GÖMÜLÜ SİSTEMLERLE BİODENT FIRIN TASARIMI VE GERÇEKLEŞTİRİLMESİ

Süleyman UZUN

YÜKSEK LİSANS TEZİ
ELEKTRONİK BİLGİSAYAR EĞİTİMİ

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

MART 2011
ANKARA

Süleyman UZUN tarafından hazırlanan GÖMÜLÜ SİSTEMLERLE BİODENT FIRIN TASARIMI VE GERÇEKLEŞTİRİLMESİ adlı bu tezin yüksek lisans tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Rahmi CANAL
Tez Yöneticisi

Bu çalışma, jürimiz tarafından Elektronik Bilgisayar Eğitimi Anabilim Dalında yüksek lisans tezi olarak kabul edilmiştir.

Başkan : : Yrd. Doç. Dr. Uğur FİDAN

Üye : Yrd. Doç. Dr. Fecir DURAN

Üye : Yrd. Doç. Dr. M. Rahmi CANAL

Tarih : 25/02/2011

Bu tez, Gazi Üniversitesi Bilişim Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orjinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Süleyman UZUN

GÖMÜLÜ SİSTEMLERLE BİODENT FIRIN TASARIMI VE GERÇEKLEŞTİRİLMESİ

(Yüksek Lisans Tezi)

Süleyman UZUN

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ
Mart 2011

ÖZET

Bu tez çalışmasında, diş protezlerini pişirmede kullanılan biodent fırının denetimi, gömülü denetim sistemi ile gerçekleştirilmiştir. Biodent fırının denetimi, Xilinx Spartan- 3A XC3S700A FG484 FPGA ile tasarlanmış ve gerçekleştirilmiştir. Sistemin programlama arayüzü için Xilinx firmasına ait ISEWebPack9.2i programı yardımıyla VHDL programlama dili ve Şematik programlama kullanılmıştır. Sistemde; rezistans, ekovat ve selenoid valf denetlenmektedir. Sistemde, basınç sabit olup sıcaklık elle termostattan girilerek ve zaman ayarı iki seçimli bir anahtar yardımıyla girilerek sistem otomatik olarak çalıştırılmaktadır.

Bilim Kodu : 704. 3. 013
Anahtar Kelimeler : xilinx, FPGA, biodent fırın, gömülü sistemler
Sayfa Adedi : 90
Tez Yöneticisi : Yrd. Doç. Dr. M. Rahmi CANAL

BIODENT OVEN DESİGN WİTH EMBEDDED SYSTEMS AND IMPLEMENTATİON

(M.Sc. Thesis)

Süleyman UZUN

GAZI UNIVERSITY
INSTITUTE OF INFORMATICS
March 2011

ABSTRACT

In this thesis, dental prosthesis is used for cooking, biodent oven which is performed with control of embedded systems is realized. Biodent oven control, the FPGA with the Xilinx Spartan- 3A XC3S700A FG484 is designed and implemented. Company with the help of the Xilinx ISE ISEWebPack9.2i program for the system programming interface VHDL programming language is used for programming and schematic. On the system; heater, ekovat and selenoid valve is used as a load. Constant pressure, temperature and time can be manually entered from the termostat setting by entering the furnace with the help of a switch is executed automatically.

Science Code : 704. 3 013
Key Words : xilinx, FPGA, biodent oven, embeded systems
Page Number : 90
Adviser : Asist. Prof. Dr. M. Rahmi CANAL

TEŞEKKÜR

Tezimin hazırlanmasında yardımlarını, destekleyici tavırlarını esirgemeyen ve kıymetli bilgilerinden yararlandığım sevgili hocam Yrd. Doç. Dr. M. Rahmi CANAL'a sonsuz teşekkürlerimi sunarım. Ayrıca çalışmalarım boyunca yardım ve katkılarını esirgemeyen sevgili arkadaşlarım Mahir CANKAÇAR, Oktay YILDIZ, Ersin AKULUT' a ve çalışmalarım boyunca desteğini esirgemeyen sevgili eşime teşekkürlerimi bir borç bilirim.

İÇİNDEKİLER

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

KISALTMALAR LİSTESİ

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamalarıyla birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklama
ASIC	Application Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
CPLD	Complex Programmable Logic Device
EEPLD	Electrically-Erasable Programmable Logic Device
EPLD	Erasable Programmable Logic Device
EPROM	Erasable Programmable Read Only Memory
FPGA	Field Programmable Gate Array
FPIC	Field Programmable InterConnect
HDL	Hardware Description Language
I/O	Input/Output
IEEE	Institute of Electrical and Electronics Engineers
ISE	Integrated Sotware Environment
JTAG	Joint Test Advisory Group
LSB	Least Significant Bit
LSI	Large Scale Integration
MAX	Multiple Array matrix
MSB	Most Significant Bit
MXE	Modelsim xe
PAL	Programmable Logic Array
PALASM	PAL Assembler
PALCE	Programmable Array Logic Configurable Erasable
PLA	Programmable Array Logic
PLD	Programmable Logic Device
PROM	Programmable Read Only Memory
RAM	Random Access Memory
ROM	Read Only Memory

Kısaltmalar	Açıklama
RTL	Register Transfer Level
SPLD	Simple Programmable Logic Devices
SRAM	Static Random Access Memory
VHDL	VHISC High Level Description Language
VHISC	Very High Speed Integrated Circuit
DDR	Double Data Rate
CRC	Cyclice Redundancy Check
CLB	Configurable Logic Blocks
IOBs	Input/ Output Blocks
SPI	Serial Peripheral Interface

1. GİRİŞ

Protez diş yapımı için insanlardan alınan kalıplar içerisine akrilik maddesi doldurulur. Bu yumuşak maddenin sertleşmesi gerekmektedir. Bu maddenin pişirilmesine polimerizasyon adı verilmiştir. Polimerizasyonun farklı bir tanımı; bir maddenin iki veya daha fazla molekülünün, yeni bir bileşik yapmak üzere kimyasal olarak birleşmesi demektir. Polimerizasyon işlemi farklı şekillerde gerçekleştirilir. Birinci yöntem düdüklü tencere ile yapılan pişirme işlemidir. Bu yöntem günümüzde çok fazla kullanılmamaktadır (Şekil 1.1). Diğer bir yöntem ise bu çalışmada gerçekleştirilen elektrikli fırınlardır. Bu fırınlar polimerizasyon işlemini buhar ile gerçekleştirmektedir [1-2].

(a) Düdüklü tencere ile polimerizasyon[1]

(b) Biodent fırın ile polimerizasyon

Şekil 1.1. Buhar ile polimerizasyon sistemleri

Protez dişler belirli basınç ve sıcaklık altında pişirilirler. Pişirme işlemi 4-6 bar sabit basınçta, 90^{o}C - 120 ^{o}C sıcaklıkta 120sn ve 240sn' lik zamanda yapılır.

Piyasada kullanılan biodent fırınların kontrol kartlarında; sıcaklık analog, zaman ise sayısal kontrol tekniği ile ya röle mantığı ya da mikro denetleyici kullanılarak oluşturulmuştur. Piyasada kullanılan sitemlerin dezavantajları; kontrol kartında çok fazla elemanın kullanılması kartın çok karmaşık yapıya ve tam güvenilir olmamasına neden olmaktadır. Zamanlama ve basınç kontrolü için ayrı bir devre kullanımı kartın maliyetlerini yükseltmekte ve çok yer kaplamasına neden olmaktadır. Ayrıca basınç kontrolünde sadece alt sınır basıncı kontrol edilmektedir. Üst basınç sınırı kontrol

edilmediğinden suyun ısınması esnasında yükselen basınç bir süre sonra kazan dairesinin kapak kısmının patlamasına sebep olmaktadır. Sayısal tasarım mantığı kullanılarak oluşturulan kontrol kartları çok sayıda elemandan meydana geldiği için oldukça karmaşık ve kaba bir yapıya sahiptir. Bu tip devrelerin kontrol kartı tek bir karta yerleştirilemediğinden iki veya üç karta montajı yapılmıştır. Bu kontrol kartı fırına montaj edildiğinde ise çok yer kaplamakta ve maliyeti fazla olmaktadır. Kristal osilatör kullanılmadığı için de zamanlama hataları olmaktadır. mikro denetleyicili olanlarının ise çok sık arızalanması, içerisindeki programın çok sık silinmesi veya zarar görmesi ve/ veya mikro denetleyicinin sık sık reset almasıdır [1,3]. Bu şekilde yapılan pişirme işlemleri de diş protezinin özelliğini bozmakta, ömrünü kısaltmakta ve diş üzerinde istenmeyen şekil bozuklukları meydana gelmektedir. Diğer taraftan ekovat, rezistans ve selenoid valf gibi yükler rölelerle kontrol edilmektedir. Röleler gürültülü çalışmakta ve kontaklarında mekanik arızalar meydana gelmektedir. Yukarıda bahsedilen dezavantajlardan dolayı gömülü sistemlerin avantajları da göz önüne alınarak, yeni bir kontrol kartı tasarlanmış, doğruluğu bilinen sistemlerle kalibrasyonu yapılmış ve mevcut sistemlerle karşılaştırılması yapılmıştır. Tasarlanan sistemin temel avantajları; gömülü bir sistem olduğu için karmaşık bir yapıya sahip olmaması, kontrol kartının daha güvenilir olması, düşük maliyetli olması, az yer kaplaması ve çok daha az parçalı (discreet) eleman kullanılmasıdır. Ayrıca osilatörün entegrenin (XC3S700A) kendi içerisinde bulunması, hem zamanlama hatalarını tamamıyla ortadan kaldırmakta hem de fiziksel ortamdan etkilenmemekte ve bu şekilde hem maliyeti düşürülmekte hem de sistemin gürültüsü azaltılmaktadır.

Bu tezde tasarlanan ve gerçekleştirilen devre ile ilgili donanım ve yazılım özellikleri 2. bölümde incelenmiştir. Devrede sıcaklık ölçümünde kullanılan termostat ölçülebilecek sıcaklık aralığında kullanılabilecek özelliklere sahip olarak seçilmiştir. Ayrıca kontrol kartının en önemli elemanı olan mikro işlemci olarak Xilinx firmasına ait Spartan-3A XC3S700A serisi FPGA (Field Programmable Gate Array)' sı seçilmiştir [4]. Bu FPGA programlama kartı ve deney seti üzerinde bulunan eğitim seti şeklindedir. Biodent fırın ile ilgili performansın ve kod geliştirilmesinin hızlı olması açısından, geliştirilen özelliklerin ve kodların anında denenerek sonuçların zaman kaybetmeden görülmesi için bu eğitim seti kullanılmıştır. Bu kontrol kartında

ekovat, rezistans ve selenoid valf gibi yükler MOC 3041 entegresi ile kontrol edilmiştir. MOC entegreleri içerisinde optokupler ile diyak sürülmektedir [5]. Bu FPGA' yı korumak için kullanılan bir entegredir. Böylece sistem koruması, röleli kontrol sistemlerinden ve mikro denetleyici kullanılan sistemlerden daha güvenilir hale getirilmiştir.

Tasarlanan ve gerçekleştirilen devrenin çalışması ve akış şeması 3. bölümde incelenmiştir. Sıcaklık termostat yardımıyla istenilen değere ayarlanabilmektedir. Devrede kullanılan FPGA' nın programlanması için Xilinx firmasına ait ISE WebPack programı kullanılmıştır [4]. Bu program sayesinde programlama dili olarak VHDL (Very High Speed Hardware Description Language), Verilog ve Şematik olarak üç programlama dili bir arada kullanılmıştır. Sistem yazılımında, basınç ve sıcaklık kontrol edilirken zamanlamanın bağımsız olması için basınç ve sıcaklık kontrolü bir program, zamanlama için ayrı bir program yazılmıştır.

Bu tezin son bölümünde ise sonuçlar irdelenmiş, tasarlanan sistemde fırının ayarlanan sıcaklıkta ve basınçta ne kadar hassas bir şekilde ekovatı ve rezistansı devreye alıp devreden çıkardığı ölçülmüştür. Ayrıca tasarlanan kontrol kartı hakkında öneriler sunulmuştur.

2. TASARLANAN SİSTEM

Bu bölümde tasarlanan sistemde kullanılan elemanlar ve kullanılan yazılımlar anlatılmaktadır.

2.1. Tasarlanan Sistemde Kullanılan Elemanlar

Tasarlanan sistemde; Xilinx XC3S700A entegresi, ekovat, selonoid valf, rezistans, manometre, basınç algılayıcısı (prostat), termostat ve zamanlayıcı göstergesi (4 Digitli 7 segment Display) kullanılmaktadır.

Sistemin kontrolü XC3S700A FPGA' sı ile gerçekleştirilmiştir. Mikroişlemci basınç algılayıcısı, termostat ve sayıcıdan aldığı bilgilere göre ekovatı, rezistansı ve selonoid valfi devreye alıp, devreden çıkartma işlemini gerçekleştirmektedir.

Ekovat, buzdolabı motoru olarakta bilinen bu eleman kazan dairesi içerisindeki basıncı sağlamaktadır. Ekovatların boyutlarının küçük olması, daha az enerji harcaması, ucuz olması ve çok sessiz çalışması tercih sebebidir. Sistemde 220V 50Hz 3/8 134A 952K.Cal' lik güçlü bir ekovat kullanılmıştır.

Rezistans, kazan dairesi içerisindeki gerekli olan sıcaklığı sağlamaktadır. Sistemde 100W' lık, homojen ısı dağılımını sağlamak için de kelebek tipi bir rezistans kullanılmıştır.

Termostat olarak, maksimum 300^{o}C' ye kadar ölçüm yapabileni seçilmiştir. Sıcaklık değeri el ile ayarlanmaktadır.

Basınç algılayıcısı, maksimum 6 bar' a kadar ölçüm yapabilmektedir. Basınç ayarı bir tornavida yardımıyla yapılmakta ve basınç değişimi manometreden görülmektedir.

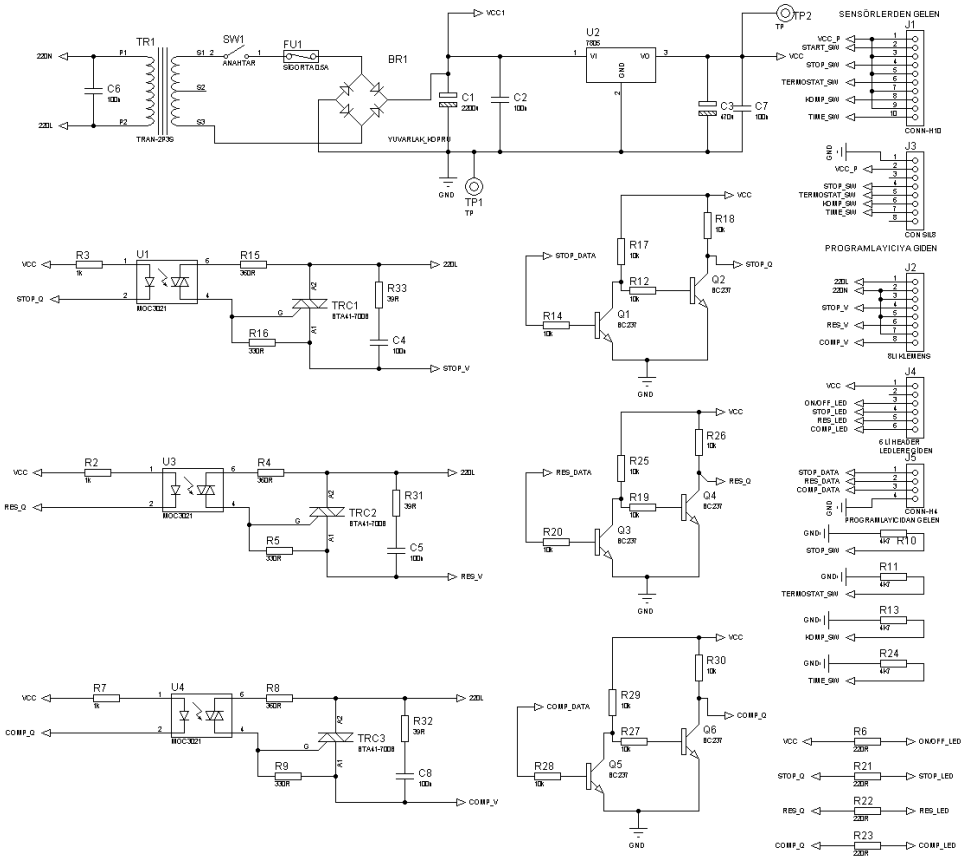
Selenoid valf, biodent fırında pişirme işlemi bittikten sonra kazan dairesi içerisindeki basınç, buhar ve sıcak suyun tahliyesinde kullanılmaktadır.

Manometre, kazan dairesi içerisindeki basıncın değerini göstermektedir. Maksimum 10bar' a kadar basınç göstermektedir.

Tasarlanan ve gerçekleştirilen sistemde kullanılan bu elemanların detaylı bilgileri ekler bölümünde verilmiştir.

2.2. Kontrol Kartı ve Bileşenleri

FPGA' lar da devre tasarımı için kullandıkları bazı basit devreler gerektirirler. Şekil 2.1' de FPGA' nın kullandığı devrelerden birtanesi görülmektedir. FPGA' lı devre tasarımlarında Clock (saat) devresine ihtiyaç yoktur, bu mikro denetleyicilerle oluşturulan sistemlere göre çok önemli bir avantajdır.

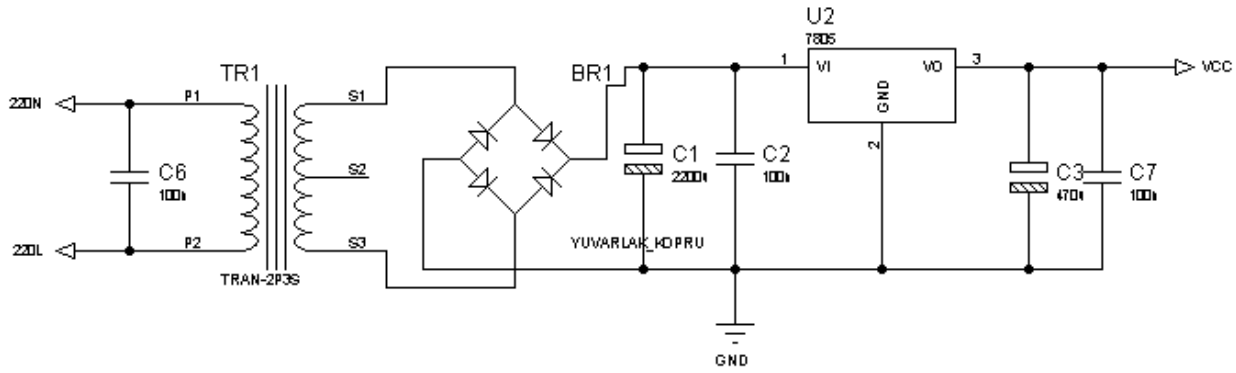


Şekil 2.1. Kontrol kartı devre şeması

FPGA kullanılan kontrol kartında, güç kaynağı, zamanlayıcı göstergesi, opto-izolatör ve triyak bulunmaktadır.

2.2.1. Güç kaynağı

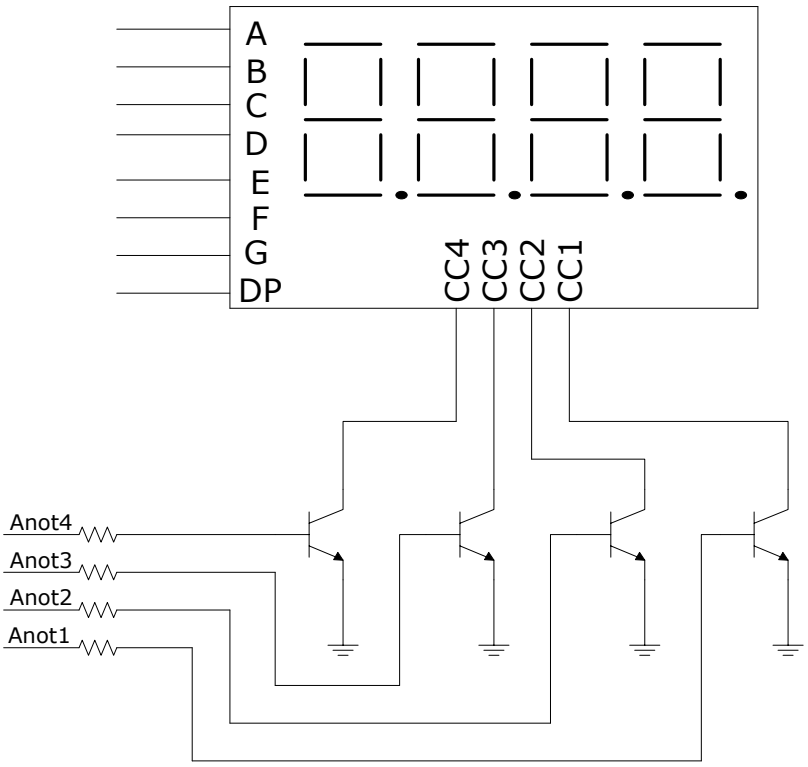
Tasarlanan kartta güç kaynağı olarak 7805 pozitif voltaj regülatörü kullanılmıştır (Şekil 2.2). Pozitif voltaj regülatörleri için 78XX üç uçlu yarı iletken gerilim düzenleyiciler kullanılmıştır [6].



Şekil 2.2. 7805 Pozitif gerilim güç kaynağı

2.2.2. Dört dijitli yedi segment display

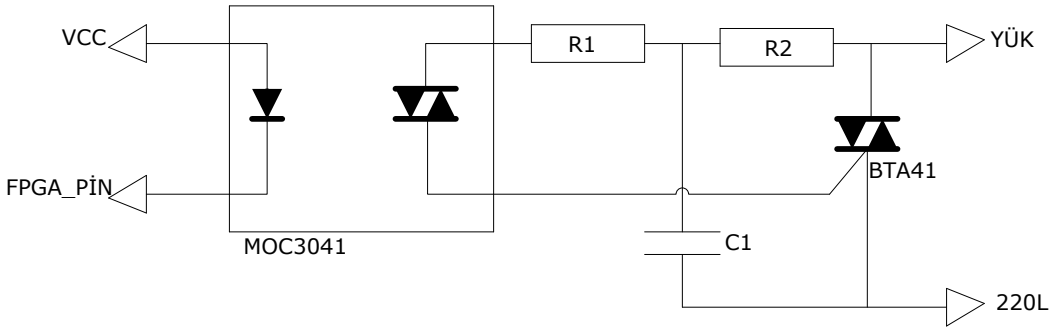
Dört dijitli bir sayıyı göstermek için display, ayrı hatlardan sürülmektedir. Zaman göstergesinde üç basamaklı sayılar gösterildiği için bu tip bir display seçilmiştir. Bu tasarımda kullanılan dört digitli displayin her bir digiti transistörler yardımıyla seçilerek aktif hale getirilmektedir. Bu bizlere mikroişlemcinin daha az portunu kullanmamızı, baskı devre üzerindeki karışıklığı azaltması ve daha az donanım kullanmamızı sağlamıştır (Şekil 2.3) [7].



Şekil 2.3. Dört dijitli displeyler

2.2.3. Opto – izolatör

Kartta, Opto – izolatörler kaynak ve yük arasındaki elektriki yalıtımı sağlamaktadır. Opto – izolatör çıkışında bir diyak ve girişindeki bir led' den ibarettir. Aralarında elektriki bir bağlantı yoktur. Genel amaçlı akım uygulamaları için yüksek akım kazançlı MOC3041 darlington transistörü olan opto – izolatör seçilmiştir. Uygulamada çıkış arabirimine bağlanan tüm yükler için MOC3041 kullanılmıştır (Şekil 2.4). Böylece devre kontrol elemanları ile mikroişlemci portu arasında yalıtım sağlanmıştır [7].



Şekil 2.4. Opto – izolatör MOC3041 ve triyak bağlantı devresi

2.2.4. Triyak

Tasarlanan kontrol kartında, triyak olarak 600V / 40A özelliğine sahip BTA41 kullanılmıştır [7]. Triyak, devrede elektrik kontrol elemanı olarak kullanılmıştır. Devrede kontrol edilen elemanları sürmektedir (Şekil 2.1).

2.3. Sistemin Yazılımında Kullanılan Programlama Dilleri

Sistemin yazılımı Xilinx firmasına ait ISEWebPack 9.2i programı yardımıyla gerçekleştirilmiştir. Bu program blok, kod ve şematik olarak üç farklı programlama seçeneği sunmaktadır. Yazılımda kod ve şematik kullanılmıştır. Kod olarak, VHDL programlama dili ve Verilog programlama dilleri kullanılmıştır. Şematik programlama, VHDL programlama dili ve Verilog programlama dilleri hakkında kısa genel bilgiler verilmiştir. Sistemin kodları ise ekler bölümünde verilmiştir.

2.3.1. Donanım tanımlama dilleri (HDL)

Donanım tanımlama dilleri, sayısal sistemlerin simülasyonu, modellenmesi test edilmesi ve tasarımı amacıyla kullanılırlar. Bazı donanım tanımlama dilleri, sayısal devrelerin şematik gösteriminin yerine geçen sembol ve rakamlar kullanırlar. Mevcut HDL (Hardware Description Language) yazılımları, simulator ve donanım

sentezleme programları içerir. HDL'ler, gün geçtikçe karmaşıklaşan tasarımların gerçekleştirilmesini mümkün kılmak amacıyla ortaya atılmışlardır. HDL tabanlı sistem tasarımı, çok sayıda tasarımcının yada tasarımcılar grubunun bir arada çalıştığı büyük projelerin gerçekleştirilmesinde önemli faydalar sağlamaktadır. Çünkü HDL yapısal geliştirmeyi mümkün kılmaktadır. Temel mimari konusundaki kararlar verildikten, temel bloklar belirlendikten ve bunların birbirleri ile olan bağlantıları ortaya konulduktan sonra, tasarım projesi, birbirinden bağımsız alt projelere bölünebilir [8-11].

2.3.1.1. Donanım tanımlama dili kullanmanın avantajları

Donanım tanımlama dillerine dayanan tasarım yönteminin, geleneksel kapı düzeyi tasarım yöntemine göre bir çok avantajı vardır.

1. Tasarım fonksiyonlarının doğruluğu, tasarım sürecinin en başında test edilebilir ve HDL dilinde tanımlanmış bir tasarım üzerinde simülasyonu yapılabilir.
2. Bir HDL sentezleme aracı kullanılarak, hazırlanan HDL tanımlaması otomatik olarak verilen bir teknolojide gerçeklemeye dönüştürülebilir. Bu adım eskiden mevcut olan kapı düzeyi tasarım problemlerini, devre tasarımına harcanan uzun zamanı ve elle yapılan tasarımda karşılaşılan diğer sorunları ortadan kaldırmaktadır.
3. Sentezleme aracının mantıksal uyumlaştırma özelliği kullanılarak, sentezlenmiş tasarım daha hızlı bir hale dönüştürülebilir. Sentezlenmiş ve uyumlaştırılmış devrelerden kazanılan tecrübeler VHDL tanımlamasına yansıtılabilir.
4. HDL tanımlamaları, bir tasarımın ve fonksiyonun teknolojiden bağımsız dokümantasyonunu mümkün kılmaktadır. Bir HDL tanımlaması, bir netlist yada bir şemaya göre daha rahat okunur ve anlaşılır. Başlangıçtaki HDL tanımlaması, teknolojiden bağımsız olduğundan, aynı tanımlama daha sonra farklı bir teknoloji için kullanılabilir.
5. VHDL, pek çok yüksek seviyeli bilgisayar dillerinde olduğu gibi, oldukça sıkı bir tip kontrolü yapmaktadır. 4 bit genişliğinde işaret bekleyen bir bloğa 3 yada 5 bitlik işaret uygulanamaz. Bu, HDL tanımlaması derlendiğinde bir hataya neden

olur. Tip kontrolü, bu hataların sentezleme işleminden önce ortaya çıkarılmasını sağlamaktadır [12].

2.3.2. VHDL donanım tasarım dili

Donanım açıklama dilleri genel ve özel amaçlı olmak üzere ikiye ayrılmaktadır. VHDL (Very High Speed Hardware Description Language, Yüksek Seviyeli Donanım Tanımlama Dili) genel amaçlı bir donanım açıklama dilidir. VHDL, çok kullanılan donanım tanımlama dilidir. VHDL donanım tanımlama dilinin özellikleri şunlardır;

- Tasarı hiyerarşik bir şekilde basit olabilmektedir.
- Her bir tasarım elementi, iyi tanımlanmış bir ara yüze ve hassas bir davranış özelliğine sahiptir.
- Davranışsal özellikler bir algoritma ya da elementin çalışmasını tanımlayan gerçek bir donanım yapısını kullanabilmektedir.
- Uyuşma, zamanlama ve zamanlayıcı ile denetim işlemlerinin tamamı modellenebilmektedir. VHDL senkronize ardışık devre yapısını ardışık olmayana göre kontrol altında tutabilmektedir.
- Mantıksal işlem ve tasarımın zamanlama davranışının benzetimi yapılabilmektedir [13].

2.3.2.1. Sentezleme

Sentezleme tasarım tanımlamasının herhangi bir seviyeden, daha düşük diğer bir seviyeye çevrilmesi olarak tanımlanabilir. Bu çevirme işlemi, C programlama dilinde, yüksek seviyeli dilin makina diline çevrilmesine benzetilebilir. Sentezleme sırasında kullanılan araçlar, HDL tanımlama, zamanlama ve alan istekleridir. Sentezleme işlemi tasarımcı tarafından belirtilen teknoloji kütüphanesindeki elemanlar kullanılarak yapılır. Sentezleme sonuçları, optimal devre şeması, beklenen performans ve sentezlenmiş devrenin kapladığı alandır. Aşağıda kısaca davranışsal

sentezleme, saklayıcı iletişim seviyesinde (RTL, Register Transfer Level) sentezleme ve mantık sentezlemeden bahsedilmiştir.

Davranışsal sentezleme, C benzeri algoritmik yazılmış tasarımın RTL tanımlamaya çevrilmesidir. RTL seviyesindeki tasarım veri yolları, bellek elemanları ve kontrol birimleri içerir. RTL sentezleme, saklayıcı iletişim fonksiyonlarından, bir ardışıl devre için devre şemasının oluşturulmasıdır. Mantık sentezleme, boolen fonksiyonlarının birleşimsel devre elemanlarıyla gerçeklenmesidir [12].

2.3.2.2. Tasarım akışı

VHDL tabanlı tasarım işleminde birkaç basamak vardır, bunlar tasarım akışı olarak adlandırılır. Bu adımlar herhangi bir HDL tabanlı tasarım işlemine uygulanabilir. Büyük lojik tasarımlar da yazılım programlarındaki gibi genelde hiyerarşiktir ve VHDL modelleri tanımlamada iyi bir çalışma çerçevesi (framework) verir.

Son adım modeller için VHDL kodlarının yazılması, bunların bağlaşımı ve bunların dahili detaylarının açıklanması aşamasıdır. VHDL text tabanlı bir dil olduğu yıllarda, prensip olarak bu görevi yerine getirmek için herhangi bir text editörünü kullanmak zorundaydı. Bununla birlikte birçok tasarım çevreleri bu görevi daha kolay yerine getirmek için özel VHDL text editörlerini kullanır. Böylece editörler, VHDL anahtar kelimesini otomatik tanıyan, sıklıkla kullanılan program yapılarını, şablonlarını kurma, yazım kurallarını otomatik kontrol etme ve derleyiciye tek tuşla ulaşım gibi özellikleri içerir [9].

VHDL derleyicisi yazım kuralı hataları açısından kodların analizini yapar ve diğer modüllerin uygunluğu açısından kodları kontrol eder. Diğer program denemelerinde olduğu gibi, diğer kodları derlemek için kodlama işleminin sonuna kadar beklemek zorunda değiliz.

VHDL simülatörü, fiziksel devreyi inşa etmeksizin çıkışları gözlemenize, tasarım girişlerine uygulamanıza ve tanımlamanıza izin verir. Küçük projelerde, dijital

tasarımlarınızın girişlerini üretebilir ve çıkışları manuel olarak elde edebilirsiniz. Ama büyük projelerde, VHDL otomatik olarak girişleri uygulayan ve bunları beklenen çıkış değerleri karşılaştıran Test Benches oluşturmanıza izin verir.

Simülasyon ile tasarımın kontrol edilmesi büyük bir adımdır. Simülasyonlu çıkışları üreten simüle edilmiş devreyi incelememizi sağlar. Beklenen devre çalışmalarını tanımlar. Bu kategoride tasarım hataları bulmak yüksek öneme sahiptir. Eğer hatalar daha sonra bulunacak olursa, back-end diye adlandırılan bütün adımlar tekrarlanmalıdır.

En az iki boyutlu kontrol vardır, fonksiyonel tanımlamada, zamanlama özelliklerine bağlı devrelerin mantıksal operasyonlarıyla çalışılır. Kapı gecikmeleri ve diğer zamanlama parametreleri sıfır olacak şekilde arzu edilir. Zamanlama kontrollerinde göz ardı edilmiş gecikmeleri içeren devrelerle çalışarak, flip-flop gibi ardışık elemanlar için kurma (setup), tutma (hold) ve diğer zamanlama gereksinimlerini tanımlanabilir [10].

2.3.2.3. VHDL'de kullanılan temel yapılar

VHDL'de kullanılan temel yapıları ele alacak olursak bunlar:

- Modül Bildirimi (Entity Declaration)
- Mimari (Architecture)
- Paket (Package)
- Konfigürasyon (Configuration)
- Tasarım Kütüphaneleri (Design Libraries)

Sayısal bir sistem, modüllerin bir hiyerarşi içinde birleştirilmesiyle oluşur. Her modül, VHDL'de bağımsız ve tektir. Bir tasarım modülü, giriş çıkışları ve fonksiyonu tamamen belirlenmiş bir donanım yapısını temsil eder. Her modülün iki bölümü vardır; modül bildirimi ve mimari. Modül bildirimi, tasarımın dış

bağlantılarını, mimari ise içeride yapılacak işlemleri gösterir. Paket, pek çok modül tarafından kullanılacak genel bilgileri içerir. Düzenleme, yapısal tanımlamada kullanılacak alt sistemlerin modül ve mimarilerini bir araya getirir. Bir VHDL tasarımı, her biri tasarım kütüphanelerinde bulunan ve önceden tanımlanmış bir çok üniteyi içerir [8].

2.3.2.3.1. Modül bildirimi (Entity Declaration)

VHDL' de her blok kendi başına bir devre olarak düşünülür ve entity olarak adlandırılır. Modül bildirimi, verilen lojik fonksiyon için bütün giriş ve çıkışlarını tanımlar. Yani lojik fonksiyonun dış dünya ile bağlantısını tanımlar. Her VHDL tasarımı mutlaka en az bir entity içerir. Modül bildirimi genel olarak aşağıdaki gibi gösterilir;

Entity entity ismi **is**
    Port(girişleri ve çıkışları tanımlama);
End entity ismi;

Port; port giriş ve çıkış işaretidir. Port ifadesi, her işaret için, port tanımlayıcı, port yönü ve port veri tipi belirlenmelidir. Portta 3 yön kullanılır. Giriş için "in", çıkış için "out" ve çift yönlü portlar için "inout" kullanılır.

Kullanılan bazı veri tipleri;

- **Bit:** 0 ve 1 değerini alabilir. Örneğin;

  Entity half_adder **is**
  Port(A, B: **in** BIT; C, D: **out** BIT);
  End half_adder;

- **Bit-vector:** aynı isim altında bir dizi 0 ve 1' leri göstermek için kullanılır.

Örneğin;

```
Entity alu is
 Port(S: IN BIT_VECTOR(2 DOWNTO 0);
      A, B: IN BIT_VECTOR(3 DOWNTO 0);
      F: OUT BIT_VECTOR(3 DOWNTO 0));
End alu;
```

- **İnteger:** -2 147 483 647 ile +2 147 483 647 sayı aralığını kapsar.

```
Entity termostat is
      Port( desired_temp, actual_temp: in integer;
            Heater_on: out boolean);
End termostat;
```

- **Boolean:** Doğru ve yanlış olmak üzere iki değerli bir veri tipidir.
- **Natural:** 0’ dan başlayıp istenen bir limit değerine kadar tamsayılar için kullanılır.
- **Positive:** 1’ den başlayıp istenen bir limit değere kadar tamsayılar için kullanılır.
- **STD_LOGIC (STD_LOGIC_VECTOR):** Aynı isim altında bir dizi 0 ve 1’ leri göstermek için kullanılır. Örneğin;

```
Entity alu is
 Port(S: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
      A, B: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      F: OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
End alu;
```

- **Real:** Reel sayıların tanımlandığı, -1 0E38’ den +1 0E38’ e kadar olan aralığı tanımlar.

2.3.2.3.2. Mimari (Architecture)

Lojik fonksiyonun işlevi mimari (architecture) tarafından belirlenir. Her entity için bir mimari tanımlanır. Genel kullanımı aşağıda gösterildiği gibidir;

Entity entity ismi **is**
    Port(A, B: **in** bit; X: **out** bit);
End entity ismi;
Architecture architecture ismi **of** entity ismi **is**
Begin
İşlemler;
End architecture ismi;

Begin – end blokları arasındaki satırlar mimarinin nasıl bir işlem gerçekleştireceğini belirtir.

2.3.2.3.3. Paket (Package Declaration)

Bir paket, veri tiplerinin, altprogramların, sabitlerin, v.b. tanımlamalardan oluşur. Bu takım çalışması için kullanışlıdır. Bu farklı tasarımcıların modüllerinin bir araya getirilip VHDL modelinin oluşturulmasını kolaylaştırmaktadır.

Bir paketi, başlık (header) ve gövde (body) olarak bölümlere ayırmak mümkündür. Paketin başlığı, fonksiyon veya prosedürün prototip belirtimi, gerekli tüm veri tiplerinin tanımlanmasını içerir. Altprogramın gerçekleştirimi gövdede yer almaktadır. Bu derleme sürecini kolaylaştırmaktadır. Çünkü kısa paket başlıkları kullanılan VHDL kodunun belirtimi (decleration), tanımlamalara (definitions) uyumlu olmalıdır.

Bir pakete bir kullanım ifadesiyle referans verilir. Anahtar sözcük ‘use’ u seçilen isim takip eder. Bu isim paketin yer aldığı kütüphanenin ismini içerir. Paket isminin

kendisi ve nesne ismi referans gösterilmiş olunur. Genellikle ‘all’ anahtar sözcüğü pakette görünen tüm objelere referans göstermek için kullanılır.

2.3.2.3.4. Konfügrasyon (Configuration Declaration)

Varlıkların tanımlanması ve örneklenmesi gerçekte kullanılan VHDL modüllerinden farklıdır. VHDL konfigürasyon işinde bileşenlerin, tam bir tasarım yapmak için varlık, mimari çiftine bağlantı oluşturulur. Özet olarak bir bileşenin tanımlanması kesin bir soket tipinin bileşen örneklenmesiyle belirlenmesi demektir. Bir aygıtın örneklenmiş sokete gerçekten eklenmesi konfigürasyon ile yapılmaktadır. Genel kullanımı aşağıda gösterildiği gibidir;

Entity entity ismi **is**
    Port(A, B: **in** bit; X: **out** bit);
End entity ismi;
Architecture architecture ismi **of** entity ismi **is**
Begin
İşlemler;
End architecture ismi;
Configuration konfigürasyon ismi **of** entity ismi **is**
        For architecture ismi
Konfigürasyon işlemleri;
End for;
End configuration konfigürasyon ismi;

Varlıkla mimari arasındaki ilişki konfigürasyonda kullanılan simülasyonla desteklenir, son tasarım hiyerarşisini oluşturur. Bu en üst seviye varlık mimarisinin seçimini içerir. Konfigürasyon VHDL’ deki sentezlenebilir ve simüle edilebilen tek objedir. Konfigürasyon sürecinin simülasyon amaçlı elle kontrolü mümkün olduğundan, sentez araçları her zaman varsayılan kural kümesine uygular.

Bunu başarılı bir şekilde gerçekleştirmek için bileşen isimleri ile var olan varlık isimleri birbiriyle uyuşmalıdır. Buna ek olarak, port isimleri, modlar ve veri tipleri birbiriyle uyumlu olmalıdır. Bileşen tanımlamadaki portların sayısı önemli değildir.

Örneğin;

Configuration CFG_FULLADDER **of** FULLADDER **is**
For STRUCT
For MDULE2: HALFADDER
Use entity work.B(Gate);
Port map(U=> A, V=>B, X=>SUM, Y=>CARRY);
End for;
For others: HALFADDER
Use entity work.A(RTL);
End for;
End for;
End CFG_FULLADDER;

FULADDER varlığı için STRUCT mimarisi seçilmiştir. Bu for döngüsü içerisinde, varlıklar ve mimariler alt bileşenler için seçilmiştir. Bunun için for ifadesi yeniden kullanılmıştır. For anahtar sözcüğünden sonra kullanılan ilk isim bileşen örneklemenin adıdır. Bundan sonra ‘:’ kullanılır ve bileşenin adı kullanılır. ‘all’ anahtar sözcüğü eğer bir bileşenin tüm örnekleri adreslenmişse kullanılabilir. For döngüsü içinde “use” ifadesi belirtilen objeye kadar olan yoldan varlığı seçer. Açıkça değiştirilmediği sürece, tüm VHDL objeleri kütüphane işi üzerinde derlenir. Seçilen varlık için mimari, ‘(‘’)’ çifti içine alınır. B varlığının port isimleri bileşen tanımlamasına uymadığı için port haritası ifadesine gerek vardır. Haritayı sırayla yapmak mümkündür. Ancak okunabilirlik açısından resmi parametrelerle kullanılan isimler eşleştirilmelidir. Ayrıca henüz konfigüre edilmemiş diğer bileşenler ‘other’ anahtar sözcüğüyle tanımlanabilir. MODULE2 için RTL mimarisi kullanılmıştır, bunun için port haritasına gerek yoktur çünkü others kullanılmıştır [11,14-17].

2.3.3. Verilog donanım tasarım dili

Verilog ilk olarak 1984' lerde Gateway Design Automation şirketi tarafından donanım modelleme dili olarak başlatılmıştır. Verilog simülatörü 1985' lerin başında ilk olarak kullanılmaya başlanmıştır ve 1987' nin sonlarında büyük ölçüde gelişmiştir. En büyük gelişme, Verilog-XL idi. Bu bir dizi özellik ve adı kötüye çıkmış XL algoritmasını kapı seviyesi simülasyon için en etkili biçime getirecek şekilde gerçekleştirmiştir.

Verilog bir donanım tanımlama dilidir. Bir donanım tanımlama dili sayısal sistemleri tanımlamak için kullanılan bir dildir. Örneğin, bir mikroişlemci veya bir bellek veya basit bir flip- filop. Verilog, sanayide donanım tasarımı için kullanılan HDL dillerinden biridir. Bize, davranışsal seviyede (Behavior Level), Yazmaç transfer seviyesinde (Register Transfer Level(RTL)), Kapı seviyesinde (Gate Level) ve anahtarlama seviyesinde (Switch Level) sayısal bir tasarım yapmamıza izin verir. Verilog donanım tasarımcılarına tasarımlarını davranışsal yapıyla belirtmesine izin verir [10].

2.3.3.1. Tasarım şekilleri

Verilog birçok donanım tanımlama dilindeki gibi aşağıdan- yukarıya (Bottom- up) veya yukarıdan aşağıya (Top- Down) metodolojiye izin vermektedir.

2.3.3.1.1. Aşağıdan- yukarıya (bottom- up) tasarım

Elektronik tasarımdaki geleneksel metot aşağıdan- yukarıyadır. Her bir tasarım standart kapılar kullanılarak kapı seviyesinde (gate level) gerçekleşmektedir. Yeni tasarımlardaki karmaşıklık arttıkça bu metodun uygulanmasını neredeyse imkansız kılmıştır. Yeni sistemler ASIC veya binlerce transistör içeren mikro işlemcilerden oluşmaktadır. Bu geleneksel tasarım yerini yeni yapısal hiyerarşik tasarım metotlara bırakmak zorundadır.

2.3.3.1.2. Yukarıdan aşağıya (Top- down) tasarım

Her tasarımcının istediği tasarım şekli yukarıdan aşağıya olanıdır. Gerçek bir yukarıdan aşağıya tasarım erken test, farklı teknolojilerin kolay değişimine, yapısal sistem tasarımlarına ve birçok diğer öneri avantajlarına izin verir. Ancak tam bir yukarıdan aşağıya tasarımı takip etmek çok zordur. Bu nedenlerden dolayı, birçok tasarım her iki metodun karışımı şeklinde, her bir tasarım sitilinin önemli elemanları gerçeklenerek tasarlanmaktadır.

2.3.3.2. Verilog soyutlama seviyeleri (Abstarction Levels)

Verilog birçok farklı seviyedeki soyutlamayı desteklemektedir. Bunlardan en önemlileri; Davranışsal Seviye (Behavioral Level), Yazmaç Transfer Seviyesi (Register- Transfer Level- RTL), Kapı Seviyesi (Gate Level)' dir.

2.3.3.2.1. Davranışsal seviye (Behavioral Level)

Bu seviye, aynı zamanda algoritmalarla bir sistem tanımlar. Her bir algoritmanın kendisi sıralıdır, bunun anlamı bir biri ardına gerçeklenen komutlar kümesini içermektedir. Fonksiyonlar, görevler ve her zaman blokları temel elemanlarıdır. Tasarım yapısal gerçeklemesiyle alakalı değildir.

2.3.3.2.1.1. Her zaman (always) bloğu

İsminden de anlaşılacağı gibi, bir always (her zaman) bloğu her zaman gerçekleştirilir, başlangıç bloğundaki gibi sadece bir kere değil. Bir her zaman bloğunun hassasiyet listesi olabilir veya onunla ilgili bir gecikme olabilir.

Hassasiyet listesi, always bloğuna kodun ne zaman gerçekleştirileceğini söyler, aşağıdaki örnekte gösterildiği gibi "always" korunmuş sözcüğünden sonra @ sembolü bloğun ne zaman tetikleneceğini parantez içinde belirtilir. Örneğin:

```
always @ (a or b  or sel)
begin
    y = 0;
    if (sel == 0) begin
        y = a;
    end else begin
        y = b;
        end
end
```

yukarıdaki örnek 2:1' lik bir çoklayıcı (mux-multiplexer) örneğidir. Giriş değerleri "a" ve "b" ile "sel" giriş seçmek için ve "y" de çıkış için kullanılmıştır. Herhangi bir kombinasyonel mantıkta, çıkış giriş değiştiğinde değişir. Bu teori always bloğuna uygulandığında bunun anlamı always bloğunun içindeki kod giriş değişkenleri değiştiğinde gerçekleştirilmektedir. Bu değişkenler hassasiyet listesine eklenmiştir ve a, b, sel olarak isimlendirilmişlerdir.

2.3.3.2.1.2. Görev (Task) ve fonksiyon (function) bloğu

Eğer daha önce kullandığımız şeyleri tekrar tekrar kullanıyorsak, verilog diğer programlama dillerindeki gibi adres tekrarlı kullanılmış kod yani görev (task) ve fonksiyonları (function) destekler. Aşağıdaki örnekte çift parity (even parity) hesaplamak için kullanılan koddur.

```
Function parity;
input [31 : 0] data;
integer i;
begin
    parity = 0;
    for (i = 0; i < 32; i = i+1) begin
    parity = parity ^data [i];
    end
```

```
end
endfunction
```

Fonksiyonlar ve görevler aynı söz dizimine sahiptir; görevlerin gecikmeleri olabilir fakat fonksiyonların gecikmesi olamaz. Bunun anlamı fonksiyonlar kombinasyonel mantığı modellemek için kullanılır. Fonksiyonlar bir değer döndürebilir fakat görevler döndüremez.

2.3.3.3. Yazmaç transfer seviyesi (Register transfer level, RTL)

Yazmaç transfer seviyesi kullanarak tasarlama, bir devrenin işlemlerini ve yazmaçlar arasındaki verilerin transferlerinin karakteristiğini belirtir. Harici bir saat kullanır. RTL tasarım kesin zaman kısıtları içerir. İşlemler kesin bir zamanda gerçekleştirecek şekilde planlamıştır. Modern RTL kodun tanımı, "sentezlenebilen herhangi bir koda RTL kodu" denir.

2.3.3.4. Verilog kontrol ifadeleri

Kontrol ifadelerinde, if, repeat, for, case, v.b kullanılmaktadır. İlk bakışta C programlama diline çok benzediği görülmektedir. Her ne kadar C dili gibi görünse de verilog bir HDL' dir. Bundan dolayı tanımlamalar donanıma dönüştürülmelidir. Bunun anlamı kullanılan ifadelere dikkat edilmelidir.

2.3.3.4.1. If-else

If-else ifadesi bir parça kodun yürütülüp yürütülmeyeceğinin koşulunu kontrol eder. Eğer bir kod koşulu sağlıyorsa, kod yürütülür. Değilse kodun diğer kısmı işletilir. Örneğin;

```
İf (enable == 1'b1) begin
Data= 10;
Address= 16 'hDEAD;
```

```
Wr_enable= 1 ‘b1;
End else begin
Data = 32’b0;
Wr_enable = 1 ‘b0;
Address = address +1;
End;
```

2.3.3.4.2. Case

Case ifadesi, eğer bir değişkenin birden çok değer için kontrol edilmeye ihtiyacı varsa kullanılmaktadır. Adres çözücü gibi girişin bir adres olduğunda, alabileceği tüm değerler kontrol edilmelidir. Birçok if-else ifadesi yerine bu kontrol komutu çok rahatça kullanılmaktadır.

Case ifadesi, korunmuş sözcük case ile başlar ve yine korunmuş sözcük endcase ile biter. Verilog da kod bloklarını ayırmak için parantez işareti kullanılmaz. Durumlar iki nokta ile takip edilir ve yürütmek istediğiniz ifade, bu iki ayıraç arasında listelenir. Örneğin;

```
Case (address)
0 : $display (“saat 11:40”);
1 : $display (“Okuldayım”);
2 : $display (“Dersteyim”);
Default : $display (“Evdeyim”);
Endcase
```

2.3.3.4.3. While

Normalde gerçek hayattaki modellemelerde kullanılmaz, fakat test bençlerde kullanılır. Diğer ifade bloklarında olduğu gibi begin ve end ile sınırlandırılırlar. Örneğin;

```
Module counter  (clk, rst, enable, count);
İnput clk, rst, enable;
Output [3:0] count;
Reg [3:0] count;
Always @ (posedge clk or posedge rst)
İf (rst) begin
    Count <= 0;
End else begin : COUNT
    While (enable) begin
        Count <= count +1;
        Disable COUNT;
    End
    End
Endmodule
```

2.3.3.4.4. For döngüsü

Verilog' daki for döngüsü neredeyse C programlama dilindekiyle aynıdır. Tek fark verilog ++ ve – operatörlerini desteklemez. C' deki i++ yazmak yerine, bunun tam işlemsel karşılığını i=i+1 yazmanız gerekmektedir. Örneğin;

```
For (i = 0; i<16; i= i+1) begin
        $display ("i' nin değeri %d", i);
End
```

2.3.3.4.5. Repeat

Repeat daha önce belirtildiği gibi for döngüsüne benzemektedir. Bunun dışında dışarıdan bir değişken belirtilir ve arttırılır, döngüyü bildirdiğimizde kaç kez kodun çalışacağını bildiririz. Örneğin:

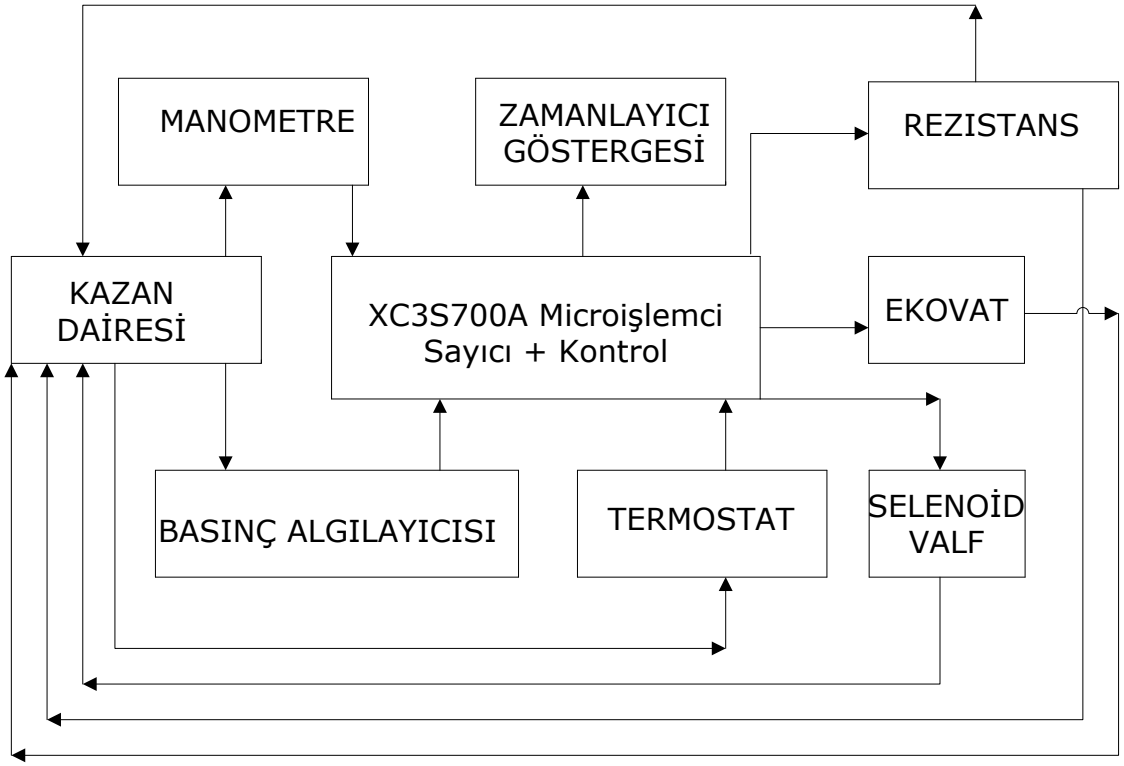
```
Repeat (16) begin
    $display (“İ’ nin değeri %d”, i);
    i= i+1;
end
```

3. TASARLANAN SİSTEMİN ÇALIŞMASI VE AKIŞ ŞEMASI

Bu bölümde tasarlanan sisteme ait blok şema ve sistemin çalışması anlatılmaktadır.

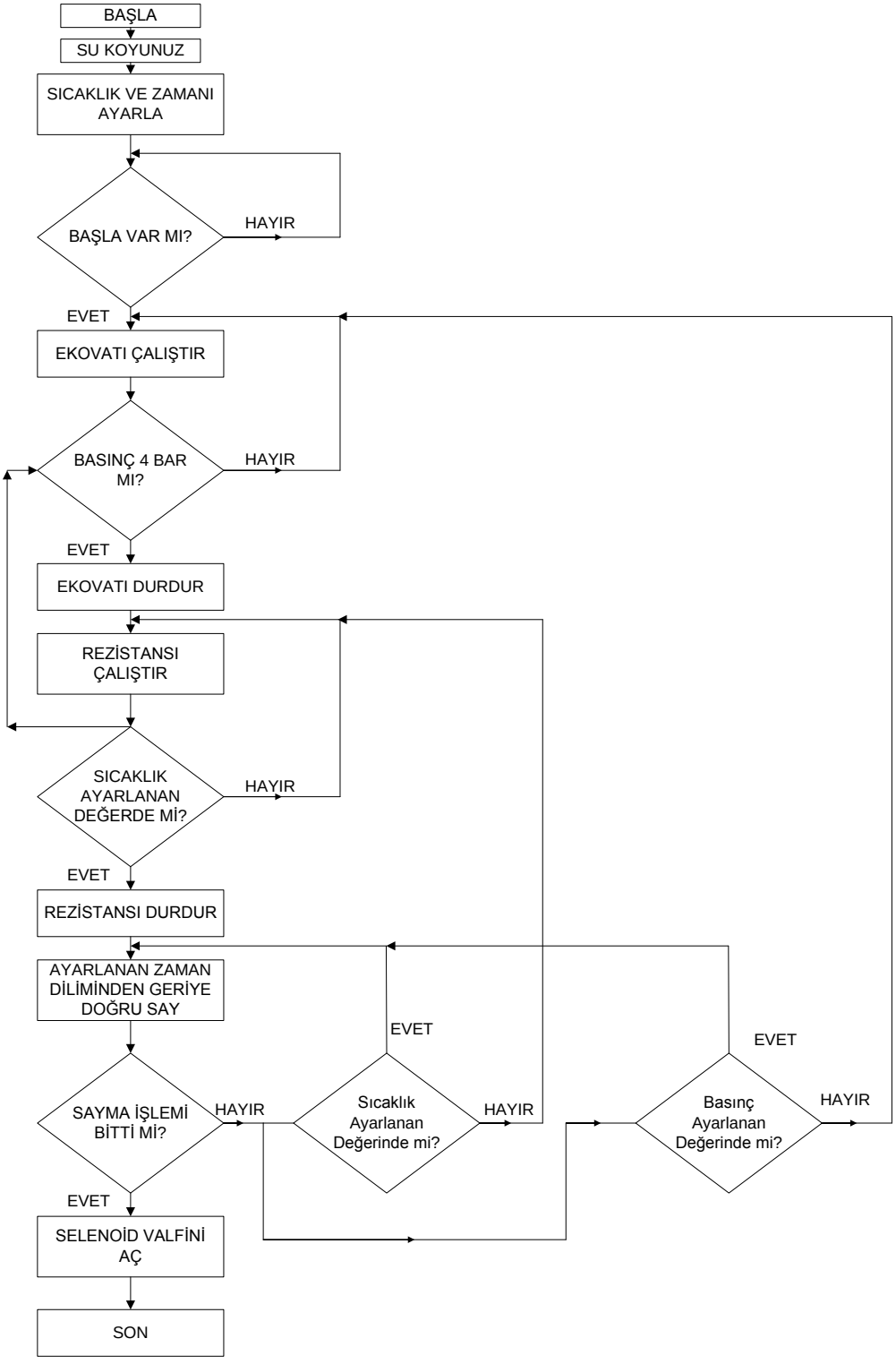
3.1. Tasarlanan Sistemin Blok Şeması

Tasarlanan sisteme ait blok diyagramı şekil 3.1.' de görülmektedir. Sistem tasarımında bir dizi önlemler alınmıştır. Bu önlemler; kazan dairesinin kapak kısmı üzerine bir muhafaza kapak yapılmıştır. Bu sayede meydana gelebilecek herhangi bir patlama ile dağılan parçacıklar ve sıcak suyun etrafa dağılması engellenmiş olmaktadır. Kazan dairesinin aşırı basınçtan dolayı patlamasını önlemek için üst basınç kontrolü yapılmıştır. Kazan dairesi içerisindeki basınç 6 bar' ı geçince selenoid valf açılıp basınç tahliye edilmektedir. Kontrol kart boyutları piyasadaki diğer kartlara göre %30 daha küçüktür. Kontrol kartta denetlenen elemanları sürmek için Triaclar kullanılmaktadır. Bu sistemlerde ilk kez FPGA kullanılmıştır.



Şekil 3.1 Tasarlanan ve gerçekleştirilen sistemin blok diyagramı

3.2. Sistem Yazılımının Akış Şeması ve Sistemin Çalışması



Şekil 3.2. Biodent fırın tasarımına ait akış şeması

Programın çalışması; ilkönce zaman seçim anahtarı yardımıyla hangi zamanın seçildiği, sıcaklık değeri ayarlanıp ayarlanmadığı program tarafından algılanır. Daha sonra program BAŞLAMA butonuna basılıp basılmadığını algılar. BAŞLAMA varsa ilkönce ekovat çalışır. Diğer taraftan ekovatın hava bastığı kazan içerisindeki basınç kontrol edilir. Basınç 4 bar' a gelince ekovat devre dışı bırakılır. Ekovatın durması program tarafından algılanınca rezistans devreye alınır. Rezistansın çalışması ayarlanan sıcaklığa ulaşınca durdurulur. Rezistansın durması program tarafından algılanır ve zamanın geriye doğru sayması başlatılır. Bu geri sayma işlemi devam ederken herhangi bir sebepten kazan içerisindeki sıcaklık, ayarlanan sıcaklığın altına düşerse rezistans tekrar devreye alınır, ayarlanan sıcaklık seviyesine gelince de devre dışı bırakılır; eğer kazan içerisindeki basınç ayarlanan basıncın altına düşerse ekovat tekrar devreye alınır, ayarlanan basınç seviyesine gelince de ekovat devre dışı bırakılır. Rezistansın çalışması esnasında oluşan buhardan dolayı bir basınç artışı meydana gelmektedir. Bu artan basınç 6 bar' ı geçince selenoid valf açılarak kazan dairesindeki basınç 6 bar' ın altına düşürülmektedir. Böylece kazan içerisindeki basınç 4 bar ile 6 bar arasında sabitlenmiş olmaktadır. Zaman sıfır olunca selenoid valfi açılır kazan içerisindeki hava ve pis su tahliye edilir. Böylece program tamamlanmış olur. AÇ / KAPA anahtarı kullanılarak program tekrar başa döndürülür. Programın akış şeması şekil 3.2.' de gösterilmiştir. mikroişlemcinin I/O bacakları lojik 1 (5 volt) veya lojik 0 (0,2 volt) yapılarak ekovat, rezistans ve selenoid valfi kontrol edilmiştir. Biodent fırının kontrolü için tasarlanan program Ek-3' de verilmiştir.

3.2.1. Tasarlanan sistem tarafından kontrol edilen yükler

Tasarlanan sistem tarafından kontrol edilen yükler; rezistans, basınç algılayıcısı, ekovat ve selenoid valf' tir.

3.2.1.1. Rezistans

Rezistans olarak telden sarılan, 100W gücünde bir ısıtıcı kullanılmıştır. Rezistans, su ısıtmada kullanılan tipte yapılmıştır. Sistemin ilk çalışma zamanlarında, rezistans bir

süre en yüksek güçte çalışmaktadır. Rezistansın enerjisi kesildikten bir süre sonra dahi su ve protezin bulunduğu kazana, ısı vermeye devam etmektedir [1]. Şekil 3.3.' de biodent fırında kullanılan rezistans görülmektedir.

Şekil 3.3. Biodent fırın rezistansı

3.2.1.2. Basınç algılayıcısı

Basınç algılayıcıları prostat olarakta bilinmektedirler. Temelde bağlandığı hattaki akışkanın basıncını izlemede kullanılırlar. Algılayıcı, istenen basınç değerine geldiğinde sinyal çıkışı verir ve boru hattındaki basıncı sürekli izlemesine yardımcı olur. Basınç algılayıcısının üç ucu bulunmaktadır. Bunlardan birtanesi ortak uç, diğeri normalde açık ve son uç ise normalde kapalı ucu vardır. Boru hattındaki basınç değeri ayarlanan seviyeye ulaşınca bu uçlarından kontrol kartına sinyal gönderir [1]. Sistemde kullanılan algılayıcı şekil 3.4.' de görülmektedir.

Şekil 3.4. Basınç algılayıcısı

3.2.1.3. Ekovat

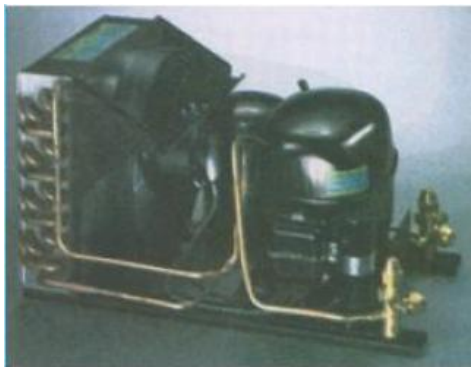
Ekovatı kısaca tanımlıyacak olursak; buzdolabı ve klimalarda kullanılan motor pompa gruplarına verilen addır. Ekovatlar soğutma sistemlerinde kullanılan kompresörlerdir. Ekovatların ilk modelleri tipik amonyak makineleriydi. O günlerde amonyak en çok tutulan soğutucu olduğu için kompresörler çok yüksek basınçları karşılayabilmek için çok ağır yapılırlardı ve günümüzde kullanılan kompresörlere göre çok daha yavaş çalışırlar. Selenoid valf tasarımı, kompresör mil contaları, yataklar ve yağlama sistemlerindeki ilerlemeler tasarım hızının kademeli olarak

artmasını sağlamıştır. Bu sayede daha hızlı çalışma ve daha çok yer değiştirme elde edilmiştir.

Ekovatlarda iki farklı uç bulunmaktadır. Bunlardan birisi dış ortamdan ekovat içerisine hava girişini sağlar diğeri ise dış ortama hava basar. İşte bu dış ortama hava basan kısmı biodent fırınının kazan kısmına bağlanmıştır. Böylece ekovat sayesinde kazan içerisine yüksek basınçlara kadar hava basıla bilmektedir. Ekovatlar kazan içerisine 20atm' lik hava basabilmektedir. Biodent fırınlarında maksimum 6atm' lik bir basınca ihtiyaç duyulduğundan ekovatlar ihtiyaçlarımızı karşılamaktadır. Tasarlanan sistemde kullanılan ekovat şekil 3.5.' de görülmektedir.

Eski sistemlerde ekovat yerine şekil 3.6.' da görülen büyük kompresörler kullanılmaktaydı. Bunların dezavantajı, fırınların bir yerden biryere taşınması zor, çalışma yerlerinde çok fazla yer kaplaması, çok gürültülü çalışması ve enerji sarfiyatının yüksek olmasıdır [23].

(a) Günümüzde Kullanılan ekovat



(b) Eski tip ekovat

Şekil 3.5. Ekovat çeşitleri

Şekil 3.6. Eski sistem biodent fırınında kullanılan kompresör

3.2.1.3.1. Ekovat Çeşitleri

Birçok çeşit ekovat bulunmaktadır fakat hepsinin yaptığı iş aynıdır. Sadece aralarında yapısal olarak bazı farklar vardır. Yapılarına göre ekovatları şu şekilde sınıflandırmak mümkündür [23];

- Pistonlu kompresörler
- Paletli dönel kompresörler
- Helisel- vida tipi dönel kompresörler
- Santrifüj kompresörler

3.2.1.3.2. Ekovatın Çalışması

Kompresör emme basma tulumbaya benzer. Piston aşağıya inerken sıvı giriş borusundan emilir. Piston yukarı çıkarken ise çıkış borusundan gönderilir. Bu tez çalışmasında kullanılan küçük boyutlu ekovat bir fazlı (220V' luk), yardımcı sargılı asenkron motor vardır. Bu tip motorların stator oyuklarında kalın kesitli, az sarımlı ana sargı, ince kesitli çok sarımlı yardımcı sargı vardır.

Motora enerji verildiğinde ilk önce sadece ana sargının üzerinden alternatif akım (AC) dolaşır. Sadece ana sargıdan geçen AC' nin oluşturduğu manyetik alan motorun rotorunun dönmesini sağlayamaz.

Rotor dönmeye başlayınca ana sargının çektiği akım daha yüksek bir değere çıkar. Bu esnada ana sargıya seri bağlı olan yol verme rölesinin bobininin oluşturduğu manyetik alan artar. Yol verme rölesinin akımının artması bu elemanın içinde bulunan nüvenin mıklatıslanma nedeniyle hareket etmesine yol açar. Hareket eden nüvenin ucuna bağlı olan kontaklar kapanarak yardımcı sargının da devreye girmesini sağlar. Yardımcı sargıdan geçen akımın oluşturduğu ikinci değişken manyetik alan, motorun rotorunun dönmesini sağlar.

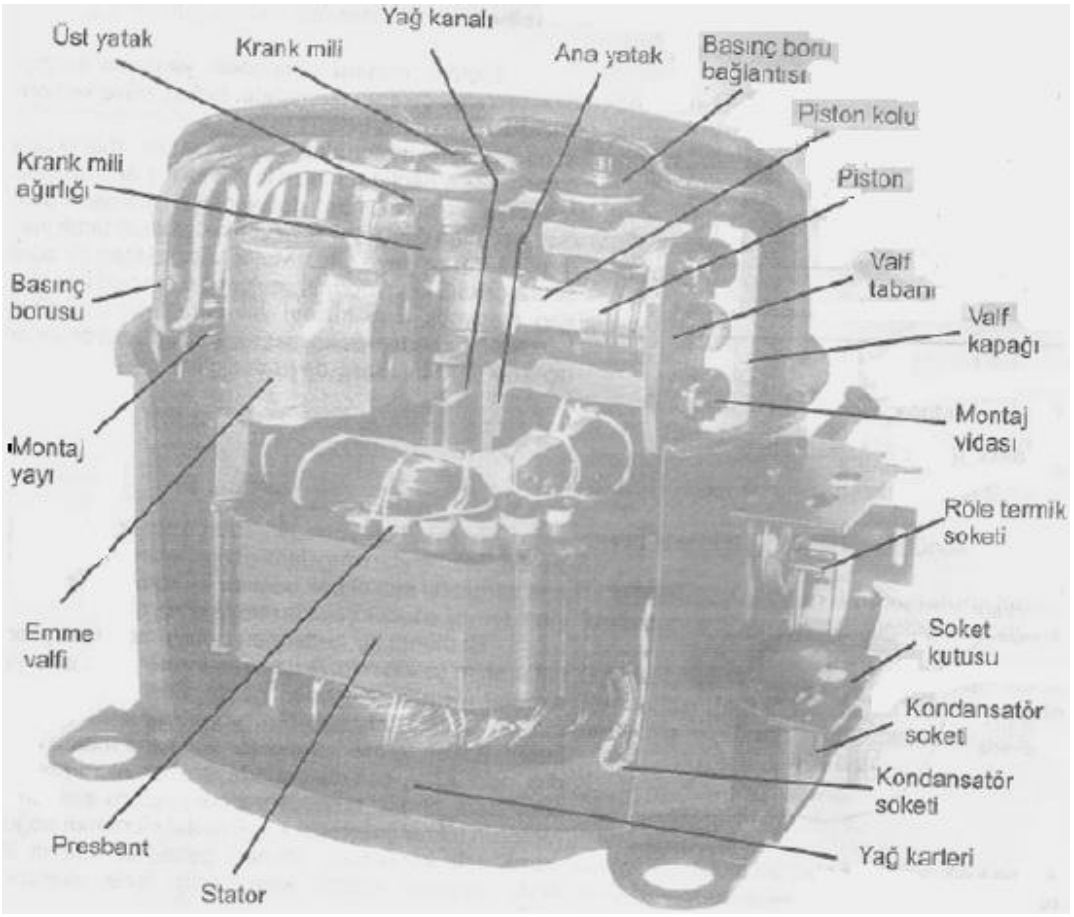
Rotor dönmeye başlayınca ana sargı üzerinden geçen akım düşer ana sargının akımının düşmesi yol verme rölesinin kontağını açmasını sağlar. Yol verme rölesinin kontağı açılınca yardımcı sargı devreden çıkar. Yardımcı sargı devreden çıkmasına rağmen motorun rotoru dönmeye devam eder. Yardımcı sargıya seri olarak bağlanmış olan kondansatör ana sargı ile yardımcı sargı arasında oluşan faz farkının 90^0 olmasını sağlayarak dönüşün başlamasını kolaylaştırır [23].

3.2.1.3.3. Ekovatın yapısı

Bir ekovat aşağıdaki elemanlardan oluşur;

- Kompresör tası,
- Elektrik motoru,
- Rotor mili,
- Krank,
- Biyel kolu,
- Piston silindir kolu,
- Emme ve basma valfleri,

Şekil 3.7.’ de tasarlanan sistemde kullanılan ekovatın iç yapısı görülmektedir.

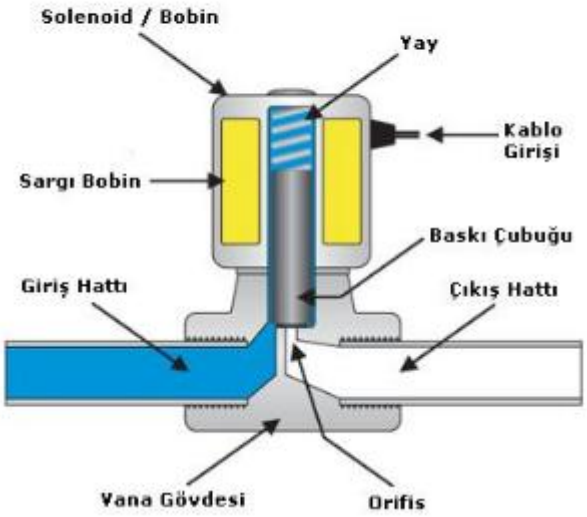


Şekil 3.7. Hermetik kompresörün iç yapısı [23]

3.2.1.4. Selenoid Valf

Selenoid valfler, sıvı ve gazların akışını kontrol eden elektromekanik bir vanadır. Selenoid üzerindeki sarmal tele akım vererek selenoid valfin konumunun değişmesini sağlar. Çalışma prensibi elektrik anahtarıyla benzerlik gösterir fakat burada hareket eden elektrik akımı değil sıvı ve gazdır. Selenoid valfler, akışkanlar üzerinde en çok kullanılan kontrol üniteleridir. Görevleri; kapatmak, açmak, dozlamak, dağıtmak ve akışkanları karıştırmak olarak özetlenebilir. Birçok alanda tercih edilmelerin nedenleri; hızlı ve tam komut almaları, yüksek güvenilirlik, uzun ömürlü, malzemelere uyum, düşük enerji kullanımı ve kopmak dizayn sahibi olmasıdır. Bu tez çalışmasında basit direk çekmeli normalde kapalı selenoid valf kullanılmıştır. Şekil 3.8.' de basit direk çekmeli normalde kapalı tip selenoid valfin

yapısı görülmektedir [1]. Şekil 3.9.' da basit direk çekmeli normalde kapalı tip selenoid valfi görülmektedir.



Şekil 3.8. Basit direk çekmeli normalde kapalı selenoid valfin yapısı

Şekil 3.9. Selenoid valf

4. SONUÇ VE ÖNERİLER

Bu çalışmada gömülü sistemlerle biodent fırın tasarımı ve gerçekleştirilmesi yapılmıştır. Sıcaklık kalibrasyonları, TSE' den kalibre edilmiş termokupl ile ölçümler alınarak yapılmıştır. Basınç kalibrasyonları ise TSE' den kalibre edilmiş manometre ile ölçümler alınarak yapılmıştır. Yapılan kalibrasyonun doğruluğu, sıcaklık ve basınç ölçümleriyle test edilmiştir.

Protezin pişirilmesinde dikkate edilmesi gereken değişken sıcaklıktır. Diş protezleri en fazla 120^{o}C' de pişirilmektedir [19]. Bundan dolayı toleranslarda göz önüne alınarak fırının kalibrasyonu 125^{o}C' ye kadar pişirme yapabilecek şekilde düzenlenmiştir. Hassas kalibrasyon yapabilmek için sıcaklık algılayıcıları termokupl olan dijital termometre kullanılmıştır. Sıcaklığın protezin pişirilme ısısının altında veya üstünde olması protezi, sert veya yumuşak yapabilir. Protezin çok sert olması kolay kırılmasına, yumuşak olması kullanışsız olmasına sebebiyet vermekte, zaman ve para kaybına sebep olmaması için de hassas ve güvenilir yapılması gerekir. Diğer sistemlerde, ekovatın, selenoid valfin ve diğer çevresel gürültülerin fazla olması ve fiziksel sebeplerden dolayı kazan içerisindeki sıcaklık tam ölçülemediği için rezistans ayarlanan sıcaklıkta kapatılamamaktadır. Bunun sonucunda protezler yanmakta, fark edilmediğinde çok daha büyük zararlar ortaya çıkmaktadır. Bu durum kullanıcıya hem zaman hem de para yönünden zarar vermektedir.

Biodent fırınlarda, kontrol edilen diğer değişken de zamandır. Protezin pişirilme süresi de önemlidir ve zamanlama hatalarının az olması gerekmektedir. Piyasadaki diğer fırınlarda, ucuz olması sebebiyle RC osilatör kullanılmaktadır. RC osilatörle tam zamanlama sağlanamamaktadır. Sıcaklık ve gürültülerden de etkilenmektedir. Mikro denetleyiciler ile oluşturulan sistemlere de ise kristal osilatörler kullanılmaktadır. Yalnız bu osilatörlerde kısa zaman içerisinde gerek fiziksel, gerekse gürültüden dolayı zarar görmektedirler ve tam bir zamanlama yapamamaktadırlar. Zamanlama tam yapılamadığından protezler kötü ve kullanışsız olmaktadır.

Diş protezleri 4 bar ile 6 bar basınç altında pişirilmektedir [19]. Kazan içerisindeki basıncı ölçen manometrenin kalibrasyonu yapılmıştır. Protez pişirilirken kazan içerisindeki basınç oldukça önemlidir. İlk basınç artışında, basıncın 4 bar' dan az olması protezin iç kısımlarında boşlukların kalmasına, 6 bar' dan fazla olması ise protez üzerinde kabarcıkların oluşmasına sebep olmaktadır. Kazan içerisinde sıcaklık artışı olduğunda buharlaşan sudan dolayı basınç artışı meydana gelmektedir. Bu artış diş protezine zarar vermemektedir; çünkü ilk basınç artışında diş protezi sertleşir ve alması gereken şekline alır, bundan sonra artan basınç zarar vermemektedir. Diğer sistemlerde en sık karşılaşılan ve hatta ölümlere sebebiyet veren sorunlardan bir tanesi basınç ile ilgili olanlarıdır. Gürültü ve fiziksel sebeplerden kazan içerisindeki basıncın algılanamaması veya kompresörün bağlı olduğu rölenin bozulmasından dolayı kompresör durdurulamamaktadır. Bunların haricinde, basıncın üst sınırı kontrol edilmediğinden rezistansın çalışması esnasında meydana gelen su buharından kaynaklanan basınç artışı devam etmektedir bu durum kontrol kartı tarafından algılanamadığından basıncın artışına müdahale edilememektedir. Bunlara bağlı olarak kazan içerisindeki basınç sürekli artmakta ve belirli bir süre sonra kapak kısmını patlatmaktadır. Bu patlama sonucunda etrafa fırlayan parçacıklar kullanıcılara zarar vermektedir. Bu durum kullanıcıların sağlığını tehdit etmektedir.

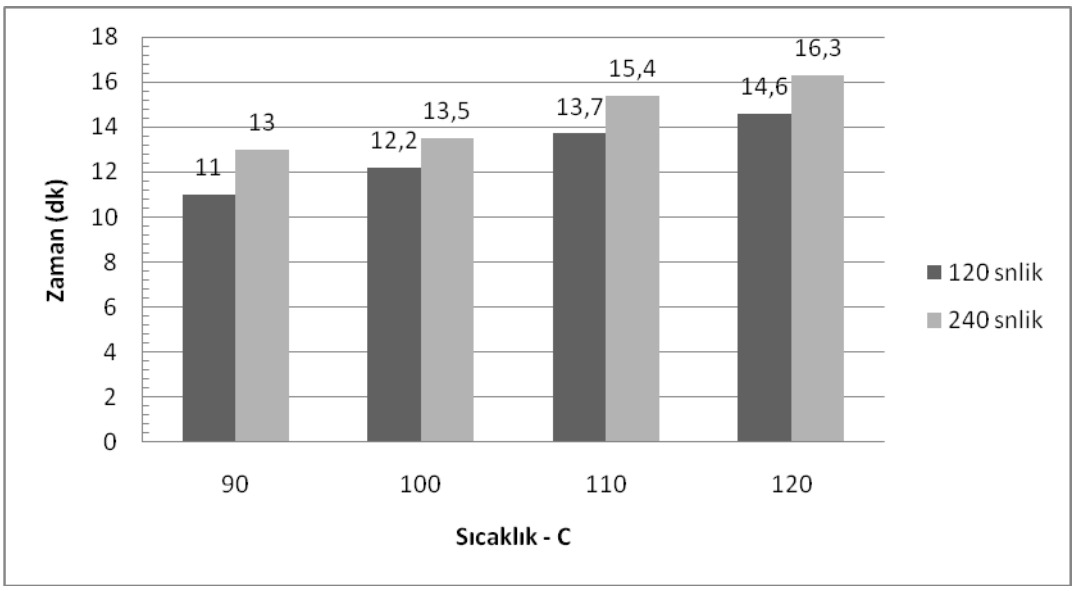
Gerçekleştirilen bu çalışmada, Yazılımın daha hızlı geliştirilmesi sonuçların daha çabuk görülmesi ve prototip geliştirmek amacıyla Xilinx spartan-3A deney seti kullanılmıştır. Kullanılan borda ait resimler Ek-5' de verilmiştir.

Bu çalışmadaki en önemli sonuçlar:

- Biodent fırınlarda kullanılan kontrol kartında ilk defa gömülü sistem olarak FPGA entegresi kullanılmıştır.
- Kontrol kartı benzerlerinden %30 daha küçüktür.
- FPGA' ların gürültülerden ve fiziksel ortamdan çok fazla etkilenmedikleri için kazan dairesi içerisindeki basınç, ayarlanan değere geldiğinde FPGA tarafından ekovat durdurulur. Bu açıdan da piyasada bulunan diğer kontrol sistemlerinden daha güvenilirdir.

- Devrede ikinci bir basınç algılayıcısı kullanılarak basıncın üst sınırı da kontrol altına alınmıştır. Su buharının meydana getirdiği basınç, 6 bar' ı geçtiğinde kazan içerisindeki hava ve su buharı selenoid valf tarafından tahliye edilerek basınç değerinin 4 bar ile 6 bar arasında kalması sağlanmıştır. Böylece kazan dairesinin aşırı basınçtan patlaması önlenmiştir.
- Rezistansın zamanında devreye alınıp ardından da devreden çıkartılmasıyla protezlerin sağlıklı pişmesi sağlanmıştır.
- Zamanlama hatalarını gidermek için sayıcı, mikroişlemci içerisinde kodlarla yapılmıştır.
- Saat osilatörü olarak mikroişlemcinin içinde bulunan 50MHz' lik osilatör kullanılmıştır.
- Mikroişlemcinin zarar görmemesi için ekovat, selonoid valf ve rezistansı çalıştıran triyaklar MOC3041 opto- izolatör kullanılarak sürülmüştür. MOC3041' lerde darlington bağlı transistörlerle sürülmüştür. Böylece kontrol kartının voltajı ile programlayıcının voltajı birbirinden yalıtılmış ve mikroişlemci korunmuştur.

Tasarımda kullanılan 120sn ve 240sn' lik programlar ile $90^{o}C$ - $120^{o}C$ sıcaklıklar arasında, pişirimleri ve bu pişirimlerin toplamda ne kadar sürede bitirildiği şekil 4.1.' de gösterilmiştir.



Şekil 4.1. 120sn ve 240sn' lik programların farklı sıcaklıklardaki pişirme işlemini bitirme süreleri

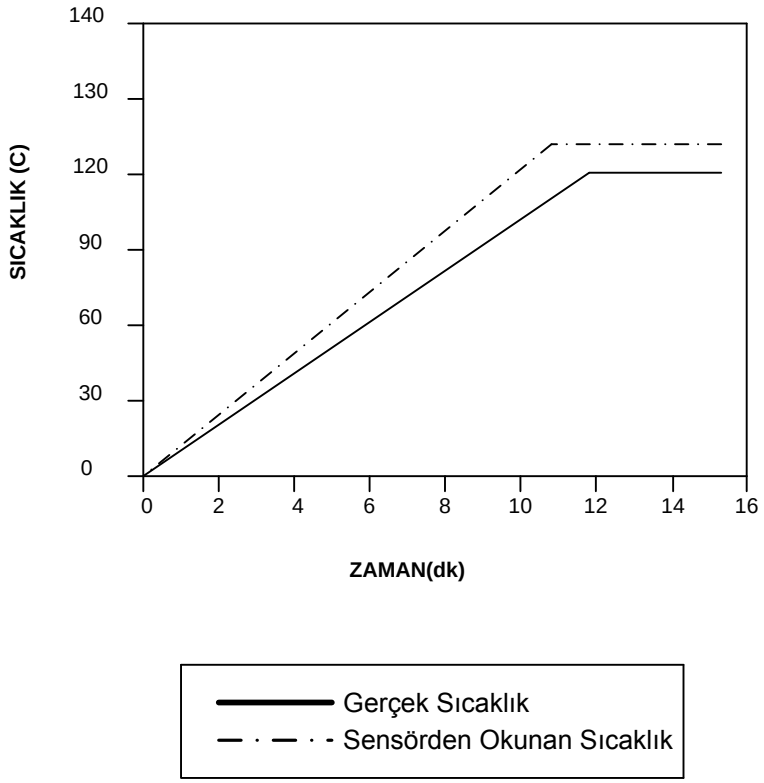
Çizelge 4.1.' de termokupl uçlarında ölçülen sıcaklık değerleri gösterilmiştir. Ayrıca bu ölçümlerle fırının olması istenilen sıcaklığı ile sıcaklık algılayıcısından ölçülen sıcaklığın standart sapması 30,01 olarak hesaplanmıştır. Bu değer bizlere biodent fırının gerçeğe ne kadar yakın pişirme işlemini gerçekleştirdiğini göstermektedir. Biodent fırında protezleri pişirme zamanı seçilen programa, kullanılan rezistansın gücüne ve ekovatın büyüklüğüne göre değişmektedir. Ölçülen sıcaklık ile gerçek sıcaklığın zamana göre değişim grafiği şekil 4.2.' de görülmektedir.

Çizelge 4.1. Termokupl ile alınan ölçümler

Gerçek Sıcaklık (^{o}C)	30	40	50	60	70	80	90	100	110
Ölçü Aletinden Okunan Sıcaklık	32	43	55	66	79	86	99	110	119
Basınç(bar) (Sabit)	4	4	4	4	4	4	4	4	4

Yukarıdaki çizelgeye ait standart sapma;

σ (Standart Sapma) = 30,01

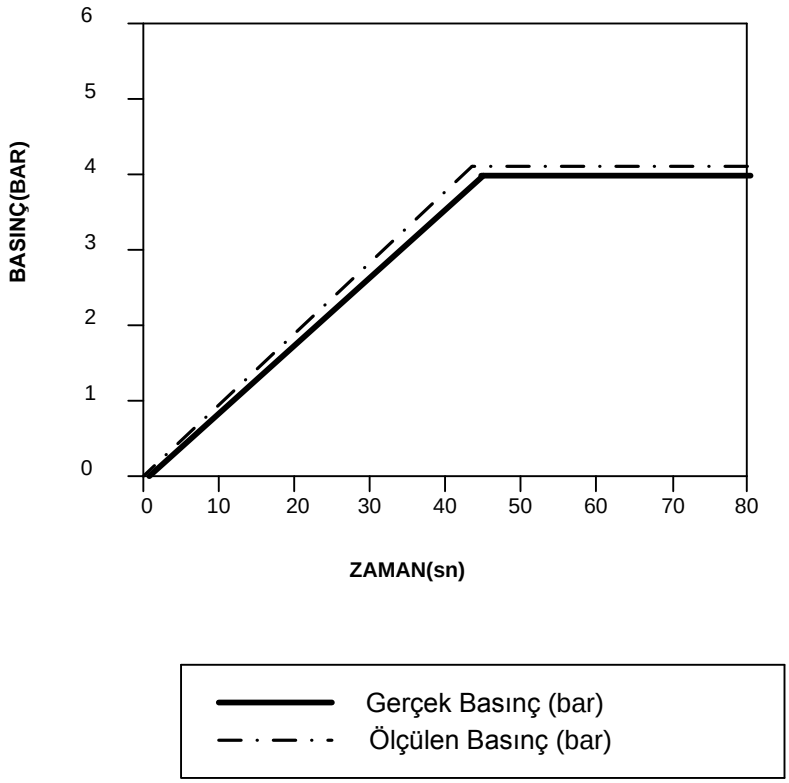


Şekil 4.2. Termokupldan ölçülen sıcaklık ile fırının gerçekte olması istenilen sıcaklığı

Çizelge 4.2.' te basıncın zamana göre değişim ölçümleri verilmiştir. Bu çizelgede basınç 4bar' a ayarlanmıştır. Şekil 4.3.' te basıncın 4bar' a 44sn de geldiği görülmüştür. Bu zaman kullanılan ekovatın gücüne bağlıdır. Daha güçlü bir ekovat kullanılırsa, süre 44sn, den çok daha kısa olacaktır; fakat bu süre 20 – 25 sn kadar azaltacaktır.

Çizelge 4.2. Basıncın zamana bağlı değişim ölçümleri

Zaman (sn)	12	23	33	44	55	65	70	80
Ölçülen Basınç (bar)	1	2	3	4	4	4	4	4
Kullanılan Ekovat: 220V 50Hz 3/8 134A 952K.CAL Test süresi 15 dk.								



Şekil 4.3. Basıncın zamana bağlı değişim grafiği

Bu çalışmada, yeni nesil Xilinx FPGA' ları kullanılabilirdi. Bu FPGA' lar daha hızlı çalışan, giriş/çıkış uçları fazla olan mikroişlemcilerdir.

Bioden fırınında gösterge olarak LCD gösterge kullanılabilirdi; fakat LCD göstergenin kullanımı hedeflenen maliyetleri çok aşmasına neden olmaktadır. Bu da piyasanın tasarlanan cihazın diğerleri ile rekabetini (tercihini) azaltacaktır. Ayrıca LCD gösterge kullanımında kod sayısı artacaktır. Bu da mikroişlemci içerisinde kapladığı yeri ve kullanılan mikroişlemcinin giriş/çıkış bacak sayısını arttıracaktır. Programın daha yavaş çalışmasına, dolayısıyla performansın azalmasına sebep olacaktır.

Kartta protezin pişirilme süreleri 120sn ve 240sn şeklinde sabit değil de elle girilerek süreleri ayarlanabilinecek şekilde yapılırsa, diş laboratuarları haricinde işitme cihazı imalatında ve buna benzer daha çeşitli yerlerde kullanılma imkanı sunulabilir.

Termostat olarak, yeni nesil sayısal termostatlar kullanılabilir. Bu, dış görünümü daha da güzelleştirir. Fakat; biodent fırınların kullanıldıkları yerde su kullanımı fazla olduğu için bu gibi sayısal termostatların bozulma ihtimalleri çok yüksektir. Ayrıca sayısal termostatlar analog termostatların fiyatlarının neredeyse 5 katı kadardır. Bu durum hem maliyeti arttırdığı hem de bozulma ihtimalini yükselttiği için bu çalışmada tercih edilmemiştir.

Ekovat yerine hızı kontrol edilebilir DC fırçasız motorlar kullanılabilir. Böyle bir motor kullanılırsa kartın boyutları ve maliyeti artar; fakat enerjiden tasarruf sağlanır.

Yukarıdaki önerilerin enerji tasarrufu sağlayacağı; fakat maliyeti ve kartın ebatlarını arttıracağı bilinmelidir. Ancak piyasada yapılan araştırmalarda müşterilerin ucuz, güvenilir ve sağlam fırın talep ettikleri gözlenmiştir. LCD gösterge, sayısal termostat ve hızı kontrol edilebilir motor daha teknolojik bir görünüm sağlayabilir; fakat toplamda maliyet neredeyse %30 artış göstereceğinden, müşterilerin bu fırına olan ilgisini azaltmaktadır.

KAYNAKLAR

1. Kapıdere, M., "Mikrodenetleyici Kontrollü Diş Protezi Pişirme Fırını", Yüksek Lisans Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 1-31 (1998).

2. Martinelli, N., "Dental Laboratory Technology", ***The C. V. Mosby Company***, Saint Lois, 172-173 (1975).

3. İnternet: Lider Diş A.Ş. "Teknik servis yetkilisiyle yapılan görüşmeler" www.liderdis.com.tr (2009).

4. İnternet: Xilinx Company "Spartan-3A FPGA family datasheet" http://www.xilinx.com (2010).

5. İnternet: Fairchild "MOC3041M" http://www.datasheetcatalog.org/datasheet/fairchild/MOC3041M.pdf (2010).

6. Garland, H., "Introduction Microprocessor System Design", ***McGraw Hill Press,*** London, 27-31 (1979).

7. İnternet: Texas Instruments "Data book" http://www.ti.com (1985).

8. Kıymaz, S., "CPLD Deney Seti Tasarımı", Yüksek Lisans Tezi, ***Gazi üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 21-27 (2005).

9. Chang, M., "Teaching Top-down Design Using VHDL and CPLD", ***Frontiers in Education Conference FIE '96***, Utah, 514-517 (1996).

10. Wunnava, S. V., "Correlating Software Modelling and Hardware Responses for VHDL and Varilog Based Designs", ***Southeastcon '99. Proceedings. IEEE***, Lexington, 122-125 (1999).

11. Pedroni, V. A., "Code Structure, Data Types",Circuit Design With VHDL, ***MIT Press Cambridge***, London, 13-45 (2004).

12. Çetin, Ö., Özcerit, A., Çakıroğlu, M., "VHDL Cookbook", ***IEEE Transactions on VLSI Systems,*** 82(1): 26-30 (2000).

13. Kıymaz, S., Canal, M. R., "Sayısal Elektronik Devrelerin CPLD Tabanlı Uygulaması", ***5. Uluslar arası İleri Teknolojileri Sempozyumu***, Karabük, 42-47 (2009).

14. Lee, W. F., "VHDL Coding", VHDL Coding and Logic Synthesis with Synopsys, ***Academic Press A Harcourt Science and Technology Company***, San Diego, 3-17 (2000).

15. Asbenden, P. J., "Packages and Use Clauses", The Designer' s Guide to VHDL 2nd ed., ***Academic Press A Harcourt Science and Technology Company***, San Diego, 231-256 (2002).

16. Hwang, E. O., " Digital Logic and Microprocessor Design with VHDL", ***A Division of Thomson Canada Press***, Canada, 119-122 (2006).

17. Bhasker, J., "A tutorial", A VHDL Primer 3th ed., ***Prentice Hall***, New Jersey, 9-27 (1999).

18. Yıldız., O., "Biodent Fırın Kalibratörleri. Kalibrasyon No:253-03-05-08-25-08-1", Lider Diş A.Ş., Ankara, 55-60 (2010).

19. Çalıkkocaoğlu, S., "Tam Protezler", ***İstanbul Üniversitesi Diş Hekimliği Fakültesi Dergisi,*** 81(1): 162-164 (1993).

20. Aydın, A., "FPGA Yonga Mimarisi ve Kullanımı", Lisans Tezi, ***Süleyman Demirel Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümü***, Isparta, 9-21 (2005).

21. Tabaru, İ. E., Bicakcı, S., Karaboğa, N., "APKD' lerinde Fır Süzgeç Tasarlamak için Geliştirilen VHDL kodu ve Şematiği", ***Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi***, 40(1): 1-4 (2006).

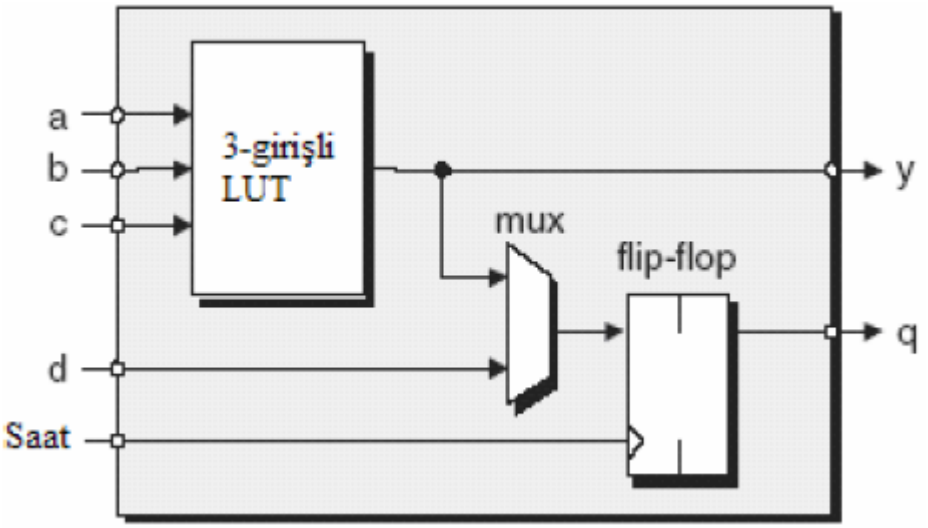
22. İnternet: Xilinx company "Spartan-3A FPGA Starter Kit Board: User Guide" www.xilinx.com (2010).

23. İnternet: MEGEP "Soğutuculardа ekovat bakım onarımı" ***http://cygm.meb.gov.tr/modulerprogramlar/kursprogramlari/elektrik/moduller/sogutuculardaekovatbakimonarimi.pdf***, (2010).

EKLER

EK-1. Alan programlanabilir kapı dizisi

FPGA' lar yapılandırılabilir mantık blokları ile birlikte bu bloklar arasındaki değiştirilebilir ara bağlantılardan oluşan sayısal tümleşik devrelerdir. FPGA' lar bağımsız olarak oluşturulabilen mantıksal kapılar, kaydediciler ve bunları birbirine bağlayan programlanabilir bağlantılardan oluşmaktadır. İlk FPGA' lar CMOS tabanlı ve yapılandırma için SRAM hücreleri kullanıyordu. İlk örnekleri çok daha basit olmalarına rağmen temelde var olan mimari çoğu açıdan hala kullanılmaktadır. İlk yongalar 3-girişli LUT, yazmaç ve MUX' tan oluşan programlanır mantık öbeklerine dayanıyordu.

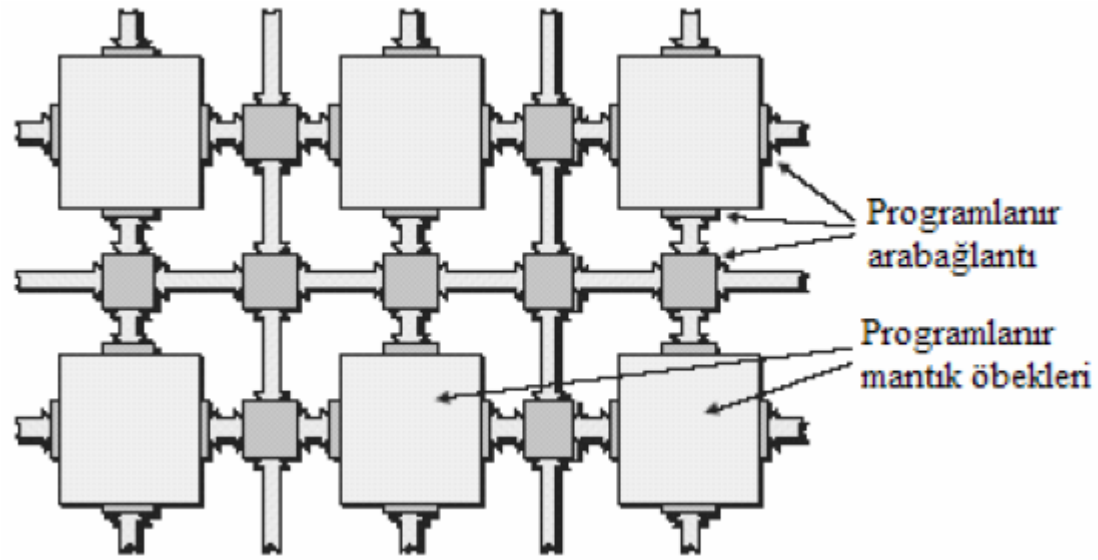


Şekil Ek-1.1. Programlanır mantık öbeğini biçimlendiren anahtar öğeler

SRAM hücreleri vasıtasıyla yongadaki her mantık öbeği farklı işlevler için yapılandırılabilir. Her yazmaç mantıksal 1 veya mantıksal 0 yapılarak başlangıç durumuna getirilmeli ve iki durumlu veya mandal gibi davranması için ayarlanmalı. İki durumlu seçilmişse, yazmaç saat darbesinin düşen veya yükselen kenarı ile tetiklenmesi için yapılandırılmalı. İki durumluyu besleyen çoğullayıcı (MUX), LUT çıkışı veya ayrı giriş için yapılandırılmalı. LUT da herhangi 3 girişli mantık işlevi için ayarlanmalı.

EK-1. (Devam) Alan programlanabilir kapı dizisi

Aşağıdaki basitleştirilmiş şekle ek olarak: yerel bağlantıları es geçecek yüksek hızlı genel ara bağlantı yolları, başlıca I/O bacakları vardır.



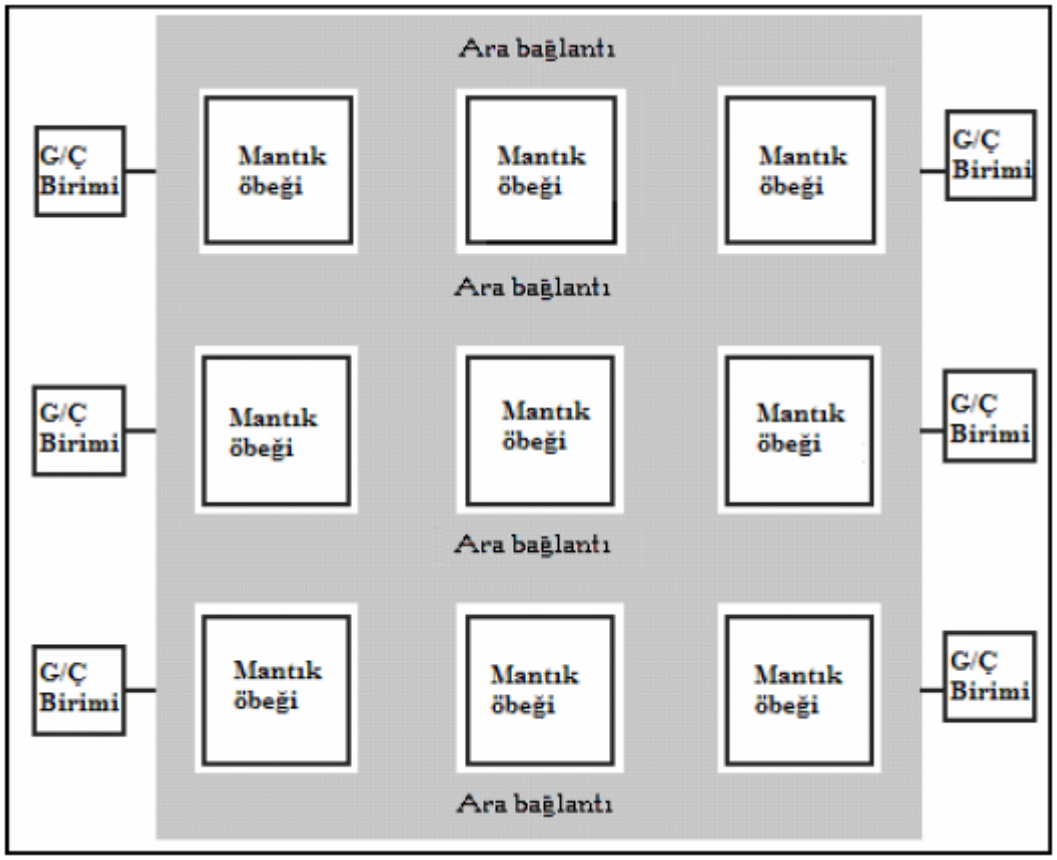
Şekil Ek-1.2. Genel FPGA mimarisi üstten görünümü

FPGA mimarisi

Genel anlamıyla FPGA' lar üç ana birime gereksinim duyarlar;

- Birleşimsel Mantık
- Ara Bağlantı
- I/O Bacakları

EK-1. (Devam) Alan programlanabilir kapı dizisi



Şekil Ek-1.3. FPGA yongasının genel görünümü

Üretim teknikleri

FPGA farklı yongaları, farklı teknolojileri bazı alarak üretilebilir. Aşağıda bu teknolojilerden birkaç tanesi kısaca görülmektedir.

SRAM temelli aygıtlar

FPGA' larda çoğunlukla SRAM temelli yapılandırma hücreleri yaklaşımı kullanılmaktadır. Bu tekniğin ana üstünlüğü yeni tasarım fikirlerin çabucak geliştirilebilir ve sınanabilir olmasıdır.

EK-1. (Devam) Alan programlanabilir kapı dizisi

Diğer bir üstünlüğü ise, teknolojinin ön saflarda yer almasıdır. FPGA sağlayıcılarının talepleri neticesinde bellekler üzerinde uzmanlaşmış firmalar bu alanda araştırma ve geliştirme için büyük kaynak harcamaktalar. Üstelik SRAM hücreler yongadaki diğer birimlerle tamamen aynı CMOS teknolojisi ile üretildiklerinden özel işlem basamağı gerektirmezler.

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

Antifuse temelli aygıtlar

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

Bir diğer önemli üstünlüğü ara bağlantı yapısının doğal olarak ışınım etkisine görece bağışıklık olmasıdır. Bu durum askeri ve uzay uygulamalarında özel ilgi sebebidir. Çünkü SRAM temelli bileşenler ışınıma maruz kalacak olurlarsa hatalara yol açabilirler. Antifuse bir kez programlandıktan sonra bu yolla değiştirilemez.

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

EK-1. (Devam) Alan programlanabilir kapı dizisi

Programlama verisinin aygıttan okunmasını engellemek için özel güvenlik antifuse öğesi ayarlamak mümkündür.

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

Antifuse tekniği ana üretim sürecine ek birkaç basamak daha gerektirir. Antifuse teknolojisi SRAM temelli yaklaşımın birkaç nesil gerisindedir. Bu güç tüketimi ve hız gibi üstünlüklerini bertaraf eder. Temel kusuru bir kez programlanmasıdır. Bu nedenle uygulama geliştirme için kullanılması doğru değildir.

EEPROM/ FLASH temelli aygıtlar

SRAM karşıtlarına benzer olarak yapılandırma hücreleri birbirlerine uzun ötelemeli yazmaç biçimli zincirler şeklinde bağlanmışlardır. Bu yongalar aygıt programlayıcılarıyla devre dışında programlanabilirler. Bazı çeşitleri devre içerisinde de programlanabilirler. Programlanma hızları SRAM' lere göre üç kat daha fazladır.

Programlandıktan sonra içeriği uçucu olmadığından sisteme ilk güç verildiğinde hazır olacaklardır. Yongayı programlarken kullanıcı tanımlı bit dizisi veri güvenliği amacıyla yüklenir. Aygıttan veriyi okumak veya yeni yapılandırmayı yüklemek için anahtarın kopyası JTAG bağlantı kablosu ile aktarılmalıdır.

SRAM hücrelerine göre çok daha küçük olduklarından ara bağlantı gecikmeleri daha azdır. Ancak standart CMOS teknolojisine ilaveten yaklaşık beş işlem basamağı gerektirdiğinden SRAM temelli aygıtların birkaç nesil gerisinde kalır. Ayrıca çok sayıda pull-up dirençleri içerdiklerinden dolayı çok fazla güç tüketimi yaparlar.

EK-1. (Devam) Alan programlanabilir kapı dizisi

Karma FLASH-SRAM aygıtları

Bu teknolojinin yapılandırma öğesi FLASH (veya EEPROM) hücre ve SRAM hücre biçiminden oluşur. Kullanımı çok sınırlıdır.

FLASH hücre önceden programlanabilir. Sonrasında sisteme güç verildiğinde FLASH içeriği ilişkili SRAM hücresine kopyalanır. Bu teknik antifuse teknolojisindeki kalıcılığı verir. Antifuse teknolojisinden farklı olarak sistemde yerleşik iken SRAM hücreleri tekrar yapılandırma için sonradan kullanılabilir.

Çizelge Ek-1.1. FPGA üretim tekniklerinin karşılaştırılması[4]

ÖZELLİK	**SRAM**	**ANTİFUSE**	**EEPROM/FLASH**
Teknoloji Noktası	En Gelişkin (State of the Art)	Birkaç Nesil Geride	Birkaç Nesil Geride
Tekrar Programlanırlık	EVET (Sistem İçerisinde)	HAYIR	EVET (Sistemde ve Devre Dışı)
Tekrar Programlama Hızı	HIZLI	****	SDRAM' den 3 kat yavaş
Uçuculuk (başta programlanmalı)	EVET (Sistem İçerisinde)	HAYIR	HAYIR (Gerekirse Yapılabilir)
Harici Program verisi gereksinimi	EVET (Sistem İçerisinde)	HAYIR	HAYIR
İlk örnek geliştirme	EVET (Çok iyi)	HAYIR	EVET (Kabul edilebilir)
Başlangışta Çalışırlık	HAYIR	EVET	EVET
IP Güvenliği	Kabul Edilebilir	Çok İyi	Çok İyi
Hücre Genişliği	Geniş (6 Transistör)	Çok Küçük	Orta (2 Transistör)
Güç Tüketimi	Orta	Düşük	Orta
Işınım Dayanırlığı	HAYIR	EVET	HAYIR

EK-1. (Devam) Alan programlanabilir kapı dizisi

Birim hücre mimarileri

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

İnce taneli (fine- grained) mimaride, her mantık öbeği sadece basit işlevler gerçekleştirmek için kullanılır. Örneğin bu öğeler ilkel mantık kapıları (VE, VEYA, vb.) gibi veya depolama öğesi gibi (D tipi flip-filop, D tipi latch, vb.) herhangi üç girişli işlev için yapılandırılabilir.

Birleştirici mantık (glue logic) ve durum makineleri gibi düzensiz yapılara ilaveten ince taneli mimari, pararel uygulamalardan faydalanan sistolik algoritmalar yürütürken özellikle verimli olurlar.

İri taneli (coarse- grained) mimaride, her mantık öbeği ince taneli yaklaşıma göre daha fazla mantık birimi içerir. Örneğin, bir mantık öbeği dört tane 4 girişli LUT, 4 tane MUX, 4 tane D tipi flip- filop ve bazı hızlı taşıma mantıkları içerebilir.

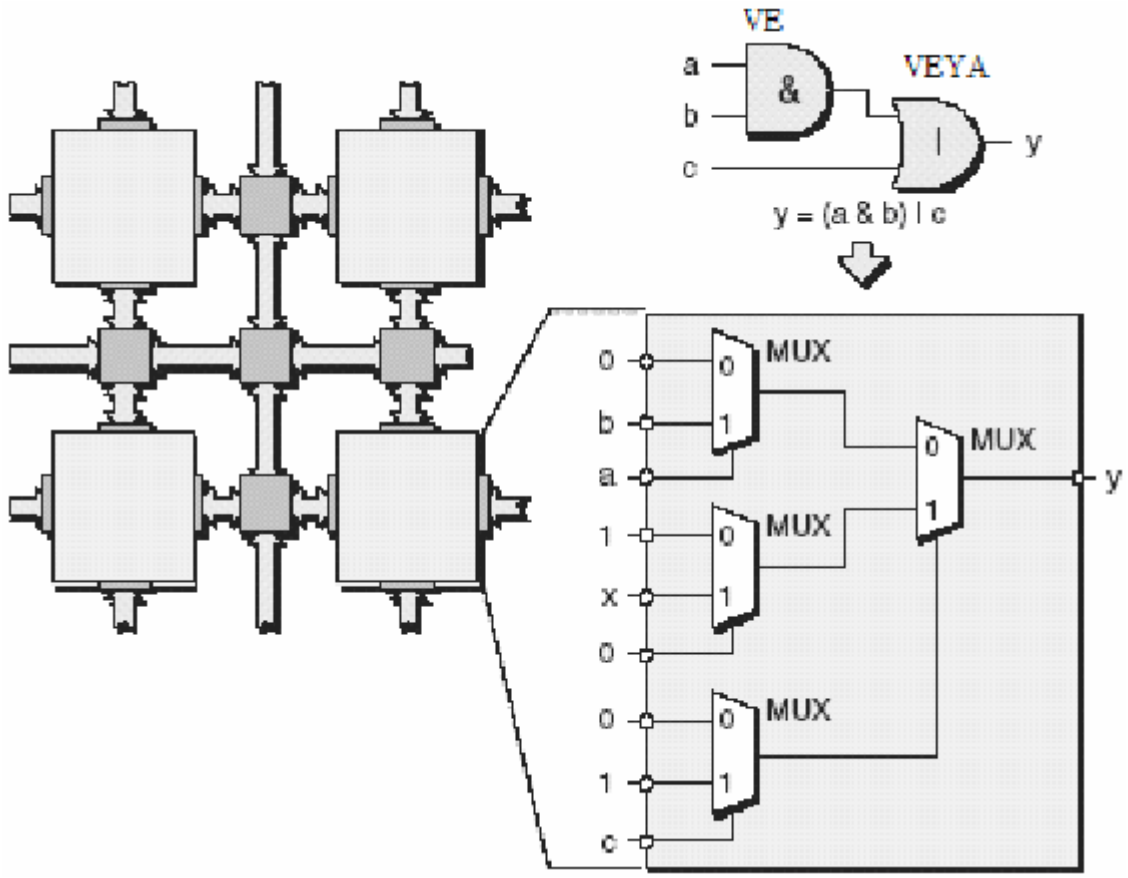
İki temel programlanır mantık öbeği düzenlemesi vardır; MUX ve LUT tabanlı.

MUX tabanlı programlanır mantık öbeği

Çoklayıcılı yaklaşıma örnek olarak 3 girişli y= (a&b) | c işlevini MUX içeren öbek ile gerçekleştirilmesini inceleyelim.

Aygıt mantıksal 0, mantıksal 1 veya diğer öbek ve asıl girişten gelebilecek sinyallerin asıl veya tümleyen değeri ile programlanabilir. Bu durum olası işlev için sayısız yol demektir. Aşağıdaki örnekte ortadaki MUX' un girişlerindeki x ifadesi, girişin mantıksal 1 veya mantıksal 0 olmasının işlemi değiştirmeyeceğini belirtir.

EK-1. (Devam) Alan programlanabilir kapı dizisi



Şekil Ek-1.4. MUX tabanlı mantık bloğu

LUT tabanlı programlanır mantık öbeği

Bu grup giriş sinyali başvuru çizelgesi için işaretçi olarak kullanılır. Çizelgenin içeriği, istenen değeri içeren her giriş birleşimini gösteren hücreler ile düzenlenir. Bu işlev 3-girişli LUT' u uygun değerler ile yükleyerek yapılabilir.

N- girişli LUT ile n- girişli herhangi birleşimsel işlev gerçeklenebilir. Daha fazla kapı eklemek daha karmaşık işlevlere müsaade eder ama her giriş eklentisinde SRAM hücre sayısı ikiye katlanır. Hücre sayısı 2^n olarak ifade edilir.

EK-1. (Devam) Alan programlanabilir kapı dizisi

Birkaç kademe derinliğindeki mantık kapıları için LUT yaklaşımı kaynak kullanımı ve giriş- çıkış gecikmeleri açısından daha verimli olacaktır. Günümüzde FPGA mimarisi çoğunlukla LUT tabanlıdır.

Kısaca hatırlatmak gerekirse; LUT yapıları mantıksal bloklar içerisinde bulunmaktadır. LUT, temel olarak hafıza birimleri ve bu hafıza birimlerinden hangisinin çıkışa aktarılacağını belirleyen bir çoğullayıcı yapıdan meydana gelmektedir [4,6,23].

EK-2. Xilinx spartan 3a ailesi

Spartan 3A ailesi, çok fazla I/O elektronik uygulamaları olan, çok hassas, çok yüksek miktardaki değişik tasarımlara çözüm üreten bir FPGA' dır. Bu aileye mensup beş adet entegre bulunmaktadır. Bu entegreler 50.000' den 1.4 milyon' a kadar sistem kapısı içermektedir. Aşağıdaki tabloda bu beş entegre ile ilgili özellikler görülmektedir.

Tablo Ek-2.1. Spartan 3A FPGA' sının özellikleri[4]

FPGA Entegresi	*Sistem Kapısı*	*Eşdeğer Sayısal*	*Ayarlanabilir Sayısal Bloklar (CLB) Dizisi (Bir CLB = Dört Dilim)*			
		Hücreler	*Satırlar*	*Kolonlar*	*CLBs*	*Dilimler*
XC3S50A	50K	1,584	16	12	176	704
XC3S200A	200K	4,032	32	16	448	1,792
XC3S400A	400K	8,064	40	24	896	3,584
XC3S700A	700K	13,248	48	32	1,472	5,888
XC3S1400A	1400K	25,344	72	40	2,816	11,264

FPGA Entegresi	*Dağıtılmış RAM Bitleri*	*Blok RAM Bitleri*	*Özel Çoklayıcılar*	*Dijital Saat Yönetimi (DCMs)*	*Kullanılabilir Maksimum Giriş/Çıkış*	*Maksimum Farklılaşmış Giriş/Çıkış Çiftleri*
XC3S50A	11K	54K	3	2	144	64
XC3S200A	28K	288K	16	4	248	112
XC3S400A	56K	360K	20	4	311	142
XC3S700A	92K	360K	20	8	372	165
XC3S1400A	176K	576K	32	8	502	227

Özellikleri;

1. Çok ucuz, yüksek kapasiteli devre tasarımları için yüksek performanslı sayısal çözüm, uygulamalarda maliyete karşı duyarlı.
2. Yalnızca 3.3V ile çalışabilmektedir.
3. Askıya alma, hazırda bekletme modu sayesinde sistem enerjisini düşürmektedir.

EK-2. (Devam) Xilinx spartan 3a ailesi

4. Çoklu voltaj, çoklu standart ara yüz I/O seçimi;
    - 502 I/O pinleri veya 227 farklı sinyal çiftlerine sahip,
    - LVCMOS, LVTTL, HSTL ve SSTL tek taraflı I/O' lar,
    - 3.3V, 2.5V, 1.8V, 1.5V ve 1.2V sinyalleri,
    - Çıkış pinleri seçilebilir ve her pin 24mA akım vermektedir,
    - I/O anahtar gürültülerini yok eden QUIETIO standartına sahip,
    - Tam 3.3V ± 10% uyumluluk ve hot swap uyumluluğu,
    - Farklı I/O' lar arası veri transfer oranı 640 + Mb/s,
    - LVDS, RSDS, mini-LVDS, HSTL/SSTL farklı I/O' lar ile birleştirilmiş farklı sınır dirençlerine sahip,
    - Geliştirilmiş çift veri oranı desteği (DDR- Double Data Rate),
    - 400 Mb/s' e kadar DDR/DDR2, SDRAM desteği,
    - Tamamiyle 32-/64-bit ve 33/66 MHZ PCI (Peripheral Component Interconnect) teknolojisi ile uyumlu,
5. Çok fazla, esnek sayısal kaynaklar;
    - 25' den 344' e kadar sayısal hücreler, opsiyonel shift register veya dağıtılmış RAM desteği,
    - Verimli geniş multiplexerlar, geniş mantık,
    - Göz önünde hızlı mantık yürütme,
    - İsteğe bağlığı gizli veri hattı ile geliştirilmiş 18x18 çoğaltıcılar,
    - IEEE 1149.1/1532 standartında JTAG programlama portuna sahip,
6. Hiyerarşik SelectRAM bellek mimarisi;
    - 576 Kilobit' e kadar hızlı blok RAM' ler ile işlemci uygulamaları için byte yazma olanağı sağlar,
    - Verimli dağıtılmış RAM için 176Kbit' e kadar destek,
7. 8 Adet dijital saat yönetimi (DCM);
    - Saat çarpık eleme (Döngü kilitli gecikme),
    - Frekans sentezi, çarpma, bölme,
    - Yüksek çözünürlüklü faz kaydırması,

EK-2. (Devam) Xilinx spartan 3a ailesi

- Geniş frekans oranı (5MHZ ile 320MHz)

8. Endüstri standartı PROM' lar için yapılandırma ara yüzü;
   - Düşük maliyetli, yerden tasarruf sağlayan SPI seri Flash PROM,
   - Flash PROM NOR x8 veya x8/x16 BPI pararel,
   - Düşük maliyetli Xilinx Platform Flash ile JTAG,
   - Tasarım doğrulama için benzersiz aygıt DNA tanımlayıcısı,
   - FPGA kontrolü altında yük çoklu veri akışı,
   - Post yapılandırma CRC (Cyclic Redundancy Check- Döngüsel artıklı denetim) kontrol,
9. Tamamıyla Xilinx ISEWebPack geliştirilmiş sistem yazılım destekli,
10. MicroBlaze ve PicoBlaze gömülü soft işlemcilere sahip,
11. Düşük maliyetli QFP ve BGA kılıflar, Pb serbest ayarlar [4,23].

XİLİNX SPARTAN 3A AİLESİ MİMARİ YAPISI

Spartan 3A ailesi mimarisi 5 temel programlanabilir fonksiyon elementini içerir;

- **Yapılandırılabilir Mantık Blokları (Configurable Logic Blocks- CLB):** Flip- filopları veya latch' ları kullanarak mantık depo elementleri ile gerçekleştirilen biçimlendirilebilir LUT' ları içerir. CLB' ler Veri depolarının yanısıra geniş çeşitli mantık fonksiyonlar şeklinde ifade edilirler.

- **Giriş Çıkış Blokları (IOBs):** Giriş çıkış blokları, veri akışını I/O pinleri ve dâhili mantık elemanları arasında kontrol eder. Giriş çıkış blokları veri akışını çift yönlü olarak sağlarlar. Birçok değişik sinyal standartlarını destekler, birkaç tane farklı yüksek performanslı farklı standartları da içerir. DDR registerlerini de içerir.

EK-2. (Devam) Xilinx spartan 3a ailesi

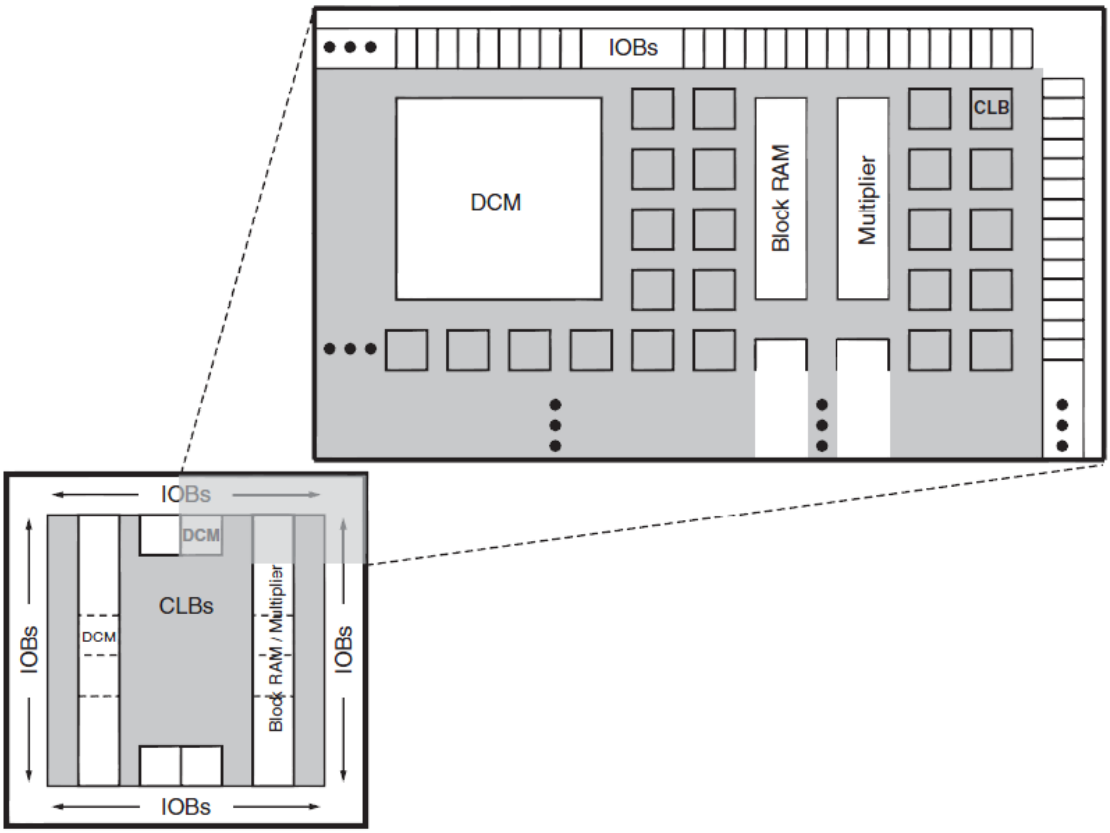
- **Blok RAM:** Blok Ram' ler 18Kbit çift port blokları şeklinde veri depolama sağlar.

- **Çoğaltıcı Bloklar:** Çoğaltıcı bloklar girdi olarak iki adet 18 bit' lik ikili sayıyı kabul eder ve ürünü hesaplar.

- **Sayısal Saat Yönetim (Digital Clock Manager- DCM) Blokları:** Sayısal saat yönetim blokları kendi kendini kalibre eder, dağıtım için tam sayısal çözümler üretir, gecikme sağlar, çarpar, böler ve faz saat sinyalleri kayması sağlar.

Burada bahsedilen programlanabilir fonksiyon elementleri aşağıdaki şekilde gösterilmektedir. Her bir eleman iki kolon blok RAM' e sahiptir. Her bir RAM kolonu birkaç tane 18 Kbit' lik RAM blokları içerir. Her bir blok RAM özel çoklayıcılar ile birleştirilmiştir. DCM' lerin ikisi merkezde üst kısımında konumlandırılmış, ikisi de alt kısımda konumlandırılmıştır. XC3S50A' nın DCM' leri sadece üstte, XC3S700A ve XC3S1400A' larda iki kolon olan blok RAM' lar ve çoklayıcıların tam merkezine fazladan iki adet DCM yerleştirilmiştir.

EK-2. (Devam) Xilinx spartan 3a ailesi



Şekil Ek-2.1. Spartan 3A FPGA' sının Mimari Yapısı

XİLİNX SPARTAN 3A AİLESİNİN KONFİGÜRASYONU

Spartan 3A FPGA' sı içerisine yapılandırma verileri yükleyerek daha sağlam programlanabilirler, tekrar programlanabilirler. FPGA' ların yapılandırma verileri harici bir PROM veya silinmeyecek bazı ortamlarda saklanmalıdır. Enerji uygulandıktan sonra, yapılandırma verileri FPGA yedi farklı mod kullanarak yazılır;

1. Master Serial' den Xilinx Platform Flash PROM,
2. Seri çevresel arayüzden (Serial Peripheral Interface- SPI)' den Endüstriyel standart SPI,

EK-2. (Devam) Xilinx spartan 3a ailesi

3. Byte çevresel arayüz (Byte Peripheral Interface- BPI)' den endüstriyel standart x8 veya x8/x16 pararel NOR flash,
4. Slave serial den mikroişlemciye yükleme,
5. Slave paralelden mikroişlemciye yükleme,
6. JTAG kablo ile standart yükleme,

XİLİNX SPARTAN 3A AİLESİNİN I/O KAPASİTESİ

Spartan 3A FPGA' sı farklı standartları ve yaygın tek taraflı seçilebilir I/O' ları destekleyebilir. Tablo Ek1.1.' de her bir eleman kombinasyonu için kullanılan I/O ları ve farklı I/O çiftleri görülmektedir.

Spartan 3A FPGA' sının desteklediği tek taraflı standartlar;

- 3.3V düşük voltaj TTL (LVTTL),
- Düşük voltajlı CMOS (LVCMOS), 3.3V, 2.5V, 1.8V, 1.5V veya 1.2V,
- 3.3V PCI, 33MHz veya 66MHz,
- Genel kullanımlı hafıza uygulamalarında, HSTL I, II ve III için 1.5V ve 1.8V,
- Genel kullanımlı hafıza uygulamaları için, STL I ve II için 1.8V, 2.5V ve 3.3V,

Spartan 3A FPGA' sının desteklediği farklı standartlar;

- LVDS, mini LVDS, RSDS ve PPDS I/O' ları 2.5V veya 3.3V,
- Veri yolu LVDS I/O' da 2.5V,
- TMDS I/O' da 3.3V,
- Farklı HSTL ve SSTL I/O' ları,
- LVPECL girişlerinde 2.5V veya 3.3V

EK-2. (Devam) Xilinx spartan 3a ailesi

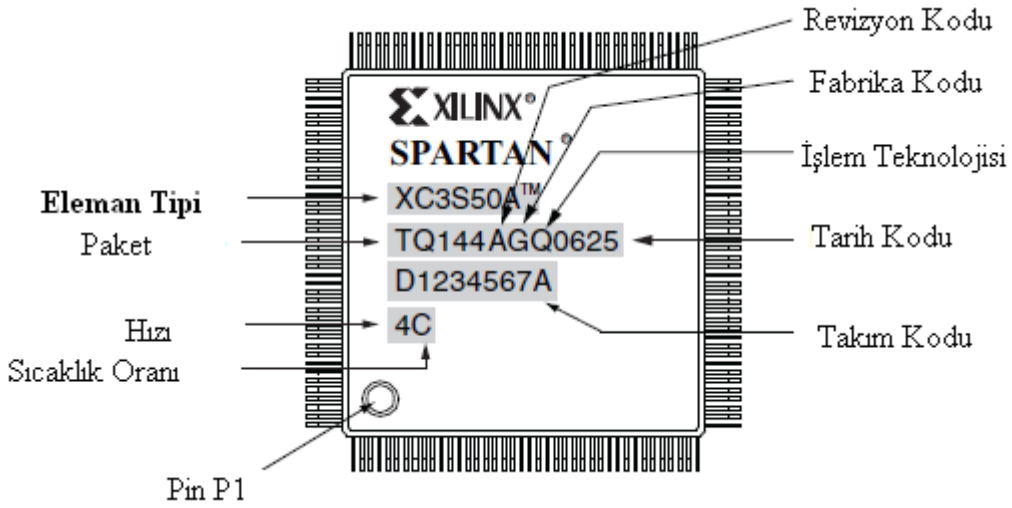
Tablo Ek-2.2. Mevcut kullanıcı I/O' ları ve farklı I/O çiftleri[4]

Kılıf Boyutu (mm)	VQ100 VQG100		TQ144 TQG144		FT256 FTG256		FG320 FGG320		FG400 FGG400		FG484 FGG484		FG676 FGG676	
	14X14		20X20		17X17		19X19		21X21		23X23		27X27	
Eleman	**User**	**Diff**	**User**	**Diff**	**User**	**Diff**	**User**	**Diff**	**User**	**Diff**	**User**	**Diff**	**User**	**Diff**
XC3S50A	68 [13]	60 [24]	108 [7]	50 [24]	144 [32]	64 [32]	*	*	*	*	*	*	*	*
XC3S200A	68 [13]	60 [24]	*	*	195 [35]	90 [50]	248 [56]	112 [64]	*	*	*	*	*	*
XC3S400A	*	*	*	*	195 [35]	90 [50]	251 [59]	112 [64]	311 [63]	142 [78]	*	*	*	*
XC3S700A	*	*	*	*	161 [13]	74 [36]	*	*	311 [63]	142 [78]	372 [84]	165 [93]	*	*
XC3S1400A	*	*	*	*	161 [13]	74 [36]	*	*	*	*	375 [87]	165 [93]	502 [94]	227 [131]

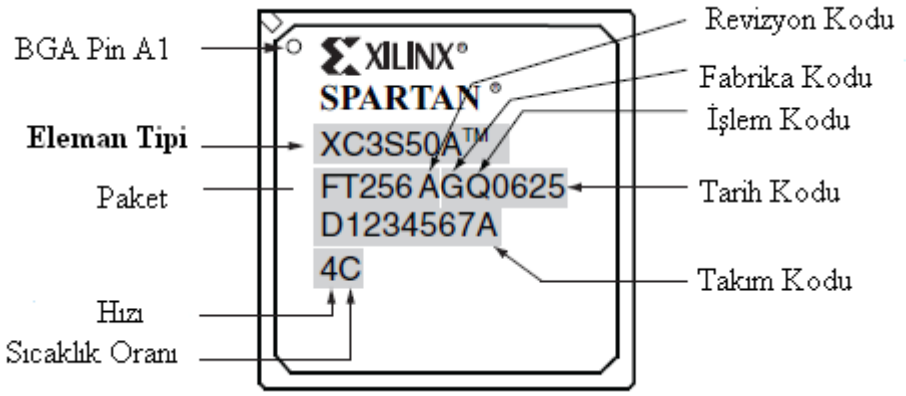
XİLİNX SPARTAN 3A AİLESİNİN KILIF İŞARETLERİ

Şekil Ek.1.2.' de örnek olarak spartan 3A FPGA' sının quad-flat kılıfı üzerindeki işaret ve yazıların ne anlama geldiği görülmektedir. Şekil Ek.1.3.' de ise spartan 3A FPGA' sının BGA kılıfı görülmektedir. BGA kılıflarındaki işaretler tamamıyla quad-flat kılıfındaki ile aynıdır.

EK-2. (Devam) Xilinx spartan 3a ailesi



Şekil Ek-2.2. Spartan 3A quad-flat kılıf örneği

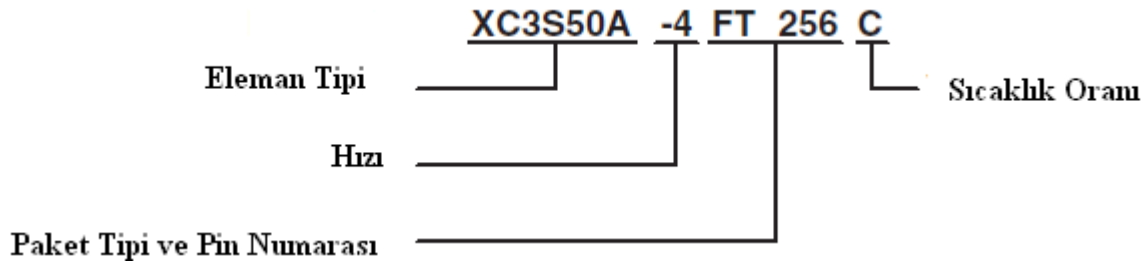


Şekil Ek-2.3. Spartan 3A BGA kılıf örneği

XİLİNX SPARTAN 3A AİLESİNİN ENTEGRE SİPARİŞ BİLGİLERİ

Spartan 3A FPGA' ları her iki standart kılıfda ve her pakette mevcut bulunmaktadır. Aşağıda entegre siparişi verirken satıcı firmaya vermeniz gereken bilgilerin bir örneği gösterilmektedir [4].

EK-2. (Devam) Xilinx spartan 3a ailesi

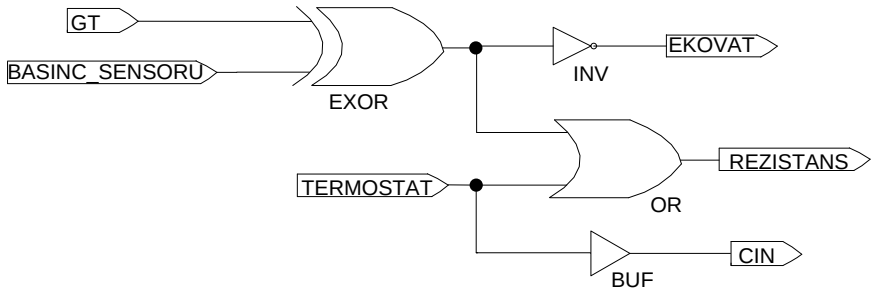


Tablo Ek-2.3. Eleman tipleri ve paket pinleri tablosu[4]

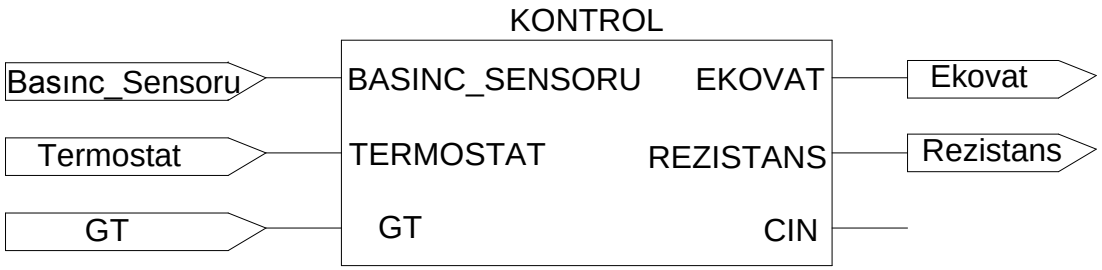
Eleman	Hızı		Paket Tipleri ve Pin Sayıları		Sıcaklık Oranı (Tj)	
XC3S50A	**-4**	Standart Performans	VQ100/	100 pinlik çok ince	**C**	Ticari (0' dan 85oC
XC3S200A	**-5**	Yüksek Performans	VQG100	çift katlı kılıf (VQFP)	**I**	Endüstriyel (-40' dan 100 oC)
XC3S400A			TQ144/	144 pinli ince		
XC3S700A			TQG144	çift katlı kılıf (TQFP)		
XC3S1400A			FT256/	256 pinli çok		
			FTG256	ince grid dizili(FTBGA)		
			FG320/	320 pinli çok		
			FGG320	ince grid dizili(FBGA)		
			FG400/	400 pinli çok		
			FGG400	ince grid dizili(FBGA)		
			FG484/	484 pinli çok		
			FGG484	ince grid dizili(FBGA)		
			FG676/	676 pinli çok		
			FGG676	ince grid dizili(FBGA)		

EK-3. Biodent firinin kontrolü için tasarlanan program

Biodent fırınının programı Xilinx ISE WebPack programı kullanılarak oluşturulmuştur. Bu programlama bizlere kod, şematik ve blok diyagram şeklinde programlama diye üç ayrı seçenek sunmaktadır. Biodent fırınının yazılımı gerçekleştirilirken kod ile ve şematik ile programlama seçenekleri kullanılmaktadır. Kodlar VHDL ve VERİLOG programlama dilleri ile yazılmıştır. Bu bölümde sırası ile VHDL ve VERİLOG kodları, bunların şematikleri verilmiştir.



Şekil Ek-3.1. Termostat, ekovat, start butonu, basınç algılayıcısı değişkenlerinin kontrolünü yapan şematik program



Şekil Ek-3.2. Şekil Ek.4.1.' e ait Schematik programın schematiği

Sistemimiz kendi içerisinde 50MHz' lik saat sinyaline sahiptir, bunu bizim ihtiyacımız olan 1Hz ve 1KHz' e çevirme işlemini gerçekleştiren programlar;

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

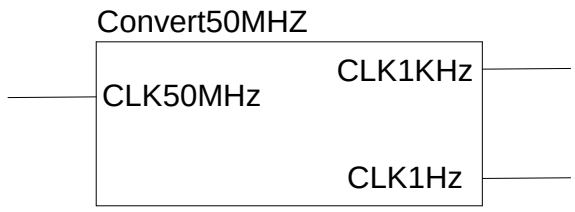
50MHz' lik sinyalden 1Hz' lik sinyal elde edilmesi.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sig_1hz is--1HZ
Port (clk : in STD_LOGIC;
reset_n : in STD_LOGIC;
clk_out : out STD_LOGIC);
end entity sig_1hz;

architecture Behavioral of sig_1hz is
signal clk_sig : std_logic;

begin

process(reset_n,clk)
variable cnt : integer;
begin

if (reset_n='0') then
clk_sig<='0';
cnt:=0;
elsif rising_edge(clk) then
if (cnt=24999999) then  -- 24999999
clk_sig<=NOT(clk_sig);
cnt:=0;
else
cnt:=cnt+1;
end if;
end if;
end process;
clk_out <= clk_sig;
end Behavioral;
```

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

50MHz' lik sinyalden 1KHz' lik sinyal elde edilmesi.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sig_1Khz is--1KHZ
Port (clk : in STD_LOGIC;
reset_n : in STD_LOGIC;
clk_out : out STD_LOGIC);
end entity sig_1Khz;

architecture Behavioral of sig_1Khz is
signal clk_sig : std_logic;
begin

process(reset_n,clk)
variable cnt : integer;
begin

if (reset_n='0') then
clk_sig<='0';
cnt:=0;
elsif rising_edge(clk) then
if (cnt=24999) then  -- 24999
clk_sig<=NOT(clk_sig);
cnt:=0;
else
cnt:=cnt+1;
end if;
end if;
end process;
clk_out <= clk_sig;
end Behavioral;
```



Şekil Ek-3.3. 50MHz' i 1KHz ve 1Hz' e çeviren program kodlarının şematiği

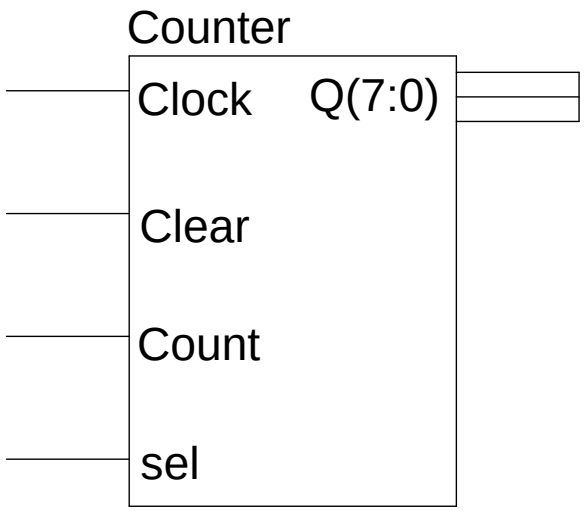
EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

COUNTER_8 bitlik_resetli VHDL kodları.

```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--------------------------------------------------
entity counter is
generic(n: natural :=2);
port(     clock:   in std_logic;
      clear:    in std_logic;
      count:   in std_logic;
        sel:     in std_logic;
      Q:   out std_logic_vector(7 downto 0));
end counter;
--------------------------------------------------
architecture behv of counter is

   signal Pre_Q: std_logic_vector(7 downto 0);

begin
   -- behavior describe the counter

   process(clock, count, clear)
   begin

   if (Pre_Q = 240 and sel = '1'  ) then
   Pre_Q <= Pre_Q - Pre_Q;

   elsif   (Pre_Q = 120 and sel = '0'  ) then
       Pre_Q <= Pre_Q - Pre_Q;

   else
     if clear = '1' then

        Pre_Q <= Pre_Q - Pre_Q;
     elsif (clock='1' and clock'event) then

        if count = '1' then
          Pre_Q <= Pre_Q + 1;

        end if;
     end if;
     end if;
```

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

```
 if (sel = '1' ) then
     Q <= 240-Pre_Q;

     Else

Q <= 120-Pre_Q;
end if;
    end process;
    -- concurrent assignment statement
  end behv;
```



Şekil Ek-3.4. Sayıcı kod kod şeması

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

Sayıcının saydığı darbeleri ikili sayılara çeviren VERİLOG program kodları

```
`timescale 1ns / 1ps

// Copyright (C) 2008 DJ Delorie <dj@delorie.com>
// Distributed under the terms of the GNU General Public License,
// either verion 2 or (at your choice) any later version.

module bcd4(ibin, bcd2, bcd1, bcd0);
   input [7:0] ibin;
   output [3:0] bcd2;
   output [3:0] bcd1;
   output [3:0] bcd0;

   reg [3:0] bcd2;
   reg [3:0] bcd1;
   reg [3:0] bcd0;

always @ (ibin)
begin

  case (ibin)
   0 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b0000; end
   1 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b0001; end
   2 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b0010; end
   3 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b0011; end
   4 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b0100; end
   5 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b0101; end
   6 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b0110; end
   7 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b0111; end
   8 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b1000; end
   9 : begin bcd2 <= 2'b00; bcd1 <= 4'b0000; bcd0 <= 4'b1001; end
   10 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b0000; end
   11 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b0001; end
   12 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b0010; end
   13 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b0011; end
   14 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b0100; end
   15 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b0101; end
   16 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b0110; end
   17 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b0111; end
   18 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b1000; end
   19 : begin bcd2 <= 2'b00; bcd1 <= 4'b0001; bcd0 <= 4'b1001; end
   20 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b0000; end
   21 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b0001; end
```

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

```
22 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b0010; end
23 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b0011; end
24 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b0100; end
25 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b0101; end
26 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b0110; end
27 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b0111; end
28 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b1000; end
29 : begin bcd2 <= 2'b00; bcd1 <= 4'b0010; bcd0 <= 4'b1001; end
30 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b0000; end
31 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b0001; end
32 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b0010; end
33 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b0011; end
34 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b0100; end
35 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b0101; end
36 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b0110; end
37 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b0111; end
38 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b1000; end
39 : begin bcd2 <= 2'b00; bcd1 <= 4'b0011; bcd0 <= 4'b1001; end
40 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b0000; end
41 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b0001; end
42 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b0010; end
43 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b0011; end
44 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b0100; end
45 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b0101; end
46 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b0110; end
47 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b0111; end
48 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b1000; end
49 : begin bcd2 <= 2'b00; bcd1 <= 4'b0100; bcd0 <= 4'b1001; end
50 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b0000; end
51 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b0001; end
52 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b0010; end
53 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b0011; end
54 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b0100; end
55 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b0101; end
56 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b0110; end
57 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b0111; end
58 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b1000; end
59 : begin bcd2 <= 2'b00; bcd1 <= 4'b0101; bcd0 <= 4'b1001; end
60 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b0000; end
61 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b0001; end
62 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b0010; end
63 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b0011; end
64 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b0100; end
65 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b0101; end
66 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b0110; end
```

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

```
67 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b0111; end
68 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b1000; end
69 : begin bcd2 <= 2'b00; bcd1 <= 4'b0110; bcd0 <= 4'b1001; end
70 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b0000; end
71 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b0001; end
72 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b0010; end
73 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b0011; end
74 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b0100; end
75 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b0101; end
76 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b0110; end
77 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b0111; end
78 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b1000; end
79 : begin bcd2 <= 2'b00; bcd1 <= 4'b0111; bcd0 <= 4'b1001; end
80 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b0000; end
81 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b0001; end
82 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b0010; end
83 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b0011; end
84 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b0100; end
85 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b0101; end
86 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b0110; end
87 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b0111; end
88 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b1000; end
89 : begin bcd2 <= 2'b00; bcd1 <= 4'b1000; bcd0 <= 4'b1001; end
90 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b0000; end
91 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b0001; end
92 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b0010; end
93 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b0011; end
94 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b0100; end
95 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b0101; end
96 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b0110; end
97 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b0111; end
98 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b1000; end
99 : begin bcd2 <= 2'b00; bcd1 <= 4'b1001; bcd0 <= 4'b1001; end
100 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b0000; end
101 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b0001; end
102 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b0010; end
103 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b0011; end
104 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b0100; end
105 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b0101; end
106 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b0110; end
107 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b0111; end
108 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b1000; end
109 : begin bcd2 <= 2'b01; bcd1 <= 4'b0000; bcd0 <= 4'b1001; end
110 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b0000; end
111 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b0001; end
```

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

```
112 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b0010; end
113 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b0011; end
114 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b0100; end
115 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b0101; end
116 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b0110; end
117 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b0111; end
118 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b1000; end
119 : begin bcd2 <= 2'b01; bcd1 <= 4'b0001; bcd0 <= 4'b1001; end
120 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b0000; end
121 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b0001; end
122 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b0010; end
123 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b0011; end
124 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b0100; end
125 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b0101; end
126 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b0110; end
127 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b0111; end
128 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b1000; end
129 : begin bcd2 <= 2'b01; bcd1 <= 4'b0010; bcd0 <= 4'b1001; end
130 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b0000; end
131 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b0001; end
132 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b0010; end
133 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b0011; end
134 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b0100; end
135 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b0101; end
136 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b0110; end
137 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b0111; end
138 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b1000; end
139 : begin bcd2 <= 2'b01; bcd1 <= 4'b0011; bcd0 <= 4'b1001; end
140 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b0000; end
141 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b0001; end
142 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b0010; end
143 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b0011; end
144 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b0100; end
145 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b0101; end
146 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b0110; end
147 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b0111; end
148 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b1000; end
149 : begin bcd2 <= 2'b01; bcd1 <= 4'b0100; bcd0 <= 4'b1001; end
150 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b0000; end
151 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b0001; end
152 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b0010; end
153 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b0011; end
154 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b0100; end
155 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b0101; end
156 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b0110; end
```

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

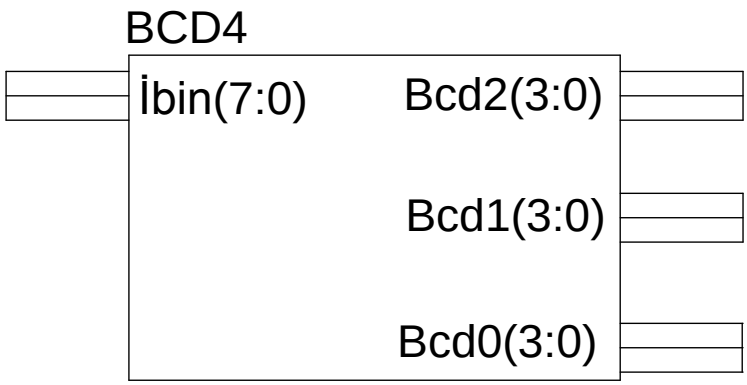
```
157 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b0111; end
158 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b1000; end
159 : begin bcd2 <= 2'b01; bcd1 <= 4'b0101; bcd0 <= 4'b1001; end
160 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b0000; end
161 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b0001; end
162 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b0010; end
163 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b0011; end
164 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b0100; end
165 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b0101; end
166 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b0110; end
167 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b0111; end
168 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b1000; end
169 : begin bcd2 <= 2'b01; bcd1 <= 4'b0110; bcd0 <= 4'b1001; end
170 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b0000; end
171 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b0001; end
172 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b0010; end
173 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b0011; end
174 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b0100; end
175 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b0101; end
176 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b0110; end
177 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b0111; end
178 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b1000; end
179 : begin bcd2 <= 2'b01; bcd1 <= 4'b0111; bcd0 <= 4'b1001; end
180 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b0000; end
181 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b0001; end
182 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b0010; end
183 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b0011; end
184 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b0100; end
185 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b0101; end
186 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b0110; end
187 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b0111; end
188 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b1000; end
189 : begin bcd2 <= 2'b01; bcd1 <= 4'b1000; bcd0 <= 4'b1001; end
190 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b0000; end
191 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b0001; end
192 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b0010; end
193 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b0011; end
194 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b0100; end
195 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b0101; end
196 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b0110; end
197 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b0111; end
198 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b1000; end
199 : begin bcd2 <= 2'b01; bcd1 <= 4'b1001; bcd0 <= 4'b1001; end
200 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b0000; end
201 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b0001; end
```

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

```
202 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b0010; end
203 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b0011; end
204 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b0100; end
205 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b0101; end
206 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b0110; end
207 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b0111; end
208 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b1000; end
209 : begin bcd2 <= 2'b10; bcd1 <= 4'b0000; bcd0 <= 4'b1001; end
210 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b0000; end
211 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b0001; end
212 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b0010; end
213 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b0011; end
214 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b0100; end
215 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b0101; end
216 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b0110; end
217 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b0111; end
218 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b1000; end
219 : begin bcd2 <= 2'b10; bcd1 <= 4'b0001; bcd0 <= 4'b1001; end
220 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b0000; end
221 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b0001; end
222 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b0010; end
223 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b0011; end
224 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b0100; end
225 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b0101; end
226 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b0110; end
227 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b0111; end
228 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b1000; end
229 : begin bcd2 <= 2'b10; bcd1 <= 4'b0010; bcd0 <= 4'b1001; end
230 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b0000; end
231 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b0001; end
232 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b0010; end
233 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b0011; end
234 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b0100; end
235 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b0101; end
236 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b0110; end
237 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b0111; end
238 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b1000; end
239 : begin bcd2 <= 2'b10; bcd1 <= 4'b0011; bcd0 <= 4'b1001; end
240 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b0000; end
241 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b0001; end
242 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b0010; end
243 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b0011; end
244 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b0100; end
245 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b0101; end
246 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b0110; end
```

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

```
247 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b0111; end
248 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b1000; end
249 : begin bcd2 <= 2'b10; bcd1 <= 4'b0100; bcd0 <= 4'b1001; end
250 : begin bcd2 <= 2'b10; bcd1 <= 4'b0101; bcd0 <= 4'b0000; end
251 : begin bcd2 <= 2'b10; bcd1 <= 4'b0101; bcd0 <= 4'b0001; end
252 : begin bcd2 <= 2'b10; bcd1 <= 4'b0101; bcd0 <= 4'b0010; end
253 : begin bcd2 <= 2'b10; bcd1 <= 4'b0101; bcd0 <= 4'b0011; end
254 : begin bcd2 <= 2'b10; bcd1 <= 4'b0101; bcd0 <= 4'b0100; end
255 : begin bcd2 <= 2'b10; bcd1 <= 4'b0101; bcd0 <= 4'b0101; end

 Endcase

End

Endmodule
```



Şekil Ek-3.5. Sayıcının saydığı clock palsleri ikili sayılara çeviren programın şematiği

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

Display sürücü kısmının kodları.

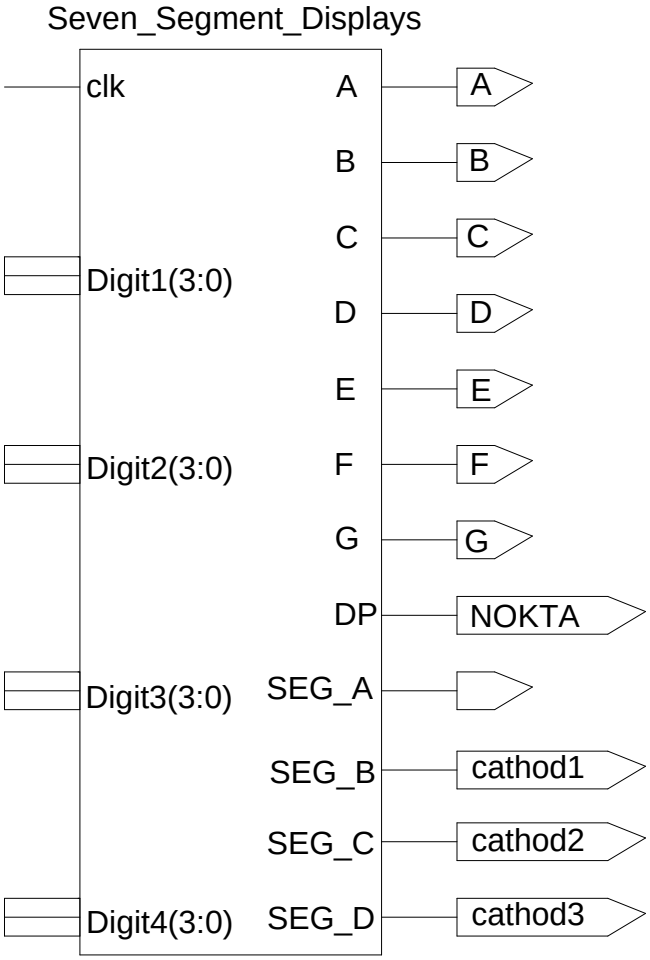
```
module seven_segment_displays(digit1, digit2, digit3, digit4, clk, A, B, C, D, E, F,
G, DP, SEG_A, SEG_B, SEG_C, SEG_D);

// Port declarations
  input [3:0] digit1;  //Least significant digit input
  input [3:0] digit2;
  input [3:0] digit3;
  input [3:0] digit4;  // Most significant digit input
  input clk;
  output reg A;
  output reg B;
  output reg C;
  output reg D;
  output reg E;
  output reg F;
  output reg G;
  output reg DP;
  output reg SEG_A;
  output reg SEG_B;
  output reg SEG_C;
  output reg SEG_D;

// Internal Variables
  reg [4:0] display;

initial SEG_D = 1;

// Multiplex digits
  always @ (posedge clk)
```

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

```
begin
   {SEG_A, SEG_B, SEG_C, SEG_D} <= {SEG_B, SEG_C, SEG_D, SEG_A};

   case({SEG_A, SEG_B, SEG_C, SEG_D})
     4'b0001: display <= digit1;
     4'b0010: display <= digit2;
     4'b0100: display <= digit3;
     4'b1000: display <= digit4;
   endcase
  end

// Convert BCD to output
  always @ (posedge clk)

   begin

    if(display[8]) {G, F, E, D, C, B, A, DP} <= 8'b00000000;

    else

      case(display)
        5'h01: {G, F, E, D, C, B, A, DP} <= 8'b00001100;    //      --A---
        5'h02: {G, F, E, D, C, B, A, DP} <= 8'b10110110;    //     /     /
        5'h03: {G, F, E, D, C, B, A, DP} <= 8'b10011110;    //     F     B
        5'h04: {G, F, E, D, C, B, A, DP} <= 8'b11001100;    //    /       /
        5'h05: {G, F, E, D, C, B, A, DP} <= 8'b11011010;    //    --G---
        5'h06: {G, F, E, D, C, B, A, DP} <= 8'b11111010;    //   /       /
        5'h07: {G, F, E, D, C, B, A, DP} <= 8'b00001110;    //   E       C
        5'h08: {G, F, E, D, C, B, A, DP} <= 8'b11111110;    //  /      /
        5'h09: {G, F, E, D, C, B, A, DP} <= 8'b11011110;    //  --D---   DP
```

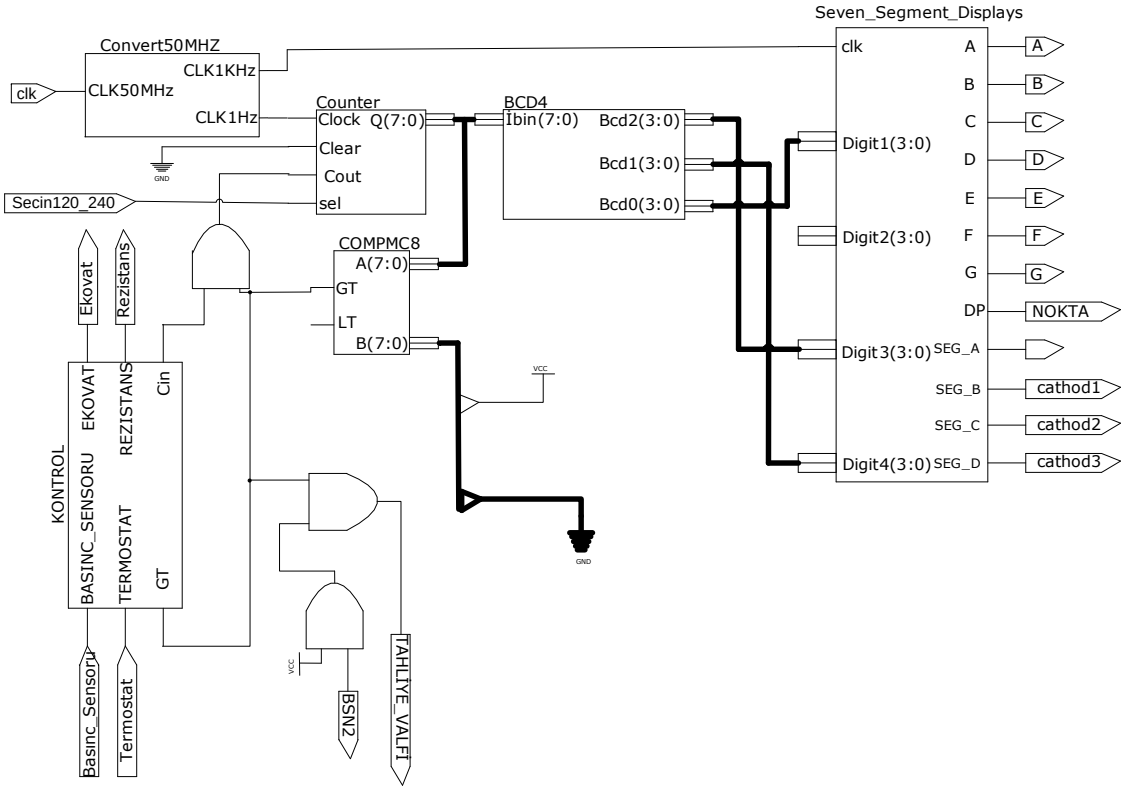
EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program

```
    // 5'h0A: {G, F, E, D, C, B, A, DP} <= 8'b11101110;
    // 5'h0B: {G, F, E, D, C, B, A, DP} <= 8'b11111000;
    // 5'h0C: {G, F, E, D, C, B, A, DP} <= 8'b01110010;
    // 5'h0D: {G, F, E, D, C, B, A, DP} <= 8'b10111100;
    // 5'h0E: {G, F, E, D, C, B, A, DP} <= 8'b11110010;
    // 5'h0F: {G, F, E, D, C, B, A, DP} <= 8'b11100010;
      5'h00: {G, F, E, D, C, B, A, DP} <= 8'b01111110;
    endcase
  end
endmodule
```



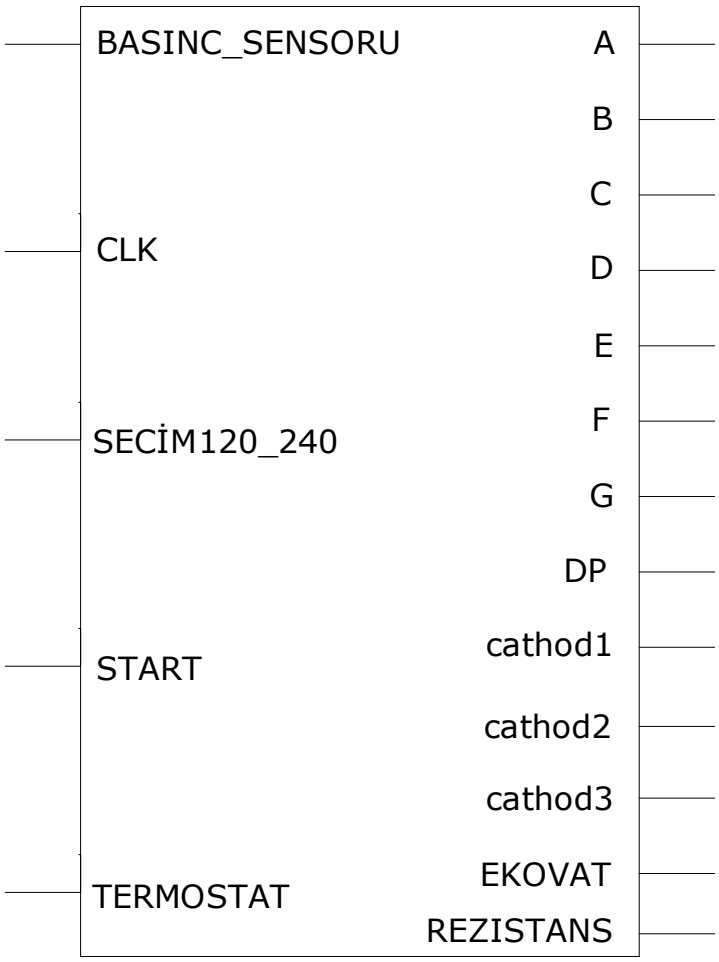
Şekil Ek-3.6. Display sürücü kodlarının şematiği

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program



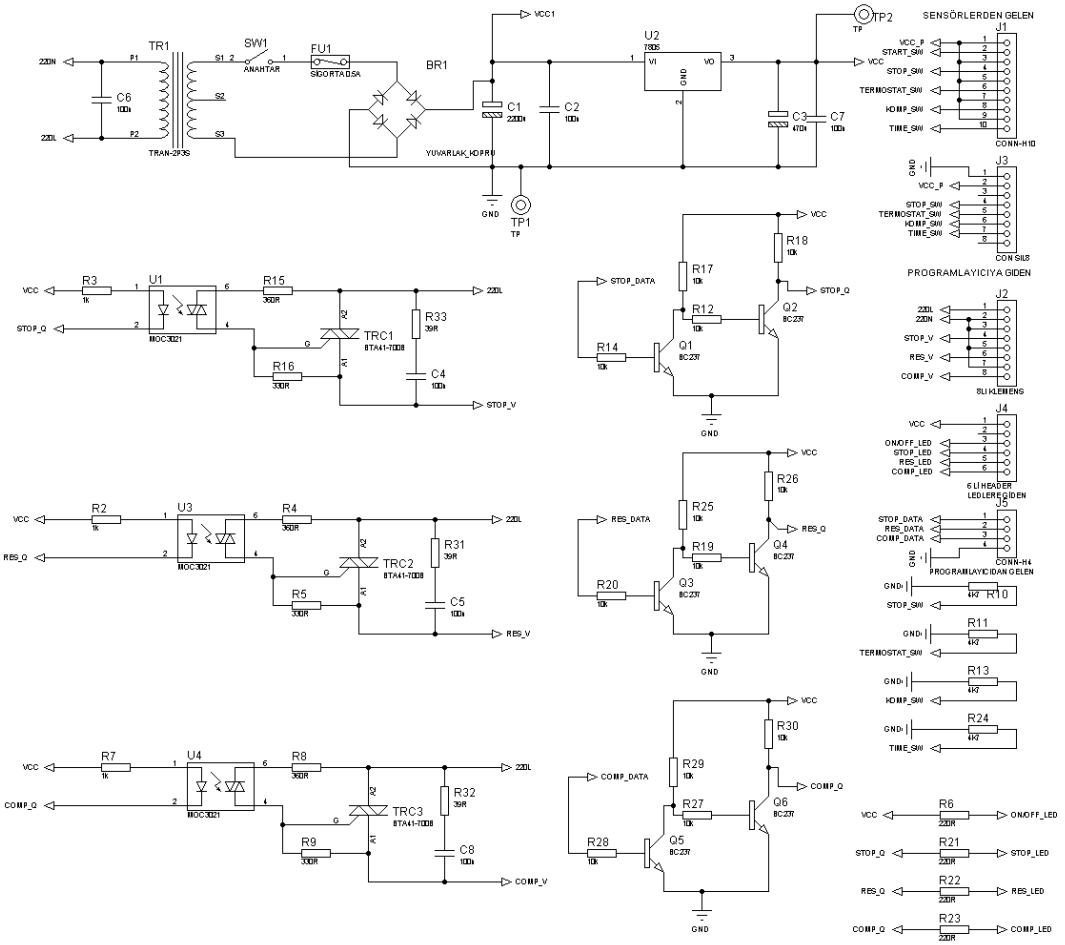
Şekil Ek-3.7. Biodent fırınına ait şematik program kodu

EK-3. (Devam) Biodent firinin kontrolü için tasarlanan program



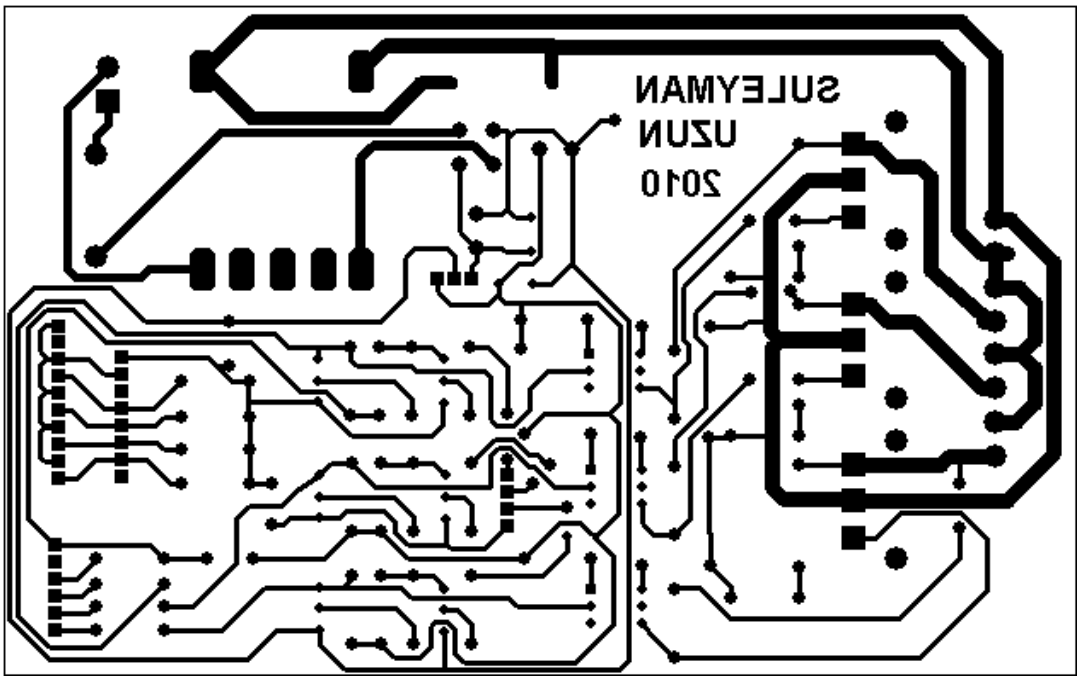
Şekil Ek-3.8. Biodent fırınına ait şematik program kodunun şematiği

EK-4. Biodent fırını için tasarlanan devrelerin devre şeması ve baskı devreleri



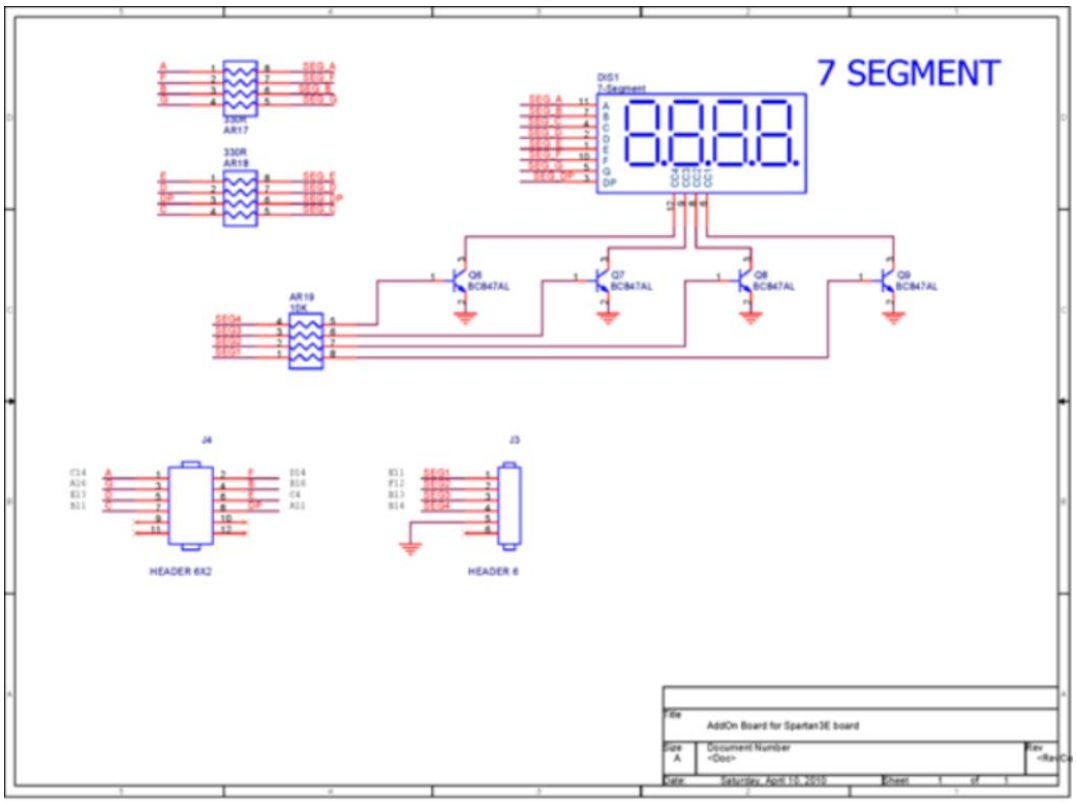
Şekil Ek-4.1. Kontrol kartı devre şeması

EK-4. (Devam) Biodent fırını için tasarlanan devrelerin devre şeması ve baskı devreleri



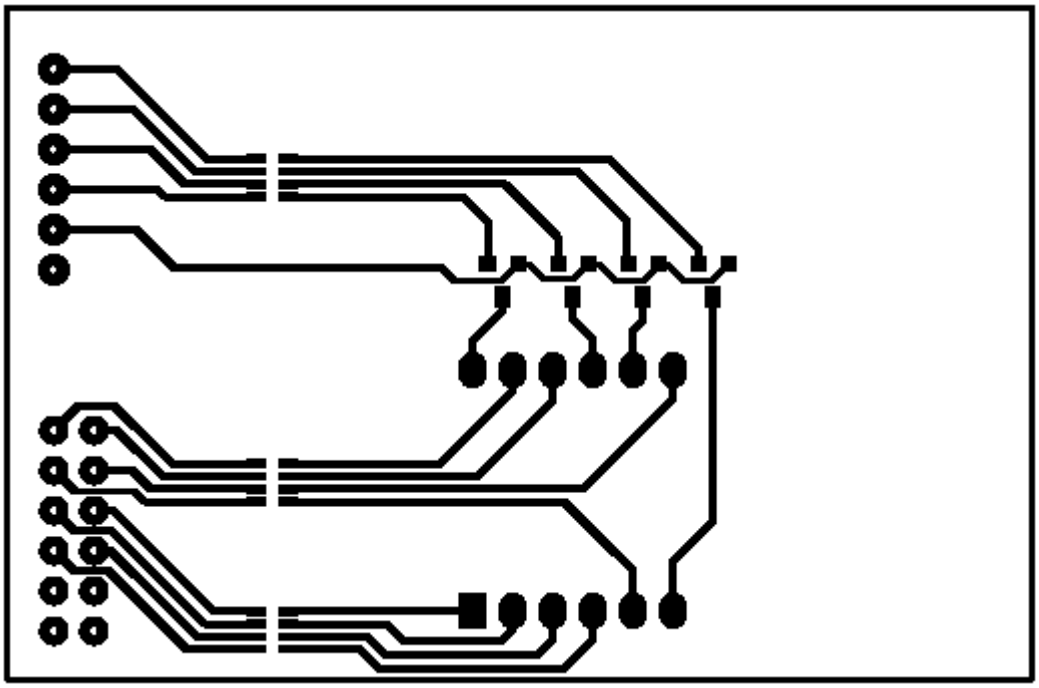
Şekil Ek-4.4. Kontrol kartı baskı devresi

EK-4. (Devam) Biodent fırını için tasarlanan devrelerin devre şeması ve baskı devreleri



Şekil Ek-4.3. Display kartı devre şeması

EK-4. (Devam) Biodent fırını için tasarlanan devrelerin devre şeması ve baskı devreleri

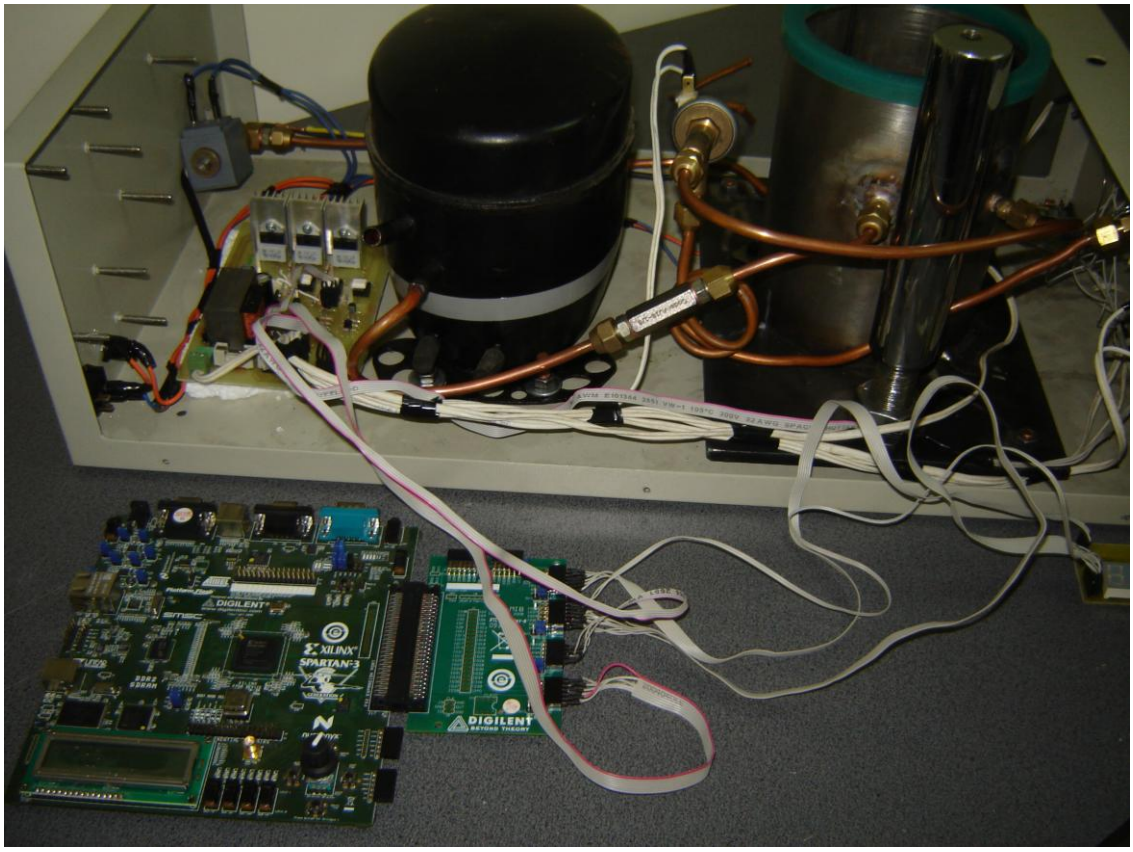


Şekil Ek-4.4. Display kartı baskı devre şeması

EK-5. (Devam) Tasarlanan biodent fırınının resimleri

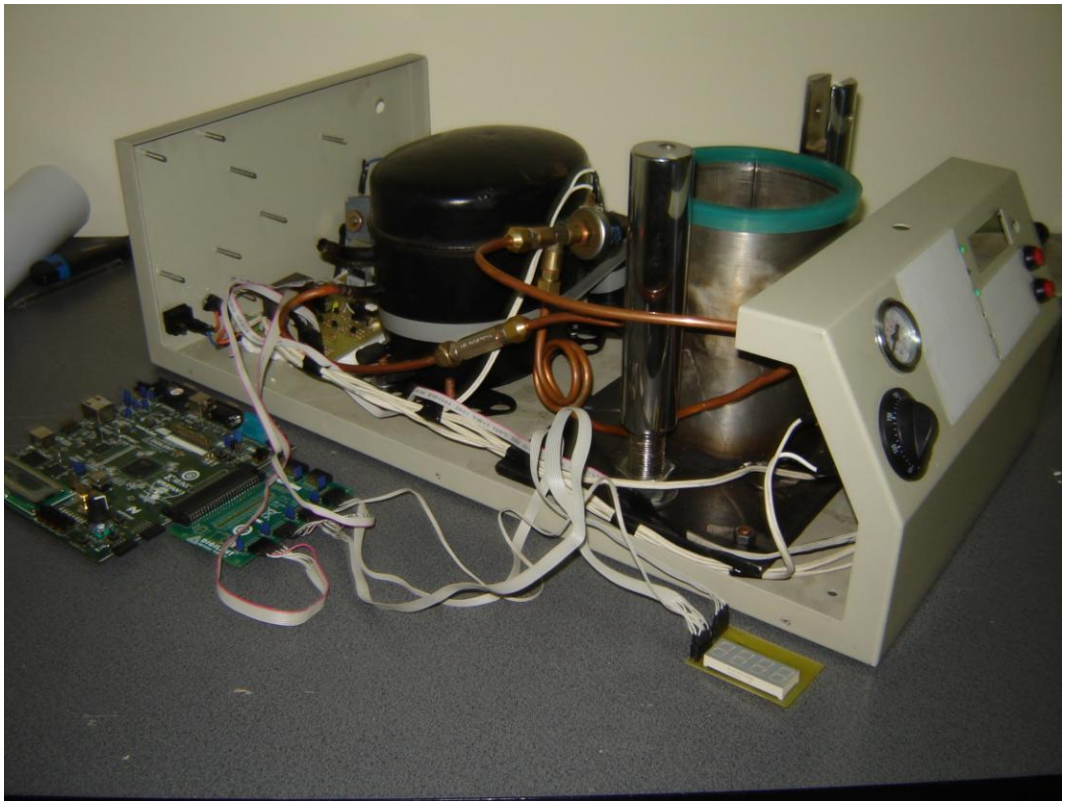
Resim Ek-5.1. Biodent fırınının iç görünümü

EK-5. (Devam) Tasarlanan biodent fırınının resimleri



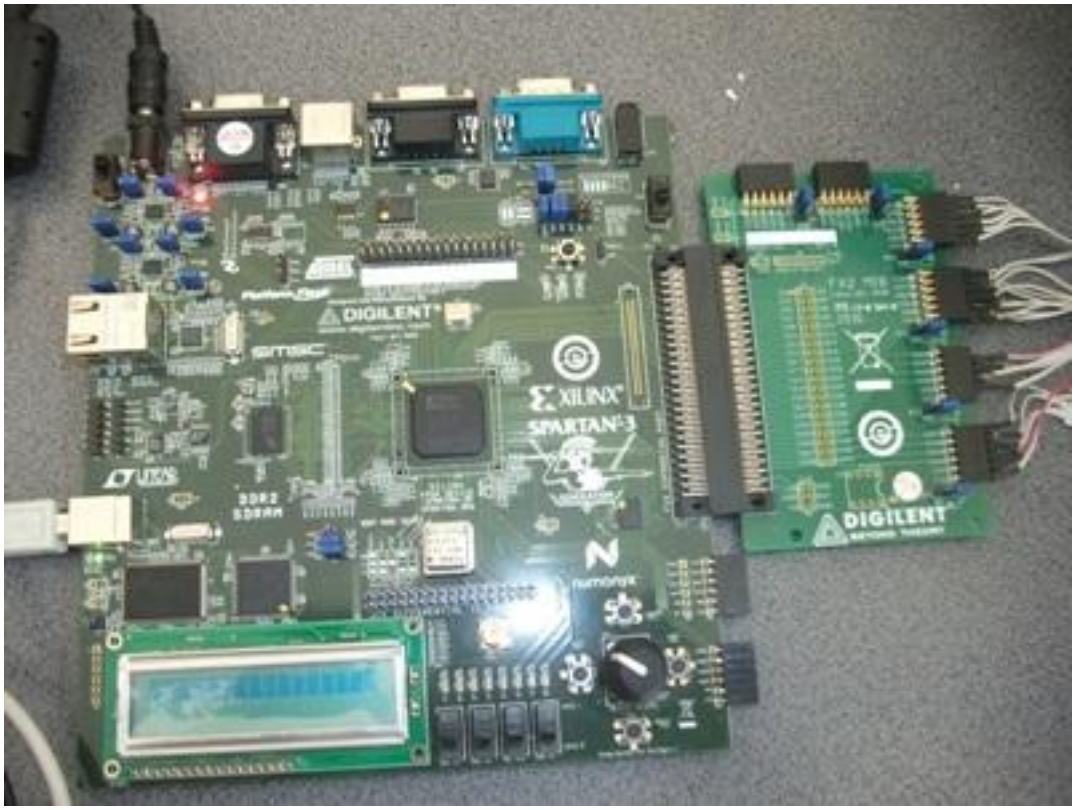
Resim Ek-5.2. Biodent fırını deney düzeneği

EK-5. (Devam) Tasarlanan biodent fırınının resimleri



Resim Ek-5.3. Biodent fırını programın yüklenmesi

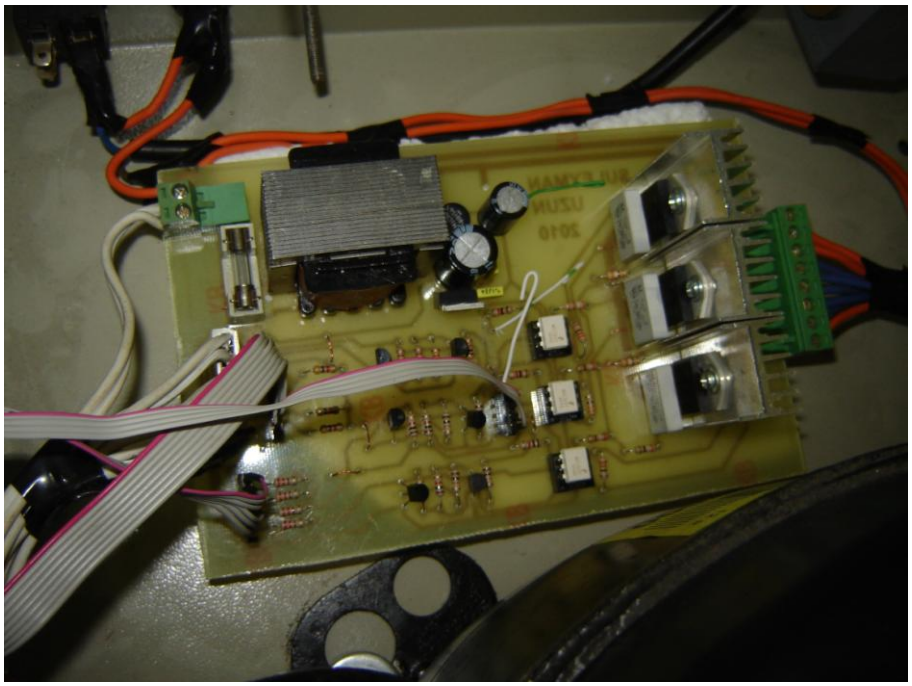
EK-5. (Devam) Tasarlanan biodent fırınının resimleri



Resim Ek-5.4. FPGA deney kartı

EK-5. (Devam) Tasarlanan biodent fırınının resimleri

Resim Ek-5.5. Display kartı



Resim Ek-5.6. Biodent fırının kontrol kartı

EK-5. (Devam) Tasarlanan biodent fırınının resimleri

Resim Ek-5.7. Biodent fırını kontrol kartı bağlantısı

Resim Ek-5.8. Biodent fırının son hali

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: UZUN, Süleyman
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 06.11.1984 Mersin
Medeni hali	: Evli
Telefon	: 0 (446) 751 30 21
Fax	: 0 (446) 751 30 22
e- mail	: uzun_suleyman@hotmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Gazi Üniversitesi/Elektronik Öğretmenliği	2008
Lise	Mersin Endüstri Meslek Lisesi/ Elektronik Bölümü	2002

İş Deneyimi

Yıl	Yer	Görev
2000 – 2004	Platform Bilgisayar	Teknisyen
2008 – 2009	Lider Diş A.Ş.	Tekniker

Yabancı Dil

İngilizce

Yayınlar

Hobiler

Bilgisayar teknolojileri, Futbol, Basketbol, Güreş