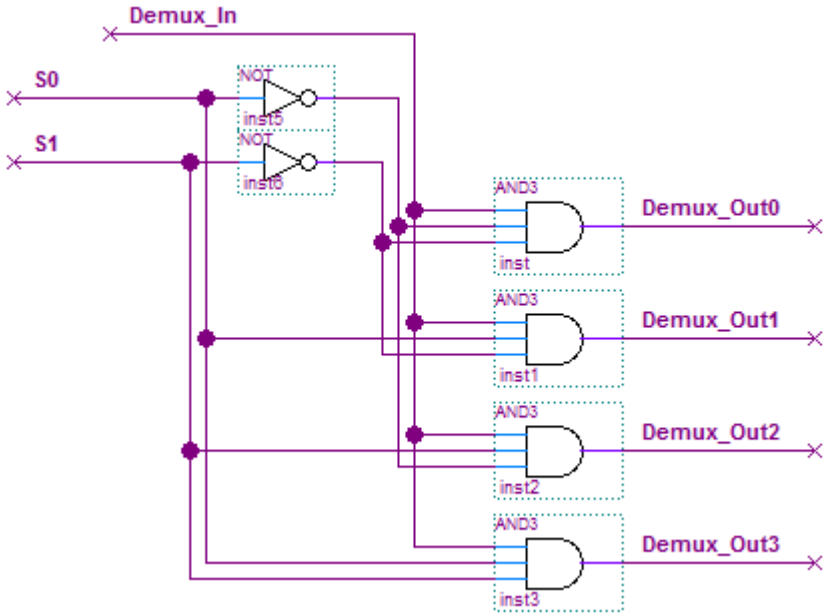


Şekil 7.3. 4x1 Multiplexer

Tablo 7.2. 4x1 Multiplexer Doğruluk Tablosu

S1	**S0**	**Mux_Out**
0	0	Mux_In0
0	1	Mux_In1
1	0	Mux_In2
1	1	Mux_In3

Demultiplexer'lar (Tekleyici) ise multiplexer'ların tam tersi şeklinde çalışırlar. Tek bir sayısal veri kaynağını birden fazla olan çıkış kanallarından birini seçerek iletebilen sistemlerdir.

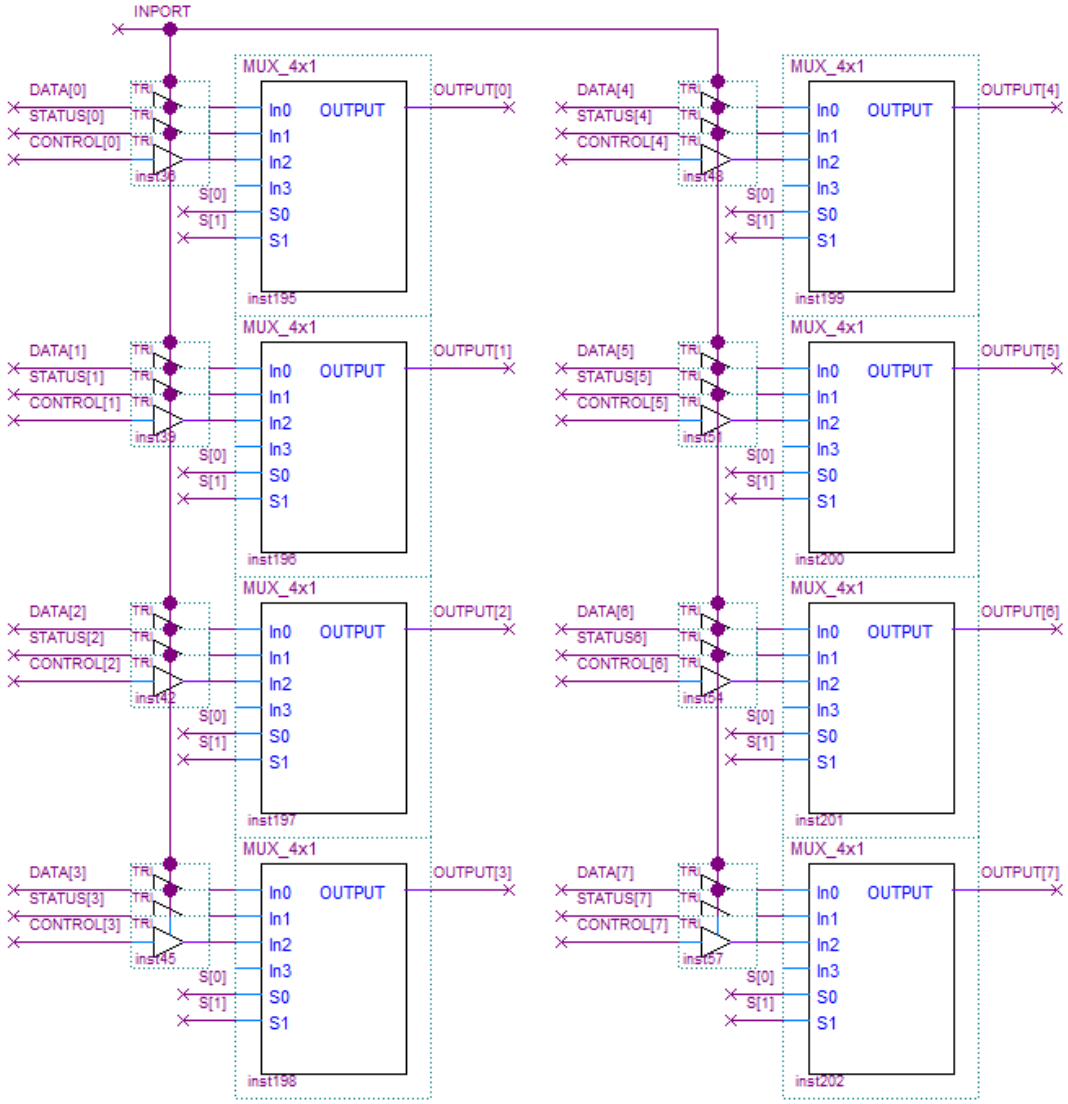


Şekil 7.4. 1x4 DeMultiplexer

Tablo 7.3. 1x4 DeMultiplexer Doğruluk Tablosu

S1	S0	Demux_Out0	Demux_Out1	Demux_Out2	Demux_Out3
0	0	Demux_In	0	0	0
0	1	0	Demux_In	0	0
1	0	0	0	Demux_In	0
1	1	0	0	0	Demux_In

Tasarımımızda multiplexer'lar okuma yapacağımızı portu seçmemizi sağlarken, demultiplexer'lar veri yazacağımız portu seçmemizi sağlarlar. Multiplexer'in seçim uçlarına göre data, status veya control portlarından birindeki veri input register'a yönlendirilirken, demultiplexer'ın seçim uçlarına göre ise output registerdaki veri data veya control portuna yönlendirilir. 8 bitlik bir veri ise seçim uçları birleştirilmiş 8 adet 4x1 mux ve 8 adet 1x4 demux ile transfer edilebilir.

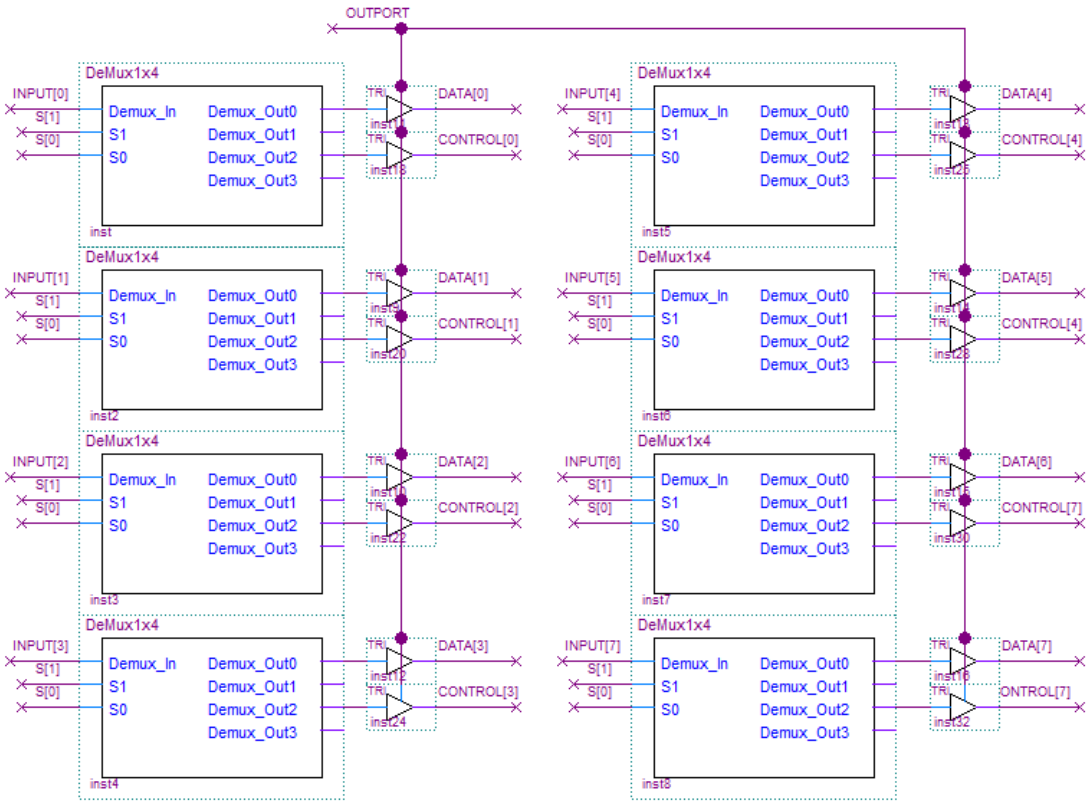


Şekil 7.5. 8x4x1 Multiplexer

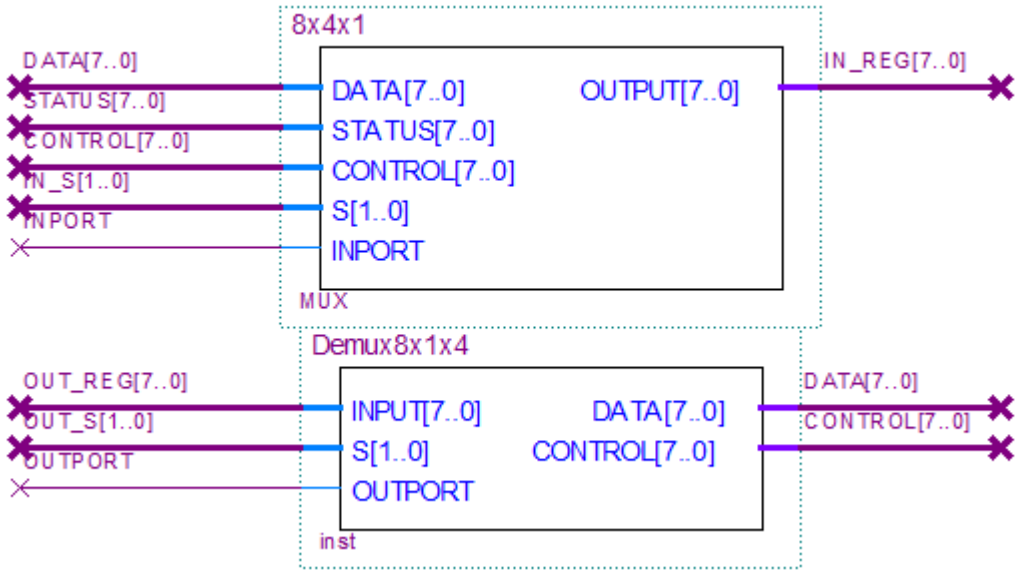
Şekil 7.5.'de görülen 8x4x1 multiplexer ile seçim uçlarına bağlı olarak data, status veya control portlarının herhangi birindeki 8 bitlik veri okunabilir. Şekil 7.6.'da ki 8x1x4 demultiplexer ile de yine seçim uçlarına bağlı olarak data veya control portundan hehangi birine 8 bitlik veri gönderilebilir.

Multiplexer'ların girişinde ve demultiplexer'ların çıkışlarında kullanılan tri-state buffer'lar ise Data ve Control portları hem giriş hem çıkış portları olduğu için bu noktalarda meydana gelecek çakışmayı önlemek amacıyla konulmuştur. Input komutu süresince multiplexer'ların girişindeki buffer'lar iletimdedir ve portlar input konumundadır. Out komutunda ise multiplexer'ların önündeki buffer'lar tıkanarak

yüksek empedans durumuna geçerken, demultiplexer'ların çıkışındaki bufferler ise iletime geçerek portları çıkış konumuna geçirirler.

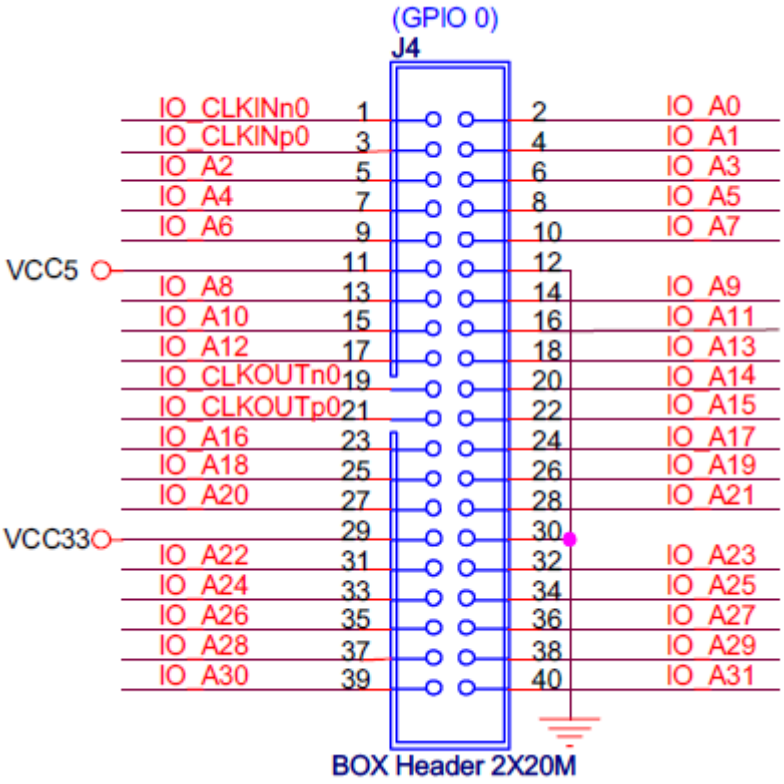


Şekil 7.6. 8x1x4 DeMultiplexer



Şekil 7.7. Paralel Port Modülü İç Yapısı

Son olarak tasarlanan paralel port modülünün Status, Control ve Data pinleri Altera DE2-70 Board'un çevresel arabirimi olan 40 pinlik GPIO portuna bağlanarak paralel port tasarım işlemi gerçekleştirilmiş oldu.



Şekil 7.8. DE2-70 Board GPIO 0 Portu [29]

Tablo 7.4. Paralel Port Pin Bağlantıları

Sinyal İsmi	**GPIO Bağlantısı**	**Sinyal İsmi**	**GPIO Bağlantısı**
DATA[0]	I0_A0	STATUS[0]	I0_A14
DATA[1]	I0_A1	STATUS[1]	I0_A15
DATA[2]	I0_A3	STATUS[2]	I0_A17
DATA[3]	I0_A5	STATUS[3]	I0_A19
DATA[4]	I0_A7	STATUS[4]	I0_A21
DATA[5]	I0_A9	CONTROL[0]	I0_A23
DATA[6]	I0_A11	CONTROL[1]	I0_A25
DATA[7]	I0_A13	CONTROL[2]	I0_A27
		CONTROL[3]	I0_A29

7.2. Sürücü Devre

Sürücü devre fırçasız DC motor için gerekli elektriksel gücü sağlayan devredir. 5. Bölümde fırçasız DC motor sürmek için gerekli temel bilgiler verilmiştir. Bu çalışmada rotor pozisyon bilgisine göre motor kontrolü yapılmıştır. Bu metotta rotorun pozisyon bilgisi motor içerisindeki sensör vasıtasıyla belirlenerek motorun en fazla torku üretecek şekilde dönmesini sağlayacak sargı uçları enerjilendirilir. Sensörden alınan pozisyon bilgisine karşılık motorun hangi sargı uçlarının enerjilendirileceği ise kullanılan motor kataloglarından bulunabilir. Bu çalışmada Maxon firmasının üretmiş olduğu 8 watt gücünde 24 volt nominal gerilimle çalışan EC-max 16 modeli fırçasız DC motor kullanılmıştır. İlgili motoru sürebilmek için motorun hall sensöründen alınan bilgiye göre motor sargı uçlarına uygulanması gereken gerilimler Tablo 7.5.'de gösterilmiştir. Bu tablo maxon motor katalogundan alınmıştır.

Tablo 7.5. Hall Sensör İçin Sinyal Diyagramı [30]

Block Commutation							
Signal Sequence Diagram For The Hall Sensors							
Conductive Phases		**I**	**II**	**III**	**IV**	**V**	**VI**
Rotor Position		60	120	180	240	300	360
Hall Sensor1	1	■	■	■			
	0				■	■	■
Hall Sensor2	1			■	■	■	
	0	■	■				■
Hall Sensor3	1	■				■	■
	0		■	■	■		
Supplied Motor Voltage (phase to phase)							
U1-2	+	■					
	-				■	■	■
U2-3	+			■			
	-						■
U3-1	+					■	
	-		■				

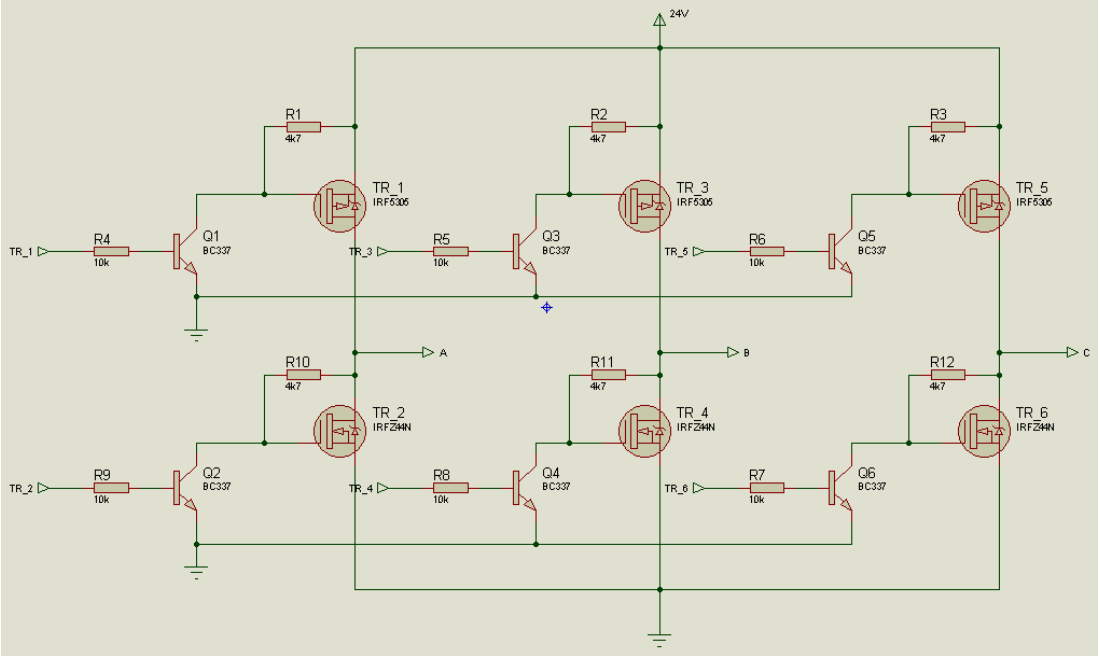
Tablo 7.5.'i kullanarak hall sensör bilgisine göre motor sargı uçlarına uygulamamız gereken gerilimleri tespit edebiliriz. Örnek olarak hall sensörlerden sırasıyla 1-0-1 bilgisini aldığımızda 1. adımda olduğumuzu anlarız ve 24 voltluk kaynağın pozitif çıkışını (+) 1 nolu sargı ucuna negatif çıkışını (-) 2 nolu sargı ucuna yönlendiririz. Sensörlerden 1-0-0 bilgisini aldığımızda ise 2. adımda olduğumuzu anlarız ve 3 nolu sargı ucuna negatif (-) voltaj, 1 nolu sargı ucuna pozitif (+) voltaj göndeririz.

Motor sargı uçlarına uygulayacağımız voltajı belirlediğimize göre bu voltajları sağlayacak devrenin tasarımına geçebiliriz. 5. Bölümde anlatılan 3 faz köprüsü şeklindeki devre yapısı 24 voltuk DC gerilimi anahtarlayarak istediğimiz sargı uçlarına yönlendirebilmemizi sağlar. Bu çalışmada fırçasız DC motor sürücü devre olarak Şekil 5.3.'de prensip olarak gösterilen üç faz köprüsü kullanıldı. Devrede güç anahtarlama elemanı olarak mosfetler kullanıldı. Pozitif voltaj tarafı için 3 adet P-kanallı (TR_1, TR_3, TR_5) mosfet kullanılırken, negatif voltaj tarafı için ise 3 adet N-kanallı (TR_2, TR_4, TR_6) mosfet kullanıldı. FPGA çıkışlarının sağladığı 3.3 volt mosfetleri sürmede yetersiz kalacağından, FPGA çıkışlarını yükselterek mosfetleri sürmek için devrede BC337 NPN transistörler kullanıldı.

Tasarlanan devre Şekil 7.9.'da görülmektedir. Devre üzerindeki A, B, C terminalleri sırasıyla motorun 1, 2 ve 3 nolu sargı uçlarına bağlanır, TR_1, TR_2, …,TR_6 bağlantı noktalarına mikrobilgisayar tarafından veri gönderilerek kontrol işlemi gerçekleştirilir.

Devrenin çalışma şekli şu şekildedir: Tablo 7.6.'da ki 1. adım için gerekli sargı gerilimini (U1-2 = +24 volt) motora uygulamak için devremizde TR_1 ve TR_4 mosfetlerinin iletimde diğer mosfetlerin ise kesimde olması gereklidir. P kanallı TR_1 mosfetinin iletime geçmesi için Q1 transistörünün iletime geçmesi gerekmektedir. Q1 transistörünün girişine lojik 1 (3.3 volt) gönderilmesi ile Q1 transistörü iletime geçerek TR_1 mosfetinin girişini 0 volta çekerek mosfeti iletime geçirir. N kanallı olan TR_4 mosfetinin iletime geçmesi için Q4 transistörünün kesimde olması gerekmektedir. Q4 transistörünün girişine lojik 0 (0 volt) gönderilmesi ile Q4 transistörü kesimde kalarak TR_4 mosfetinin iletimde kalmasını

sağlar. Q4 transistörünün iletime geçmesi ile de TR_4 transistörünün girişine 0 volt uygulanarak transistör kesime gider. Tablo 7.6.'da Tablo 7.5. ve sürücü devre referans alınarak hall sensör bilgisine göre iletimde olması gereken transistörler ve transistör girişlerine uygulanması gereken sinyaller verilmiştir.



Şekil 7.9. Sürücü Devre Şeması

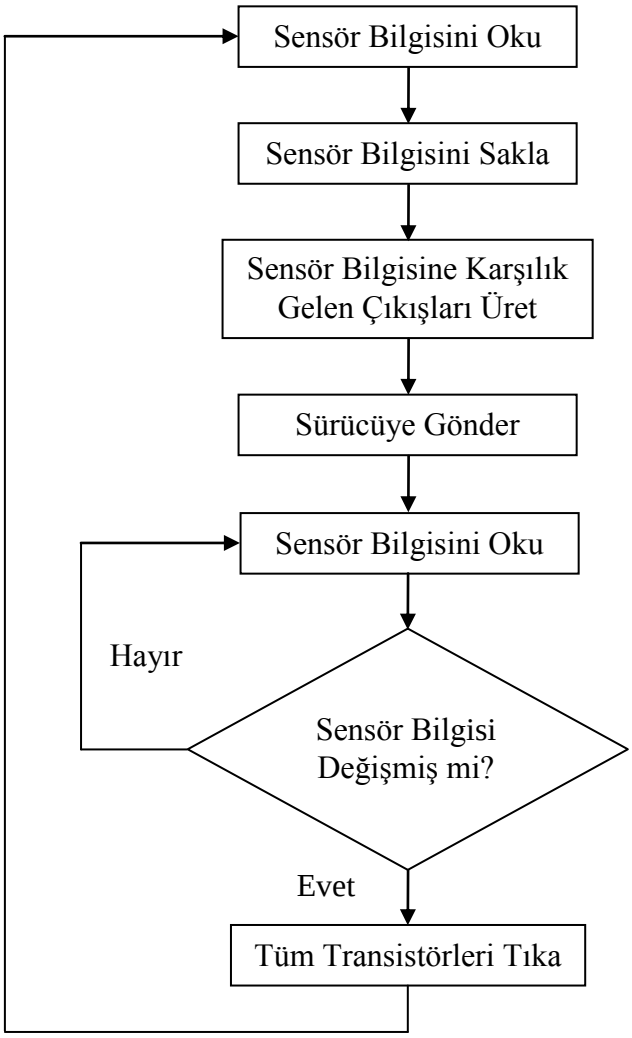
Tablo 7.6. Sensör Verisine Göre Transistör Gerilimleri

	Hall 1-2-3	Enerjilenecek Sargı Uçları	İletimdeki Transistörler	TR_1	TR_3	TR_5	TR_2	TR_4	TR_6
1	101	1-2	TR_1-TR-4	1	0	0	1	0	1
2	100	1-3	TR_1-TR-6	1	0	0	1	1	0
3	110	2-3	TR_3-TR-6	0	1	0	1	1	0
4	010	2-1	TR_3-TR-2	0	1	0	0	1	1
5	011	3-1	TR_5-TR-2	0	0	1	0	1	1
6	001	3-2	TR_5-TR-4	0	0	1	1	0	1

7.3. BZK.SAU.FPGA Motor Kontrol Kodu

Bu kısımda BZK.SAU.FPGA için fırçasız DC motor kontrol uygulamasını gerçekleştirecek olan program kodu yazımı anlatılacaktır. Program kodu hall sensörü ile motordan alınan pozisyon bilgisine göre motor sargı uçlarına gerekli gerilimi uygulamak için sürücü devreye gönderilecek veriyi üretir.

Program kodu yazmaya başlamadan önce bir akış şeması oluşturmak, işin tamamlanması için gerekli adımların çok daha iyi anlaşılmasını sağlar. Durum analizi veya hata arama gibi durumlarda da akış şeması işlerin kolaylaşmasını sağlar. Şekil 7.10.'da fırçasız DC motor kontrolü için oluşturulmuş akış şeması görülmektedir.



Şekil 7.10. Akış Şeması

Programımız temel olarak alınan sensör bilgisine karşılık transistör bloğuna uygulanması gereken veriyi üretir. İlk olarak hall sensörden pozisyon bilgisi okunur. Alınan bilgi bellekte bir adreste saklanır. Pozisyon bilgisine göre transistörleri sürecek veri oluşturularak sürücüye gönderilir. İlgili çıkışlar sürücüye uygulandıktan sonra sensör bilgisi tekrar okunarak bellekte saklanan bilgi ile karşılaştırılır. Sensör bilgisi değişmemiş ise bir sonraki pozisyona kadar mevcut transistör grubu

sürülmeye devam eder. Sensör bilgisi değiştiğinde ise motorun bir sonraki adıma geçtiği anlaşılır. Mevcut transistör grubu tıkanarak bir sonraki adımın transistörleri sürülür.

Transistörlerin kesime gitme süreleri iletime geçme sürelerinden daha uzun olduğu için fazlar arasındaki geçişlerde ölü zaman gecikmesi tabir edilen yeterli bir gecikme süresine ihtiyaç vardır. Program sensördeki değişimi fark ettiği anda yeni transistör grubunu sürmeden tüm transistörlere tıkama komutu gönderir ardından sıradaki transistör grubu sürülür. Bu sırada geçen süre ölü zaman gecikmesi için kullanılır. Ölü zaman gecikmesi uygulanmaz ise sarsıntılı bir sürüşün yanı sıra motor sargılarında ve transistörlerde ısınmalar meydana gelir.

Program kodu yazımında veriler hexadecimal olarak girileceği için yazılıma geçmeden önce sensör bilgisinin ve sürücüye gönderilecek verinin hexadecimal karşılıklarının belirlenmesi gerekmektedir. Tablo 7.6. referans alınarak hazırlanan sensörden okunan bilginin hexadecimal karşılığı ile transistörlere gönderilecek verinin hexadecimal karşılığı Tablo 7.7.'de verilmiştir.

Tablo 7.7. Sensör Bilgisi ve Gönderilecek Kod'un Hexadecimal Karşılıkları

Sensör Bilgisi	**Hex Karşılığı**	**Gönderilecek Kod**	**Hex Karşılığı**
101	0005h	100101	0025h
100	0004h	100110	0026h
110	0006h	010110	0016h
010	0002h	010011	0013h
011	0003h	001011	000Bh
001	0001h	001101	000Dh

Program kodu yazımında BZK.SAU.FPGA.'in 59 komuttan oluşan komut setinden 7 adet komut kullanılmıştır. Bu komutlar ve kullanım şekilleri aşağıda anlatılmıştır.

IN Komutu: IN komutu paralel porttan okuma yapan komuttur. Bu çalışmada IN komutu ile alınacak veri paralel portun status portundan alınacağı için, paralel port

modülünün IN_S[1] ucu lojik 1'e, IN_S[0] ucu ise lojik 0'a donanımsal olarak çekilmiştir. Bu şekilde IN komutu ile birlikte paralel portun status portunadaki veri alınarak akümülatöre aktarılır.

OUT Komutu: OUT komutu paralel porta veri gönderen komuttur. Bu çalışmada OUT komutu ile yazılacak veri paralel portun data portuna yazılacağı için, paralel port modülünün OUT_S[1] ve OUT_S[0] uçları donanımsal olarak lojik 0'a çekilmiştir. Böylece OUT komutu ile birlikte akümülatördeki veri, paralel portun data portuna aktarılır.

STA Komutu: Akümültardeki veriyi alarak istenilen adrese yazar. STA $FE00H komut satırı akümülatördeki veriyi FE00H adresine yazar.

LDA Komutu: İstenilen adresteki veriyi akümülatöre yazar. LDA $FE00H komut satırı FE00H adresindeki veriyi akümülatöre yazar.

CMP Komutu: Karşılaştırma komutudur. Akümülatördeki veri ile istenilen veriyi karşılaştırır. CMP #0005H komut satırı akümülatördeki veri ile 0005H verisini karşılaştırır.

BZR Komutu: CMP komutu ile yapılan karşılaştırmanın sonucuna göre dallanma yapar. BZR S0 komut satırı karşılaştırmanın sonucunda veriler aynı değerde ise S0 etiketine programı dallandırır. Değilse bir alt satırdan program akışı devam eder.

BRA Komutu: Koşulsuz dallanma yapar. BRA A komut satırına gelindiğinde program A etiketine dallanır.

Kullanılan komutların açıklamaları da verildikten sonra yazılan program kodu ve açıklaması aşağıda Tablo 7.8.'de verilmiştir.

Tablo 7.8. Program Kodu

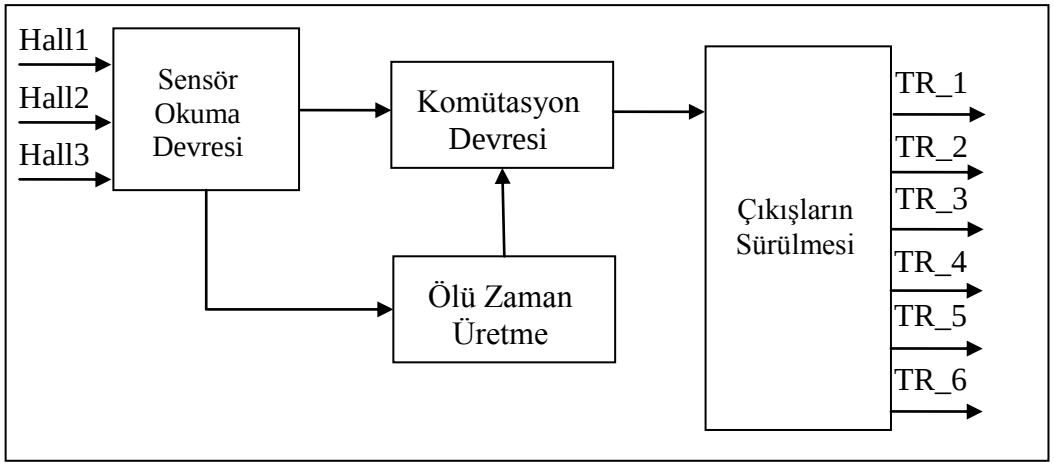
1	A	IN	Portu Oku
2		STA $FE00H	FE00H Adresine Sakla
3		LDA $FE00H	FE00H Adresinden Oku
4		CMP #0005H	5H ise
5		BZR S0	S0 Etiketine Dallan
6		LDA $FE00H	
7		CMP #0004H	4H ise
8		BZR S1	S1 Etiketine Dallan
9		LDA $FE00H	
10		CMP #0006H	6H ise
11		BZR S2	S2 Etiketine Dallan
12		LDA $FE00H	
13		CMP #0002H	2H ise
14		BZR S3	S3 Etiketine Dallan
15		LDA $FE00H	
16		CMP #0003H	3H ise
17		BZR S4	S4 Etiketine Dallan
18		LDA $FE00H	
19		CMP #0001H	1H ise
20		BZR S5	S5 Etiketine Dallan
21	S0	LDA #0025H	Akümülatöre 25H yükle
22		OUT	Porta Gönder
23	Y0	IN	Portu Oku
24		CMP #FE00H	FE00H deki veri ile karşılaştır
25		BZR Y0	Aynı ise Y0 etiketine git (sensör değişmemiş ise)
26		LDA #0007H	Farklı ise Akümülatöre 7H yükle (Tıkama)
27		OUT	Porta Gönder
28		BRA A	A etiketine Dallan
29	S1	LDA #0026H	Akümülatöre 26H yükle
30		OUT	Porta Gönder

31	Y1	IN	Portu Oku
32		CMP #FE00H	FE00H deki veri ile karşılaştır
33		BZR Y1	Aynı ise Y1 etiketine git (sensör değişmemiş ise)
34		LDA #0007H	Farklı ise Akümülatöre 7H yükle (Tıkama)
35		OUT	Porta Gönder
36		BRA A	A etiketine Dallan
37	S2	LDA #0016H	Akümülatöre 16H yükle
38		OUT	Porta Gönder
39	Y2	IN	Portu Oku
40		CMP #FE00H	FE00H deki veri ile karşılaştır
41		BZR Y2	Aynı ise Y2 etiketine git (sensör değişmemiş ise)
42		LDA #0007H	Farklı ise Akümülatöre 7H yükle (Tıkama)
43		OUT	Porta Gönder
44		BRA A	A etiketine Dallan
45	S3	LDA #0014H	Akümülatöre 14H yükle
46		OUT	Porta Gönder
47	Y3	IN	Portu Oku
48		CMP #FE00H	FE00H deki veri ile karşılaştır
49		BZR Y3	Aynı ise Y3 etiketine git (sensör değişmemiş ise)
50		LDA #0007H	Farklı ise Akümülatöre 7H yükle (Tıkama)
51		OUT	Porta Gönder
52		BRA A	A etiketine Dallan
53	S4	LDA #000BH	Akümülatöre BH yükle
54		OUT	Porta Gönder
55	Y4	IN	Portu Oku
56		CMP #FE00H	FE00H deki veri ile karşılaştır
57		BZR Y4	Aynı ise Y4 etiketine git (sensör değişmemiş ise)
58		LDA #0007H	Farklı ise Akümülatöre 7H yükle (Tıkama)
59		OUT	Porta Gönder
60		BRA A	A etiketine Dallan
61	S5	LDA #000DH	Akümülatöre DH yükle

62		OUT	Porta Gönder
63	Y5	IN	Portu Oku
64		CMP #FE00H	FE00H deki veri ile karşılaştır
65		BZR Y5	Aynı ise Y5 etiketine git(sensör değişmemiş ise)
66		LDA #0007H	Farklı ise Akümülatöre 7H yükle (Tıkama)
67		OUT	Porta Gönder
68		BRA A	A etiketine Dallan
69		IN	Portu Oku
70		BRA A	A etiketine Dallan

7.4. Lojik Kontrolör

Bu kısımda program kodu yazmak yerine tamamen lojik kapılar seviyesinde alternatif bir kontrolör tasarlandı. Tasarlanan lojik kontrolörün blok diyagramı Şekil 7.11.'de görülmektedir. Kontrolör temel olarak üç ana kısımdan oluşmaktadır. Bunlar sensör okuma devresi, komütasyon devresi ve ölü zaman üreticisidir. Sensör okuma devresi sensör bilgisini sürekli okuyarak sensör bilgisindeki değişimi takip eder. Komütasyon devresi sensör bilgisi girişlerine karşılık gelen çıkışları oluşturur. Ölü zaman üreticisi ise ihtiyacımız olan ölü zaman gecikmesini bize sağlar.



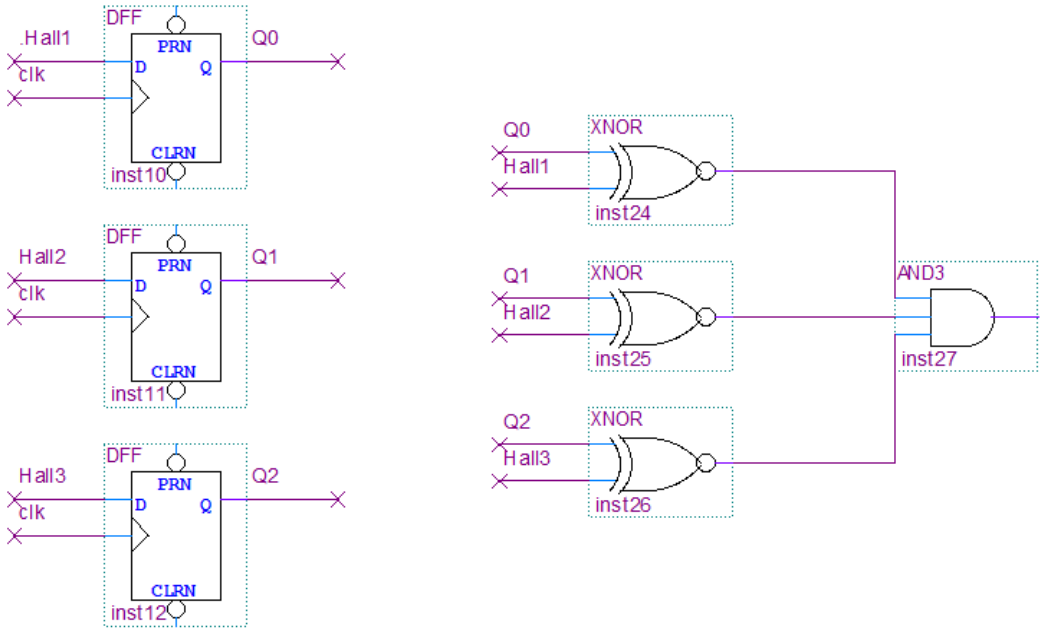
Şekil 7.11. Lojik Kontrolörün Blok Diyagramı

7.4.1. Sensör Okuma Devresi

Sensör okuma devresi sensör bilgisini okuyarak sensör bilgisindeki değişimi sürekli olarak takip eder. Sensör bilgisindeki değişimi takip etmekteki amacımız bir durumdan diğerine geçerken gerekli ölü zamanı uygulayacağımız anı yakalamaktır. Ölü zaman gecikmesini uygulamamızın sebebi transistörlerin tıkamaya geçme sürelerinin iletime geçme süresinden daha uzun olmasından kaynaklanmaktadır. Ölü zaman gecikmesini uygulamadığımız takdirde Tablo 7.6.'daki bir adımdan diğer adıma geçişte ilk adımın transistörleri tıkamaya geçmeden ikinci adımın transistörleri iletime girecektir. Bu durumdada transistörlerde ısınma ve yanmalar meydana gelecek, motorda ise ısınma ve sarsıntı oluşacaktır. Ölü zaman gecikmesiyle 1. adımın transistörlerinin tam olarak kesime gitmesinden sonra 2. adımın transistörlerinin iletime geçmesi sağlanarak emniyetli ve sarsıntısız bir sürüş sağlanmış olur.

Sensör okuma devresinin yapısı 3 adet D flip flop ve karşılaştırıcı bir devreden meydana gelmektedir. D flip flopların girişine gelen 3 bitlik sensör bilgisi flip floplara uygulanan clock sinyali ile clock sinyalinin herbir düşen kenarında flip flopların çıkışına yazılır. Karşılaştırıcı devre ise flip flopların giriş ve çıkışını sürekli karşılaştırarak sensör bilgisinde bir değişiklik olup olmadığını tarar. Karşılaştırıcıyı kullanmaktaki amacımız sensör bilgisindeki değişikliği anında algılayıp ölü zaman üreticisinin saymaya başlaması için gerekli tetikleme sinyalini göndermektir. Sensör bilgisinde değişiklik olduğu anda karşılaştırıcının çıkışı lojik 0 a düşer ve ölü zaman için gerekli sayma işlemini tetikler.

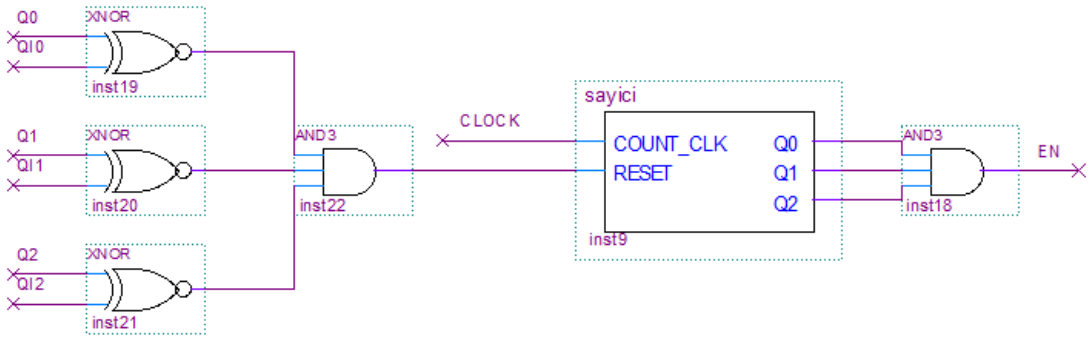
Şekil 7.12.'deki karşılaştırıcı devre 3 adet XNOR kapısından oluşur. XNOR kapısı girişlerine uygulanan lojik değerler aynı ise 1 farklı ise 0 üretir. Sensör bilgisinin değişimi anında XNOR kapısının girişlerindeki eşitlik bozulup çıkışı lojik 0'a düşer. Herhangi bir XNOR'un çıkışının lojik 0'a düşmesi ile karşılaştırıcının çıkışı lojik 0'a düşerek ölü zaman sayıcısını resetler. Resetlenen sayıcı tekrar değerine ulaşana kadar geçen süre ölü zaman süresi olarak kullanılır.



Şekil 7.12. Sensör Okuma Devresi

7.4.2. Ölü zaman Üreticisi

Devrede kullandığımız ölü zaman üreticisi aslında 0 dan 7'ye kadar sayan ve 7'ye ulaştığında resetlenene kadar sabit kalan 8 bitlik özel bir sayıcıdan başka bir şey değildir. Sensör okuma devresindeki karşılaştırıcının çıkışının 0'a düşmesiyle yani yeni sensör bilgisi gelmesiyle sayıcı resetlenerek 0 a düşer ve tekrar 7'ye kadar sayar. Bu sayma işleminde geçen süre ölü zaman gecikmesi olarak kullanılır. Karşılaştırıcının çıkışı her 0'a düştüğünde yeni çıkışlar aktif edilmeden bu ölü zaman gecikmesi uygulanır. Ölü zaman süresi boyunca tüm transistörlere tıkama bilgisi gönderilir.



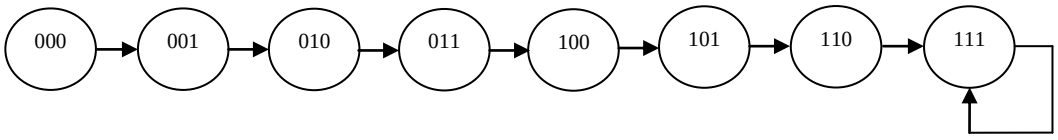
Şekil 7.13. Ölü zaman Üreticisi ve Karşılaştırıcı

Sayıcı sayma işlemini devam ettirdiği süre boyunca AND kapısı üzerinde çıkışına bağlanan EN sinyal ucu lojik 0 dır. Sayıcı 7'ye (111) ulaşıp sayma işlemini bitirdiğinde EN sinyalinin değeri lojik 1 olur ve çıkışların sürülmesine izin verir.

Ölü zaman için tasarlanan senkron sayıcı aynı zamanda bir sonlu durum makinesidir (Finite State Machine). Sonlu durum makinelerinde Mealy modeli ve Moore modeli olmak üzere iki tip model vardır. Moore modelinde çıkışlar sadece durumlara bağlıdır. Mealy modelinde ise çıkışlar girişlere ve durumlara bağlıdır [31]. Aşağıda tasarladığımız sayıcının (sonlu durum makinesi) tasarım aşamaları sırasıyla anlatılmıştır.

7.4.2.1. Durum Diyagramı

İlk olarak tasarlanacak sayıcının işlem basamaklarını tanımlamak için bir durum diyagramı çizilir. Şekil 7.14.'de tasarımımızda kullandığımız sayıcıya ait durum diyagramı görülmektedir. Sayıcı 0 dan 7'ye kadar sayacak 7'ye geldiğinde sabit kalacaktır.



Şekil 7.14. Durum Diyagramı

7.4.2.2. Flip-Flop Geçiş Tablosu

Tasarımda kullanılacak flip-flop tiplerine göre flip-flopların bir durumdan diğer duruma geçiş tabloları çıkartılır. Bu tasarımda J-K flip-floplar kulladıldığı için Tablo 7.9.'da J-K flip-floplara ait geçiş tablosu verilmiştir.

Tablo 7.9. J-K Flip-Flop Geçiş Tablosu

Çıkış Geçişleri		Girişler	
Q_N	Q_{N+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Q_N: Şimdiki Durum Q_{N+1}:Sonraki Durum

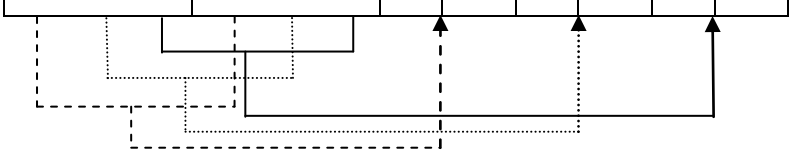
x: Dikkate Alınmayan Giriş

7.4.2.3. Sonraki Durum Tablosu

Durum diyagramı ve kullanacağımız flip-flopların geçiş tablosundan yararlanarak bir sonraki durum tablosu çıkartılır. İlk olarak önceki sütununa durum diyagramındaki bütün durumlar yazılır. Sonraki sütununa ise her durumdan bir sonra gelen durumlar işlenir. Daha sonra flip-flop geçiş tablosundan faydalanılarak bu iki sütunun aynı anlamlı bitleri sırasıyla karşılaştırılarak tablo doldurulur.

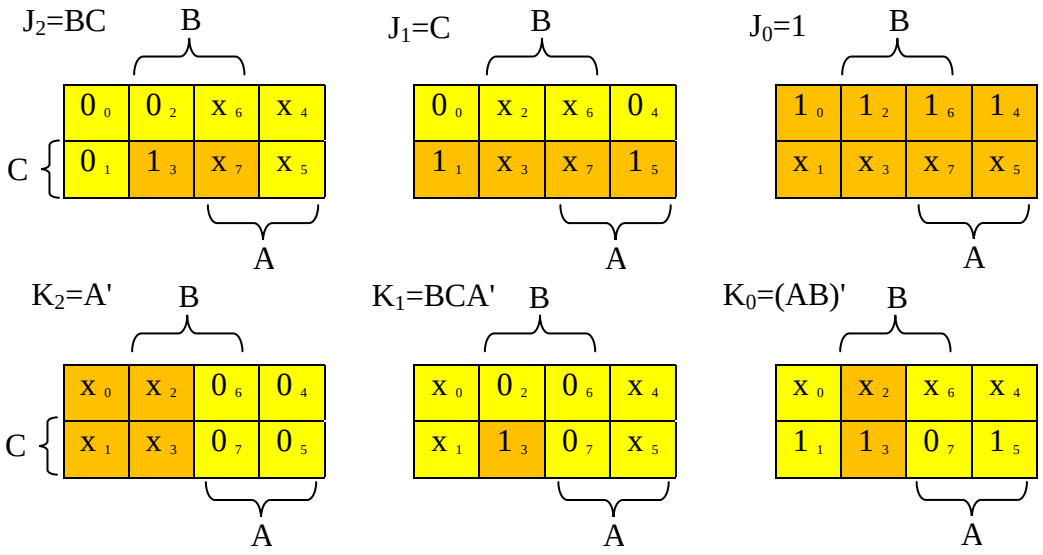
Tablo 7.10. Sonraki Durum Tablosu

Önceki			**Sonraki**								
A	**B**	**C**	**A**	**B**	**C**	**J_2**	**K_2**	**J_1**	**K_1**	**J_0**	**K_0**
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	1	1	0	x	0	1	x	x	1
1	1	0	1	1	1	x	0	x	0	1	x
1	1	1	1	1	1	x	0	x	0	x	0

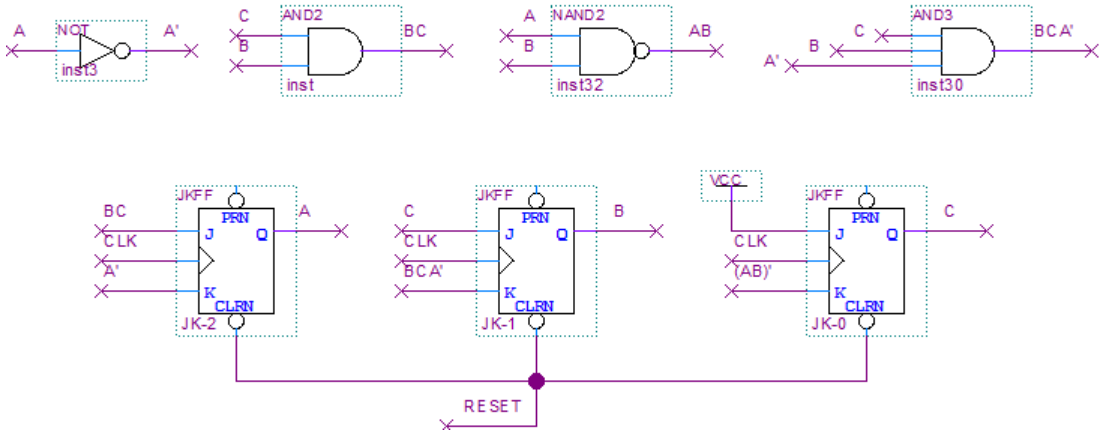


7.4.2.4. Karnaugh Haritası

Son olarak ta sonraki durum tablosu karnaugh haritasına aktarılarak flip-flop girişlerine ait eşitlikler çıkarılarak devre çizimi yapılır. Şekil 7.15.'de karnaugh haritaları şekil 7.16.'da ise devrenin tamamlanmış hali görülmektedir.



Şekil 7.15. Karnaugh Haritası

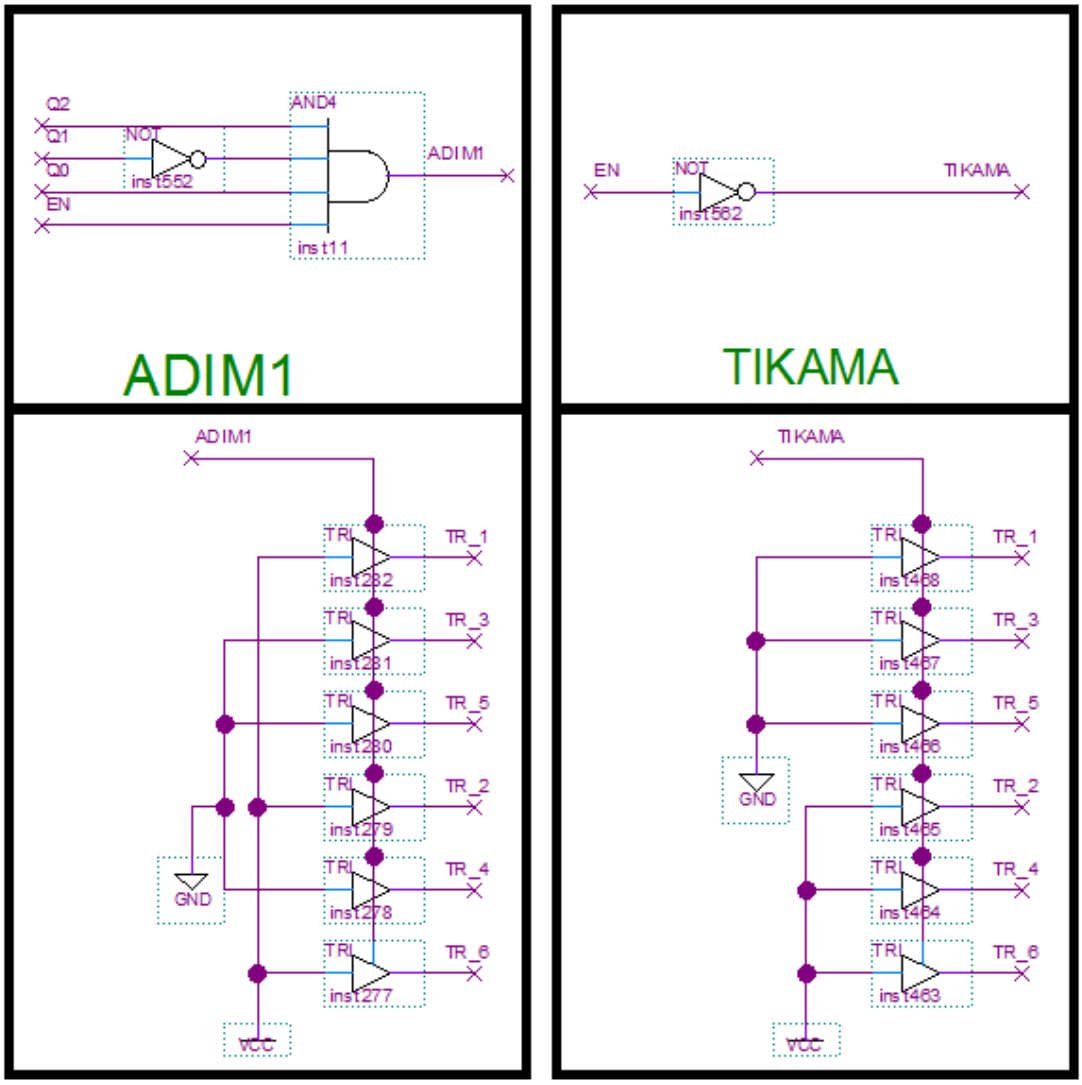


Şekil 7.16. Tamamlanmış Sayıcı Devre

7.4.3. Komütasyon Devresi

Komütasyon devresi ölü zaman üreticisini kontrol ederek, sensör okuma devresinden aldığı verilere göre transistörleri sürecek kodu üretir. Ölü zaman sayıcısı saymakta ise, yani Şekil 7.13.'deki EN sinyali lojik 0 ise transislere tıkama bilgisi gönderir. Ölü zaman sayıcısı 7'ye ulaşmış ise Tablo 7.6.'daki sensör bilgisine karşılık gelen kodu transistörlere gönderir. Komütasyon devresi sensör bilgilerine karşılık gelen 6 adım ve bir tıkama bloğundan oluşur. Şekil 7.14.'de komütasyon devresinin 1. adımı ve tıkama bloğu gösterilmiştir.

EN sinyali lojik 0'ise yani ölü zaman sayıcısı saymakta ise tıkama bloğu aktif olup tüm transistörler kesime gider. EN sinyali lojik 1 olduğunda ise sensör bilgisine göre ilgili adımın transistörleri sürülür. Örneğin hall sensörden 101 bilgisi alındığında ADIM1 sinyali lojik 1 olup bağlı olduğu tri-state buffer'ları iletime geçirerek sürücüye 100101 bilgisini gönderir. Diğer 5 adımında çalışma şekli 1. adımla aynı şekilde gerçekleşir.



Şekil 7.17. Komütasyon Devresi

SONUÇ

FPGA geliştirme kartları uygun maliyetli olmaları, yeniden yapılandırılabilir olmaları, tasarım sürelerinin uygun olması ve tasarım sonuçlarının gerçek ortamda görülebilir olması gibi özelliklerinden dolayı eğitimsel amaçlı çalışmalarda veya prototip hazırlamada sıklıkla kullanılırlar. Özellikle yeniden yapılandırılabilmeleri eğitimsel çalışmalarda ve prototip hazırlamada farklı tasarımları ek bir maliyet getirmeden aynı kart üzerinde gerçekleştirmeye olanak sağlar.

Bu tez çalışmasında temel amaç eğitimsel olduğu için tüm tasarım bileşenlerinin temel yapıda, incelenebilir, müdahale edilebilir ve geliştirilebilir olmasına özen gösterildi.

İlk olarak bu tez çalışmasında kullanılacak mikrobilgisayarın (BZK.SAU.FPGA) kontrol yönündeki eksiklikleri tespit edilerek çevresel cihazları kontrol etmesini sağlayacak bir paralel port arabirimi tasarlandı. Daha sonra mikrobilgisayardan gelecek kontrol sinyallerine göre motor için gerekli faz gerilimlerini üretecek güç devresi tasarlandı. Tasarlanan güç devresinin temel yapıda olmasına ve devrede kullanılan malzemelerin piyasadan kolaylıkla temin edilebilir olmasına özen gösterildi. Son olarak kontrol kısmı tasarımı için iki metot izlendi. Birinci metot da BZK.SAU.FPGA'in assembly komut seti kullanılarak yazılan kontrol programı ile kontrol işlemi gerçekleştirildi. İkinci alternatif metotta ise tamamen lojik kapılar seviyesinde bir kontrolör tasarlanarak motor kontrol işlemi gerçekleştirildi.

Bu çalışmada Kullanılan BZK.SAU.FPGA mikrobilgisayarın açık kaynak kodlu ve mimarisinin modüler yapıda olması sayesinde mevcut bilgisayar mimarisinin işleyişini aksatmadan, tasarlanan paralel port arabirimi mimariye entegre edildi. Bu şekilde mikrobilgisayarın çevresel birimlerle haberleşmesi yönündeki eksiklik giderilerek, kontrol uygulamalarında kullanılabilir hale getirildi. Ayrıca bu şekilde kullanıcının çalışan bir bilgisayar mimarisine müdahale ederek eklenti yapmasıyla, bir bilgisayarın çevresel birimlerle nasıl haberleştiğini bu esnada kaynaklarının nasıl

kullandığını öğrenirken özgün sistemler üretme konusunda motivasyonunun artırılması sağlanmış oldu.

Çalışmada kontrol işlemi hem bir mikrobilgisayara kod yazarak hem de lojik seviyede bir donanım tasarımı ile gerçekleştirildiği için iki yöntemin tasarım kolaylığı, esnekliği, aynı clock frekansında çalışma hızları vs. bakımından birbirleri ile kıyaslanma olanağı sağlanmış oldu.

Çalışmanın temel olarak eğitimsel amaçlı kullanımı hedeflense de pratik amaçlı olarak ta fırçasız DC motor sürme uygulamaları için bir rehber niteliğindedir. Mevcut haliyle mikroişlemci kontrollü bir fırçasız DC motor sürücünün temellerini taşıyan tasarım ileriki çalışmalarda geliştirilebilecek ilave özelliklere (tork kontrolü, hız kontrolü, yön kontrolü, sensörsüz kontrol vs.) her türlü BLDC uygulamasını gerçekleştirebilecek endüstriyel bir sürücü haline getirilebilir.

KAYNAKLAR

1. Sathyan, A., et al., An FPGA-Based Novel Digital PWM Control Scheme for BLDC Motor Drives, IEEE Transactions on Industrial Electronics , Vol. 56, No. 8, pp. 0278-0046, August 2009.

2. Sathyan, A., et al., A Low-Cost Digital Control Scheme for Brushless DC Motor Drives in Domestic Applications, IEEE Electric Machines and Drives Conference 2009, 10.1109/IEMDC.2009.5075186, 2009.

3. Alecsa, Bogdan., Onea, Alexandru., An FPGA Implementation of a Brushless DC Motor Speed Controller, 2010 IEEE 16th International Symposium for Design and Technology in Electronic Packaging, 10.1109/SIITME.2010.5653617, 2010.

4. Lin, C., Hung, C., Liu, C., Position Sensorless Control for Four-Switch Three-Phase Brushless DC Motor Drives, IEEE Transactions on Power Electronics , Vol. 23, No. 1, pp. 0885-8993, January 2008.

5. Shanmugapriya, M., Michael, P,A., Sensorless Control of an Four SwitchThree Phase Inverter using FPGA, IEEE - International Conference On Advances In Engineering, Science And Management, 978-81-909042-2-3, 2012.

6. Tsai, M., et al., Model Construction and Verification of a BLDC Motor Using MATLAB/SIMULINK and FPGA Control, 6th IEEE Conference on Industrial Electronics and Applications, 10.1109/ICIEA.2011.5975884, 2011.

7. H. Oztekin, F. Temurtas and A. Gulbag, "BZK.SAU: Implementing a hardware and software-based Computer Architecture simulator for educational purpose," Int. Conf. Computer Design and Applications (ICCDA 10, vol.4, pp.V4-90-V4-97

8. H. Oztekin, F. Temurtas and A. Gulbag, " BZK.SAU.FPGA10.0: Microprocessor architecture design on reconfigurable hardware as an educational tool," Int. Conf. Computers&Informatics(ISCI 11, pp. 385-389, doi: 10.1109/ISCI.2011.5958946), March 2011.

9. H. Oztekin, F. Temurtas and A. Gulbag, "BZK.SAU.FPGA10.1: A modular approach to FPGA-based micro computer architecture design for educational purpose," Comput. Appl. Eng. Educ.. doi: 10.1002/cae.20553, in press.

10. H. Oztekin, F. Temurtas and A. Gulbag, “A modular approach to VGA Monitor Controller for BZK.SAU.FPGA10.1 microcomputer architecture design,” Int. Proc. Computer Science and Information Technology(ICICA 12), vol. 24, pp. 27-31, February 2012.

11. H. Oztekin, F. Temurtas, E. Olmez, and A. Gulbag “FPGA-Based Flash Memory Controller for BZK.SAU.FPGA10.1 Microcomputer Architecture Design as an Educational Tool” International Journal of Computer and Communication Engineering, Vol. 1, No. 3, pp. 241-245, September 2012

12. http://www.ami.ac.uk/courses/ami4460_fpga/u01/ (Erişim Tarihi: 20.12.2012)

13. AYDIN, A., FPGA Yonga Mimarisi ve Kullanımı, Lisans, Süleyman Demirel Üniversitesi, Elektronik ve Haberleşme Mühendisliği, 2005

14. http://web.itu.edu.tr/orencik/BilgMimYenYakl2007/Mehmet_Aktas/ FPGA_Mimarisi_Rapor.pdf (Erişim Tarihi: 20/12/2012)

15. http://www.ami.ac.uk/courses/ami4460_fpga/u02/index.asp (Erişim Tarihi: 20.12.2012)

16. De2-70 User Manuel, version 1.01, 2007 Terasic Technologies, Sf. 4

17. Quartus II Subscription License, 2100@194.27.212.50

18. OZTEKIN, H., Bilgisayar Mimarisi Simülatörü Tasarımı, Y. Lisans, Sakarya Üniversitesi, Bilgisayar ve Bilişim Mühendisliği, 2009.

19. OZTEKIN, H., Eğitim Amaçlı Yapılandırılabilir Modüler Donanım Üzerine Gömülü İşletim Sistemi Tasarımı, Doktora, Sakarya Üniversitesi, Bilgisayar ve Bilişim Mühendisliği, 2012.

20. http://en.wikipedia.org/wiki/ABC_80 (Erişim Tarihi: 20/12/2012)

21. en.wikipedia.org/wiki/List_of_home_computers_by_video_hardware (Erişim Tarihi: 20/12/2012)

22. COLTON, D., EMBREY, G., FIFE, L., MIKOLYSKI, S., PRIGMORE, D., STANLEY, T. D., Pedagogic Value in Understanding Computer Architecture of Implementing the Marie Computer from Null and Lobur in the Logic Emulation Software, Multimedia Logic. Workshop On Computer Architecture Education(WCAE 2007), pp. 66–71, 2007.

23. TIEJUN, X., FANG, L., 16-bit teaching microprocessor design and application, IEEE International Symposium on IT in Medicine and Education(ITME 2008), pp. 160-163, 2008. doi: 10.1109/ITME.2008.4743843

24. GHEORGHE, A. S., BURILEANU, C., Savage16 - 16-bit RISC Architecture General Purpose Microprocessor, International Semiconductor Conference (CAS 2010), Vol. 2, pp.521-524, 2010. doi: 10.1109/SMICND.2010.5650480

25. MANO, M. M., Bilgisayar Sistemleri Mimarisi, MARŞOĞLU, A., 3. Basım, SUÇSUZ., N., Literatür Yayıncılık, pp. 129-159, İstanbul, 2002.

26. BROWN, W., Brushless DC Motor Control Made Easy, Microchip AN857 2002 Microchip Technology Inc. DS00857A

27. IEEE 1284: Parallel Ports. 2002 Lava Computer MFG Inc.

28. http://en.wikipedia.org/wiki/Multiplexer(Erişim Tarihi: 20/12/2012)

29. De2-70 User Manuel, version 1.01, 2007 Terasic Technologies, Sf. 42

30. Maxon Motor High Precision Drives And Systems Catalog, Program 2012/2013, May 2012 edition S.28

31. FLOYD, T. L., Digital Fundamentals, Ninth Edition, Pearson International Edition, pp. 447-451.

ÖZGEÇMİŞ

1982 yılında Kırıkkale'de doğan Emre ÖLMEZ, orta ve lise öğrenimini sırasıyla Kırıkkale Anadolu Lisesi, Yozgat Fen Lisesi ve Kırıkkale Süleyman Demirel Lisesinde tamamlamıştır. 2001 yılında kazandığı Kırıkkale Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği Bölümünü 2006 yılında başarıyla bitirmiştir.

2011 yılında yüksek lisans eğitimine Bozok Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Mühendisliği Anabilim Dalında başlamıştır. 2010 yılından beri Bozok Üniversitesi Araştırma Uygulama Hastanesinde Mühendis olarak çalışmakta olan Emre ÖLMEZ, evli ve 1 çocuk babasıdır.

İletişim Bilgileri

Adres : Bozok Üniversitesi Araştırma Uygulama Hastanesi YOZGAT

Telefon: (354) 212 70 60

Faks: (354) 212 71 50

E-posta: emre_olmez@hotmail.com