T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİR DC MOTORUN FPGA TABANLI BULANIK KONTROLÜ

Fatma Betül ARICI

YÜKSEK LİSANS TEZİ

Elektronik ve Bilgisayar Sistemleri Eğitimi

Anabilim Dalı

Ağustos-2017
KONYA

TEZ KABUL VE ONAYI

Fatma Betül Arıcı tarafından hazırlanan "BİR DC MOTORUN FPGA TABANLI BULANIK KONTROLÜ " adlı tez çalışması 08/09/2017 tarihinde aşağıdaki jüri tarafından oy birliği ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Bilgisayar Sistemleri Eğitimi Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan-Danışman Doç. Dr. İsmail SARITAŞ	.....
Üye Yrd. Doç. Dr. Abdullah Erdal TÜMER	.....
Üye Yrd. Doç. Dr. İlker Ali ÖZKAN	.....

Yukarıdaki sonucu onaylarım.

Prof. Dr. Mustafa YILMAZ
FBE Müdürü

Bu tez çalışması S.Ü. BAP tarafından 13201037 nolu proje ile desteklenmiştir.

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Fatma Betül Arıcı

Tarih: 08.09.2017

ÖZET

YÜKSEK LİSANS TEZİ

BİR DC MOTORUN FPGA TABANLI BULANIK KONTROLÜ

Fatma Betül Arıcı

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Elektronik ve Bilgisayar Sistemleri Eğitimi Anabilim Dalı

Danışman: Doç. Dr. İsmail SARITAŞ

2017, 80 Sayfa

Jüri
Doç. Dr. İsmail SARITAŞ
Yrd. Doç. Dr. Abdullah Erdal TÜMER
Yrd. Doç. Dr. İlker Ali ÖZKAN

DC motorlar günümüzde birçok alanda yaygın olarak kullanılmaktadır. Kullanılan DC motorların literatürde birçok kontrol metodu bulunmaktadır

Bu çalışmada literatürde bulunan kontrol metotlarına ek olarak FPGA tabanlı Bulanık mantık denetleyici kullanılmıştır.

Çalışmada ilk olarak DC motorun denetleyicisiz olarak boştaki ve farklı yük değerlerinde Akım, Gerilim, Devir, Güç ve Yük parametreleri üzerinden çalışması incelenmiştir. Motorun boşta ve kademeli olarak yük altında çalışırken "Devir-zaman", "Gerilim-zaman", "Akım-zaman", "Güç-zaman" ve "Yük-zaman" grafikleri elde edilmiştir. DC motor sabit gerilim ile çalıştırıldığında yüke bağlı olarak devir değeri düşmekte, yükü karşılayabilmek için akım değeri yükselmektedir. Bu problemi gidermek için DC motor parametlerini kullanan bir bulanık kontrol yapısı Matlab üzerinde tasarlanmıştır. Tasarlanan bu kontrolörün FPGA tabanlı programlaması gerçekleştirilmiştir. FPGA tabanlı Bulanık Denetleyici ile Motor milindeki yük ve devir değişimine göre uygun PWM sinyali gönderilmiştir. Aynı zamanda Motor milindeki yük miktarı değişimlerinde FPGA'in yüksek hızlı tepkime verme özelliği motorun performanslı bir şekilde çalışması sağlamıştır.

Anahtar Kelimeler: Bulanık Mantık Denetleyici, Doğru Akım Motor, FPGA, Kontrol

ABSTRACT

MS THESIS

THE CONTROL OF DC MOTOR WITH FPGA-BASED FUZZY CONTROL

Fatma Betül Arıcı

Selcuk University
Faculty of Technical Education
Department of Education of Electronic and Computer Systems

Advisor: Assoc. Prof. Dr. İsmail SARITAŞ

2017, 80 Pages

Jury
Assoc. Prof. Dr. Dr. İsmail SARITAŞ
Assist. Prof. Dr. Abdullah Erdal TÜMER
Assist. Prof. Dr. İlker Ali ÖZKAN

DC motors are widely used in many fields today. There are lots of control methods for DC motors which are used in the literature. FPGA based fuzzy logic controller is used in addition to control methods in this study. In the study, firstly the operation of DC motor without any controllers and in the position of out on the parameters of Current, Voltage, Revolution, Power, Load in different load values is reviewed. The graphics of "Revolution-Time", "Voltage-Time", "Current-Time", "Load-Time" are obtained while motor is out and operated under any load in stages. When DC motor is operated with constant voltage, the value of revolution decreases based on load and the value of current increases in order to cover load. In order to get rid of this problem, a structure of fuzzy logic, which uses the parameters of DC motor, is designed on Matlab. FPGA based programming of this designed controller is realized. Appropriate PWM signal is sent according to the changes in load and revolution in motor shaft by the FPGA based fuzzy logic controller. At the same time, the specification of FPGA of quick reaction in the changes of the amount of load in motor craft provides that the motor can be operated with performance.

Keywords: Control, Direct Current Motor, Fuzzy Logic Controller, FPGA

ÖNSÖZ

Kolay kontrol edilebilme ve yüksek performans gibi üstünlüklere sahip olan doğru akım motorlarının hızları geniş sınırlar içerisinde ayarlanabilmektedir. DC motorları endüstride hızlı taşımacılık, elektrik trenleri, elektrikli taşıtlar ve elektrikli vinçler gibi uygulamalarda ayarlanabilir hız ve hassas konumlandırma uygulamalarında kullanılırlar. Son yıllarda teknolojik gelişmelerle birlikte ev aletleri uygulamalarında, düşük güçlü ve düşük maliyet istenen ayarlanabilir hız kontrollü yerlerde yaygın bir kullanım alanı bulmuştur.

Bu projede, iki adet deney kartı tasarlanmış tasarımda Microchip'in PIC serisi 16F877 mikro denetleyicisi kullanılmıştır. Bu denetleyicilerin üretim amacı çok fonksiyonlu lojik uygulamalarının hızlı ve ucuz bir denetleyici ile yazılım yoluyla karşılanmasını sağlamaktadır. Bu denetleyici fiyat, çevre birimleri, kolay programlama ve kullanım esnekliği gibi üstün özelliklere sahiptir. Birinci deney kartında motor milindeki değişik yük miktarına göre; Yük, devir, akım ve gerilim değerlerine ait veriler bilgisayar ortamına aktarılarak grafiksel olarak incelenmiş ve gözlenmiştir. İkinci deney kartında ise DC motora ait veriler (Yük, akım, gerilim, devir) FPGA kartına aktarılarak Bulanık Mantık tekniğiyle en uygun yük miktarına göre DC motora uygulanan gerilim PWM değeri ile kontrol edilerek DC motorun verimli çalışması sağlanmıştır. Daha başarılı sonuçların alınması için büyük güçlü DC motor, dinamometre vb. gereksinimler ortaya çıkmış ancak bütçenin sınırlı olması nedeniyle fakülte imkânları kullanılmıştır. Tez çalışmasında revize edilen hedefler ve eldeki imkânlar ölçüsünde çalışmalar yapılmış ve büyük oranda başarılı sonuçlar alınarak tamamlanmıştır.

Bu tez çalışmasında her zaman yanımda olan ve desteklerini esirgemeyen danışmanım Doç. Dr. İsmail SARITAŞ'a, programlamada yardımcı olan Yrd. Doç. Dr. İlker Ali ÖZKAN'a ve Aziz KAĞITÇI'ya, 13201037 numaralı proje ile destek olan Selçuk Üniversitesi Bilimsel Araştırmalar (BAP) Koordinatörlüğüne teşekkürlerimi sunarım.

Fatma Betül Arıcı
KONYA-2017

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Simgeler

Simge	Açıklaması
V	Volt
A	Amper
W	Watt
kW	Kilowat
AC	Alternatif Akım
DC	Doğru Akım
Hz	Hertz
cm	Santimetre
mm	Milimetre
ms	Milisaniye
Kg	Kilogram

Kısaltmalar

Kısaltma	Açıklama
FPGA	Alanda programlamalı kapı dizisi
VHDL	Çok yüksek hızlı entegre devre donanım tanımlama dili
Matlab	MATrix LABoratory yazılımı
SPLD	Basit programlanabilir mantıksal aygıt
CPLD	Karmaşık programlanabilir mantıksal aygıt
IC	Tümleşik devre
CMOS	Bütünleyici Metal Oksit Yarıiletken
CLB	Yapılandırılabilir Lojik Bloklar
RAM	Rastgele erişimli bellek
LUT	Arama tablosu
CLB	Yapılandırılabilir Mantık Blok
EPROM	Silinebilir programlanabilir salt okunur bellek
EEPROM	Elektronik silinebilir programlanabilir salt okunur bellek
DSP	Dijital sinyal işleyici
HDL	Donanım tanımlama dili
LCA	Mantıksal hücredizisi
IP	İnternet protokol
PV	Fotovoltaik

1. GİRİŞ

Kolay kontrol edilebilme ve yüksek performans gibi üstünlüklere sahip olan doğru akım motorlarının hızları geniş sınırlar içerisinde ayarlanabilmektedir. DC motorları endüstride hızlı taşımacılık, elektrik trenleri, elektrikli taşıtlar ve elektrikli vinçler gibi uygulamalarda ayarlanabilir hız ve hassas konumlandırma uygulamalarında kullanılırlar. Son yıllarda teknolojik gelişmelerle birlikte ev aletleri uygulamalarında, düşük güçlü ve düşük maliyet istenen ayarlanabilir hız kontrollü yerlerde yaygın bir kullanım alanı bulmuştur. Geniş uygulama alanı bulmasının diğer bir sebebi de alternatif akım motorlarına göre kontrolünün daha kolay olmasıdır. Alternatif akım motor sürücüleri ile kıyaslandığında, DC motor sürücü devreleri hem basit, hem de ucuzdur. Sürücü tasarımı ile uygulamaya geçirilmenin daha basit olması, ayarlanabilir hız uygulamalarında doğru akım motor sürücülerini ön plana çıkarmıştır. Endüstride değişken hızlı sürücü sistemleri arasında geniş bir uygulama alanı bulabilen DC sürücülerinin uygulamasında kullanılan analog sürücüler, analog devre elemanları ile uygulanan karmaşık kontrol şemaları gibi dezavantajlara sahiptir. Yarı iletken teknolojisindeki gelişmeler mevcut olan sistemlerden daha küçük olan, daha hızlı işlem yapabilen, ekonomik ve ayarlanabilir hızlı DC sürücülerin uygulamada kullanılmasına yol açmıştır. Uygulamalarda sürücünün yanı sıra, kullanılan hız denetim sistemi de önemlidir. DC motorunun sabit hız uygulamalarında açık döngülü sistem kullanılmaz. Bu sistemlerde yükle motor gerilimindeki ve motor devir sayısındaki değişiklikler dikkate alınmadığından, yük durumuna göre devir artar veya azalır. Bu nedenle, sabit hız uygulamalarda açık döngülü sistem tercih edilmeyip, bunun yerine kapalı döngü sistemleri kullanılır.

Bu çalışmada kullanılan iki adet deney kartında Microchip'in PIC serisi 16F877 mikrodenetleyicisi kullanılmıştır. Bu denetleyicilerin üretim amacı çok fonksiyonlu lojik uygulamalarının hızlı ve ucuz bir denetleyici ile yazılım yoluyla karşılanmasını sağlamaktadır. Bu denetleyici 8031 ve 8051 ailesine göre fiyat, çevre birimleri, kolay programlama ve kullanım esnekliği gibi üstün özelliklere sahiptir. Birinci deney kartında motor milindeki değişik yük miktarına göre; Yük, Devir, Akım, gerilim ve moment değerlerine ait veriler bilgisayar ortamına aktarılarak grafiksel olarak incelenmiş ve gözlenmiştir. İkinci deney kartında ise DC motora ait veriler (Yük, Akım, Gerilim, Devir) FPGA kartına aktarılarak Bulanık Mantık tekniğiyle en uygun yük miktarına göre DC motora uygulanan gerilim PWM değeri ile kontrol edilerek DC motorun verimli çalışması sağlanacaktır.

2. KAYNAK ARAŞTIRMASI

Literatürde FPGA ile motor kontrolleri ile ilgili çalışmalarla karşılaşılmıştır. Ancak tez amaç ve kapsamı ile ilgili bir çalışmaya rastlanmamıştır. Bu bölümde FPGA ile DC motor kontrolüyle ilgili yapılan çalışmalar ve deneylerin literatür özetlerine yer verilmiş olup tez projesi kapsamında kaynak araştırması aşağıya özet olarak listelenmiştir.

(Şahin 2004) yaptığı çalışmasında FPGA kullanılarak YSA'nın donanımsal gerçekleştirilmesi sağlanmıştır. Dijital sistem mimarisi ile birçok katmanlı yapay sinir ağı tasarımı yapılmıştır. Tasarım mimarisi Very High Speed Integrated Circuits Hardware Description Language (VHDL) ile tamamlanmış ve FPGA entegre devresi üzerinde gerçeklenmiştir. Yapılan tasarımda VLSI ile yapılan tasarımlara göre hem zaman hem maliyet açısından avantajları olduğu görülmektedir.

(Yazıcı 2004) çalışmasında bir girişli-bir çıkışlı ve bir girişli-iki çıkışlı fonksiyonların, FPGA kullanarak bulanık modellemesi yapılmıştır. Girişin üyelik derecesinin belirlenmesinde, tablo yöntemi kullanılarak, girişe ait bulanık değerler, FPGA'de bir ROM içerisinde tutulmaktadır. Sistemde bulanık çıkarım metodu çarpım yöntemi ve durulama metodu merkezi ortalama yöntemidir. Modellemenin tüm aşamalarının tanımlanmasın VHDL kullanılmıştır. Bir girişli- bir çıkışlı sistemde çarpma ve toplama modelleri paralel olarak çalışmaktadır.

(Cirstea ve ark 2001) yaptıkları araştırma çalışmasında dizel arabalarda synchronous generator sistemi için bir bulanık mantık kontrolörün tasarımı, simülasyonu ve ve uygulaması gerçekleştirilmiştir. Kontrolör VHDL kullanılarak geliştirilmiş ve FPGA üzerinde uygulanmıştır.

(Cirstea ve ark 2000) çalışmalarında VHDL kullanılarak yeni bir yapay sinir ağı motor indüksiyon kontrol stratejisi geliştirilip XILINX FPGA üzerinde uygulaması sunulmuştur. Çalışmada hız kontrol sistemi tartışılmıştır. Yeni geliştirilen yapay sinir ağı kontrol sisteminin bilgisayar ile simülasyon ve uygulama testleri gerçekleştirilmiştir.

(Petko ve Uhl 2001) çalışmalarında kompleks mekatronik sistemler için bir kontrol tasarımı fikri sunulmaktadır. Bu fikir model simülasyonlarını kullanarak sanal prototiplerin kontrolüne dayanmaktadır. Çevrimdışı ve gerçek zamanlı olmak üzere iki tip model simülasyon türü vardır. Gömülü kontrol tasarımı konseptinde FPGA kullanımı temel alınmaktadır. Burada bir robot kolunun kontrol tasarımı sunulmaktadır.

(Kongmunvattana ve Chongstivatana 1998) çalışmalarında, programlanabilir FPGA tabanlı kontrole sahip bir gezgin robot kapsama mimarisinin tasarımı ve uygulanması tanımlanmaktadır. Bu çalışma FPGA tasarımı ve kavrama mimarisinden

oluşan iki kavramın birleştirilmesini göstermektedir. Bu yaklaşım bir mobil kontrol çipinin etkinliğini göstermektedir. Tasarım Xilinx 4003 kullanılarak uygulanmıştır. Bir davranış modeli örneği tartışılmış ve mantıksal tasarım olarak formüle edilmiştir. Bu kontrol kapsama mimarisinin gerçek zamanlı kontrolü üzerine tasarlanmıştır. Burada FPGA'in tasarım ve uygulamada başarıyla yapılandırılabildiği görülmektedir.

(Ichikawa 2005) çalışmasında bir FPGA yongasının bütün bir kontrol sistemine entegresi için bir kontrol kütüphanesi geliştirilmesi ve uygulanması anlatılmaktadır. FPGA kontrol sistemi bir örnek uygulama ile sunulmuştur. Çalışmada FPGA ve PLC kontrolleri karşılaştırılmış, zeki sistemleri oluşturmada ve analizi zor olduğu için ticari sırları gizlemede FPGA'ın daha iyi olduğu belirtilmiştir.

Cirstea ve ark (2000) yaptıkları çalışmada, üç fazlı elektrik sürücü sistemleri için dijital akım kontrol mimarisi VHDL ile tasarımı anlatılmaktadır. Kontrol donanımsal yapay sinir ağı ile gerçekleştirilmiştir. Sistemin programlanması VHDL ile yapılmış ve tek bir FPGA yongasına uygulanmıştır.

(Hassan ve Sharif 2007) çalışmalarının amacı FPGA üzerinde PID benzeri bir bulanık mantık kontrolörün gerçekleştirilmesidir. Kontrolörde bulanık çıkarım mekanizması aktif kural seçim mekanizmasını desteklemektedir.

(Klarenbach ve Krah 2008) yaptıkları çalışmada; kontrollü servo sürücüleri üretim makineleri ve makine araçları sürücüleri gibi otomasyon teknolojisinin, robot ve taşıma sistemlerinin birçok alanında kullanıldığını, kontrol döngülerinin artan bant genişliği, hız ve istikrar gibi gereksinimleri bulunduğunu, servo sürücü kontrolü fonksiyonu özel bir kontrolör olarak değil FPGA kullanılarak yapıldığını, bu yaklaşımın yeni teknolojilerin üzerine kurulmuş olduğunu belirtmişlerdir.

(Poorani ve ark 2005) çalışmalarında doğru akım motorlarının, özellikle şönt motorların, günümüzde elektrikli taşıtlar için popüler olduğunu ifade etmişlerdir. Bu çalışmada aracın motor hızı kontrol edilerek, taşıtın sürülmesi amaçlanmaktadır. Bu bağlamda ivme, fren, enerji durumu, vites, arazi vb. parametreler olarak kabul edilebilir. Kapalı döngülü bu sistem, yukarıdaki parametreler ile birlikte motor hızı parametresini alıp motor hızını tahmin etmektedir. Bu makale, elektrikli araçların hız kontrolü için FPGA tabanlı bulanık mantık kontrolü uygulamasını göstermektedir.

(Hsu ve Lee 2011) yaptıkları çalışmada bir DC motorun kontrolü için adaptif PID kontrol gerçekleştirmişlerdir. Yaptıkları kontrolde bulanık mantık ile PID katsayılarını ayarlamışlardır. FPGA tabanlı kontrolünü gerçekleştirmişlerdir. Belirsizlik olduğu yerde üst sınırı belirlemek için bulanık mantık kullanmışlardır. Böylece daha verimli sonuçlar aldıklarını ifade etmektedirler.

(Chen ve Lin 2002)'ın sunulan araştırmalarında, alan programlanabilir FPGA uygulamaları için geçerlidir. Gerçek çekirdek tabanlı tasarım ile uygun olarak motor kontrolü sağlanır. (USM) *Ultrasonik Servo* Motor sürücü tasarlanmış ve hayata geçirilmiştir. VHDL ile tasarımı ve simülasyonu yapılan PID IP çekirdek denetleyicisinin akışı takip edilir. Bu şekilde tasarlanan ve mikroişlemci sensörlerinin, FPGA'ye gömülmesiyle oluşturulan sistemlerde yazılımın oldukça güçlü ve performansın yüksek olduğu tespit edilmiştir.

(Astarloa ve ark 2009) araştırmalarında, motor kontrol cihazlarını, gerçek çekirdek tabanlı tasarımına uygun olarak FPGA ile gerçekleştirilmiştir. Donanım (PID IP çekirdek) ve yazılım tasarımının kontrolü ile yörünge hesaplama işlemleri yapılmış olup FPGA içine mikroişlemciler gömülerek bu sistemin kontrolü sağlanmıştır. Sistemin yazılım kısmı VHDL programlama dilinde yazılmış ve PID IP çekirdek denetleyicisi ile sistemin donanım kısmı tamamlanmıştır. Bu yöntemin yüksek performans sağladığı, daha hızlı ve daha güçlü çalışmalar elde edildiği sonucuna varmışlardır.

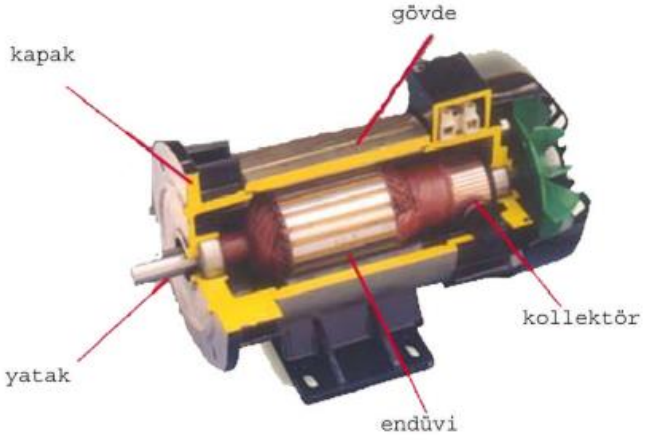
(Hasanien 2011) çalışmasında, bir step motoru için adaptif YSA (Yapay Sinir Ağları) kontrolü ile FPGA tabanlı sürücü hız ayarı kontrolü yapmıştır. Öncesinde geleneksel PI kontrol ile tasarlanan sistemde step motor yük sorunlarıyla karşılaşıldığı ve iyi bir performansın elde edilemediği görülmüştür. FPGA tabanlı YSA kontrolü ile iyi bir performans sağlandığını belirtmiştir.

(de Jesús ve ark 2004) çalışmalarında, CNC freze makineleri için FPGA tabanlı kırık takım tespit sistemi gerçeklenmiştir. Otomatik CNC makinelerinde kırık takım tespiti mekatronik sorunlardan en önemlilerden biridir. Bu çalışmada, donanım sinyal işleme teknikleri kullanılarak, otokolerasyon algoritması tespit edilir ve ortaya çıkan kesme kuvveti dönüşümü ile birtakım kırılma ağırlık fonksiyonu elde edilir. Daha sonra kesme kuvveti sinyalleri alınarak kırılma detektörü ile hataların tespiti yapılır.

(Chekired ve ark 2011)'nın çalışmalarının amacı FPGA tümleşik devreleri kullanılarak fotovoltaik modüller için maksimum güç elde etmektir. Fotovoltaik modüller, güneş ışığını DC akıma çeviren sistemlerdir. Bulanık mantık tabanlı MPPT (Maksimum güç noktası takibi) yöntemi ile fotovoltaik sistem için değişken sıcaklık ve ışınım altında kontrol amaçlanmıştır. FPGA devreleri ile kısa geliştirme süresi, düşük maliyet ve çalışma esnekliği görülmüştür.

3. DC MOTOR

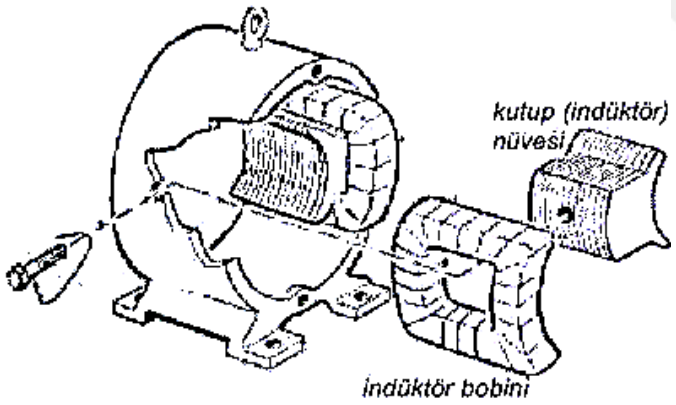
Doğru akım elektrik enerjisini hareket enerjisine dönüştüren elektrik makinalarına DC Motor adı verilir. Doğru akım makineleri dönen kısım (endüvi), duran kısım (endüktör), yatak, kapak, fırça ve kolektörden oluşur (Şekil 3.1).



Şekil 3.1. DC Motor Yapısı (MEGEP 2011)

3.1 Endüktör (Duran Kısım)

Görevi manyetik alan meydana getirmektir. Endüktör sargısı DA makinesinin gövdesinde bulunur, vida veya somunlarla gövdeye tutturulur (Şekil 3.2).



Şekil 3.2. Endüktör

Doğru akım makinesinin özelliğine göre endüktör sargısı yapısal değişiklikler gösterir. Küçük güçlü DC makinelerinde ve pilli oyuncakta daimi mıknatıs, endüktör olarak görev yapmaktadır. Doğru akım motorlarında kutup sayısı, alternatif akım makinelerinde olduğu gibi hız, indüklenen gerilim ve akımın frekansına bağlı değildir. Burada kutup sayısı makinenin gücüne ve devir sayısına göre değişir. Endüktör, makinenin gücüne (büyüklüğüne, çapına) ve devir sayısına göre 2, 4, 6, 8 veya daha çok kutuplu olur (MEGEP 2011).

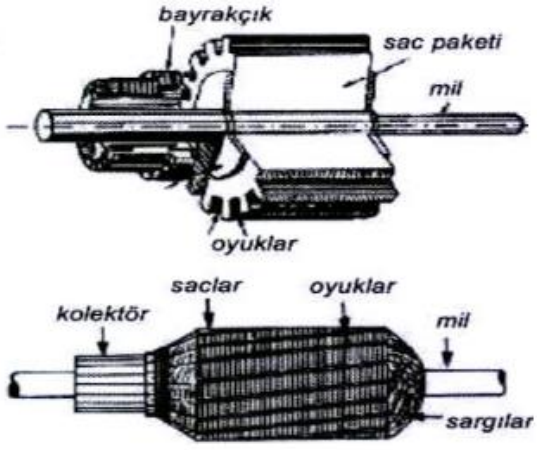
Şekil 3.3. Sabit mıknatıslı kutuplar

3.2 Endüvi (Dönen Kısım)

DA makinelerinde dönen, mekanik enerjinin alındığı kısımdır. Doğru akım makinesinin yapısına göre çeşitli ebatlarda yapılmaktadır. Endüvi üzerinde kolektör ve preslenmiş sac paket bulunur. Sac üzerindeki emaye yalıtkanlı iletkenlerden akım geçtiğinde motor olarak çalışır yani döner. Manyetik alan içindeki endüvi dışarıdan bir kuvvetle döndürülürse DA gerilim üretir yani dinamo görevi yapar(MEGEP 2011).

Şekil 3.4. Endüvi

Periyodik aralıklarla endüvinin bakımının yapılması, kollektör yüzeyinin temizlenmesi gerekir. Büyük, güçlü doğru akım makineleri yüksek akım çekmektedir. Bundan dolayı kolektöre iki ya da daha fazla fırça ile doğru gerilim uygulanır. Endüvi yapısı Şekil 3.5'te görülmektedir.



Şekil 3.5. Endüvinin yapısı

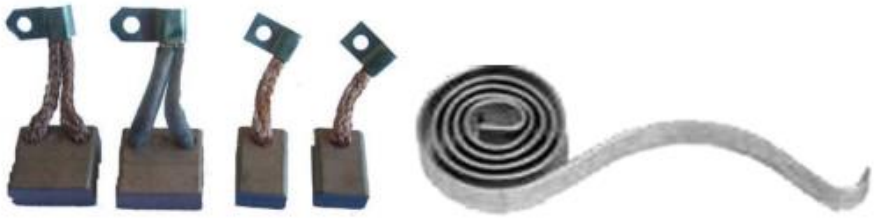
Endüvi üzerinde kolektör vardır ve bakır dilimlerden oluşur (Şekil 3.6). Endüvide bulunan iletkenler bu dilimlere lehimlenerek ya da presle bağlanır (Şekil 3.7).

Şekil 3.6. Kollektör

Şekil 3.7. Kolektöre İletken Bağlantısı

3.3. Fırça ve Fırça Yatağı:

Fırça, doğru akım makinesi motor olarak çalışıyorsa gerilim uygulanmasını sağlar. Doğru akım makinesinin özelliğine göre boyutu değişmektedir. Fırçanın kolektörlere uygun basınçla basması gereklidir. Bu nedenle doğru akım makinelerinin fırçaları üzerinde baskı yayları bulunur (Şekil 3.8) (MEGEP 2011).



Şekil 3.8. Fırçalar ve Baskı Yayı

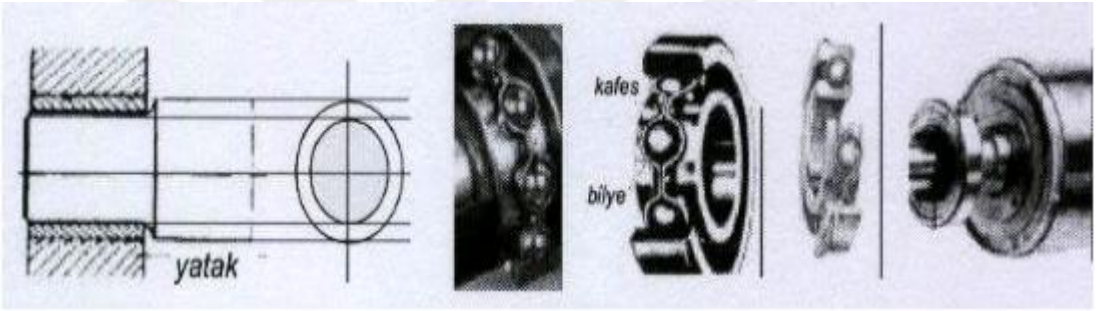
Fırça (Şekil 3.10), fırça yuvasına yerleştirilir (Şekil 3.9). Baskı yayının gevşek ya da çok sıkı olması motorun verimli çalışmasını engeller.

Şekil 3.9. Fırça Yuvası

Şekil 3.10. Baskı Yayları ve Fırçalar

3.4. Yatak Kapak ve Diğer Parçalar

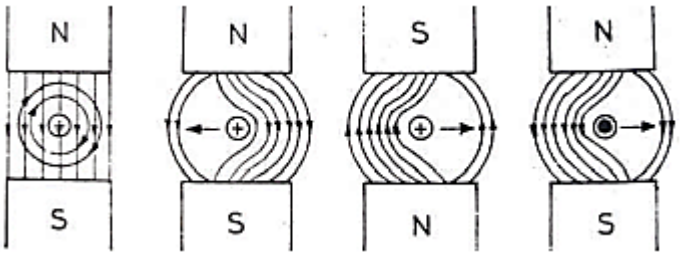
Doğru akım makinelerinin en önemli parçalarından biri de yataklarıdır. Endüvide olduğu gibi yataklar da periyodik bakım gerektirir. Bilezikli tip metal yataklar ya da bilyeli yatak kullanılır. DA makinelerinin soğutulması için çeşitli tip yapıda pervaneler kullanılır (Şekil 3.11) (MEGEP 2011).



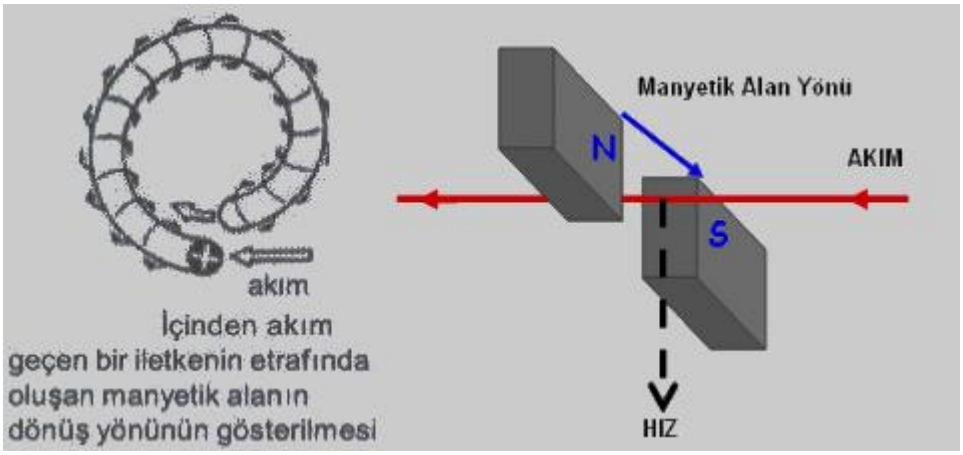
Şekil 3.11. Yatak ve Rulmanlar

3.5. DC Motorunun Çalışması

Doğru akım motoru, içinden akım geçen iletkenin manyetik ortam dışına itilmesi prensibine göre çalışır (Şekil 3.12). Motorlarda manyetik alanı endüktör oluşturmaktadır. İçinden akım geçen iletkenler ise endüvi üzerinde bulunur (MEGEP 2011).

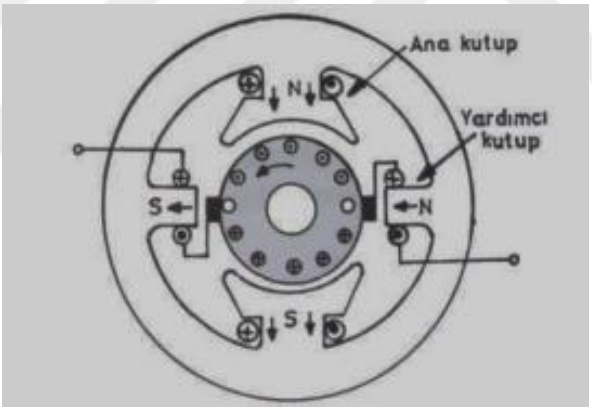


Şekil 3.12. Akım geçen iletkenin manyetik alan içindeki durumu



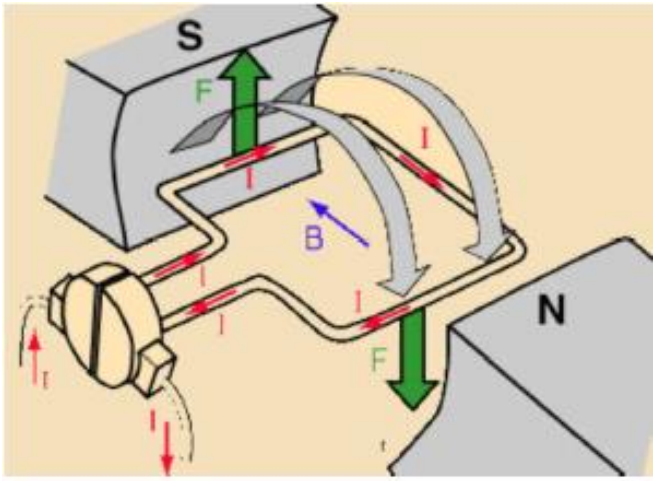
Şekil 3.13. İçerisinden akım geçen iletkenin durumu

Endüvi üzerindeki iletkenlere fırça ve kolektör yardımıyla doğru gerilim uygulanır. Böylece endüvi üzerindeki iletkenden akım geçer ve manyetik alan oluşur (Şekil 3.13). Endüviden geçen akım, manyetik alan oluşturur ve kutuplarda oluşan manyetik alanı bozar. Bu durumda endüvi reaksiyonu oluşur. Bu istenmeyen durumu ortadan kaldırmak için 8 yardımcı kutup kullanılır (Şekil 3.14). Bazı küçük güçlü motorlarda yardımcı kutup olmayabilir.



Şekil 3.14. Yardımcı kutbun endüviye bağlanması

Endüktör sargısının manyetik alanı (N–S), Endüvide üzerinde manyetik alan oluşturan iletken veya iletken demetini dışa doğru iter. Bu itilme, mil etrafında dönmeyi meydana getirir. N ve S kutupları, Endüviden geçen akım yönüne göre iletken veya iletken demetini manyetik ortamın dışına iter (Şekil 3.15). Bu itilme prensibi, doğru akım motorlarının çalışma esasını oluşturur. İletkenden geçen akım yön değiştirirse itilme yönü de değişir. İtilme yönünün değişmesi motorun dönüş yönünü de değiştirir (MEGEP 2011).

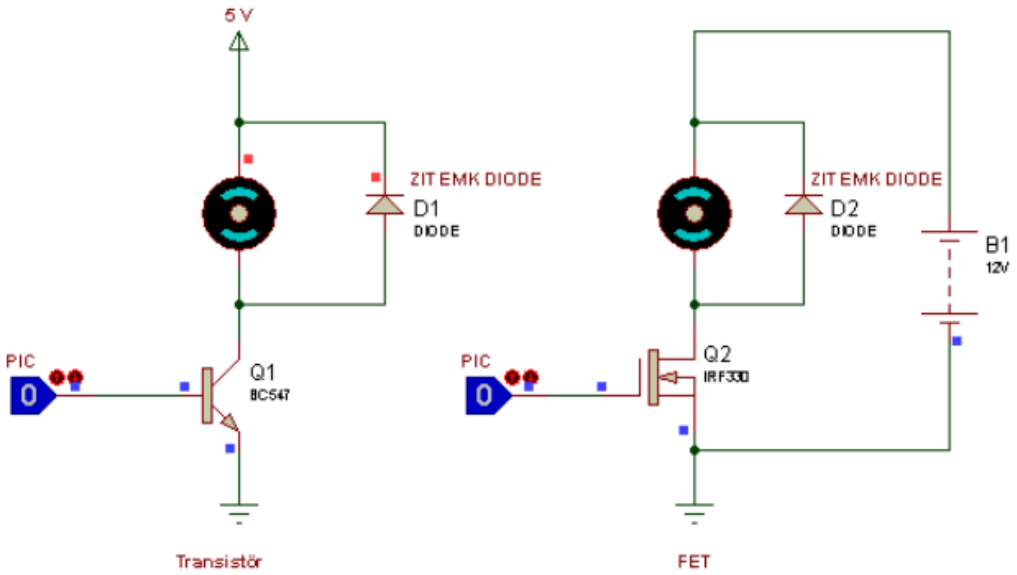


Şekil 3.15. İçinden akım geçen iletkenin manyetik alan dışına itilmesi

DC motorlarda besleme uçlarının yerleri değiştirilirse motor devir yönü değişir. Aynı zamanda Besleme gerilimi değiştirilerek motor devri ayarlanır.

4. MİKROİŞLEMCİ İLE DC MOTOR KONTROLÜ

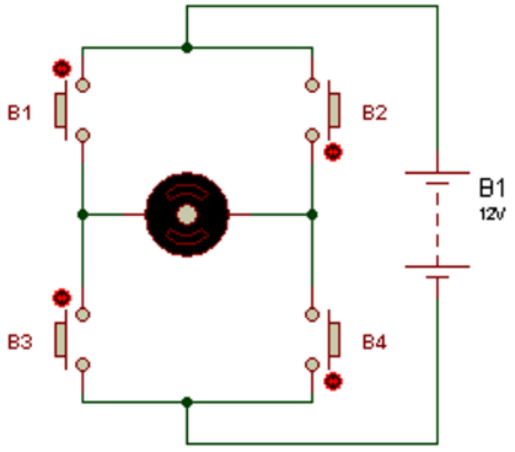
DC motorlar uçlarına uygulanan DC gerilim ile çalışırlar. Bu gerilimin değeri ve yönü değiştirilerek motorun hızı ve yönü değiştirilebilir. Gerilim değeri yükseldikçe devir sayısı artar, devir yönü için besleme uçları yer değiştirilir. Motorları çalıştırma(sürme) işlemi uygun transistör veya FET'ler ile yapılabilir (Şekil 4.1).



Şekil 4.1. DC Motor Kontrol devresi

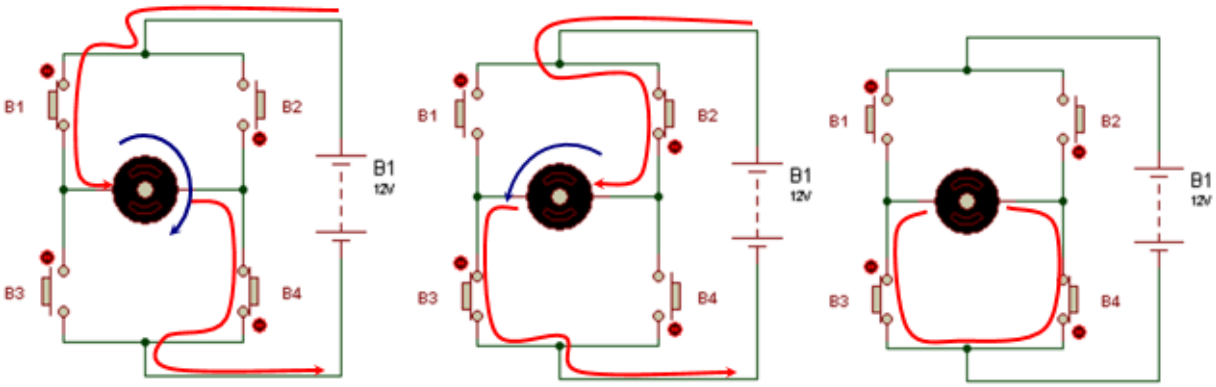
DC motorlar uçlarına uygulanan DC gerilimin değeri ve yönü değiştirilerek motorun devir sayısı ve devir yönü değiştirilebilir. DC motorun devir yönünü değiştirmek için "H köprüsü yöntemi" uygulanır (Şekil 4.2).

H köprüsü yöntemini daha iyi anlatabilmek için şekil deki devreyi inceleyelim. Bu devrede 4 adet buton ile dc motordan geçen akım yönü değiştirilerek motor devir yönü değiştirilmiş olur.



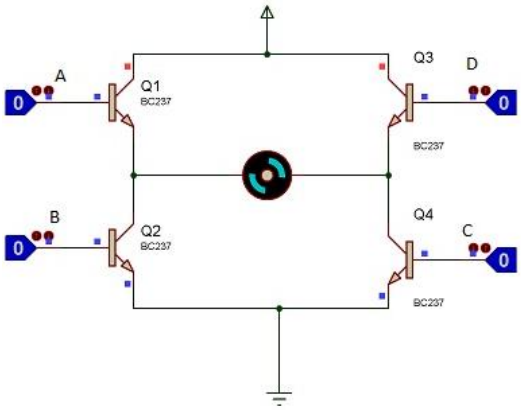
Şekil 4.2. H köprüsü

1.Buton(B1) ve 4.Buton(B4) kapatıldığında DC motordan geçen akım yönü şekil 4.3 -a daki gibi olacağından motor devir yönü saat ibresi yönünde olur. 2.Buton(B2) ve 3.Buton(B3) kapatıldığında DC motordan geçen akım yönü şekil 4.3-b deki gibi olacağından motor devir yönü saat ibresi yönünün tersinde olacaktır. İstenmeyen fakat düşük devirli DC motorlarda kullanılan bir yöntemde frenleme yöntemidir. Şekil 4.3-c deki bu yöntemde motorun her iki ucuna + potansiyel verilerek motorun frenlemesi yapılır.



Şekil 4.3. Motor Devir Yönleri

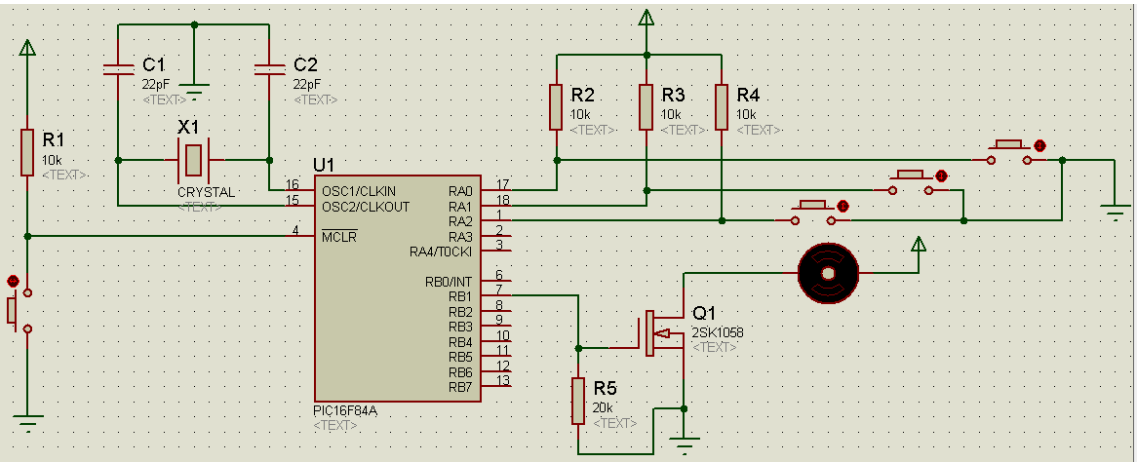
Şekil 4.4'teki devrede Q1 Transistörü ve Q4 Transistörü iletime geçtiğinde motor sağa dönüyorsa, Q2 Transistörü ve Q3 Transistörü iletime geçtiğinde sola dönecektir. Transistörlerin PIC ile tetiklenme yöntemi tamamen kullanıcıya ve donanım özelliklerine bağlıdır. Her bir transistörü PIC'in bir pini ile tetikleyebileceğiniz gibi Q1 Transistörünün ve Q4'ün Transistörünün base uçları ile Q2 Transistörünün ve Q3 Transistörünün base uçları birleştirilip, sadece 2 pin kullanma imkânı da vardır. Fakat frenleme bu şekilde mümkün olamayacaktır. Motorun çektiği akıma göre transistör seçilir. Yüksek akımda çalışılacaksa FET veya MOSFETler kullanılır.



Şekil 4.4. Transistörlü H Köprüsü

4.1.Örnek Uygulama 1

Tek yön DC motor kontrol devresi Şekil 4.5'teki gibidir.



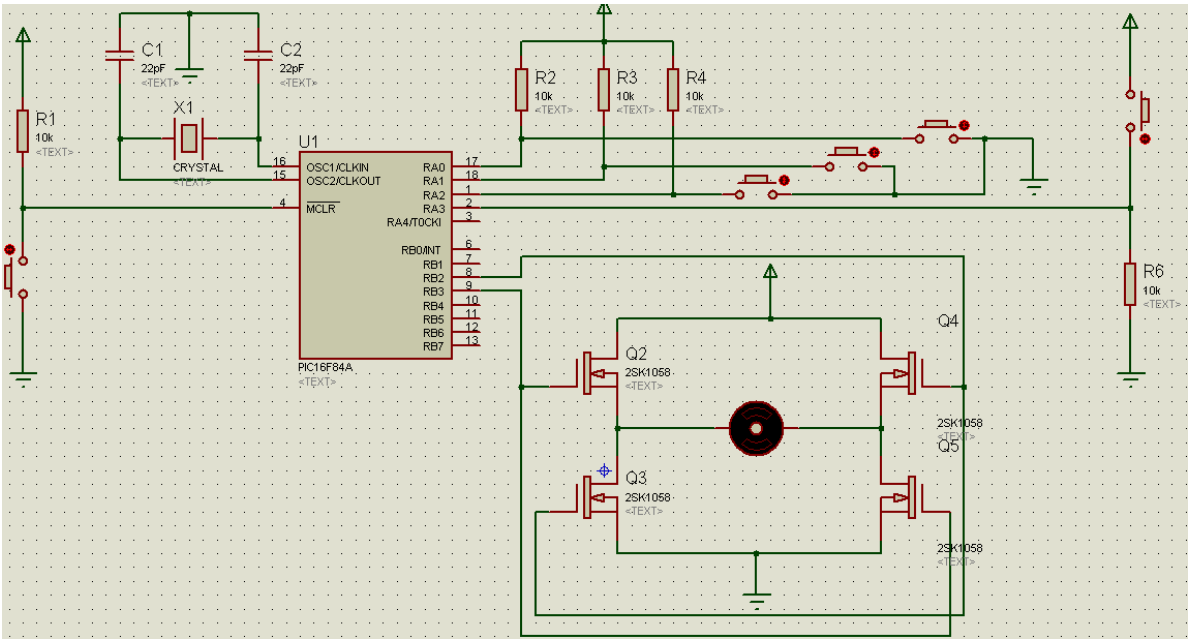
Şekil 4.5. DC Motor Kontrol devresine ait ISIS devre çizimi(Tek Yön)

Mikroişlemcinin B portunun (PORTB) 1.pinine bağlı DC motorun tek yönlü çalıştırılmasına ait programın C kodu:

```
#include<16f84a.h>
#fuses XT, NOWDT, PUT, NOPROTECT
                    #byte port_a=5        //a portunun tanımlanması
#byte port_b=6        //b portunun tanımlanması
void main()
{
        Set_tris_a(0x0f);      //a portunun giriş olarak atanması
        Set_tris_b(0x00);      //b portunun çıkış olarak atanması
        port_b=0x00;
        while(1) {        // sonsuz döngü
                if (input(PIN_A0)==0) port_b=0x02;  //motor çalışıyor
                if (input(PIN_A1)==0) port_b=0x00;  //motor duruyor
        }
}
```

4.2.Örnek Uygulama 2

Tek yön DC motor kontrol devresi Şekil 4.6'daki gibidir.



Şekil 4.6. DC Motor Kontrol devresine ait ISIS devre çizimi(Çift Yön)

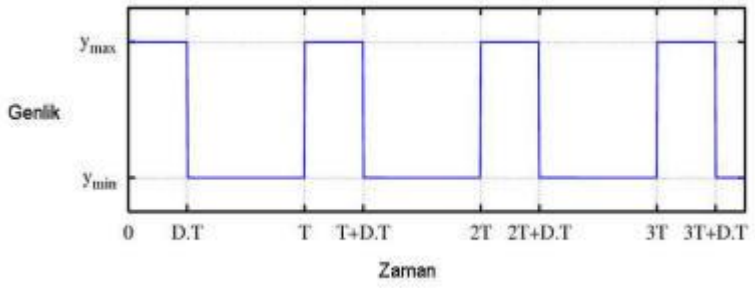
Mikroişlemcinin B portunun(PORTB) 2.pinine ve 3.pinine bir H köprüsü ile bağlı DC motorun çift yönlü çalıştırılmasına ait programın C kodu:

```
#include<16f84a.h>
#fuses XT, NOWDT, PUT, NOPROTECT
#byte port_a=5        //a portunun tanımlanması
#byte port_b=6        //b portunun tanımlanması
void main() {
        Set_tris_a(0x0f);     //a portunun giriş olarak atanması
        Set_tris_b(0x00);     //b portunun çıkış olarak atanması
        port_b=0x00;
        while(1) {       // sonsuz döngü
                if (input(PIN_A0)==0) port_b=0x04;  //motor ileri çalışıyor
                if (input(PIN_A1)==0) port_b=0x08;  //motor geri çalışıyor
                if (input(PIN_A2)==0) port_b=0x00;  //motor duruyor
  }
}
```

5. PWM (PULSE WİDTH MODULATİON)

Sinyal Genişlik Modülasyonu olan bu teknik, sinyal işleme veya sinyal aktarma gibi daha çok elektronik devrelerin yanı sıra elektrik makineleri gibi özel uygulama alanlarında da yer alan bir tekniktir. PWM (Darbe genişlik modülasyonu) , üretilecek olan darbelerin, genişliklerini kontrol ederek, çıkışta üretilmek istenen analog elektriksel değerin veya sinyalin elde edilmesi tekniğidir. Sinyal bilgisinin aktarım için uygun hale çevrilmesi amacının yanı sıra güç kontrolü sağlamak ve elektrik makineleri, güneş şarj üniteleri gibi özel devrelere destek olmak amacı da taşır (WEB1).

Üretilen kare dalga darbe sinyallerinin genişliklerinin ortalaması, çıkışta üretilecek olan analog değerin elde edilmesini sağlar (Şekil 5.1).



Şekil 5.1. Genlik-Zaman Diyagramı

Kare dalganın frekansına f(t), en düşün genlik değerine y_{min}, en yüksek genlik değerine ymax ve sinyal oranına (duty cycle) D diyelim, ortalama sinyal,

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\,dt$$

f(t) kare dalga olduğundan, f(t), y_{max} için:

$0 < t < D \cdot T$

y_{min} için:

$D \cdot T < t < T$

değerlerini alırlar.

Buradan

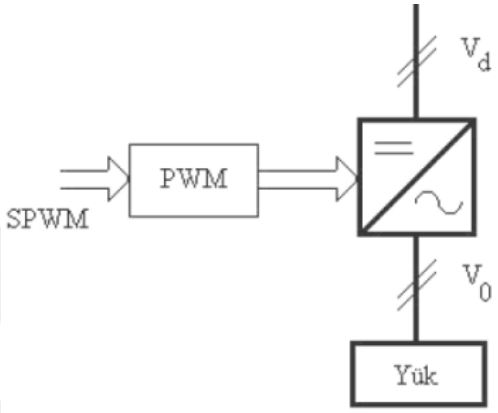
$$\begin{aligned} \bar{y} &= \frac{1}{T}\left(\int_0^{DT} y_{max}\,dt + \int_{DT}^{T} y_{min}\,dt\right) \\ &= \frac{D\cdot T\cdot y_{max} + T(1-D)y_{min}}{T} \\ &= D\cdot y_{max} + (1-D)\,y_{min} \end{aligned}$$

elde edilir.

Yukarıda verilen formül genellikle $y_{min} = 0$ iken $\bar{y} = D \cdot y_{max}$ olarak kullanılır. Görüldüğü gibi elde edilecek ortalama değer direk sinyal oranına (*duty cycle*) bağlıdır (WEB1).

5.1.Gerilim Kontrollü PWM

Literatürde gerilim kontrollü PWM açık döngü kontrol tekniği olarak adlandırılır. Açık kontrol tekniğinde bir referans giriş gerilimi alınarak sistemin sürekliliği gerçekleştirilir (Şekil 5.2). Bu teknikte alınan giriş referans gerilimi işaretin farklı bir üçgen dalga işareti ile karşılaştırılması sonucu taşıyıcı temelli PWM oluşur.

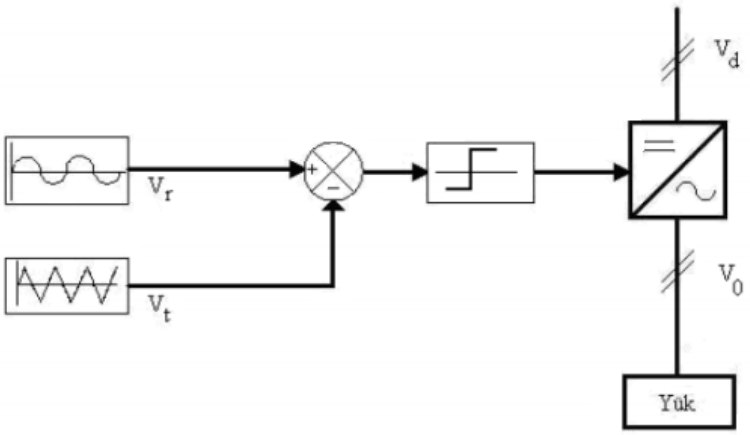


Şekil 5.2. Gerilim Kontrollü PWM blok şeması

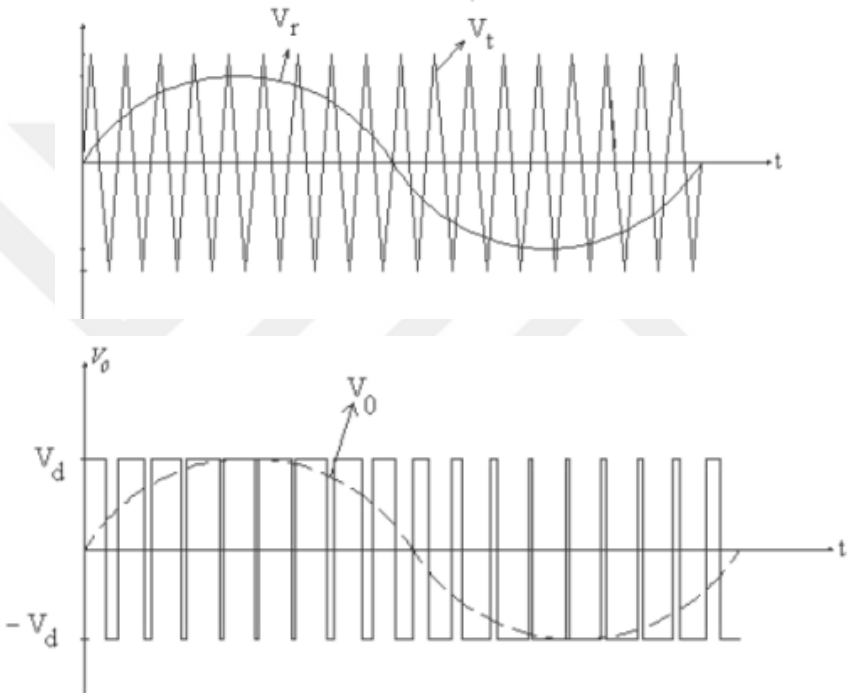
Taşıyıcı temelli sinüzoidal PWM, eviricideki yarı iletken anahtarlama elemanlarının tetikleme anlarını belirlemek ve eş zamanlamayı sağlayabilmek için sinüzoidal PWM metodu endüstriyel uygulamalarda çoğunlukla kullanılmaktadır. Şekil 5.3'te kontrol blok şemasında gösterildiği gibi evirici çıkışının gerilimini ve frekansını belirleyecek bir sinüs referans işareti, frekans ve genliği sinüs işaretinden daha büyük bir üçgen dalga işaret ile karşılaştırılır. Bu iki işaretin kesiştiği noktalarda evirici içindeki aynı koldaki anahtarlama elemanları durum değiştirir.

Evirici çıkış gerilimi ve frekansı değerinin değiştirilmesi için referans işareti (kontrol işareti) genliğinin ve frekansının değiştirmesi yeterli olacaktır. Genliği değişken sinüzoidal referans işareti, daha yüksek frekanslı üçgen dalga taşıyıcı işaretin karşılaştırılması yapılarak oluşan kesişme noktaları ile anahtarlama elemanlarının anahtarlama süreleri belirlenmektedir.

Sinüs dalgasının genliğinin yükseltilip azaltılmasıyla, çıkışta elde edilen PWM işaretinin darbe genişliklerinin değişmesi, temel bileşenin genliğinde değişme sağlar (Şekil 5.4) (MEGEP 2013).



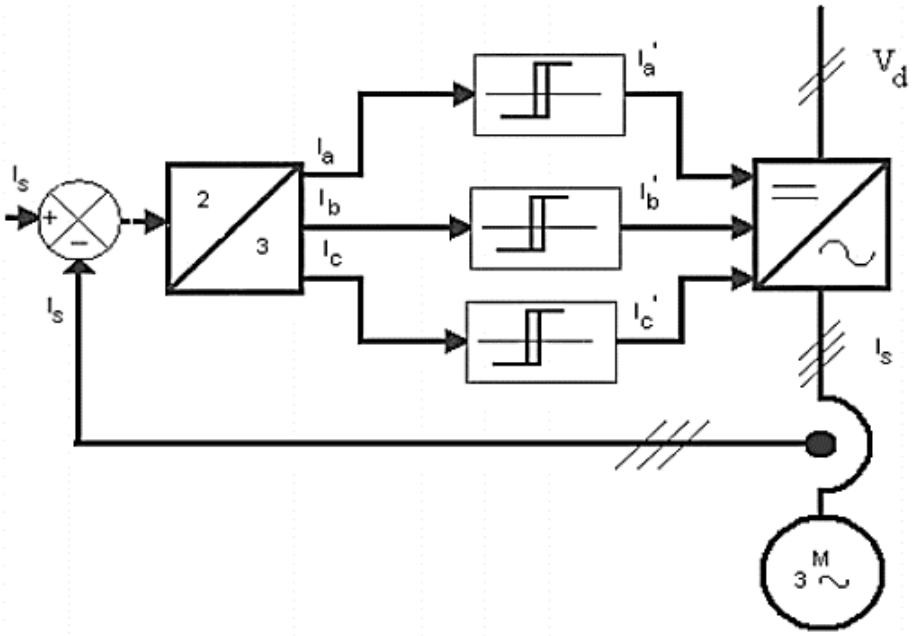
Şekil 5.3. Sinüzoidal PWM kontrol Şeması



Şekil 5.4. Çift yönlü gerilim anahtarlamalı sinüzoidal PWM

5.2. Akım Kontrollü PWM

Akım kontrollü PWM eviriciler, yüksek performanslı AC sürücülerinde yaygın olarak kullanılmaktadır. Bu kontrol metodu aynı zamanda kapalı döngü PWM kontrol tekniği diye de adlandırılır. Şekil 5.5'te gerçekleştirilen kontrol, stator vektör akımı için ya da stator akı vektörü için bir geri döngü kontrol yapılarak gerçekleştirilir. Akım kontrollü PWM evirici, geleneksel gerilim kaynaklı PWM eviriciye kontrol çıkış akımı sağlamak amacıyla akım düzenleme (ayarlama) döngülerinin eklenmesi ile oluşturulur. Bu kontrol tekniği lineer olmayan yükler için yeterince hızlıdır.



Şekil 5.5. Akım kontrollü PWM tekniği ile çalışan eviricinin blok şeması

Akım kontrolü değişik formlarda olabilir. Genellikle sinüsoidal bir referans akım dalga şekli üretilir ve bu dalga motorun gerçek ölçülen akımı ile birlikte karşılaştırıcıyı besler. Eğer motorun faz akımı, referans akım değerinden daha pozitif ise üst kısımdaki elemanlar kesime, alt kısımdaki elemanlar iletime geçerek motorun akımının azalması sonucunu doğurur. Karşılaştırıcı ölü bant (histerisiz) genişliği vardır. Bu bant eviricinin anahtarlamaya başlamadan önceki izin verilen referans akım ile gerçek akımı arasındaki farkı belirler. Böylece gerçek akım, referans akımı, faz gecikmesi ve önemli hata büyüklüğü olmadan izler. Üç fazlı sistemlerde, genellikle her faz için ayrı akım kontrolü vardır (MEGEP 2013).

Microchip firmasına ait 16F877 mikro denetleyicisi dâhili PWM modülüne sahiptir. Bu modül Capture, Compare, PWM (CCP) modülü olarak adlandırılmıştır. Bu modül ile maksimum 8bit çözünürlüklü PWM sinyalleri elde edilebilir.

6. BULANIK MANTIK

Dünyadaki bazı olayları açıklamak için kesin tanımlamalarda bulunabilmek imkânsızdır ve olaylar çoğu kere belirsizlikler ve doğrusal olmama özellikleri taşır. Cismin ısısını kaybetmesi, kapasitörün şarj veya deşarj olayı bu doğrusal olmama özelliklerine birer örnektir. Belli bir miktar uranyumun bozulması esnasında hangi atomun ne zaman bozulacağının bilinmemesi de belirsizlik taşıyan bir olaydır. Bu nedenle eşya ve olaylar bulanıklık perspektifinde ele alındıkça, çok daha doğru ve verimli sonuçlar elde edilebilir. Bulanık mantık, bu yaklaşım için kullanılabilecek oldukça tesirli bir mantık anlayışıdır (Günal 1997).

Terimler ya da ölçüler kesin olarak tanımlanıp ölçülemediğinden dolayı insanlar çoğu zaman belirsiz (kesin olmayan) ifadeler kullanırlar. İşte bulanık mantık bazı sorulara basitçe evet-hayır cevabı verilemeyen durumları kapsar. Bulanıklığın ve bulanık mantığın temeli de budur.

Bulanık mantık, klasik mantık sistemlerinden ziyade insan düşüncesi ve tabii dil ruhuna daha yakındır. Temel olarak, gerçek dünyanın eksik ve yaklaşık özelliğini yakalayan etkili bir araç sağlar.

Matematiksel model ve ölçülen değerlerin yansıra insan düşüncesini de mühendislik sistemine katmak üzere insan düşüncesini formüle eder. Günlük konuşma dilini kullanan bulanık mantık, dilsel değişkenler (*linquistic variables*) yardımıyla biraz sıcak, ılık, uzun, çok uzun, soğuk gibi günlük hayatımızda kullandığımız kelimeler yardımıyla insan mantığına en yakın doğrulukta denetimi sağlayabilir. Bulanık mantık denetleyici kullanılarak elektrikli ev aletlerinden oto elektroniğine, gündelik kullandığımız iş makinelerinden üretim mühendisliğine, endüstriyel denetim teknolojilerinden otomasyona kadar aklımıza gelecek her yerde kendisine uygulama alanı bulabilir.

İkili mantık, iki ayrık değer alabilen değişkenleri ve mantıksal anlam taşıyan işlemleri ele alır. Değişkenlerin alabileceği iki değer farklı şekillerde adlandırılabilir(örneğin doğru ve yanlış, evet ve hayır, vs.), burada her değişken ancak ve ancak olası iki ayrı değerden birini alabilir: 1 ve 0 (Mano 2010).

Bulanık mantık; ikili mantık sistemine karşı geliştirilen ve günlük hayatta kullandığımız değişkenlere üyelik dereceleri atayarak, olayların hangi oranlarda gerçekleştiğini belirleyen çoklu mantık sistemidir.

Bulanıklık, çoklu değerlilik (*multi – valued*) demektir. İkili mantığın 0-1 önermelerine karşın bulanıklık, üç veya daha fazla, belki de sonsuz sayıda önermeler yapar. Yani bu mantıkta küme üyeleri derecelendirilebilir. Başka bir değişle siyah ile

beyaz arasında yer alan sonsuz sayıda gri tonlarını içermektedir. Örneğin uzaklıkla ilgili bir problemde mesafenin yalnızca yakın ya da uzak olduğunu belirtmekle kalmayıp ne kadar yakın ya da ne kadar uzak olduğunu da belirtir. Karar vericiler hangi şartlarda ve boyutlarda karar verirlerse versinler, bir belirsizlik ortamı içinde bu işlevlerini yerine getirmek zorundadırlar. Verilen kararların doğruluğu ise, söz konusu belirsizliğin riske dönüştürülebildiği ölçüde sağlanacaktır. Ancak karar vericiler karar sürecinde klasik bilimsel yaklaşım ve bu yaklaşımın içerdiği yöntemleri kullanıyorlarsa, sonuçta verilen kararlar, iyi – kötü, güzel – çirkin, doğru – yanlış, evet – hayır, siyah – beyaz ya da 0 – 1 gibi yönlü kararlar olacaktır. Oysa gerçek yaşam mutlak ayrım üzerine kurulu değildir. Diğer bir deyişle karar ortamlarında mutlak siyah ve mutlak beyazın yanında binlerce gri tonunun varlığı unutulmamalıdır (Günal 1997).

Bulanık mantığın gücü basit şeyleri basit tutmaktır. Klasik mantık bizleri çok katı sınırlar çizmeye zorlar. Mesela batı edebiyatında "novel" denilen roman, 90 veya daha fazla sayfadan , "novella" ise 90'dan daha az sayfadan oluşur. Bu standarda göre 91 sayfalık bir eser, roman, olurken, 89 sayfalık bir çalışma "*novella*" (uzun hikâye) olur. Eğer bir bilgisayarda kelimelerin puntosu büyütülürse uzun hikâye, roman haline gelebilir. Bulanık mantık bu tür saçmalıkları önler.

Klasik mantıkta büyüklük-küçüklük, uzunluk-kısalık gibi kavramların kesin sınırları vardır. Diyelim ki uzun insanların alt sınırı 1.70 m'dir. Klasik mantığa, "Ali uzun mudur?" sorusu sorulursa, bu sınıra bakıp, eğer Ali'nin boyu 1.70 m'in üzerinde ise Ali uzun, 1.69 m ise kısadır. Hâlbuki bulanık mantık, Ali'nin ne kadar uzun olduğunu sorar. Klasik mantık gibi uzuna 1, kısaya 0 gibi katı(kesin) değerler vermez. 0.1, 0.2, 0.3… gibi daha hassas ve esnek değerler verir. Böylelikle 1.69 m boyundaki bir insana kısa (0) demez, 0,2 gibi bir uzunluktadır der.

Üzüm suyunun şaraba dönüşme sürecini ele alalım. Üzüm suyu şaraba dönüştürülürken, arada şıra vs. gibi formlardan geçmektedir. Üzüm suyunun şıradan sonra alacağı form, şaraba daha yakın olacaktır ve şarabın nerede başlayıp başlamadığı bu bulanıklıktan tam bilinemez.

Renklerin birbirinden ayırt edilmesinde de aynı güçlük vardır. Sarı ile açık sarı veya turuncu arasındaki sınır nereden geçer? Bu sınırın net bir yeri yoktur.

Tabiî bulanık mantığında belli sınırları vardır ve bu sınırlar makama, ele alınan eleman ve şartlara göre değişirler. Onu klasik mantıktan ayıran nokta bu sınırların daha esnek olmasıdır. İşte bu esneklik sayesinde bulanık mantık tatbik edildiği her sahada çok daha hassas sonuçlar doğurmaktadır.

Bu iki mantık arasındaki temel farklar Çizelge 6.1'de verilmiştir.

Çizelge 6.1. Klasik mantık-Bulanık mantık arasındaki temel farklılıklar

Klasik Mantık	Bulanık Mantık
A veya A değil	A ve A değil
Kesin	Kısmi
Hepsi veya Hiçbiri	Belli Derecelerde
0 veya 1	0 ve 1 Arasında Sürekli
Dijital Bilgisayar	Nöral Ağ {Beyin}
Fortran, Pascal v.s.	Türkçe { Doğal Dil }
İkili Birimler {bit }	Bulanık Birimler {fit}

6.1.Bulanık Mantığın Etkileri

Bulanık Mantık, makineleri "daha zeki" yapmış ve birçok ürünün ve üretim sürecinin makine IQ' su (Zekâ seviyesi) bu sayede artmıştır. Bu makineler arasında fotoğraf makineleri, kameralar, televizyonlar, mikro dalga fırınlar, çamaşır makineleri, elektrikli süpürgeler, otomatik şanzımanlar, motor kontrolü, metro denetim mekanizmaları, asansörler ve mikro devreler sıralanabilir.

Bulanık teori her bir kelimenin anlamında saklı olan belirsizliği temsil eden teoridir. Bu teorinin bir uygulaması olarak "Bulanık Yapay Zekâ" nın gelecekte insanlar ile bilgisayarlar arasında kurulacak olan yakın ilişkide büyük bir rol oynayacağı beklenmektedir.

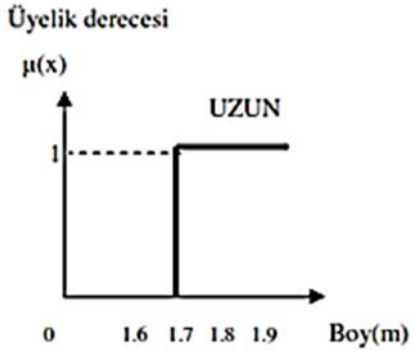
Pilav pişirme aletlerinden asansörlere, arabaların motor ve süspansiyon sistemlerinden nükleer reaktörlerdeki soğutma ünitelerine, klimalardan elektrikli süpürgelere kadar bulanık mantığın uygulandığı birçok saha mevcuttur. Bu sahalarda temin ettiği enerji, iş gücü ve zaman tasarrufu ise, onun "iktisat" adına ne kadar çok önem verilmesi gereken bir sistem olduğunu göstermektedir. Bulanık mantığın gelecekteki uygulama sahaları, daha da genişleyecek gibi gözükmektedir.

Şeker hastaları için vücuttaki insülin miktarını ayarlayarak suni bir pankreas görevi yapan minik yapıların imalinde, prematüre doğumlarda bebeğin ihtiyaç duyduğu ortamı devam ettiren sistemlerin hazırlanmasında, suların klorlanmasında, kalp pillerinin üretiminde, oda içindeki ışığın miktarının ayarlanmasında ve bilgisayar sistemlerinin soğutulmasında bulanık mantık çok şeyler vaad etmektedir (Çobanoğlu 2000).

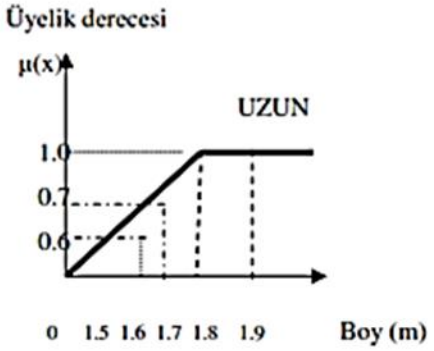
6.2.Bulanık Kümeler (*Fuzzy Sets*)

Bulanık küme kavramında klasik kümelerdeki elemandır (Şekil 6.1) veya değildir ifadesi yerine şu kadar elemandır ya da şu kadar eleman değildir ifadeleri yer alır. Bir

eleman için eleman olma durumu 1 ve olmama durumu 0 ile değil 0 ile 1 arasındaki üyelik derecesi ile gösterilir (Şekil 6.2). Örneğin "Uzun boylu kime denir?" sorusuna cevap verecek olan bir UZUN alt kümesini her iki mantığa göre tanımlayalım. Şekilde görüldüğü gibi klasik küme mantığına göre 160 cm. boyundaki bir kişi uzun boylu insanlar kümesi içinde değildir. Hatta 169 cm. boyundaki bir kişi uzun boylu insanlar kümesi içinde değildir. Oysa bulanık mantığa göre 160 cm. boyundaki kişiye kısa denilmez. Çünkü kısmen de olsa uzun boylu insanlar kümesi içindedir. Bulanık mantıkta 160 cm boyundaki biri 0,6 üyelik derecesiyle, 170 cm. boyundaki biri 0,7 üyelik derecesiyle, 180 cm. boyundaki biri de 1,0 üyelik derecesiyle uzun boylu olabilir (Çobanoğlu 2000).



Şekil 6.1. Klasik Küme



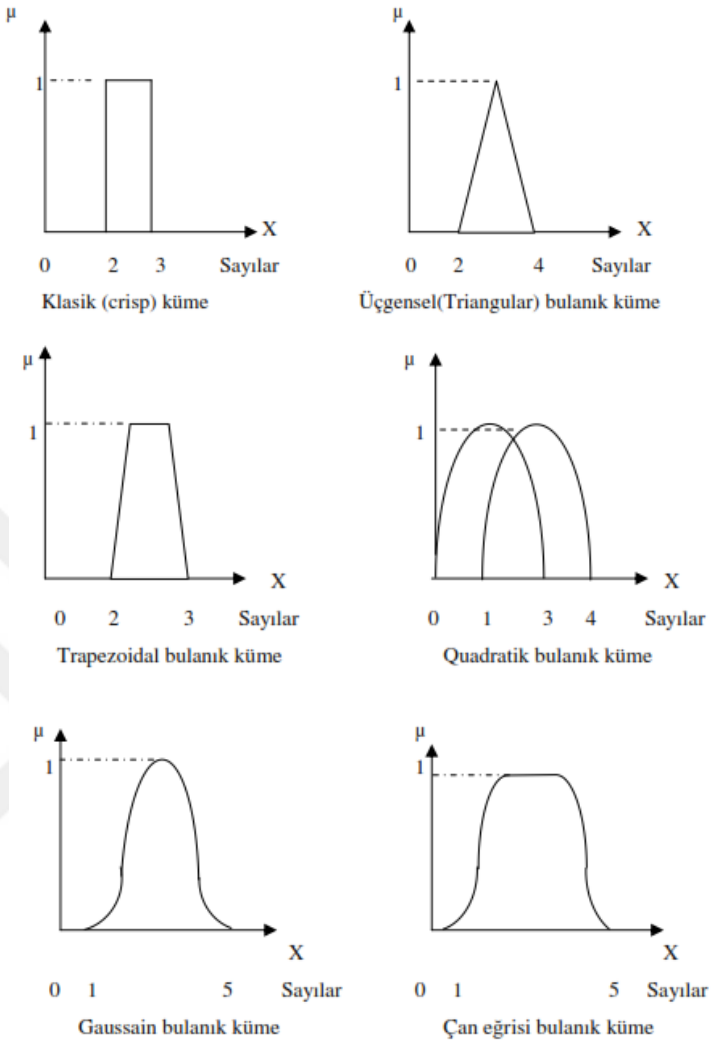
Şekil 6.2. Bulanık Küme

Bunun gibi bir insanın uzun boylu olması, bulanık küme mantığında derecelerine ayrılabilmektedir. Büyük üyelik dereceleri az bulanık kabul edilirken, küçük üyelik dereceleri daha bulanık olarak kabul edilir.

6.3.Üyelik Fonksiyonları (*Membership Functions*)

Uygulamaların birçoğunda üyelik fonksiyonu, örnekte verilen UZUN gibi basit bir şekilde olmayacaktır. Üyelik fonksiyonlarının alabileceği muhtemel temel şekiller Şekil 6.3'te gösterilmiştir. Konuşma dilinde kullanılan her bir nitelikli tanımlamalar bir

üyelik fonksiyonu olarak yazılırlar. Her noktada ve uygulanan sınırlarda üyelik sınıfları belirlenir.



Şekil 6.3. Üyelik fonksiyonlarının alabildiği değişik şekiller

Bulanık mantıkta, dilsel ifade kolaylığı sağlayacak bölgelerin sınırlarını belirtmede ve algılayıcı bilgilerine (gerçek bilgiler) ait üyelik ağırlıklarının tespit edilmesinde kullanılmak üzere uygun üyelik fonksiyonlarının belirlenmesi gerekir.

Üyelik fonksiyonları, sistem parametrelerini tanımlar. Üyelik fonksiyonlarının sayısına ve şekline ait hiçbir kısıtlama yoktur. Tamamıyla tasarımcının istek ve tecrübesine bağlıdır. Bu zamana kadar olan çalışmalarda en çok üçgen, yamuk, çan eğrisi şeklinde üyelik fonksiyonları kullanıldığı görülmektedir.

6.4.Bulanık Kontrol (Denetim)

Günümüzde kontrol teknolojisi geleneksel kontrol tekniklerinden matematik modellere dayanan kontrol teknikleri ve bilgi tabanlı zeki kontrol tekniklerine doğru hızla yol almaktadır.

Bulanık sistemler bir süreci modellemek için kullanılırsa; bu modele dayanarak tasarlanan kontrolörler bulanık kontrolörlerdir.

Bulanık kontrolün temel avantajları sistemin matematiksel formülasyona ihtiyaç duymaması, tam olmayan eksik nesnelerin tanımlanması ve çok amaçlı kontrolün başarılmasında kullanılan dilsel değişkenler ve yaklaşık çıkarsama.

Klasik kontrol teorisinde sistemin yapısını açıklayan bilgiler, kesin değerler halinde verilir. Kontrol stratejisinin temelini; sisteme ait bilgilerle sistem değişkenleri arasında ilişkiler oluşturur. Klasik kontrol, sürecin matematiksel bir modeli ile başlar ve kontrolör de bu modele göre tasarlanır.

Bulanık kontrol ise bir insanın uzmanlığına (IF – THEN kuralları yapısında) ya da gözlemlerine dayanır ve kontrolör bu kurallar sentezinden yola çıkılarak tasarlanır. Klasik ve modern kontrol teorisinde olduğu gibi kesin ve tam matematik modellere ihtiyaç duymaz. Denetlemesi zor olan karmaşık süreçlerde (çimento ocakları, çelik fırınları, çöp işleme fabrikaları gibi), bulanık kontrolü kullanmak zorunlu hale gelmektedir. “Bulanık mantık kullanarak fiziksel sistemlerin kontrolünü yapmak istersek önümüze dört seçenek çıkar.

- Bir mikrodenetleyici ile bir çıkarım işlemcisini kaskat bağlayıp beraber çalıştırmak,
- DSP (“Designers of digital signal processing”) ve yazılım destekli denetleyici kullanmak,
- Bilgisayar temelli uygulamalarda ise; kural tabanı, veri tabanı, bulandırıcı, çıkarım ünitesi ve durulayıcı (netleyici) olarak yazılım kullanmak ve paralel iletişimle denetleyici tasarlamak,
- İçinde RAM, EEPROM, I/O birimlerinin yansıra bulandırıcı, çıkarım motoru ve durulayıcı bölümleri de bulunduran tek entegre şeklinde bulanık işlemciler kullanarak fiziksel sistemlerin pratikte denetimini sağlamak mümkündür”

Bir Bulanık Mantık Denetleyicisi’nin tasarımında, bilinmesi gereken temel Faktörler şunlardır:

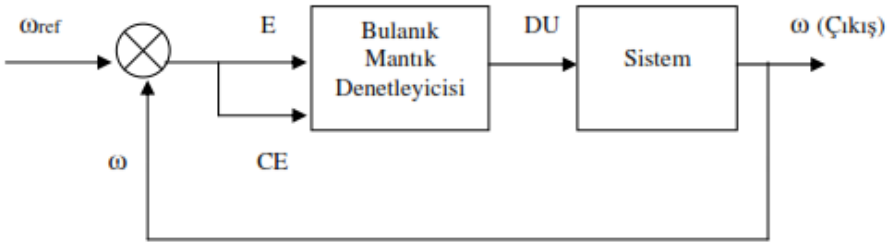
- Gerçek giriş ve çıkışlar ve bunların evrensel kümeleri, yani her bir değişkenin alması muhtemel değerler aralığı.
- Giriş ve çıkış değişkenlerinin ölçekleme faktörleri.
- Her bir giriş ve çıkış değişkenleri için bulanık değerlerin kurulmasında kullanılacak bulanık üyelik fonksiyonları.
- Bulanık kontrol kuralları tabanı.

Üyelik fonksiyonlarının ve bulanık kontrol kurallarının belirlenmesi bir Bulanık Mantık Denetleyicisi tasarımın anahtar konusudur. Tasarım amacına bağlı olarak, bir Bulanık

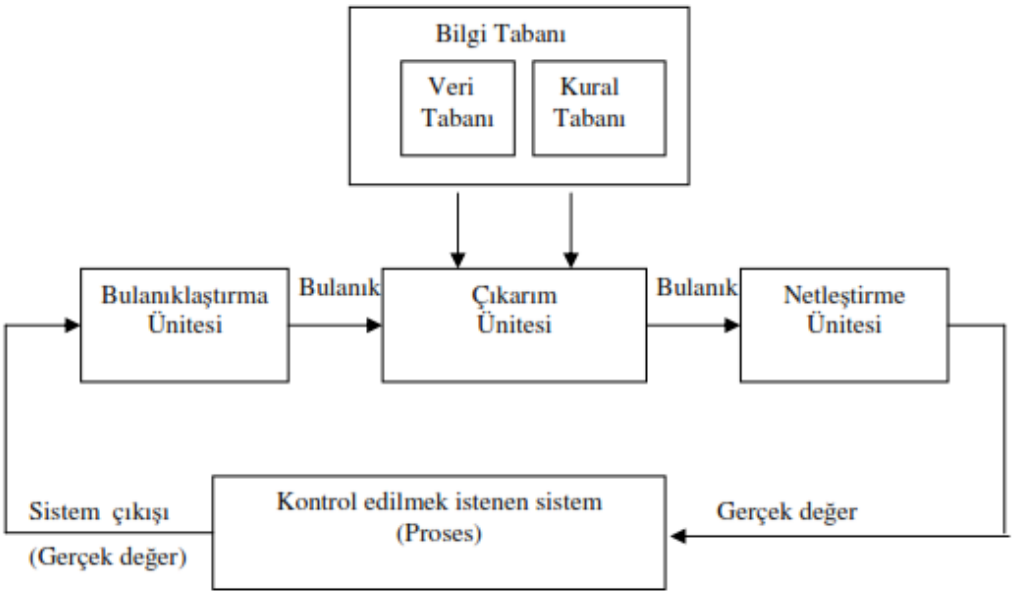
Mantık Denetleyicisi aynı zamanda kendi kendini organize etme veya öğrenme kabiliyetine sahip olabilmektedir.

6.4.1. Bulanık mantık denetleyici

Bir Bulanık Mantık Denetleyicisi tipik olarak aşağıdaki şekil de görüldüğü gibi kapalı halka kontrol sistemi şeklindedir. Sistem değişkenleri denetlenen sistemden ölçülen giriş değişkenleri hata (E) ve hatadaki değişim (CE) ve süreci kontrol etmek için Bulanık Mantık Denetleyicisi tarafından kullanılan çıkış değişkenleri (DU) olmak üzere iki ana çeşittir. Kuralların ifadesinde kullanılan her bir sistem değişkeni için izin verilen değerler uygun evrensel kümede bulanık değerler olarak tanımlanmaktadırlar (Şekil 6.4). Bu kümelerin tanımlanmaları tasarım işleminde en kritik adımlardan biridir ve sistem performansını doğrudan etkilemektedir.



Şekil 6.4. Bulanık mantık denetleyicili kapalı halka kontrol sistemi



Şekil 6.5. Bulanık Mantık Denetleyicinin Temel Yapısı

Bulanık mantık denetleyicinin dayandığı temel nokta, uzman bir sistem operatörünün bilgi, deneyim, sezgi ve denetim stratejisini, denetleyici tasarımında bilgi tabanı olarak oluşturmaktır. Denetleme işlemleri, karmaşık ve klasik denetim

algoritmalarıyla değil de bilgi ve deneyime dayanan sözel kurallarla gerçekleştirilir. Örneğin uzman sistem için gerekli denetim davranışlarını küçük, hızlı, yavaş gibi sözel bulanık terimlerini içeren komut kümesi ile ifade temsil eder. Bu komut kümeleri "Eğer – İse" kuralları yardımıyla oluşturulur(Zadeh 1988).

Şekil de görüldüğü gibi bulanık mantık denetleyici dört temel bileşenden oluşur. Bunlar; Bulanıklaştırma ünitesi, bilgi tabanı, çıkarım (karar verme) ünitesi, netleştirme (bulanıklığı giderme). Bu bileşenlerin her biri de aşağıdaki fonksiyonlardan oluşur.

1.Bulanıklaştırma (fuzzifier):

a). Giriş değişkenlerinin değerlerini ölçme

b). Giriş değişkenlerinin değerler aralığını ilgili örnek uzayına taşıyan ölçekli şekli, haritayı oluşturma.

c). Bulanık kümelerin işareti olarak gözlenebilen giriş verilerini uygun dilbilimsel değerlere dönüştüren bulanıklaştırma fonksiyonunu oluşturma.

2. Bilgi tabanı; uygulamaya ait tanım aralığı ve buna ait kontrol hedefleri bilgilerinden oluşur. "Veri tabanı" ve "Kural tabanı" ndan oluşur (Zadeh 1988).

a). Veri tabanı, dilbilimsel denetim kurallarını ve BMK (FLC) deki bulanık veri kullanmayı tanımlamada kullanılan gerekli tanımları içerir.

b). Kural tabanı, bulanık şart cümlelerinin tamamını içerir. Denetim amaçlarına uygun dilsel denetim kuralları burada bulunur ve çıkarım ünitesine buradan verilir.

3. Çıkarım (Karar verme mantığı) Ünitesi, bulanık mantık kontrol (BMK) 'un esasıdır. Bulanık kavramlara dayanan insani karar vermeyi simüle etme ve bulanık kapalı ifade ile işlem yapan bulanık kontrol faaliyetlerinde çıkarsama yapma kabiliyetine ve bulanık mantıkta çıkarsama kurallarına sahiptir.

4. Netleştirme (Durulaştırma) fonksiyonları,

a). Çıkış değişkenlerinin değerler aralığını ilgili örnek uzaylarına dönüştüren bir ölçek (derece) haritalama (scale mapping).

b). Çıkarsama yapılmış bulanık kontrol faaliyetinden bulanık olmayan kontrol faaliyetini sonuçlandıran bulanıksızlaştırma (Lee 1990).

7. FPGA

FPGA (Field Programmable Gate Array - Alanda Programlanabilir Kapı Dizileri), programlanabilir mantık blokları ve bu bloklar arasındaki ara bağlantılardan oluşan ve geniş uygulama alanlarına sahip olan sayısal tümleşik devrelerdir. Tasarımcının ihtiyaç duyduğu mantık işlevlerini gerçekleştirme amacına yönelik olarak üretilmiştir. Dolayısıyla her bir mantık bloğunun işlevi kullanıcı tarafından düzenlenebilmektedir. FPGA ile temel mantık kapılarının ve yapısı daha karmaşık olan devre elemanlarının işlevselliği artırılmaktadır. Alanda programlanabilir ismi verilmesinin nedeni, mantık bloklarının ve ara bağlantıların imalat sürecinden sonra programlanabilmesidir(WEB2).

Girişteki tanımda da belirttiğimiz gibi FGPA bir tür entegredir fakat onu diğer entegrelerden ayıran yanı donanım yapısını yani iç konfigürasyonunu istediğimiz gibi değiştirebilmemizdir. Bunu şöyle açıklayabiliriz: Donanım-programlanabilir olmayan Standart entegrelerde, transistörler arasındaki bağlantılar sabittir ve (entegre yanmadığı yada başına başka talihsiz bir olay gelmediği sürece) değişmezler.

FPGA'i, içindeki transistörleri birbirinden bağımsız ve serbest olarak üretilmiş ham bir entegre olarak düşünebiliriz. Bizim belirlediğimiz fonksiyona göre FPGA içindeki transistörler birbirlerine bağlanır ve bu sayede istediğimiz fonksiyonu gerçekleştirir. Yani teorik olarak transistör kapasitesi dâhilinde aklımıza gelen herhangi bir entegrenin yaptığı işi FPGA ile yapabiliriz.

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

Sıradan entegreler ya hiç paralel işlem yapamazlar ya da çok sınırlı yapabilirler. FPGA'de ise uygulamaya ve kapasiteye göre, birbirine paralel onlarca belki binlerce işlemi aynı anda yapabiliriz. Bu da paralel işlem gerektiren uygulamalarda FPGA'leri eşsiz kılmaktadır(WEB2).

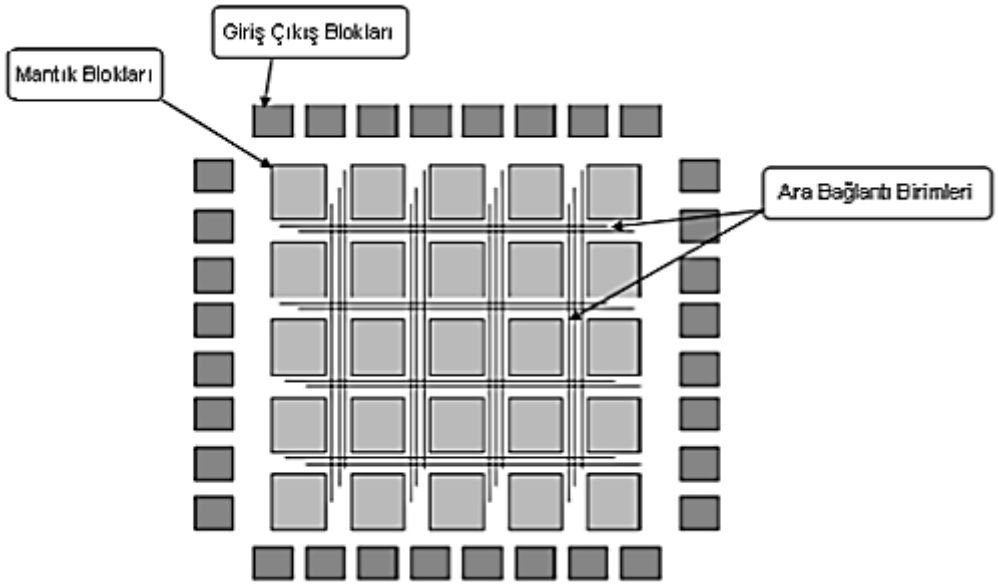
Örneğin gerçek zamanlı yüksek çözünürlüklü bir video görüntüsü üzerinde filtreleme işlemi yapmak istiyoruz. Video aslında peş peşe sıralanan resimlerdir ve bu resimlerin her birine "frame" denilmektedir. En basit haliyle bunun için videonun bir resim karesini giriş portlarından almamız, onu filtrelememiz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

işlemini yaparken ikinci resmi almaya başlarız. İlk resmi gönderirken ikinci resmi filtrelemeye ve üçüncü resmi almaya başlarız. Bunun yanında, filtreleme işlemi genel olarak yoğun çarpım gerektirmektedir. Standart bir işlemci ile bu çarpma işlemlerini de sırayla yapmak zorundayız. Oysaki FPGA ile bu işlemleri de paralel olarak yani çok hızlı bir şekilde yapabiliriz.

Özet olarak, FPGA'ler bize paralel işlem kabiliyeti sunan ve içyapısını istediğimiz fonksiyon ve uygulamaya göre değiştirebildiğimiz donanım-programlanabilir entegrelerdir (WEB3).

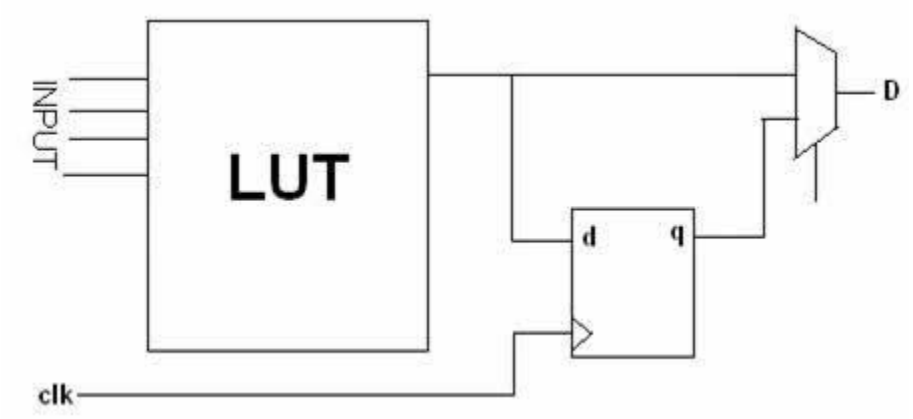
7.1.FPGA Yapısı

FPGA temel olarak Mantık Hücreleri (Logic Cell), Giriş/Çıkış Blokları (IO Block) ve Ara bağlantılardan oluşur (Şekil 7.1).



Şekil 7.1. FPGA yapısı

FPGA'in ana yapısını Mantık Hücreleri oluşturur. Bir Logic-Cell 1 adet Lookup Table (LUT), 1adet D Flip-Flop ve bir adet 2 to 1 Mux' tan oluşur (Şekil 7.2).



Şekil 7.2. Logic Cell (Mantık Hücresi)

LUT 'lar aslında bir mantık işlemi yerine getiren küçük belleklerdir (RAM). N girişli bir LUT, 2 eN ’li bir belleğe işaret eder. Binlerce Mantık Hücreleri ‘nin birleşimi sonucunca karmaşık ve büyük programlar oluşturulur.

Mantık hücrelerinin ara bağlantıları matris şeklindeki veri yolları ve programlanabilir anahtarlarla (FPGA yüklenen programa göre) sağlanır. FPGA tasarımı, her bir mantık hücresinin uygulayacağı fonksiyonu ve programlanabilir anahtarların durumunu (açık/kapalı) belirleyerek bu mantık hücreleri arasındaki bağlantıları tanımlar (WEB3).

7.1.1 FPGA Pinleri

FPGA pinleri genel olarak 2 kategoriye ayrılır.

- Ayrılmış pinler (Dedicated pins)
- Kullanıcı pinleri (User pins)

a) Ayrılmış pinler:

Bir FPGA'de tüm pinlerin %20 ila %30'u ayrılmış pindir. Bu pinler, FPGA'de gerçekleştirdikleri özel fonksiyonlara göre 3'e ayrılır.

Güç Pinleri: FPGA için gerekli olan güç ve ground (Toprak) sağlayan pinlerdir.

Konfigürasyon Pinleri: Oluşturulan programın FPGA yüklenmesi (download) için kullanılan pinlerdir.

Clock Pinleri: Clock sinyalleri için ayrılmış özel pinlerdir (WEB3).

b) Kullanıcı Pinleri:

Bunlar kullanıcı tarafından konfigüre edilebilen standart I/O pinleridir. Input, Output, input/Output olarak üç kategoriye ayrılır. Her bir I/O pini FPGA'de bir IO hücresine bağlıdır. IO hücrelerinin güçleri VCCIO tarafından sağlanır. Eski FPGA' ler birden fazla VCCIO pinine sahip olmalarına rağmen, bütün pinler aynı voltajla beslenirdi.

Yeni üretilen FPGA'lerde ise IO'lar gruplara ayrılabilir ve bu gruplar farklı voltajlardan beslenebilirler. Böylelikle bir grup IO pinleri 3.3 V ile çalışırken; diğer grup IO pinleri de 2.5 volt ile çalışabilmektedir (WEB3).

7.1.2 CLOCK ve GLOBAL Lines

FPGA genellikle senkronize dizayn edilir (synchronous). Yani FPGA tasarımları clock tabanlıdır ve FPGA içerisindeki D Flip-Flop lar, clock sinyalinin yardımıyla durum değiştirirler. Senkronize dizaynlarda bir clock sinyalinin, bütün flip-flop'ları aynı anda tetiklemesi gerekir. Aksi takdirde FPGA'de elektriksel ve zamansal problemler oluşmaktadır. FPGA üreticileri bu problemleri ortadan kaldırabilmek için, "Global Routing" veya "Global Line" olarak adlandırılan özel bir iç bağlantı geliştirmişlerdir. Bu bağlantı sayesinde, Clock sinyalininin FPGA içerisindeki bütün Flip-Flop'lara aynı anda ulaşması sağlanır (WEB3).Bundan dolayı clock beslemelerinin, FPGA'in clock için ayrılmış pinlerinden verilmesi gerekir.

7.1.3 RAM Blokları

Günümüzde FPGA’lerin hemen hepsinde ayrılmış RAM yani bellek üniteleri bulunmaktadır. Bunlar mantık devrelerinin işleyişi sırasında duyulan geçici depolama ihtiyacı için kullanılırlar. Bu RAM’ler tek veya çoklu erişimi destekleyebilirler. Çoklu erişimde birden fazla uygulama RAM üzerinde okuma/yazma yapabilmektedir. Çoklu erişim farklı clock’ta çalışan işlem blokları arasında veri aktarımı için iyi bir çözümdür. Örneğin 25 MHz clock ile çalışan bir veri toplama ünitesinden 50 Hz ile çalışan bir veri işleme ünitesine veri aktarmak için 2 port’lu bir RAM kullanabiliriz. 25 MHz ile çalışan veri toplama ünitesi veriyi RAM’a yazar ve 50 MHz de çalışan veri işleme ünitesi veriyi RAM’den okuyarak kullanır (WEB3).

FPGA içerisinde büyük RAM ihtiyaçları için Block RAM’ler bulunurken küçük veriler için mantık hücreleri arasına serpiştirilmiş dağınık (distributed) küçük RAM’ler bulunmaktadır. Dağınık RAM için Xilinx ihtiyaca göre mantık hücrelerinin bazılarını RAM olarak kullanırken Altera block RAM’leri FPGA içerisinde değişik boyutlarda paylaştırmaktadır.

7.2 FPGA Programlama

FPGA'leri yazılım ile programlanabilen donanımlar olarak tanımlamıştık.

FPGA’lerin kullanıma hazır hale getirilebilmesi için programlanmaları gerekir. (Xilinx ‘de derleyici program olarak ISE , Altera ‘da ise Quartus II programı kullanılır.)

FPGA programlamak için iki yöntem kullanılır.

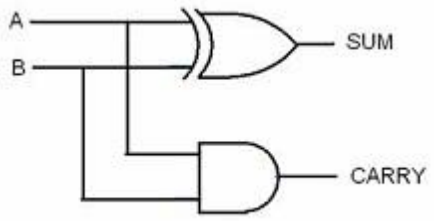
- Grafiksel tasarım
- HDL

Grafiksel tasarımda tasarım, derleyici programın (ISE, Quartus vs.) kütüphanesinde yer alan araç ve mantık kapıları kullanılarak yapılır. HDL(Donanım Tanımlama Dili)Tasarımın, HDL dillerinden her hangi bir tanesinin kullanılarak yapılmasıdır.

HDL: bir donanım parçasını modellemek için kullanılan yazılım dilidir. VHDL ile Verilog en yaygın kullanılan iki türüdür. Biz örneklerimizde VHDL dilini kullanacağız. Bunun sebebi ise Verilog ile VHDL arasında herhangi bir üstünlük veya yetersizlik yoktur. Her iki dilide FPGA programlamak için kullanabilirsiniz. Biz FPGA nedir ekibi olarak VHDL'e aşina olduğumuz için, örneklerimizde bu dili kullanacağız.

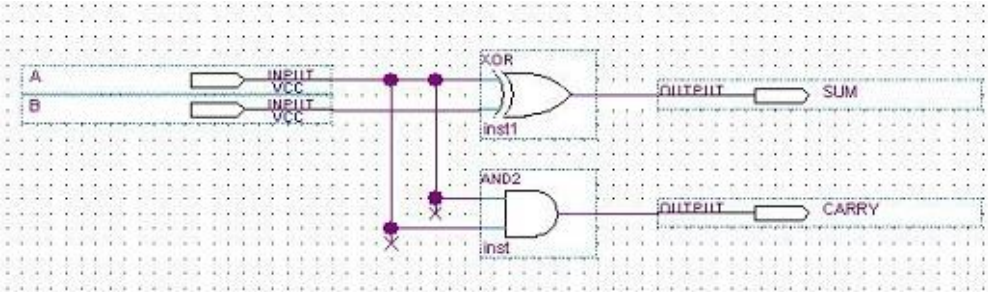
Program tasarlanırken hem grafik hemde VHDL kodu kullanılabilir. HDL ile oluşturduğunuz modellerinizin şematiklerini oluşturup, tasarımlara grafiksel olarak da devam edilebilir.

Anlatımımıza bir örnek ile devam edelim. Örneğimizde bir adet Half Adder'in (Yarım Toplayıcı) (Şekil 7.3) hem Grafiksel hemde VHDL kullanılarak nasıl tasarlandığını görelim.



Şekil 7.3. Half Adder (Yarım Toplayıcı)

Grafik metodunda kullandığımız FPGA programında bir adet AND kapısı ile bir adet XOR kapısı seçip şematik dokümanına aşağıda verilen şekildeki gibi yerleştirelim ve pin bağlantılarını yapalım. Böylelikle Half Adder'ımızı tasarlamış olduk (Şekil 7.4).



Şekil 7.4. Half Adder (Yarım Toplayıcı) Tasarım devresi

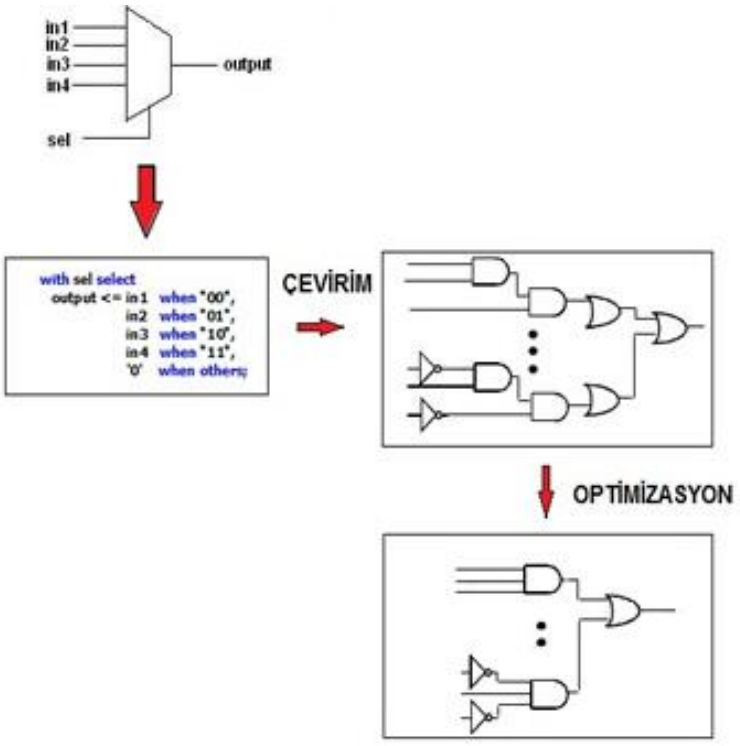
VHDL kullanarak programı yazar isek aşağıda kodu elde ederiz.

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

entity HALF_ADDER is
      Port ( A            :  in  STD_LOGIC;
             B            :  in  STD_LOGIC;
             SUM          : out  STD_LOGIC;
             CARRY        : out  STD_LOGIC);
      end HALF_ADDER;

architecture Behavioral of HALF_ADDER is
    begin
        SUM       <= A XOR B;
         CARRY   <= A AND B;
   end Behavioral;
```

7.2.1 Analiz ve sentezleme

Bu kısımda kullandığımız derleyici program, projenin yada kodu analiz edip, donanımsal olarak mümkün olup olmadığının kararını verir. Eğer yazılımın donanımsal çevirisi mümkünse kod sentezlenir ve RTL şeması çıkarılır.

RTL; oluşturulan kodun, Register cinsinden tasarımının belirtilmesidir. Yani daha basit bir anlatımla VHDL kodumuza karşılık gelen mantık kapılarımdan oluşan devre diyebiliriz (Şekil 7.5.).



Şekil 7.5. RTL Sentezleme

7.2.2 Kısıtlamaların girilmesi

Program dosyasını oluşturmadan önce projemizde kısıtlamaların girilmesi gerekir. Bu kısımda zaman(var ise) ve pin kısıtlamaları(projeden oluşturduğumuz portlara FPGA pinlerinin atanması) belirlenir.

7.2.3 Yerleşim ve bağlantıların yapılması (Place ve Routing)

Derleyici programımız kısıtlamalarıda göz önünde bulundurarak programımızın yerleştirileceği logic elementleri belirler ve logic elementler ile pinler arasındaki bağlantılar yapılır.

7.2.4 Program dosyasının oluşturulması

Place ve Routing işleminden sonra derleyici program, programın yükleneceği cihazlara uygun programlama dosyalarını oluşturur. Mesala program direk FPGA yüklenecek ise JTAG dosyası, Konfigürasyon elemanı veya flash kullanılacaksa, bu cihazların desteklediği program dosyaları oluşturulur.

7.2.5 Programın yüklenmesi

Yukardaki işlemlerden sonra program dosyası FPGA veya yapılandırma elemanlarına yüklenerek işlem tamamlanır. Böylelikle FPGA konfigüre edilmiş olur.

7.3 FPGA Program Yüklenmesi

FPGA program-uçucu (volitile) özellikte olduğundan programı kendi üzerinde saklamaz. FPGA'in elektriksel bağlantısı kesildiğinde program kaybolur. Bu yüzden FPGA tekrar kullanılacağı zaman, programın da yeniden yüklenmesi gerekir.

Programın her seferinde uçmasının önüne geçmek amacıyla FPGA ile birlikte yapılandırma elemanı kullanılmalıdır.

FPGA; bilgisayar, mikrodenetleyiciler veya Boot-PROM kullanılarak konfigüre edilebilir:

- Bilgisayar: Hızlı ve kolay program geliştirme amacıyla kullanılır. Tasarım tamamlandıktan sonra, bilgisayara gerek kalmaz. Bu yöntem genellikle geliştirme aşamasında kullanılır. Geliştirme devreleriyle birlikte verilen özel kablolar da bu amaçla kullanılır.
- Mikrodenetleyici: FPGA'i konfigüre etmek için, içinde gömülü bir yazılım bulunan mikrodenetleyici kullanılabilir.
- Boot-PROM: FPGA üreticilerinin kendilerine özel Boot-RPOM'ları vardır. Bunlar güç verildiğinde FPGA'in otomatik olarak programlanması için kullanılır.

Aslında FPGA programlamak (konfigüre etmek) demek, özel pinleri yarımıyla 1 ve 0'lardan oluşan anlamlı veri paketlerini FPGA'e göndermek demektir.

FPGA'in iki modu vardır.

- Konfigürasyon (configuration)
- Kullanıcı (user) modu.

Konfigürasyon modu: Güç verildiğinde FPGA, yapılandırma moduna geçer. Bu modda FPGA bekleme durumundadır ve bütün çıkışları pasiftir. FPGA programlaması bu aşamada yapılır.

Kullanıcı modu: Yükleme bittikten sonra, FPGA kullanıcı moduna girer ve pinleri aktif hale gelir. Böylece FPGA kendine verilen görevi yerine getirmeye başlar. Altera ve Xilinx FPGA'lerinin yapılandırma yöntemleri benzerdir. FPGA programlarken aşağıdaki iki ara yüzden biri kullanılır (WEB3).

- JTAG Ara yüzü
- Synchronous Serial Ara yüzü

7.3.1 JTAG (Joint Test Action Group) Ara yüzü

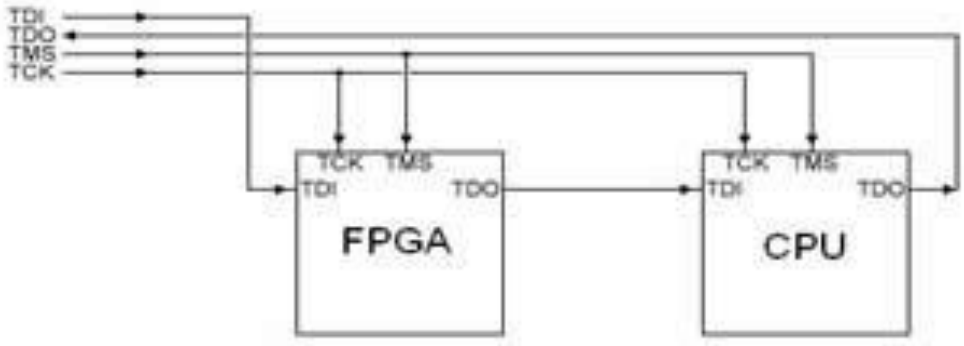
1980 yıllında geliştirilen bir IEEE standardıdır. Günümüzde bütünleşmiş devrelerde hata ayıklamak için kullanılan en yaygın metodudur. JTAG kullanmak isteniyorsa, tasarımlarda FPGA gibi JTAG uyumlu entegreler kullanılmalıdır.

FPGA'in bütün I/O pinleri JTAG Ara yüzü (interface) ile kontrol edilebilmektedir. Böylelikle JTAG'in kendine özgü komutları (proprietary JTAG commands) kullanılarak FPGA programlanabilmektedir.

JTAG beş adet sinyalden oluşan bir seri veri yoludur:

- Test Data Input (TDI) : Veri almak için kullanılır.
- Test Data Output (TDO) : Veri göndermek için kullanır.
- Test Mode Select (TMS) : Cihaza komutları göndermek için kullanılır.
- Test Clock (TCK) : Clock sinyalidir.
- Test Rest (TRST) : Opsiyoneldir yani kullanılması şart değildir. JTAG'i resetlemek için kullanılır.

Tek bir JTAG portu bir yâda daha fazla cihaza bağlanabilir. Bu şekildeki çoklu cihaz kullanımında, JTAG zinciri (JTAG chain) oluşturulur. TMS ve TCK pinleri bütün cihazlara doğrudan bağlanır. Bir cihazın TDI pini, diğer cihazın TDO pinine bir zincir oluşturacak şekilde bağlanır. Bir ana (master) kontrolcü ise zinciri tamamlar (Şekil 7.6) (WEB3).



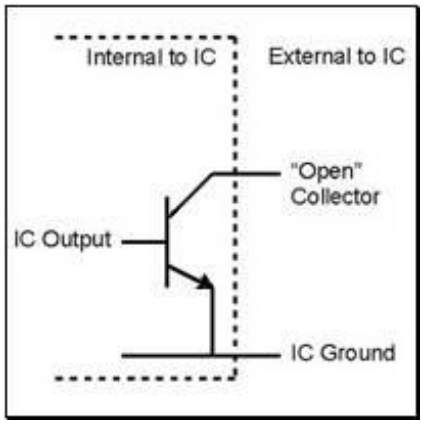
Şekil 7.6. JTAG Yapısı

Zincir içerisindeki bütün cihazların kendine özel kimlik kartları (ID) vardır ve bu sayede bilgisayar zincir içerisindeki bütün cihazları ayırt edebilmektedir. JTAG pinleri genelde ayrılmış özel (Dedicated) pinlerdir ve yalnızca bu amaç için kullanılır. Test edilecek devreyi bilgisayara bağlamak için JTAG kablo kullanılmalıdır. Bu kablo; Ethernet, USB ve paralel Ara yüzlerinden birine sahip olabilir (WEB3).

7.3.2 Synchronous Serial Arayüzü

Bir bitlik bu data/clock Ara yüzü ile FPGA'e senkronize bir şekilde birer bit gönderilir. Bu ara yüzde kullanılan pinlerden aşağıda belirtilen beş tanesi en önemlileridir:

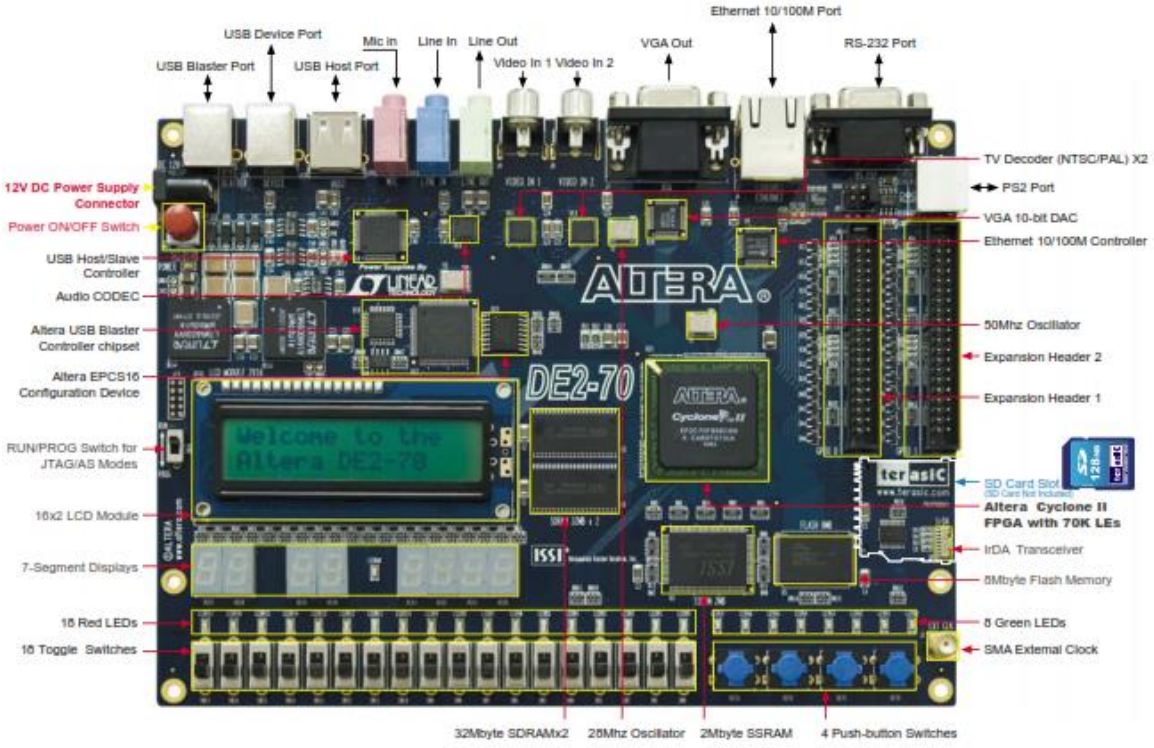
- **Data / data0:** Konfigürasyon veri bitidir. FPGA için inputtur.
- **Clk / Clk:** Konfigürasyon clock bitidir. FPGA'e gönderilen yapılandırma veri bitleri clock 'un her yükselen ucunda kaydırılır.
- **Prog_b / nconfig: FPGA** için input olan bu bit Active low (0) özellikli çalışır. Değeri "0" olduğunda FPGA resetlenerek, yüklenen yapılandırma (program) silinir. Eğer FPGA kullanıcı modunda çalışıyorken bu bit aktif olursa; FPGA hemen işlemini durdurur ve bütün I/O pinleri tri_state modunaa geçer (high empedance).
- **Init_b / nStatus: FPGA**' in oluşturduğu bir Output olan bu bit active low (0) özellikli çalışır. FPGA'in yapılandırma işlemine hazır olup olmadığını gösterir.
- **Done / confDone: FPGA'in** oluşturduğu bir Output olan bu bitin değeri 1 (high) ise, FPGA konfigürasyonunun tamamlandığı ve kullanıcı moduna geçildiği anlamına gelir.
- Init_b ve done pinleri open-collector pinleridir. Bu yüzden bu pinlere bağlantı yaparken birer adet pull-up direnci kullanılmak zorundayız (WEB3).



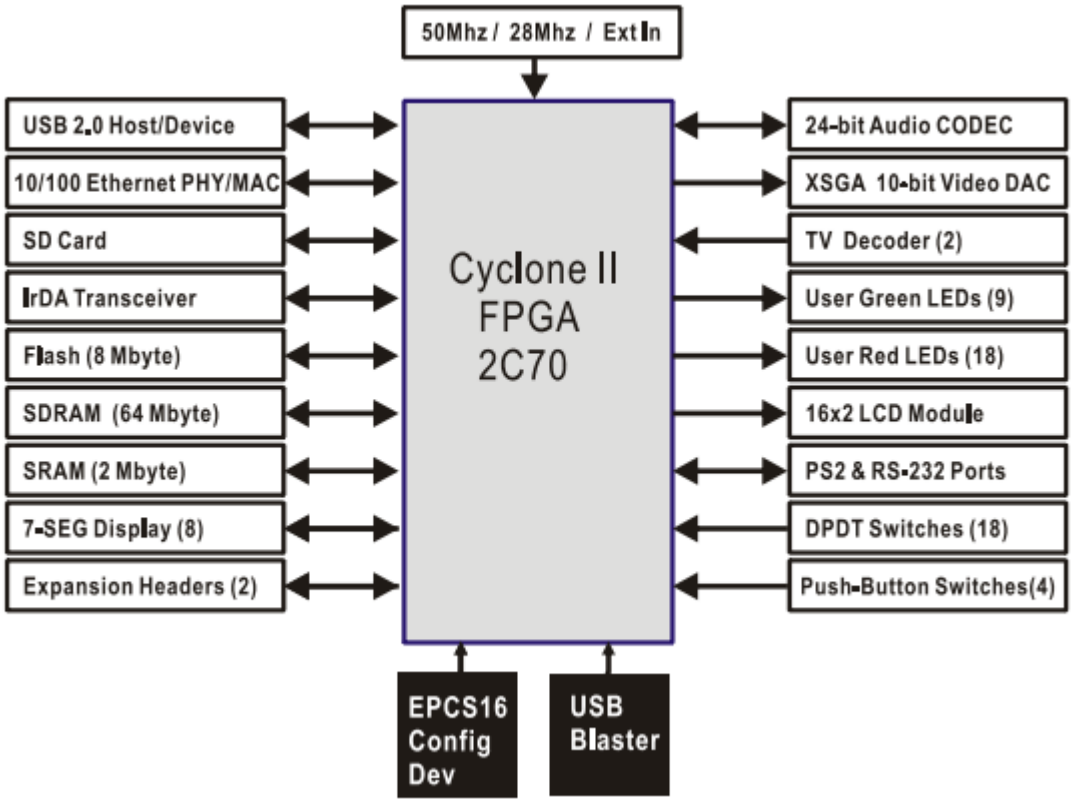
Şekil 7.7. Synchronous Serial Arayüzü

8. KULLANILAN TEST BOARD YERLEŞİMİ (ALTERA DE2-70)

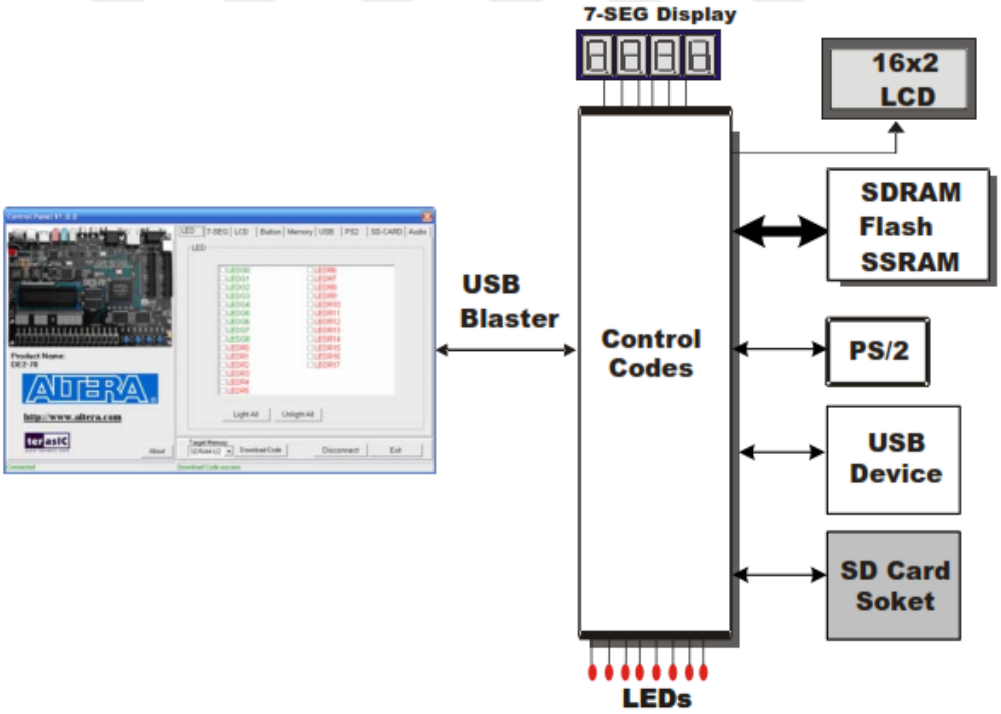
Fakültemizde bulunan ve deneysel çalışmalarımızda kullanacağımız Altera DE-2 70 board (Şekil 8.1), blok diyagramı (Şekil 8.2) ve kontrol yapısı (Şekil 8.3) verilmiştir. (WEB4)



Şekil 8.1. Altera DE-2 70 Test Bordu (WEB4)



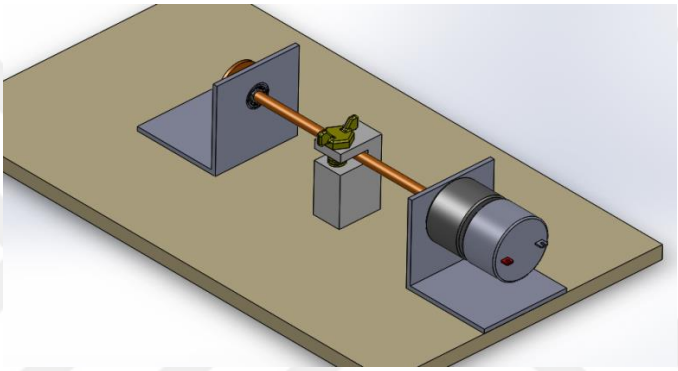
Şekil 8.2. ALTERA DE2-70 Blok Diyagramı (WEB4)



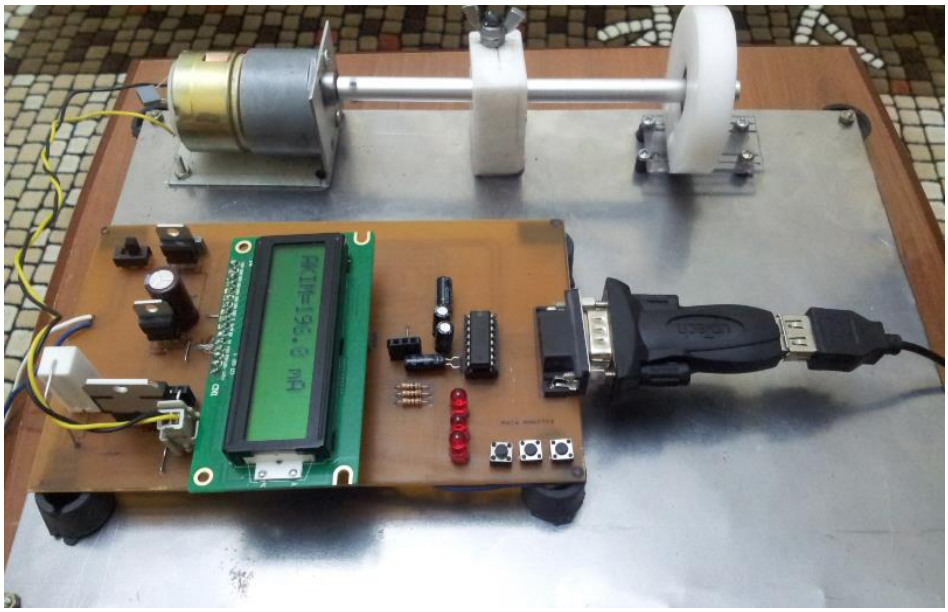
Şekil 8.3. ALTERA DE2-70 Kontrol Yapısı Görünümü (WEB4)

9. BİRİNCİ DENEY SETİ

Deney setimizde DC motorun rotor miline akuple edilmiş bir düzenek ile motora belirli oranlarda yük bindirilmiştir (Şekil 9.1). Motor milinin rulmanla yataklanan ucuna ise bir opto-kuplör devresi bağlanmış olup, motor milinin anlık devir bilgisi dijital bir bilgi olarak mikrodenetleyiciye gönderilmiştir. Motorun çektiği akım değeri ve motor uçlarındaki gerilim ile motor devri kontrol kartı üzerindeki seri port üzerinden Visual Basic arayüzünde hazırlanan bir form ile grafiksel hale getirilmiştir. Motor miline uygulanan yük miktarının değişimi de bu formda grafiksel olarak gösterilmiştir. Ayrıca veriler belirli periyodlar dâhilinde Microsoft Excel ortamına aktarılarak kayıt altına alınmıştır (Şekil 9.2).

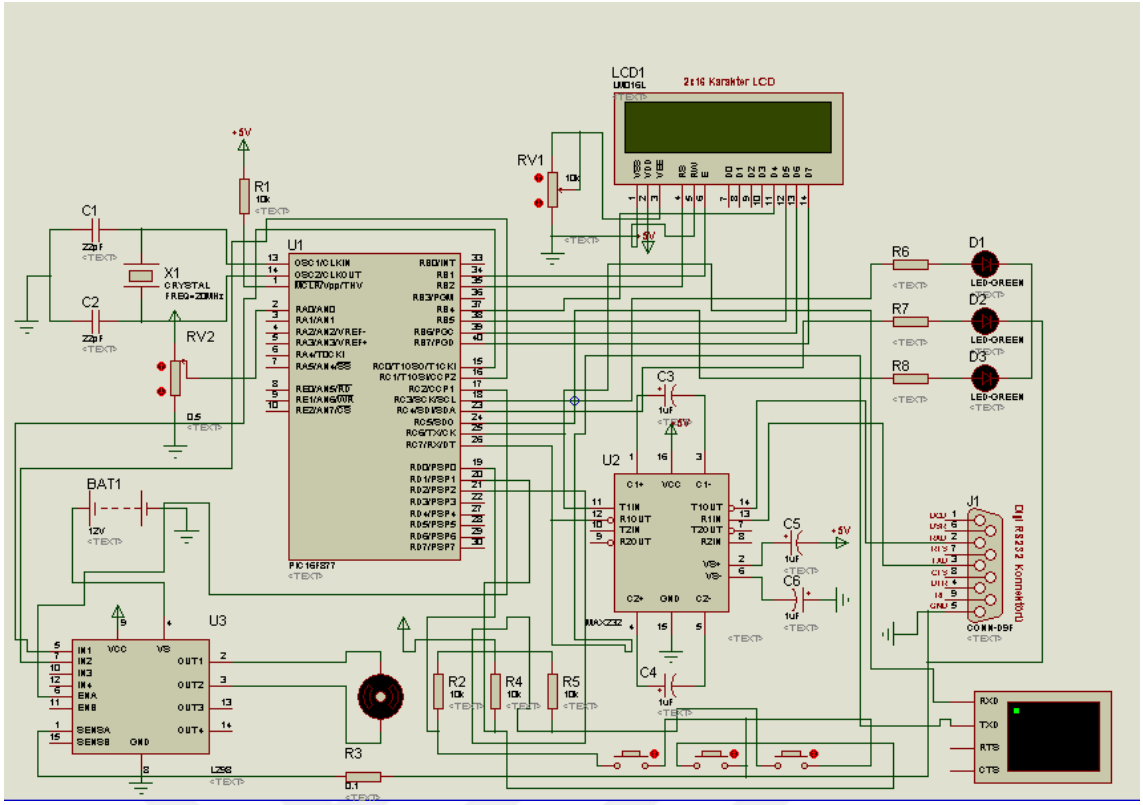


Şekil 9.1. DC Motor frenleme sistemine ait SolidWorks Çizimi

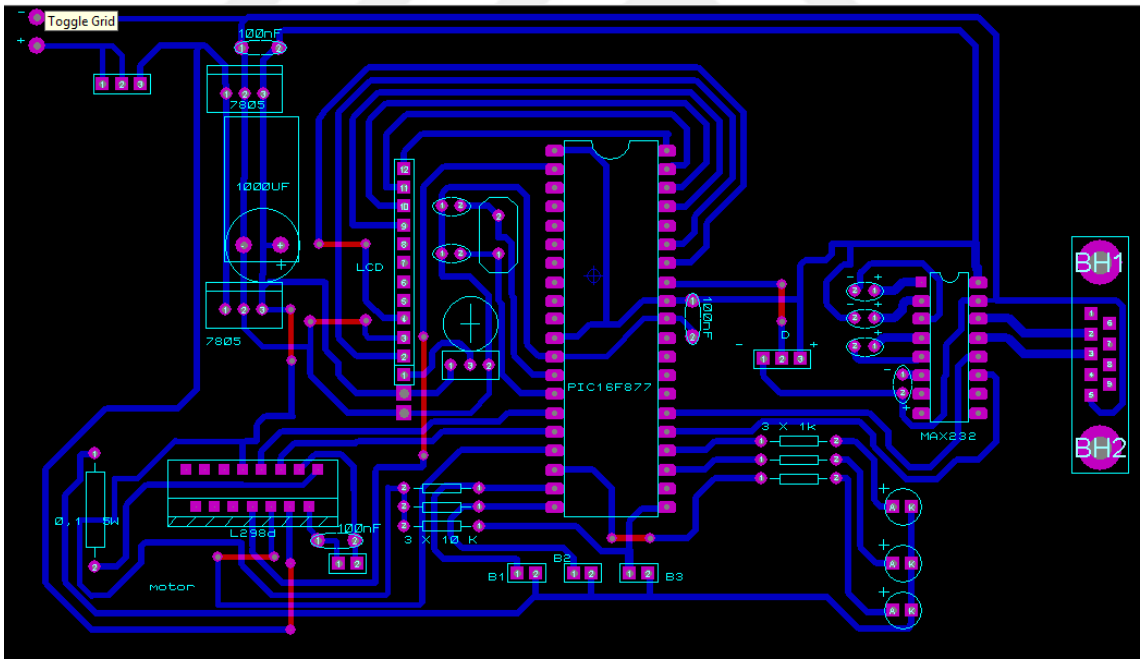


Şekil 9.2. DC Motor Yük Kontrol Seti

Birinci deney seti için devre şeması (Şekil 9.3) ve baskı devresi (Şekil 9.4) ISIS programında çizilmiş ve gerçekleştirilmiştir.



Şekil 9.3. DC Motor yük kontrol setine ait devrenin ISIS şeması



Şekil 9.4. DC Motor yük kontrol setine ait devrenin ARES şeması

Birinci deney mikroişlemci devresi C yazılımı:

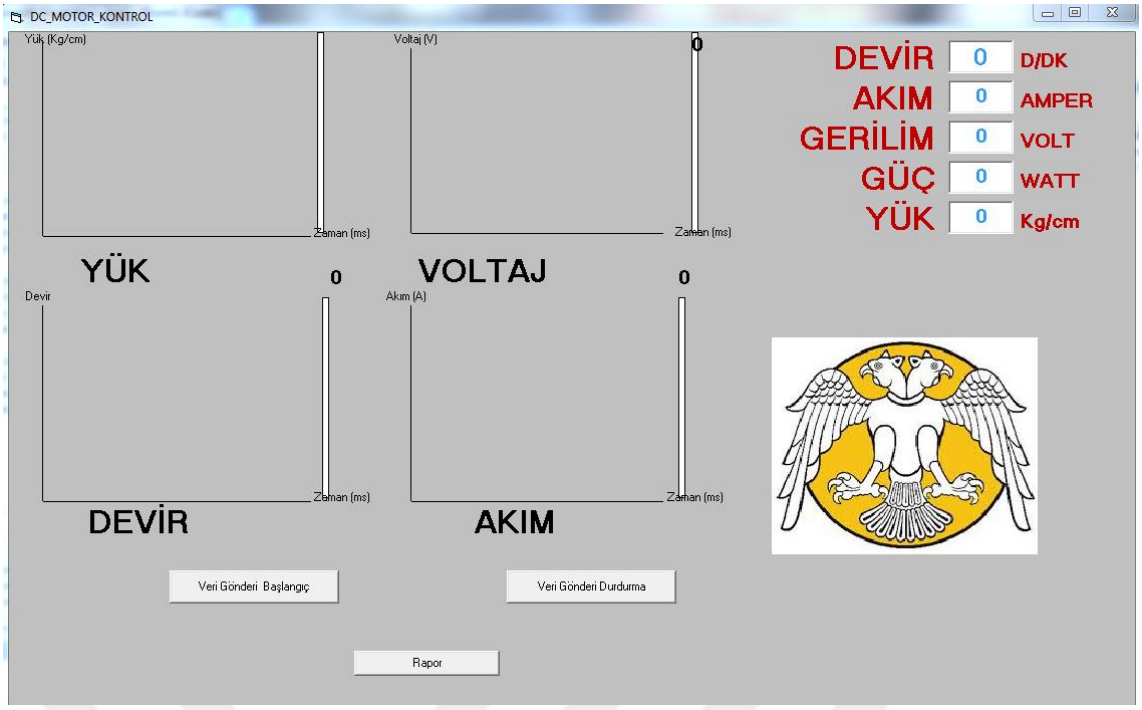
```
#include <16f877a.h>
#device ADC=10  // 10 bitlik ADC kullanılacağı belirtiliyor.
#fuses HS,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,NODEBUG,NOCPD
#use delay (clock=20000000)
```

```
#use rs232 (baud=9600, xmit=pin_C6, rcv=pin_C7) // RS232 protokolünün 9600 bit/sn baud hızında olacağı
#byte porta=5
#byte portb=6
#byte portc=7
#byte portd=8
#include <lcd2x16.c>  // lcd  dosyamızı  eklenmesi.
unsigned int16 devir,i=0; // Tamsayı tipinde değişken tanımlanıyor
unsigned long int bilgi; // İşaretsiz 16 bitlik tam sayı tipinde değişken tanımlanıyor
float voltaj,akim;   // ondalıklı tipte değişkenler tanıtılıyor
int data;
hesapla(){
bilgi=read_adc();  // ADC sonucu okunuyor ve bilgi değişkenine aktarılıyor
voltaj=(0.0048828125*bilgi*1000); // Dijital çevirme işlemine uğrayan sinyalin mV olarak gerilimi hesaplanıyor
akim=voltaj;
}
main()
{  setup_psp(PSP_DISABLED);        // PSP birimi devre dışı
  setup_spi(SPI_SS_DISABLED);     // SPI birimi devre dışı
  setup_timer_1(T1_DISABLED);     // T1 zamanlayıcısı devre dışı
   set_tris_a(0x1f);   set_tris_b(0x01);   set_tris_c(0x80);   set_tris_d(0x07);   output_c(0x00);
  output_d(0x00);
  setup_adc(ADC_CLOCK_INTERNAL);   // ADC clock frekansı fosc/32
  setup_adc_ports(AN0);                        //RA0/AN0 girişi analog
  lcd_ayarla();
  set_adc_channel(0);   // RA0/AN0 ucundaki sinyal A/D işlemine tabi tutulacak
  delay_us(20);                 // Kanal seçiminde sonra bu bekleme süresi verilmelidir
  adres(1,1);                     // imleç 1.sütun, 1.satırda.
  printf(lcd_yaz,"  projeprojepppp 2015 ");
  delay_ms(1000);
    lcd_sil();
    while(1)
  {
    output_high(pin_c0);
    output_low(pin_c1);
       while(1)   {
    output_high(pin_c0);
    output_low(pin_c1);
    bilgi=read_adc();  // ADC sonucu okunuyor ve bilgi değişkenine aktarılıyor
    voltaj=(0.0048828125*bilgi*1000);  // Dijitale çevirme işlemine uğrayan sinyalin mV olarak gerilimi  hesaplanıyor
    akim=voltaj;
```

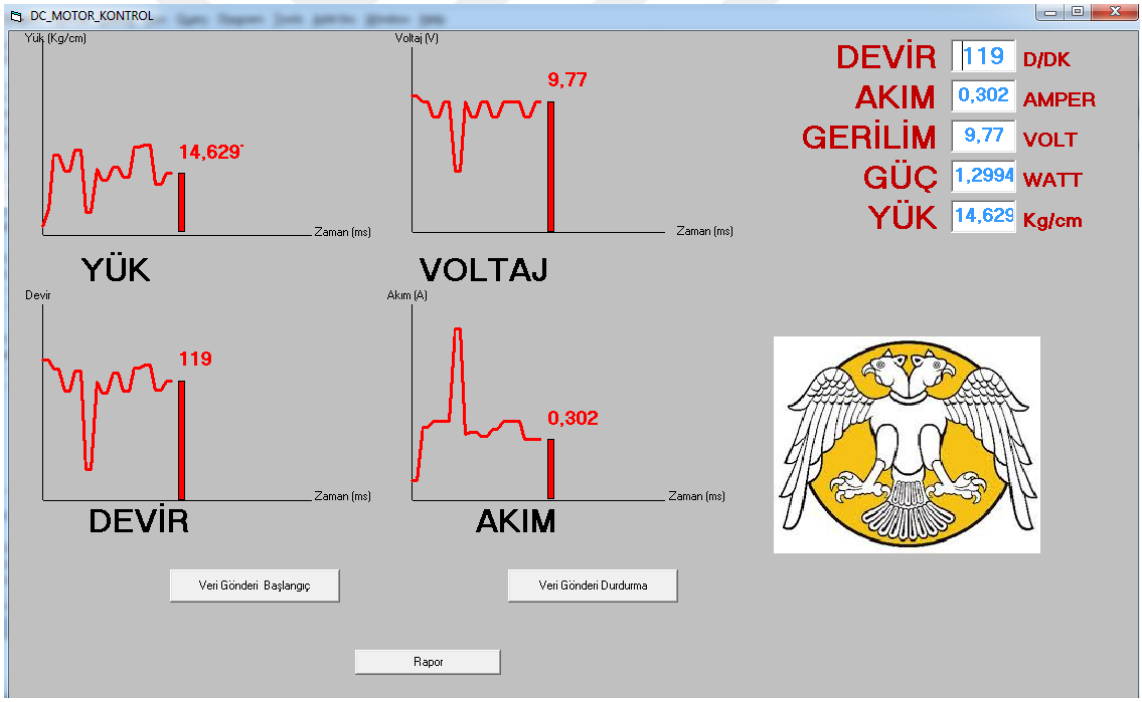
```
    adres(1,1);
    printf(lcd_yaz,"I=%4.1f mA ",akim);
    if (akim>=350 && akim<400){portc=0x01;delay_ms(50);portc=0x00;delay_ms(50);
    devir=36;putc(devir); adres(1,2);printf(lcd_yaz,"D=%ld",devir); }
    if (akim>=400 && akim<450){portc=0x01;delay_ms(55);portc=0x00;delay_ms(45);
    devir=40;putc(devir);adres(1,2);printf(lcd_yaz,"D=%ld",devir);    }//%80 pwm
    if (akim>=450 && akim<500){ portc=0x01;delay_ms(60);portc=0x00;delay_ms(40);
    devir=45;putc(devir);adres(1,2);printf(lcd_yaz,"D=%ld",devir);   }//%75 pwm
    if (akim>=550 && akim<600){portc=0x01;delay_ms(65);portc=0x00;delay_ms(35);
    devir=60;putc(devir);adres(1,2);printf(lcd_yaz,"D=%ld",devir);   }//%66 pwm
    if (akim>=600 && akim<650){portc=0x01;delay_ms(70);portc=0x00;delay_ms(30);
    devir=90;putc(devir);adres(1,2);printf(lcd_yaz,"D=%ld",devir);  }//%50 pwm
    if (akim>=650 && akim<700){portc=0x01;delay_ms(75);portc=0x00;delay_ms(25);
    devir=120;putc(devir);adres(1,2);printf(lcd_yaz,"D=%ld",devir);    }//%33 pwm
    if (akim>=700 && akim<750){portc=0x01;delay_ms(80);portc=0x00;delay_ms(20);
    devir=135;putc(devir);adres(1,2);printf(lcd_yaz,"D=%ld",devir);  }//%25 pwm
    if (akim>=750 && akim<800){portc=0x01;delay_ms(85);portc=0x00;delay_ms(15);
    devir=144;putc(devir);adres(1,2);printf(lcd_yaz,"D=%ld",devir);   }//%20 pwm
    if (akim>=800 && akim<850){portc=0x01;delay_ms(90);portc=0x00;delay_ms(10);
    devir=150;putc(devir);adres(1,2);printf(lcd_yaz,"D=%ld",devir);   }//%16 pwm
    if (akim>=850){portc=0x01;  devir=180;putc(devir);adres(1,2);printf(lcd_yaz,"D=%ld",devir);
     }//%0 pwm
    delay_ms(100);

  }
}
```

9.1. Birinci Deney Setine Ait Bilgisayar Arayüz Programı

Deney verilerini almak ve saklamak için Visual Studio yazılımında arayüz yazılımı gerçekleştirilmiştir. RS232 kullanılarak deney setindeki veriler bilgisayar ortamına aktarılmış ve kaydedilmiştir. Bu arayüz ile aynı zamanda arayüzde grafiksel olarak yük, devir, akım ve gerilim değerlerini de görmek mümkün olmuştur (Şekil 9.5, Şekil 9.6).



Şekil 9.5. DC Motor yük kontrol setine ait Ara yüz devresi



Şekil 9.6. DC Motor yük kontrol setine ait Ara yüz devresi(Veri Transferi Mevcut)

9.2. Grafiksel Ara Yüz Devresi Yazılımı

Option Base 1

Dim nokta1x, nokta1y, nokta2x, nokta2y

Dim yuknokta1x, yuknokta1y, yuknokta2x, yuknokta2y

Dim akimnokta1x, akimnokta1y, akimnokta2x, akimnokta2y

Dim vnokta1x, vnokta1y, vnokta2x, vnokta2y

```
Dim akim(100)
Dim devir(100)
Dim voltaj(100)
Dim renkdevir, renkakim, renkvoltaj, renkyuk
Dim a, s, f, koorx1, koorx2, koory1, koory2, r
Dim b, c As Double
Private Sub Command5_Click()
MSComm1.PortOpen = False
End
End Sub
Private Sub Command1_Click()
'Timer2.Enabled = True
'Timer1.Enabled = True
 Form1.Show
 End Sub
Private Sub Command2_Click()
Open App.Path & "\veri.txt" For Output As #1
For i = 1 To 10
Write #1, i & "- Devir:" & devir(i) & " - Akım:" & akim(i) & " - Voltaj:" & voltaj(i)
Next i
End Sub
Private Sub Form_Load()
Form1.DrawWidth = 3
MSComm1.CommPort = 3
MSComm1.PortOpen = True
f = 2
s = 1
r = 1

'Devir başlangış koordinatları
nokta1x = 500
nokta1y = 7000 - 0 * 15
'Akım başlangış koordinatları
akimnokta1x = 6000
akimnokta1y = 7000
'Gerilim  Başlangıç
vnokta1x = 6000
vnokta1y = 3000
'Yük  Başlangıç
yuknokta1x = 500
yuknokta1y = 3000
'renk ayarları
renkdevir = RGB(255, 0, 0)
```

```
 renkakim = RGB(255, 0, 0)
 renkvoltaj = RGB(255, 0, 0)
 renkyuk = RGB(255, 0, 0)
End Sub
Private Sub Timer1_Timer()
a = MSComm1.Input
If a = "" Then
Timer2.Enabled = False
Else
Timer2.Enabled = True
End If
a = Left(a, 1)
If a = "A" Then
Text1.Text = 140
aa = (Int(Rnd(1) * 30)) / 1000
b = 0.09 + aa
b = Round(b * 1000) / 1000
c = 10.2
Text2.Text = b
Text3.Text = c
Text4.Text = b * c
ElseIf a = "B" Then
rr = Int(Rnd(1) * 23) + 117
Text1.Text = rr

aa = Rnd(1) / 10 + 0.1
aa = Round(aa * 1000) / 1000
b = aa
c = 9.77
Text3.Text = c
Text4.Text = b * c
ElseIf a = "C" Then
rr = Int(Rnd(1) * 5) + 112
Text1.Text = rr
aa = Rnd(1) / 10 + 0.2
aa = Round(aa * 1000) / 1000
b = aa
c = 9.2
Text2.Text = b
Text3.Text = c
Text4.Text = b * c
ElseIf a = "D" Then
rr = Int(Rnd(1) * 7) + 105
```

```
Text1.Text = rr
aa = Rnd(1) / 10 + 0.3
aa = Round(aa * 1000) / 1000
b = aa
c = 8.6
Text2.Text = b
Text3.Text = c
Text4.Text = b * c
ElseIf a = "E" Then
rr = Int(Rnd(1) * 12) + 93
Text1.Text = rr

aa = Rnd(1) / 10 + 0.4
aa = Round(aa * 1000) / 1000
b = aa
c = 7.9
Text2.Text = b
Text3.Text = c
Text4.Text = b * c
ElseIf a = "F" Then
rr = Int(Rnd(1) * 23) + 70
Text1.Text = rr
aa = Rnd(1) / 10 + 0.5
aa = Round(aa * 1000) / 1000
b = aa
c = 7.2
Text2.Text = b
Text3.Text = c
Text4.Text = b * c
ElseIf a = "G" Then
rr = Int(Rnd(1) * 24) + 46
Text1.Text = rr
'Text1.Text = 46        %66 pwm
aa = Rnd(1) / 10 + 0.6
aa = Round(aa * 1000) / 1000
b = aa
c = 6.5
Text2.Text = b
Text3.Text = c
Text4.Text = b * c
ElseIf a = "H" Then
rr = Int(Rnd(1) * 11) + 35
Text1.Text = rr
```

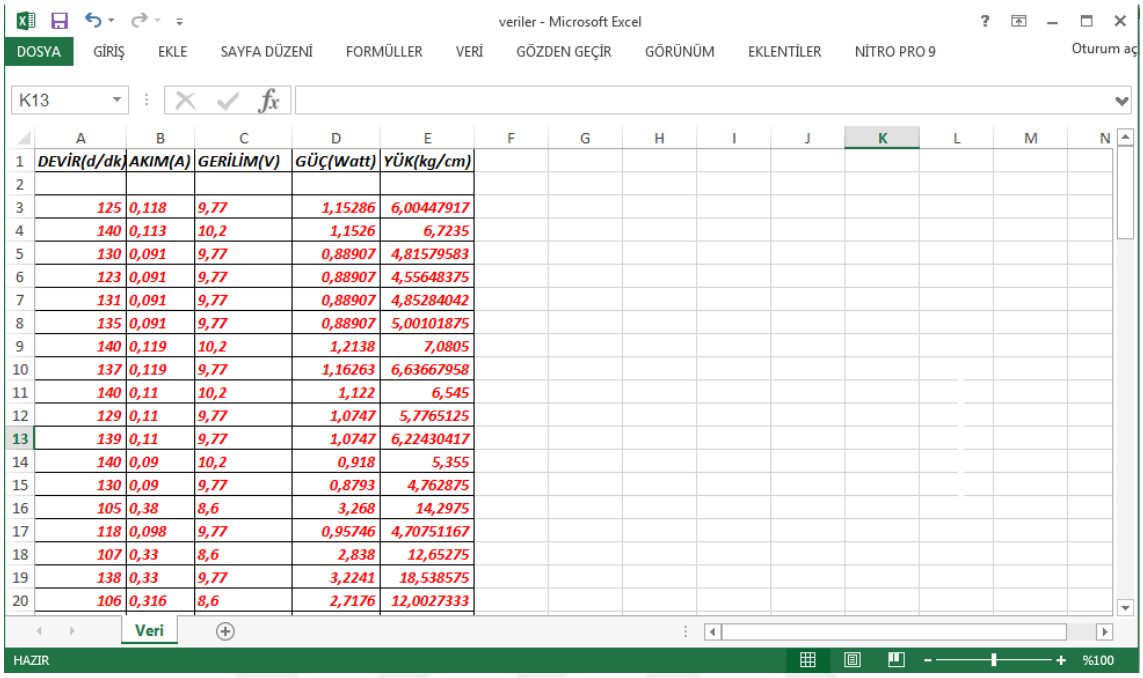
```
'Text1.Text = 35        %75 pwm

aa = Rnd(1) / 10 + 0.7
aa = Round(aa * 1000) / 1000
b = aa
c = 5.6
Text2.Text = b
Text3.Text = c
Text4.Text = b * c
ElseIf a = "K" Then
rr = Int(Rnd(1) * 7) + 28
Text1.Text = rr
aa = Rnd(1) / 10 + 0.8
aa = Round(aa * 1000) / 1000
b = aa
c = 4.6
Text2.Text = b
Text3.Text = c
Text4.Text = b * c
ElseIf a = "L" Then
rr = Int(Rnd(1) * 6) + 22
Text1.Text = rr
aa = Rnd(1) / 10 + 0.9
aa = Round(aa * 1000) / 1000
b = aa
c = 3.2
Text2.Text = b
Text3.Text = c
Text4.Text = b * c
End If
akim(s) = Text2.Text
devir(s) = Text1.Text
voltaj(s) = Text3.Text
s = s + 1
If s = 11 Then s = 1

Text5.Text = Text1.Text * Text2.Text * Text3.Text / 24 ' Yük formülü
End Sub
Private Sub Timer2_Timer()
On Error GoTo son
Shape1.BackColor = renkdevir
Shape1.Height = Text1.Text * 15
h = 3000 - Shape1.Height
```

```
Shape1.Top = 4000 + h
devlabel.Top = Shape1.Top - 500
devlabel.Caption = Text1.Text
devlabel.ForeColor = renkdevir
Shape2.BackColor = renkakim
Shape2.Height = Round(Text2.Text * 3000)
h = 3000 - Shape2.Height
Shape2.Top = 4000 + h
akimlabel.Top = Shape2.Top - 500
akimlabel.Caption = Text2.Text
akimlabel.ForeColor = renkakim
Shape4.Height = Round(Text5.Text * 60)
Shape4.BackColor = renkyuk
h = 3000 - Shape4.Height
Shape4.Top = h
yuklabel.Top = Shape4.Top - 500
yuklabel.Caption = Text5.Text
yuklabel.ForeColor = renkyuk
Shape3.Height = Round(Text3.Text * 200)
Shape3.BackColor = renkvoltaj
h = 3000 - Shape3.Height
Shape3.Top = h
vlabel.Top = Shape3.Top - 500
vlabel.Caption = Text3.Text
vlabel.ForeColor = renkvoltaj
r = r + 1
'Devir 2.nokta koordinatları
nokta2y = Shape1.Top
nokta2x = 500 + r * 80
'Akım 2.nokta koordinatları
akimnokta2y = Shape2.Top
akimnokta2x = 6000 + r * 80
'Gerilim 2.nokta koordinatları
vnokta2y = Shape3.Top
vnokta2x = 6000 + r * 80
'Yük 2.nokta koordinatları
yuknokta2y = Shape4.Top
yuknokta2x = 500 + r * 80
If r = 50 Then
Cls
nokta1y = 7000 - Text1.Text * 15
nokta1x = 500
akimnokta1y = 7000 - Text2.Text * 3000
```

```
akimnokta1x = 6000
vnokta1y = 3000 - Text3.Text * 200
vnokta1x = 6000
yuknokta1y = 3000 - Text5.Text * 15
yuknokta1x = 500
r = 0
GoTo son
End If
Form1.ForeColor = renkdevir
Line (nokta1x, nokta1y)-(nokta2x, nokta2y) 'Devir Grafiği
Form1.ForeColor = renkakim
Line (akimnokta1x, akimnokta1y)-(akimnokta2x, akimnokta2y) 'Akım Grafiği
Form1.ForeColor = renkvoltaj
Line (vnokta1x, vnokta1y)-(vnokta2x, vnokta2y) 'Gerilim Grafiği
Form1.ForeColor = renkyuk
Line (yuknokta1x, yuknokta1y)-(yuknokta2x, yuknokta2y) 'Yük Grafiği
Shape1.Left = nokta2x + 100
devlabel.Left = nokta2x + 100
nokta1y = nokta2y
nokta1x = nokta2x
Shape2.Left = akimnokta2x + 100
akimlabel.Left = akimnokta2x + 100
akimnokta1y = akimnokta2y
akimnokta1x = akimnokta2x
Shape3.Left = vnokta2x + 100
vlabel.Left = vnokta2x + 100
vnokta1y = vnokta2y
vnokta1x = vnokta2x
Shape4.Left = yuknokta2x + 100
yuklabel. Left = yuknokta2x + 100
yuknokta1y = yuknokta2y
yuknokta1x = yuknokta2x
son:
End Sub
```

9.3. DC Motor Veri Aktarım Sayfası

Bilgisayara Excel formatında online kaydedilen veri dosyası içeri Şekil 9.7'de verilmiştir.



	A	B	C	D	E
1	DEVİR(d/dk)	AKIM(A)	GERİLİM(V)	GÜÇ(Watt)	YÜK(kg/cm)
2					
3	125	0,118	9,77	1,15286	6,00447917
4	140	0,113	10,2	1,1526	6,7235
5	130	0,091	9,77	0,88907	4,81579583
6	123	0,091	9,77	0,88907	4,55648375
7	131	0,091	9,77	0,88907	4,85284042
8	135	0,091	9,77	0,88907	5,00101875
9	140	0,119	10,2	1,2138	7,0805
10	137	0,119	9,77	1,16263	6,63667958
11	140	0,11	10,2	1,122	6,545
12	129	0,11	9,77	1,0747	5,7765125
13	139	0,11	9,77	1,0747	6,22430417
14	140	0,09	10,2	0,918	5,355
15	130	0,09	9,77	0,8793	4,762875
16	105	0,38	8,6	3,268	14,2975
17	118	0,098	9,77	0,95746	4,70751167
18	107	0,33	8,6	2,838	12,65275
19	138	0,33	9,77	3,2241	18,538575
20	106	0,316	8,6	2,7176	12,0027333

Şekil 9.7. DC Motor yük kontrol setine ait Excel Veri Aktarım Sayfası

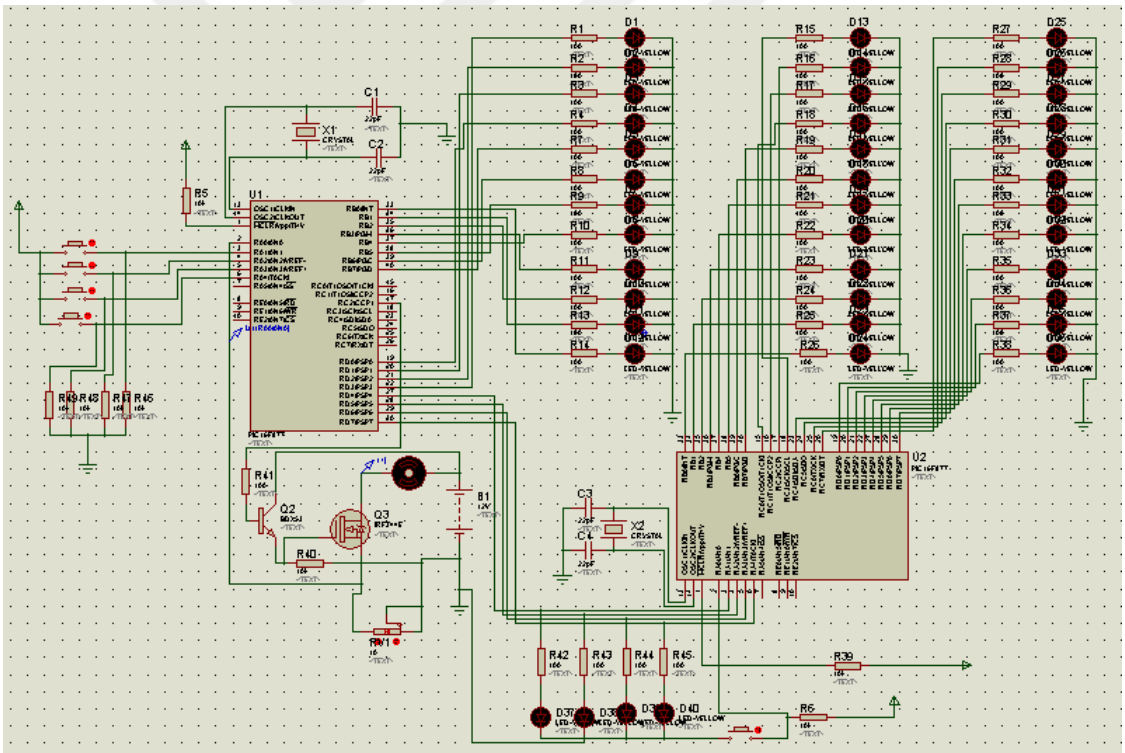
DC motor veri aktarımı Ara yüz üzerinde bulunan "veri aktarımına başlat" butonu sayesinde saniyede 10 veri olacak şekilde (100 ms) bir Excel dosyasına gönderilir. Aynı Ara yüzde bulunan "veri aktarımına durdur" butonu ilede veri dosyasına veri gönderim işlemi durdurulur. Veri aktarımı gerçekleştikten sonra deney kartından gelen farklı PWM değerlerine ve farklı motor mili yük değerlerine sahip veriler ile aşağıdaki grafikler elde edilmiştir.

10. İKİNCİ DENEY SETİ

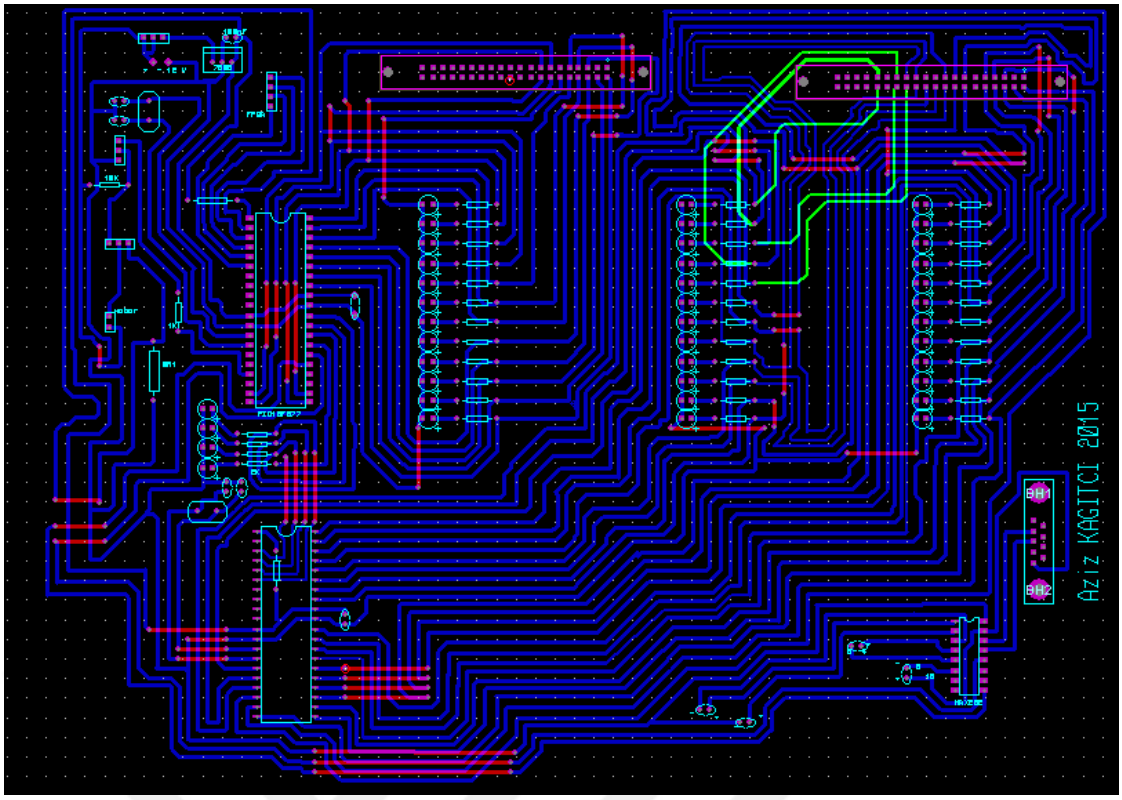
Deney setimizde DC motorun rotor miline akuple edilmiş bir düzenek ile motora belirli oranlarda yük bindirilmiştir. Motor milinin rulmanla yataklanan ucuna ise bir opto-kuplör devresi bağlanmış olup, motor milinin anlık devir bilgisi dijital bir bilgi olarak mikro denetleyiciye gönderilmiştir.

Motorun çektiği akım değeri ve motor uçlarındaki gerilim ile motor devir değerlerinin gönderildiği kartta her bir değer 12 Bitlik veri olarak kontrol kartı üzerindeki Bir IDE portu sayesinde FPGA kartına aktarılarak belirli bir referans değerine göre kart üzerindeki mikro denetleyiciye PWM sinyali gönderilir. Ayrıca Test Kartı üzerinde her 12 Bitlik veri için 12 adet led ile anlık gösterim sağlanmıştır. Motor miline uygulanan yük miktarının değişimi FPGA kartındaki Bulanık Mantık yazılımına göre analiz edilerek DC motora en uygun PWM değerinde sinyal gönderir.

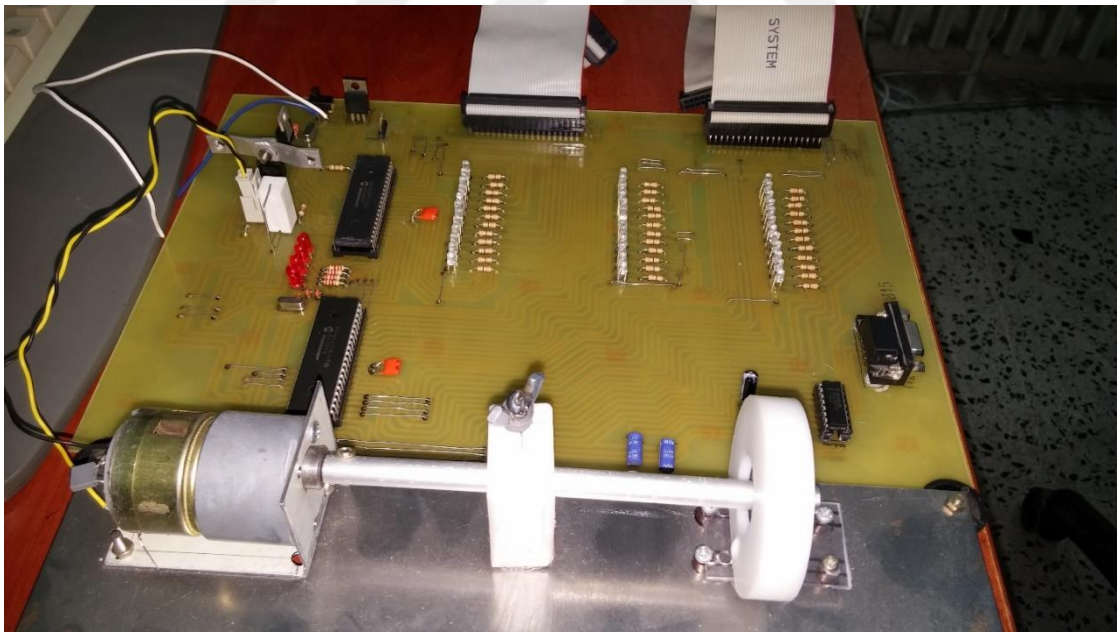
Deney seti için devre şeması (Şekil 10.1) ve baskı devresi (Şekil 10.2) ISIS programında çizilmiş ve gerçekleştirilmiştir (Şekil 10.3).



Şekil 10.1. DC Motor Bulanık Mantık yük kontrol setine ait devrenin ISIS şeması



Şekil 10.2. DC Motor Bulanık Mantık yük kontrol setine ait devrenin ARES şeması



Şekil 10.3. DC Motor Bulanık Mantık yük kontrol seti görünümü

10.1. Mikroişlemci Devresi C Yazılımı

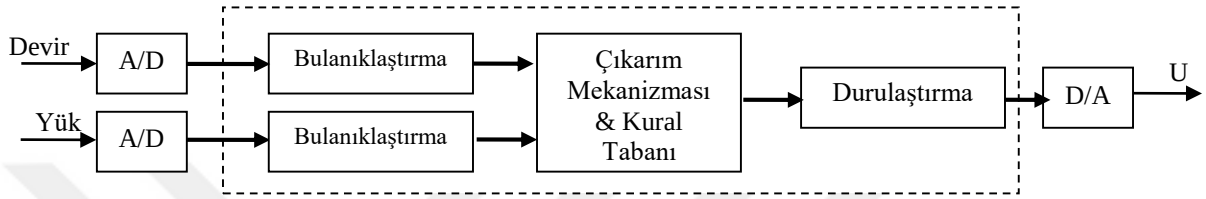
```
#include<16f877.h>
#device ADC=10  // 10 bitlik ADC kullanılacağı belirtiliyor.
#fuses HS,NOWDT,PUT,NOPROTECT
#use Delay(Clock=20000000)
```

```
#byte port_a=5
#byte portb=6          // b portunun tanımlanması
#byte port_c=7         // c portunun tanımlanması
#byte portd=8          // d portunun tanımlanması
float bilgi,fpga;
fonksiyon1(){
setup_adc(adc_clock_div_32);   // ADC clock frekansı fosc/32
 setup_adc_ports(ALL_ANALOG); //TÜM AN girişleri analog
 set_adc_channel(0);   // RA0/AN0 ucundaki sinyal A/D işlemine tabi tutulacak
 delay_us(20);        // Kanal seçiminde sonra bu bekleme süresi verilmelidir
 bilgi=read_adc();  // ADC sonucu okunuyor ve bilgi değişkenine aktarılıyor
 delay_us(20);        // Kanal seçiminde sonra bu bekleme süresi verilmelidir
 bilgi=bilgi*10;
set_adc_channel(1);   // RA1/AN1 ucundaki sinyal A/D işlemine tabi tutulacak
 delay_us(20);        // Kanal seçiminde sonra bu bekleme süresi verilmelidir
 fpga=read_adc();  // ADC sonucu okunuyor ve bilgi değişkenine aktarılıyor
 fpga=0.0048828125*fpga;
if (bilgi<50) {portb=0;}
          else if ((bilgi>50) & (bilgi<100)){portb=1;portd=0;}
          else if ((bilgi>100) & (bilgi<150)){portb=3;portd=0;}
          else if ((bilgi>150) & (bilgi<200)){portb=7;portd=0;}
          else if ((bilgi>200) & (bilgi<250)){portb=15;portd=0;}
          else if ((bilgi>250) & (bilgi<300)){portb=31;portd=0;}
          else if ((bilgi>300) & (bilgi<350)){portb=63;portd=0;}
          else if ((bilgi>350) & (bilgi<400)){portb=127;portd=0;}
          else if ((bilgi>400) & (bilgi<450)){portb=255;portd=0;}
          else if ((bilgi>450) & (bilgi<500)){portb=255;portd=1;}
          else if ((bilgi>500) & (bilgi<550)){portb=255;portd=3;}
          else if ((bilgi>550) & (bilgi<600)){portb=255;portd=7;}
            else if ((bilgi>600) & (bilgi<650)){portb=255;portd=15;}
}
fonksiyon2 ()
{
if (fpga<0.2)
{portd=0x01;output_high(pin_c2);delay_us(1000);output_low(pin_c2);delay_us(1000);}
else if ((fpga>=0.2)   & (fpga<0.5))
{portd=0x02;output_high(pin_c2);delay_us(1050);output_low(pin_c2);delay_us(950);}
else if ((fpga>=0.5)   & (fpga<0.8))
{portd=0x03;output_high(pin_c2);delay_us(1100);output_low(pin_c2);delay_us(900);}
else if ((fpga>=0.8)   & (fpga<1.1))
{portd=0x04;output_high(pin_c2);delay_us(1150);output_low(pin_c2);delay_us(850);}
else if ((fpga>=1.1)   & (fpga<1.4))
{portd=0x05;output_high(pin_c2);delay_us(1200);output_low(pin_c2);delay_us(800);}
```

```
else if ((fpga>=1.4)   & (fpga<1.7))
{portd=0x06;output_high(pin_c2);delay_us(1250);output_low(pin_c2);delay_us(750);}
else if ((fpga>=1.7)   & (fpga<2))
{portd=0x07;output_high(pin_c2);delay_us(1300);output_low(pin_c2);delay_us(700);}
else if ((fpga>=2)     & (fpga<2.3))
{portd=0x08;output_high(pin_c2);delay_us(1350);output_low(pin_c2);delay_us(650);}
else if ((fpga>=2.3)   & (fpga<2.6))
{portd=0x09;output_high(pin_c2);delay_us(1400);output_low(pin_c2);delay_us(600);}
else if ((fpga>=2.6)   & (fpga<2.9))
{portd=0x0a;output_high(pin_c2);delay_us(1450);output_low(pin_c2);delay_us(550);}
else if ((fpga>=2.9)   & (fpga<3.2))
{portd=0x0b;output_high(pin_c2);delay_us(1500);output_low(pin_c2);delay_us(500);}
else if ((fpga>=3.2)   & (fpga<3.5))
{portd=0x0c;output_high(pin_c2);delay_us(1550);output_low(pin_c2);delay_us(450);}
else if ((fpga>=3.5)   & (fpga<3.8))
{portd=0x0d;output_high(pin_c2);delay_us(1600);output_low(pin_c2);delay_us(400);}
else if ((fpga>=3.8)   & (fpga<4.1))
{portd=0x0e;output_high(pin_c2);delay_us(1650);output_low(pin_c2);delay_us(350);}
else if ((fpga>=4.1)   & (fpga<4.4))
{portd=0x0f;output_high(pin_c2);delay_us(1700);output_low(pin_c2);delay_us(300);}
else if ((fpga>=4.4)   & (fpga<4.7))
{portd=0x10;output_high(pin_c2);delay_us(1750);output_low(pin_c2);delay_us(250);}
else if ((fpga>=4.7)   & (fpga<5))
{portd=0x11;output_high(pin_c2);delay_us(1800);output_low(pin_c2);delay_us(200);}
else if (fpga==5) {portd=0xff;output_high(pin_c2);}
}
void main()
{
  set_tris_a(0x1f);     //port a nın giriş olarak atanması
  set_tris_b(0x00);
  set_tris_c(0x00);
  set_tris_d(0x00);
  portb=0x00;
  port_c=0x00;
  portd=0x00;

 while(1){
    fonksiyon1();
    fonksiyon2();
   }
    }
```

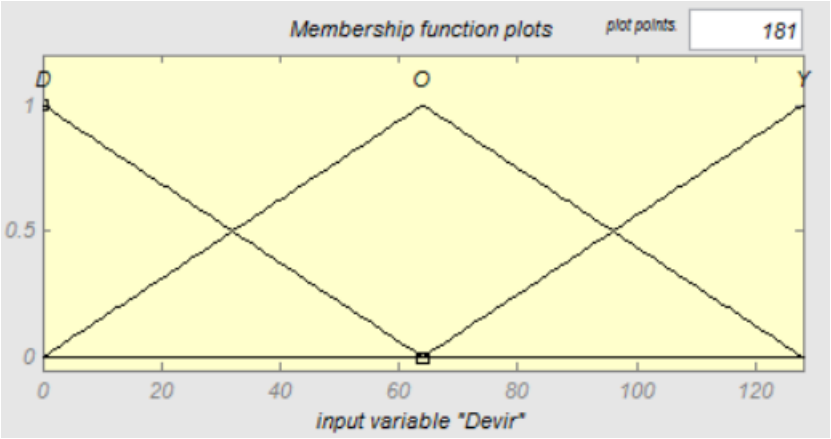
11. BULANIK DENETLEYİCİ YAPISI

DC motorlar üzerinde uygulanan yüke bağlı olarak motorun devri düşmekte ve çektiği akım artmaktadır. Moturu aynı devirde çalıştırabilmek için motora uygulanan DC gerilimin arttırılması gerekmektedir. Bu çalışmada bu nedenle DC Motor için tasarlanan bulanık mantık kontrolörün giriş parametreleri DC Motorun devri (D) ve DC moturun yükü (kg/cm) olarak belirlenmiştir. Bulanık denetleyicinin çıkış parametresi olarak DC motora uygulanacak olan gerilimi kontrol edebilmek için PWM referans gerilimi (U) belirlenmiştir. Tasarlanan kontrol sisteminin yapısı Şekil 11.1'de verilmiştir.

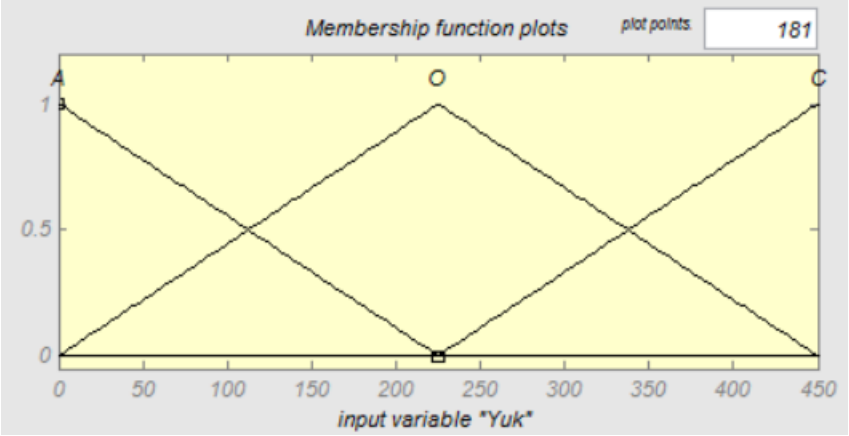


Şekil 11.1. Tasarlanan DC Motor Bulanık kontrol yapısı

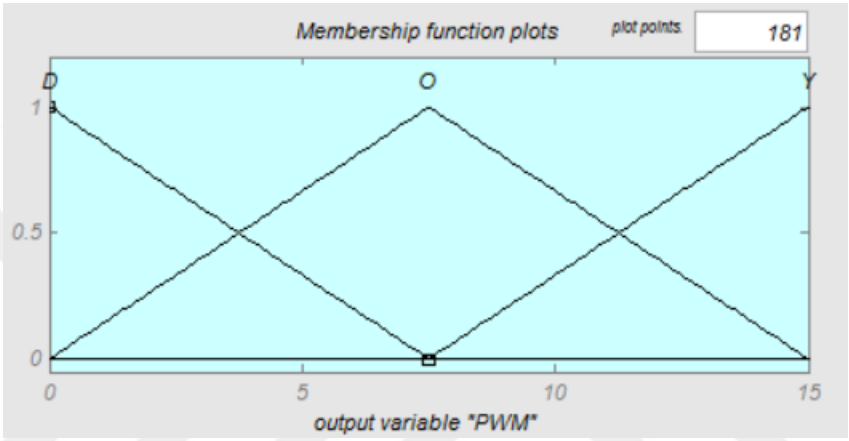
Bulanık mantık kontrolörün ilk bileşeni bulanıklaştırıcı, kesin giriş değerlerini ilgili bulanık kümenin [0,1] aralığındaki üyelik değerlerine dönüştürmektedir. Devir giriş değişkeni 0-128 aralığında {Düşük (D), Orta (O), yüksek (Y) } olmak üzere 3 dilsel ifadeye ayrılmıştır. Yük giriş değişkeni 0-4096 aralığında {Az (A), Orta (O), Çok (C) } olmak üzere 3 dilsel ifadeye ayrılmıştır. U çıkış değişkeni ise {Düşük (D), Orta (O), yüksek (Y) } olmak üzere 3 dilsel ifadeye ayrılmıştır. Giriş değişkenleri Devir ve Yük için üyelik fonksiyonlarının gösterimi Şekil 11.2 ve Şekil 11.3'te, çıkış değişkeni PWM referans gerilimi üyelik fonksiyonlarının gösterimi Şekil 11.4'te verilmiştir.



Şekil 11.2. Devir üyelik fonksiyonları



Şekil 11.3. Yük üyelik fonksiyonları



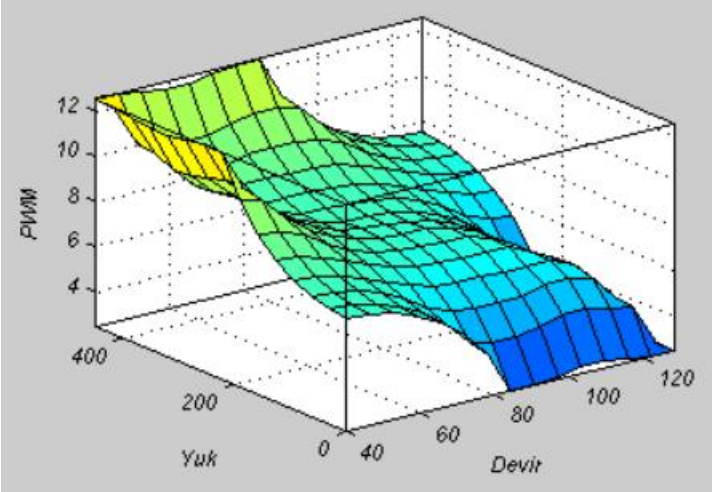
Şekil 11.4. PWM referans gerilimi üyelik fonksiyonları

Giriş ve çıkış değişkenleri için bulanık kural tablosu Çizelge 11.1'de verilmiştir. Bulanık kural tablosu Devir ve Yük değişimlerine bağlı olarak PWM referans gerilimini değiştirecek şekilde düzenlenmiştir.

Çizelge 11.1. Devir-Akım BK için kural tabanı

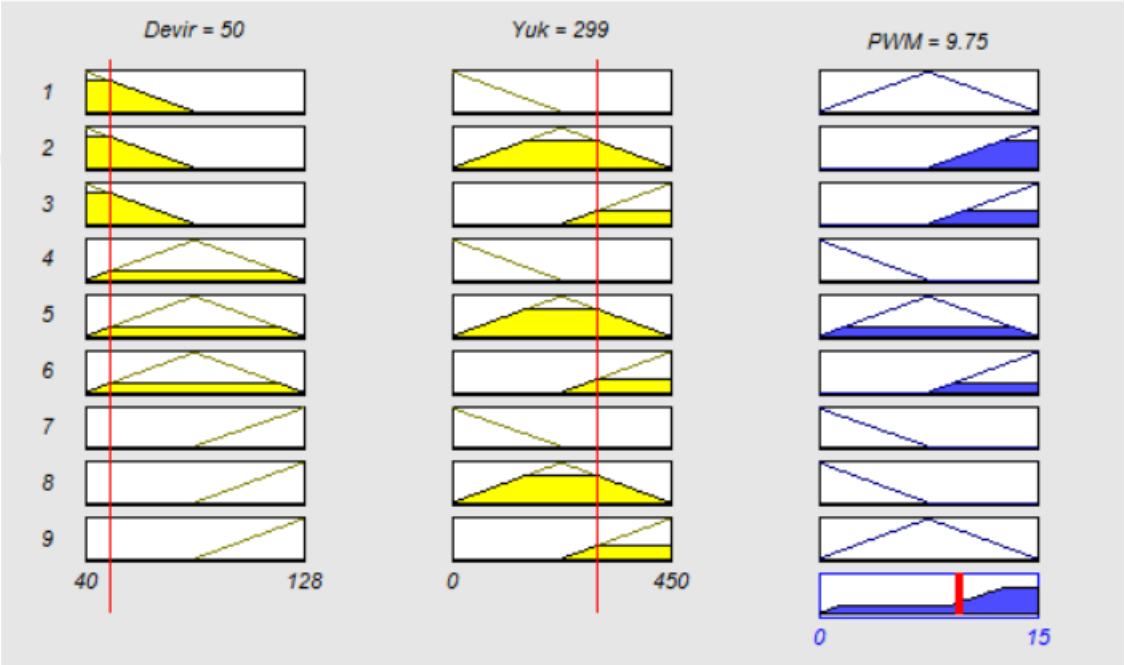
		Devir		
		D	**O**	**Y**
Yük	**A**	O	D	D
	O	Y	O	D
	C	Y	Y	O

Devir ve Akım giriş değişkenlerine göre PWM çıkışının değişimi Şekil 11.5'te gösterilmiştir



Şekil 11.5. Devir ve yüke göre PWM Değişimi

Giriş, çıkış değişkenleri ve bulanık kural tablosu verilen bulanık kontrol sistemi "Matlab&Fuzzy Logic Toolbox" yazılımı kullanılarak oluşturulmuştur. Örnek olarak Devir için 50, Akım için 755 değerleri girildiğinde PWM değeri 6.25 olarak değeri elde edilmektedir (Şekil 11.6).



Şekil 11.6. Devir ve Yüke göre durulaştırma İşlemi

11.1. Devir- Yük Bulanık Kontrolörün FPGA'da Programlanması

Bulanık mantık kontrolör donanımsal olarak tanımlanmadan önce bu işlem için kullanılan verilerin gösteriminin seçilmesi duyarlılık açısından önemlidir. FPGA giriş olarak sadece sayısal girişleri kabul eder. Uygulama safhasına ilk olarak verilerin girişe hazırlanması yani sayısal biçime dönüştürülmesi gerekmektedir. FPGA içinde kullanılabilen iki çeşit sayı formatı vardır. Bunlardan biri kayan noktalı sayı formatı

ikincisi ise sabit noktalı sayı formatıdır. FPGA içinde yapılan uygulamalar bu iki sayı tipine göre yapılmak zorundadır. Bulanık mantık kontrolör programlanırken sabit noktalı sayı formatı kullanılmıştır.

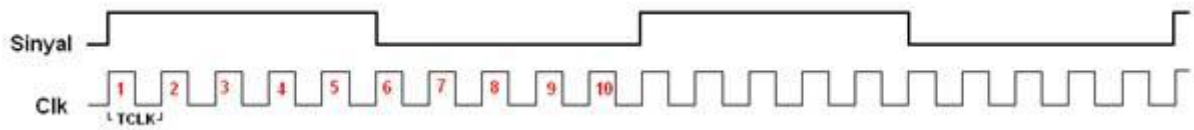
IEEE standartlarına göre sabit sayı gösterimi işaret, tam ve ondalıklı kısım olarak ayrılmıştır. Sabit noktalı sayıların sunumumu için internette farklı paketler bulunmaktadır. Bunlar fix_std.vhdl (Altera'nın sunduğu) ve fixed_pkg.vhdl dosyalarıdır. Oluşturulan devrede fized_pkg.vhdl kullanılmıştır. Sabit noktalı sayı gösteriminde iki türlü noktalı sayı bulunur. SFixed ve Ufixed dir. Sfixed ve Ufixed sayı tipleri numeric_std içindeki unsigned ve signed tiplerini kullanır. Ufixed ve Sfixed std_logic veri dizisi şeklindedir. Ufixed işaretsiz sayı, Sfixed ise pozitif ve negatif veri tipinin ikili gösterimidir. Kontrolör sistemi tasarlanırken sfixed formatında 17 bitlik bir veri formatı kullanılmıştır.

Giriş ve çıkış arayüzleri bulanık kontrolün çekirdek kısmı ile DC kontrol sistemi ile bağlantı kurulmaktadır. 12 bitlik akım bilgisi GPIO_1 bağlantısı kullanılarak ADC kartı üzerinden FPGA'ya aktarılmaktadır. Bu bilgi FPGA üzerindeki işleme tabii tutularak daha önce yapılan ölçümlerle elde edilen ampirik formül ile yük (kg/cm) bilgisine dönüştürülmektedir. Bu işlem için aşağıdaki denklem uygulanmaktadır.

Yük=Akım * 10

Ayrıca GPIO_0(11) bağlantısı kullanılarak tek bitlik devir bilgisi alınmaktadır. Tek bitlik devir bilgisinin periyod bilgisinin elde edilmesi için bir edge_detector devresi kullanılmıştır.

Sinyal periodu, input sinyalinin iki yükselen ucu (rising edge) arasındaki sistem clock sayısı ile sistem clock'unun periodu çarpılarak bulunabilir (Şekil 11.7).



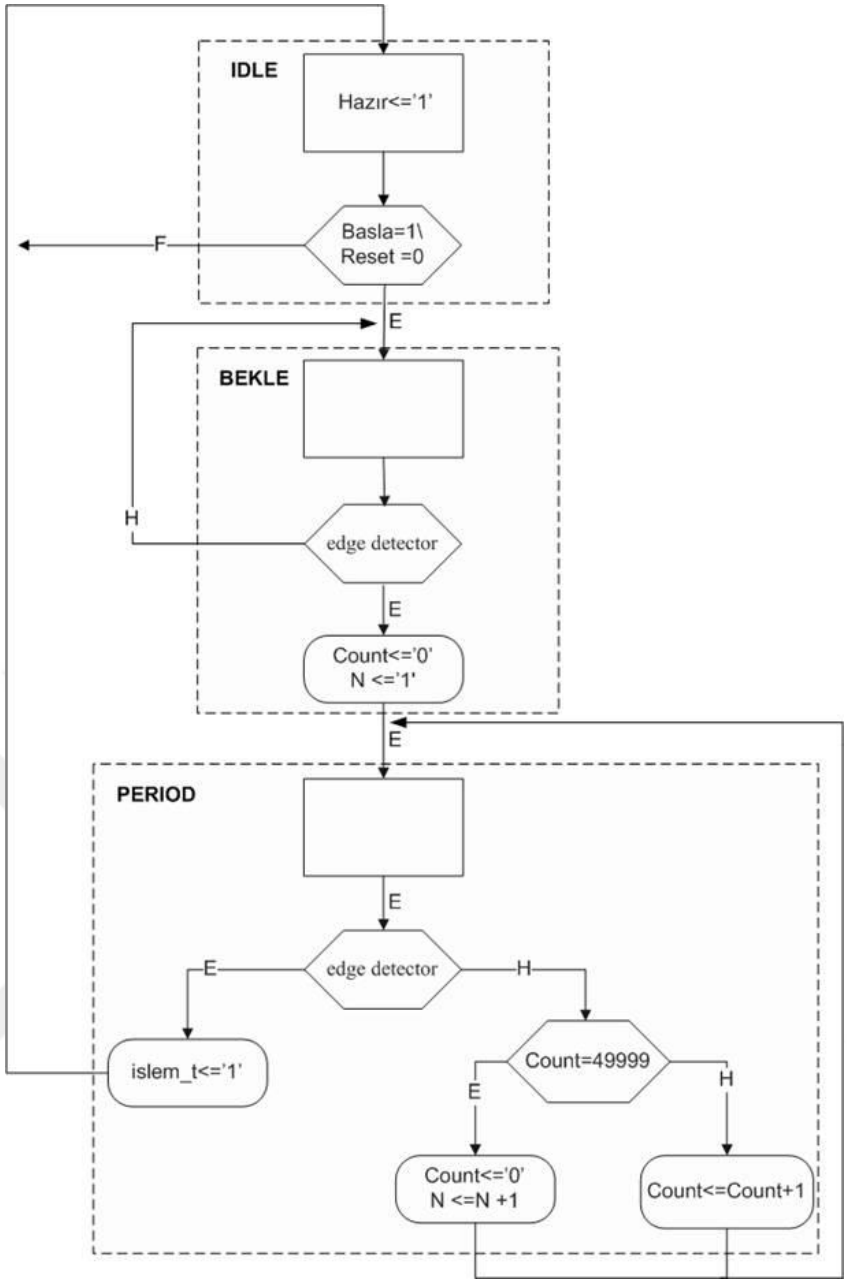
Şekil 11.7. Sinyal Periyodunun CLK'a göre değişimi

Input(giriş) sinyalinin perioduna Toutput, Input sinyalinin iki yükselen ucu arasındaki sistem clock saysına N ; System clockun perioduna Tclk der isek ;

Tout= N x Tckl olur.

Altera DE2-70 sistem clock'uğunu 50 Mhz (20 ns)olarak alınmıştır.

Edge Detector'un akış şeması Şekil 11.8'de verilmiştir.

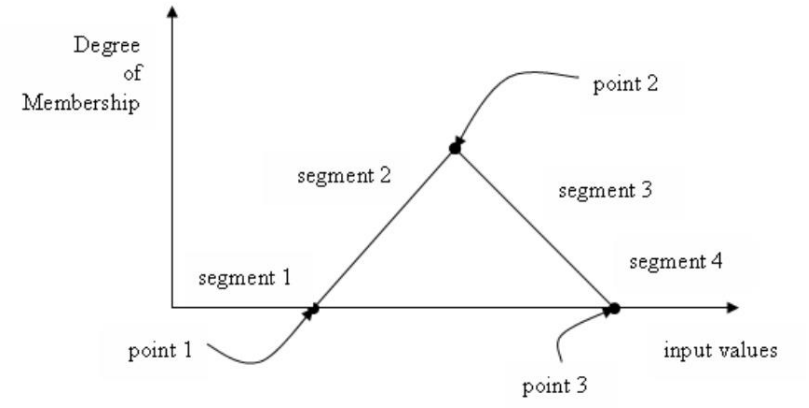


Şekil 11.8. Edge Detector Akış şeması

DC motoru kontrolün devrini sabit tutmak için motor gerilimi PWM ile kontrol edilmektedir. FPGA üzerinde PWM oluşturmak için bir PWM modülü tasarlanarak kullanılmıştır. PWM modülü 4 bitlik duty cycle oran bitine göre PWM sinyalini ayarlamaktadır.

11.2. Bulanıklaştırma Arayüzü

Bulanıklaştırıcı bileşeninin işlevi kesin giriş değerlerini bulanık değerlere çevirmektir. Bu çalışmada üçgensen üyelik fonksiyonu seçilmiştir. Bir üçgensel üyelik fonksiyonu üç nokta ve iki eğim ile etkili biçimde tanımlanabilir (Şekil 11.9).



Şekil 11.9. Üçgensel üyelik fonksiyonunun sunumu

Bu dört segmentin üyelik derecelerinin hesaplanması aşağıdaki gibi hesaplanır.

Segment 1,"input value"<point 1, degree of membership (μ)=0;

*Segment 2, point 1<"input value"<point2, degree of membership (μ)=(input value-point1)*slope1;*

*Segment 3, point 2<"input value"<point3, degree of membership (μ)=1-(input value-point2)*slope2;*

Segment 4,"input value">point 3, degree of membership (μ)=0;

11.3. Çıkarım bileşeni

Bulanık çıkarım bileşeni mamdani min-max çıkarım yöntemi ile kural tabanından çıkarım yapmaktadır. Tipik olarak bir kural IF-THEN formunda "IF is x a AND y is B THEN z is c" olarak verildiğinde iki bulanık set arasında ki AND ifadesi min ile gösterilir.

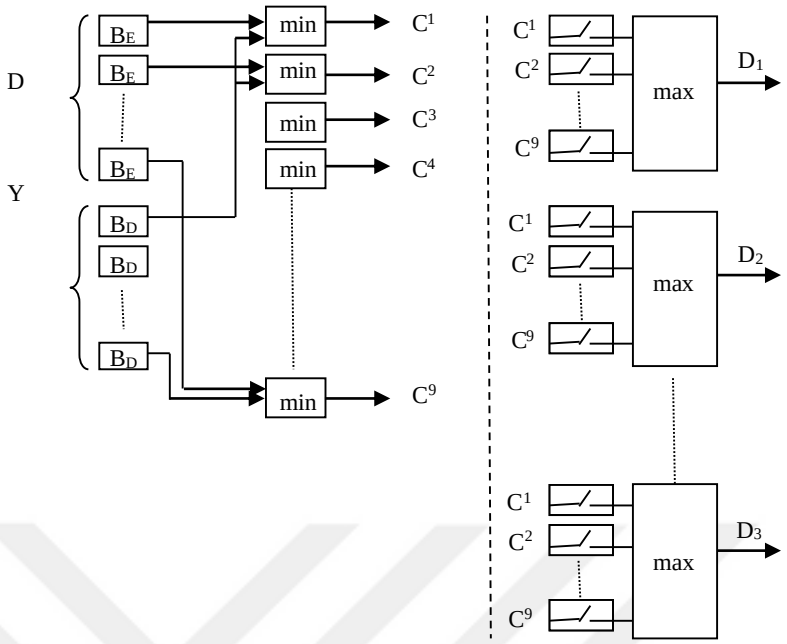
c=min(a,b)

Eğer bir çıkış iki kuraldan oluşuyor ise iki kural arasındaki OR işlemi max işlemi ile ifade edilir.

IF (x is a_1, AND y is b_1) OR (x is a_2, AND y is b_2) THEN z is c;

c=max(min(a_1,b_1),min(a_2,b_2))

Şekil 11.10'da çıkarım mekanizmasının donanımsal olarak gerçekleştirmesinin nasıl adapte edildiği gösterilmektedir.



Şekil 11.10. Çıkarım mekanizmasının donanımsal modeli

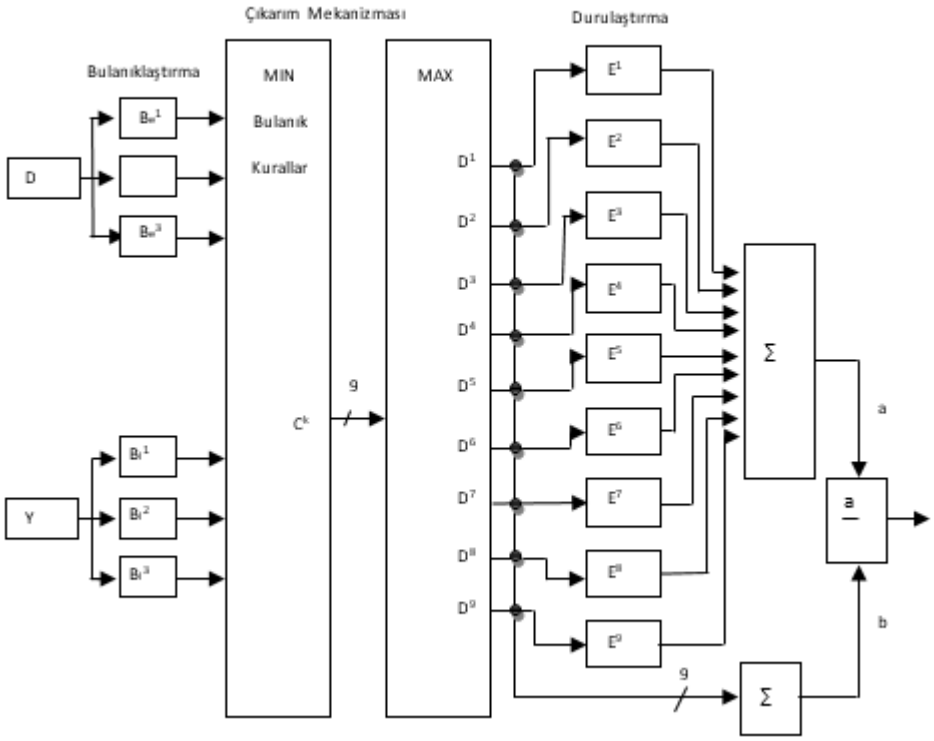
Modelde görüldüğü gibi öncelikle hata ve hata değişiminin dilsel ifadelerinin minimum değerleri bulunmaktadır. Daha sonra bulunan bu minimum değerler çıkış kuralının FAM tablosuna göre gerekli switchler ON değerine getirilerek çıkış değerleri oluşturulmaktadır.

11.4. Durulaştırıcı bileşeni

Durulama bileşeni karar verme biriminden gelen bulanık bir bilgiden bulanık olamayan ve kontrolde kullanılacak gerçek değerlerin elde edilmesini sağlar. Bu işlem için ağırlıklaştırılmış ortalama durulaştırma metodu kullanılmaktadır. Bu yöntem de girişlerden elde edilen bütün bulanık değerler ile üyelik değerleri kullanılarak aşağıdaki formül ile durulama yapılmaktadır.

$$z^* = \frac{\sum \mu_c(z).z}{\sum \mu_c(z)}$$

İşlem sırasında sıfıra bölme hatasını engellemek için bölen sıfıra eşit olduğunda Δu çıkışı 0 olarak alınmıştır. Bulanık kontrol bileşenlerinden oluşturulan kontrol sisteminin genel blok diyagramı Şekil 11.11'de verilmiştir.

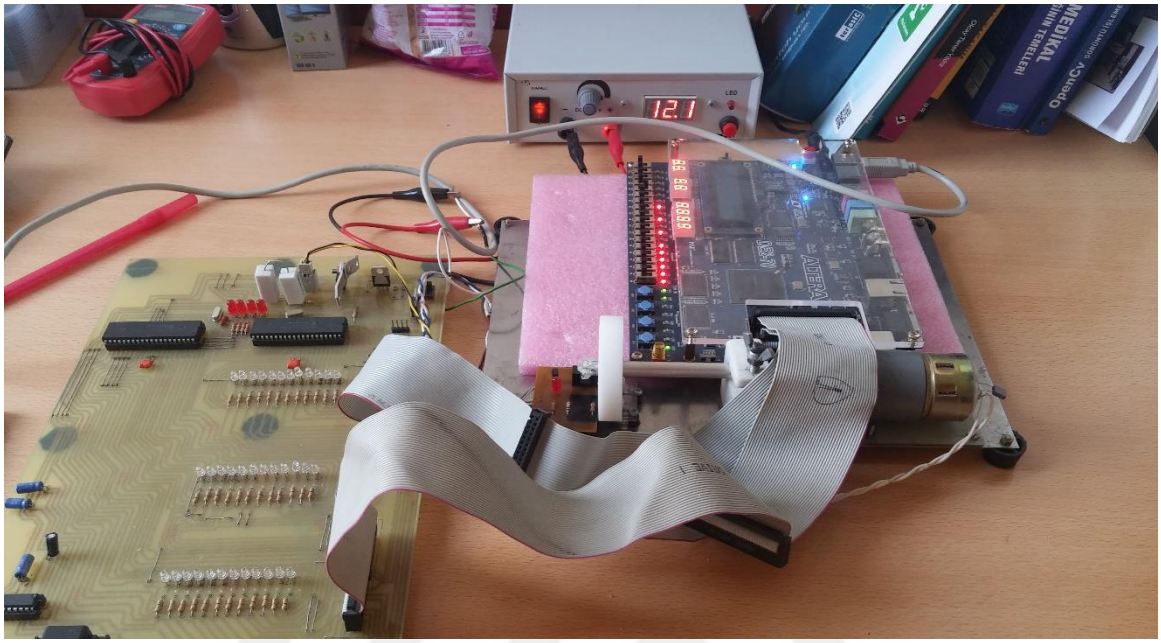


Şekil 11.11. Bulanık mantık kontrolörün genel blok diyagramı

VHDL ile kodlaması yapılan bulanık kontrolörün Modelsim programı ile simülasyon sonuçları alınmıştır. Bulanık kontrolörün lojik sentezleme, yerleştirme ve route işlemleri Altera Quartus programı ile yapılmıştır. Cyclone II EP2C70F896 FPGA için 68416 lojik elementin 23546'i kullanılarak sentezlenmiştir.

12. DC MOTOR FPGA İLE KONTROL DENEY DÜZENEĞİ

Tasarımı yapılan ve gerçekleştirilen deney seti Şekil 12.1'de görüldüğü gibi bağlantıları yapılarak denemeler yapılmıştır.



Şekil 12.1. Devir ve Akıma göre durulaştırma İşlemi

Yapılan deney çalışmasında DC motor FPGA tabanlı bulanık kontrolör ile denetimi yapılmıştır. Yapılan deneyde motorun yüksüz, 11 kg/cm yükün %33, %66 ve %90 oranın da uygulanarak çalıştırılmıştır. Bu çalışmalarda elde edilen grafikler Şekil 13.16-Şekil 13.18'deki gibi elde edilmiştir.

Verilen grafiklerde 1. Bölge DC motorun yüksüz çalışması, 2. Bölge %33 yük altında çalışması, 3. Bölge %66 yük altında çalışmasını, 4. Bölge %90 yük altında çalışması verilerini içermektedir.

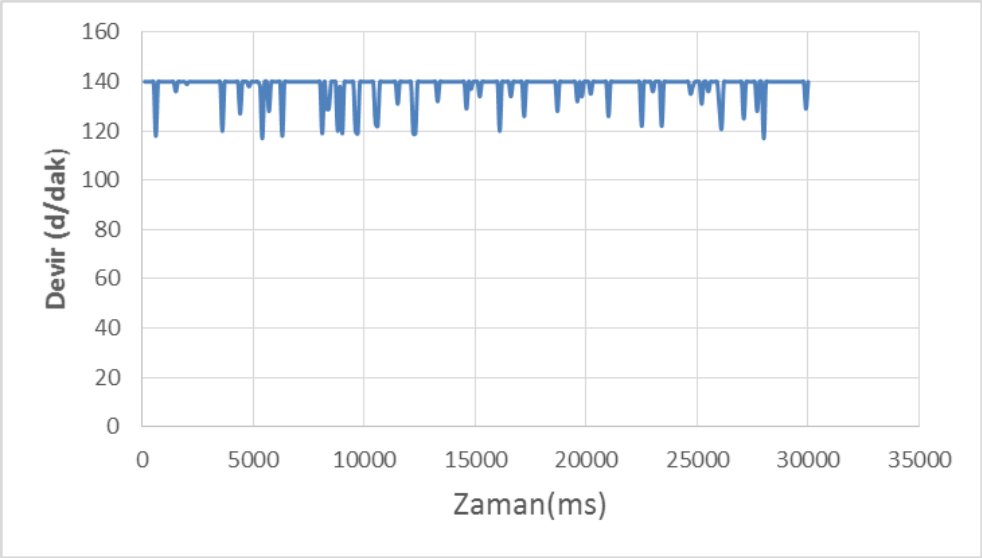
13. SONUÇ VE ÖNERİLER

13.1. Birinci deney setinde elde edilen sonuçlar

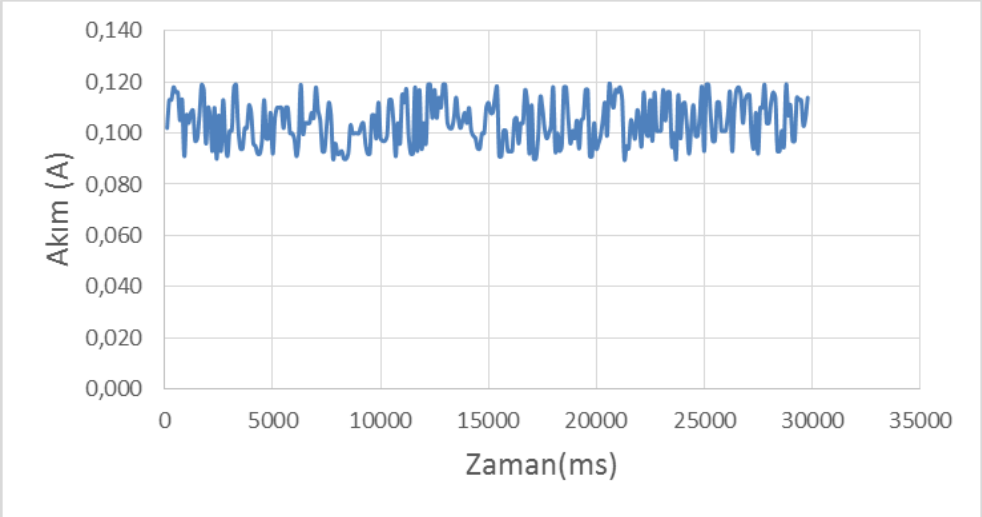
Birinci deney seti ile elde edilen grafikler Şekil 13.1– Şekil 13.19'da verilmiştir.

13.1.1. Motor Mili Boşta İken:

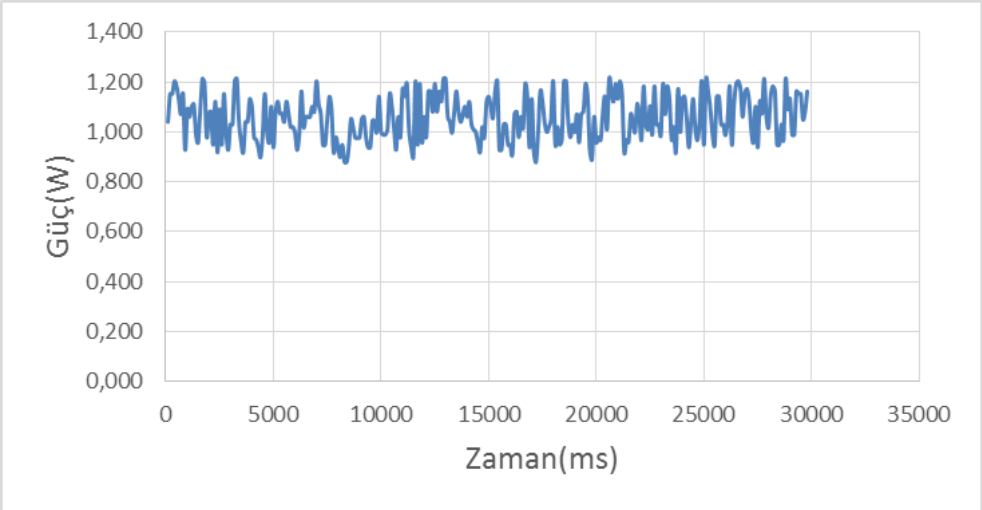
Motor boşta iken; 10,20 V besleme geriliminde devir=140 d/dk, Akım=0,12 A ve Güç=1,10 W değerleri elde edilmiştir (Şekil 13.1-Şekil 13.3).



Şekil 13.1. Devir-Zaman grafiği



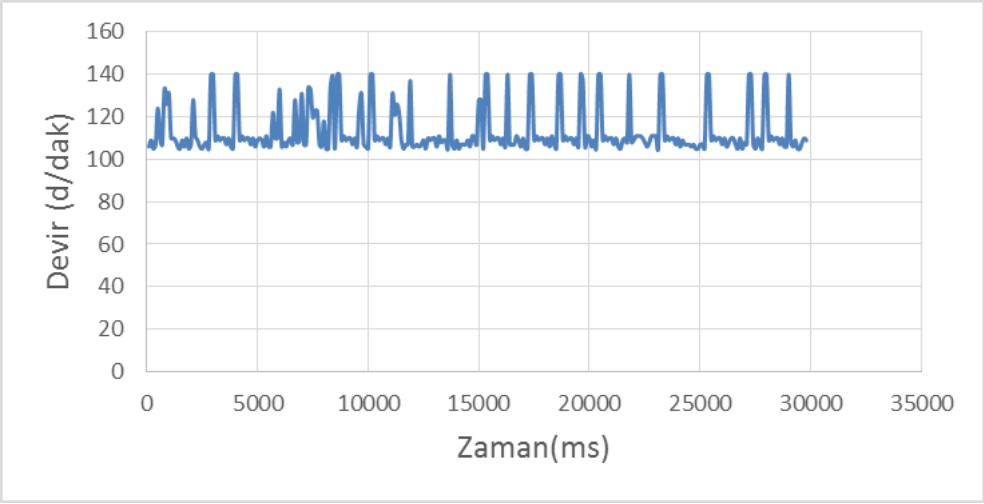
Şekil 13.2. Akım-zaman grafiği



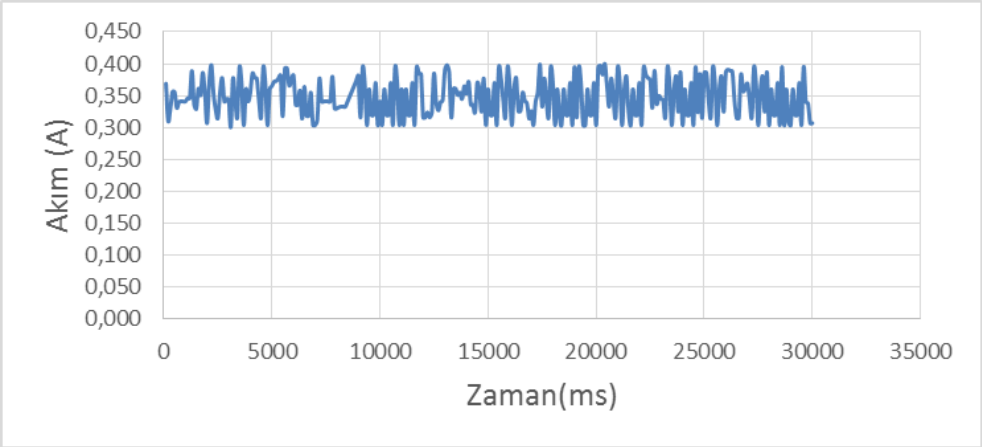
Şekil 13.3. Güç-zaman grafiği

13.1.2. Motor Mili %33 Yüklü İken

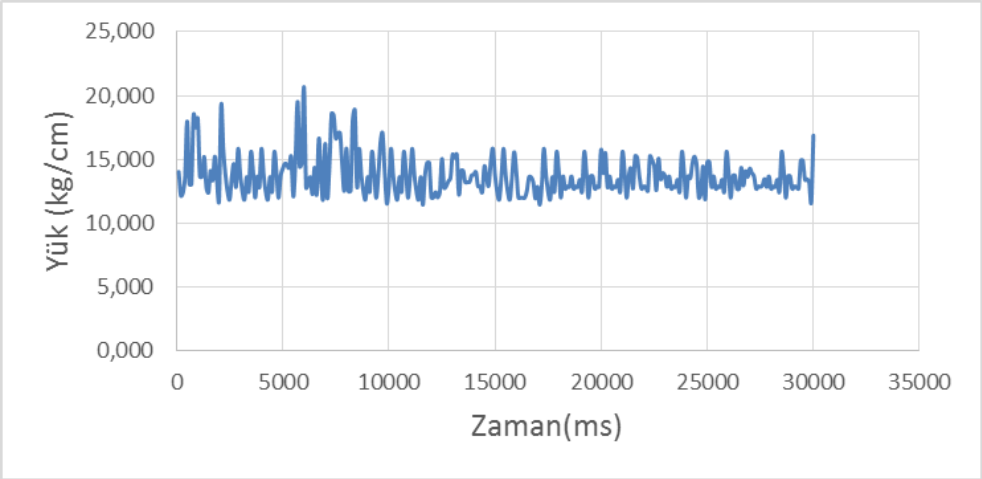
Motor %33 yükte iken; besleme gerilimi= 10,2 V, devir=110 d/dk, Akım=0,3-4,10 A, Yük=13-22 kg/cm ve Güç=2,30-3,81 W değerleri elde edilmiştir (Şekil 13.4-Şekil 13.7).



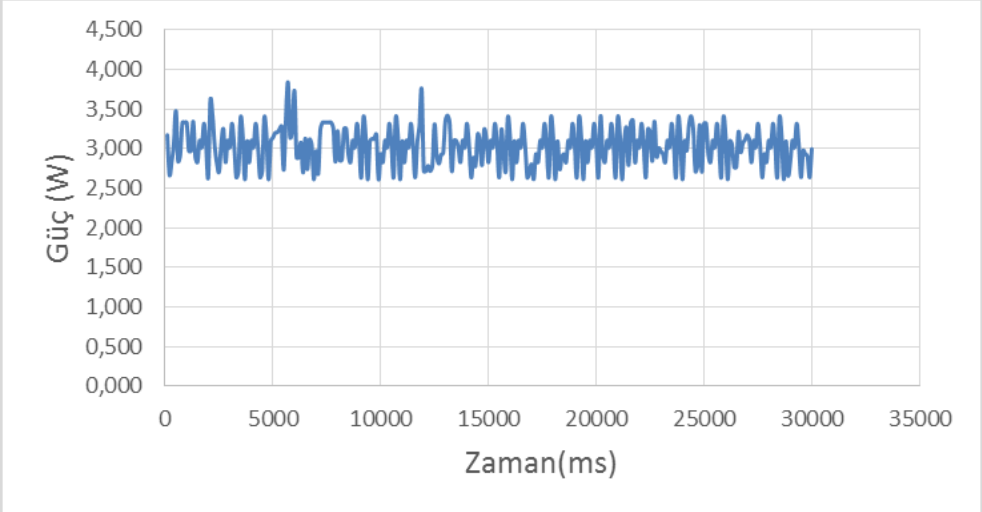
Şekil 13.4. Devir-zaman grafiği



Şekil 13.5. Akım-zaman grafiği



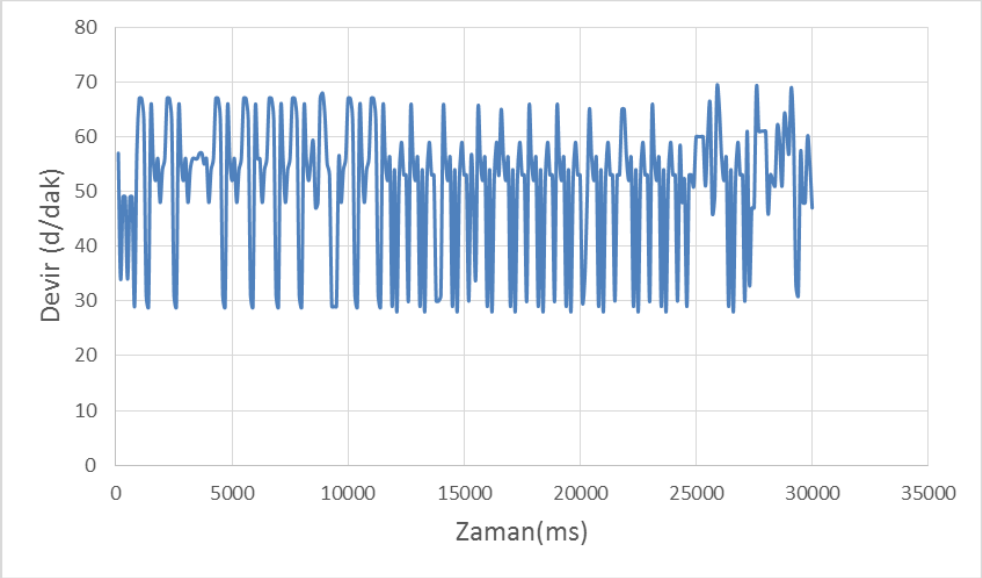
Şekil 13.6. Yük-zaman grafiği



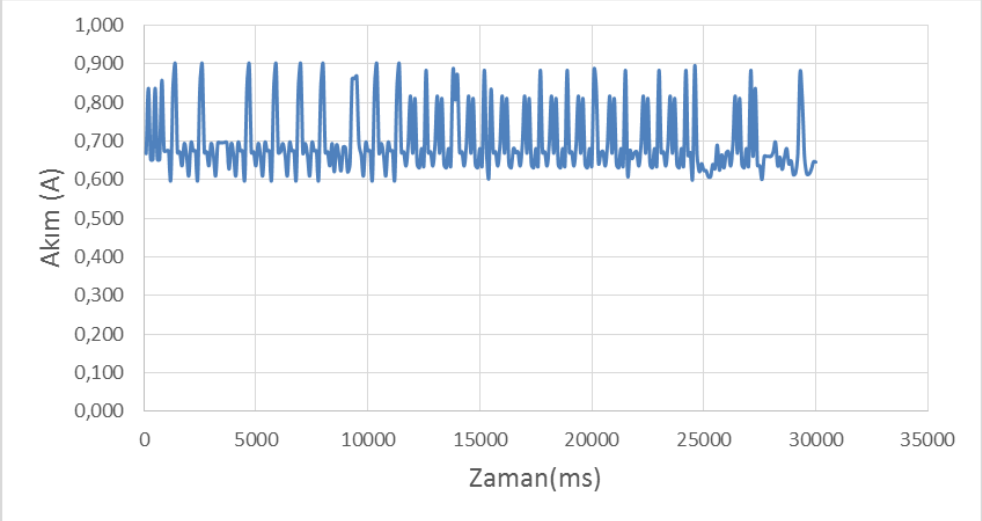
Şekil 13.7. Güç-zaman grafiği

13.1.3. Motor Mili %66 Yüklü İken

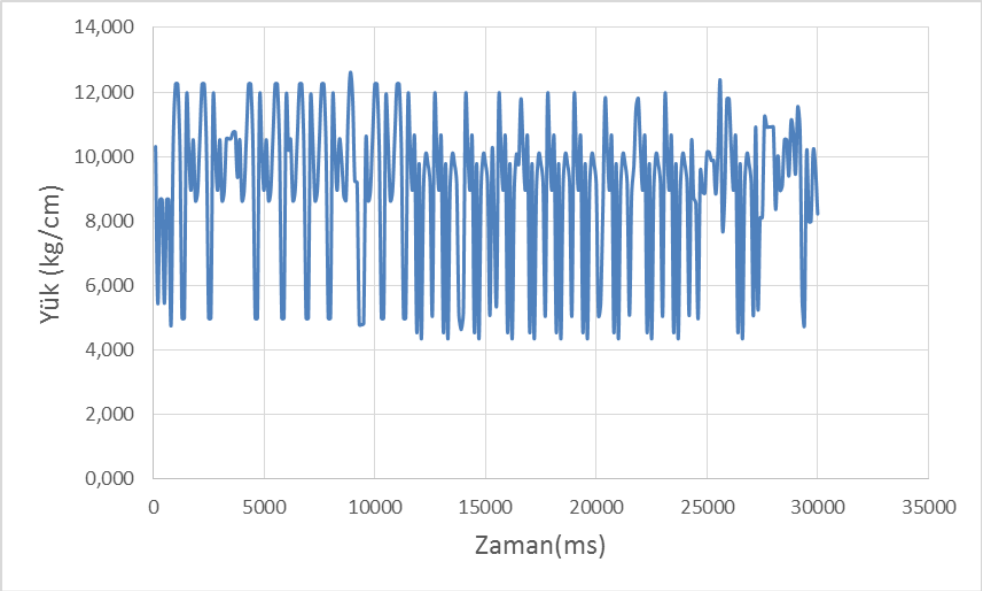
Motor %33 yükte iken; besleme gerilimi= 10,2 V, devir=29-69 d/dk, Akım=0,59-0,91 A, Yük=4,26-12,40 kg/cm ve Güç=3,80-4,51 W değerleri elde edilmiştir (Şekil 13.8-Şekil 13.11).



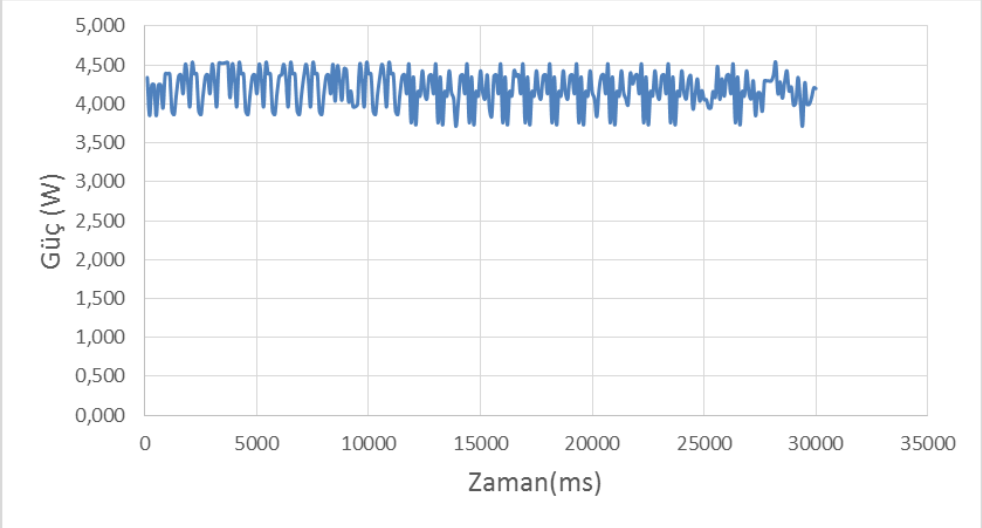
Şekil 13.8. Devir-zaman grafiği



Şekil 13.9. Akım-zaman grafiği



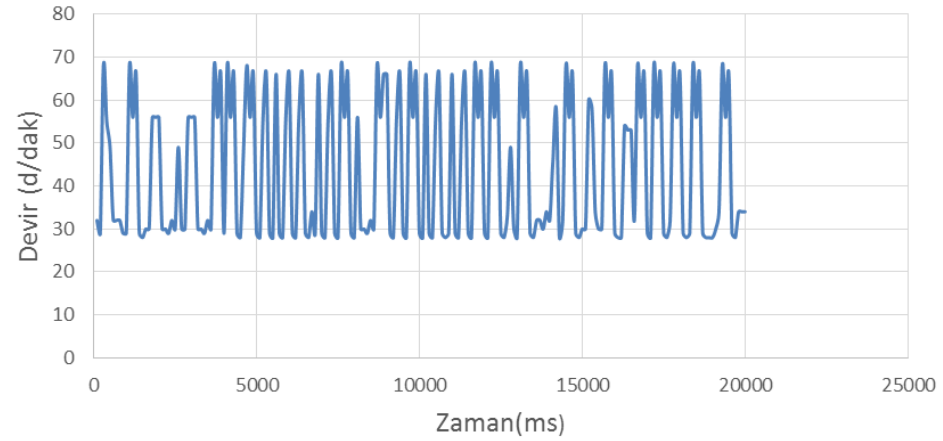
Şekil 13.10. Yük-zaman grafiği



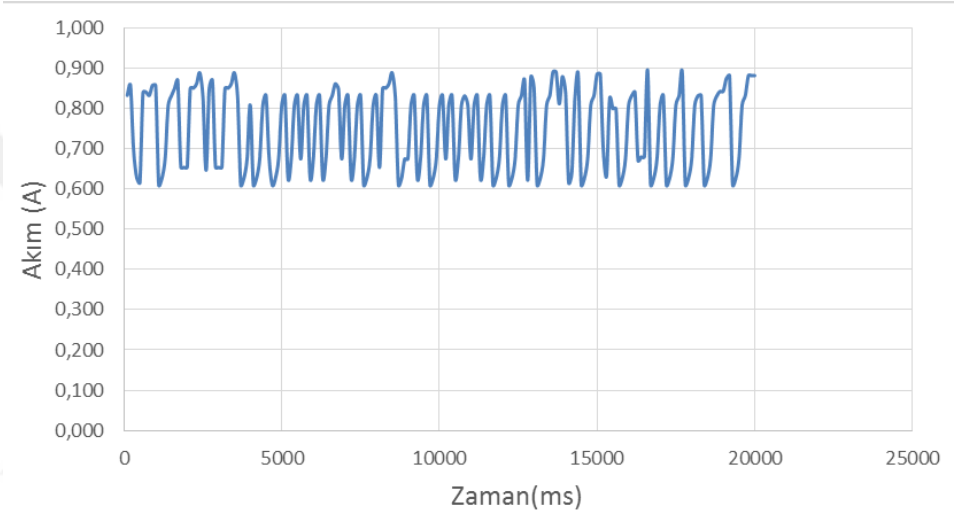
Şekil 13.11. Güç-zaman grafiği

13.1.4. Motor Mili %90 Yüklü İken

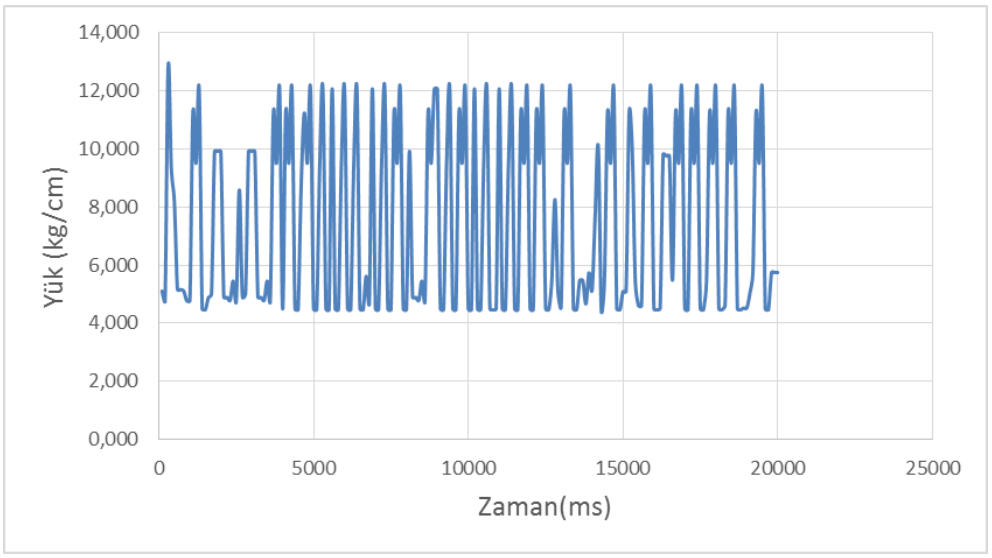
Motor %33 yükte iken; besleme gerilimi= 10,2 V, devir=30-70 d/dk, Akım=0,06-0,92 A, Yük=4,30-12,50 kg/cm ve Güç=3,50-4,51 W değerleri elde edilmiştir (Şekil 13.12-Şekil 13.15).



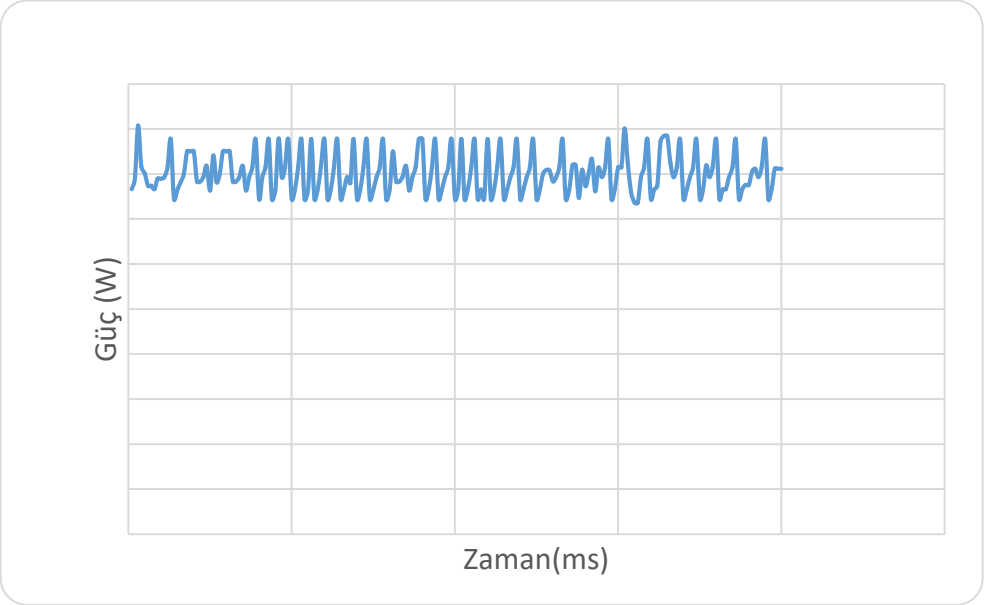
Şekil 13.12. Devir-zaman grafiği



Şekil 13.13. Akım-zaman grafiği



Şekil 13.14. Yük-zaman grafiği

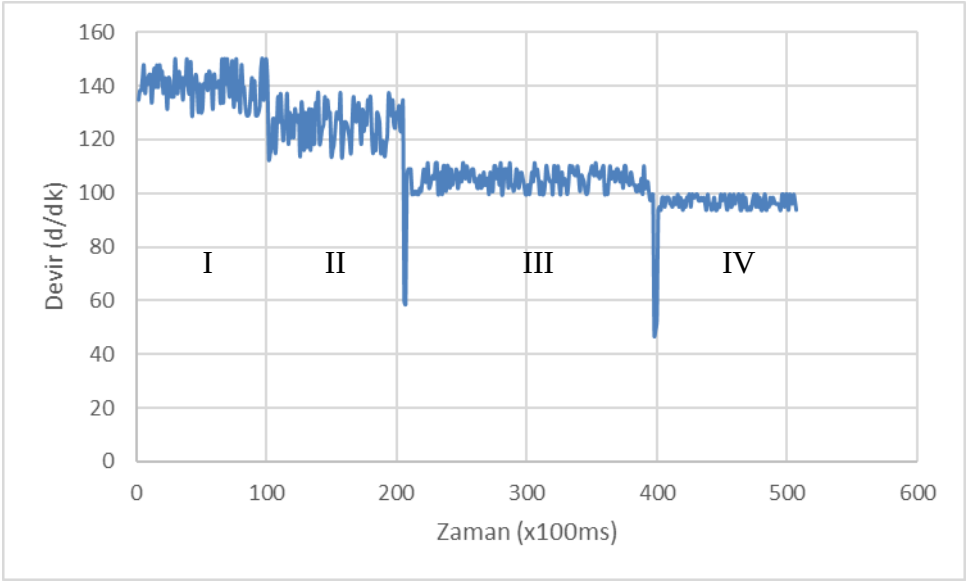


Şekil 13.15. Güç-zaman grafiği

13.2. İkinci Deney seti ve FPGA Tabanlı Bulanık Denetleyici Deney Sonuçları

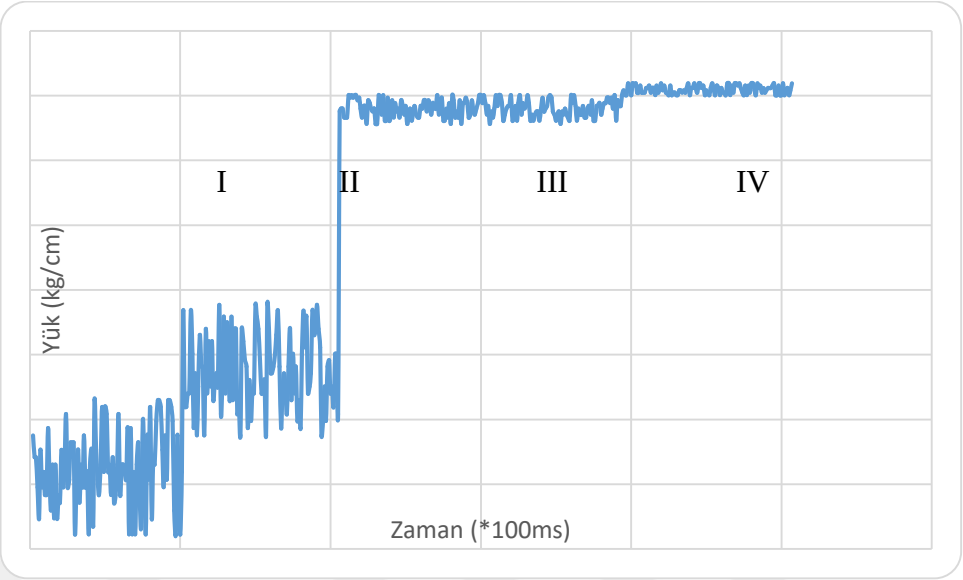
İkinci deney seti kullanılarak FPGA da programlanan bulanık denetleyici ile 4 bölge olark motorun boşta, %33 yükye, %66 yükte ve %90 yükte çalışma deney grafikleri Şekil 13.16-Şekil 13.19 arasında verilmiştir.

Şekil 13.16'de devir zaman grafiği incelendiğinde 1. bölgede ortalama 140 devir/dk, 2. Bölgede devrin ortalama 130 devir/dk, 3. Bölgede 110 devir/dk, 4. Bölgede 98 devir/dk olduğu görülmektedir. Bu bölgeler arasındaki geçişte devirin hızla düştüğü (özellikle 2. Bölgeden 3. Bölgeye ve 3. Bölgeden 4. Bölgeye geçişte) ve denetleyici tarafından verilen değerlere yükseltildiği görülmektedir.



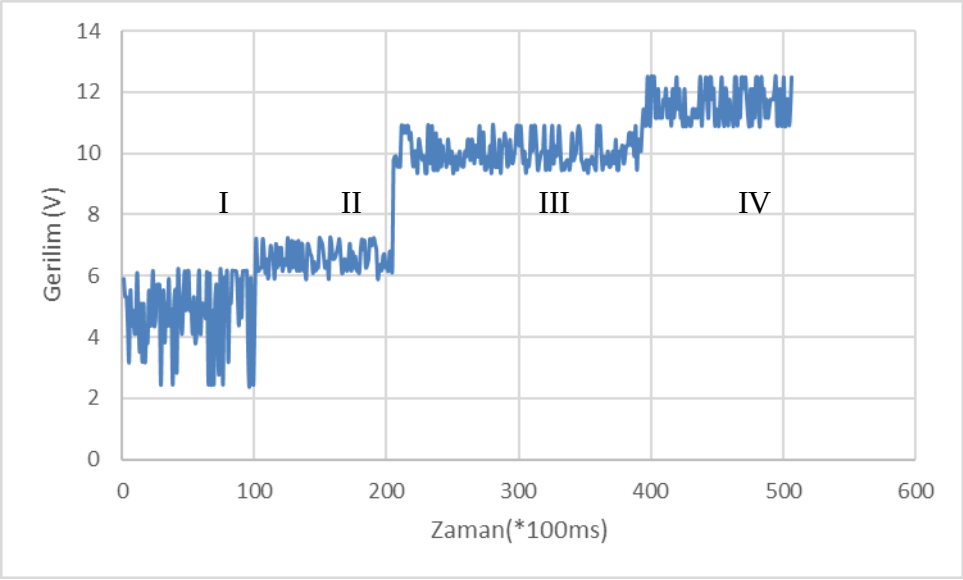
Şekil 13.16. Bulanık mantık denetleyici ile elde edilen devir-zaman grafiği

Şekil 13.17'de yük zaman grafiği incelendiğinde 1. bölgede ortalama 1 kg/cm, 2. Bölgede akımın ortalama 3 kg/cm, 3. Bölgede 6,7 kg/cm, 4. Bölgede 7,2 kg/cm olduğu görülmektedir. Bu bölgeler arasındaki geçişte akımın yüke bağımlı olarak yükseldiği görülmektedir.



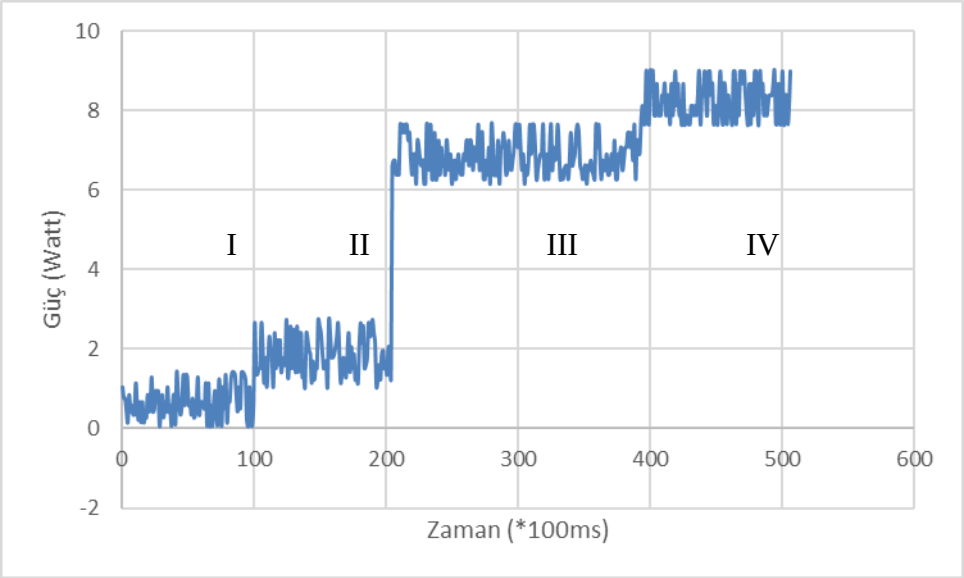
Şekil 13.17. Bulanık mantık denetleyici ile elde edilen akım-zaman grafiği

Şekil 13.18'de gerilim zaman grafiği incelendiğinde 1. bölgede ortalama 5 V, 2. Bölgede akımın ortalama 6,5 A, 3. Bölgede 10 V, 4. Bölgede 11,5 V olduğu görülmektedir. Yük değişimlerine göre denetleyici tarafından devir ve yüke bağlı olarak motora uygulanan gerilim arttırılmaktadır. Bu sayede yük altında DC motor istenilen devir ile çalışabilmesi sağlanmaktadır.



Şekil 13.18. Bulanık mantık denetleyici ile elde edilen gerilim-zaman grafiği

Şekil 13.19'te DC motorun yüksüz ve yük altındaki güç değişimi grafiği verilmiştir. Yük artışının motor üzerinde ki güç tüketimini arttırdığı görülmektedir.



Şekil 13.19. Bulanık mantık denetleyici ile elde edilen güç-zaman grafiği

Sonuç olarak:

- Kontrol sistemi olmadan sabit gerilim altında yüke bağlı olarak motor devrinin düştüğü görülmektedir
- Denetleyici sistem devir ve yük değişimine bağlı olarak motora uygulanan gerilimi yükseltmekte dolayısıyla motorun gücü de arttığı görülmektedir,
- Bu denetleyici sistem yükten dolayı motorun verimli çalışmasını sağladığı görülmektedir,
- Denetleyicinin FPGA üzerinde programlanmış olması yük değişimlerine bağlı olarak çok hızlı tepkime süresi ile motorun verimli çalıştığı görülmektedir,

Öneriler:

- Sistemde bulanık denetleyicinin parametreleri arttırılarak denetleme performansı iyileştirilebilir
- Elde edilen sistem çok hızlı yük değişimleri ile çalışan motorlar üzerinde uygulanabilir,
- Farklı denetleyici yöntemleri de uygulanarak verimlilik karşılaştırılması yapılabilir,
- Farklı DC motorlar ile denetleyici parametreleri kendini uyarlayabilir bir sistem tasarlanabilir,

KAYNAKLAR

Astarloa A, Lázaro J, Bidarte U, Jiménez J, Zuloaga A, 2009. FPGA technology for multi-axis control systems. Mechatronics, 19, 2, 258-268.

Chekired F, Larbes C, Rekioua D, Haddad F, 2011. Implementation of a MPPT fuzzy controller for photovoltaic systems on FPGA circuit. Energy Procedia, 6, 541-549.

Chen J-S, Lin I-D, 2002. Toward the implementation of an ultrasonic motor servo drive using FPGA. Mechatronics, 12, 4, 511-524.

Cirstea M, Dinu A, McCormick M, Nicula D. A VHDL success story: Electric drive system using neural controller. VHDL International Users Forum Fall Workshop, 2000. Proceedings, 118-122.

Cirstea M, Khor J, McCormick M. FPGA fuzzy logic controller for variable speed generators. Control Applications, 2001.(CCA'01). Proceedings of the 2001 IEEE International Conference on, 301-304.

Bulanık Mantık ve Bulanık Küme Teorisi, 2000. Erişim. Erişim adresi, https://cobanoglu.wikispaces.com/file/view/bulanikmantik.pdf.

de Jesús R-TR, Gilberto H-R, Iván T-V, Carlos J-CJ, 2004. FPGA based on-line tool breakage detection system for CNC milling machines. Mechatronics, 14, 4, 439-454.

Günal Ü, 1997. Bulanık Mantık. Otomasyon dergisi, 55, 50-55.

Hasanien HM, 2011. FPGA implementation of adaptive ANN controller for speed regulation of permanent magnet stepper motor drives. Energy Conversion and Management, 52, 2, 1252-1257.

Hassan MY, Sharif WF, 2007. Design of FPGA based PID-like fuzzy controller for industrial applications. IAENG International Journal of Computer Science, 34, 2, 192-198.

Hsu C-F, Lee B-K, 2011. FPGA-based adaptive PID control of a DC motor driver via sliding-mode approach. Expert Systems with Applications, 38, 9, 11866-11872.

Ichikawa S, 2005. A Study on Real-Time Control System with FPGA.

Klarenbach JOK-C, Krah J, 2008. FPGA based field oriented current controller for high performance servo drives. Proceedings PCIM Europe, 2008.

Kongmunvattana A, Chongstivatana P. A FPGA-based behavioral control system for a mobile robot. Circuits and Systems, 1998. IEEE APCCAS 1998. The 1998 IEEE Asia-Pacific Conference on, 759-762.

Lee C-C, 1990. Fuzzy logic in control systems: fuzzy logic controller. I. IEEE Transactions on systems, man, and cybernetics, 20, 2, 404-418.

Mano MM, 2010. Sayısal Tasarım. 5, Istanbul, Literatür Yayıncılık, p.

MEGEP, (2011). Megep Modülleri, Elektrik Elektronik Teknolojisi; Doğru Akım Motorları. Ankara, MEB.

MEGEP, (2013). Megep Modülleri; Yenilenebilir Enerji Teknolojileri; Doğrultucu ve Evirici Devreleri. Ankara, MEB.

Petko M, Uhl T. Embedded controller design-mechatronic approach. Robot Motion and Control, 2001 Proceedings of the Second International Workshop on, 195-200.

Poorani S, Priya TU, Kumar KU, Renganarayanan S, 2005. FPGA based fuzzy logic controller for electric vehicle. Journal of the Institution of Engineers, 45, 5, 1-14.

Şahin S, 2004. FPGA ile yapay sinir ağının donanımsal gerçeklenmesi, Kocaeli Üniversitesi.

WEB1. Erişim tarihi Ekim 2016. Erişim adresi, http://en.wikipedia.org/wiki/Pulse-width_modulationPWM.

WEB2. Erişim tarihi Ekim 2016. Erişim adresi, https://tr.wikipedia.org/wiki/FPGA.

WEB3. Erişim tarihi Ekim 2016. Erişim adresi, http://fpganedir.com/FPGA/index.php.

WEB4. Erişim tarihi Ekim 2016. Erişim adresi, http://www.altera.com, DE2_70_User_manual_v105.

Yazıcı S, 2004. FPGA İle Bulanık Mantığın Donanımsal Gerçeklemesi, Kocaeli Üniversitesi.

Zadeh LA, 1988. Fuzzy logic. Computer, 21, 4, 83-93.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Fatma Betül ARICI
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Çumra 02/07/1987
Telefon	**:**	0507 428 68 56
Faks	**:**	-
e-mail	**:**	arc.fatmabetl@gmail.com

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	Atatürk Lisesi, Selçuklu, Konya	2004
Üniversite	:	Selçuk Üniversitesi	2010
Yüksek Lisans	:		
Doktora	:		

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2011	Milli Eğitim Bakanlığı	Bilişim Teknolojileri Öğretmeni

UZMANLIK ALANI

Web Tasarımı, Bulanık Mantık, C yazılım

YABANCI DİLLER

İngilizce

BELİRTMEK İSTEĞİNİZ DİĞER ÖZELLİKLER

YAYINLAR