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

Yüksek Lisans Tezi

FPGA TABANLI MİKROBİLGİSAYAR MİMARİSİ KULLANILARAK SERİ HABERLEŞME ÜZERİNDEN ADIM MOTOR HAREKET KONTROLÜ VE UYGULAMASI

Kutlucan GÖRÜR

Tez Danışmanı
Yrd.Doç.Dr.Halit ÖZTEKİN

Yozgat 2014

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

Yüksek Lisans Tezi

FPGA TABANLI MİKROBİLGİSAYAR MİMARİSİ KULLANILARAK SERİ HABERLEŞME ÜZERİNDEN ADIM MOTOR HAREKET KONTROLÜ VE UYGULAMASI

Kutlucan GÖRÜR

Tez Danışmanı
Yrd.Doç.Dr.Halit Öztekin

Yozgat 2014

T.C.
BOZOK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEZ ONAYI

Enstitümüzün Mekatronik Mühendisliği Anabilim Dalı 70111712004 numaralı öğrencisi Kutlucan GÖRÜR'ün hazırladığı "**FPGA Tabanlı Mikrobilgisayar Mimarisi Kullanılarak Seri Haberleşme Üzerinden Adım Motor Hareket Kontrolü ve Uygulaması**" başlıklı YÜKSEK LİSANS tezi ile ilgili TEZ SAVUNMA SINAVI, Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği uyarınca 20/12/2014 günü saat 10.00'te yapılmış, tezin onayına ~~OY ÇOKLUĞU~~/ OY BİRLİĞİYLE karar verilmiştir.

Başkan: ..... Prof.Dr. Feyzullah TEMURTAŞ

Üye: ...Yrd.Doç.Dr. Halit Gözlem

Üye: ..... Yrd. Doç. Dr. M.Cumhur EROĞLU Dekan Yardımcısı

ONAY:

Bu tezin kabulü, Enstitü Yönetim Kurulu'nun 20/11/2014 tarih ve 36 sayılı kararı ile onaylanmıştır.

20/11/2014

Doç. Dr. Hidayet ÇETİN
Bozok Üniversitesi
Fen Bil.Enst.Müdürü

İÇİNDEKİLER

Sayfa

FPGA TABANLI MİKROBİLGİSAYAR MİMARİSİ KULLANILARAK SERİ HABERLEŞME ÜZERİNDEN ADIM MOTOR HAREKET KONTROLÜ VE UYGULAMASI

Kutlucan GÖRÜR

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

2014; Sayfa: 102

Tez Danışmanı: Yrd. Doç. Dr. Halit ÖZTEKİN

ÖZET

Bu çalışmada, BZK.SAU mikrobilgisayar mimarisi üzerinde RS232 asenkron seri haberleşme protokolü oluşturulmuştur. Böylece eğitimsel amaçlı olarak oluşturulan BZK.SAU mikrobilgisayar mimarisinin çevresel birimlerle ilgili eksiği oluşturalan protokol ile giderilmiş olmakla beraber diğer seri haberleşme protokolleri için gerekli olan tecrübe kazanılmıştır.İlgili protokol, donanım tanımlama dili VHDL ile oluşturulmuş, FPGA üzerinde sentezlenmiş, PC ve ModelSim programı yardımı ile doğru şekilde çalıştığı gösterilmiştir. Tez çalışmasının ikinci kısımda ise BZK.SAU mikrobilgisayar mimarisinin kendi dili olan BZK.SAU.Assm dili ile oluşturulan yazılımla adım motorlar üzerine kurulmuş Pan/Tilt mekanizması sayesinde PC bağımsız FPGA tabanlı klavye kullanılarak mekanizmaya istenen yön ve açılarda dönme işlemini gerçekleştirme yeteneği kazandırılmıştır. Böylece çalışmada iki FPGA arasında iletilen seri veriler ve FPGA'e bağlı Pan/Tilt mekanizması tez çalışmasının, gömülü sistemler ve hareketli ilişkisel mekanizmalar için temel oluşturduğu görülmüştür.

Anahtar Kelimeler: BZK.SAU, FPGA, RS232, Adım Motor

CONTROLLING AND APPLICATION OF STEPPER MOTORS OVER SERIAL COMMUNICATION USING FPGA BASED MICROCOMPUTER ARCHITECTURE

Kutlucan GÖRÜR

Bozok University
The Institute of Science and Technology
Department of Mechatronics Engineering
Master of Science Thesis

2014; Pages: 102

Supervisor: Assist. Prof. Dr. Halit ÖZTEKİN

ABSTRACT

In this study, RS232 asynchronous serial communication protocol was implemented on BZK.SAU microcomputer architecture. So in addition to fulfilling deficiency of peripheral devices for BZK.SAU microcomputer architecture, also essential background was gained for other serial communication protocols. Concerned protocol was coded by using hardware description language (VHDL) and synthesized on FPGA board, after these processes, correct functioning was verified and showed using PC and ModelSim simulation program properly. At the second part of the thesis, coding BZK.SAU.Assm language achieved for moving PAN/TİLT mechanism of desired angle and direction using keyboard without PC. Thus communicated serial data between two FPGAs, which one has depended on PAN/TİLT mechanism, can be seen to form basis of embedded systems and moving relational mechanisms.

Keywords: BZK.SAU, FPGA, RS232, Stepper Motor

TEŞEKKÜR

Tez süresince değerli tecrübe ve görüşlerini benden esirgemeyen danışman hocam Sayın Yrd. Doç. Dr. Halit ÖZTEKİN'e teşekkürü bir borç bilirim. Bununla beraber aileme ve eşime verdikleri manevi destekten dolayı teşekkür ederim.

TABLOLAR LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

KISALTMALAR LİSTESİ

FPGA : Alan Programlanabilir Kapı Dizi (Field Programmable Gate Array)

VHDL: Yüksek Seviyeli Donanım Dili (Very High Description Language)

SPLD : Basit Programlanabilir Lojik Dizisi (Simple Programmable Logic Array)

CPLD: Komleks Programlanabilir Kapı Dizisi (Complex Programmable Gate Array)

PAL : Programlanabilir Dizi Lojigi (Programmable Array Logic)

GAL : Genel Dizi Lojigi (Genaral Array Logic)

PROM: Programlanabilir Bellek (Programmable ROM)

PLA : Programlanabilir Lojik Dizisi (Progammable Logic Array)

ASIC : Uygulamaya Özel Entegre Devre (Application Specific Integrated Circuit)

LUT : Başvuru Tablosu (Look Up Table)

EDA : Elektronik Tasarım Otomasyon (Electronic Design Automation)

USB : Evrensel Veri Seri Yolu (Universal Serial Bus)

UART: Evrensel Asenkron Alıcı Gönderici (Universal Asynchronous Receiver Transmitter)

EIA : Elektronik Endüstri Topluluğu (Electronics Industries Association)

DTE : Veri Sonlandırma Cihazı (Data Terminal Equipment)

DCE : Veri Devre Cihazı (Data Circuit Equipment)

RTS : Göndermek İçin Talep (Request to Send)

CTS : Göndermek İçin Temizle (Clear to Send)

ASCII : Uluslararası Değişim İçin Amerikan Standart Kodu (American Standard Code for Information Interchange)

DC : Doğru Akım (Direct Current)

LSB : En Düşük Öneme Sahip Bit (Least Significant Bit)

MSB : En Önemli Değere Sahip Bit (Most Significant Bit)

FIFO : İlk Giren İlk Çıkar (First Input-First Output)

CAN : Kontrol Edilebilir Ağ (Controller Area Network)

PWM : Sinyal Genişliği Modulasyonu (Pulse Width Modulation)

PC : Kişisel Bilgisayar (Personal Computer)

FSM : Sonlu Durum Makinası (Finite State Machine)

1.GİRİŞ

FPGA'ler hızla gelişen elektronik devre tasarım uygulamalarında istenen sayısal devre tasarımının hızlı ve esnek bir şekilde uygulanmasını ve gömülü sistemlerin her türlü platformda gerçeklenmesini sağlayan yeniden programlanabilir entegrelerdir. FPGA'ler isminden de anlaşılabildiği gibi istenen devre yapılarının lojiksel temeller ile basit bir sayıcıdan, kompleks mikroişlemcilere kadar her türlü sayısal yapının oldukça kısa sürede tasarımının yapılmasını ve endüstriyel piyasaya sürülmesi ve uygulamaya özel entegre dizaynının yapılmasını sağlayan paralel işlem yapma yeteneğine sahip tümdevrelerdir. FPGA'lerin paralel işlem yapma yeteniği, hızlı tasarım yapılabilmesi ve üretim maliyetlerinin azalması gibi sebeplerden dolayı son yıllarda hızla artan kullanım alanları tıbbi elektronik uygulamalardan, askeri ve endüstriyel uygulamalara kadar çok sayıda uygulama alanına sahiptir. Bu sebeplerden dolayı FPGA tabanlı yapılan akademik ve Ar-Ge çalışmaları gün geçtikçe artmaktadır. Üzerinde çalışılan tez çalışmasının FPGA ve programlanmasına yönelik bilgi ve yeteneğinin artması sebebi önem arz etmektedir.

Bilgisayar ve mikroişlemci/mikrodenetleyici tabanlı haberleşme sistemleri günümüzde her uygulamada vazgeçilmez donanımların başında gelmektedir. Seri haberleşme protokolleri deyince ilk akla gelen RS232 seri haberleşme protokolünün FPGA tabanlı uygulaması akademik bir çalışmanın yanında endüstriyel uygulaması hemen hemen her bilgisayar sisteminde oldukça yaygın olan bir donanımdır. Bu açıdan ilgili tez çalışmasında RS232 seri haberleşme protokolü ile yapılan gömülü bir sistem uygulaması oldukça geçerli bir çalışma oluşturduğu düşünülmektedir..

Adım motorlar sayısal sinyalleri hareket yeteneğine dönüştüren motorlar olması sebebi ile çok sayıda endüstriyel uygulamada kullanılmaktadır. Uygulamalarda istenen açılarda ve/veya istenen sayıda dönüş yeteneğinin hassas olması sebebi ile kullanım alanları oldukça yaygındır. Tez çalışmasında FPGA, RS232 seri haberleşme protokolü ve adım motor tabanlı uygulama tez çalışmasının temel yapısını oluşturmaktadır.

BZK.SAU isimli FPGA tabanlı mikrobilgisayar doktora tez çalışmasında dış dünya ile veri haberleşmesi için gereken protokollerden RS232 seri haberleşme protokolünü VHDL dili ile oluşturulması ve yine BZK.SAU içinde BZK.SAU.Assm dili ile yazılımı yapılan kontrol tasarımı adım motorların bağlı olduğu PAN/TİLT mekanizmasını istenen açılarda iki eksenli olarak sistemin bağlı olduğu klavye ile gerçeklenmektedir.

İlgili tez çalışmasında BZK.SAU isimli FPGA tabanlı mikrobilgisayarın dış dünya ile veri haberleşme yeteneği kazandırılmasının yanı sıra mekaniksel uygulamalara dönüştürmesi tez çalışmasının akademik ve Ar-Ge'ye yönelik çalışmalara temel oluşturacağı düşünülmektedir.

1.1.Literatür Taraması

Literatürde FPGA ortamında yapılmış seri haberleşme-RS232 ve adım motor kontrolü ile ilgili çalışmalar mevcuttur. Bunların bir kısmı aşağıda sıralanmıştır.

V.Vijaya ve arkadaşları USB-RS232 protokolleri arasında veri aktarımını sağlayan dijital bir tasarım üzerinde çalışmıştır. Bu çalışmada USB tabanlı cihazlar ve RS232 tabanlı cihazlar arasında seri veri aktarımı sağlayan devre yapısı asenkron veri haberleşme mantığı çerçevesinde FPGA üzerinde gerçeklenmiştir. Alıcı ve gönderici devre yapıları USB-RS232 devre yapıları arasında ilk giren-ilk çıkar(FIFO) mantığı içerisinde tasarlanmıştır. Bu suretle aktarımı yapılan verilerin karşı cihazın müsait olmaması durumunda belirli bir süre saklanması sağlanmıştır [1].

D.Antonio-Torres ve arkadaşları Xilinx firmasının geliştirmiş olduğu FPGA tabanlı 8 bitlik PicoBlaze mikroişlemci üzerinde VGA kontrol, PS/2 kontrol , UART modülü, LCD modülü gibi çevresel yapılar tasarlanmıştır. Bunlardan UART kontrol modülü tasarımı asenkron seri haberleşme modülü tasarımı için oluşturulmuştur. Kaydedici tabanlı UART modülu tasarımında TX-RX iletişimine ek olarak TX-RX için interrupt kesmesi de söz konusudur [2].

Ming Li FPGA tabanlı dijital ve analog ölçüm alabilen entegre devresi üzerinde çalışmalarda bulunmuştur. Seri olarak haberleşebilen A/D çevirici kontrol tasarımı Verilog donanım tanımlama dili kullanarak FPGA üzerinde gerçeklenmiştir. Simulasyon sonuçları da ModelSim'de yapılmıştır. Yine aynı çalışmada RS232 seri haberleşme protokolünün yanında USB seri haberleşme protokolunun tasarımı da gerçeklenmiştir [3].

Garima Bandhawarkar Wakhle ve arkadaşları FPGA üzerinde gerçeklenen UART modülü tasarımını gerçekleştirmiştir. İlgili çalışmada UART modülü alan (receiver) ve gönderen (transmitter) olmak üzere iki kısımda tasarımı yapılmış ve frekans bölücü devre de tasarıma eklenmiştir. Böylece istenen Baud Rate oranında veri aktarımı mümkün kılınmıştır. Yine bu çalışmada, RS232 seri haberleşme tasarımı için ilgili devrenin her iki modülünün durum diyagramları ve simulasyon sonuçları gerçeklenmiştir [4].

D.V.Bhatt ve arkadaşları mikrodenetleyici içinde bulunan RS232'nin de içinde bulunduğu çesitli portların aynı evrensel port üzerinden kullanılmasına yönelik bir çalışma yapmıştır ve ilgili çalışma FPGA üzerinde gerçeklenmiştir [5].

Nurul Fatihah Jusoh ve arkadaşları FPGA tabanlı yaptıkları çalışmada RS232-USB çevirici üzerinde çalışmıştır.İlgili çalışmada FIFO tasarımına ek olarak kaydıran bellek yapısı (shift register) tasarımıda mevcuttur. Alınan veri sinyalleri 16 kez örneklenmiş olup ilgili devrenin tasarımı Verilog dilinde yapılmış ModelSim programında simule edilmiştir [6].

Shouqian Yu ve arkadaşları yaptıkları çalışmada çok kanallı FIFO tekniğini kullanarak farklı ve aynı frekanslarda UART modülü tasarımı mümkün kılınmıştır. Bu çalışmada senkron olarak ve asenkron olarak haberleşmede UART modülü tasarımına eklenmiştir. Yine bu çalışmada Master-Slave haberleşme kontrol yapısı tasarımı yapılan UART modülü içerisinde mevcuttur. İlgili devrenin simulasyon sonuçları ModelSim programında gerçeklenmiştir [7].

Yu Zhu ve arkadaşları UART modülü ve CAN haberleşme protokolu üzerine çalışmışlardır. FPGA tabanlı UART modülü ve MCU tabanlı CAN haberleşme

protokolü üzerine yapılan çalışmada her iki haberleşme protokolünün birbirleri arasında en düşük veri kaybı ve en yüksek çalışma frekansanlarının belirlenmesi üzerine yapılmıştır. Bu suretle 8 bitlik veri taşıyan UART modülü ile 47-64 bitlik veri taşıyan CAN modülü arasındaki iletişim mümkün hale gelmiştir. İlgili çalışmada sonuçlar LABview programında gözlenmiştir [8].

FANG Yi-yuan ve arkadaşları yaptıkları çalışmada benzeri örneklerinde olduğu gibi UART modül tasarımını VHDL dilini kullanarak FPGA üzerinde gerçekleştirmişlerdir. Yapılan çalışmada baud rate üreteci ve alıcı-verici tasarımları UART modülü içinde tasarlanmış, full duplex haberleşme için gerçeklenmesi sağlanmıştır [9].

Biplab Roy benzeri bir çalışma olarak yine UART modülü tasarımı gerçekleştirmiş. Xilinx FPGA üzerinde yapılan çalışmayı gerçeklemiştir [10]. Xuemei LI ve arkadaşları UART modülünün de içerisinde bulunduğu TC35 devre yapısını kullanarak SMS atan sistem üzerine çalışmalarda bulunmuştur. İlgili çalışmada GSM modülüne veri aktarımı için FPGA tabanlı RS232 arayüzü kullanılmıştır [11].

Rourab Paull ve arkadaşları Altera Firmasının MicroBlaze mikroişlemcisi üzerinde iki FPGA arasında yüksek hızlarda doğru veri aktarımı sağlamak için tasarlanan veri şifreleme algoritmasi kullanarak, yine tasarımlarını yaptıkları RS232 arayüzü üzerinden gerçekleştirmişlerdir [12].

Sa'ed Abed ve arkadaşları iki FPGA'in haberleşmesi için arayüz tasarımı ve gerçeklenmesi üzerine çalışmıştır. Tasarımı yapılan ve gerçeklenen arayüz RS232 haberleşme protokolü üzerinden veri aktarımı yapması yönünden önemlidir [13].

Daniel Carrica ve arkadaşları FPGA tabanlı adım motor kontrol yapısı üzerinde çalışmışlardır. Gerçekleştirilen kontrol tasarımı belirtilen referans giriş değerleri ve ilgili algoritma tarafından güç elektroniği devre yapısını kullanarak adım motorları hareket ettirir [14].

George K. Adam FPGA tabanlı adım motor uygulamasında neural ağ yapısını kullanarak, adım motorların hareket kontrolü sağlanmıştır. Neural ağ yapısı VHDL

dili tasarımı yapılmış ModelSim simulasyon programında simule edilmiş ve FPGA üzerine gömülmüştür [15].

Zhaojin Wen ve arkadaşları adım motor kontrol üzerine yaptıkları çalışmada FPGA tabanlı kontrol tasarımı yapmışlardır. Sinus/cosinus fonksiyonlarına bağlı mikroadım kontrol yapısı FPGA içerisine Look-Up tablosuna gömülmüştür. Bu suretle adım motorların her fazının akımını kontrol eden tasarım gerçeklenmiştir [16].

Dejan Kos ve arkadaşları yaptıkları çalışmada FPGA tabanlı adım motorun tork dalgalanmasının minumuma indirgenmesi üzerinedir. Yapılan çalışmada kapalı çevrim kontrol yapısı oluşturulmuş, adım motorun akım ve açısal değerleri kullanılarak üzerinde çalışan algoritma ile FPGA tabanlı adım motor hız ve konum kontrolü yapılabilmektedir [17].

Ngoc Quy Le ve arkadaşları FPGA tabanlı yaptıkları çalışmada açık çevrim adım motor kontrol uygulaması yapmışlardır. H köprü devresi adım motorları sürmek için kullanılmış güç elektroniği devresi FPGA tabanlı PWM kontrolör tarafından yapılmaktadır [18].

Jos´e Rafael Guzm´an-Sep´ulveda1 ve arkadaşları ilgili çalışmada FPGA tabanlı 3 eksenli adım motor kontrolü üzerine çalışmışlardır. Aynı çalışmada adım motorlar mekanik tasarım üzerine uygulanmış yarım adım ve tam adım sürme teknikleri 3'er derecelik dönüş yetenegi kazandırmıştır [19].

Tzung-Cheng Chen ve arkadaşları yaptıkları çalışmada adım motorların dinamik özellikleri üzerine çalışmalarda bulunmuşlardır. Adım motorların hız ve tork arasındaki ilişkiyi kullanarak,adım motorları sürmek için yeni algoritmalar üzerinde çalışılmıştır. Bu algoritmalar parabolik, trapezoidal ve S-curve hızlanma eğrilerini kullanarak kritik uygulamalarda adım motorların daha hassas olmasını sağlamıştır [20].

Satyam ve arkadaşları yaptıkları çalışmada 3 eksenli elektrotların mikrodenetleyici ve FPGA ile kontrolü sağlanmıştır. Klavye, EEPROM, görüntü ekranı ve adım

motorlar FPGA tabanlı platformda VHDL dili kullanarak tasarımı yapılan kontrol yapısı ile mümkün kılınmıştır [21].

Quy Ngoc Le ve arkadaşları yaptıkları çalışmada adım motorların düzgün bir şekilde hızlanma eğrilerinin ve hız sönümlemede geliştirilmiş metodun uygulaması anlatılmıştır. Adım motorun hız-tork karakteristik eğrileri, FPGA tabanlı olarak kontrol edilen sistemde ortaya konulmuştur. Sönümleme metoduda yine FPGA tabanlı kontrol algoritması içinde oluşturulmuştur [22].

Mrs. Urmila Meshram (Thulkar) ve arkadaşları robot kol üzerine yaptıkları çalışmada FPGA tabanlı adım motor kontrol yazılımını VHDL dili ile yapmışlar ve gerçeklemişlerdir [23].

Quy Ngoc Le ve arkadaşları düşük hızlı adım motor sönümlenmesi üzerine yaptıkları çalışmada FPGA tabanlı adaptif bir kontrol yapısı oluşturmuşlardır [24].

Zoonubiya Ali ve arkadaşları yaptıkları çalışmada adım motoru CPLD tabanlı açık kontrol tasarımı ile sürmüşlerdir. İlgili kontrol algoritmasi VHDL dili ile oluşturulmuş ve ilgili sonuçlar çalışmaya eklenmiştir [25].

Yang Mengda ve arkadasları FPGA tabanlı mikroadım motor sürme tekniği kontrol algoritmasi geliştirilmesi üzerine çalışmalarda bulunmuşlardır [26].

2.FPGA (ALAN PROGRAMLANABİLİR KAPI DİZİLERİ)

2.1. Programlanabilir Mantıksal Yapılar

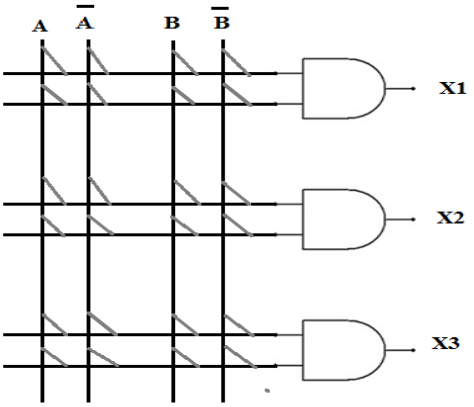
2.1.1. Programlanabilir Mantık

Basit olarak iki genel tipte programlanabilir yapı (Simple Programmable Logic Devices-SPLD) mevcuttur, bunlar PAL ve GAL'dir. PAL programlanabilir dizi mantığı (Programmable Array Logic-PAL) ve GAL ise genel dizi mantığı (General Array Logic-GAL) anlamına gelmektedir. Genel olarak bir PAL tek sefer programlanabilir bir yapı iken (One Time Programmable-OTP), GAL ise yeniden programlabilir bir çeşit PAL yapı olarak adlandılmaktadır. Çünkü bazı programlanabilir SPLD yapılar hala PAL olarak adlandırılmaktadır, PAL ve GAL'leri adlandırmakta bu suretle bir belirsizlik söz konusudur. PAL ve GAL lerin temel yapısı programlabilir VE (ANDs) ve sabit VEYA (ORs) ' lardan oluşmaktadır ki bu da çarpımların toplamı (Sum of Products) şeklinde adlandırılmaktadır. Kompleks programlanabilir mantıksal yapılar (Complex Programmable Logic Devices-CPLD) tek bir yapı olarak karşımıza çıkar ve birçok SPLD den oluşur ve daha yüksek programlabilir mantıksal tasarımlara izin vermektedir [27].

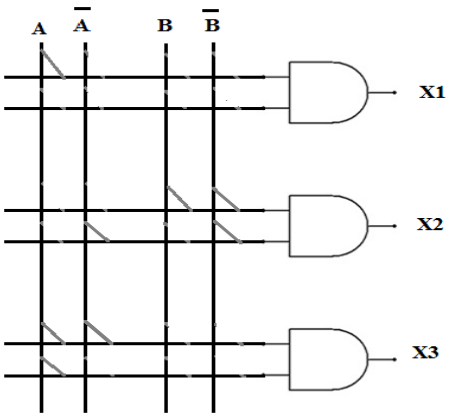
2.1.1.1.PAL(Programlanabilir Lojik Dizileri)

Programlanabilir Mantıksal Diziler (PAL) 'ler programlanabilir VE (ANDs) yapılarından ve sabit VEYA (ORs) kapılarından oluşur. Genel olarak PAL'ler sigorta teknolojisi (Fuse Process Technology) kullanarak gerçekleştirilirler. PAL içerisindeki, her programlanabilen bağlantı (Link) sigorta (fuse) olarak adlandırılır ve buna bir hücre (Cell) de denilir.

Her satır (Row) ve (AND) yapılarına bağlanır ve her sütün (Column) giriş değişkenine veya tersine (Complement) bağlanır. Bu suretle VE (ANDs) kapıları VEYA (ORs) değişkenine bağlanır ve çarpımların toplamını oluşturur (Sum of Products). Bu suretle istenen fonksiyon çıkışı elde edilir [27].



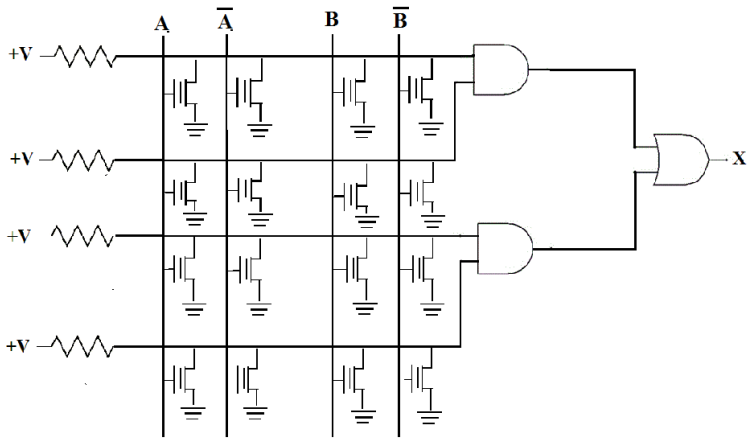
Şekil 2. 1. Programlanmayan Yapı [27]



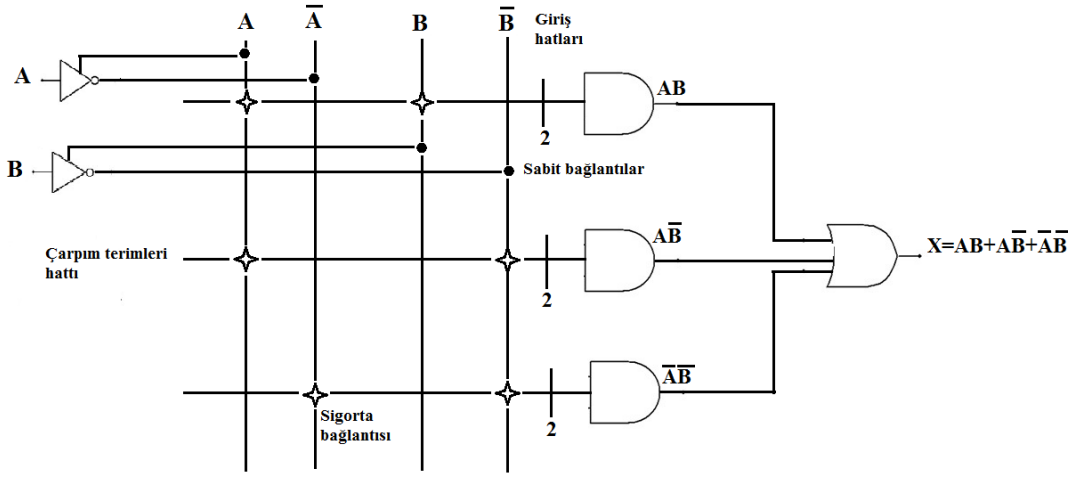
Şekil 2. 2. Programlanan Yapı [27]

2.1.1.2 GAL (Kapı Dizi Lojigi)

GAL'ler yeniden programlanabilen PAL lerdir. PAL yapısındaki VE/VEYA yapılarına benzer bir organizasyon yapısı kullanırlar. Temel bir fark ise sigorta (Fuse) teknolojisini kullanmak yerine, EPROM (E2PROM) teknolojisini kullanarak yeniden programlanabilen bir özellige sahip olmasıdır.



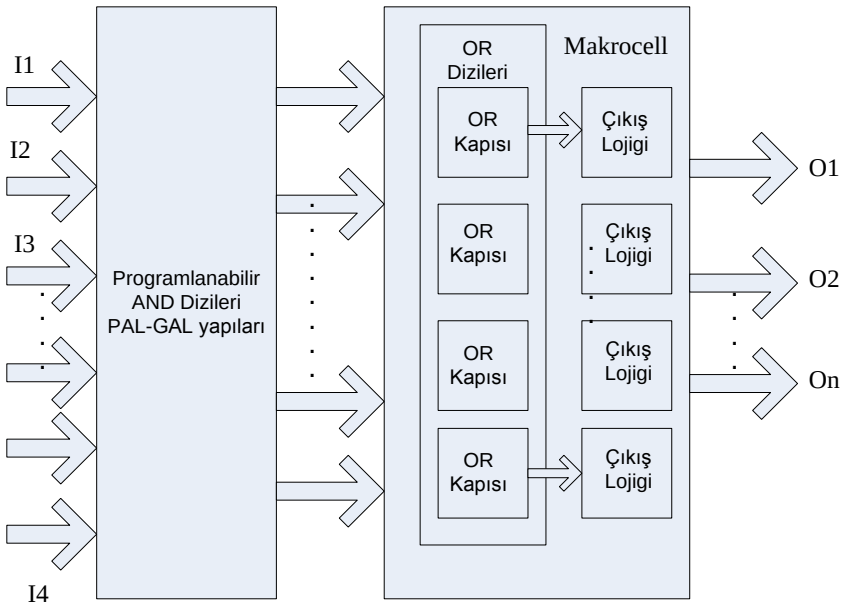
Şekil 2. 3.Basitleştirilmiş Bir GAL Yapısı [27]



Şekil 2. 4.Basitleştirilmiş Bir PAL/GAL Yapısı [27]

2.1.1.3. PAL/GAL Genel Blok Diyagramı

Hatırlanacağı gibi GAL yapıları yeniden programlanabilir iken (reprogrammable) PAL yapıları yalnızca tek sefer için programlanan yapılardır. Programlanan VE (ANDs) dizileri çıkışları doğrudan sabit VEYA(ORs) kapılarına bağlanır ve OR kapılarıda ilgili çıkış mantıksal yapılarına bağlanırlar. Bu iki yapıya Macrocell yapısı denir [27].

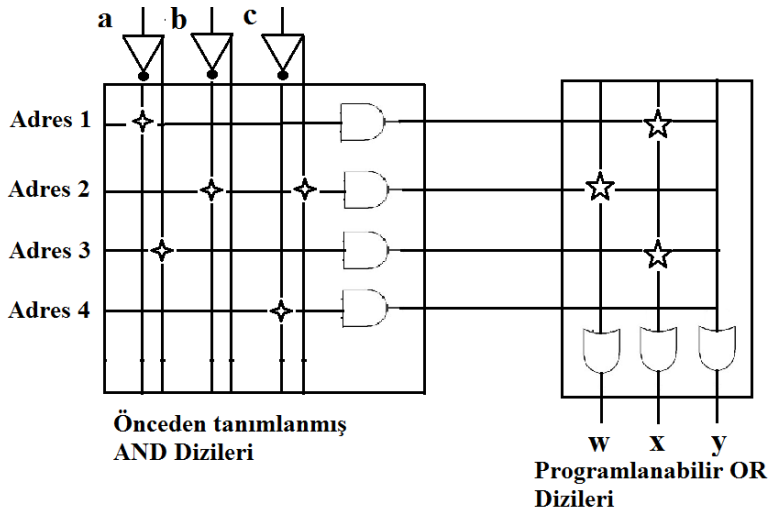


Şekil 2. 5.PAL ve GAL Yapılarının Genel Blok Diyagramı [27]

2.1.2. PROM (Programlanabilir Yalnızca Okunabilir Bellek)

PROM'lar kullanıcı tarafından kolay şekilde programlananbilen basit belleklerdir. Bir PROM içerisine, bir mikroişlemci programı, basit bir algoritma veya durum makinesi kodu yüklenebilir. PROM'lar oldukça yavaş çalıştıklarından, hız gerektiren tasarımlarda kullanışlı değillerdirler.

Bazı PROM'lar sadece bir defalığına programlanabilirken, EPROM veya EEPROM gibi PROM'ları silip tekrar programlamak mümkündür. PROM'lar sınırlı sayıda giriş/çıkışı bulunan herhangi bir kombinasyonel devrenin gerçekleştirilmesi için uygun cihazlardır [28].



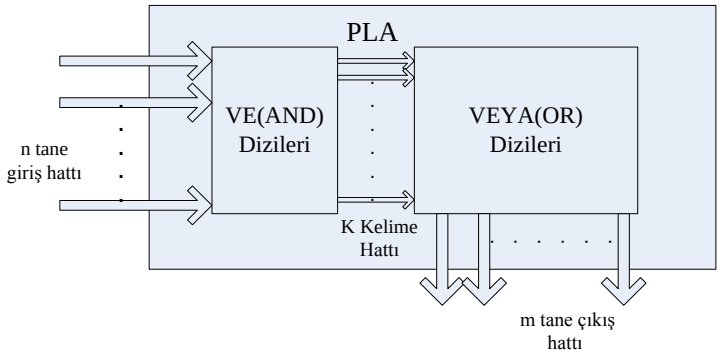
Şekil 2. 6.PROM Yapısı [29]

2.1.3. PLA (Programlanabilir Lojik Dizileri)

PROM'lardaki hız ve sınırlı sayıda giriş sorunlarına çözüm olarak PLA'lar geliştirilebilmiştir. Genel olarak bakıldığında bu cihazlar çok sayıda girişi destekleyebilirler ve daha hızlı çalışırlar.

PLA'de girişler VE kapılarından oluşan programlanabilir bir yapıya bağlıdır. Bu kısımda gerçeklestirilecek tasarıma ait VE işlemleri yapılır. VE işlemlerinin çıkışları VEYA kapılarından oluşan başka bir programlabilir yapıya bağlıdır. Burada tasarıma ait gerekli VEYA işlemleri yapılır ve çıkışlar elde edilir. PROM' daki gibi bütün kombinasyonlar gerçekleştirelemez ve sadece gerekli işlemler yapılır.

PLA'ların iki adet programlabilir alana sahip olması devrenin karmaşıklığını arttırdığı gibi fazladan kullanılan her bir sigorta bağlantısı daha büyük gecikmelere neden olur [28].

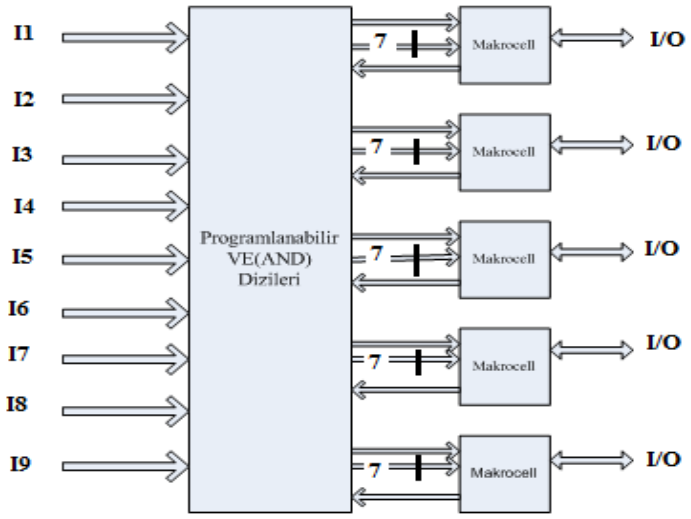


Şekil 2. 7.PLA Yapısı [30]

2.1.4. SPLD (Basit Programlanabilir Lojik Cihazları)

SPLD'ler en ucuz programlanabilen mantıksal yapılardır. SPLD'ler 4 ten 22 ye kadar macrocell'e sahiptir ve 7400 seri TTL cihazına sahiptir. Her makrocell diğer makrocellere bütünüyle bağlanır. Yukarıda da belirtildiği gibi aşağıda belirtilen mantıksal yapılar SPLD olarak isimlendirilir [31].

- PAL (Programmable Array Logic, Vantis)
- GAL (Generic Array Logic, Lattice)
- PLA (Programmable Logic Array)
- PLD (Programmable Logic Device)

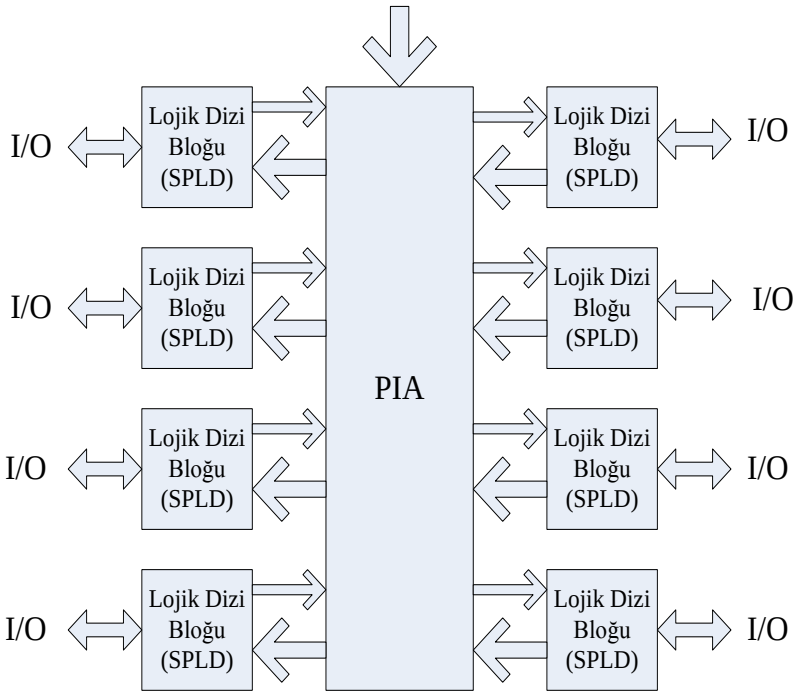


Şekil 2. 8.SPLD İç Yapısı [27]

2.1.5. CPLD (Kompleks Programlanabilir Lojik Dizileri)

PAL ve PLA'lar küçük devreler için uygulanabilir olup, çok fazla giriş/çıkış gerektiren büyük ve karmaşık devrelerin tasarımları için uygulanabilir değildirler. Bu yüzden daha karışık devre tasarımları için tek bir entegre devre içerisinde birbiri ile etkileşimli birden fazla PAL'in işlevini yerine getirebilen Complex PLD yanı CPLD'ler üretilmeye başlanmıştır.

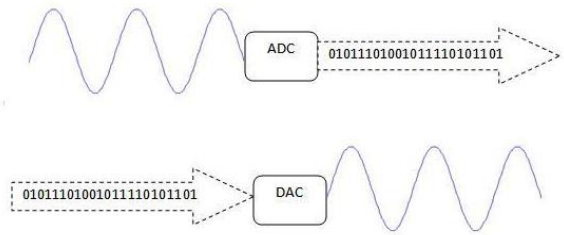
PAL ve PLA'lar ile sadece birkaç yüz mantık kapısı eşdeğerinde olan küçük devreler gerçeklestirebilirken, CPLD'lerde binler hatta yüz binlerce mantık kapısı eşdeğerindeki devreleri gerçekleştirmek mümkün olmaktadır [28].



Şekil 2. 9.CPLD İç Yapısı [27]

ASIC (Uygulamaya Özel Tümleşik Devre), genel amaçlı mikroişlemcilerin ve mikrodenetleyicilerin aksine, belirli özel bir işlemi, görevi yerine getirmek üzere tasarlanmış tümleşik devrelerdir.

Örnek olarak bir devre için adc ve adc ile birlikte bir program kullanılıyor olsun eğer satış adedi çok yüksek ise tasarlanacak bir entegre ile daha uygun fiyatlara mal edilebilir işte bu tasarlanan özel entegre devrelere ASIC denir.



Şekil 2. 10.Örnek Bir ASIC Kullanımı [32]

2.2. FPGA Özellikleri (Alan Programlanabilir Kapı Dizileri)

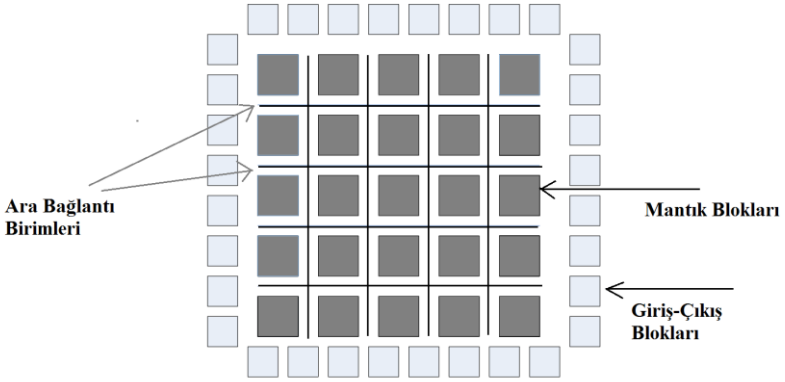
FPGA , programlanabilir mantık blokları ve bu bloklar arasındaki ara bağlantılardan oluşan ve geniş uygulama alanlarına sahip olan sayısal tümleşik devrelerdir. Tasarımcının ihtiyaç duyduğu mantık işlevlerini gerçekleştirme amacına yönelik olarak üretilmiştir. Dolayısıyla her bir mantık bloğunun işlevi kullanıcı tarafından düzenlenebilmektedir. FPGA ile temel mantık kapılarının ve yapısı daha karmaşık olan devre elemanlarının işlevselliği artırılmaktadır. Alanda programlanabilir ismi verilmesinin nedeni, mantık bloklarının ve ara bağlantıların imalat sürecinden sonra programlanabilmesidir [33].

FPGA'lerin en önemli özelliklerinden biri de paralel işlem yapabilme yeteneğidir. Paralel işlem yapabilmek aynı anda birden fazla işlemi yapabilmek demektir. Örneğin bir insanın aynı anda hem kitap okuyup hem de müzik dinlemesi ve bu arada kahve içiyor olması gibi.

Sıradan entegreler ya hiç paralel işlem yapamazlar ya da çok sınırlı yapabilirler. FPGA'de ise uygulamaya ve kapasiteye göre, birbirine paralel onlarca belki binlerce işlemi aynı anda yapabiliriz. Bu da paralel işlem gerektiren uygulamalarda FPGA'leri eşsiz kılmaktadır.

Örneğin gerçek zamanlı yüksek çözünürlüklü bir video görüntüsü üzerinde filtreleme işlemi yapmak istiyoruz. Video aslında peş peşe sıralanan resimlerdir ve bu resimlerin her birine "frame" denilmektedir. En basit haliyle bunun için videonun bir resim karesini giriş portlarından almamız, onu filtrelememiz ve çıkış portlarından göndermemiz gerekir. Sonra ikinci resim için de aynı işlemleri gerçek zamanlı olarak tekrarlamak durumundayız. Standart entegreler (örneğin bir mikroişlemci) kullanırsak bu üç işlemi (alma, filtreleme, gönderme) sırayla yapıp bitirdikten sonra gelen ikinci resmi almaya başlarız. Eğer bu işlemleri yeterince hızlı yapamazsak sıradaki resmi kaçırabiliriz. FPGA'de ise bu işlemler paralel olarak devam eder. Örneğin ilk resmi alıp filtreleme işlemini yaparken ikinci resmi almaya başlarız. İlk resmi gönderirken ikinci resmi filtrelemeye ve üçüncü resmi almaya başlarız. Bunun yanında, filtreleme işlemi genel olarak yoğun çarpım gerektirmektedir. Standart bir

işlemci ile bu çarpma işlemlerini de sırayla yapmak zorundayız. Oysa ki FPGA ile bu işlemleri de paralel olarak yanı çok hızlı bir şekilde yapabiliriz [34].



Şekil 2. 11.FPGA İç Yapısı [33]

Şekil 2. 12.Örnek Bir FPGA Entegresi [33]

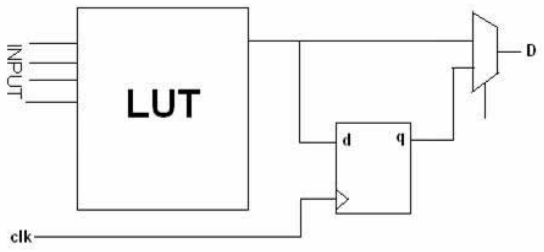
2.2.1.Mantık Hücresi

FPGA' in ana yapısını Mantık Hücreleri (Logic Cell) oluşturur. Bir Logic-Cell 1 adet Lookup Table (LUT), 1 adet D Flip-Flop ve bir adet 2 to 1 Mux'tan oluşur.

LUT ' lar aslında bir mantık işlemi yerine getiren küçük belleklerdir (RAM). N girişli bir LUT, 2eN'li bir belleğe işaret eder.

Binlerce Mantık Hücreleri'nin birleşimi sonuçunca kompleks ve büyük programlar oluşturulur.

Mantık hücrelerinin arabağlantıları matris şeklindeki veri yolları ve programlanabilir anahtarlarla (FPGA yüklenen programa göre) sağlanır. FPGA tasarımı, her bir mantık hücresinin uygulayacağı fonksiyonu ve programlanabilir anahtarların durumunu (açık/kapalı) belirleyerek bu mantık hücreleri arasındaki bağlantıları tanımlar [28].



Şekil 2. 13.Mantık Hücresi [34]

2.2.2. FPGA'in Programlanması

FPGA'i programlamak için şematik tasarımın yanında kullanılan diller şunlardır:

- VHDL(Very High Description Language)
- Verilog [37]

Sonraki bölümde VHDL dili tasarım mantığı ve örnekler ile anlatılacaktır.

2.2.3. FPGA'in Kullanım Alanları

- Savunma Sanayi
- Kriptolu İletişim
- Radar&Sonar
- Elektronik Harp
- Aviyonik Sistemler
- Silah Sistemleri

- Otomotiv
  - Görüntü İşleme
  - Araç Kontrol Sistemleri
  - Araç içi Bilgi ve Eğlence Sistemleri

- Kablolu/Kablosuz İletişim

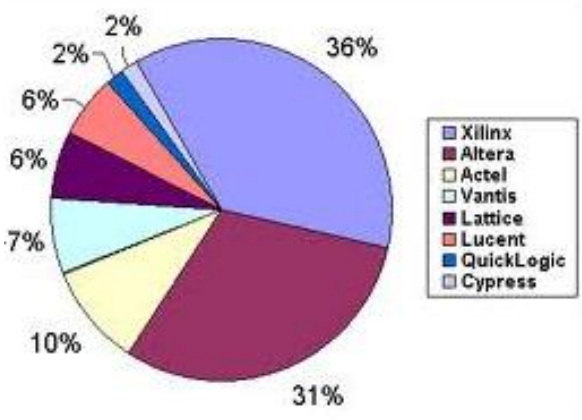
        - 3G Teknolojisi
        - GSM
        - ADSL,VDSL
        - Radyo Dalgaları
        - Optik Ağlar
        - Bilgisayar Ağları
- TV/Radyo Yayıncılığı
        - Gerçek Zamanlı ve Yüksek Çözünürlüklü Grafik İşlemcileri
        - Video Dağıtım Cihazlari
        - Video Anahtarlama ve Yönlendirme Cihazlari
        - Profesyonel Kameralar
        - Profesyonel Görüntüleme Sistemleri
- Tüketici Elektroniği
        - Akıllı Telefonlar
        - LED,LCD,Plazma TV'ler
        - Uydu Alıcılar
        - Projektorler
        - Multimedya Cihazları
        - Digital Kameralar
        - Navigasyon & GPS
- Endüstriyel Otomasyon ve Kontrol Sistemleri
        - Motor Kontrol
        - Endüstriyel Görüntüleme
        - Endüstriyel Ağlar
- Bilgisayar/ Depolama
        - Sunucular
        - Disk Sürücüler
        - Yazıcı/ Fotokopi
- Tıbbı Elektronik
        - Ultrason Görüntüleme
        - Bilgisayarlı Tomografi

        - MRI Görünteleme
        - PET Görüntüleme
        - Yaşam Destek Üniteleri
        - Hastane ve Laboratuar Elektronigi
- Test/Ölçüm
        - Ağ/Protokol Analizörleri
        - Spektrum Analizörleri
        - Bit-Hata Test Cihazlari
        - Yariiletken Tabanlı Test Cihazlari
        - Osiloskoplar
        - Lojik Analizörleri
        - Sinyal Üreteçleri
- Güvenlik/Şifreleme
        - Kripto Cihazlari
        - Askeri İletişim Cihazlari
        - Gömülü Şifreleme
        - Ağ Şifrelemesi
        - Datalink Uygulamaları
        - Taktik Telsizler
        - Uydu İletişimi
        - Finans ve Bankacılık [28].

2.2.4. FPGA Üreticileri ve Fiyatları

FPGA yüksek teknoloji bir ürün olduğundan bu pazarda genel anlamda söz sahibi olan çok az üretici bulunmaktadır. Bunların önde gelen pazar liderleri Xilinx ve Altera'dir. İkisi FPGA pazarının %80'inden fazlasına hakimdir [35].

2013 yılındaki toplam FPGA pazarının büyüklüğü yaklaşık 5 Milyar $[35]. FPGA üreticilerinin elektronik pazarındaki toplam satışına baktığımızda 2013 yılındaki üretici ve satışları arasındaki ilişkiyi gösteren grafik şöyledir.



Şekil 2. 14.FPGA Pazarı ve Şirketler [36]

2.3.VHDL

VHDL donanım tanımlama dili anlamına gelmektedir (Hardware Description Language). Fiziksel olarak gerçeklenebilecek sistem veya devrenin elektronik olarak veya sistem olarak davranışını tanımlayan özel bir dildir.

VHDL VHSIC kelimesini simgelemektedir. VHSIC'in kendisi ise 1980'lerde VHDL dilinin oluşturulmasına öncülük eden Birleşik Devletler Savunma Departmanı (DARPA) tarafından oluşturulmuştur ve Yüksek Hızlı Entegre Devre anlamına gelmektedir (Very High Speed Integrated Circuits).

VHDL'in ilk versiyonu VHDL-87 isminde ortaya çıktı. Daha sonraki versiyonu geliştirilmiş olarak VHDL-93 isminde ortaya çıktı.

VHDL dili IEEE (Institute of Electrical and Electronics Engineers) tarafından IEEE-1076 isminde standart hale getirilmiş ilk donanım tanımlama dilidir. Bu standarta ek olarak IEEE-1164 isminde çoklu-değer mantıksal sistem de ayrı bir standart olarak VHDL diline eklenmiştir.

VHDL dili tasarımi yapılan devrenin simulasyonunu gerçeklestirmenin yanında aynı zamanda ilgili devrenin sentezlenmesi (synthesis) için amaçlanmıştır. Her ne kadar VHDL dilinde tasarımı yapılan devreler tam olarak simule edilebilir olmasına rağmen, aynı yapılar sentezlenebilir değildir.

Temel bir motivasyon olarak VHDL dilinin en büyük avantajlarından birisi de belirli bir standart sahip olup teknoloji/sirket bağımsız bir dil olmasıdır. Böylece VHDL dili hareket ettirilebilir ve yeniden kullanilabilir bir özelliğe sahiptir.

VHDL dilinin en önemli iki uygulama alanına sahip olan platform CPLD (Complex Programmable Gate Array) – FPGA (Field Programmable Gate Array) ve ASIC (Application Specific Integrated Circuit)'dir.

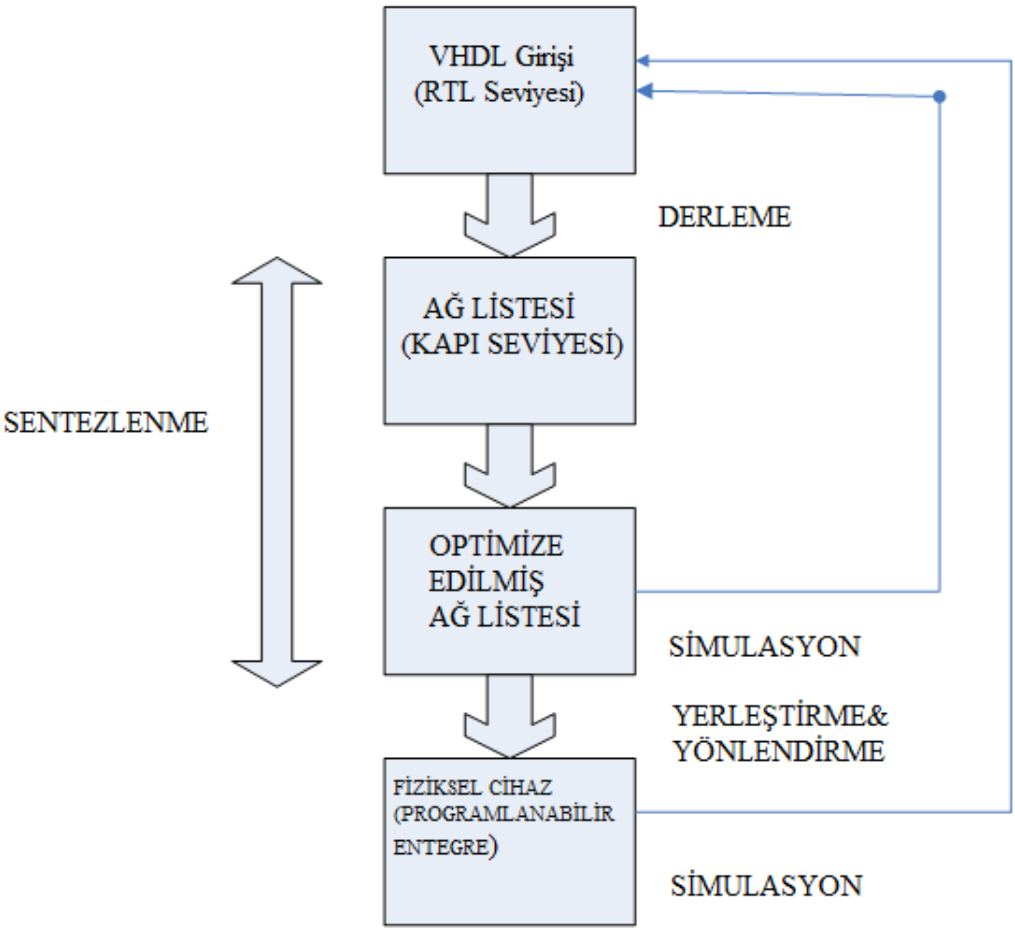
Bir kez VHDL kodu yazıldıktan sonra devre programlanabilir cihazlar üzerinde (Altera, Xilinx, Atmel) gerçeklenebilir veya ASIC oluşturulması için fabrikaya gönderilebilir. Halen varolan birçok kompleks entegre (mikrodenetleyiciler vb.) bu yol ile tasarlanır.

Bütün bunlarla birlikte VHDL dili diğer klasik programlama dillerine benzemez. Klasik programlama dillerinde program kodları yukarıdan aşığıya doğru sıralı olarak işlem görürken, VHDL dilinde ise kodlar parallel (concurrent) olarak işlem görür. VHDL dilinde yalnızca PROCESS, FUNCTION veya PROCEDURE içinde kodlar sıralı olarak çalışır [38].

2.3.1.VHDL'de Tasarım Akışı

VHDL'in en büyük avantajı bir devre veya sistemin programlanabilir yapılar (PLD veya FPGA) üzerinde veya ASIC içinde sentezlenebilir olmasıdır.

Tasarımı yapılan VHDL kodları .vhd uzantısı olarak kaydedilir. Tasarımın ismi ENTITY, dosya ismi ile aynı olmalıdır [38].



Şekil 2. 15.VHDL Tasarım Akışı Özeti [38]

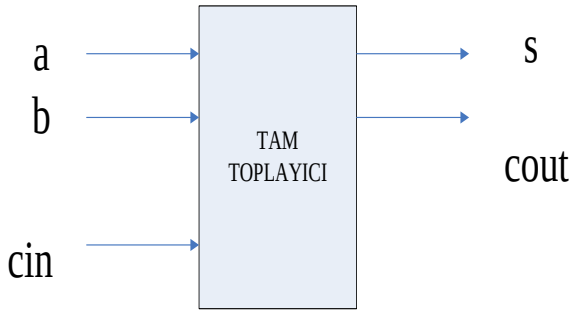
2.3.2.EDA Araçları

VHDL dilini kullanan, devrelerin sentezlenmesi, gerçeklenmesi ve simulasyonu için Elektronik Tasarım Otomasyon araçları mevcuttur. Bunlardan bazıları şunlardır: Altera firmasının ürettiği Quartus II yazılımı ile Xilinx firmasının ürettiği ISE Suite yazılımı önde gelen tasarım araçlarından biridir.

Bununla beraber yalnızca VHDL kodlarını sentezleyen veya simule eden araçlarda mevcuttur. Bunlardan bazıları: Mentor Graphics, Synopsis, Synplicity'dir.

2.3.3.VHDL Kodundan Devre Sentezlenmesi

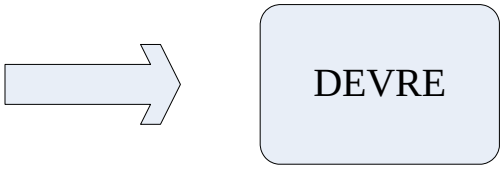
Örnek bir devrenin VHDL kod yapısından devrenin oluşturulmasına kadar ki süreçleri şu şekildedir. Devre yapısı, Tam Toplayıcı (Full Adder) olarak isimlendirilen mantıksal bir devre yapısıdır.



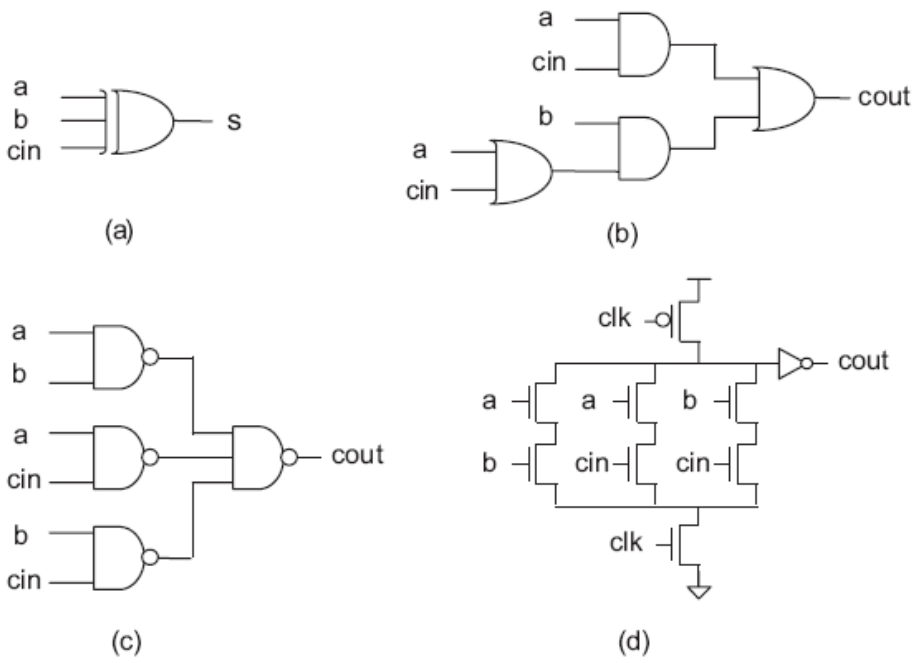
a b	cin	s	cout
0 0	0	0	0
0 1	0	1	0
1 0	0	1	0
1 1	0	0	1
0 0	1	1	0
0 1	1	0	1
1 0	1	0	1
1 1	1	1	1

Şekil 2. 16.Tam Toplayıcı Tasarımı ve Doğruluk Tablosu [38]

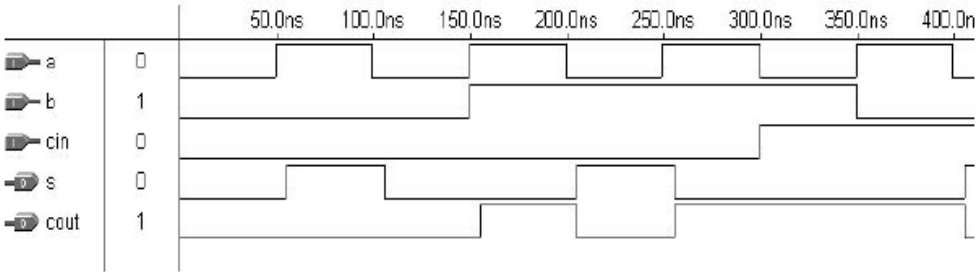
```
ENTITY full_adder IS
PORT (a, b, cin: IN BIT;
s, cout: OUT BIT);
END full_adder;
---------------------------------------
ARCHITECTURE dataflow OF full_adder IS
BEGIN
s <= a XOR b XOR cin;
cout <= (a AND b) OR (a AND cin) OR
(b AND cin);
END dataflow;
```



Şekil 2. 17.Tam Toplayıcı VHDL Kodları [38]



Şekil 2. 18.Tam Toplayıcı Devresinin Oluşabilecek Devre Yapıları [38]



Şekil 2. 19.Tam Toplayıcı Devresinin Simulasyon Sonucu [38]

2.3.4.VHDL Kod Yapısı

VHDL kodu temel olarak 3 bölümden oluşur.Bunlar: LIBRARY, ENTITY ve ARCHITECTURE'dır [38].

2.3.4.1.VHDL Temel Yapıları

VHDL dili temel olarak 3 kısımdan oluşur. Bunlar [38]:

LIBRARY: VHDL dilinde tasarım gerçeklestirmek için kütüphanelere ihtiyac vardir. Bunlardan bazıları: ieee, std,work vb.dir.

ENTITY: Devrelerin I/O pinlerini tanımlar.

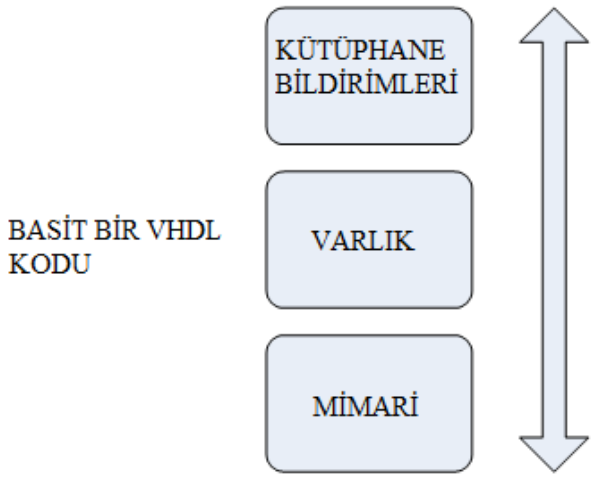
ARCHITECTURE: VHDL kodları ile oluşturulmuş olan bu yapı, devrenin davranışını tanımlar.

2.3.4.2.VHDL Kütüphane Bildirimleri

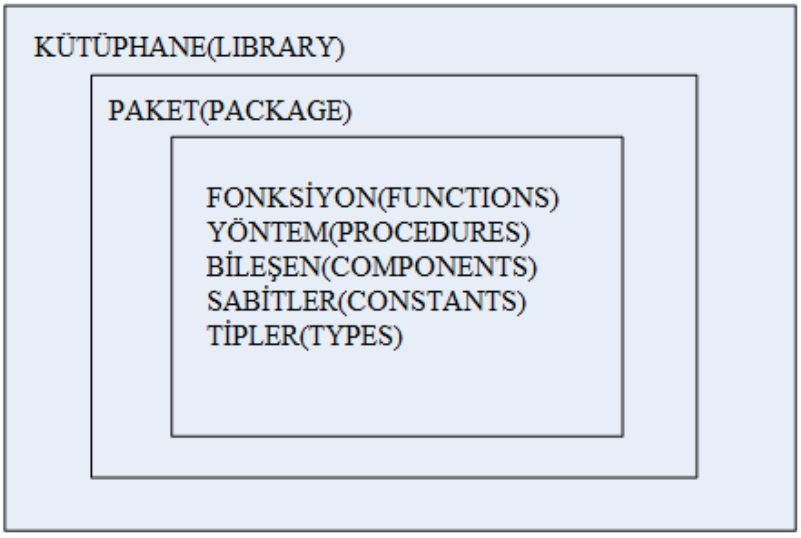
Kütüphane (LIBRARY) içerisinde daha sonra tekrar tekrar kullanılacak ve diğer tasarımlara paylaşılacak kodlar söz konusudur [38].

Bununla beraber FUNCTION'lar, PROCEDURE'ler veya COMPONENT'ler de yine kütüphane içinde yer alır.

Kütüphane yapısının şu şekilde bir kullanımı söz konusudur.
LIBRARY library_name;
USE library_name.package_name.package_parts;
Tasarımda Kütüphane kullanımı ve gerekli olan paketler şunlardır:
ieee.std_logic_1164 (from the ieee library),
standard (from the std library),
work (work library).



Şekil 2. 20.Basit Bir VHDL Kod Yapısının Temel Bölümleri [38]



Şekil 2. 21.Bir Kütüphanenin Temel Yapıları [38]

2.3.4.3.VHDL'de Varlık

Varlık (ENTITY) giriş/çıkış portları olarak adlandırılan kod yapılarını içerir.Örnek kod yapısına sahip yapılar şunlardır [38]:

ENTITY entity_name IS

PORT (

port_name : signal_mode signal_type;
port_name : signal_mode signal_type;
...);
END entity_name;

Signal modları IN, OUT, INOUT veya BUFFER olarak adlandılır. IN ve OUT tek yönlü olarak veri iletişimi sağlarken, INOUT ile iki yönlü veri transferi sağlanır.

2.3.4.4.VHDL'de Mimari

Mimari (Architecture) bir devrenin davranışını ortaya koyan kod yapılarıdır

[38].ARCHITECTURE architecture_name OF entity_name IS
[declarations]

BEGIN

(code)

END architecture_name;

Örnek bir devre yapısı üzerinde, mimari yapısının kullanımı şu şekildedir:

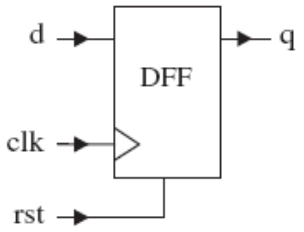
ARCHITECTURE myarch OF nand_gate IS

BEGIN

x <= a NAND b;

END myarch;

Örnek bir devre yapısının devre yapısı ve VHDL kodları şu şekildedir:



Şekil 2. 22.D flip-flop

```
1 ---------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ---------------------------------------
5 ENTITY dff IS
6 PORT ( d, clk, rst: IN STD_LOGIC;
7 q: OUT STD_LOGIC);
8 END dff;
9 ---------------------------------------
10 ARCHITECTURE behavior OF dff IS
11 BEGIN
12 PROCESS (rst, clk)
13 BEGIN
14 IF (rst='1') THEN
15 q <= '0';
```

```
16 ELSIF (clk'EVENT AND clk='1') THEN
17 q <= d;
18 END IF;
19 END PROCESS;
20 END behavior;
21 ---------------------------------------
```

D flip-flop yapısının VHDL kod örnekleri yukarıdaki gibidir. D flip-flop bir bitlik bir giriş sinyalini clock'un yükselen kenarı gelince çıkışa ileten mantıksal bir devre yapısıdır.

2.3.5.Veri Tipleri

VHDL dilinde kodların etkili bir şekilde yazılabilmesi için veri (data) tiplerinin bilinmesi gerekir. Bütün temel veri tipleri, VHDL dilinde sentezlenebilirdir. VHDL dilindeki temel veri tipleri ve bunların birbirlerine dönüşümleri şu şekildedir [38].

2.3.5.1.Ön Tanımlı Veri Yapıları

VHDL IEEE-1076 ve IEEE-1164 standartlarında bir dizi öntanımlı veri yapılarına sahiptir.

Paket/Kütüphaneler içerisindeki veri yapıları şu şekildedir.

- std kütüphanesi, standart paket içinde: BIT,BOOLEAN,INTEGER ve REAL veri yapıları mevcuttur.
- ieee kütüphanesi, std_logic_1164 paket içinde: STD_LOGIC ve STD_ULOGIC veri yapıları mevcuttur.
- ieee kütüphanesi, std_logic_arith paket içinde SIGNED, UNSIGNED veri tipleri yanında, conv_integer(p), conv_unsigned(p, b), conv_signed(p, b), and conv_std_logic_vector(p, b) gibi veri çevrim fonksiyonları da mevcuttur.
- ieee kütüphanesi, std_logic_signed ve std_logic_unsigned paket içinde std_logic_vector veri yapısını içerir.

std_logic ve std_logic_vector IEEE-1164 standartı içinde 8 mantıksal değer alabilir. Bunlar:

- ‘X’ Forcing Unknown-Zorlanmış bilinmeyen (Sentezlenebilir)
- ‘0’ Forcing Low-Zorlanmış düşük (Sentezlenebilir)
- ‘1’ Forcing High-Zorlanmış yüksek (Sentezlenebilir)
- ‘Z’ High impedance-Yüksek empedans (Sentezlenebilir-3 durumlu buffer)
- ‘W’ Weak unknown-Zayif bilinmeyen
- ‘L’ Weak low-Zayıf düşük
- ‘H’ Weak high-Zayıf yüksek
- ‘–’ Don’t Care-Önemsenme yok

VHDL dili içinde kullanılan veri yapılarının alabileceği değer yapıları ise şu şekildedir:

- BOOLEAN: True,False
- INTEGER: 32-bit integers-tam sayı (-2,147,483,647’dan +2,147,483,647)
- NATURAL: Negatif olmayan sayılar (0’dan +2,147,483,647)
- REAL: Gerçek Sayılardan Oluşur(-1.0E38’den +1.0E38 kadar)
- Fiziksel Gercegi: Voltaj, zaman gibi fiziksel büyüklükleri ifade etmek için kullanılır. Sentezlenebilir değildir.
- Karakter Gerçeği: ASCII karakterinden veya dizisinden oluşur. Sentezlenebilir değildir.

Veri yapıları ile ilgili bazi örnekler şunlardır:

- SIGNAL x: BIT; (Tek bitlik veri alanı için kullanılır.)
- SIGNAL y: BIT_VECTOR (3 DOWNTO 0); (4 bitlik veri alanı için kullanılır. En soldaki bit en önemli-MSB bittir.)
- SIGNAL w: BIT_VECTOR (0 TO 7); (8 bitlik veri alanı için kullanılır. En sağdaki bit en önemli-MSB bittir.)
  - x <= '1';
  - y <= "0111";
  - w <= "01110001";

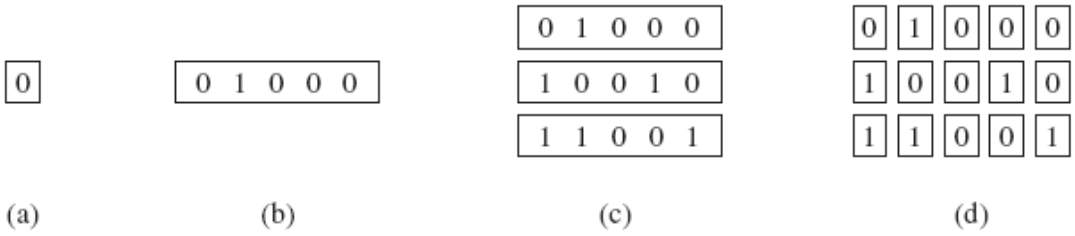
Veri yapıları ve veri atamaları ile ilgili diğer örnekler şu şekildedir:

- x0 <= '0'; -- bit, std_logic, veya std_ulogic değer
- x1 <= "00011111"; -- bit_vector, std_logic_vector, std_ulogic_vector, signed veya unsigned
- x2 <= "0001_1111"; -- Daha kolay okunabilmesi için.
- x3 <= "101111" -- Ondalıklı sayının binari okunması -47
- x4 <= B"101111" -- Ondalıklı sayının binari okunması- 47
- x5 <= O"57" -- Ondalıklı sayının oktal tabanda gösterimi- 47
- x6 <= X"2F" --Ondalıklı sayının oktal tabanda gösterimi- 47
- n <= 1200; -- integer (tam sayı)
- y <= 1.2E-5; -- real(gerçek), sentezlenemez
- q <= d sonra 10 ns; -- Fiziksel, sentezlenemez

2.3.6.Diziler

VHDL dilinde aynı veri yapısına sahip objelerin oluşturduğu yapılara dizi (array) denir.

Bir boyutlu (1D),iki boyutlu (2D) ve üç boyutlu (3D) diziler VHDL dilinde mevcuttur [38].



Şekil 2. 23.a) Scalar b) 1D Dizi c) 1D*1D Dizi d) 2D Dizi [38]

2.3.7.Veri Çevrimleri

VHDL dili farklı veri tipleri arasında doğrudan işleme (aritmatik, mantıksal) izin vermez.

Bu sebepten dolayı verilerin birbirlerine dönüşümleri gereklidir. Bu dönüşüm fonksiyonlarından bazıları şunlardır [38]:

- conv_integer(p): INTEGER,UNSIGNED,SIGNED veya STD_ULOGIC tipindeki parametreleri INTEGER değerine dönüştürür.

- conv_unsigned(p,b): INTEGER, UNSIGNED, SIGNED veya STD_ULOGIC tipindeki parametreleri UNSIGNED değerine dönüştürür.
- conv_signed(p,b): INTEGER, UNSIGNED, SIGNED veya STD_ULOGIC tipindeki parametreleri SIGNED değerine dönüştürür.
- conv_std_logic_vector(p,b): INTEGER, UNSIGNED, SIGNED veya STD_ULOGIC tipindeki parametreleri STD_LOGIC_VECTOR değerine dönüştürür.

2.3.8.Operatör ve Atamalar

Operator ve atamalar VHDL dilinde kod yazman için önemli bir konudur.

2.3.8.1.Operatörler

VHDL dili bazı ön tanımlı operatörleri destekler.Bunlar [38] :

- Atama Operatörler(Assignment Operators)
- Lojiksel Operatörler (Logical Operators)
- Aritmatik Operatörler (Arithmetic Operators)
- İlişkisel Operatörler(Relational Operators)
- Kaydırma Operatörler (Shift Operators)
- Birleştirme Operatörler(Concatenation Operators)

2.3.8.2.Atama Operatörleri

Atama Operatörleri signal, değişken (variable), sabit (constant)' lere değer atamalarda kullanılır [38].

<= SIGNAL'e değer atamalarında kullanılır.
:= VARIABLE,CONSTANT veya GENERIC'e değer atamalarında kullanılır.
=> Bir vektörün herhangi bir bitine değer atamalarında kullanılır.
Atama Operatörleri ile ilgili bazı örnekler şu şekildedir:

```
SIGNAL x : STD_LOGIC;
VARIABLE y : STD_LOGIC_VECTOR(3 DOWNTO 0); -- En yüksek değerlikli bit(MSB) en soldadır.
```

SIGNAL w: STD_LOGIC_VECTOR (0 TO 7); -- En yüksek değerlikli bit(MSB) en sağdadır.

x <= '1'; -- '1' değerinin x'e atanması.

y := "0000"; -- "0000" değerinin y'atanması.

w <= "10000000"; -- "10000000" değerinin w'e atanması

w <= (0 =>'1', OTHERS =>'0'); -- En düşük değerlikli bit (LSB) '1' iken, diğerleri '0' dir.

2.3.8.3.Mantıksal Operatörler

VHDL dilinde mantıksal işlemleri gerçeklestirmek için kullanılır. İşlem görecek veri tipleri şunlardır. BIT, STD_LOGIC, STD_ULOGIC, BIT_VECTOR, STD_LOGIC_VECTOR, STD_ULOGIC_VECTOR [38].

Bu mantıksal operatörler şunlardır:

- NOT
- AND
- OR
- NAND
- NOR
- XOR
- XNOR

2.3.8.4.Aritmatik Operatörler

VHDL dilinde aritmatik işlemleri gerçekleştirmek için kullanılır. İşlem görecek veri tipleri şu şekildedir: INTEGER, SIGNED, UNSIGNED veya REAL.

Tablo 2. 1.VHDL'de Aritmatik Operatörler [38]

+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
**	Exponansiyel
MOD	Mod Alma
REM	Kalan

ABS	Mutlak Değer

2.3.8.5.Karşılaştırma Operatörleri

VHDL dilinde karşılaştırma işlemlerini gerçekleştirmek için kullanılırlar. İşlem görecek veri tipleri şunlardır: INTEGER, SIGNED, UNSIGNED veya REAL.

Tablo 2. 2.VHDL'de Karşılaştırma Operatörleri [38]

=	Eşit
/=	Eşit Değil
<	Daha az
>	Daha çok
<=	Daha az veya eşit
>=	Daha çok veya eşit

2.3.8.6.Kaydırma Operatörleri

Kaydırma (Shift) operatörleri VHDL dilinde bir vektör veya dizideki her biti sağa veya sola kaydırmak için kullanılır. Kaydırma operatörleri şunlardır:

Sll--Tek bir bit sola kaydırır. En sağdaki pozisyon '0' ile doldurulur.

Srl—Tek bir bit sağa kaydırır. En soldaki pozisyon '0' ile doldurulur.

2.3.9. Veri Özellikleri

VHDL dilinde bir dizi veya vektor üzerindeki verilerin istenilen pozisyondaki bilgileri verir. Bunlardan bazıları şunlardır [38].

Tablo 2. 3.VHDL'de Dizi ve Vektör Özellikleri Tablosu

d'LOW	Bir dizideki en düşük index'i geri dönderir.
d'HIGH	Bir dizideki en büyük index'i geri dönderir.
d'LEFT	Bir dizideki en soldaki index değerini dönderir.
d'RIGHT	Bir dizideki en sağdaki index değerini dönderir.
d'LENGTH	Bir dizinin uzunluğu bilgisini dönderir.
d'RANGE	Bir dizinin vektor büyüklüğü bilgisini dönderir.
d'REVERSE RANGE	Bir dizinin vektör büyüklüğünün sırasını ters dönderir.

Eğer signal numaralandırılmış ise:

d'VAL(pos): Bir dizi veya vektörde belirtilen pozisyondaki değeri geri dönderir.

d'POS(value): Bir dizi veya vektörde belirtilen değerin pozisyonunu geri dönderir.

d'LEFTOF(value): Bir dizi veya vektörde belirtilen değerin sol yanındaki değerin pozisyonunu geri dönderir.

d’VAL(row,column): Bir dizideki belirtilen pozisyonlardaki değeri geri dönderir.

2.3.10.Signal Özellikleri

VHDL dilinde signal özellikleri şu şekildedir.

Tablo 2. 4.VHDL’de Signal Özellikleri [38]

s’EVENT	s olayı gerçekleştigi zaman geriye mantıksal olarak 1=doğru değeri dönderir
s’STABLE	s olayı gerçekleşmediği zaman geriye mantıksal olarak 1=doğru değeri dönderir.
s’ACTIVE	s=’1’ gerçeklestiği zaman geriye mantıksal olarak 1=doğru değeri dönderir.
s’QUIET<time>	Belirtilen zaman içinde olay gerçekleşmez ise 1=doğru değeri geri dönderir.
s’LAST_EVENT	Son olaydan beri geri dönen zamanı geri dönderir.
s’LAST_ACTIVE	Son olaydan beri s=1 olduğu zaman geri dönen zamanı geri dönderir.
s’LAST_VALUE	Son olaydan sonraki s olayını geri dönderir.

İlk iki signal özelliği dışındaki özellikler yalnızca simulasyon amaçlıdır. İlk iki özellik aynı zamanda sentezlenebilirdir. Özellikle s’EVENT özelliği, VHDL’de en çok kullanılan signal özelliğidir. Örnek kullanımlar şu şekildedir:

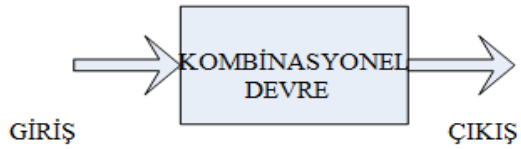
- If (s’EVENT and clk=’1’) -- s olayının yükselen kenarında ve clk=’1’ sinyali gerçekleştiği zaman doğru=true geri dönderir.
- IF (NOT clk'STABLE AND clk='1') -- s olayı durumu sabitse ve clk=’1’ sinyali gerçekleştiği zaman doğru=true geri dönderir.
- WAIT UNTIL (clk'EVENT AND clk='1') -- clk olayının yükselen kenarı tetiklenene kadar bekler.
- IF RISING_EDGE (clk)...-- clk sinyalinin yükselen kenarında doğru=true geri dönderir.

2.3.11.Paralel Kodlar

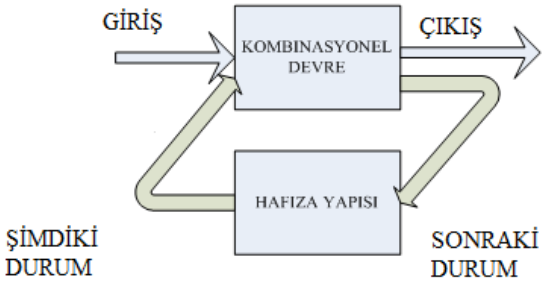
VHDL kod yapısı tasarımı içinde, parallel (concurrent) ve ardışıl (sequential) kod yapıları önemlidir.

2.3.11.1.Paralel-Ardışıl Kod Karşılaştırması

VHDL dilinde paralel kod yapısının temelinde kombinasyonel kod ve devre yapısı vardır. Kombinasyonel devrelerde çıkışlar yalnızca girişlere bağlı olarak değer üretirler. Ardışıl devreler de ise giriş ve çıkış yapıları arasında o an var olan değerleri-durumları tutmaya yarayan hafıza yapıları bulunur. Böylece bir önceki durumlar kombinasyonel yapılara geribağlantı (feedback) sağlayarak bir sonraki durumu etkilemiş olurlar.



Şekil 2. 24.Kombinasyonel Devre [38]



Şekil 2. 25.Ardışıl Devre [38]

VHDL kod yapısı içinde paralel yapılar PROCESS, FUNCTIONS veya PROCEDURES dışında işlem görür. Özelikle PROCESS, VHDL dilinde en çok kullanılan kod yapısından birisidir. PROCESS ARCHITECTURE içinde kullanılır ve içine yazılan kod yapısı ardışıl olarak işlem görür.

PROCESS kod yapısının kullanım şekli şu şekildedir.
[label:] PROCESS (sensitivity list-hassasiyet listesi)
[VARIABLE name type [range] [:= initial_value;]]
BEGIN
(sequential code-ardışıl kodlar)

END PROCESS [label-etiket];

2.3.12.When-Case Karşılaştırması

VHDL dilinde When-Case (Ne Zaman- Durum) yapısı en çok kullanılan kod yapılarından birisidir. When-Case yapısı arasındaki karşılaştırma şu şekildedir [38].

Tablo 2. 5.VHDL'de When ve Case Karşılaştırması

	WHEN	**CASE**
	Concurrent-Paralel	Sequential-Ardışıl
Kullanımı	PROCESS, FUNCTION veya PROCEDURES dışında işlem görür.	PROCESS, FUNCTION veya PROCEDURES içinde işlem görür.

2.3.13. Sinyal ve Değişkenler

VHDL dilinde sinyal (SIGNAL), değişkenler (VARIABLE), sabitler (CONSTANT) paralel ve seri kod yazım yapılarında en çok kullanılan kod yapılarındandır [38].

Constant (sabit) VHDL kod yapısı içinde önceden atanmış, var olan değerleri ifade etmek kullanılır. Örnek kullanım şablonu ve örnek kullanım şekli şu şekildedir.

```
CONSTANT name : type := value;
CONSTANT set_bit : BIT := '1';
CONSTANT datamemory : memory := (('0','0','0','0'),
('0','0','0','1'),
('0','0','1','1'));
```

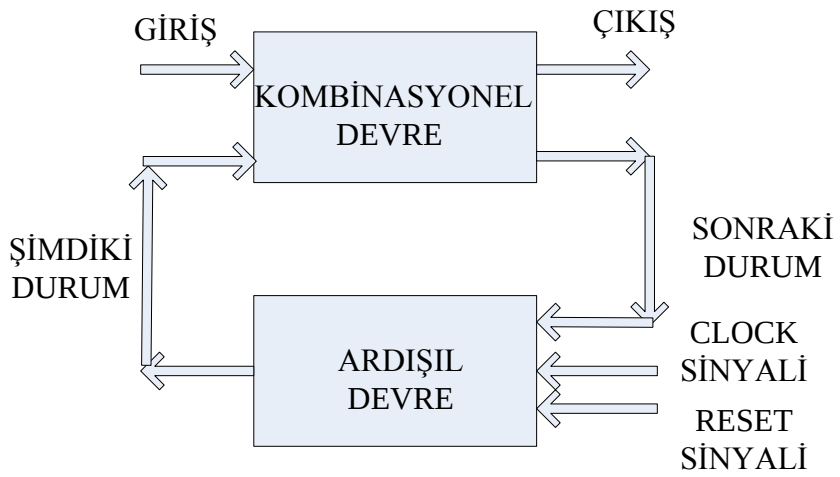
- Signal (Sinyal) VHDL kod yapısı içinde, devreye giren ve çıkan sayısal bloklar arasında iletişimi sağlayan yapılardır. Örnek kullanım şekli şu şekildedir:
  SIGNAL name : type [range] [:= initial_value];
- Variable (Değişken) VHDL kod yapısı içinde,sabit (constant) veya sinyal(signal) gibi global bir değişken değildir, yerel (local) bir değişkendir. Örnek kullanım şekli şu şekildedir:
  VARIABLE name : type [range] [:= init_value];

Tablo 2. 6.VHDL'de Sinyal ve Değişken Karşılaştırması [38]

	SİNYAL(SIGNAL)	**DEĞİŞKEN(VARIABLE)**
Atama Operatoru	<=	:=
Görevi	Devre Bağlantıları sağlamaktadır.	Yerel bilgi sunar.
Kapsam	Küresel (Global) değişkenlerdir.	Yerel Değişkenlerdir.Process, Function veya Procedure içinde kullanılır.
Davranış	Güncelleme işlemi o an içinde gerçeklesmez. PROCESS, FUNCTION, PROCEDURES'ların sonunda günceleme işlemi gerçekleşir.	Güncelleme işlemi o an içinde, bir sonraki satırda gerçekleşir.
Kullanımı	Paket (Package), Varlık (Entity) veya Mimari (Architecture) içinde kullanılır.	Yalnızca seri kod yapısı içinde kullanılır.PROCESS, FUNCTION veya PROCEDURE içinde kullanılır.

2.3.14. Sonlu Durum Makineleri

Sonlu durum makineleri (Finite State Machine) VHDL dilinde ardışıl (sequential) devreleri tasarlamak için kullanılan özel tasarım tekniğini barındırır.

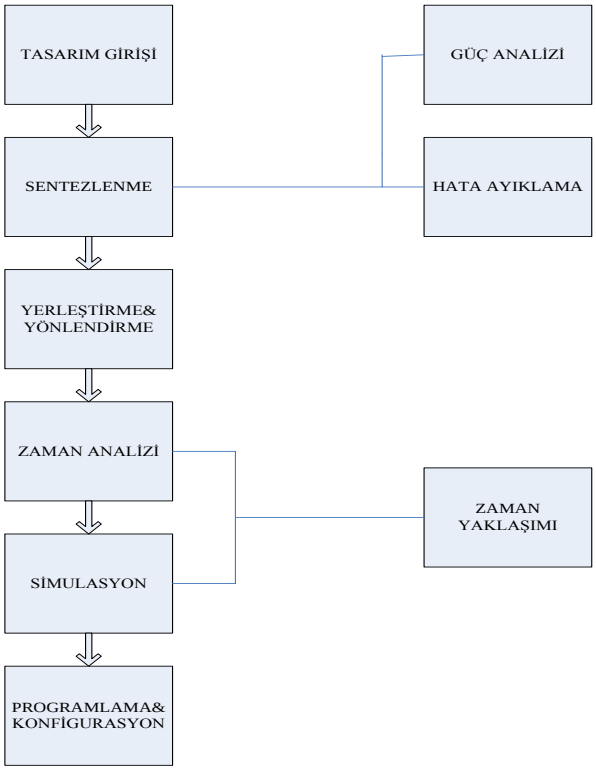


Şekil 2. 26.Sonlu Durum Makinesi (FSM) Şematik Yapısı [38]

2.4.Quartus II Yazılımı

CAD yazılımları FPGA entegresini VHDL, Şematik ve diğer tasarım yöntemleri kullanarak programlamak için kullanılır. Altera firmasının Quartus II yazılımı bu tez

çalışmasında kullanılan ilgili yazılımdır. Quartus II yazılımı ve tasarım akışı şu şekildedir [40].



Şekil 2. 27.Quartus II Yazılımda Tasarım Akışı [40]

2.4.1.Tasarım Girişi: VHDL dili veya şematik olarak istenen tasarımın girdisi sağlanır.

2.4.2. Sentezlenme: Proje dosyası olarak oluşturan tasarımı yapılan kodlar veya şematik yapı lojik kapıları seviyesinde sayısal devreye dönüştürülür.

2.4.3.Yerleştirme&Yönlendirme: Yerleştirme&Yönlendirme (Fitting&Routing) işlemi, tasarımı yapılan ve sentezlenen devrenin lojik kapı seviyesinde FPGA içinde nereye yerleştirileceğini ve bağlantı yapılarının gerçeklendiği bölümdür.

2.4.4. Zaman Analizi: Devre üzerindeki gecikme zamanının ve böylece devrenin performansının hesaplandığı bölümdür.

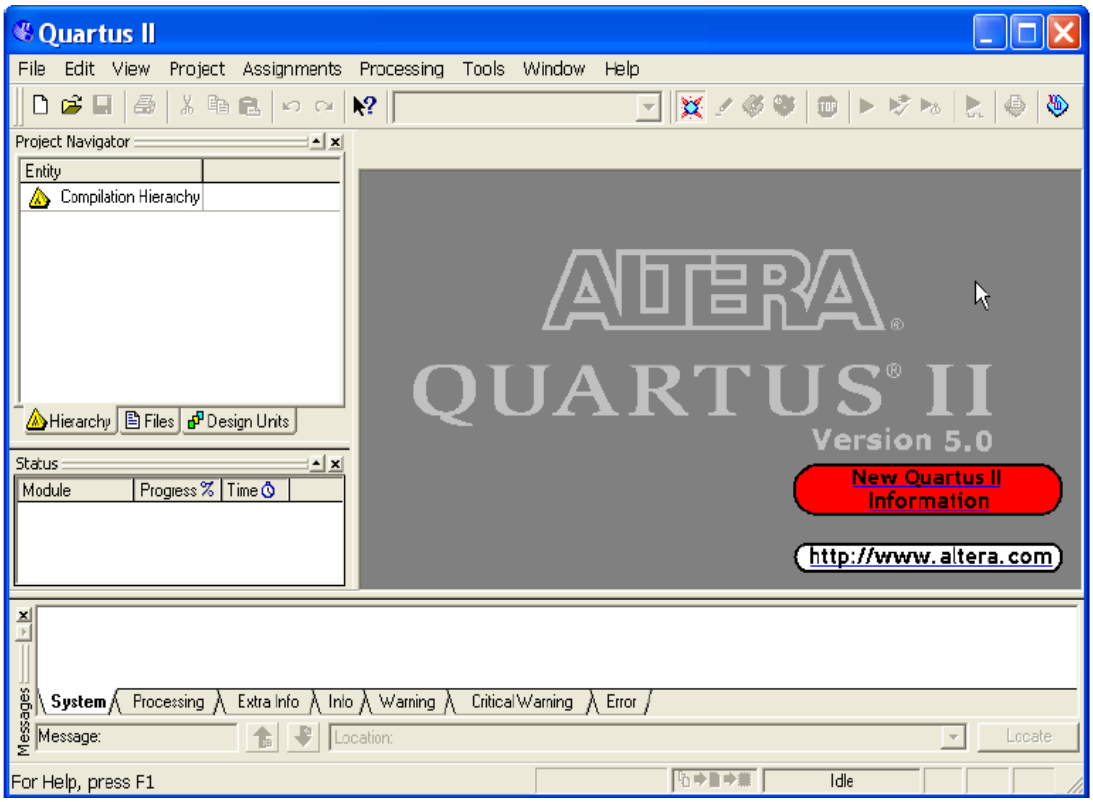
2.4.5.Simulasyon: İlgili devrenin doğru çalışıp çalışmadıgını ve zaman devrenin zamanlama sonuçlarının doğru olup olmadığını görmek için simulasyon kullanılır. Bu tez çalışmasında Mentor Graphic firmasının ModelSim simulasyon program kullanılacaktır.

2.4.6.Programlama&Konfigurasyon: Tasarlanan devre FPGA entegresinin içine gönderilir ve entegre programlanmış olur.

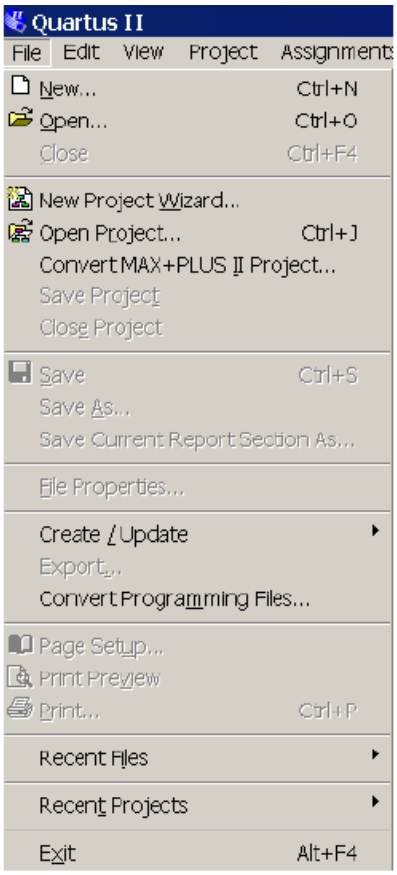
2.4.7.Güç Analizi: FPGA entegresinin ve içine gömülen devrenin oluşabilecek güç harcamaları konuşunda bilgi veren tasarım akışı bölümüdür.

2.4.8.Hata Ayıklama: Tasarımı yapılan devrenin hata kodlarının veya şematik düzeydeki hataların ayıklandığı tasarım akışı bölümüdür [40].

Quartus II yazılımında proje dosyası oluşturmaktan tasarım ve programlama surecine kadar olan ilgili adımlar şöyledir.



Şekil 2. 28.Quartus II Yazılımı Genel Penceresi [40]

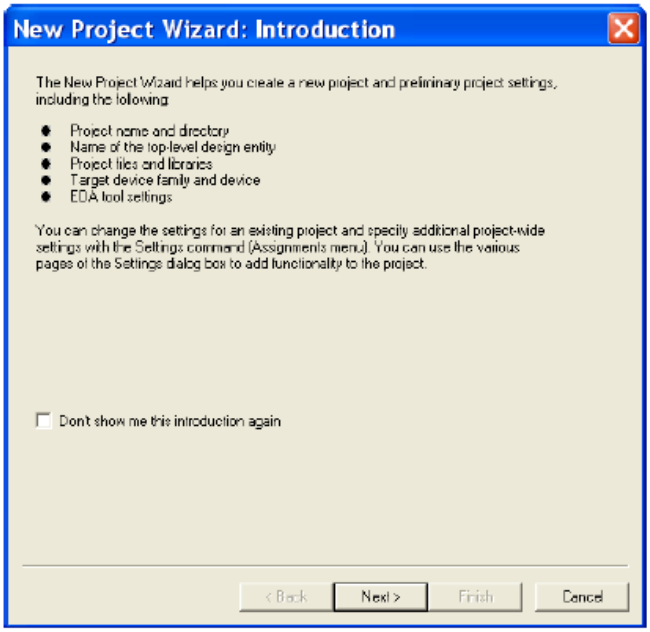


Şekil 2. 29.Quartus II Yazılımı Dosya Menüsü [40]

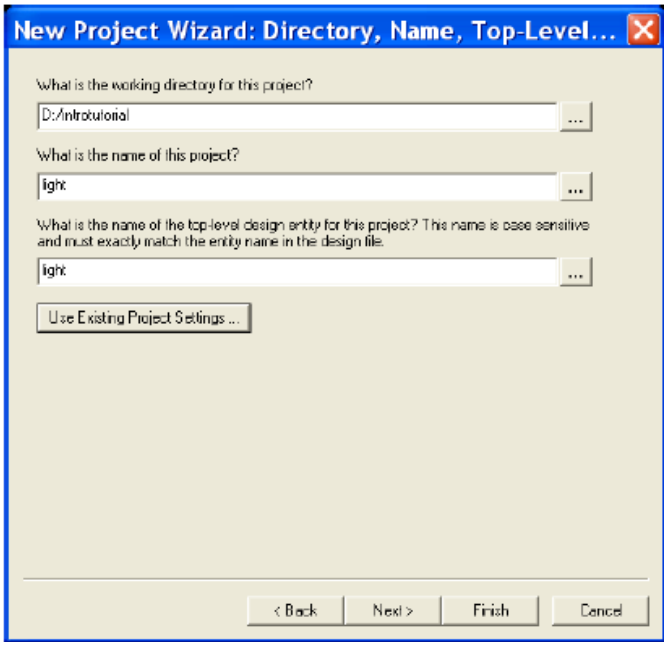
2.4.9. Yeni Bir Proje Dosyası Oluşturmak

Quartus II yazılımında proje dosyası oluşturmak için şu adımlar izlenir:

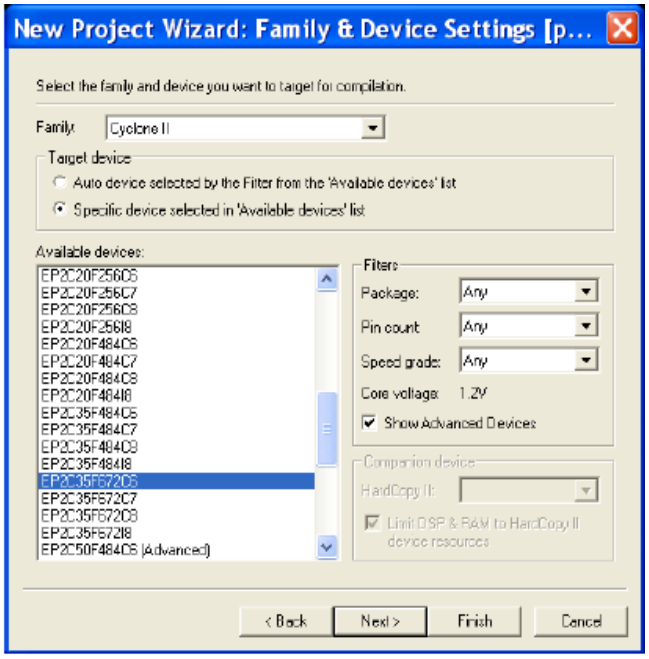
Select File > New Project Wizard [40].



Şekil 2. 30.Quartus II Yazılımı Yeni Proje Sihirbazı [40]



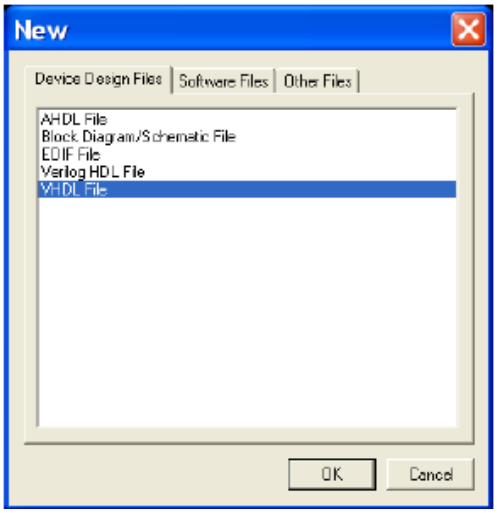
Şekil 2. 31.Quartus II Yazılımda Yeni Proje Dosyası Oluşturma [40]



Şekil 2. 32.Quartus II Yazılımda Aile ve Cihaz Seçimi [40]

2.4.10.Quartus II Yazılımda Text Editörü

Quartus II yazılımda proje dosyası oluşturduktan sonra text editöründen VHDL, Şematik, Verilog uzantılı dosyalar oluşturmak suretiyle kod yazılımının yapılacağı ekran açılmış olur [40].



Şekil 2. 33.Quartus II Yazılımda VHDL Dosyası Oluşturma [40]

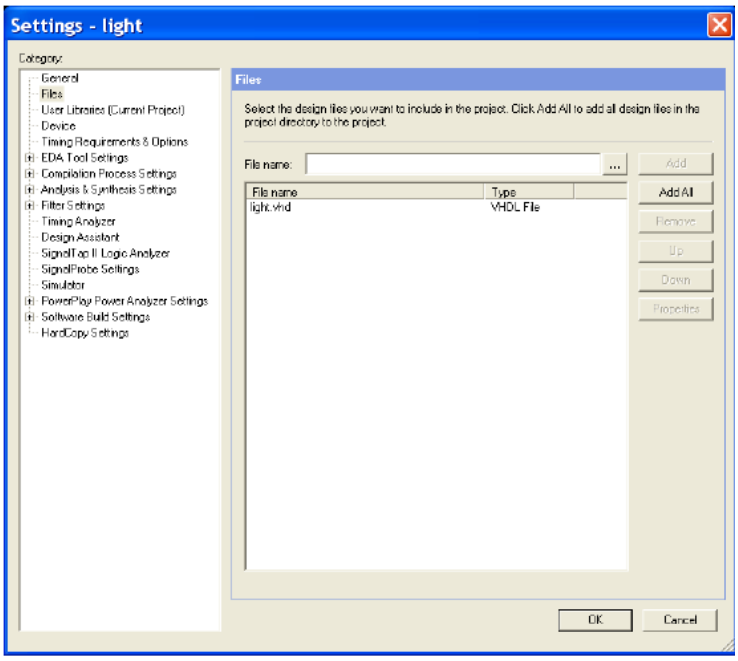
Şekil 2. 34. Quartus II Yazılımda .vhd Uzantılı Text Editörü [40]

Quartus II yazılımda VHDL kodu ile oluşturulmuş olan dosyalar .vhd uzantısı ile kaydedilir [40].

2.4.11.Proje Dosyası Eklemek

Quartus II yazılımda proje dosyası oluşturmak için şu adımlar izlenir:

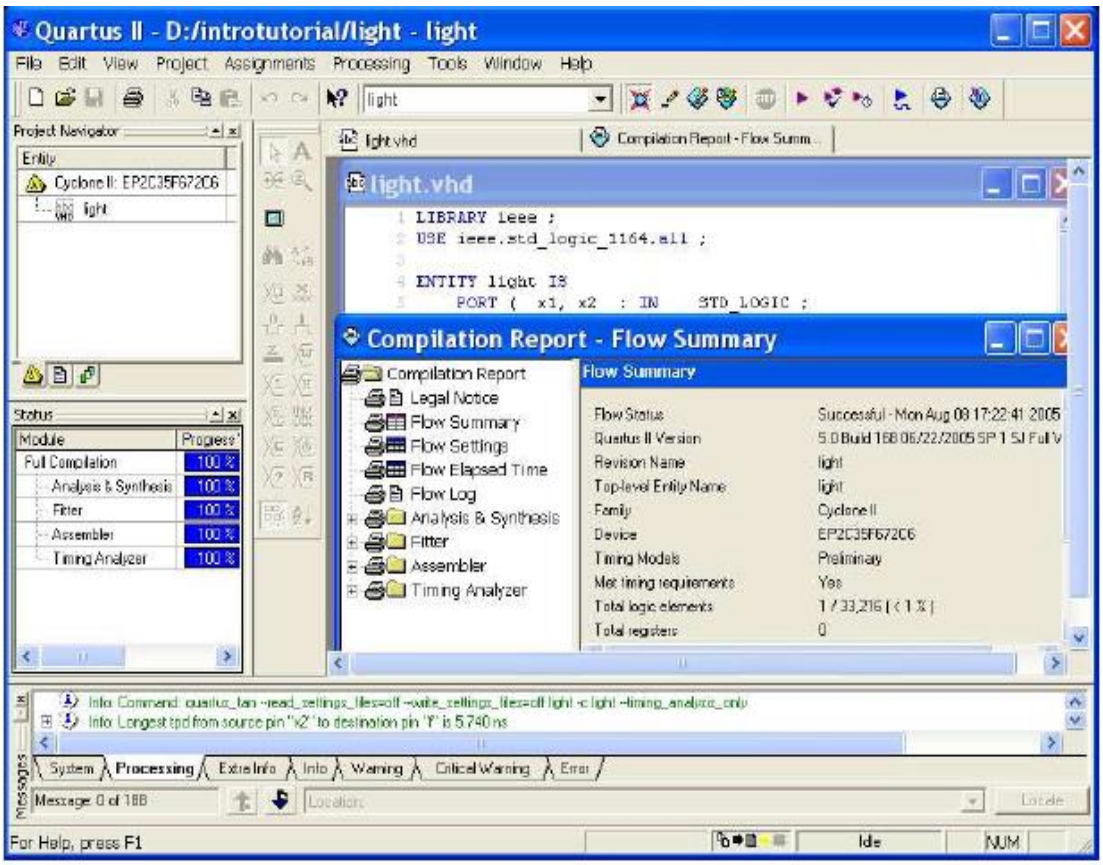
Assignments >Settings>Files>Project>Add/Remove[40].



Şekil 2. 35. Projeye Tasarım Dosyası Ekleme Penceresi [39]

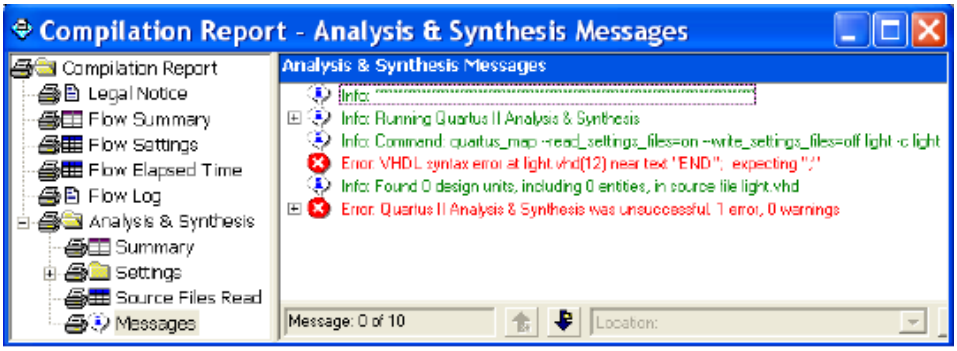
2.4.12.Tasarımı Yapılan Devrenin Derlenmesi

VHDL kodu ile tasarımi yapılan devrenin derlenmesi şu adımlarda gerçeklesir: Processing > Start Compilation sekmelerini seçerek veya ikonu üzerine tıklayarak derleme işlemi başlatılır [40].



Şekil 2. 36.Quartus II'de Başarılı Bir Derleme Sonucu Ekran Çıktısı [40]

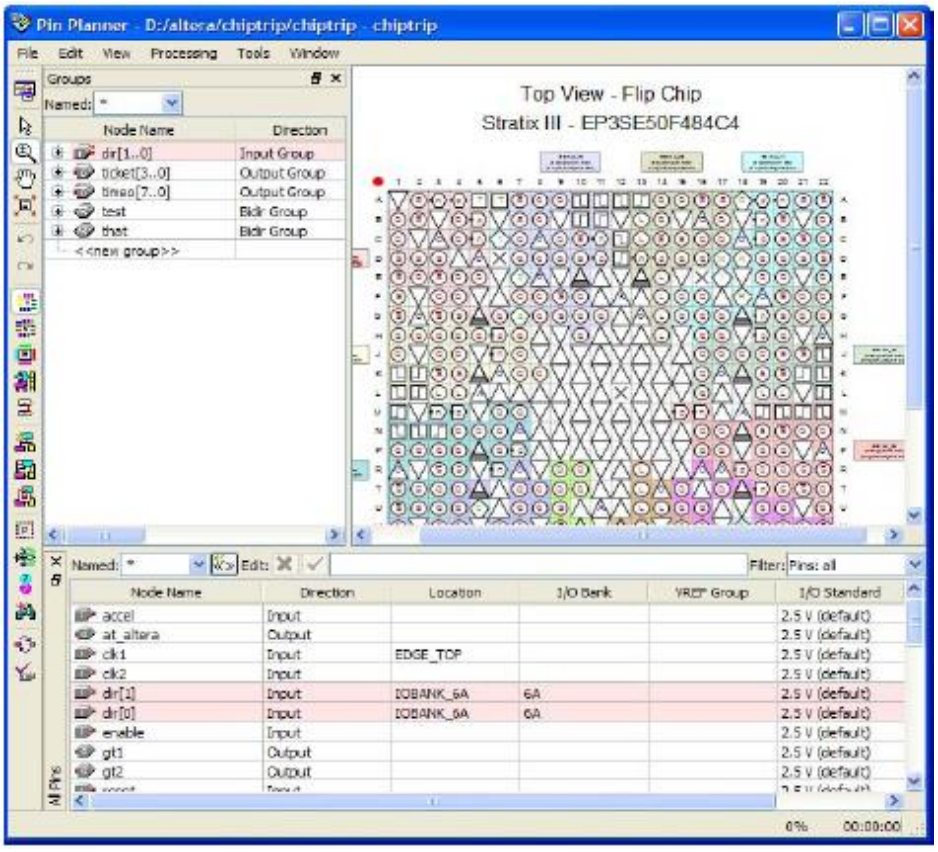
Derleme işlemi sonucunda derleme raporu derlemesi yapılan tasarım ile ilgili bilgiler verir. Bunların içinde devreyi oluşturacak lojik eleman sayısı, programlanacak cihazın ailesi ve numarası gibi bilgiler içerir. Derleme sonucu hata oluşması durumunda derleme raporu hatanın hangi satırda olduğunu ve hatanın tipini de yine gösterir [40].



Şekil 2. 37.Quartus II'de Derleme İşlemi Sonrası Hata Penceresi [40]

2.4.13.PIN Atamaları

Quartus II yazılımda seçilen cihaz üzerinde PIN atamasi yapmak için. PIN atama editörü şu adımlarla seçilir: Assignments > Pins [40].

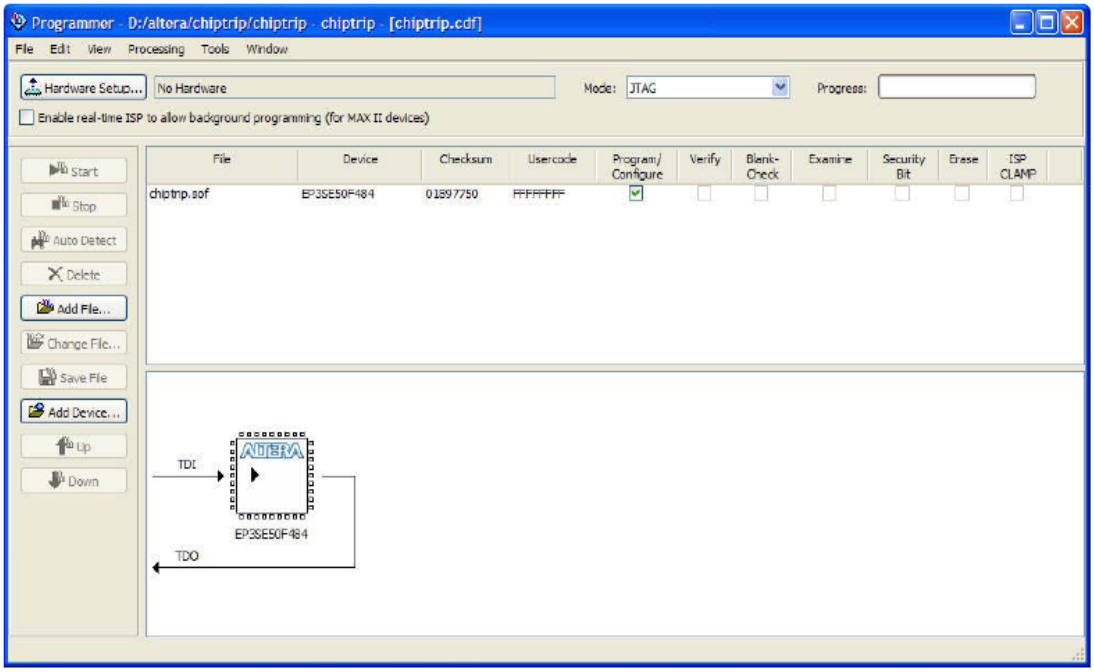


Şekil 2. 38.Quartus II Yazılımında PIN Editörü [39]

PIN atama editöründe giriş (input)-çıkış (output) atamaları ENTITY (Varlık) içinde belirtilen Node Name (Dugum ismi) olarak adlandılır. Direction (Yön)-Location (Yer) sekmeleri ile ataması yapılan PIN'lerin giriş-çıkış PIN'i ve FPGA üzerindeki yerleşiminin neresi olacağı gibi atamalar yapılır [40].

2.4.14.FPGA Programlama

Devre kartı üzerindeki FPGA entegresi Quartus II yazılımda JTAG isimli arayüz programı kullanılarak USB üzerinden programlanır. Quartus II yazılımda şu adımlar izlenir: Tools > Programmer [40].

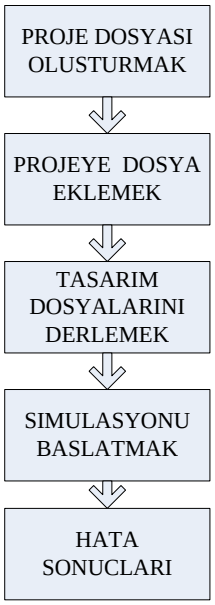


Şekil 2. 39.Quartus II Yazılımda Programlayıcı Penceresi [39]

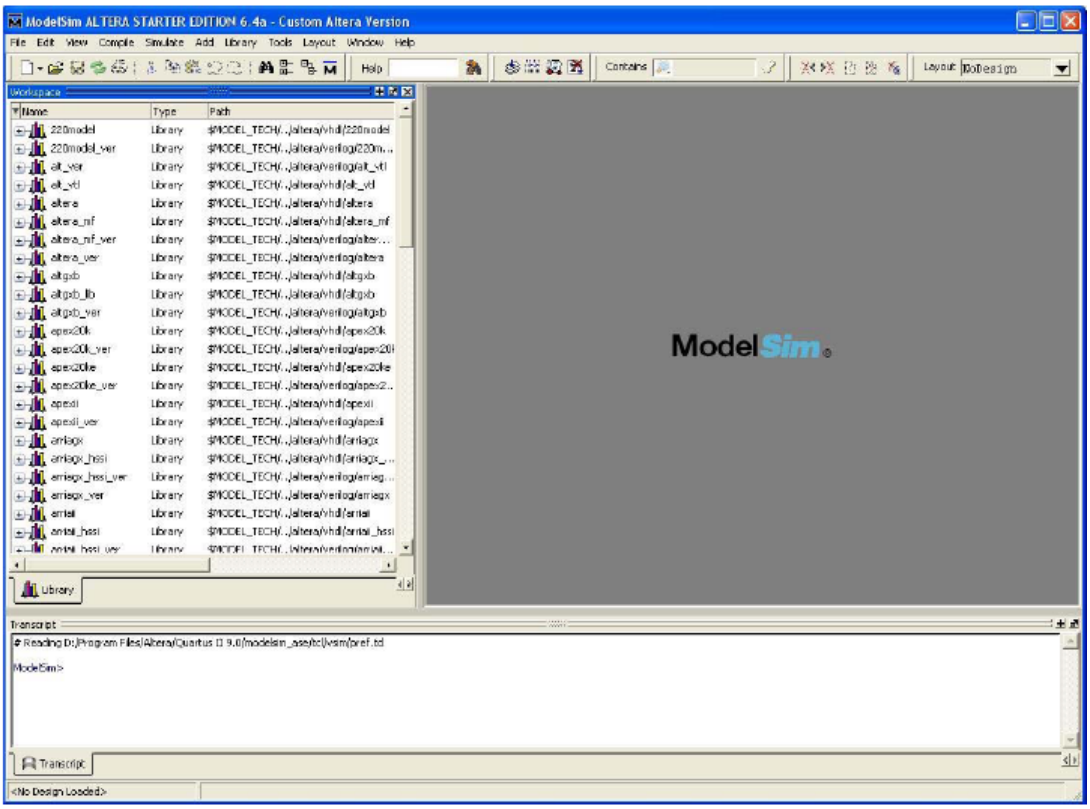
2.5. ModelSim

ModelSim simulasyon programı Mentor Graphics tarafından VHDL, Verilog dillerini kullanarak tasarımı yapılan dijital yapıların simulasyonlarını yapmak için üretilmiştir.

ModelSim programında proje akışı şu şekildedir:



Şekil 2. 40.ModelSim Programında Proje Akışı [41]

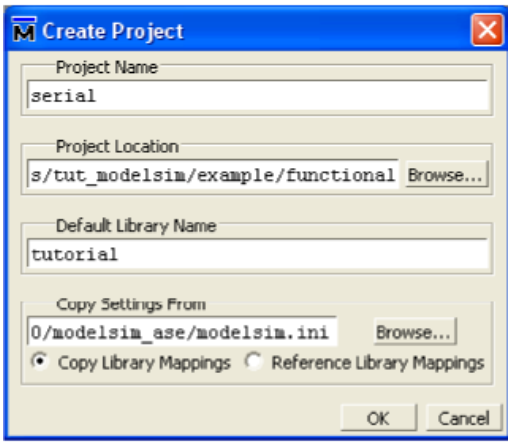


Şekil 2. 41.ModelSim Programı Ana Ekran Arayüzü [42]

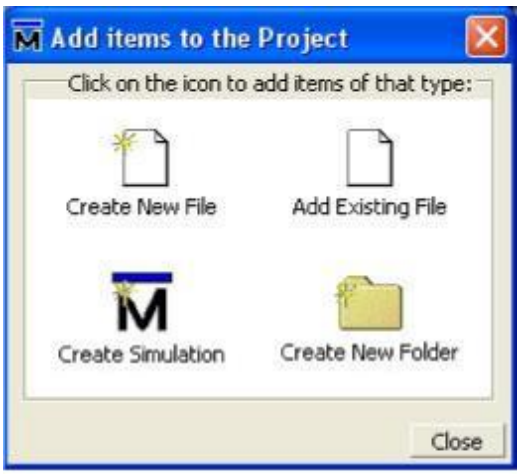
2.5.1.ModelSim'de Proje Dosyası Oluşturmak

ModelSim programında proje dosyası oluşturmak için şu adımlar izlenir :

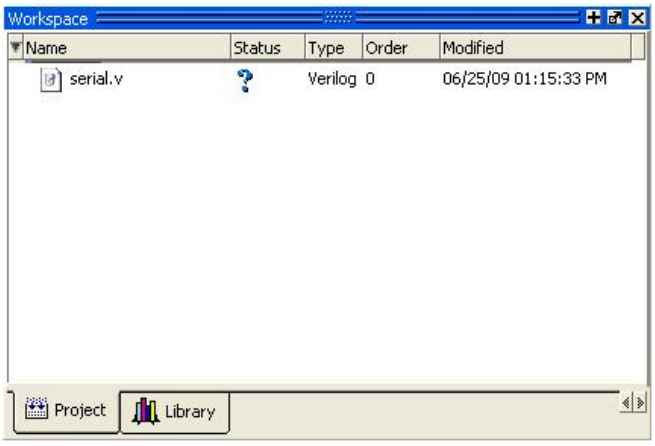
File > New > Project



Şekil 2. 42.ModelSim'de Proje Dosyası Oluşturmak [42]



Şekil 2. 43.ModelSim'de Proje Dosyasına Dosya Eklemek [42]



Şekil 2. 44.ModelSim'de .vhd Uzantılı Örnek Bir Dosya [42]

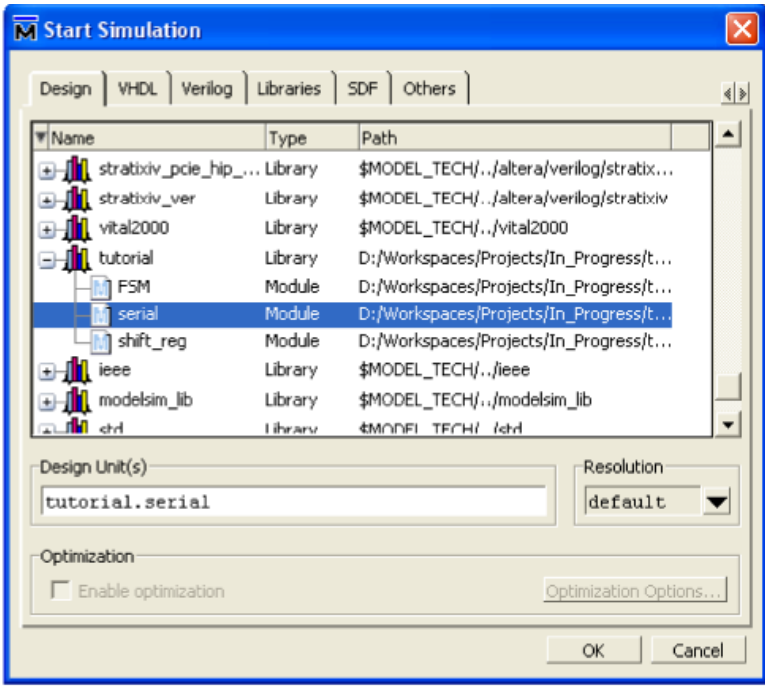
2.5.2.ModelSim'de Derleme ve Simulasyon

ModelSim programında derleme ve simulasyon işlemi şu adımlarda gerçekleşir:

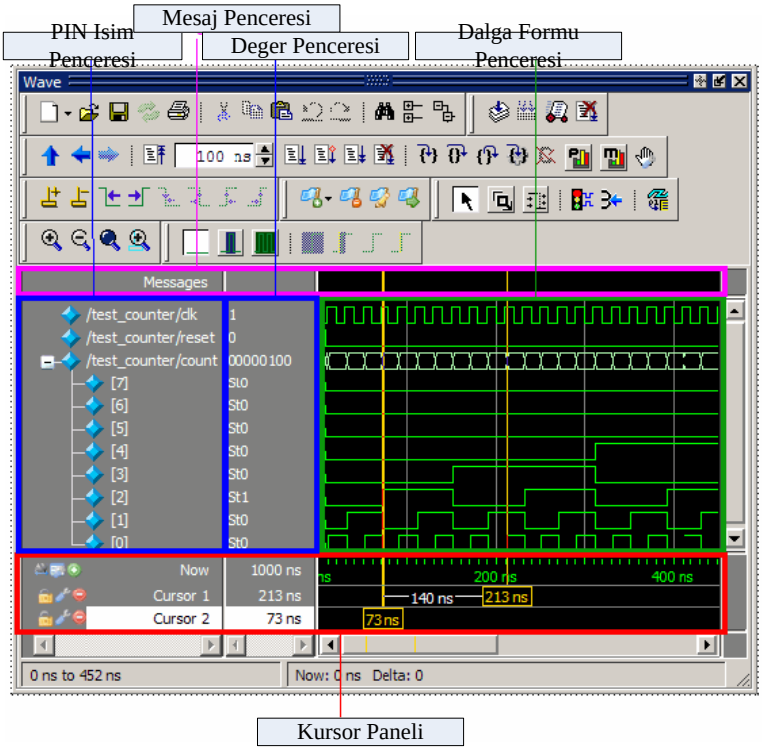
Compile > Compile All

Derleme işlemi sonrası hata olması durumda kaynak penceresi (source window) üzerinden ilgili hatalar bulunup yeniden derleme işlemi gerçekleştirilir. İlgili tasarımda hata olmaması durumunda simulasyon işlemine geçilir ve şu adımlar izlenir:

Simulate > Start Simulation [41].



Şekil 2. 45.ModelSim'de Simulasyon Başlatma Penceresi [42]



Şekil 2. 46.ModelSim'de Simulasyon Penceresi Sonuç Çıktı Penceresi [42]

3.SERİ HABERLEŞME ve RS232 PROTOKOLÜ

Telekom Endüstrisinde ve bilgisayar bilimlerinde her saniye tek bir bitin haberleşme kanalları veya bilgisayar yolları üzerinden aktarılmasına seri haberleşme denir. Paralel haberleşmede ise veriler-bitler aynı anda aktarılması yönüyle seri haberleşmeden farklıdır.

Seri haberleşme, paralel haberleşmenin maliyet ve eş zamanlı olması yönüyle zorluklar taşıdığı sistemlerde, özellikle bilgisayar ağlarında kullanılır.

Son yıllarda seri bilgisayar veri yolları paralel veri haberleşmesinin üstünlügünü elde edip düşük mesafelerde sinyal entegrasyonu ve hızı açısından önemli mesafeler kat etmiştir.

Terim olarak seri haberleşme denince IBM PC'ler için oluşturalan RS232 seri haberleşme protokolu kastetilmistir. Pratik olarak bütün uzun mesafe haberleşmelerinde seri haberleşmenin en önemli kullanılma sebeplerinden birisi veri iletişiminin paralel haberleşmeye göre daha ucuz olması aynı zamanda bir çok kablo kullanarak yapılan paralel haberleşmenin oluşturdugu karışıklık giderilmiş olur.

Klavye, Mouse, USB, Ethernet gibi cihazlarin altyapısını da yine seri haberleşme protokolleri oluşturur.

Birçok iletişim sistemi aynı PCB bordu üzerinde iki entegre arasında veri haberleşmesi için bord üzerindeki yolları kullanır. Entegre devreler arasındaki bu iletişim paralel olarak gerçekleşirse entegre üzerinde daha fazla PIN giris-çıkışı gerektirir. Bu da entegrenin maliyetini arttırır. Hızın çok önemli olmadığı durumlarda seri haberleşme entegrenin maliyetini azaltmak açısından önemlidir. Entegre üzerinde en düşük maliyetli seri haberleşme protokollerinden bazıları şunlardır: SPI,I^2C,UNI/O ve 1-Wire[43].

"Seri haberleşme paralel haberleşmeye göre daha zordur". "Seri haberleşmede ilgili verinin kullanılabilmesi için öncelikle seri verilerin paralele veya paralel verilerin seriye

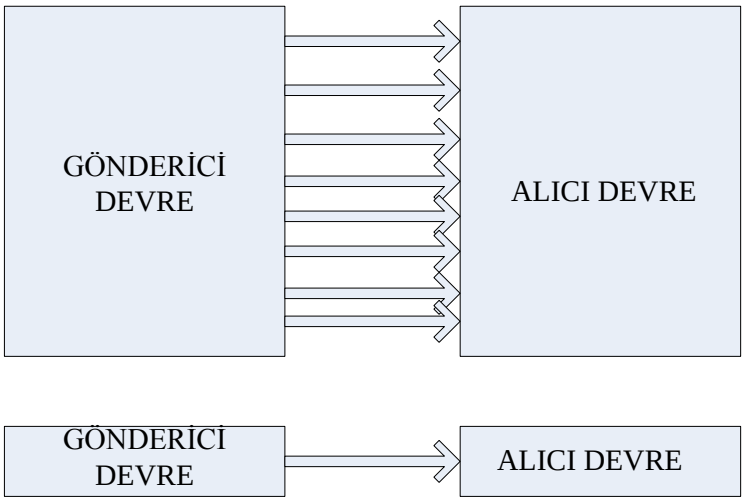
dönüştürülmesi gerekir". "Bilgisayar sistemlerinde bunun için UART (Universal Asynchronous Receiver Transmitter) entegresi kullanılır."

"Seri port haberleşmesinde lojik '1' seviyesi -3 ile -25 volt ve lojik '0' seviyesi ise +3 ile +25 volt arasında iletilir. Buna karşın paralel portta lojik '1' seviyesi 5 V ile lojik '0' seviyesi 0 V olarak iletilir. Bu nedenle seri port 50 V maksimum voltaj değişim aralığına sahip iken, paralel port 5 V maksimum aralığa sahiptir. Bundan dolayı kabloda oluşan kayıp seri kablolarda, paralele göre çok önemli değildir. " [44].

3.1.Seri Haberleşmenin Temel Esasları

"Bir mikroişlemci, dış dünya (Hafiza ve I/O birimleri) ile genelde 8 bitlik parçalarla (8,16,32,64) haberleşir. Bu şekildeki veri aktarımı paralel veri aktarımı olarak adlandırılır. Bazı durumlarda, örneğin PC'nin yazıcı ile haberleşmesinde, veri yolundan 8 bit veri ile paralel haberleşme yapılır." [44].

"Eğer mesafe uzunsa paralel veri aktarımı pek uygun değildir. Ayrıca, 8 bit veri yolu pahalıdır. Bu gibi durumlarda seri haberleşme daha uygun olur. Tek bir veri hattının kullanıldığı bu tür haberleşmenin ucuz olmasının yanında, iki farklı şehirde bulunan iki bilgisayarın telefon hattı üzerinden, bu yöntemi kullanarak, haberleşmesi de mümkün olur."[44].



Şekil 3. 1.Paralel ve Seri Haberleşme [44]

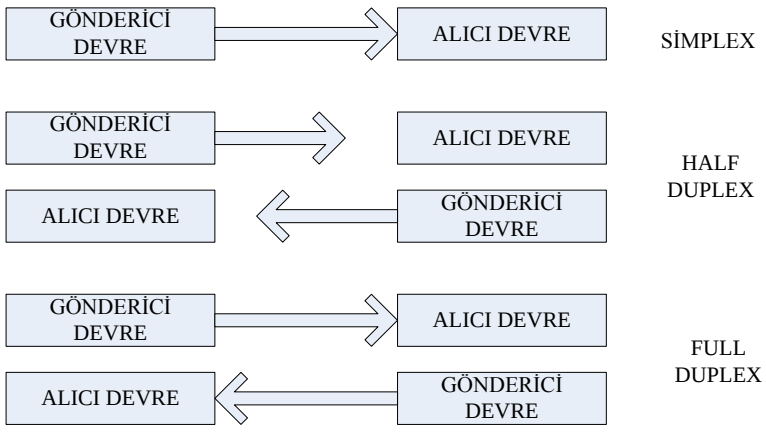
3.2.Senkron ve Asenkron Haberleşmenin Temelleri

"Seri veri haberleşmesinde senkron ve asenkron iki yöntem kullanılır. Senkron haberleşmede, bir anda bir veri blogu aktarımı (karakterler dizisi) yapılırken, asenkron haberleşmede bir anda sadece bir byte iletilir."

"Her iki haberleşme yöntemi yazılım ile gerçeklestirilebilir. Bununla beraber, bu şekildeki haberleşmede programlar uzun ve zor olacağı için, genelde seri veri iletişiminde özel tümdevreler kullanılır. Bu tümdevreler yaygın şekilde, UART (Universal Asynchronous Receiver Transmitter) olarak anılır."[44].

3.3.Seri Veri Aktarım Kanalları Temelleri

"Seri veri iletimi tek yönde oluyorsa, PC'den yazıcıya olduğu gibi, bu veri iletimi simplex olarak adlandılır. Veri, hem gönderilip hem alınabiliyorsa bu yönteme duplex denir. Bu iletişimde eğer veri bir anda sadece bir yönde aktarılabiliyorsa half duplex, aynı anda her iki yönde aktarilabiliyorsa full duplex olarak adlandırılır. Full duplex haberleşmede, bir hat gönderme ve bir hatta almak için olmak üzere, toplam iki tane hat kullanılır. "



Şekil 3. 2.Seri Veri Aktarım Yöntemleri [44]

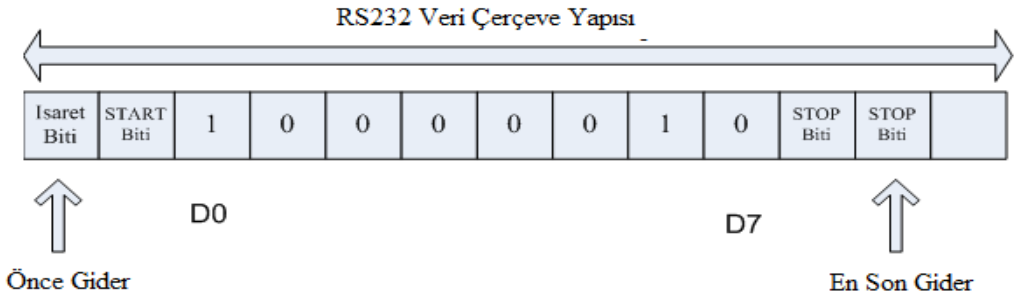
3.4.Asenkron Seri Haberleşme Temelleri

"Seri veri haberleşmesinde, veri gönderen ve alan uçların belli kurallara göre haberleşmesi gerekir. Protokol olarak adlandırılan bu kurallar, verinin nasıl

paketleneceği, bir karakterdeki bit sayısı ve verinin ne zaman başlayıp bitecegi gibi bilgileri belirler.”

“Asenkron seri veri haberleşmesi, karakter-tabanlı iletişimlerde yaygın olarak kullanılır. Buna karşın blok-tabanlı veri aktarımları, senkron yöntemi kullanir. Asenkron yöntemde, her gönderilecek karakter, başlama (START) ve bitme (STOP) bitlerinin arasına yerleştirilir. Bu işlem çerçeveleme (framing) olarak adlandırılır. Asenkron haberleşmedeki veri çerçeveleme işleminde, ASCII karakterler bir START ile bir STOP bit’i arasına yerleştilir. START biti her zaman bir bittir, STOP biti ise bir veya iki bittir.”

“START biti her zaman lojik 0(düşük), STOP biti lojik 1’dir (yüksek). Verilen örnekte ASCII ‘A’, ikili olarak 01000001, START biti ile 2 STOP bit arasında çerçevelenmiştir. Seri iletişimde önce en düşük değerli bit D0 (LSB) dışarı gönderilir.”[44].



Şekil 3. 3.ASCII ‘A’ (41H) Karakterinin Çerçevesi [44]

“Asenkron seri haberleşmede, seri veri aktarımının yapılmadığı durumda, sinyal seviyesi 1’dir (yüksek) ve bu durum işaret (mark) olarak adlandırılır. Lojik 0 (düşük) ise boşluktur (space). Veri iletimi bir START bit’i ile başlar ve bunu takiben D0 (LSB) daha sonra D7’ye (MSB) kadar geri kalan bit’ler ve son olarak ‘A’ karakterinin sonunu belirten 2 STOP bit’i gelir.”[44].

“Asenkron seri haberleşmede kullanılan tümdevreler ve modemler 5,6,7 ve 8 bit veri uzunlukları için programlanabilir. Veri uzunluğuna ek olarak, 1 veya 2 tane STOP bit’i kullanılır. Eski sistemlerde ASCII karakterler 7 bit idi. Yeni uzatılmış ASCII karakterler yüzünden, bir ASCII karakter için 8-bit gerekir. ”[44].

“Eski sistemlerde 2 STOP biti kullanılır iken günümüzde 1 STOP biti kullanılır. ”[44]

“Bir seri veri haberleşmesinde, iletilen verinin dışında fazladan kullanılan bit’ler bir fazladan zaman (overhead) ve yük oluşturur. ASCII karakterin iletiminde, eğer 2 STOP bit’i kullanılıyorsa, her bir 8 bit ASCII kodu için toplam 11 bit iletilir. Bu da her 8-bit için iletişim hattinda fazladan 3 bit veya %30 ek yük demektir. ”[44].

“Bazı durumlarda, veri bütünlüğünü korumak için karakter byte’inin eşlik (parity) bit’i veri çerçevesine eklenir. Her karakter için START ve STOP bit’lerine ek olarak bir tek eşlik bit’i eklenir. Eşlik bit’i tek (odd) veya çift (even) olur. Tek-eşlik bit durumunda, eşlik bit’i dahil, veri bit’lerinin sayısı bir tek sayıdır. Benzeri şekilde, çift-eşlik bit durumunda, eşlik bit’i dahil veri bit’lerinin sayısı bir çift sayıdır. Eğer sistem eşlik bit’i gerektiriyorsa, eşlik bit’i verideki en değerli bit’ten (MSB) sonra gönderilir. ” [44].

3.5.Veri Aktarım Hızı Temelleri

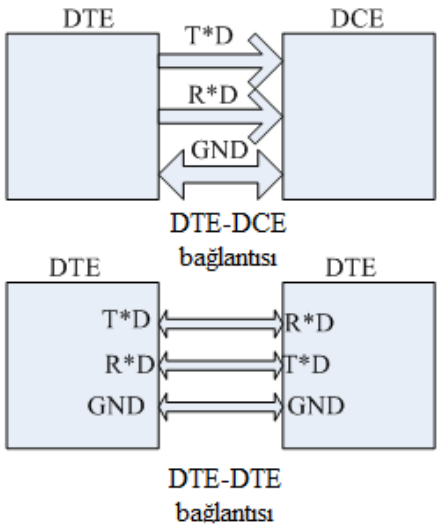
“Seri veri haberleşmesinde veri aktarım hızı **saniyedeki bit sayısi (bps-bit per seçond)** olarak belirtilir.Veri aktarım hızını belirtmede diğer yaygın olarak kullanılan İngilizce terim **baud rate’**tir. Bununla beraber, bu iki ifadenin birbirine eşit olması gerekmez. Çünkü **baud rate** bir modem terminolojisidir ve saniyede sinyaldeki değişim sayısı olarak tanımlanır. İletişim teli düşünüldüğünde **bps** ve **baud rate** aynıdır.”[44].

“Bir bilgisayarın seri veri transfer hızı haberleşme port’larına bağlıdır. Eski IBM PC/XT 100 ile 9600 bps hızlarında veri transfer edebilmekteydi. Bununla beraber, yeni PC’ler 19200 bps gibi yüksek hızlara çıkabilmektedir. Asenkron veri haberleşmesinde, baud rate genellikle 100000 bps ile sınırlıdır.”[44].

3.6.Veri Haberleşme Sınıfları Temelleri

"Seri haberleşmeyi kullanan cihazlar iki sınıfa ayrılır. Bunlar **DTE (Data Terminal Equipment)** ve **DCE (Data Communication Equipment)** olarak adlandırılır. DTE, bilgisayar veya terminal gibi veri gönderen veya alan cihazlardır. Buna karşın DCE, modem ve yazıcı gibi veri aktaran cihazlardır. "[44].

"DTE ve DCE arasındaki en basit bağlantı, en az üç tane uç, T*D, R*D ve toprak gerektirir."



Şekil 3. 4.DTE-DCE ve DTE-DTE Bağlantıları [44]

3.7. RS232 Standartı Temelleri

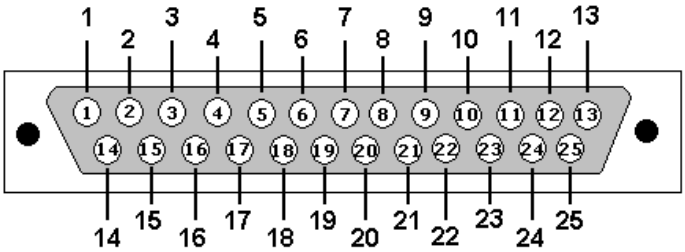
"Değişik üreticiler tarafından yapılmış veri haberleşme cihazlarının uyumluluğunu sağlamak amaçıyla, EIA (Electronics Industries Association) tarafından RS232 olarak adlandırılan standart 1960 yılında belirlendi. 1963 yılında bu ilk standart değiştirildi ve RS232A olarak adlandırıldı. Daha sonra 1965'te RS232B ve 1969'da RS232C standartları ilan edildi. "

"Günümüzde RS232 en yaygın kullanılan seri I/O arabirim standartıdır. Bu standart TTL lojik ailesinden önce belirlendiği için, giriş ve çıkış voltaj seviyeleri TTL uyumlu değildir. RS232'de lojik 1 -3V ile -25V arasında, lojik 0 +3V ile +25V

arasında tanımlanır. -3V ile +3V arası tanımsızdır. Bu yüzden TSC232 gibi voltaj çeviriciler kullanılır. Bu tümdevreler hat sürücüleri/alıcıları (line driver/receiver) olarak adlandırılır. ” [44].

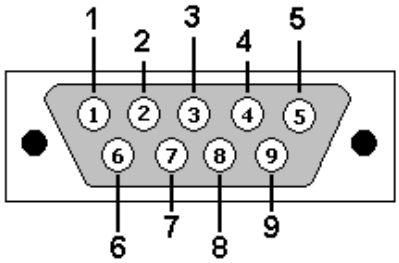
3.8. RS232 Sinyalleri Temelleri

“RS232 için DB-25 konnektorü ile erişilen toplam 25 uç tanımlanmıştır. Bu uçların hepsinde sinyal bulunmaz. Kullanılan konnektorü belirlemede, DB-25P(Plug) erkek konnektör, DB25S(Socket) dişi konnektör için kullanılır.” [44].



Şekil 3. 5.RS232 DB-25P Erkek Konnektör [45]

“Modern bilgisayarlarda bütün 25 uça gerek olmadığı için, IBM, DB-9 seri I/O standardını tanımlamıştır.” [44].



Şekil 3. 6.RS232 9-Uçlu Erkek Konnektör [46]

Tablo 3. 1.RS232 DB-9 Sinyal İsimleri [44]

Uç No(9-Uçlu)	**Uç No (25-Uçlu)**	**Sinyal Adı**	
1	8	Data Carrier Detect(DCD)	Veri Taşıma Belirleme
2	3	Receive Data(R*D)	Veri Al
3	2	Transmit Data(T*D)	Veri Gönder
4	20	Data Terminal Ready(DTR)	Veri Terminal Hazır
5	7	Signal Ground(GND)	Sinyal Topraklaması
6	6	Data Set Ready(DSR)	Veri Seti Hazır
7	4	Request To	Veri Göndermek için

		Send(RTS)	Talep
8	5	Clear To Send(CTS)	Veri Göndermek için Temizle
9	22	Ring Indicator(RI)	Telefonun Çaldığını Gösterir

"İki cihaz arasında güvenli veri iletişimin olabilmesi için, veri aktarımının düzenlenmesi gerekir. Bu amaçla seri veri haberleşmesinde senkronizyon sinyalleri kullanılır."[44]

1-"DTR (Data Terminal Ready): DTE(PC COM port'u) çıkış, modem giriş uçudur. Bu hat, modem'e, terminalin (bir PC COM port'u,dolayısi ile UART) veri iletişimi için hazır olduğunu belirtir."

2-"DSR (Data Set Ready): DTE (PC COM port'u) giriş, modem çıkış ucudur.Bu hat, terminale (bir PC Com port'u,UART), modem'in veri iletişimi için hazır olduğunu belirtir. Eğer modem herhangi bir nedenden dolayı telefon hattına bağlanamıyorsa, bu sinyal pasif yapılarak, veri göndermek veya almak için modemin hazır olmadığı PC'ye belirtir."

3-"RTS (Request To Send): DTE (PC gibi) göndereceği verisi olduğunda, bu uçla modeme haber verir."

4-"CTS (Clear To Send): RTS sinyaline cevap olarak, modem alacağı veri için yeri olduğunda, bu sinyali DTE'ye (PC UART'ına) göndererek veri almaya hazır olduğunu belirtir".

5-"DCD (Data Carrier Detect): Modem, telefon hattından bir taşıyıcıyı (carrier) belirlediğinde, bu hattı aktif yaparak DTE'ye (PC) bu durumu haber verir."

6-"RI (Ring Indicator): Telefon çaldığında,modem, DTE'ye (PC) bu sinyalle haber verir. Altı senkronizasyon sinyalinden en az kullanılanı bu sinyaldir. Çünkü telefona modem cevap verir. Eğer telefona PC'nin cevap vermesi isteniyorsa bu sinyal kullanılabilir."

“RTS ve CTS veri akışını kontrol etmede kullanılır. PC veri göndermek istediğinde, RTS hattını aktif yapmakta, buna cevaben, modem veri kabul etmek için hazır ise, CTS sinyalini gönderir. ” [44].

3.9.Veri Akış Kontrolü

“DTE-DCE arasındaki hız DCE-DCE hızından çok fazla ise, PC’nin gönderdiği veri, modem’deki buffer alanını kısa zamanda doldurduğundan bir veri akış kontrolü gerekir. Akış kontrolü için yazılım ve donanım olmak üzere iki yöntem vardır.”

“Yazılım akış kontrolünde, bazı yerlerde Xon/Xoff olarak belirtilen iki karakter kullanılır.

Xon olarak normalde ASCII 17 ve Xoff olarak 19 karakteri kullanılır. Modemin içinde veriler için sınırlı bir alan olduğundan, bilgisayar bu alanı doldurduğunda, modem Xoff karakteri göndererek, bilgisayara daha fazla veri göndermemesini belirtir. Modem, veri için yeri oldugunda bilgisayara bir Xon karakteri gönderir. Bunun sonucu bilgisayar daha fazla veri gönderir. Bu çesit akış kontrolünün avantajı, veri iletiminin yapıldığı TD/RD hatlarına ek olarak başka hatlara gerek olmaz. Bununla beraber, yavaş hızlarda olan haberleşmelerde, bu tür karşılıklı el sıkışma senkronizasyonu, iletişimi daha da yavaşlatır.”

“Donanım veri akış kontrolü, aynı zamanda RTS/CTS akış kontrolu olarak da bilinir. Bu yöntemde, yazılım yöntemindeki veri hatlarından aktarılan fazladan karakterler yerine, seri kablodaki iki hat kullanılır. Bu şekilde yapılan bir akış kontrolü veri hızını yavaşlatmaz. Bilgisayar veri göndermek istediginde RTS (Request to Send) hattını aktif yapar. Eğer modemde veri için yer var ise, CTS (Clear to Send) hattını aktif yapar ve bilgisayar veri göndermeye başlar. Eğer modem yere sahip değilse CTS sinyalini göndermez.”[44].

3.10.Diğer Standartlar

“Bir RS232 kablosunun uzunluğu arttığında, sinyalde kapasitif yük artmaktadır. Bunun sonucu yüksek veri hızları güvensiz olmaktadır. Eğer kablo uzunluğu 5 feet veya daha az olursa, RS232 ile 100000 bps veri aktarım hızı elde edilebilir. Veri

hızını ve kablo uzunluğunu arttırmak için RS232'nin elektriksel özellikleri yeniden tanımlanabilir. Bunun sonucu RS422 ve RS423 gibi yeni standartlar çıkmıştır. Bu standartlar ve karşılaştırması şöyledir." [44].

Tablo 3. 2.RS232, RS422, RS423 Kablo Uzunlukları ve Veri Hızları [44]

	RS232	**RS422**	**RS423**
Maks.Kablo Uzunlugu(ft)	50	4000	4000
Maks.Hız(baud)	20K	10M/40ft 1M/400ft 100K/4000ft	100K/30ft 10K/300ft 1K/4000ft

3.11.Seri Port Adresleri

"Bir PC açıldığında POST (Power-On Self Test) esnasında, 4 COM port'undan her biri için UART tümdevresi test edilir. Eğer COM port'ları bulunur ise, I/O port adresleri hafızada belirli yerlere yazılır." [44].

Tablo 3. 3.PC Seri Port Örnek Adres Atamaları [44]

COM Taban Adresi	**İsim**
0000:0400	COM1
0000:0402	COM2
0000:0404	COM3
0000:0406	COM4

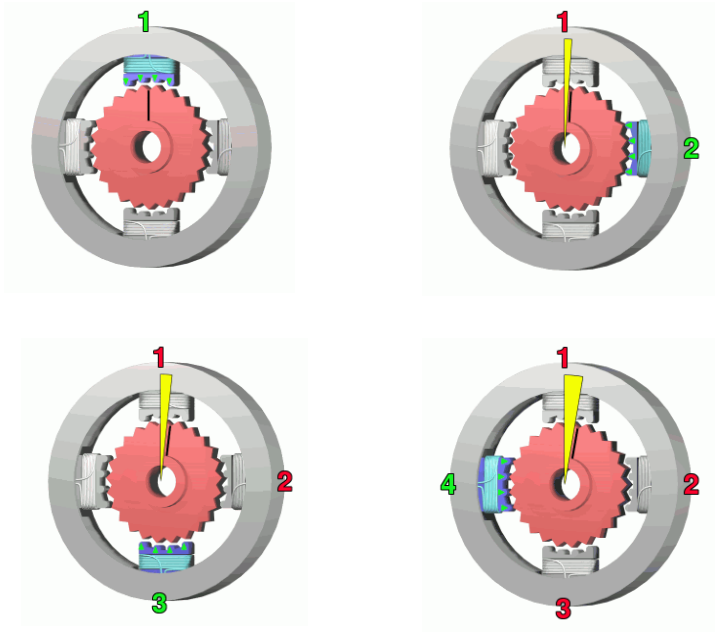
3.12.Adım Motorlar

Adım motorlar (Stepper-Stepping Motors), sayısal sinyalleri mekanik hareketlere çeviren motorlardır. Adım motorların çalışma prensibi 1920'lerden beri bilinmekte olmasına rağmen uygulamaları digital bilgisayarların kullanılmasının artması ile büyük bir artış göstermiştir. Adım motorlar, isminden de anlaşılacağı gibi lojiksel sinyalleri kontrol ederek ayrık adımlarda dönme işlemi gerçeklestirir. Adım motorların tipik uygulamaları printırlar, manyetik disklerde pozisyon kafası ve devamlı veya adım yönlü mesafe gerektiren uygulamalarda kullanılır [47].

Adım motorlar tam dönüşlerini eşit mesafelerdeki adımlarla gerçeklestiren fırçasız DC motorlardır. Motorun pozisyonu sensörlerden gelen herhangi bir geribesleme (feedback) yapısına ihtiyaç duymadan oldukça hassas bir şekilde gerçekleştirilir [48].

Adım motorlar genel anlamda üç katagoriye ayrılır: Değişken-relaktans, sabit-manyetik ve hibrid tipleri mevcuttur. Adım motorların belirli özellikleri adım motorları oldukça kullanışlı kılar. Özellikle giriş sinyalinin pulse sayısına doğrudan orantılı olarak dönmesi adım motorların en önemli özelliğidir. Bununla beraber dönüş esnasında açısal hatanın az olması da adım motorların önemli özelliklerinden birisidir. Adım motorlar başlama, durma, geri dönüş gibi sayısal giriş sinyallerine karşı hızlı cevap verme yeteneklerine sahiptirler. Adım motorların diğer bir özelligi de herhangi bir fren mekanizmasi olmadan rotor sistemlerinin durabilmesidir. Bütün bu özelliklerinin yanında adım motorlar, uygulanan sayısal sinyallerin frekansına bağlı olarak dönüş hızları değişir [47].

Şekil 3. 7.Adım Motor Örnekleri [48]



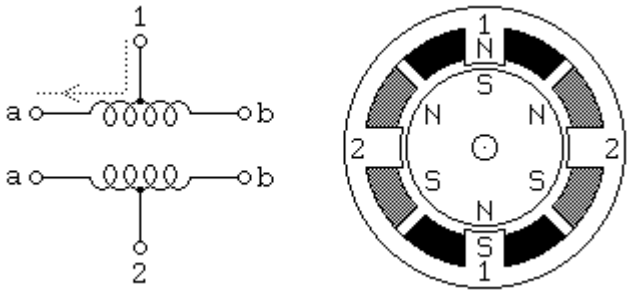
Şekil 3. 8.Adım Motorun Verilen Her Pulse'daki Adım Hareket Örneği [49]

3.12.1.İki Fazlı Adım Motorlar

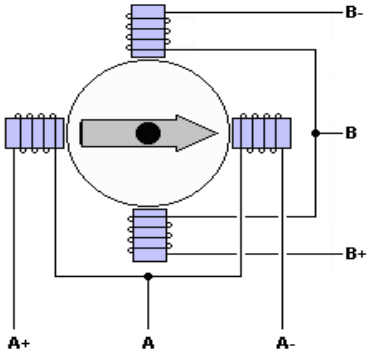
İki fazlı sarımlı adım motorlar ikiye ayrılır. Bunlar :Unipolar(Tek Kutuplu) ve Bipolar(Çift Kutuplu).

3.12.1.1.Tek Kutuplu (Unipolar) Motorlar

Tek kutuplu adım motorlar tek yönlü akım ileten motorlardır [49]. Tek kutuplu adım motorlar tek sarımlı ve her fazı ortasından ikiye ayıran yapıya sahiptir. Her yöndeki manyetik alanın yönü için sarımların her parçası anahtarlanır. Manyetik kutuptaki bu düzen akım yönünün ters çevrilmesine gerek olmadan manyetik kutuplar ters çevrilebilir. Komutasyon devresi her sargı için oldukça kolaydır. Her sargı tek bir transistör kullanılarak surülebilir. Tipik olarak verilen fazda her sargının ortasının birleşik olması oldukça yaygındır. Her faz için 3 uç iki faz için 6 uç söz konusudur. Bazı adım motorlarda iki fazın ortak noktaları içerisinden bağlı olduğu için adım motordan 6 uç yerine yalnızca 5 uç çıkması da mümkündür. Bu nedenden ötürü 6 veya 5 uçlu adım motorlar vardır [48].



Şekil 3. 9.Tek Kutuplu Adım Motor Örnek İç Yapısı [49]



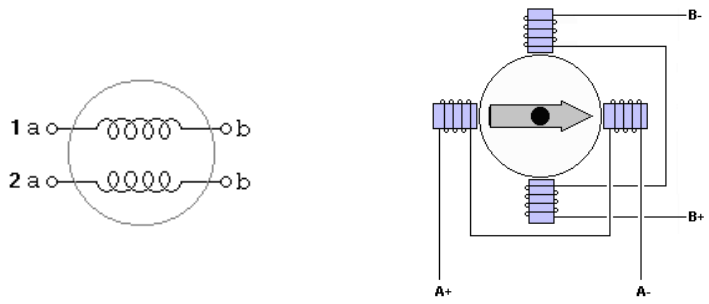
Şekil 3. 10.Tek Kutuplu Adım Motor [50]

3.12.1.2.İki Kutuplu (Bipolar) Motorlar

İki kutuplu motorlar her fazda tek sargıya sahiptir. Magnetik kutubu değiştirmek için sargı içinden akan akım ters çevrilmelidir bu da motoru süren devrelerin daha kompleks olmasına yol açmaktadır. İki kutuplu motorları sürmek için genelde H-köprüsü (H-Bridge) denen devre yapıları kullanılmaktadır. Genelde her faz için 2 uç bulunmaktadır, herhangi bir ortak bağlantı noktası yoktur.

Bipolar motorlar, unipolar motorlardan içerisindeki sarım itibari ile çıkış torku açısından daha verimli bir yapıya sahiptir [48].

Bipolar motorlar çift kutuplu olup, her iki yönden de akım akabilen motorlardır. Bipolar motorların genelde 4 ucu bulunur [49].



Şekil 3. 11. 4 Uçlu Bipolar Motor Örnek İç Yapısı [23-24]

3.13.Adım Motorları Sürmek

Adım motorlar 2 farklı şekilde sürülebilir. Bunlar tek fazlı sürme tekniği ve iki fazlı sürme tekniğidir [49].

3.13.1.Tam Adım Sürme Tekniği

4 uçtan 2 uçun aynı anda tetiklenmesi ile sürüş tekniği gerçekleştirilir. Bu tek fazlı sürüş tekniğine göre 2 kat kuvvetli bir dönme etkisi yaratırken 2 kat fazla akım çekilmesine sebep olur [49].

3.13.2.Yarım Adım Sürme Tekniği

Yarım adım sürme işleminde, tek fazlı ve yarım adım sürme işlemleri birlikte gerçekleştirilir. Böylece her adımında $90^{o'}$ 'lik dönme yeteneğine sahip bir adım motor 45^{0}'lik dönme yeteğine sahip olur ve adım hassasiyeti artar [49].

Tablo 3. 4.Tek ve İki Fazlı Adım Motor Sürüş Yöntemleri [49]

Tek Kutuplu Tek Fazlı Sürme Tablosu				
Adım	**A1**	**B1**	**A2**	**B2**
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Tek Kutuplu İki Fazlı Tam Adım Sürme Tablosu				
Adım	**A1**	**B1**	**A2**	**B2**
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

Tek Kutuplu İki Fazlı Yarım Adım Sürme Tablosu				
Adım	**A1**	**B1**	**A2**	**B2**
1	1	0	0	1
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

4. BZK.SAU MİMARİSİ, SERİ HABERLEŞME VE KONUM KONTROLÜ

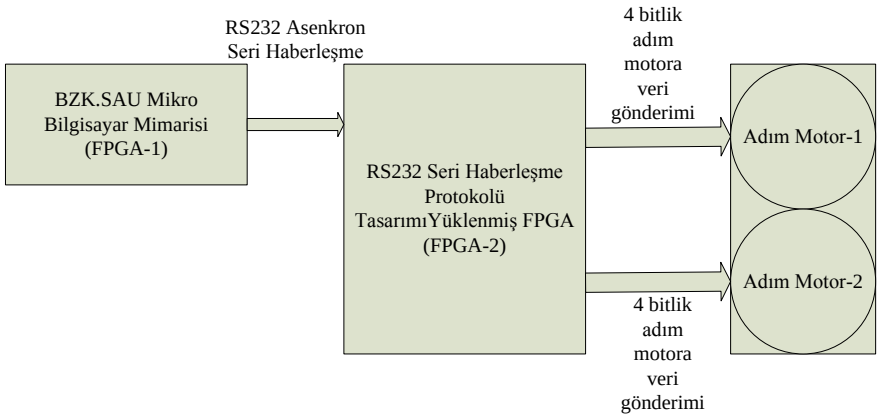
4.1.Pan/Tilt Mekanizması

Bu tez çalışmasında danışmam hocam Dr.Halit Öztekin tarafından 2009 yılında yapılan BZK.SAU isimli mikrobilgisayar mimarisi tasarımi simulatörü yüksek lisans tezi [51], 2012 yılında yine Dr.Halit Oztekin tarafından FPGA ortamına aktarılmıştır [52]. İlgili çalışmada eğitimsel amaçlı tasarlanan BZK.SAU isimli doktora tezinde modülerlik özelliği ile kullanıcının da sisteme müdahil olması sağlanmıştır.

Bu tez çalışmasında BZK.SAU mikrobilgisayar mimarisi, RS232 seri haberleşme protokolü ve BZK.SAU mikro bilgisayar içerinde yine BZK.SAU.ASSM ile yazılan kodlar ve bu sistemlerin açısal anlamda kontrol ettiği Pan-Tilt mekanizması ayrı alt başlıklar altında anlatılmıştır. Özetle alt başlıklar şunlardır:

- BZK.SAU mikro bilgisayar mimarisi genel anlatımı
- BZK.SAU mikro bilgisayar mimarisi içinde RS232 asenkron seri haberleşme Tasarımı
- BZK.SAU.ASSM dili ile tasarımı yapılan açısal anlamda kontrol edilen Pan/Tilt mekanizması

Bu tez çalışmasında amaçlanan sistem tasarımı aşağıda Şekil 4.1'de gösterilmiştir.



Şekil 4. 1.Tez Çalışmasında Üzerinde Çalışılan Sistemin Şematik Gösterimi

Tez çalışmasında FPGA-1 olarak gösterilen BZK.SAU mikro bilgisayar mimarisi yüklü FPGA ile FPGA-2 olarak ifade edilen RS232 seri haberleşme protokolü yüklü FPGA arasında bağlantısı yapılan kablo tek bir kablo yapısı ile veya 9 Pin Headerlar için standart olarak üretilmiş kablo yapısı ile iki FPGA birbirlerine bağlanmıştır. Böylece BZK.SAU içinde derlenen ve çıkış kaydedicilerine gönderilen her veri RS232 seri haberleşme protokolünü kullanarak diğer FPGA ile iletişim kurmakta ve istenen seri veri aktarım hızlarında veri gönderimi sorunsuz ve kayıpsız olarak gerçekleşmektedir.

Tasarımı yapılan RS232 asenkron seri haberleşme protokolü seri veri gönderen-alan olmak üzere iki kısım halinde VHDL dili oluşturulmuş ve BZK.SAU mikrobilgisayar mimarisine entegrasyonu tam olarak yapılmıştır.

İlgili tez çalışmasında ikinci kısım olarak ifade edebileceğimiz bölümde BZK.SAU mikro bilgisayar mimarisi içerisinde BZK.SAU.ASSM dili ile tasarımı yapılan kontrol algoritması kodlaması yapılmış olup iki eksenli bir Pan-Tilt mekanizması içinde hareket eden adım motorlar üzerinde açısal anlamda kontrol olayı gerçekleştirilmiştir. İlgili kontrol sisteminde iki eksenli kontrol için veri girişi yine BZK.SAU'nun klavye yapısından alınan yön verileri ile sağlanmıştır. Bu suretle açık döngü kontrol yapısı istenen açılarda dönmenin yanı sıra adım motorların sahip oldukları her adımları için açısal dönme özellikleri bu sistem sayesinde uygulamaya konulmuştur. İki eksenli kontrol yapısı şematik gösterimde adım motor-1 ve adım motor-2 olarak adlandırılmıştır.

FPGA-2 üzerinde bulunan GPIO-1 ve GPIO-2 pin çıkışlarına bağlı adım motor sürücü devresi ile adım motorların yönsel ve açısal kontrolü sağlanmıştır.

4.2.BZK.SAU MikroBilgisayar Mimarisi Genel Anlatımı

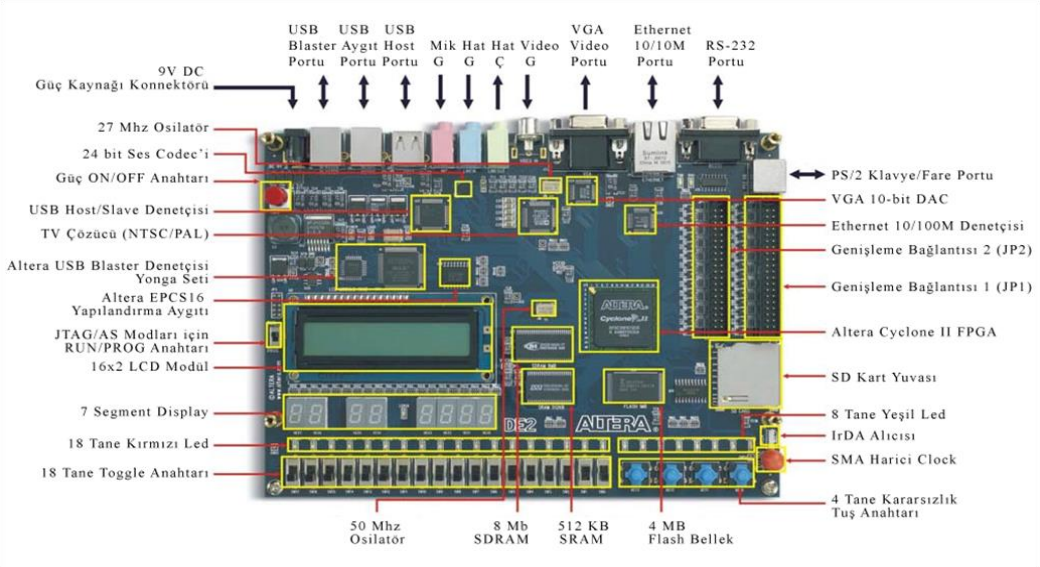
BZK.SAU isimli mikrobilgisayar tasarımına ait özellikler asağıdaki gibidir .

Tablo 4. 1.BZK.SAU Mikrobilgisayar Mimarisine Ait Özellikler [52]

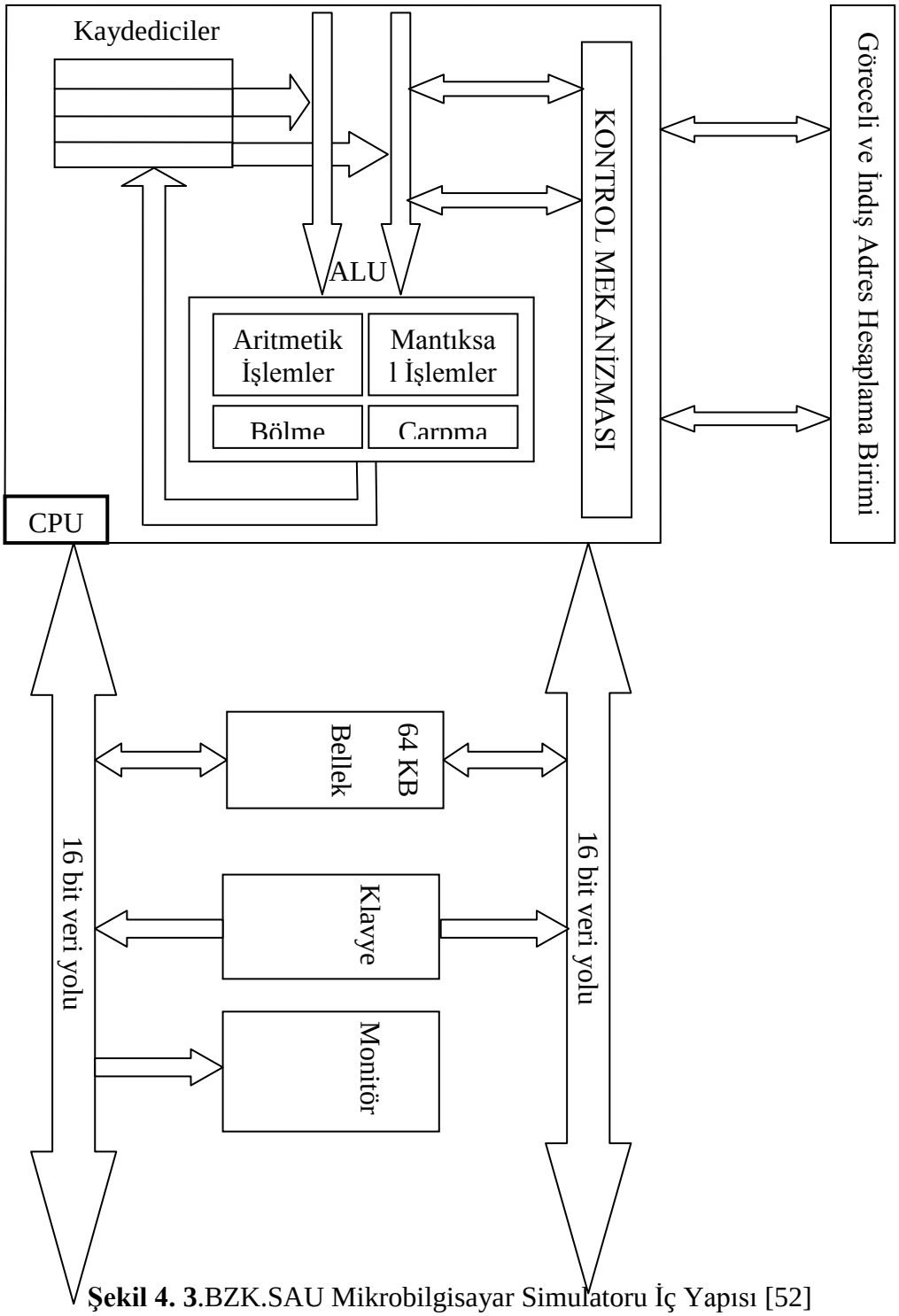
Özellik	**Açıklama**
Sistem Adı	BZK.SAU.FPGA
Sistemin İnşa Edildiği Dönanım	Altera DE2 ve CycloneIII FPGA kartları
Çıkış Birimi	VGA Monitör (640×480)
Ekran Alanı	40 sütun×24 satır(320×384)
Giriş Birimi	PS/2 klavye
Sistem Tanımlama Dili	Şematik(Donanımsal)
İşlemci Mimarisi	Von-Neumann(SISD mimarisi)
İşlemci 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	Fonksiyönel, 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	Dönanımsal
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

BZK.SAU mikrobilgisayar mimarisinde 16 bitlik veri ve adres yolu kullanılmıştır. İlgili çalışma BZK.SAU mikrobilgisayar mimarisinde 16 bitlik veri yolu 64Kbyte bellek adreslemiştir. Tasarımı yapılan BZK.SAU mimarisinde 640×480 piksel çözünürlügüne sahip VGA tipinde monitör kullanılmış, monitör donanımı sadece

metin ve tek renk(kırmızı) kipinde çalısan bir donanım olup, monitör ekranındaki her bir karakter 8×16 piksel ile temsil edilmektedir. İlgili çalışma klavye yapısı ve monitörü ile Altera DE2-70 FPGA bordu üzerinde çalışmaktadır.

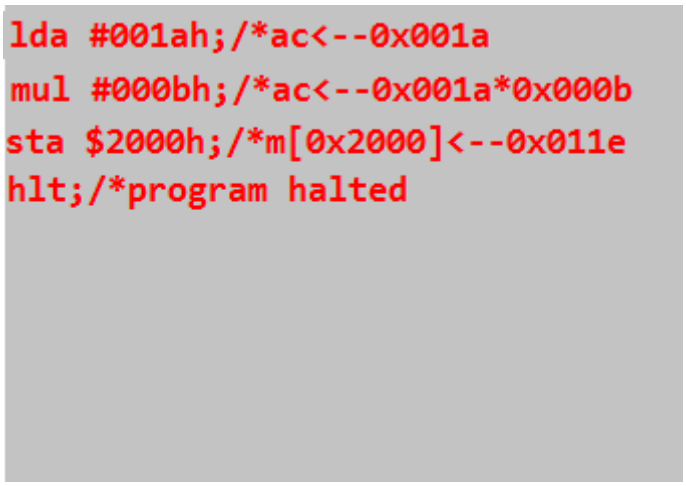


Şekil 4. 2.Altera DE2-70 Bordu [52]



Şekil 4. 3.BZK.SAU Mikrobilgisayar Simulatoru İç Yapısı [52]

BZK.SAU mikrobilgisayar mimarisi 59 komuttan oluşur ve her kodun uzunluğu 16 bittir. İlgili assembly kodlarının kodlandığı pencere Şekil 4.4'deki gibidir. BZK.SAU mimarisinde 6 adet adresleme modu mevcuttur. Bunlar tablodaki gibidir.



Şekil 4. 4.BZK.SAU Mikrobilgisayar Mimarisinde Assembly Programlama Arayüzü [52]

Tablo 4. 2.BZK.SAU'da Adresleme Modları ve Sembolleri [52]

Adresleme Modu	**Sembol**
İvedi adresleme	#
Direkt adresleme	$
Dolaylı adresleme	@
İndis adresleme	%
Göreceli adresleme	*
Doğal adresleme	

BZK.SAU mikrobilgisayar mimarisinde arayüz ekranında yazılabilecek 5 komut vardır. Bunları yazmak için gerekli olan kurallar ve komutlar şunlardır.

-" Komut adı küçük harf ile yazılmalıdır.

- Komut adı küçük yazıldıktan sonra bir boşluk bırakılmalıdır.

- Komut adı yazıldıktan ("dir" komutu hariç) sonra dosya ismi maksimum 8 karakter uzunlugunda olmalı ve dosya isminden sonra uzantısı "." karakteri ile başlamalıdır. Dosya uzantısı 3 karakter uzunluğunda olmalıdır.

- 1 ve 2 nolu adımlarda bahsedilen kurallara uygun komut yazıldıktan sonra, komutun çözümlenme işleminin başlayabilmesi için klavyeden "Enter" tuşuna basılmalıdır. Aksi takdirde hiçbir işlem yerine getirilmeyecektir.

- "dir" komutu tek basına bir komut olup, yazımı tamamlandıktan sonra

“Enter” tuşuna basılması yeterlidir.” [52].

Tablo 4. 3. Mikrobilgisayar Mimarisinde Kullanılan Komutlar ve Açıklamaları [52]

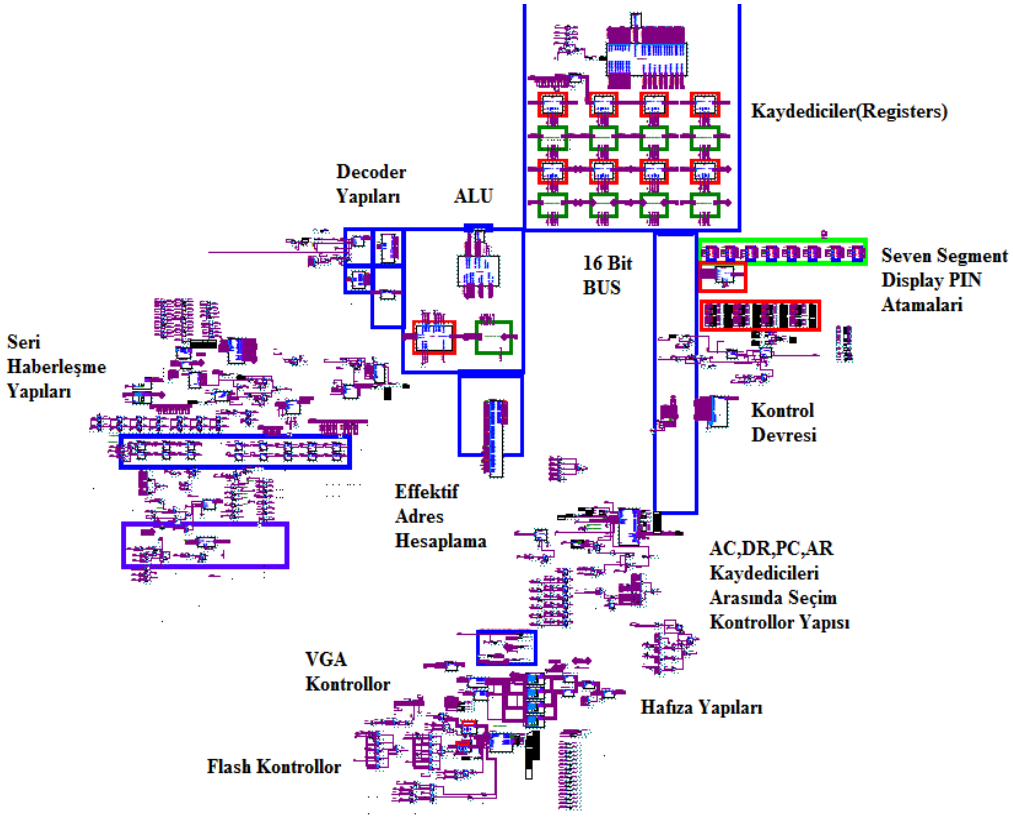
Komut	**Yapılan iş**	**Örnek Komut Yazımı**
new	Yeni bir dosya oluşturma	new abcde.asm
save	Aktif dosyayı kaydetme	save
edit	İstenilen bir dosyayı düzenleme	edit abcde.asm
run	Aktif dosyayı derleyip çalıştırma	run
dir	Kök dizindeki dosyaları görüntüleme	dir

4.2.1.BZK.SAU’da Modülerlik

BZK.SAU eğitimsel amaçlı olarak tasarlanan mikrobilgisayar olması sebebi ile içerisindeki sayısal tasarımlara kullanıcının müdahil olması tasarımda kullanıcıya verilen özelliklerden birisidir. Modülerlik kavramı sayesinde BZK.SAU mikrobilgisayar mimarisine kullanıcılar müdahil olabilmesi ile Elektrik-Elektronik Muhendislikleri ve Bilgisayar Muhendisliklerinde okutulan Bilgisayar Mimarisi ve Organizasyonu vb. derslerin kolay anlaşılması açısından önemli bir çalışma olduğu düşünülmektedir [52].

BZK.SAU mikrobilgisayar mimarisi içinde modülerlik özelligine sahip tasarımlar şunlardır.

- Modüler Aritmatik ve Mantık Birimi
- Modüler Bellek Tasarımı
- Modüler Kaydedici Tasarımı
- Modüler Adres ve Veri Yolu Tasarımı [52]



Şekil 4. 5.BZK.SAU Mikrobilgisayar Mimarisi İç Yapısı ve İlgili Tasarımlar [52]

4.3.Mikrobilgisayar Mimari İçinde Seri Haberleşme Protokolü Tasarımı

BZK.SAU mikrobilgisayar mimarisi içinde RS232 asenkron seri haberleşme protokolü tasarımını gerçekleştirmek ilgili tez çalışmasının temel amaçlarından biri olmuştur. RS232 protokolü genel yapısı RS232 standartı temelleri bölümünde anlatılmıştır. Bu kısım içinde tasarımı yapılan seri gönder-al tasarımı, frekans bölücü devre tasarımı , bunların akış diyagramları ve simulasyon sonuçları anlatılmıştır.

4.3.1.Frekans Bölücü Devre Tasarımı

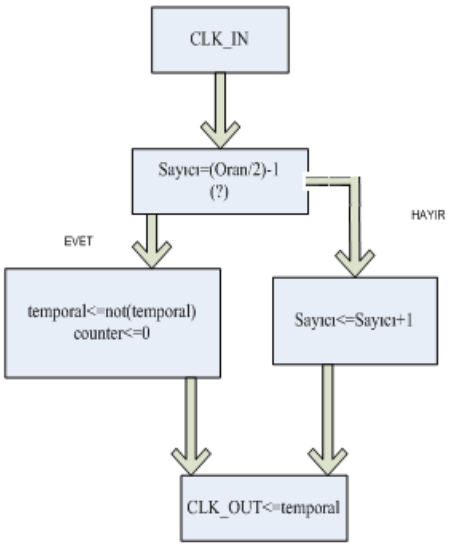
Tasarımı yapılan her sayısal devrenin çalışması için clock frekansının istenen frekans değerlerinde elde edilmesi gerekmektedir. Bunun için gerekli olan devre yapıları flip-flop ve 555 Timer ile tasarımı yapılmasının yanı sıra VHDL dili ile kodsal olarak da gerçeklenebilmektedir. İlgili tez çalışmasınında frekans bölücü devre yapısı VHDL dilinde kodlanmış ve UART modülü olarak tasarımı yapılan seri veri gönderen ve alan devre yapıları içinde kullanılmıştır. İlgili devrenin frekans

bölücü işlevini gerçekleştirmesi için kullanılan temel formül şu şekilde hesaplanmaktadır:

Oran=CLK_IN/CLK_OUT
Sayıcı=(Oran/2)-1

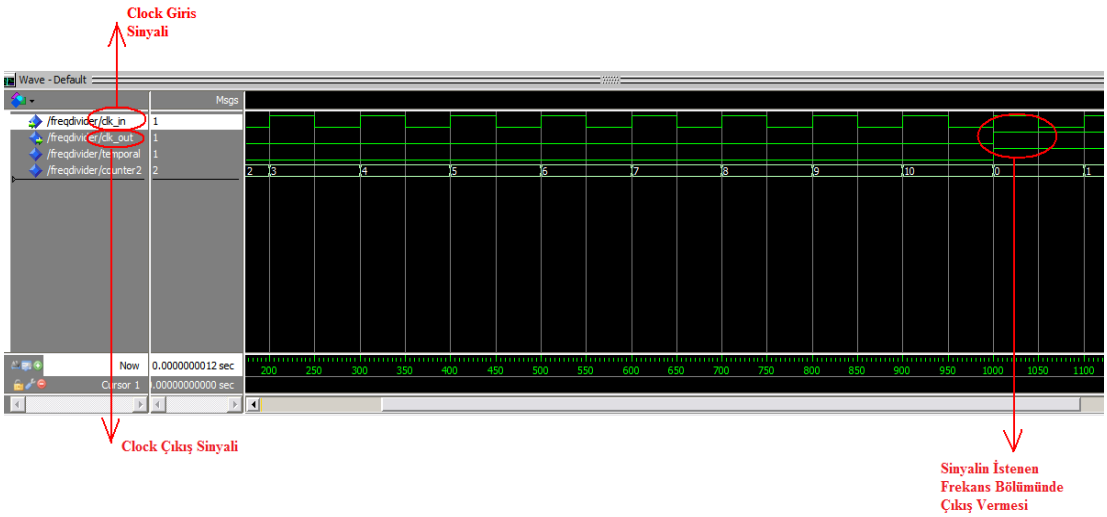
- CLK_IN olarak ifade edilen kısaltma devreye giren giriş frekansını ifade eder.CLK_OUT olarak ifade edilen kısaltma da devrenen çıkan ve arzu edilen çıkış frekansını ifade eder.
- Oran olarak ifade edilen ifade ise giriş frekansı ve çıkış frekansı arasındaki orandır.
- Sayıcı ise bir peryodun karesel bir pulse için lojik seviye 1 ve lojik seviye 0 olması dolayısı ile yarıya bölünmüş ve Sayıcının saymaya sıfırdan başlaması sebebiyle bir azaltılmıştır.
- temporal olarak ifade edilen değişken ise sinyal türünde tanımlanmış Sayıcı'daki değişimi gösterir.

İlgili devrenin algoritmik yapısını akış diyagramı şöyledir:



Şekil 4. 6.VHDL Dilinde Tasarımı Yapılan Frekans Bölücü Devre Yapısı

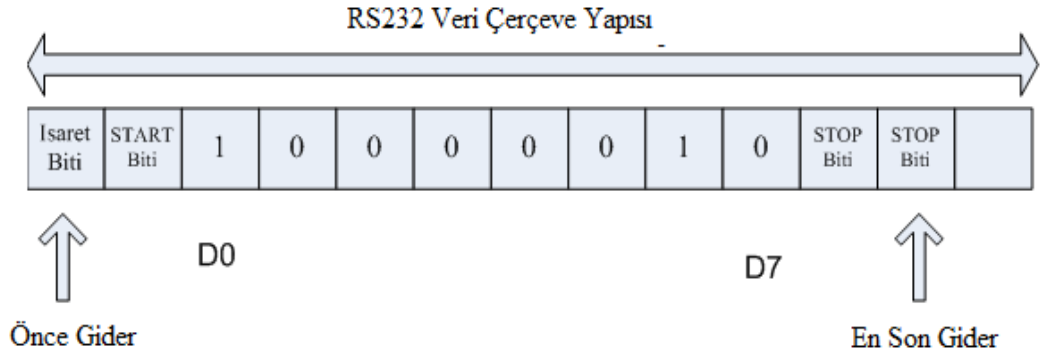
Yine aynı devre yapısın ModelSim simulasyon programında doğru çalıştığını gösteren simulasyon ekran çıktısı şöyledir.



Şekil 4. 7. Frekans Bölücü Devrenin Simulasyon Çıktısı

4.3.2.Seri Veri Gönderme Algoritma ve Simulasyonu

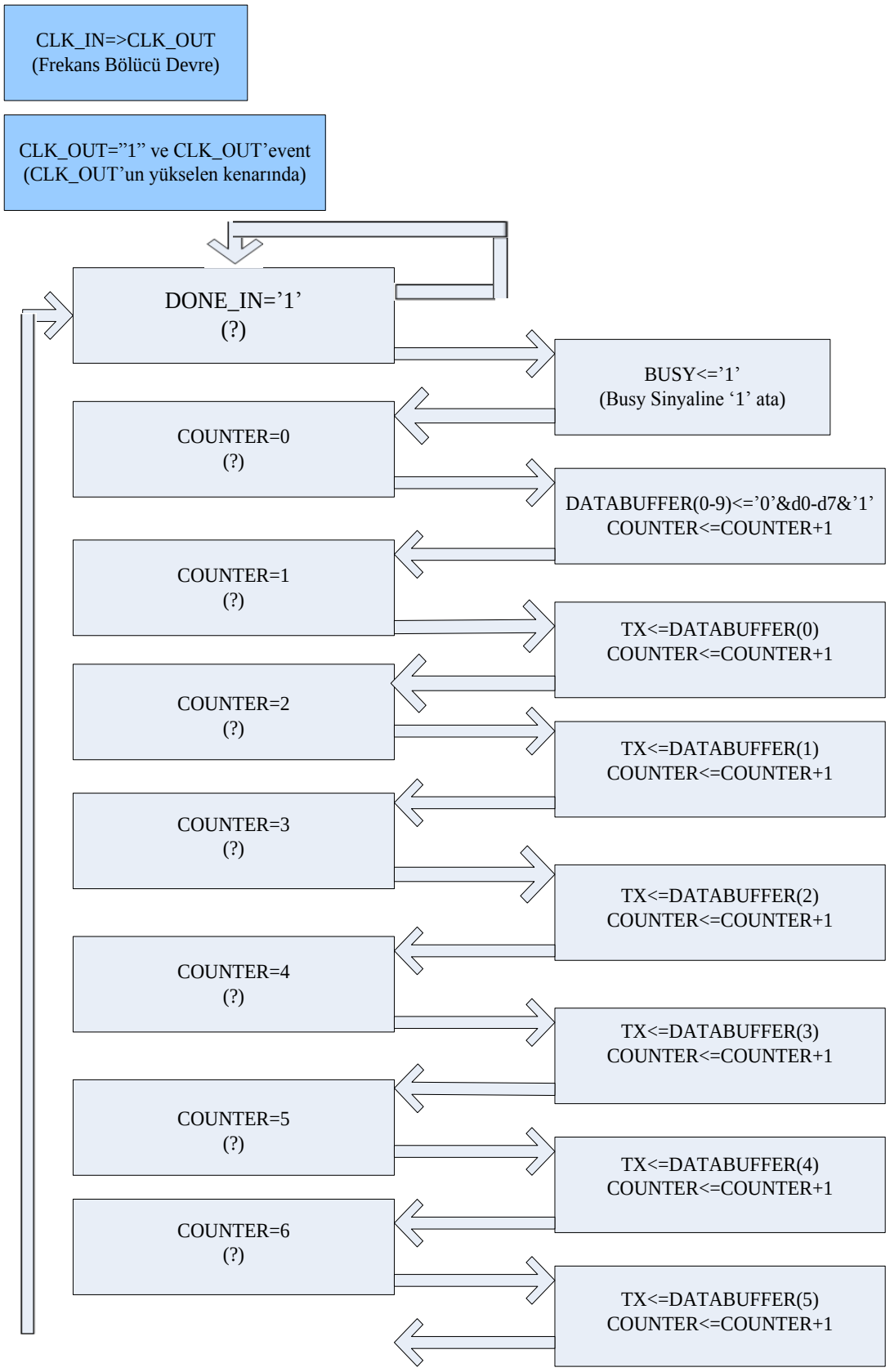
Yapılan tez çalışmasında RS232 seri haberleşme protokolü VHDL dili ile iki kısım halinde kodlanmıştır. Bu iki kısım seri veri gönderen ve alan olarak tasarlanmıştır. Tasarımı yapılan algoritma RS232 asenkron seri haberleşme protokolüne göre oluşturulmuş simulasyonu yapılmış ve FPGA devre kartı üzerinde denenmiştir. Bu suretle gerçek zamanlı olarak çalışan sistem arasında veri akışı seri olarak sağlanmıştır. Seri olarak veri gönderen devrenin istenen hızlarda doğru bir şekilde çalışıp çalışmadığının denenmesi "RealTerm Serial Capture Program" isimli program ile yapılmıştır.

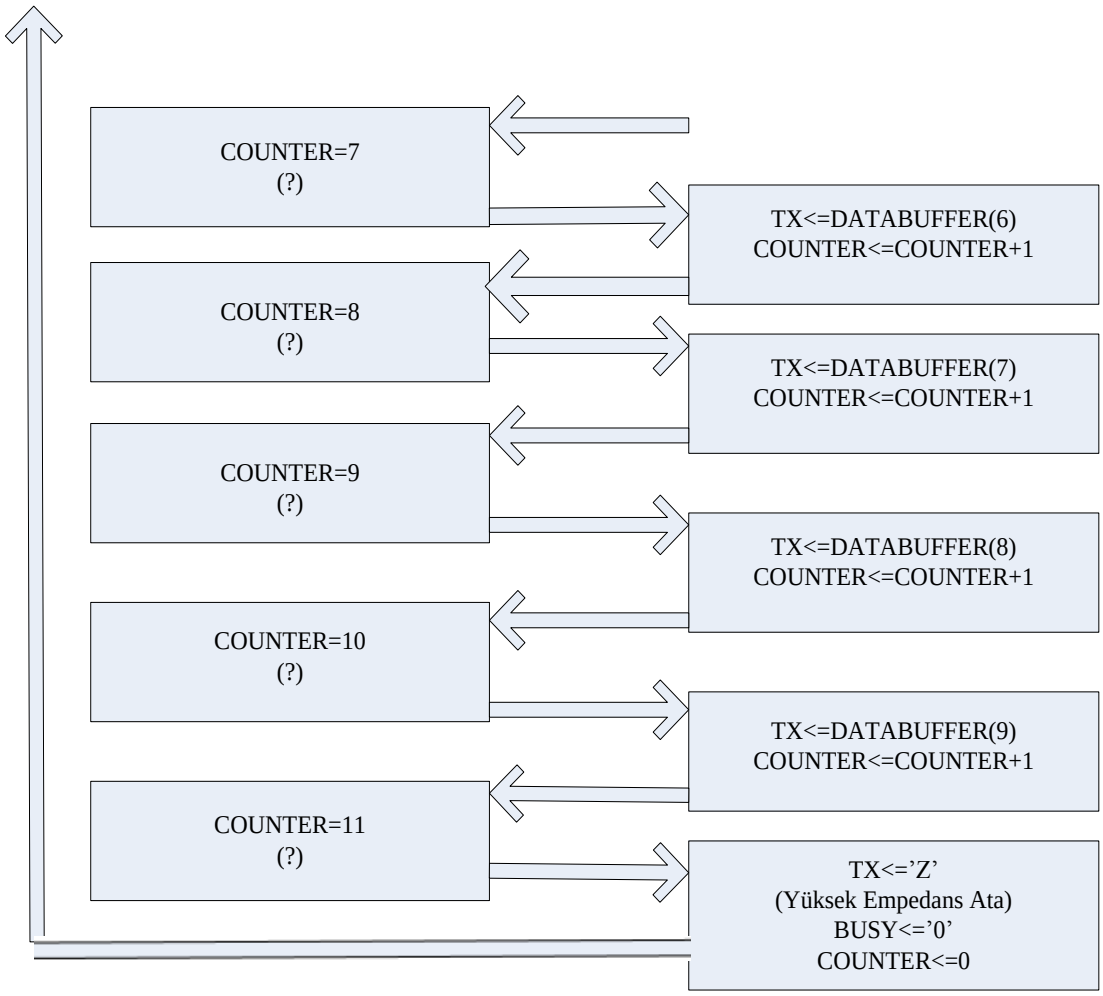


Şekil 4. 8.RS232 Veri Çerçeve Yapısı

RS232 veri çerçeve yapısı olarak ifade edilen şematik yapı seri olarak gönderilecek ve alınacak verilerin yapısını göstermektedir. Seri olarak gönderilen her veri aynı çerçeve yapısını kullanarak gönderilmektedir. İlgili çalışmada VHDL dili ile yapılan tasarım ve algoritma bu veri çerçevesi kullanarak oluşturulmuştur.

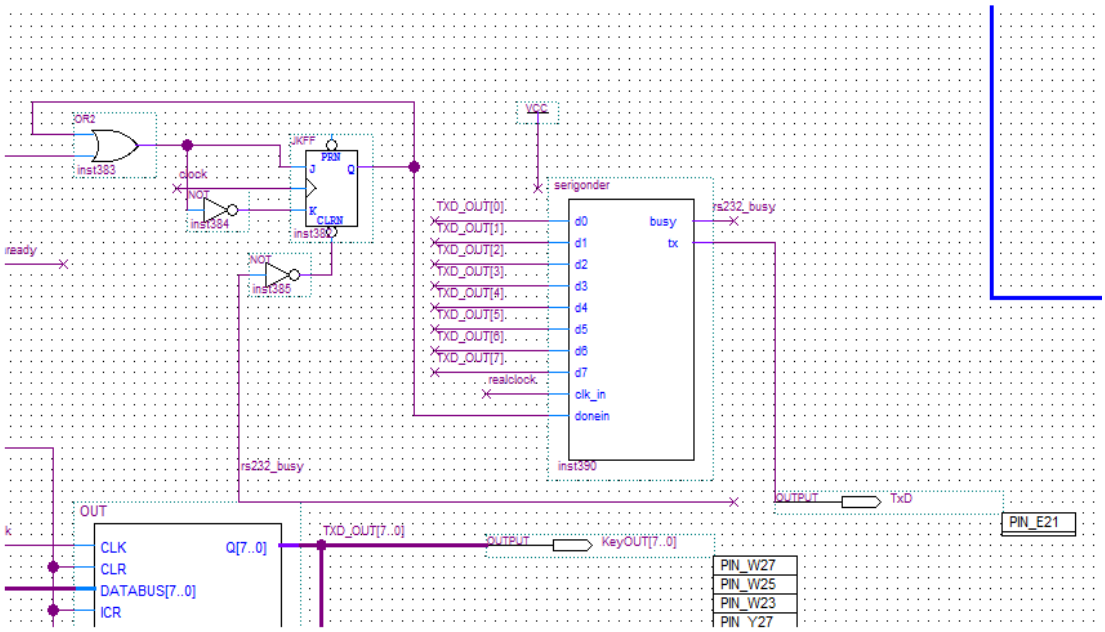
İlgili çalışmada RS232 asenkron seri haberleşme protokolü olarak VHDL dilinde tasarımı yapılan algoritma Şekil 4.9'da gösterilmiştir.


CLK_IN=>CLK_OUT
(Frekans Bölücü Devre)
CLK_OUT="1" ve CLK_OUT'event
(CLK_OUT'un yükselen kenarında)
DONE_IN='1'
(?)
BUSY<='1'
(Busy Sinyaline '1' ata)
COUNTER=0
(?)
DATABUFFER(0-9)<='0'&d0-d7&'1'
COUNTER<=COUNTER+1
COUNTER=1
(?)
TX<=DATABUFFER(0)
COUNTER<=COUNTER+1
COUNTER=2
(?)
TX<=DATABUFFER(1)
COUNTER<=COUNTER+1
COUNTER=3
(?)
TX<=DATABUFFER(2)
COUNTER<=COUNTER+1
COUNTER=4
(?)
TX<=DATABUFFER(3)
COUNTER<=COUNTER+1
COUNTER=5
(?)
TX<=DATABUFFER(4)
COUNTER<=COUNTER+1
COUNTER=6
(?)
TX<=DATABUFFER(5)
COUNTER<=COUNTER+1



Şekil 4. 9.Seri Veri Gönderen Devre Yapısı Kod Algoritması

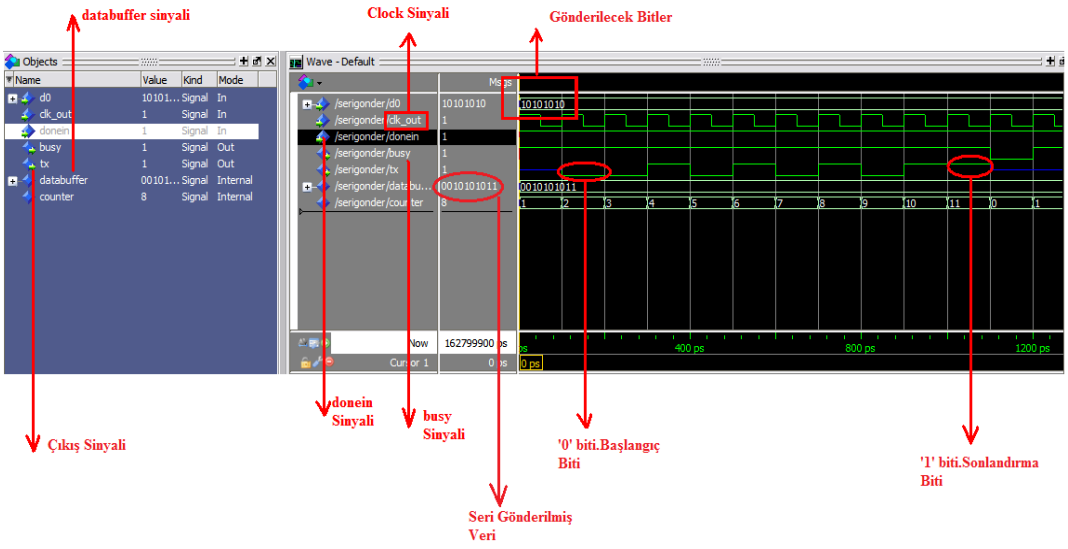
İlgili seri veri gönderen devre tasarımının BZK.SAU içinde Quartus ortamındaki şematik gösterimi Şekil 4.10'da gösterilmiştir.



Şekil 4. 10.Quartus Ortamında Seri Veri Gönderen Devre Şematik Gösterim

Şekil 4.10'da gözüktüğü gibi BZK.SAU mikro bilgisayarın çıkış kaydedicisinden gelen 8 bitlik parelel veriler "serigönder" ismi ile adlandırılan devrede "tx" olarak isimlendirilen Pin çıkışından seri olarak olarak dış dünyaya veri göndermektedir. Bu suretle mikro bilgisayar içinde derlenen ve makine koduna çevrilen her veri RS232 asenkron seri veri aktarımı protokolünü kullanarak gerek bilgisayar ile gerekse aynı protokolü kullanan cihazlar ile istenen hızlarda veri aktarımı yapmaktadır.

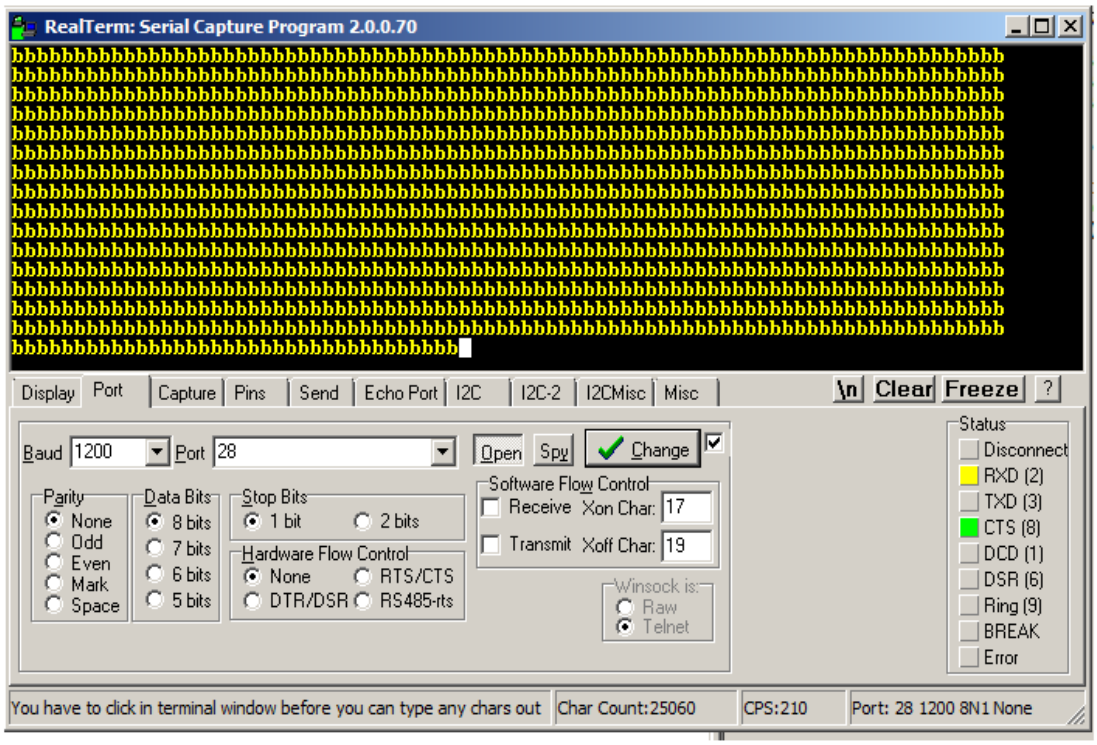
Seri veri gönderen devrenin ModelSim simulasyon programında alınan sonuçları Şekil 4.11'de gösterilmiştir.



Şekil 4. 11.ModelSim Programında Seri Veri Gönderimi Simulasyonu

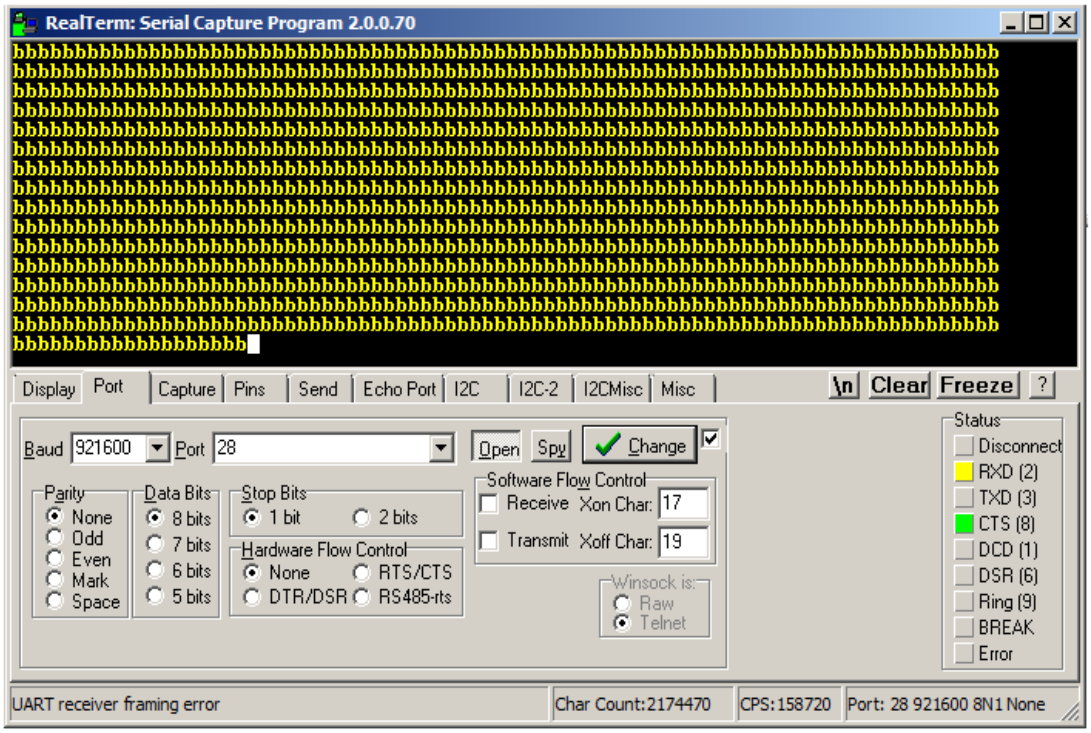
RS232 seri veri gönderme protokolü standartına göre verilerin doğru ve istenen baud oranı hızında gönderildiğini görmek için "RealTerm Serial Capture" isimli program kullanılmıştır. İlgili seri veri gönderen devrenin belirli baud oranında RealTerm program ile veri alması esnasında elde edilen ekran çıktılarından bazıları aşağıda Şekil 4.12, Şekil 4.13' de gösterilmiştir.

DE2-70 FPGA bordu üzerinde bulunan anahtarlar (switch) ile oluşturulan 8 bitlik ikili sayı sistemi olarak oluşturalan ASCII karakterler RealTerm programı ile seçilen baud oranı hızlarında alınmıştır. BZK.SAU içinde yüksek doğrulukta çalışan devrenin RealTerm programında elde edilen ekran çıktıları Şekil 4.12, Şekil 4.13' de gösterilmektedir.



Şekil 4. 12.RealTerm Programında 1200 Baud Oranı Hızında Veri Alımı

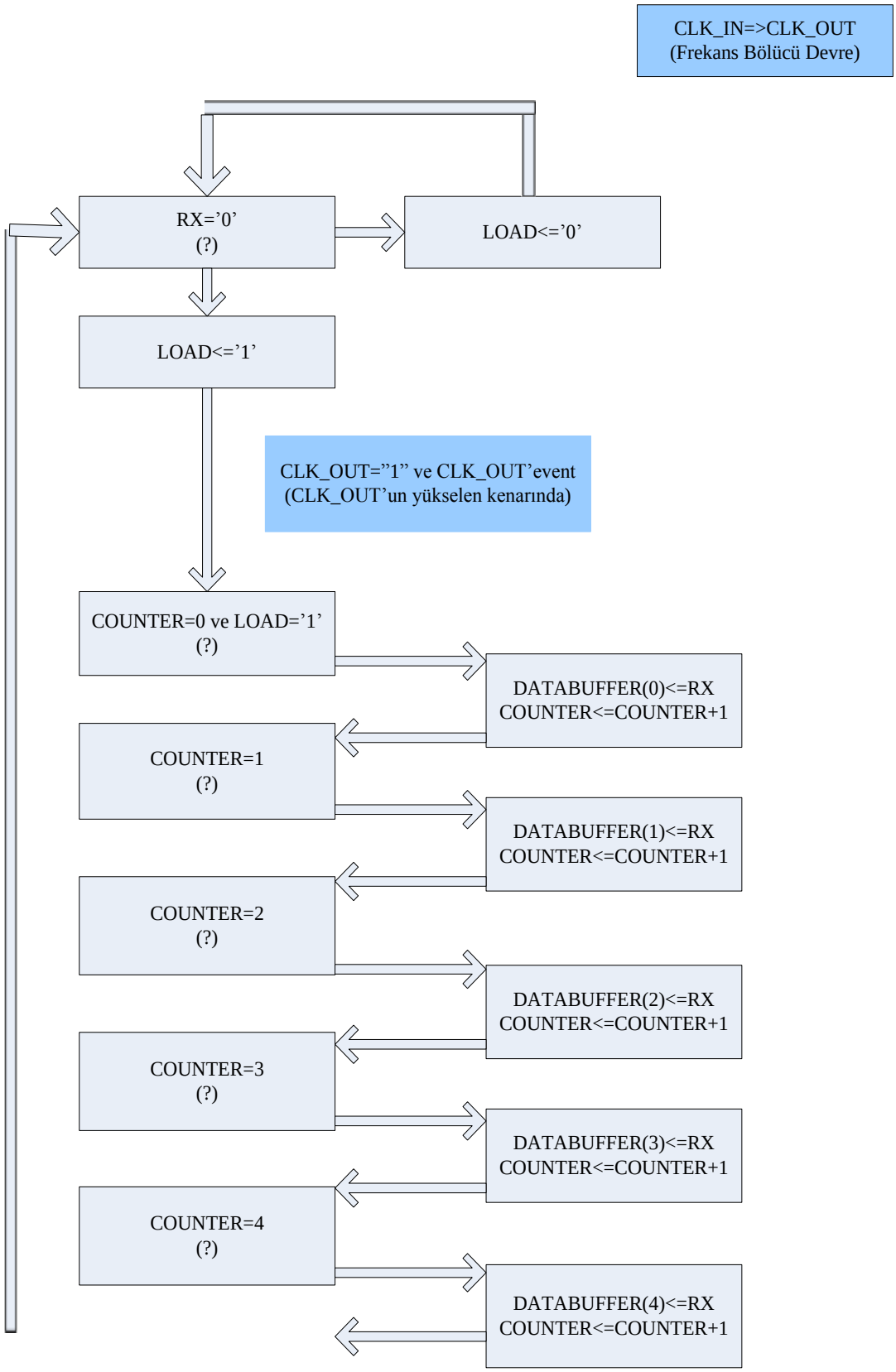
Şekil 4.12 ve Şekil 4.13'de gösterilen RealTerm programına ait ekran çıktısında 'b' ASCII karakteri FPGA bordu üzerinde anahtarların "01100010" şeklinde sıralanması ile oluşmuştur. Böylece PC ve FPGA bordu arasında istenen baud oranında seri veri aktarımı mümkün kılınmıştır.

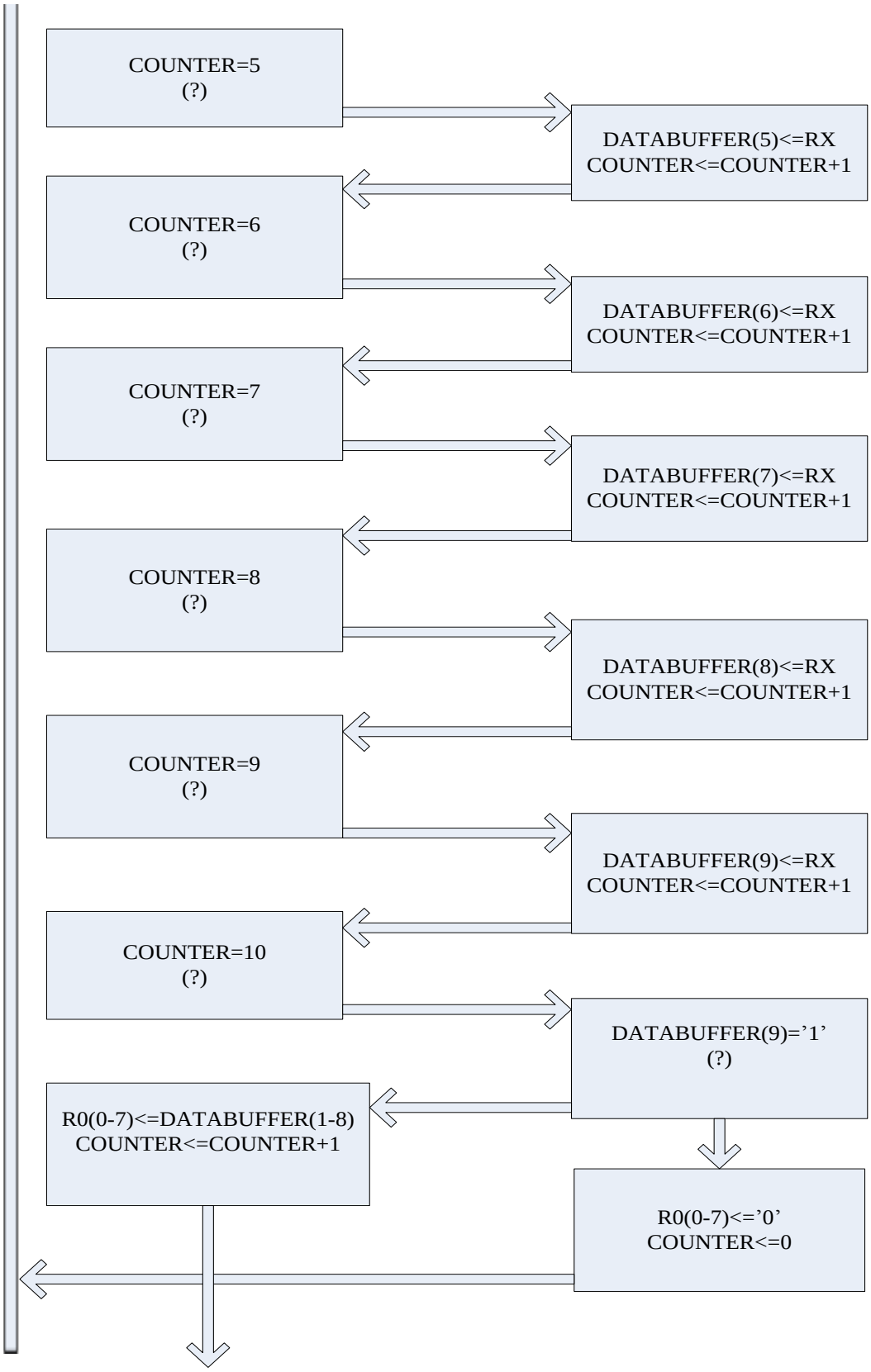


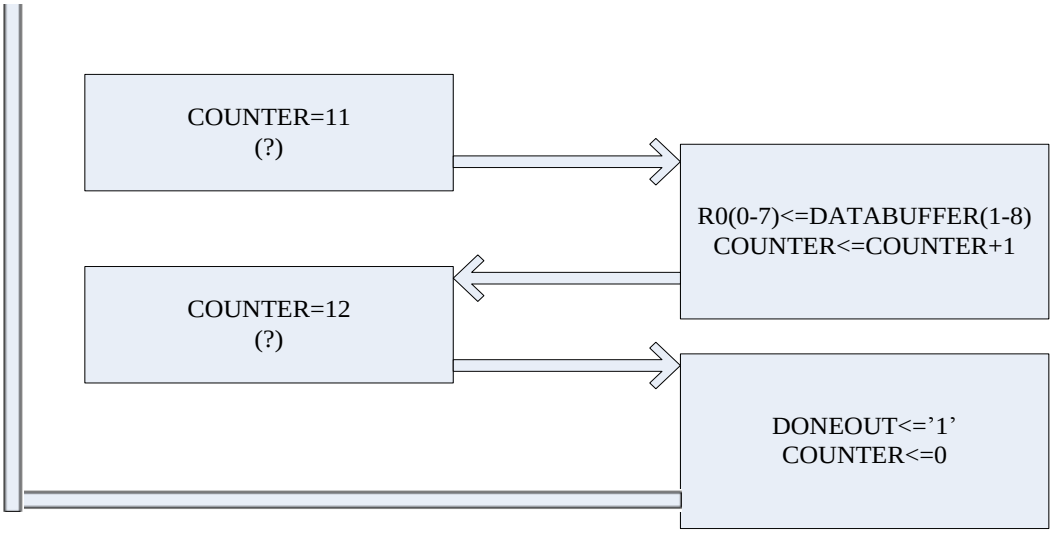
Şekil 4. 13. RealTerm Programında 921600 Baud Oranı Hızında Veri Alımı

4.3.3.Seri Veri Alma Algoritması ve Simulasyonu

Tez çalışmasında RS232 seri haberleşme protokolü modülünün seri veri alma kısmının algoritması, simulasyon sonuçları ve FPGA devresi üzerinde çalışmasını gösteren ilgili resimler eklenmiştir. UART entegresi olarak bilinen RS232 seri haberleşme protokolünün VHDL dilinde iki bölüm halinde kodlanması (tasarlanması) kod yapısının daha kolay oluşturulmasını ve esnekliğini arttırmaktadır.


CLK_IN=>CLK_OUT
(Frekans Bölücü Devre)
RX='0'
(?)
LOAD<='0'
LOAD<='1'
CLK_OUT="1" ve CLK_OUT'event
(CLK_OUT'un yükselen kenarında)
COUNTER=0 ve LOAD='1'
(?)
DATABUFFER(0)<=RX
COUNTER<=COUNTER+1
COUNTER=1
(?)
DATABUFFER(1)<=RX
COUNTER<=COUNTER+1
COUNTER=2
(?)
DATABUFFER(2)<=RX
COUNTER<=COUNTER+1
COUNTER=3
(?)
DATABUFFER(3)<=RX
COUNTER<=COUNTER+1
COUNTER=4
(?)
DATABUFFER(4)<=RX
COUNTER<=COUNTER+1

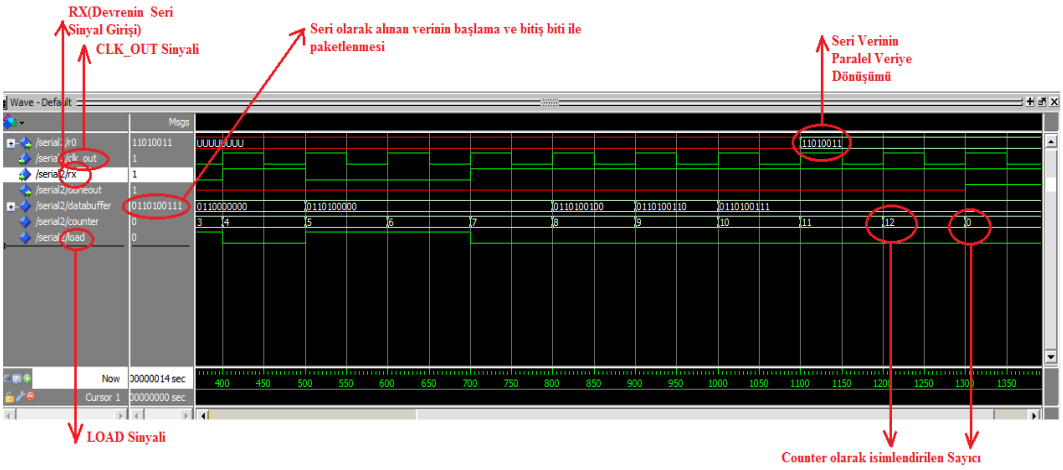

COUNTER=5
(?)
DATABUFFER(5)<=RX
COUNTER<=COUNTER+1
COUNTER=6
(?)
DATABUFFER(6)<=RX
COUNTER<=COUNTER+1
COUNTER=7
(?)
DATABUFFER(7)<=RX
COUNTER<=COUNTER+1
COUNTER=8
(?)
DATABUFFER(8)<=RX
COUNTER<=COUNTER+1
COUNTER=9
(?)
DATABUFFER(9)<=RX
COUNTER<=COUNTER+1
COUNTER=10
(?)
DATABUFFER(9)='1'
(?)
R0(0-7)<=DATABUFFER(1-8)
COUNTER<=COUNTER+1
R0(0-7)<='0'
COUNTER<=0



Şekil 4. 14.Seri Veri Alma Devre Tasarımı Algoritması

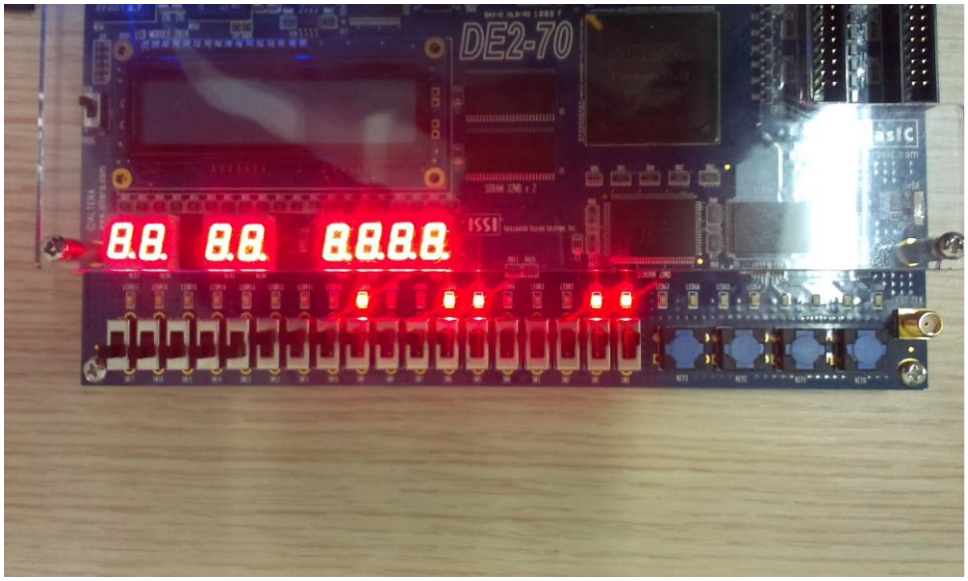
Seri veri alma algoritmasının temel olarak yaptığı iş seri olarak gönderilen verileri istenen baud oranı hızında almak ve paralel verilere dönüştürmektir. BZK.SAU.FPGA'in 8 bitlik giriş kaydedicisinden (Register) gelen verilerin anlamlı veri olup olmadığını anlamak için öncelikle RS232 protokolü gereğince veri hattının lojik 1 den lojik 0 a düşme anına bakılır ve ilk düşme anında gelen veri, lojik 0 verisi ise databuffer(0) olarak ifade edilen vektör veya dizinin ilkine kaydedilir. databuffer(0)'ın lojik 0 seviyesinde olması "Load" isminde bir sinyali aktif yapması sonucu daha sonrasında gelen veriler aynı dizi içerisine alınır. En son bit olan databuffer(9) bitinin lojik 1 olması, RS232 protokolü veri çerçevesi içinde durdurma biti olması sebebi ile bir sonraki algoritmik adım içinde databuffer(1-8) veri bitlerinin çıkışa aktarılmasına olanak sağlar. Tasarım içinde gerçekleşen her adım bütün dijital tasarımlarda olduğu gibi belirli bir Clock (Saat) yapısına bağlıdır. Clock'un her yükselen kenarında kod yapısı içindeki bir sonraki duruma geçilir. Böylece algoritma içinde kodlanan bütün durumlar birbirlerini takip ederler.

İlgili seri veri alma algoritmasının ModelSim simulasyon programında elde edilen sonuçlar Şekil 4.15'de gösterilmiştir.



Şekil 4. 15.Seri Veri Alma Devresi ModelSim Simulasyonu Sonucu

İlgili seri veri alma devresinin FPGA üzerinde veri almasına yönelik örnek gösterim Şekil 4.16 'da gösterilmiştir.



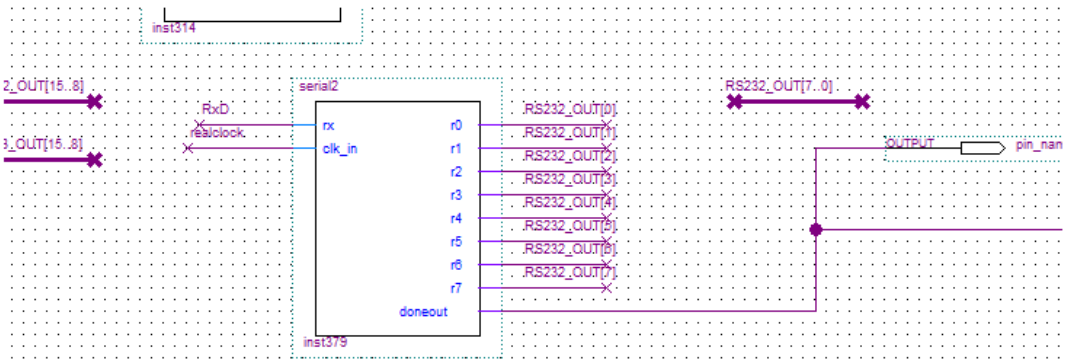
Şekil 4. 16. Veri Alma Devresinin FPGA Ortamında Veri Almasına Bir Örnek

Seri olarak veri alan devrenin FPGA ortamında istenen Baud oranı hızında veri aldığını göstermek için yine RealTerm program kullanılmıştır. RealTerm programında istenen Baud oranı hızında gönderilen veriler FPGA'in ledleri üzerinde binary olarak gösterilmiştir. İlgili örnekte Şekil 4.16'da gösterilen veri, küçük 'c' ASCII karakterinin doğru bir şekilde alındığını göstermektedir. Küçük 'c' ASCII karakteri "11000110" olarak tanımlanmıştır. En son solda görülen kırmızı led "doneout" sinyali olarak adlandırılan gönderilen verinin doğru bir şekilde alındığını

gösterir.”11000110” verisinin FPGA’ledlerinde gösterimi en sağdan başlamak üzere okunabilir. Kırmızı ledler üzerinde görülen verinin en düşük biti en sağda en önemli biti en solda olarak gösterilmiştir.

Seri olarak veri alan devrenin çalışma frekansı asenkron seri haberleşen cihazların seri alma baud oranı hızları olan 100000 bps ve altında hatasız olarak çalışmaktadır. 100000 bps ve üzeri hızlarda hata oranı artmakla birlikte RS232 için en yüksek oran olan 921600 bps oranında dahi veri alımı gerçekleşmekte fakat alınan verilerin hata oranı yüzde 50’ye kadar çıkmaktadır.

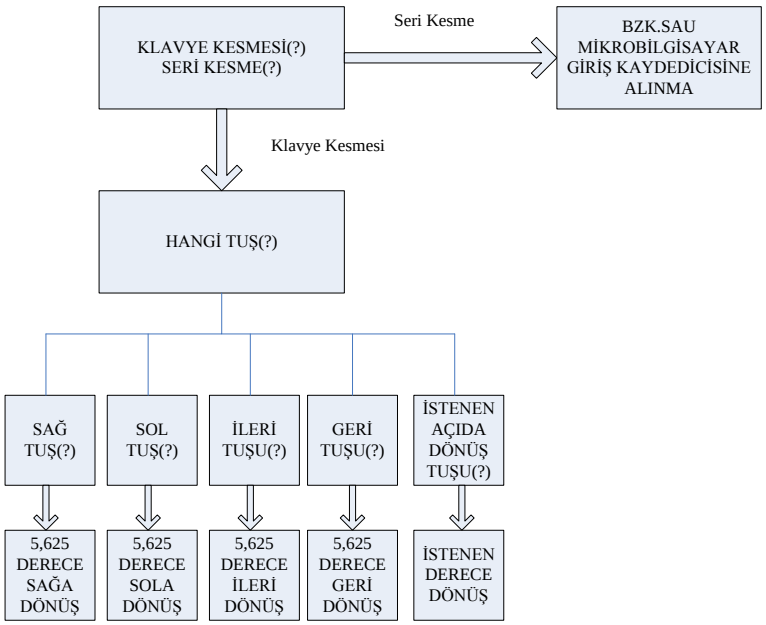
Seri olarak veri alan devrenin Quartus ortamındaki gösterimi Şekil 4.17’de gösterilmiştir.



Şekil 4. 17. Veri Alan Devrenin Quartus Programı İçindeki Şematik Görüntüsü

4.3.4. Pan/Tilt Mekanizması ve Adım Motor Kontrol Yazılımı

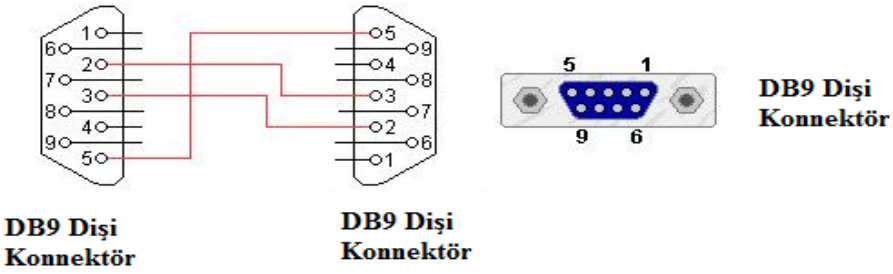
Yüksek lisans tez çalışmasında BZK.SAU mikrobilgisayar mimarisi içinde BZK.SAU.Assm dili ile istenen yönde ve/veya istenen açıda dönmesini sağlayan kod yapısı oluşturulmuştur. İlgili hareket sistemini sağlayan algoritma Şekil 4.18’de gösterilmiştir.



Şekil 4. 18.BZK.SAU.Assm Dili ile Oluşturulan Hareket Algoritması

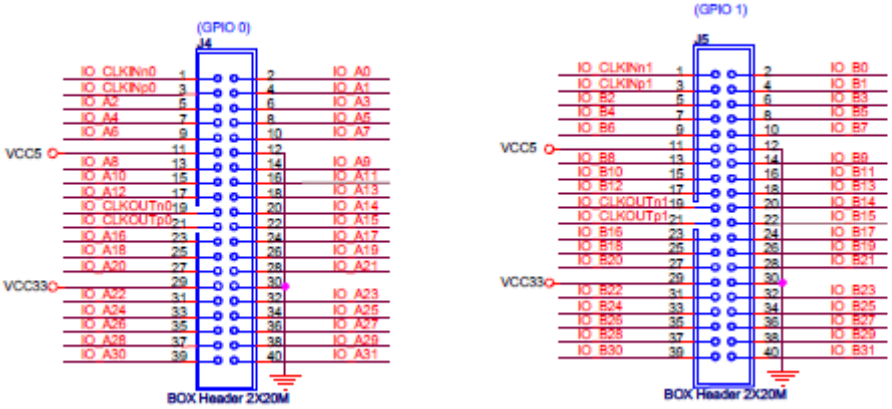
İlgili çalışmada BZK.SAU mikrobilgisayar yapısının gömülü olduğu FPGA-1 kartı Klavye veya RS232 seri haberleşme donanımdan gelen kesme işlemine göre ilgili alt programa dallanan kod yapısı hareket algoritmasının ilk adımıdır. DE2-70 FPGA kartına bağlı klavye yapısı istenen herhangi bir tuşa basılarak gönderilen ASCII kodları, mikroişlemci tarafından tarafından kesme mekanizmasına bağlı olarak giriş kaydedicisi içine kaydedilir. İlgili kesme mekanizması BZK.SAU içinde oluşturulmuş donanımsal bir mekanizmadır. RS232 seri haberleşme modülüne gelen seri sinyallerin doğru olarak alınabilmesi için BZK.SAU içine gömülmüş modülün karşı cihaz ile aynı baud oranında hıza sahip olmalıdır aksi halde doğru veri aktarımı mümkün olmaz. Şekil 4.17'de görülebileceği gibi "doneout" sinyali olarak olarak isimlendirilen sinyal seri olarak alınan sinyalin paralel sinyale dönüştürülmesi işleminin hemen bir saat darbesi sonrasında üretilir ve bu sinyal seri kesme mekanizmasını başlatan iki sinyalden diğeridir.

Sistemin iki FPGA kullanması Şekil 4.1'de FPGA-1 ve FPGA-2 olarak gösterilen FPGA'ler üzerindeki RS232 DB9 dişi bağlantıları Şekil 4.19'da gösterilmiştir.



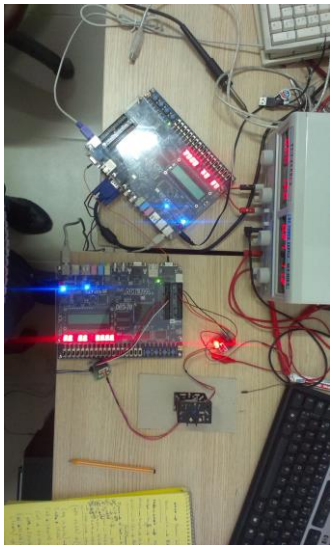
Şekil 4. 19.İki FPGA Arasında RS232 DB9 Dişi Bağlantı Pinleri [53]

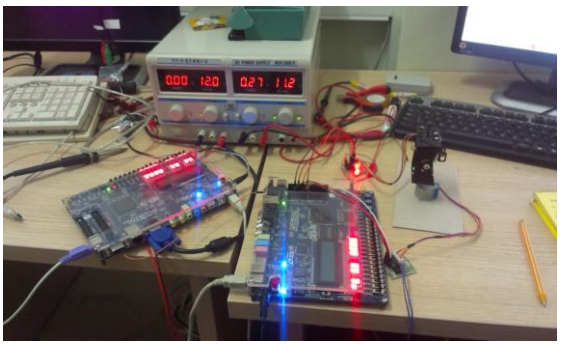
DB9 dişi konnektöründe pin 2 TX (Transmitter) ve pin 3 RX (Receiver) olarak gösterilmiş ve FPGA'ler arasındaki seri haberleşme bağlantısı bu şekilde kurulmuştur.



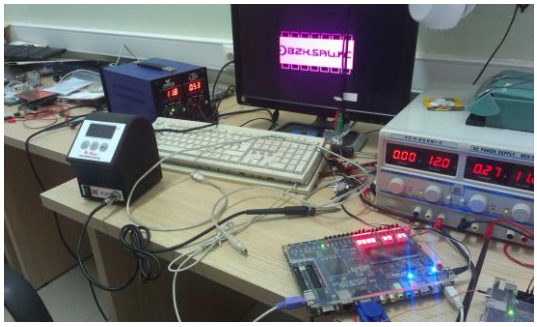
Şekil 4. 20. Altera DE2-70 FPGA Bordu GPIO Çıkışları

FPGA-2 üzerinden adım motorlara aktarılan ilgili veri sinyalleri FPGA'in GPIO pinlerinden gönderilmektedir.GPIO-0 üzerinden aktarılan veri sinyalleri ilgili pan/tilt mekanizmasının yatay eksendeki sağ-sol hareketi için veri akışı akışı sağlarken, GPIO-1 üzerinden ileri-geri hareket için veri akışı sağlanmaktadır. Her biri 40 pinlik GPIO bağlantı noktalarından GPIO-0 ve GPIO-1 üzerinde 5-10-15-20 nolu pinler çıkış olarak kullanılmaktadır.





Şekil 4. 21.Çalışan Sistemden Bir Görüntü



Şekil 4. 22.Çalışan Sistemden Bir Görüntü

İlgili çalışmada sistemin fotografları Şekil 4.21ve Şekil 4.22'de gösterilmiştir.

İlgili tez çalışmasında BZK.SAU.Assm dili ile oluşturulan kontrol yazılımı adım motorlara bağlı olan Pan/Tilt mekanizmasını açısal olarak istenen yönlerde hareket ettirme kabiliyetine sahiptir. BZK.SAU.Assm içinde 59 adet kod yapısından 9 adet kod kullanılmış olup, hareket algoritması Şekil 4.18'de gösterilmiştir.Bu komutlardan bazısı akümülatör ve bellek komutları, bazısı sıçrama ve dallanma komutları, bir kısmıda giriş ve çıkış komutlarıdır.Kullanılan kodlar ve açıklamaları şunlardır.

STA Komutu: Kısaltma olarak STA ile gösterilen komut, mikrobilgisayarın akümülatörün içereğini istenen bellek adresine yerleştirmek için kullanılır. Mikrobilgisayar mimarisi içinde akümülatör ve bellek komutlarındandır. *STA $Adres* şeklindeki kullanımı direkt adresleme modundaki kullanım örneğidir.

LDA Komutu: Kısaltma olarak LDA ile gösterilen komut, istenen ivedi bir değeri veya istenen adresteki değeri mikrobilgisayarın akümülatörüne yükler. Mikrobilgisayar mimarisi içinde akümülatör ve bellek komutlarındandır. *LDA #Değer* ve *LDA $Adres* şeklindeki kullanımlar sırasıyla ivedi ve direkt adresleme modları için geçerlidir.

CMP Komutu: Kısaltma olarak CMP ile gösterilen komut, akümülatör içindeki değeri doğrudan bir değer veya belirtilen bellek adresi içindeki bir değer ile kıyas eder. Buna göre durum kayıt edicilerine bağlı olarak dallanma işlemi için gerekli olan bayraklar aktif hale gelir. *CMP #Değer* veya *CMP $Adres* şeklindeki kullanım örnekleri ivedi ve direkt adresleme modları için geçerlidir.

BRA Komutu: Kısaltma olarak BRA ile gösterilen komut, herhangi bir şart olmadan istenen işlem adresine dallanır ve işlem akışı o bellek adresinden devam eder. *BRA *Değer* şeklindeki kullanım örneği göreceli adres modu için geçerlidir.

BZR Komutu: Kısaltma olarak BZR ile gösterilen komut, CMP bağlı olarak durum kayıt edicilerinde Z=1(Sıfır Bayrağı) olması durumunda istenen adrese dallanır.

DECR Komutu: Kısaltma olarak DECR ile gösterilen komut, mikrobilgisayar akümülatörünün içindeki değeri 1 arttırır. *DECR* şeklinde kullanım örneği mevcuttur.

INCR Komutu: Kısaltma olarak INCR ile gösterilen komut, mikrobilgisayar akümülatörünün içeriğini 1 azaltır. *INCR* şeklinde kullanım örneği mevcuttur.

IN Komutu : Kısaltma olarak IN ile gösterilen komut , mikrobilgisayarın 8 bitlik giriş kaydecisinin değerini akümülatöre aktarır. *IN* olarak kullanımı mevcuttur.

OUT Komutu: Kısaltma olarak OUT ile gösterilen komut , mikrobilgisayarda akümülatörün içeriğini 8 bitlik çıkış kaydedicisine gönderir. *OUT* şeklinde kullanımı mevcuttur.

Tablo 4. 4.BZK.SAU.Assm Dili İle Hareket Algoritması Kod Yapısı

	Seri ve Klavye Kesmesi Başlangıç	STA $F003H	Gelen Kesmeyi F003H Adresinde Sakla

	Adresi		
		LDA $F003H	F003H Adresindeki Veriyi Akümülatöre Getir
		CMP #0001H	Akümülatördeki Veriyi 0001H Verisi İle Karşılaştır *(Klavye Kesmesi Mi?)*
		BZR #0009H	0009H Kadar Dallan *(Klavyeden Giriş Alınan Adrese Dallan)*
		LDA $F003H	F003H Adresindeki Veriyi Akümülatöre Getir
		CMP #0002H	Akümülatördeki Veriyi 0002H Verisi İle Karşılaştır *(Seri Kesme Mi?)*
		BZR #0800H	0800H Kadar Dallan *(Seri Giriş Alınan Adrese Dallan)*
		BRA #FFF1H	FFF1 Kadar Geriye Dallan *(Sürekli Olarak Gelen Klavye veya Seri Kesme İşlemini Taramaya Dön)*
	Klavyeden Alınan Veriler ve İlişkisel Kodlar Başlangıç Adresi	IN	Klavyeden Gelen Veriyi Giriş Kaydedicisinden Alarak Akümülatöre Yazar
		STA $0700H	Akümülatördeki Veriyi 0700H Adresinde Sakla
		CMP #0001H	Akümülatördeki Veriyi 0001H İle Karşılaştır *(ASCII Kodu-Sağ Yön Tuşu Olup Olmadığını Kontrol Et ?)*
		BZR #004AH	004AH Kadar Dallan *(Sağ Tuşa Basıldığı Zaman Gerçekleşmesi Gereken Durumlara Dallan)*
		LDA $0700H	0700H Adresindeki Veriyi Akümülatöre Aktar
		CMP #0002H	Akümülatördeki Veriyi 0002H İle Karşılaştır

			(ASCII Kodu-Sol Yön Tuşu Olup Olmadığını Kontrol Et ?)
		BZR #0094H	0094H Kadar Dallan *(Sol Tuşa Basıldığı Zaman Gerçekleşmesi Gereken Durumlara Dallan)*
		LDA $0700H	0700H Adresindeki Veriyi Akümülatöre Aktar
		CMP #0003H	Akümülatördeki Veriyi 0003H İle Karşılaştır *(ASCII Kodu-İleri Yön Tuşu Olup Olmadığını Kontrol Et ?)*
		BZR #00DEH	00DEH Kadar Dallan *(İleri Tuşuna Basıldığı Zaman Gerçekleşmesi Gereken Durumlara Dallan)*
		LDA $0700H	0700H Adresindeki Veriyi Akümalatöre Aktar
		CMP #0004H	Akümülatördeki Veriyi 0004H İle Karşılaştır *(ASCII Kodu-Geri Yön Tuşu Olup Olmadığını Kontrol Et ?)*
		BZR #0128H	0128H Kadar Dallan *(Geri Tuşuna Basıldığı Zaman Gerçekleşmesi Gereken Durumlara Dallan)*
		LDA $0700H	0700H Adresindeki Veriyi Akümalatöre Aktar
		CMP #0063H	Akümülatördeki Veriyi 0063H İle Karşılaştır *(ASCII Kodu-'c' Tuşu Olup Olmadığını Kontrol Et ?) (İstenen Açıda Dönmek İçin Gerekli Olan Tuş)*
		BZR #0250H	0250H Kadar Dallan *('c' Tuşuna Basıldığı Zaman Gerçekleşmesi Gereken Durumlara*

			Dallan)
		BRA #FFB8H	FFB8 Kadar Geriye Dallan *(Sürekli Olarak Gelen Klavye veya Seri Kesme İşlemini Taramaya Dön)*
	Sağ Tuş İçin İlişkisel Kodlar	LDA $00A0H	00A0H Adresindeki Veriyi Akümülatöre Yükle *(Her Sağ Tuşa Basıldığı Zaman Gönderilen 8 Bitlik 8 Veri Setinin Ne Kadar Gönderildiği Bilgisini Tutar)*
		CMP #0007H	Akümülatördeki Veriyi 0007H İle Karşılaştır *(Her Sağ Tuşa Basıldığı Zaman Gönderilen 8 Veri Setinin 00A0H Adresinde Ulaşılıp Ulaşılmadığı Bilgisini Karşılaştırır) (00A0H Adresinde 0000H Verisi İlk Değer Olarak Atanmıştır)*
		BZR #003CH	003CH Kadar Dallan *(00A0H ve Kesme Kaydedicisi İçindeki Verileri Sıfırlamak İçin İlgili Adrese Dallan)*
	Yarım Adım Sürme Tekniği İçin Gönderilen Bitlerin Başlangıç Kod Adresi	LDA #0008H,#000CH,#0004H, #0006H,#0002H,#0003H, #0001H,#0009H	(“00001000” , “00001100”, “00000100”, “00000110”, “00000010”, “00000011”, “00000001”, “00001001”) *(İlgili Bitler Ardışıl Olarak Akümülatöre Yüklenir)*
		OUT	Akümülatöre Ardışıl Olarak Yüklenen Her Veri Çıkış Kaydedicisine Gönderilir
		LDA $00A0H	00A0H Adresindeki Veriyi Akümülatöre Yükle
		INC	Akümülatördeki Veriyi 1 Arttır *(Gönderilen Her 8 bitlik Veri Setinden Sonra 00A0H Adresi İçindeki Veri 1 Arttılmak Suretiyle*

			*Toplamda Her 5,625 Derecelik Dönüş İçin 8*8=64 Veri Çıkış Kaydecisine Gönderilir)*
		STA $00A0H	Akümülatördeki Veriyi 00A0H Adresinde Sakla
		BRA #FFDC	Sağ Tuş İçin İlişkisel Kodlar Adresine Şartsız Dallan
	00A1H Adresi(Klavye Kesmesi ve Veri Setini Sıfırlamak İçin Başlangıç Adresi)	LDA #0000H	Akümülatöre 0000H Verisini Yükle
		STA $00A0H	Akümülatördeki Veriyi 00A0H Adresine Yükle *(00A0H Adresine 0000H Yüklenmesi İle Tek Bir Adım İçin Gerekli Olan 64 Veri Seti Gönderilmiş ve Adresteki Veri Sıfırlanmış Demektir)*
		LDA #0000H	Akümülatöre 0000H Verisini Yükle
		STA $F003H	F003H Adresinde Kesme Kaydecisindeki Veriyi Sıfırla
		BRA #FF56H	FF56H Adresine Dallan*(Seri ve Klavye Kesmesi Başlangıç Adresine Şartsız Dallan)*
	Sol Tuş İçin İlişkisel Kodlar	LDA $00F0H	00F0H Adresindeki Veriyi Akümülatöre Yükle *(Her Sağ Tuşa Basıldığı Zaman Gönderilen 8 Bitlik 8 Veri Setinin Ne Kadar Gönderildiği Bilgisini Tutar)*
		CMP #0007H	Akümülatördeki Veriyi 0007H İle Karşılaştır *(Her Sağ Tuşa Basıldığı Zaman Gönderilen 8 Veri Setinin 00F0H Adresinde Ulaşılıp Ulaşılmadığı Bilgisini Karşılaştırır) (00F0H*

			Adresinde 0000H Verisi İlk Değer Olarak Atanmıştır)
		BZR #003CH	003CH Kadar Dallan *(00F0H ve Kesme Kaydedicisi İçindeki Verileri Sıfırlamak İçin İlgili Adrese Dallan)*
	Yarım Adım Sürme Tekniği İçin Gönderilen Bitlerin Başlangıç Kod Adresi	LDA #0009H,#0001H,#0003H, #0002H,#0006H,#0004H, #000CH,#0008H	(“00001001” , “00000001”, “00000011”, “00000010”, “00000110”, “00000100”, “00001100”, “00001000”) *(İlgili Bitler Ardışıl Olarak Akümülatöre Yüklenir)(Sağ Tuş İçin Gönderilen Bit Sırasının Tam Tersi Sıra Söz Konusu)*
		OUT	Akümülatöre Ardışıl Olarak Yüklenen Her Veri Çıkış Kaydedicisine Gönderilir
		LDA $00F0H	00F0H Adresindeki Veriyi Akümülatöre Yükle
		INC	Akümülatördeki Veriyi 1 Arttır *(Gönderilen Her 8 bitlik Veri Setinden Sonra 00F0H Adresi İçindeki Veri 1 Arttılmak Suretiyle Toplamda Her 5,625 Derecelik Dönüş İçin 8*8=64 Veri Çıkış Kaydecisine Gönderilir)*
		STA $00F0H	Akümülatördeki Veriyi 00A0H Adresinde Sakla
		BRA #FFDC	Sol Tuş İçin İlişkisel Kodlar Adresine Şartsız Dallan
	00F1H Adresi(Klavye Kesmesi ve Veri Setini Sıfırlamak İçin Başlangıç	LDA #0000H	Akümülatöre 0000H Verisini Yükle

	Adresi)		
		STA $00F0H	Akümülatördeki Veriyi 00A0H Adresine Yükle *(00F0H Adresine 0000H Yüklenmesi İle Tek Bir Adım İçin Gerekli Olan 64 Veri Seti Gönderilmiş ve Adresteki Veri Sıfırlanmış Demektir)*
		LDA #0000H	Akümülatöre 0000H Verisini Yükle
		STA $F003H	F003H Adresinde Kesme Kaydecisindeki Veriyi Sıfırla
		BRA #FF06H	FF06H Adresine Dallan*(Seri ve Klavye Kesmesi Başlangıç Adresine Şartsız Dallan)*
	İleri Tuş İçin İlişkisel Kodlar	0100H-0140H	İleri Tuşu İçin Gerekli Olan İlişkisel Kodlar Belirtilen Adreslerde Tanımlanmıştır.Sağ Tuş İçin Gerekli Olan Bütün Adımlar İleri Tuş İçin de Kullanılır.*(Yalnızca Ardışıl Olarak Çıkış Kaydecisine Gönderilen Her 8 Bitin İlk 4 MSB Biti Kullanılır.Diğer Bitler Sıfır Olarak Alınır.)*
	Geri Tuş İçin İlişkisel Kodlar	0150H-0190H	Geri Tuşu İçin Gerekli Olan İlişkisel Kodlar Belirtilen Adreslerde Tanımlanmıştır.Sol Tuş İçin Gerekli Olan Bütün Adımlar Geri Tuş İçin de Kullanılır.*(Yalnızca Ardışıl Olarak Çıkış Kaydecisine Gönderilen Her 8 Bitin İlk 4 MSB Biti Kullanılır.Diğer Bitler Sıfır Olarak Alınır.)*

	İstenen Açıda Dönüş Tuşu İçin İlişkisel Kodlar	0290H-02010H	İstenen Açıda Dönüş Tuşu İçin İlişkisel Kodlar Belirtilen Adreslerde Tanımlanmıştır.HerHangi Bir Tuş İçin Gerekli Olan Algoritma Bu Tuş İçin De Kullanılır.*(Yalnızca İstenen Açıda Dönüş Yeteniğini Görmek İçin Adım Sayısı İle İfade Edilen Hexadecimal Olarak Belirtilen Adresteki Sayı Değiştirilir.)*

SONUÇ ve ÖNERİLER

Yapılan tez çalışmasında eğitimsel amaçlı olarak tasarımı yapılan BZK.SAU ismindeki mikrobilgisayar mimarisi üzerinde tasarımı yapılan asenkron RS232 seri haberleşme devresi ilgili tez çalışmasının ana hedefi olmuştur. Bu suretle hali hazırda literatüre kazandırılmış olan BZK.SAU mikrobilgisayar mimarisi üzerindeki dış dünya ile veri haberleşmesi için gerekli olan çevresel birimlerin eksiklerinin giderilmesi ilgili tez çalışmasının özgün olmasını sağlamakla birlikte elde edilen bilgi ve tecrübe USB , I^2C , Ethernet gibi bilgisayar ve cihaz tabanlı seri haberleşme protokolleri oluşturulması ve gerçeklenmesi açısından önemlidir.

İlgili çalışmada bir mikroişlemci veya bir mikroişlemcide görülen donanım tabanlı kesme mekanizması ile başlayan kod yapısı BZK.SAU mikrobilgisiyar mimarisine gömülü BZK.SAU.ASSM dili ile oluşturulan yazılımsal yapı ile adım motor üzerine inşa edilmiş iki eksenli PAN/TİLT mekanizmasının açısal konum kontrolü gömülü sistem tabanlı kontrol sistemleri için temel bir çalışma olduğu düşünülmektedir. Günümüzde her yönlü kullanım alanı bulunan yeniden programlanabilir bir entegre olan FPGA' in hızlı ve parallel işlem yapma yeteneği her türlü iletişim, kontrol algoritmasının gerçek zamanlı olarak gerçeklenmesinde maliyet ve hız açısından oldukça iyi bir eğim yakaladığı görülmektedir. Bu açıdan asenkron RS232 seri haberleşme devre tasarımının özgün VHDL dili ile yapılmış olması ve klavye bağlı mikrobilgisayar mimarisi gömülü sistemlerin kontrol ve haberleşme yeteneklerinin eğitimsel amaçlı olarak tasarımı yapılmış sistemin geliştirebilir olmasının da temel bir çalışması olmuştur.

FPGA ve gömülü sistem tasarımlarının akademik ve endüstriyel çalışmalarda hızlı artan bir yer edinmesi yapılan tez çalışmasının önemli ve gelecek vadeden bir çalışma olduğunu göstermekle birlikte eğitimsel amaçlı mikrobilgisayar mimarisinin öğrenci tabanlı ilgili alanlardaki gerek lisans gerekse lisans üstü derslerde işlemci, veri haberleşmesi, yazılımsal tabanlı gerçek zamanlı mekaniksel kontrol konularda zevkli ve öğretici bir niteliği olduğu düşünülmektedir.

KAYNAKLAR

1. Vijaya V., FPGA Implementation of RS232 to Universal Serial Bus Converter, IEEE Symposium on Computer&Informatics, 978-1-61284-691-0, 237-242, 2011

2. Antonio-Torres D., A PicoBlaze-Based Embedded System for Monitoring Applications, International Conference on Electrical, Communications, and Computers, 978-0-7695-3587-6/09,173-177, 2011.

3. Li M., FPGA Based Design of A Control System for Electrophoresis on PCR Microfluidic Chip, International Conference On Computer Design And Applications, 978-1-4244-7164-5, Vol.3, 450-452, 2011

4. W. Dr. Garima B., Synthesis and Implementation of UART Using VHDL Codes, International Symposium on Computer, Consumer and Control, 978-0-7695-4655-1/12, 1-3, 2011

5. Bhatt D.V , Design of A Controller for A Universal Input/Output Port, 978-1-45771772-7/12, 1-6, 2011

6. Jusoh N. F.,An FPGA Implementation of Shift Converter Block Technique on FIFOfor RS232 to Universal Serial Bus Converter, IEEE Control and System Graduate Research Colloquium 2012 IEEE Control and System Graduate Research Colloquium, 978-1-4673-2036-8/12,219-224, 2012

7. Yu S., Implementation of a Multi-Channel UART Controller Based on FIFO Technique and FPGA , 2007 Second IEEE Conference on Industrial Electronics and Applications, 1-4244-0737-0/07, 2633-2638, 2007

8. ZhuY.,Study on the Communication Between FPGA and Observer Using ControllerArea Network and UART ,International Conference on Information, Networking and Automation (ICINA), 978-1-4244-8106-4,Vol.1,240-244, 2010.

9. Yi-yuan F., Design and Simulation of UART Serial Communication Module Based on VHDL, 978-1-4244-9857-4/11,1-4, 2011.

10. Roy B., Platform-Independent Customizable UART Soft-Core, Third International Conference on Intelligent Systems Modelling and Simulation, 978-0-7695-4668-1/12, 692-694, 2012.

11. LI X., Implementation of GSM SMS Remote Control System Based on FPGA, 978-1-4244- 7618-3 /10, IEEE, 1-4, 2010.

12. Paul R., Real Time Communication Between Multiple FPGA Systems in Multitasking Environment Using RTOS, IEEE, 1-5.

13. Abed S.,Design and Implementation of Interfacing two FPGAs, InternationalConference on Innovations in Information Technology (IIT), 978-1-4673-1101-4/12, IEEE, 72-77, 2012

14. Carrica D.,Novel Stepper Motor Controller Based on FPGA Hardware Implementation, IEEE/ASME Transactions on Mechatronics, VOL. 8, NO. 1, 1083-4435/03, 120-124, 2003.

15. K. Adam G.,Hybrid Neural Controller of a Stepper Motor for a Manipulator Arm, Fourth International Workshop on Robot Motion and Control, 321-326, 2004.

16. Wen Z., Analysis of Two-Phase Stepper Motor Driver Based on FPGA, 2006 IEEE International Conference 822 on Industrial Informatics, 0-7803-9701-0/06 ,821-826, 2006.

17. Kos D., Efficient Stepper Motor Torque Ripple Minimization Based on FPGA Hardware Implementation,1-4244-0136-4/06, 2006 IEEE,3916-3921, 2006.

18. Le Ngoc Q.,An Open-Loop Stepper Motor Driver Based on FPGA, International Conference on Control-Automation and Systems 2007, 978-89-950038-6-2-98560,1322-1326, 2007.

19. Rafael G., Digital System Control for Three-Degrees of Freedom Mechanical Arm with FPGA, Electronics, Robotics and Automotive Mechanics Conference 2008, 978-0-7695-3320-9/08 2008 IEEE, 496-501, 2008.

20. Cheng Chen T., High Performance Algorithm Realization on FPGA for Stepper Motor Controller , SICE Annual Conference 2008 , 2008, The University Electro-Communications,1390-1395,2008.

21. Satyam, Three-Dimensional Motion Control Using Embedded Controller and FPGA Technology, 5th International Conference on Electrical and Computer Engineering ICECE 2008, 978-1-4244-2015-5/ 2008 IEEE,851-856, Dhaka, Bangladesh ,20-22 December 2008

22. Le Quy N.,An Improved Method of Speed Damping for A Stepper Motor with a Smooth Speed Estimation, International Conference on Robotics and Biomimetics, 978-1-4244-2679-9/08, 1438-1443, Bangkok, Thailand, February 21 - 26, 2009.

23. Meshram (Thulkar) Mrs. U.,Robot Arm Controller Using FPGA, IMPACT-2009, 978-1-4244-3604-0/09, 2009 IEEE,8-11, 2009.

24. Ngoc Le Q.,Neural-Network -Based Low-Speed- Damping Controller for Stepper Motor With An FPGA, IEEE Transactions on Industrial Electronics, VOL. 57, NO. 9, September 2010, 0278-0046 , 2010 IEEE, 3167-3180, 2010.

25. Ali Z.,Development of a CPLD Based Novel Open Loop Stepper Motor Controller for High Performance Using VHDL, INES 2010 ,14th International Conference on Intelligent Engineering Systems , 978-1-4244-7652-7/10/, 2010 IEEE, 307-312,Las Palmas of Gran Canaria, Spain, May 5–7, 2010.

26. Mengda Y., A Research of A New Technique on Hardware Implementation of Control Algorithm of High-Subdivision for Stepper Motor, 2010 5th IEEE Conference on Industrial Electronics and Applications, 978-1-4244-5046-6/10/ 2010 IEEE,115-120, 2010.

27. Floyd,T.,Digital Fundamentals, Ninth Edition, Pearson International Edition.

28. Sarıtaş E.,Karataş S.,Her Yönüyle FPGA ve VHDL, Palme Yayıncılık, 2013.

29. http://www.fpgacentral.com/pld-types/prom- programmable- read- only-memory(Erişim Tarihi:24/06/2014)

30. http://webdocs.cs.ualberta.ca/~amaral/courses/329/webslides/Topic8/DocTime Dimsd038.htm (Erişim Tarihi: 24/06/2014)

31. http://www.fpga-site.com/faq.html#SPLD (Erişim Tarihi: 24/06/2014)

32. http://commons.wikimedia.org/wiki/File:ADC-DAC.jpg (Erişim Tarihi24/06/2014)

33. http://tr.wikipedia.org/wiki/FPGA (Erişim Tarihi: 24/06/2014)

34. http://www.fpganedir.com/FPGA/index.php (Erişim Tarihi: 24/06/2014)

35. http://www.fpgadeveloper.com/2011/07/ list-and-comparison - -of-fpga-companies. html (Erişim Tarihi: 24/06/2014)

36. http://sourcetech411.com/2013/04/top-fpga-companies--for--2013/ (Erişim Tarihi:24/06/2014)

37. Douglas J. S., VHDL & Verilog Compared & Contrasted - Plus Modeled Example Written in VHDL, Verilog and C, Tutorial 48.1, Design Automation Conference Proceedings, 1996.

38. Vulnei A.P., Circuit Design with VHDL, MIT Press, 2004

39. http://www.altera.com/literature/manual/archives / intro_to_quartus2.pdf (Erişim Tarihi:24/06/2014)

40. ftp://ftp.altera.com/up/pub/Tutorials/ DE2/ Digital_Logic/tut_quartus_ intro_vhdl.pdf (Erişim Tarihi:24/06/2014)

41. http://cseweb.ucsd.edu/classes/wi14/cse140L-a/modelsim_tut.pdf (Erişim Tarihi: 24/06/2014)

42. ftp://ftp.altera.com/up/pub/Altera_Material/10.1/Tutorials/Using_ModelSim.pdf (Erişim Tarihi:24/06/2014)

43. http://en.wikipedia.org/wiki/Serial_communication (Erişim Tarihi:24/06/2014)

44. Gumuskaya,H., Mikroişlemciler ve Bilgisayarlar, Alfa Yayınları, 5.Basım.

45. http://circuit-diagram.hqew.net/RS232-DB25-pinout_6876.html (Erişim Tarihi: 24/06/2014).

46. http://www.zytrax.com/tech/layer_1/cables/tech_rs232.htm(Erişim Tarihi: 24/06/2014)

47. Rizzoni Giorgio.,Principles and Applications of Electrical Engineering, McGraw- Hill International Edition, Fifth Edition.

48. http://en.wikipedia.org/wiki/Stepper_motor (Erişim Tarihi:24/06/2014)

49. http://www.mcu-turkey.com/step-motor-nedir-nasil-calisir/ (Erişim Tarihi: 24/06/2014).

50. http://www.massmind.org/Techref/io/stepper/wires.htm (Erişim Tarihi:24/06/2014)

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

52. OZTEKIN, H., Eğitim Amaçlı Yapılandırılabilir Modüler Donanım Üzerine Gömülü Isletim Sistemi Tasarımı, Doktora Tezi, Sakarya Üniversitesi, Bilgisayar ve Bilisim Mühendisliği, 2012.

53. http://www.uydudoktoru.com/acemiler-diyari/1787-standart-rs232-pin-baglanti-semasi.html (Erişim Tarihi:24/06/2014)

54. DE2-70_User_Manual.Pdf (Erişim Tarihi:24/06/2014)

ÖZGEÇMİŞ

1985 yılında Adana'da dünyaya gelen Kutlucan GÖRÜR, lise öğremini Adana Anadolu Lisesi'nde 2003 senesinde tamamladıktan sonra Çukurova Üniversitesi Elektrik-Elektronik Mühendisliği Bölümünden 2009 yılında mezun olmuştur.

2010 yılında Bozok Üniversitesi Elektrik-Elektronik Mühendisliğinde Arş.Görevlisi olarak başladığı görevine aynı bölümde devam etmektedir. Yine aynı üniversitede yüksek lisans eğitimine Fen Bilimleri Enstitüsü Mekatronik Mühendisliği Anabilim Dalında devam etmektedir.

İletişim Bilgileri

Adres : Bozok Üniversitesi Elektrik-Elektronik Müh. Bölümü Divanlı Yolu 10. km.

66100 YOZGAT

Telefon: 0505-7571213

E-posta: kutlucangorur@gmail.com