KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK ve HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

FPGA TABANLI 5 EKSENLİ MOBİL ROBOT KOLU TASARIMI ve GERÇEKLENMESİ

HASAN KARCI

KOCAELİ 2013

KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK ve HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

FPGA TABANLI 5 EKSENLİ MOBİL ROBOT KOLU TASARIMI ve GERÇEKLENMESİ

HASAN KARCI

Doç.Dr. Ali TANGEL
Danışman, Kocaeli Üniv.

Yrd.Doç.Dr. Mehmet YAKUT
Jüri Üyesi, Kocaeli Üniv.

Yrd.Doç.Dr. Halil İbrahim ESKİKURT
Jüri Üyesi, Sakarya Üniv.

Tezin Savunulduğu Tarih: 08.07.2013

ÖNSÖZ ve TEŞEKKÜR

Günümüz teknolojisi hızla ilerlerken birçok bilim alanlarını kapsayacak şekilde robot teknolojisi de gelişmektedir. Robotlar giderek insan yaşamının ortak bir parçası olmaktadır. İnsanlar birçok işlerini robotlar sayesinde kolay, hızlı ve istenilen ölçüde gerçekleştirebilmektedir. Özellikle endüstride kullanılan robotlar ve robot kolları üretimde istenilen hız ve düşük maliyet gereksinimini karşılayabilmektedir.

Bu çalışmada mobil platform üzerine monte edilmiş robot kolu, cisimleri boylarına göre ayırt edip, yerlerini değiştirebilmektedir. Mobil robot kolunun yazılımsal tasarımı Xilinx ISE programı ile yapılarak FPGA üzerinde gerçeklenmiştir. Simülasyonlar ise ModelSim programında yapılmıştır.

Tez çalışmamda hiçbir yardımı esirgemeyen ve bilgilerini benimle paylaşan değerli hocam Doç.Dr. Ali Tangel'e sonsuz teşekkür ederim. Ayrıca hayatımın bu seviyeye gelmesinde hiçbir maddi ve manevi desteği esirgemeyen aileme sonsuz minnet duygularımı sunarım.

Mayıs–2013 Hasan KARCI

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER DİZİNİ ve KISALTMALAR

T_{CLK} : Saat sinyalinin periyodu, (s)

Kısaltmalar

AC : Alternating Current (Dalgalı Akım)
ARM : Advanced RISC Machines (Gelişmiş RISC Makineleri)
CLB : Configurable Logic Block (Yapılandırılabilir Lojik Blok)
CPU : Central Processing Unit (Merkezi İşlem Birimi)
DC : Direct Current (Doğru Akım)
DOF : Degree of Freedom (Serbestlik Derecesi)
DSP : Digital Signal Processing (Sayısal İşaret İşleme)
EN : Enable (Yetkilendirme)
FPGA : Field Programmable Gate Array (Alan programlanabilir Kapı Dizileri)
HDL : Hardware Description Language (Donanım Tanımlama Dili)
HV : High Voltage (Yüksek Gerilim)
IOB : In Out Block (Giriş Çıkış Bloğu)
KSOM : Kohonen's Self Organizing Map (Kohonen'in Kendi Organizasyon Haritası)
LCD : Liquid Crystal Display (Likit Kristal Ekran)
LUT : Look up Table (Bak Oku Tablosu)
LV : Low Voltage (Düşük Gerilim)
PID : Proportional Integral Differential (Oransal, Tümlevsel, Türevsel)
PWM : Pulse With Modulation (Darbe Genişlik Modulasyonu)
RISC : Reduced Instruction Set Computer (İndirgenmiş Komut Setli Bilgisayar)
VHDL : VHISC Hardware Description Language (VHISC Donanım Tanımlama Dili)
VHISC : Very High Speed Integrated Circuit (Çok Yüksek Hızlı Tümleşik Devreler)
XST : Xilinx Synthesis Technology (Xilinx Sentezleme Teknolojisi)

FPGA TABANLI 5 EKSENLİ MOBİL ROBOT KOLU TASARIMI ve GEÇEKLENMESİ

ÖZET

Bu çalışmada, mobil bir aracın üzerine monte edilmiş 5 eksenli bir robot kolu ile cisimleri bir ultrasonik sensor aracılığı ile tanıyıp boyutlarına göre ayırt edebilen bir mobil robot kolu sistemi gerçeklenmiştir. Ultrasonik sensordan alınan veriye göre mobil aracın hız ve konum kontrolü ile robot kolunun konum kontrolüne ait sayısal tasarım tek bir FPGA yongası üzerinde gerçeklenmiştir. Mobil robot kolu üzerindeki DC motor ve DC-servo motorların sürücü devreleri ayrıca tasarlanarak, motorların hız ve konum kontrolü yine VHDL kodu ile oluşturulan PWM kontrol sinyalleri aracılığıyla sağlanmıştır. Bu çalışmada PWM sinyallerini üretmede frekans bölme tekniği kullanılmıştır. FPGA tabanlı tasarımların tasarımcıya sunduğu paralel işlem yapabilme yeteneği sayesinde, mobil robot kolu üzerindeki ünitelerin durumları aynı anda kontrol edilebilmektedir. VHDL ile oluşturulan tasarım kodu Xilinx ISE tasarım platformu aracılığı ile sentezlenerek, donanımın çalışması Modelsim benzetim programı aracılığıyla doğrulanmıştır. Daha sonra, yapılan tasarım Spartan–3 FPGA geliştirme kiti üzerinde gerçeklenmiştir. FPGA geliştirme kiti, gerçeklenen mobil robot donanımının bir parçası olarak mobil platform üzerine monte edilmiştir. Mobil robot kolunun beklenen tüm işlevlerini sorunsuz bir şekilde yerine getirdiği deneysel olarak da gözlenmiştir. Ayrıca, tasarımın literatürdeki benzer robot kolu tasarımları ile karşılaştırması yapılmıştır.

Anahtar kelimeler: FPGA, Hız Kontrolü, Mobil Robot, Ultrasonik Konum Belirleme.

DESIGN AND IMPLEMENTATION OF A 5-DOF MOBILE ROBOT ARM BASED ON FPGA

ABSTRACT

In this study, A mobile 5-DOF robot arm, which can distinguish objects according to their dimensions using an ultrasonic sensor module is designed and implemented. Digital hardware design of speed and position control of the vehicle carrying the robot arm and the position control of the robot arm are implemented on a single FPGA chip. The driver circuits of the DC motors and the RC motors mounted on the mobile robot arm system are also realized, and the motor speed and position controls are handled through the PWM signals obtained by a specific VHDL module. Frequency division technique is used to produce the PWM signals. The concurrent controls of the units mounted on the arm are possible due to parallel execution ability offered by FPGA based designs. The Modelsim program is used for VHDL code simulations. Then, the real FPGA implementations are done on a Spartan-3 FPGA evaluation board using Xilinx ISE tools. This evaluation board is also mounted on the vehicle platform as a part of the mobile robot arm system. The test results show that the robot arm is able to accomplish all expected functions successfully. Finally, the designed robot arm is also compared to similar ones exist in the literature.

Keywords: FPGA, Mobile Robot, Speed Control, Ultrasonic Position Control.

GİRİŞ

Günümüz teknolojisinde robotların önemi giderek artmakta ve birçok endüstriyel, askeri ve diğer uygulamalarda insanın yerini almaktadır. Robotlar, özellikle robot kolları endüstride yaygın olarak kullanılmaktadır. Robotların çalışabilmesi için elektronik bir sisteme ihtiyaç vardır. Bu elektronik sistem içerisinde genelde işlemci, algılayıcılar ve sürücü devreleri yer alır. Donanımın genel hatlarını değiştirmeden sistemin güncellenmesi maliyetin düşmesinde önemli bir etkendir. Maliyetin yanında zaman tasarrufu için hızlı yazılım ve donanım adaptasyonu gereklidir. Geleneksel devre tasarımlarında, herhangi bir uygulamada kullanılacak devreler üretim esnasında yapılandırılır iken, alan programlanabilir teknolojisinde ise üretimden sonra kapı seviyesinde ara bağlantılar yazılımsal olarak yeniden yapılandırılabilirler [1]. Alan programlama teknolojisinin tasarımcıya sunduğu bu esneklik sayesinde, tekrar programlanabilen, yani uygulama geliştirmeye açık tasarımlar mümkün olabilmektedir.

Robot teknolojisinde ise robotun yapacağı işler esneklik kazandıkça robotun kullanılabilirliği artmaktadır. FPGA (Field Programmable Gate Array)'nın yeniden yapılandırılması ile robotik sistemlerdeki esneklik ihtiyacı giderilebilmektedir. Genellikle robotik sistemlerin mikrodenetleyici veya DSP (Digital Signal Processor) ile kontrol edildiği göze çarpmaktadır [2]. Mikrodenetleyici ve DSP ile oluşturulan robot kontrol sistemleri büyük kapasite ve hızlı yapılandırılabilme olanağını yeterince sağlayamamaktadır [3].

Çok eksenli bir robot kolu ile birlikte, kolu taşıyan mobil aracın ve bu aracın üzerindeki bir takım algılayıcıların bilgilerinin okunması, bunun yanında karar algoritmalarının gerçeklenmesi aynı anda düşünüldüğünde, önümüze hem karmaşık hem de esneklik isteyen bir tasarım problemi çıkmaktadır. Bu da tasarımın çok sayıda bileşen içermesini gerektirmektedir.

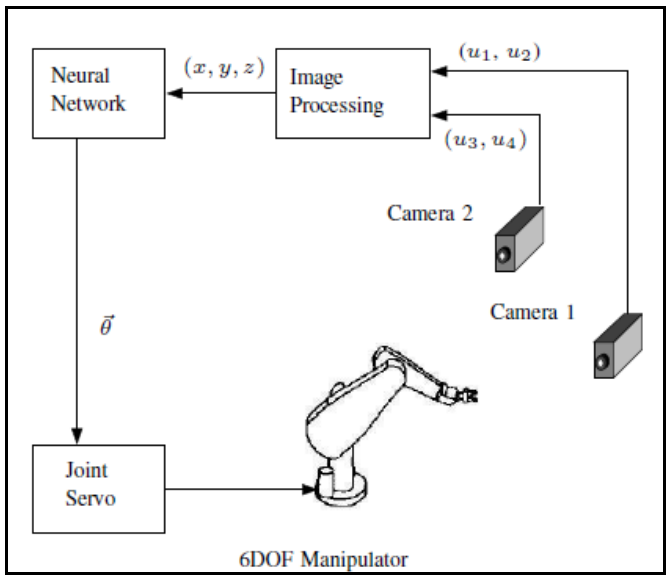
Birçok modülün eşzamanlı olarak kullanılmasıyla, modüler robotlar kontrol edilebilirlik açısından daha karmaşık hale gelebilmektedir. Bu nedenle tekdüze

çalışan mimariler, birçok bileşen içeren robot tasarımında, robotların çeşitli fonksiyonları gerçekleştirmesinde yeterli olamamaktadır [4]. Bu çalışmada çok eksenli mobil robot uygulaması ve prototip üretimi FPGA tabanlı esnek tasarım örneği olarak gerçeklenmiştir.

1. LİTERATÜR TARAMASI

1.1. Çalışma 1

"A Model-Free Redundancy Resolution Technique for Visual Motor Coordination of a 6 DOF Robot Manipulator" (Kumar ve diğ., 2007) adlı makalede 6 serbestlik derecesine sahip robot manipülatörün görsel motor koordinasyonları incelenmiştir. Şekil 1.1'de görsel motor koordinasyon yerleşimi verilmiştir.

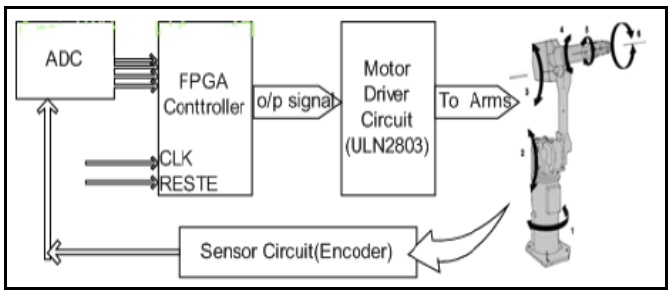


Şekil 1.1. Görsel motor koordinasyon yerleşimi

Bu sistem bir adet stereo kamera sistemi, bir adet manipülatör robot ve sistemi kontrol eden algoritmanın işlendiği işlemci biriminden oluşmuştur. Görüntü işleme birimi 4 boyutlu görüntü koordinat vektörü kullanılarak robot pozisyonu belirlemiş ve robotun hedef noktaya uzanmasını sağlamıştır. Görsel motor koordinasyonunda KSOM (Kohonen's self-organizing map) metodu kullanılmıştır. Ters kinematik çözümünün giriş çıkış uzay kümesinin KSOM algoritması ile sağlanabileceği bu çalışmada kanıtlanmıştır [1].

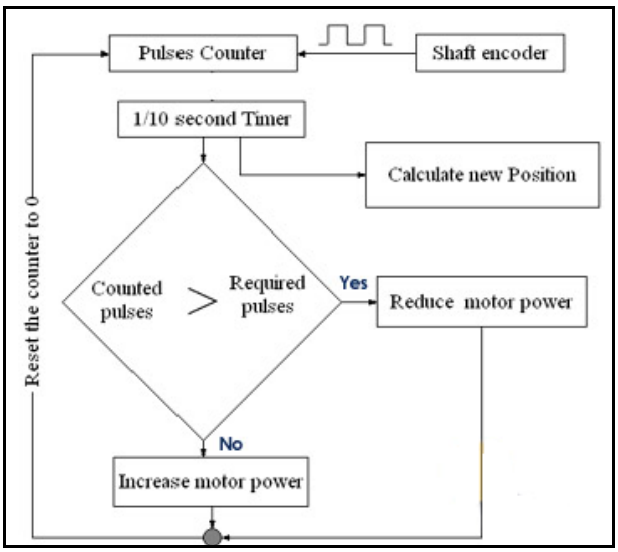
1.2. Çalışma 2

"Robot Arm Controller Using FPGA" (Meshram ve diğ., 2009) adlı çalışmada robot kolunun hareketlerinin kontrolü FPGA üzerinde tasarlanan sistem ile yapılmıştır. Robot kolu kontrol sistemi Şekil 1.2'de verildiği gibi bir adet robot kolu, motor sürücü devreleri, sensörler ve FPGA'dan oluşmuştur.



Şekil 1.2. Robot kolu kontrol blokları

VHDL (VHISC Hardware Description Language) kodu kullanılarak Spartan-II FPGA kitine kontrol sistemi yüklenmiştir. Robot kolunun konum almasını sağlamak için kullanılan servo ve adım motorlarının hareket kontrolü temel alınmıştır. FPGA ile robot arasında yer alan motor sürücü devresinde ise ULN280A entegresi kullanılmıştır.



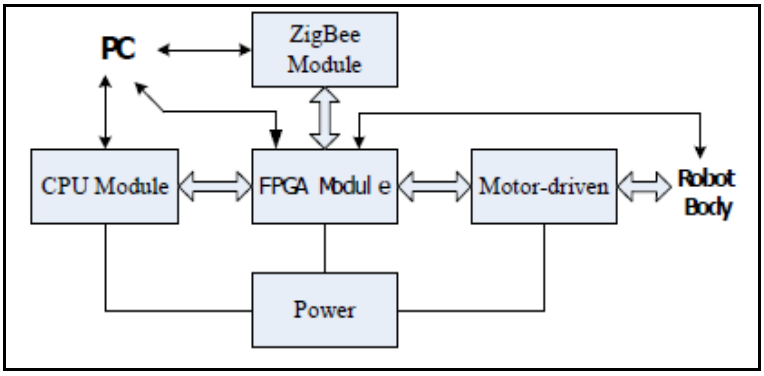
Şekil 1.3. Şaft kod çözücüden gelen sinyalin kontrol biriminde işlenmesi

Motor konumlarını FPGA kontrol sistemine geribildirim için şaft kod çözücüde üretilen sinyaller gönderilmiştir. Kod çözücü diski motorun şaftına bağlanarak motorla birlikte dönmesi sağlanmakta ve disk konumu kodlanarak kod çözücü devresine gönderilmektedir.

Geri bildirim olarak gelen sinyalin FPGA da tasarlanan kontrol devresinde işlenmesini anlatan akış şeması Şekil 1.3'te verilmiştir [3].

1.3. Çalışma 3

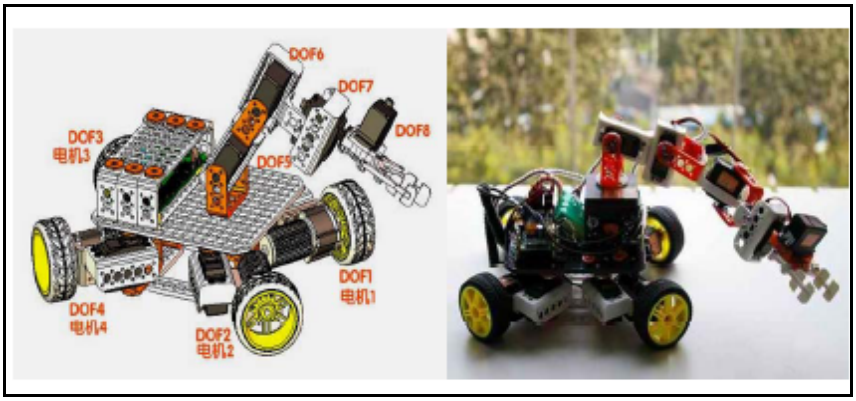
"Design of a Reconfigurable Robot Controller Based on FPGA" (Xu ve diğ., 2008) adlı çalışmada FPGA'nın dinamiksel yeniden yapılandırma prensibinin metot ve yapılandırma akışının tasarımı incelemiştir. Bu çalışmada, yeniden yapılanabilir robot kontrol metodu tasarımı için kontrol birimleri olarak ARM (Advanced RISC Machines) işlemci ve FPGA kullanılmıştır. Yeniden yapılanma, robotun birçok fonksiyonu gerçeklemesine olanak tanımıştır. Bu sayede esnek kullanımlar ve tasarımlar meydana gelmiştir.



Şekil 1.4. Robot kontrol modülünün mimarisi

Şekil 1.4'te verilen kontrol sisteminin mimarisinde ARM işlemci ve FPGA, mimarinin çekirdeğini oluşturmuştur. Donanımsal olarak da CPU (Central Processing Unit) modülü, FPGA modülü, Zig-Bee kablosuz modülü, DC (Direct Current) motor sürücü devre modülü ve güç kaynağı modülünü içermektedir. CPU modülüne çeşitli fonksiyonlar ve görevler yüklenmiş, işlemlerin yazılımsal kontrolü ile modüler robotun hareket kontrolü gerçek zamanlı olarak izlenerek insan ile makine arası ara yüzü, FPGA'nın yeniden yapılanmasıyla sağlanmıştır. Modüler

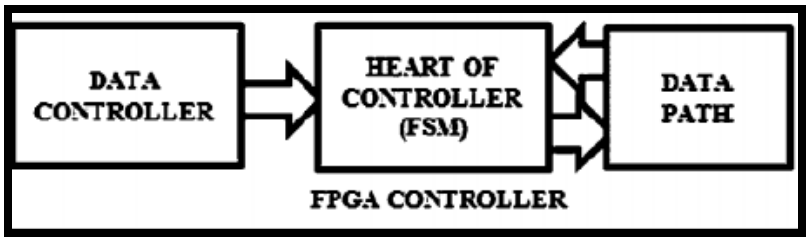
robota görevlerin bilgisayardan iletilmesini Zig-Bee kablosuz iletişim modülü sağlamaktadır. Modüler robotun vinç şeklindeki yapısı Şekil 1.5'te verilmiştir. [4]



Şekil 1.5. Modüler robot yapısı ve resmi

1.4. Çalışma 4

"FPGA Based Five Axis Robot Arm Controller" (Meshram ve Harkare, 2010) adlı makalede FPGA, H-köprüsü sürücü devresi ve sensör sürücü devresi kullanılarak beş eksenli robot için yazılım ve donanım tararımı ile hız ve konum kontrolü yaparak cisimleri toplamak ve yerleştirmek hedeflenmiştir. DC motorların hız kontrolünde dijital PWM (Pulse Width Modulation) kullanılmıştır.



Şekil 1.6. FPGA kontrol diyagramı

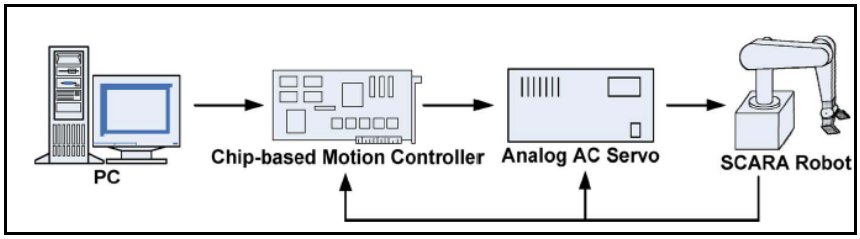
Şekil 1.6'da verilen FPGA kontrol diyagramından anlaşılacağı gibi beş eksenli robot kolunun kaynak kodu; veri kontrolü, kontrol sisteminin kalbi (FSM) ve veri yolu olarak üç ana parçaya ayrılmıştır. Bu parçalar da kendi içlerinde alt parçalara ayrılmıştır. FPGA'nın sağladığı esnek tasarım geliştirme sayesinde sistemin verimli bir şekilde çalışması için uygun bileşen tasarımı yapılmıştır.

Veri kontrol bileşeni içerisinde 5 adet motor için çeşitli frekanslarla aynı doluluk boşluk oranına sahip PWM'ler üretilebilmiş, ayrıca aynı frekanslarda farklı doluluk boşluk oranına sahip PWM'ler de üretilebilmiştir.

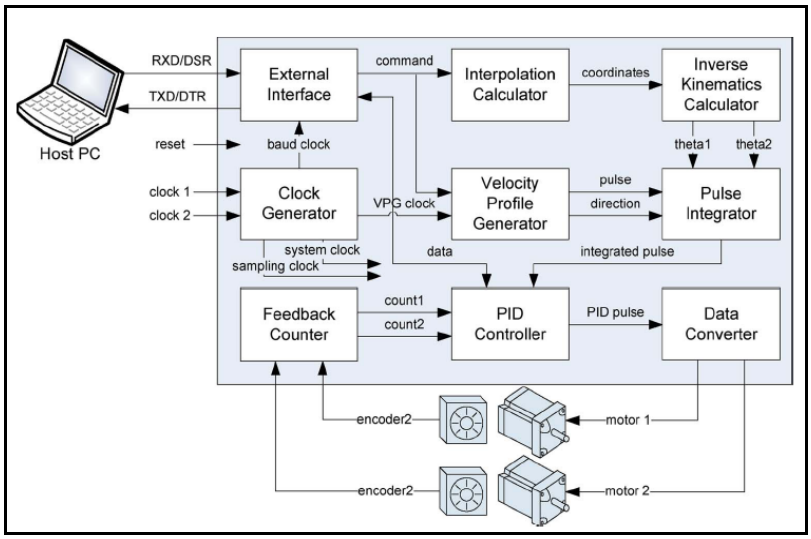
Donanım tanımlaması yapılan tasarım, robot kolunu kontrol için modele yükleme yapılmıştır. Robot kolu taban, omuz, dirsek, bilek ve tutucu olarak 5 eksene sahip robotik manipülatörden oluşmuştur. Bu eksenlerden omuz, dirsek, bilek sırasıyla 180, 300, 60 derece dikey limitlerden oluşmuştur. Tabanının ise yatay eksende döner limiti 270 derece olarak belirlenmiştir. Kontrol sinyalleri motorun hareketinin belirlenmesi için üç fazlı olarak oluşturulmuştur. Her bağlantı için ise geniş açı, dar açı ve durumu koru olarak üç durum ortaya çıkarılmıştır. Bütün bu durumlar göz önünde bulundurularak robot kolu kontrolü yapılmıştır [11].

1.5. Çalışma 5

"An FPGA- Based Multiple- Axis Motion Control Chip" (Cho ve diğ., 2009) adlı makalede FPGA kullanılarak çok eksenli hareket kontrol çipi tasarımı yapılmış ve gerçeklenmiştir. Çok eksenli robot manipülatörü çok eksenli hareket sistemi için kullanılmıştır. Bu çalışmada oluşturulan hareket kontrol çipi birçok durum kontrolü yapmaktadır. VHSIC (very high speed integrated circuit) donanım tanımlama dili kullanılarak bütün bu fonksiyonların tasarımı yapılmış ve FPGA üzerine yüklenmiştir. Şekil 1.7'de verildiği üzere donanımsal olarak da hareket kontrol sistemi, analog servo AC (Alternating Current) sürücü devreleri, robot kolu, bilgisayar ve tasarımın yüklenmesi için gerekli çip setini içermektedir.



Şekil 1.7. Önerilen hareket kontrol sistemi



Şekil 1.8. Hareket kontrol ünitesi

Şekil 1.8'de gösterildiği gibi hareket kontrol ünitesi, hız kontrol, kutupsal hesaplama, ters kinematik hesaplama, PID (Proportional Integral Differantial) kontrol, geri besleme sayıcı, darbe entegrasyonu, veri dönüşümü, saat darbesi üretimi ve ara yüz modülleri gibi bileşenlerini içermektedir.

Hareket kontrol sisteminin çipi içerisinde tasarlanan bileşenlerin görevleri aşağıda sıralanmıştır:

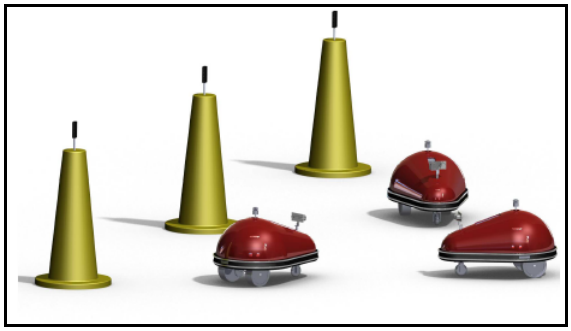
- Geri besleme modülü ile motorun anlık pozisyonunu hesaplar,
- Darbe üreteci modülü tümleşik darbenin zaman örneklemesini alır,
- PID kontrol ile motor pozisyonunu kontrol eder,
- Veri dönüştürücü servo sürücülerin birkaç tipi için sürücü sinyallerinin çıkışını dönüştürür,
- Saat darbe üreteci, sistem için gerekli saat darbelerini üretir, ayrıca ara yüz bileşeni ise ana bilgisayar ile iletişimi sağlayan seri kablodan oluşur [14].

1.6. Çalışma 6

"Low Cost Robots Used for Practical Assignments in Programming Modules" (Samuelsen ve Graven, 2009) adlı makalede küçük ve düşük maliyetli robotların değişik programlama modülleri ile herkesin yapabileceği pratik tasarımlar önermiştir.

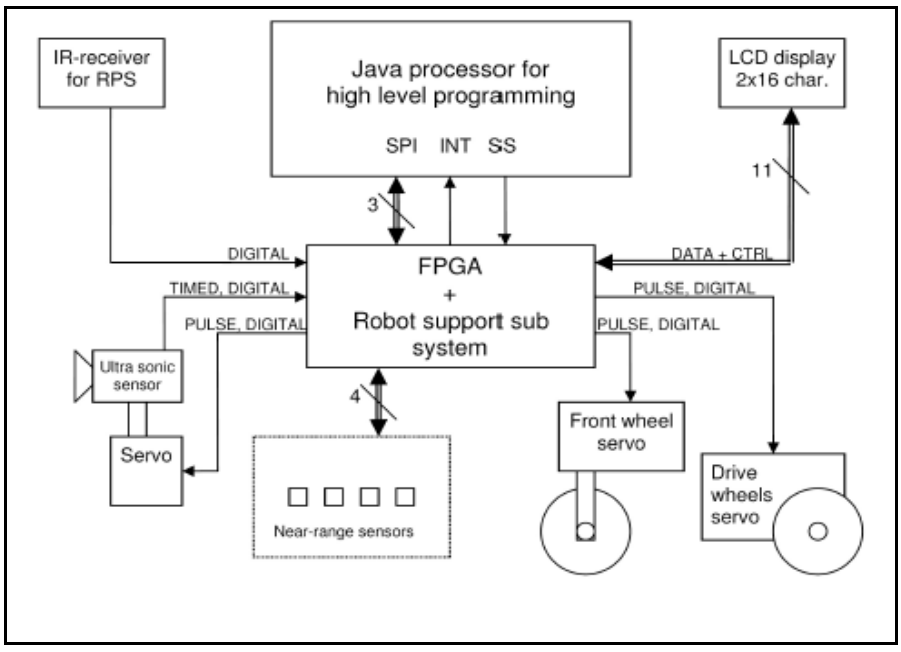
Tasarımda Java sanal makineyi çalıştırmak için ARM işlemci kullanılmış; bununla beraber donanım tanımlamada FPGA geliştirme kiti kullanılmıştır.

Bu proje iki ana hedef üzerine ilerlemiştir. Bunlar, düşük maliyet ve tekrar yapılanmadır. Tasarımın alt modüllerinin kolay değiştirilmesi ve farklı sensörlere uyum sağlaması, esnek olduğunu göstermiştir.



Şekil 1.9. Robotun cisimlerden kaçarak pozisyon alması

Robot devamlı dönen servoya bağlı iki adet tekerlekle hareket etmekte ve 180° dönebilen servo motora bağlı tekerlekle de yönlenmektedir. Robotun çevresindeki cisimlerden kaçmasını sağlamak için de ultrasonik sensör, yakınlık sensörleri ile kızıl ötesi sensör kullanılmıştır. Robotun yön bulması ve yön bulurken de cisimlerden kaçması Şekil 1.9'da gösterilmiştir.



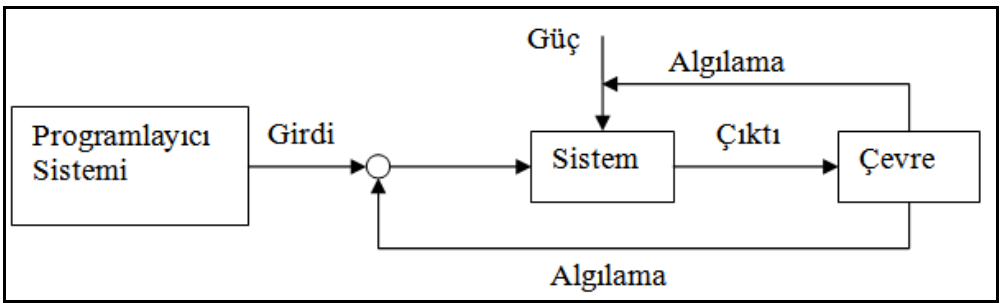
Şekil 1.10. Robot sisteminin mimari yapısı

Şekil 1.10'da verilen robot sisteminde FPGA, sistemin çekirdeğini oluşturmuştur. Robotun çevreden bilgi toplamasında servo motora monte edilmiş ultrasonik sensör kullanılmakla birlikte, kızıl ötesi sensör ve yakınlık sensörleri de kullanılmıştır. LCD (Liquid Crystal Display) ekran ise çevre hakkında toplanan bilgileri görüntülemekte kullanılmıştır [15].

2. MOBİL ROBOT KOLU GENEL YAPISI

2.1. Robot Hakkında Genel Bilgi

Robot, önceden programlanmış görevleri bir yere bağlı olmadan yerine getirebilen elektro-mekanik bir aygıttır. Robotlar dış dünya ile iletişimini çeşitli sensörler yardımıyla yapar, algıladıklarını yorumlayarak karar verir ve karar sonucuna göre hareket eder. Bunu anlatan robotun çalışma diyagramı Şekil 2.1'de verilmiştir.



Şekil 2.1. Robotun çalışma diyagramı

Robot kelimesi, ilk defa olarak Karel Čapek 'in 1920 yılında yazdığı R.U.R. (Rossum's Universal Robots) adlı tiyatro oyununda "roboti" yani köle işçi olarak kullanılmıştır. Tüm dünyada kullanılmaya başlanmıştır. Karel Čapek 'in R.U.R. adlı oyununda mekanik ve otonom, ama insanca duygulardan yoksun yaratıklar olarak kullanılan robot, daha sonra birçok bilim kurgu romanına konu olmuştur [5].

Robot teknolojisinin oluşumunda birçok bilim dalı katkıda bulunmaktadır. Robot, elektro-mekanik bir sistem olduğu için makine, elektronik, bilgisayar, sistem mühendisliği gibi birçok mühendislik dalının ortak ürünüdür. Robot teknolojisi giderek genişleyen alanında birçok bilim dalını kapsayacak şekilde ilerlemektedir. Bunun sonucunda robotun kullanım yelpazesi de oldukça gelişmektedir, buna bağlı olarak da robotun uygulama alanlarından bazıları aşağıda sıralanmıştır:

- Tekstil Mühendisliği
- Tarım Endüstrisi

- Üretim Mühendisliği
- Savunma Sanayi
- Akademik Konular
- Endüstriyel AR-GE
- Oyuncak Endüstrisi
- Uzay Uygulamaları
- Robotik Mühendisliği
- Otomotiv Mühendisliği
- Endüstriyel Otomasyon
- Beyaz Eşya Endüstrisi
- Tıp Uygulamaları
- Havacılık Endüstrisi

Robotun diğer alanlarda uygulamalarının çoğalması sonucu ortak ilgi alanları çoğaldı ve alanlar arası etkileşim arttı. Robotik ayrı bir endüstri alanı olarak hızla gelişirken diğer alanların da gelişmesine katkıda bulunmaktadır.

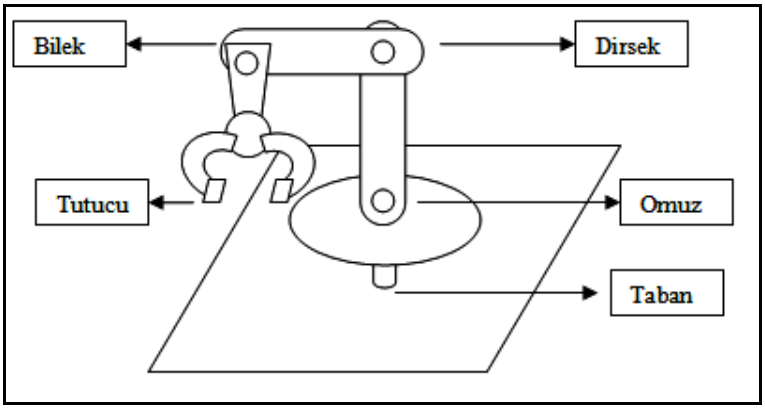
Robot insan yaşamına girerek insana birçok noktada yardımcı olmaktadır. Örneğin; evde yerlerin tozunu alan elektrikli süpürge gibi ev işlerinde kolaylık sağlayacak robotlar yaşamımızı kolaylaştırabilir. Özellikle endüstride birçok otomasyon sistemlerinde robotlar kullanılmakta ve işletmelerin verimliliğini arttırmaktadır.

Robot tasarımlarında canlılara benzeterek mekanik tasarımlar yapılmaktadır. Buna; örümcek, kaplumbağa, tırtıl, insan kolu, insan gibi örnekler verebiliriz. Bundan dolayı; robota, "canlılara benzer işlevleri olan ve davranış biçimleri sergileyen makinalar" diyebiliriz.

2.2. Robot Kolu

Robotlar, özellikle robot kolları endüstride yaygın olarak kullanılmaktadır. Endüstride özellikle otomotiv sektöründe araç monte etme ve boyama gibi prosesleri robot kolları yapmaktadır. Robot teknolojisinde ise robotun yapacağı işler esneklik kazandıkça robotun kullanılabilirliği artmaktadır [6]. Literatürde DOF (Degree of freedom) olarak bilinen serbestlik derecesine göre robot kolları işlev kazanmaktadır. Serbestlik derecesi arttıkça esnekliği de artmaktadır.

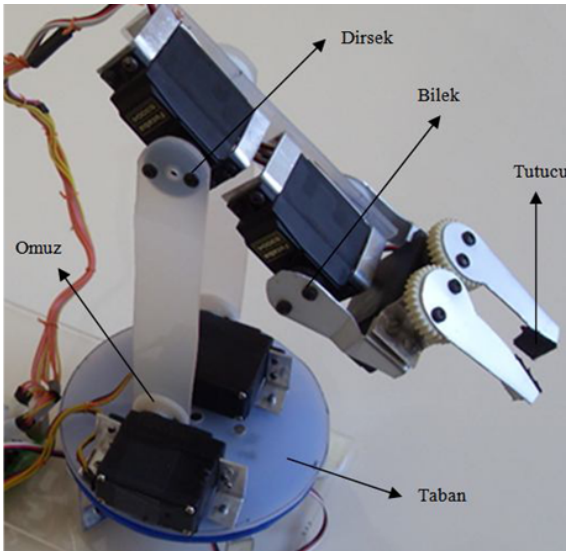
İnsanoğlu etrafındaki harika tasarımlardan esinlenerek kendi teknolojisini tasarlamakta kullanmaktadır. Bu tasarımlarının en etkili bir şekilde çalışması içinde sürekli geliştirme yapmaktadır ve geliştirmelerinin nihai hedefi olarak da etrafındaki harika tasarımlardır. Bunun neticesinde robot kolları da insan koluna benzetilerek tasarlanır ve ona göre eklemleri isimlendirilir.



Şekil 2.2. Robot kolu genel yapısı ve kısımları

Ana taban üzerine yerleştirilen robot kolunda omuz, dirsek, bilek ve el görevini gören tutucu gibi isimlendirmeler kullanılabilir. Bu isimlendirmelerden de anlaşılacağı üzere bu çalışmada kullanılan robot kolu 5 serbestlik derecesine sahiptir. Robot kolunun genel yapısı ve kısımları Şekil 2.2'de verilmiştir.

Tabana yerleştirilen servo motor robot kolunun bütününü yatayda 180°'lik açıyla hareket ettirmektedir. Omuz, dirsek ve bilek kısımlarındaki servo motorlar da bulundukları eksende 180° açı ile hareket ederek robot kolunun cisimleri tutması için konum almasını sağlamaktadır. İstenilen konuma gelen robot kolu tutucu yardımıyla cisimleri tutabilmektedir.



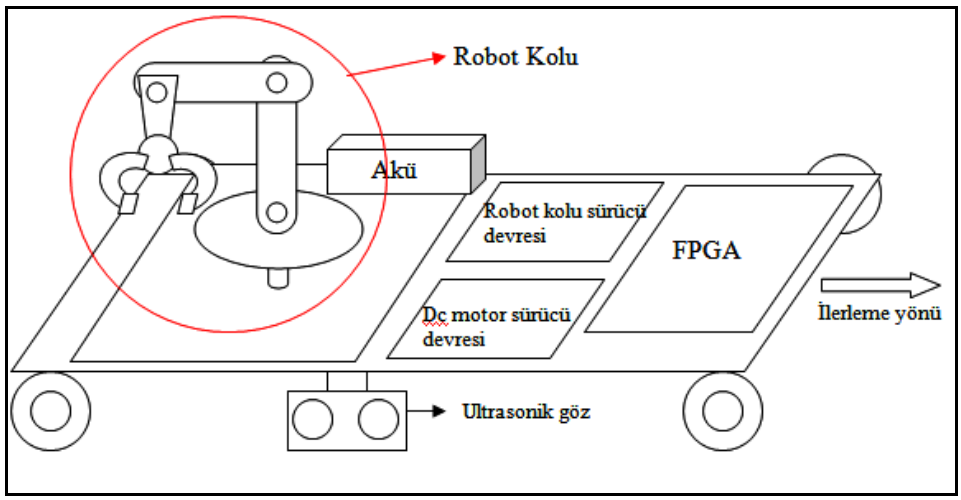
Şekil 2.3. Robot kolunun resmi

Şekil 2.3'te görüldüğü gibi hazır olarak alınan robot kolu genel tasarımında sert plastik kullanılmıştır. Servo motorlar alüminyum parçaları kullanarak plastik parçalara monte edilmiştir. Tutucu ve bilek kısmında ise belirli şekilde kesilmiş alüminyum parçaları kullanılmıştır. Robot kolunun cisimleri sağlıklı bir şekilde tutabilmesi için de tutucunun uç kısımlarına ince keçe takılmıştır. Süngerin, tutulan cismin robot kolu hareket halindeyken tutucudan kaymamasını ve istenilen şekilde tekrar yerleştirilmesinde önemli rolü vardır. Robot kolunun hareketi iki adedi omuz kısmında olmak üzere toplam altı adet servo motor ile sağlanmaktadır.

2.3. Mobil Robot Kolu

Robotlar özgürlük kazandıkça işlevselliği artmakta, tasarımlar karmaşıklaşmakta ve robot kontrol sistemleri de akıllılaşmaktadır. Karmaşık sistemlerin kontrolü için basit algoritmalar yeterli olmadığı için akıllı sistem algoritmaları geliştirilmiştir. Mobil olarak hareket edebilen bir araca robot kolu da eklenince oraya çıkan mobil robot kolunda esnek bir tasarıma ihtiyaç duymaktadır. Bunun anlamı; mobil araç hareket ederken cisimleri algılıyor, yorumluyor ve robot kolu da bu veriler dâhilinde konumunu ayarlıyor. Mobil robot kolu hareket halindeyken karşısına çıkan cisimleri yorumlayarak robot kolu yardımıyla bu cisimlerin konumları üzerinde işlem yapabilmektedir. Cisimleri mobil araca almakta ya da cisimlerin yerlerini değiştirebilmektedir.

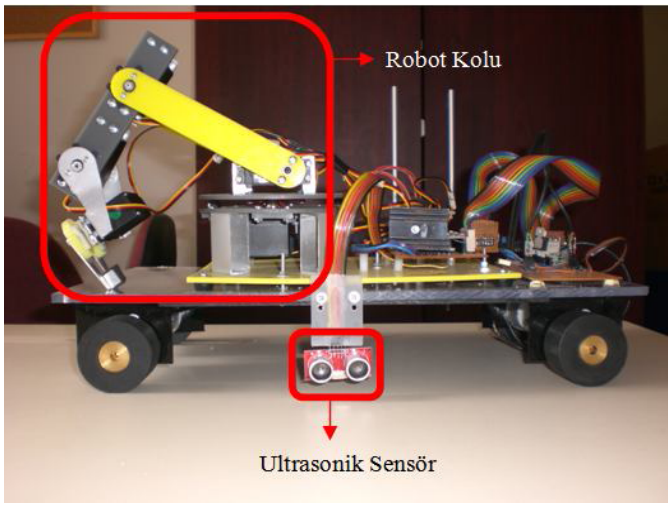
Mobil robot kolunu oluşturan robot kolu, FPGA kiti, motor sürücü devreleri, ultrasonik göz ve güç kaynağı DC motorlara bağlı tekerleklerle hareket edebilen bir platform üzerine yerleştirilmiştir. Mobil robot kolu genel yapısı ve kısımları Şekil 2.4'te verilmiştir.



Şekil 2.4. Mobil robot kolu genel yapısı ve kısımları

Mobil platform üzerine yerleştirilen 12V'luk akü ile robot kolunun 6 adet servo motoru, tekerlekleri döndüren 4 adet DC motor, motor sürücü devreler, ultrasonik sensör ve FPGA kiti beslenmektedir.

Robot tasarımında istenilen, robotun mobil olmasıdır. Bundan dolayı robot sistemi içerisinde güç kaynağı olarak sadece akü bulundurulur. Klasik işlemcilerde harici elemanlar da yer aldığından dolayı güç tüketimi artmaktadır. Bu durum ise mobil robotlar için optimum çözümü sağlayamamaktadır. FPGA teknolojisi ise bu durum için uygun bir çözümdür. Kapı seviyesinde yapılan tasarım, FPGA yongası içerisine gömülü olduğu için ekstra zaman, enerji kaybına ve maliyete yol açmaz.



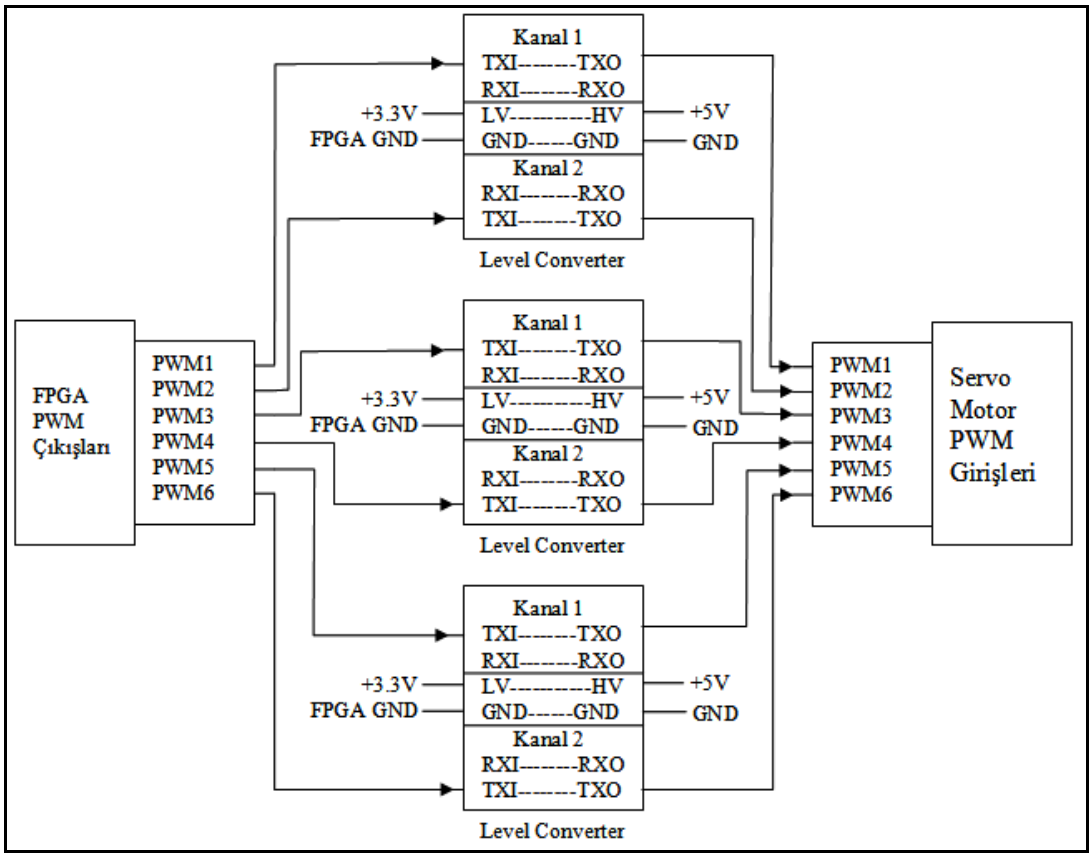
Şekil 2.5. Mobil robot kolunun yandan görünümü

Ultrasonik sensör mobil aracın yan tarafına Şekil 2.5'te görüldüğü gibi monte edilmiştir. Mobil robot kolunun çevreden bilgi toplamasını bu sensör sağlamaktadır. Ultrasonik sensör yerine kamera da konabilir ama bu sistemi daha karmaşık hale getirir ve sistemin enerji tüketimini arttırır. Ultrasonik sensör bize kolay ve hızlı çözüm olurken maliyeti de oldukça düşürür.

Mobil araç hareket halindeyken ultrasonik göz cismi algıladığında cismin boyunu ölçmek için cisim boyunca araç ilerler ve gerekli bilgi elde edilince robot kolunun cismi tutması için geri gelerek cismi ortalar. Bu işlem birbirinden farklı boyuttaki cisimlerin bulunduğu düzlemde de gerçeklenerek cisim ayıklama işlemi yapılır.

2.3.1 Robot kolu sürücü devresi

Robot kolu altı adet servo motordan oluşmaktadır. Yazılımsal olarak oluşturulan altı adet PWM sinyali FPGA çıkış pinlerinden direk olarak motorların kontrol uçlarına verilemeyeceği için bir ara motor sürücü devresine ihtiyaç duyuldu. Logic Level Converter entegresi kullanılarak +3.3V değerindeki FPGA çıkış gerilimini +5V değerine yükselterek servo motorlar sürüldü. Bir adet Logic Level Converter entegresi iki adet kanala sahiptir ve dolayısıyla robot kolunu oluşturan altı adet servo motoru sürmek için üç adet entegre kullanıldı. Robot kolu sürücü devresi genel blok diyagramı Şekil 2.6'de verilmiştir.



Şekil 2.6. Robot kolu sürücü devresi genel blok diyagramı

TXI ucuna verilen +3.3V değerindeki PWM sinyali TXO ucundan +5V olarak alınmaktadır. Devrenin dönüşüm yapması için de LV (low voltage) ucuna +3.3V, HV (high voltage) ucuna da +5V değerinde gerilim uygulanmalıdır. LV ucu FPGA'nın sabit +3.3V gerilim veren pinine bağlanır. LV tarafındaki GND (Ground) ise FPGA'nın GND'sine bağlanır. Devrenin servo motor girişleri tarafında yer alan HV ucu ise servo motorların ortak beslenme hattı üzerinden +5V ile beslenmiştir.

2.3.2 Servo motorlar

Servo motorlar, verilen girişe göre istenen açısal konuma gelen motorlardır. Servo motorların çeşitli uygulamalarda kullanılmasının, güvenilir olmasının yanında diğer nedenleri ise;

• Yüksek tork,

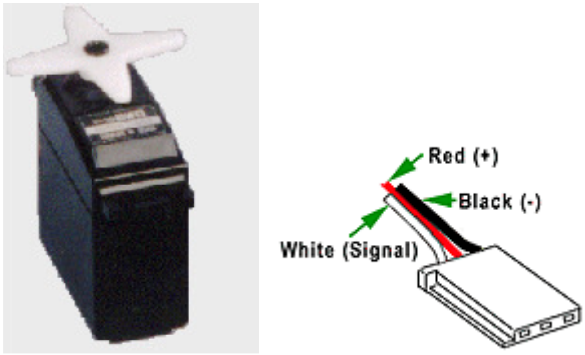
• Doğru konumlama,

• Kolay kurulum,

• Kontrol kolaylığı,

• Ekonomik oluşu,

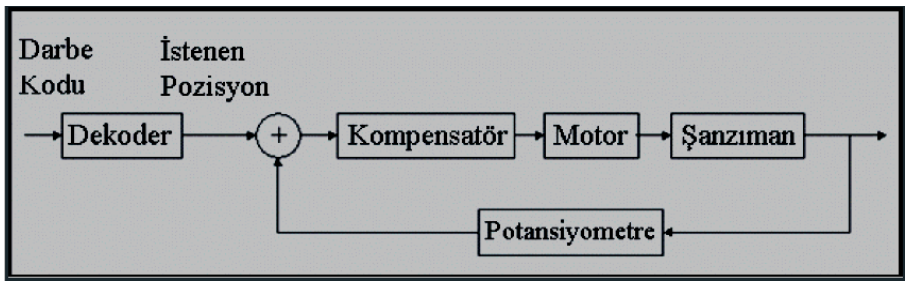
özelliklerine sahip olmasıdır.



Şekil 2.7. Servo Motor ve bağlantı kablosu

Şekil 2.7'de servo motor ve bağlantı kablosu verilmiştir. Servolarda üç adet kablo dışarı çıkar. Bunlardan kırmızı olan +4.8 V/ 6.0 V arası besleme, siyah olan şase yani Ground (toprak), diğer kablo ise (turuncu, yeşil veya beyaz olabilir) data girişidir. Bir servo motor, DC elektrik motoru, planetar dişli sistemi, geri besleme potansiyometresi ve DC motor pozisyon kumanda devresi olmak üzere dört kısımdan oluşmaktadır.

DC motor herhangi bir DC oyuncak motorundan farklı olmayan çift mıknatıslı bir statora ve fırçalı bobin rotora sahiptir. Motor mili 1:200 ile 1:300 arası dönme oranına sahip bir dişli sistemine bağlıdır, bu sayede oldukça yüksek bir tork değerine ulaşır.



Şekil 2.8. Servo motorun çalışma seması

Şekil 2.8'de servo motorun çalışma şeması verilmiştir, buna göre dişli sisteminin çıkısında yer alan bir potansiyometreden alınan mil konumu elektronik kumanda

devresine iletir. Elektronik devrenin görevi mil konumunu, gelen veri konumuna gelinceye kadar motoru iletimde tutup tam yerinde durdurmaktadır.

Elektronik devreden bu konumu algılamak için PWM tekniğinden yararlanılmaktadır. Kumanda devresi kumanda çubuğunun konumuyla doğru orantılı olarak 1 ile 2 milisaniye arasında dalga periyodu değişen bir sinyali her 20 milisaniyede bir servoya gönderir. 1 milisaniye tam sol, 2 milisaniye de tam sağ pozisyonu ifade eder. Servo içindeki elektronik devre ilk önce gelen darbelerin darbe genişliğini ölçer, daha sonra potansiyometre konumuna bakar ve kendi darbe osilatörünün darbe genişliği gelen darbelerle eşitlenene kadar motoru hareket ettirir. Servo motorun verilen PWM sinyaline göre açı değerleri Tablo 2.1'de verilmiştir.

Tablo 2.1. Servo motorun darbe sürelerine göre açıları

Süre (ms)	1.0	1.1	1.2	1.25	1.3	1.4	1.5	1.6	1.7	1.75	1.8	1.9	2.0
Açı (derece)	-90	-72	-54	-45	-36	-18	0	18	36	45	54	72	90

Servo motor uygulamalarında, uygulamaya göre servo motor tercihi yapılmalıdır. Günümüzde çok çeşitli uygulamalar için birçok servo motor vardır. Servo motorların boyutları, ağırlığı, torku gibi özellikler göz önünde bulundurularak tercih yapılır. Robot kolu tasarımında Tablo 2.2'de özellikleri verilen servo motor kullanılmıştır.

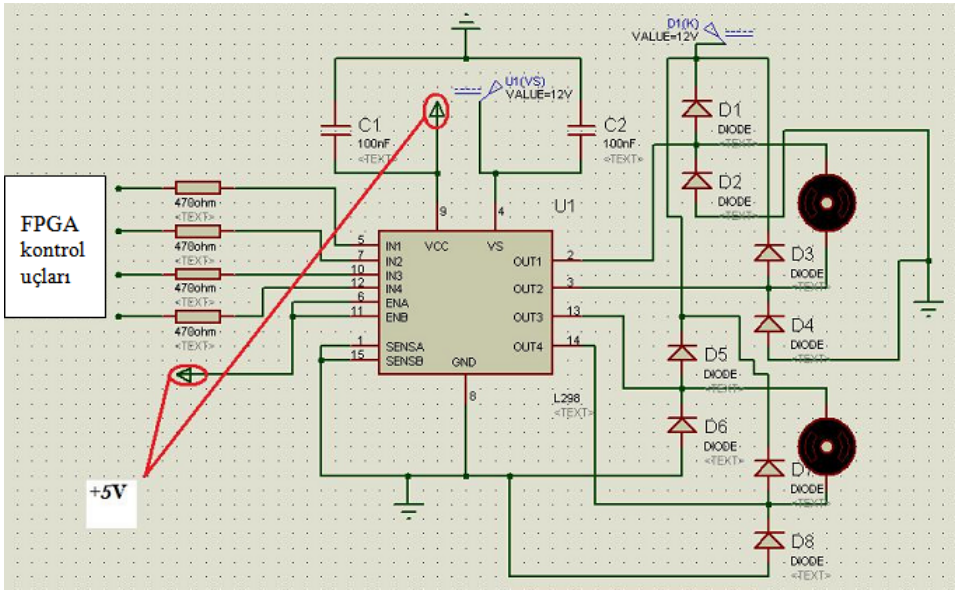
Tablo 2.2. Kullanılan servo motorun teknik özellikleri

Çalışma Voltajı		4.8 V - 6.0 V
Dönüş hızı	4.8 V	0.19 s/60°
	6.0 V	0.15 s/60°
Tork	4.8 V	3.0 kg/cm
	6.0 V	3.5 kg/cm
Motor tipi		3 kutuplu
Mil tipi		Plastik
Ağırlık		43.0 g
Boyutlar		40x20x36.5 mm

2.3.3 DC motor sürücü devresi

Mobil aracın hareket etmesi için mobil robot kolu platformu dört tane DC motor ile kontrol edilen tekerlekler üzerine yerleştirilmiştir. Mobil aracın hareketini kontrol etmek için yazılımsal tasarım DC motor sürücü devresi yardımıyla DC motorları kontrol etmektedir. DC motorları sürmek için L298 entegresi kullanılmıştır. L298 entegresi içerisinde 2 adet H köprüsü bulunur ve 2 tane motoru çift yönlü bağımsız kontrol edebilir.

Motor sürücü devresinin çalışmasını analiz etmek için Proteus ISIS paket programı kullanılmıştır.



Şekil 2.9. L298 entegreli motor sürücü devresi şeması

Şekil 2.9'da verilen devre şemasında da görüldüğü gibi 2 adet DC motor ve her motor için 2 adet kontrol ucu olmak üzere 4 adet FPGA kontrol ucu ile kontrol edilmektedir [7]. DC motorların girişlerine verilen gerilime göre motorlar ileri ya da geri dönmektedir. IN1'e +5 V, IN2'ye 0 V verilirse motor ileri dönüyorsa IN1'e 0 V, IN2'ye +5 V verilirse geri döner. Her iki giriş aynı değerde (her iki uç +5 V ve ya 0 V) olursa motor dönmez.

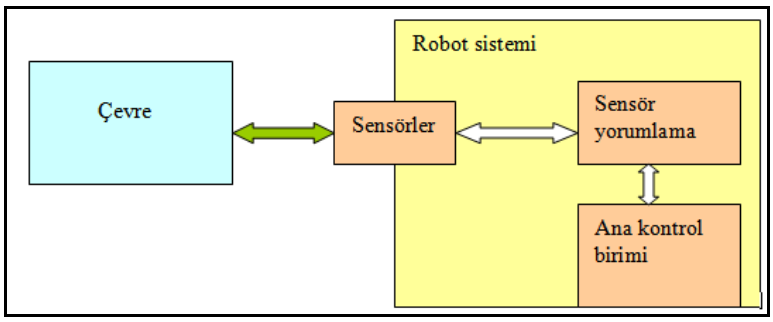
Tablo 2.3. DC motora verilen giriş değerlerine göre durum tablosu

Girişler		Durum
VEN=+5 V	IN1=+5 V;IN2=0 V	İleri
	IN1=0 V;IN2=+5 V	Geri
	IN1=IN2	Dönüş yok
VEN=0 V	Herhangi bir değer	Dönüş yok

Tablo 2.3'den de anlaşılacağı gibi EN yani yetkilendirme ucu aktif edilince motor ileri ya da geri dönmektedir. EN uçları Vss ucuyla birlikte +5 V gerilime bağlanır. Vss ucu entegrenin çalışmasını sağlayan gerilimin verildiği uçtur.

2.3.4 Robot Sensörleri

Robotların çevreyi yorumlayabilmesi için çeşitli algılayıcılara ihtiyacı vardır. Algılayıcılar dış ortam ile robot ortamı arasında köprü görevini görür. Algılayıcı sözcüğü yerine daha yaygın olarak sensör kelimesi kullanılmaktadır. Sensörler çevreden aldıkları çeşitli analog verileri elektrik enerjisine dönüştürür.



Şekil 2.10. Robot ile çevre ilişkisi

Şekil 2.10'da verildiği gibi sensörde elde edilen elektriksel veriler sensör yorumlama biriminde yorumlanarak ana kontrol birimine gönderilir. Ana kontrol birimi de robotun bütün sistemini göz önüne alarak gerekli kontrol sinyallerinin üretilmesini sağlar. Robot, ana kontrol biriminin ürettiği sinyaller doğrultusunda konumunu belirler. Sensörler, robotla dış çevre arasındaki iletişimi sürekli bu şekilde tutar.

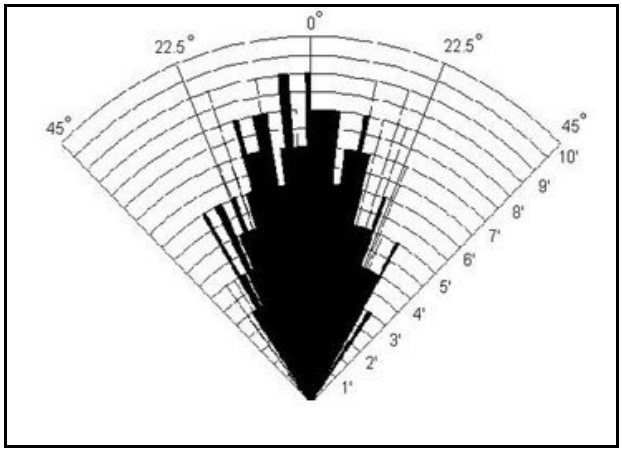
Robotik uygulamalarda genel olarak iki çeşit sensör kullanılmaktadır. Bunlar aktif ve pasif sensörlerdir. Aktif sensör, çevreyi algılayabilmek için enerji yayması gereken sensördür. Radar, lazer, sonar sensörleri aktif sensörlere örnek verilebilir. Bu sensörlerin yaydığı sinyal, çevredeki cisimler tarafından sensöre geri yansıtılır ve bu sayede çevre hakkında istenilen bilgi elde edilebilir. Pasif sensör ise çevredeki diğer kaynakların yansıttığı enerjiyi algılar. Kamera pasif sensöre bir örnektir [8].

Aktif sensör mevcut sistemin enerjisini kullanarak çevreye enerji yaydığı için güç tüketimini olumsuz yönde etkileyebilir. Kamera gibi pasif sensörleri de bir sisteme dâhil ettiğimizde, daha fazla işlem yapan ve maliyeti artıran bir durum ortaya çıkar. Robot kontrol sistemlerinde bu belirtilen durum göz önünde bulundurularak sensör seçimi yapılmalıdır.

2.3.5 Ultrasonik sensörler

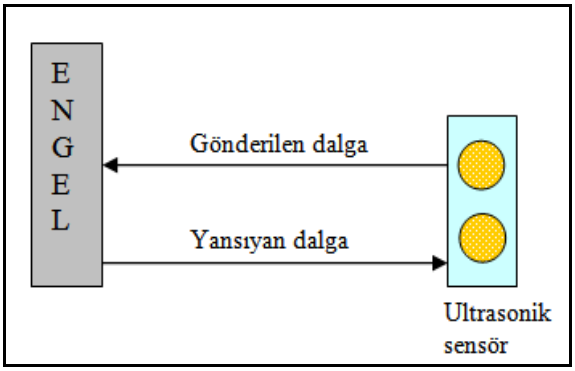
İnsanın duyma sınırı olan 30 Hz–14000 Hz aralığından daha yüksek frekanslı sesler ultrasonik ses olarak adlandırılır. Bu tanımdan da anlaşılacağı gibi ultrasonik sinyaller çok yüksek frekansta ses sinyalleridir. Ultrasonik ses dalgaların oldukça düzgün ve doğrusal bir şekilde ilerlemeleri, taşıdığı enerjinin yüksek oluşu ve sert yüzeyli nesnelerden kolaylıkla yansımaları kullanımını yaygınlaştırmaktadır.

Şekil 2.11'den anlaşıldığı gibi ultrasonik ses dalgasının etki alanı 30°'dir [9]. Etki alanındaki açı daraldıkça daha düzgün ve uzak menzilli ultrasonik ses dalgası elde edilir. Aynı zamanda daha küçük adımlı ölçüm hassasiyeti de sağlanmış olur.



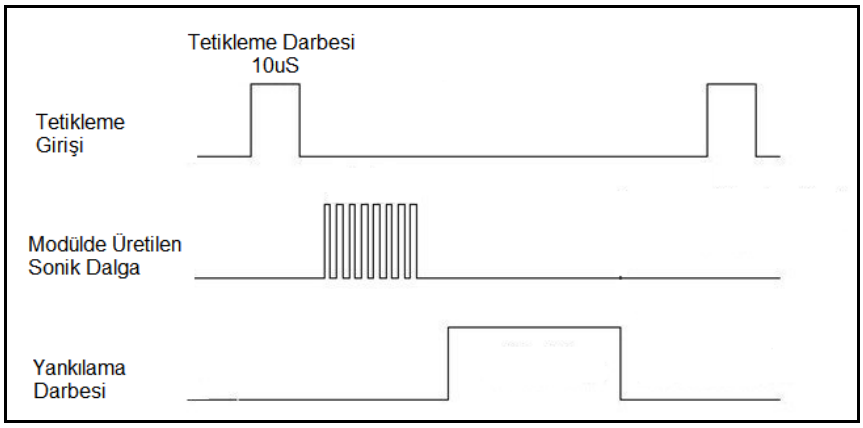
Şekil 2.11. Ultrasonik sensörün algılama paterni

Ultrasonik sensörler 40 kHz frekansta titreşimler yapan piezo kristalden meydana gelir. Yayıcı ve algılayıcı olarak iki kısımdan oluşur. Yayıcı (Transmitter) kısım, titreşen kristalin üzerindeki elektrik enerjisini ses dalgasına dönüştürür. Algılayıcı (Receiver) kısmı ise cisimlerden geri yansıyan ses dalgalarını benzer işlemlerin tersini yaparak elektrik enerjisine dönüştürür. Ultrasonik sensör ile mesafe ölçümü şekil 2.12'de verilmiştir.



Şekil 2.12. Ultrasonik mesafe ölçümü

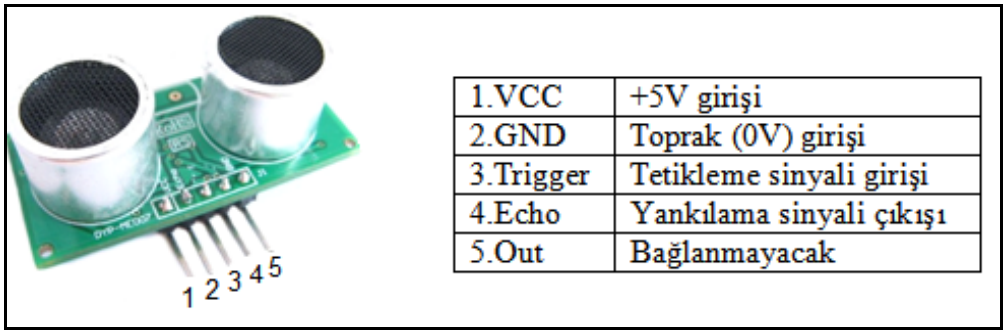
Mobil robotun dış çevre ile ilgili bilgi toplamasında hazır ultrasonik tümleşik devre kullanılmıştır. Şekil 2.13'te verildiği gibi bu devrenin tetikleme (Trigger) ucuna 10 µs'lik kısa darbe verilerek sensörün 40 kHz'lik ses dalgasını göndermesi sağlanır. 10 µs'lik ses dalgasının gönderilmesiyle yankılama (Echo) ucu lojik 1 olur ve ses dalgasının geri gelmesini bekler. Dalga geri geldiğinde yankılama ucu lojik 0 olur [9]. Lojik 1 olduğu süre ile sesin havada yayılması (344 m/s) çarpılarak ses dalgasının aldığı yol bulunur. Bu mesafe ikiye bölünerek cismin ultrasonik sensöre uzaklığı bulunmuş olur.



Şekil 2.13. Ultrasonik sensör çalışma zaman diyagramı

Ultrasonik sensörün çalışmasını etkileyen faktörler, hedef cismin sıcaklık, nem, yüzey açısı ve cismin gönderilen ses dalgasını soğurması sayılabilir ayrıca ultrasonik sensör cisimlerin renginden etkilenmez.

Mobil robot kolunda DYP-ME007 ultrasonik entegre devresi kullanılmıştır. DYP-ME007 ultrasonik sensör kitinin görünümü ve pin bağlantıları Şekil 2.14'te verilmiştir. Ultrasonik sensörün pin bağlantılarını direk olarak FPGA'nın giriş-çıkış pinlerine bağlayarak ultrasonik mesafe ölçümünü gerçekleştirebiliriz. Bu ultrasonik sensörün teknik özellikleri Tablo 2.4'de verilmiştir.



Şekil 2.14. DYP-ME007 ultrasonik entegre devresi ve pin bağlantıları

Tablo 2.4. Ultrasonik sensörün teknik bilgileri

Besleme gerilimi	+5 V
Akım Tüketimi	15 mA
Ultrasonik Frekans	40 kHz
Etki alanı	3 cm-300 cm
Çözünürlük	1 cm
Tetikleme darbe genişliği	10 us
Boyutları	43x20x15 mm

Ultrasonik sensörün bazı kullanım alanları:

- Seviye ölçümü,
- Yaklaşım Uygulaması,
- Boyutlandırma,

- Rulo Çapı Ölçümü,
- Seçme / Sınıflandırma,
- Bağlantı Kopma Belirlenmesi / Döngü Kontrolü,
- Araç alarm sistemleri,
- Işıklandırma kontrolü,
- Araç park sistemleri,
- Otomatik kapı kontrolü,

olarak sıralanabilir [16].

3. PWM MODÜLÜNÜN VHDL DONANIM TANIMLAMA DİLİNDE YAZILMASI ve FPGA ÜZERİNDE GERÇEKLENMESİ

Robotik uygulamalarda robotların hareketi motorlar yardımıyla yapılmaktadır. Servo motorun ve DC motorun hız ve konum kontrolü için motorun kontrol ucuna PWM (pulse width modulation) sinyali verilmesi gerekir. PWM sinyalinin doluluk boşluk oranına göre motor kontrolü yapılmaktadır. Özellikle servo motorların dönüş açılarının belirlenmesi için, doluluk boşluk oranı ayarlanabilir sinyal üretilerek motora verilmesi gerekir. Servo motorlar gelen kontrol sinyaline göre robotların hareketli parçalarının hareketini sağlar ve konumunu belirler. Örneğin, uygulamadaki robot bir robot kolu ise kolun hareketli eksenlerinin anlamlı ve bütünlük ifade eden hareket sergilemesi için robotu oluşturan servo motorlara uygun PWM sinyalleri belirli düzende gönderilir.

Mobil robot kolu tasarımının temelinde PWM üretme modülü vardır. PWM modülünün XILINX paket programında VHDL ile oluşturulup FPGA'ya yükleme aşamalarını adım adım anlatılması aşağıdaki gibidir:

Xilinx-ISE paket programı çalıştırılır. Yeni bir proje oluşturmak için ise aşağıdaki adımlar izlenir:

1. File > New Project... Yeni proje penceresi açılır.

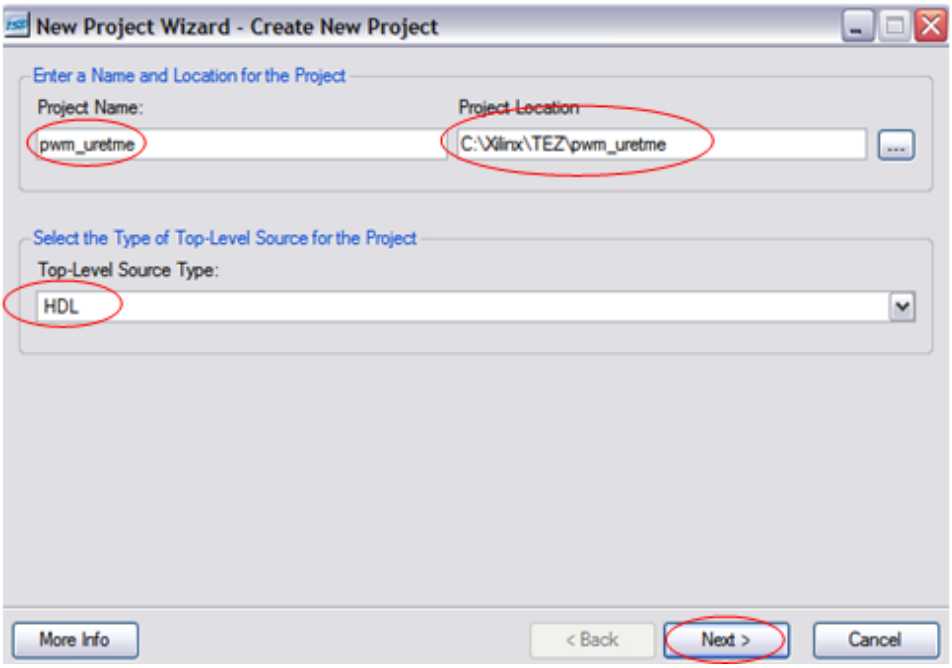
2. "Project name" yazan kısma proje ismi yazılır. PWM sinyali oluşturulacağı için proje ismine de "pwm_uretme" yazılabilir.

3. Proje dosyasının yeri seçilir. Genelde bu kısım proje klasörünün içinde olmalıdır, yoksa proje çalışmasında ve analizinde sorun oluşturabilir.

4. "Top-Level Source Type" listesinden HDL (Hardware Description Language) seçilir, çünkü tasarımın kodları HDL dilinde yazılacak.

5. Aygıt özelliklerini seçmek için Next butonuna basılır.

Şekil 3.1'de kırmızı çember içine alınan kısımlar üzerinde işlem yapılır.



Şekil 3.1. Yeni proje oluşturma ve proje isminin yazılması

6. Aşağıda verilen FPGA aygıt özellikleri seçilir:

♦ Product Category: All

♦ Family: Spartan3

♦ Device: XC3S200

♦ Package: FT256

♦ Speed Grade: -4

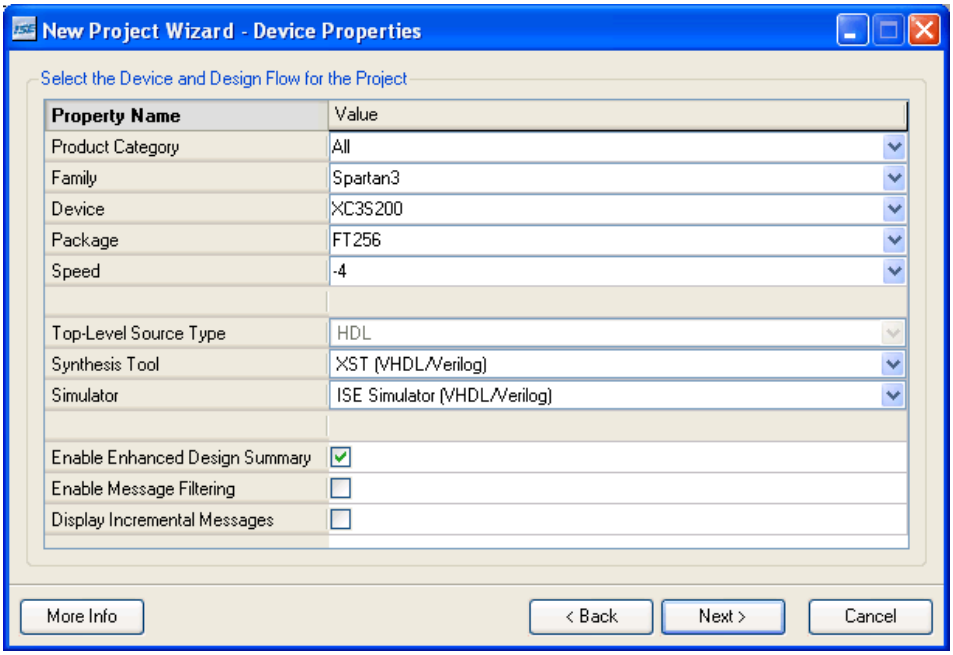
♦ Top-Level Module Type: HDL

♦ Synthesis Tool: XST (VHDL/Verilog)

♦ Simulator: ISE Simulator (VHDL/Verilog) veya Modelsim-SE VHDL seçilir.

♦ "Enable Enhanced Design Summary" seçilir.

Bu özellikler kullanılan FPGA kitine göre değişiklik gösterebilir. Gerekli tercihler yapıldıktan sonraki pencerenin hali Şekil 3.2'deki gibi olmalıdır:

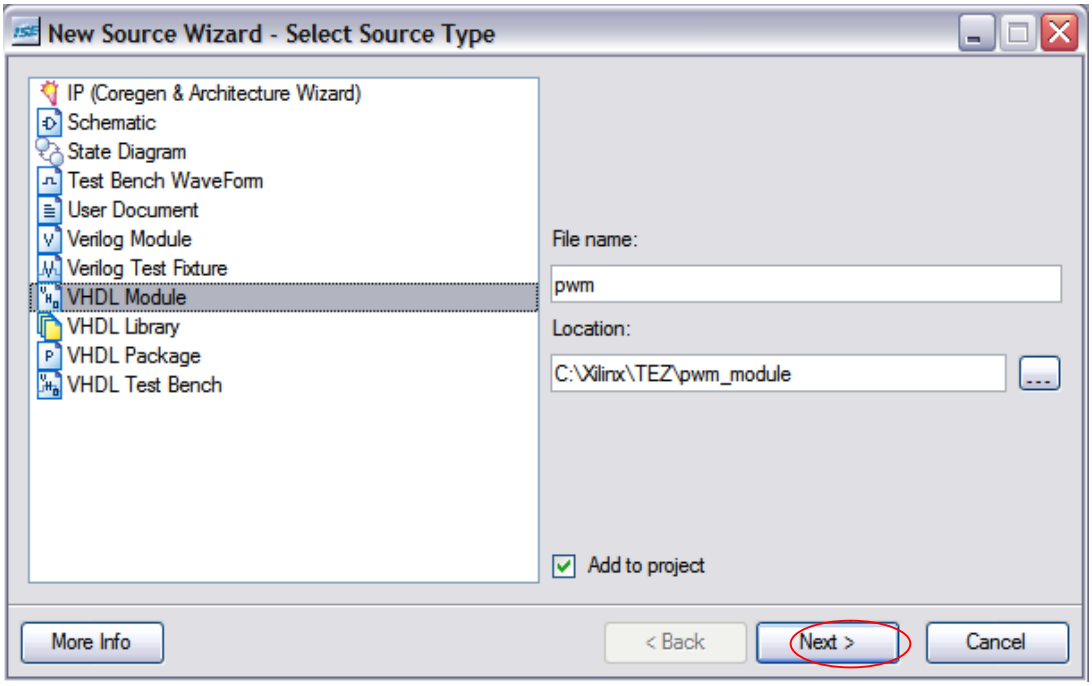


Şekil 3.2. Projenin aygıt özellikleri

7. "Next" butonuna basılarak XILINX programının ana proje ekranı gelir ve ekranın sol kısmındaki prosesler bölümünden "Create New Source" seçilir ve projenin içerisine yerleştirilecek olan dosyalar burada oluşturulur.

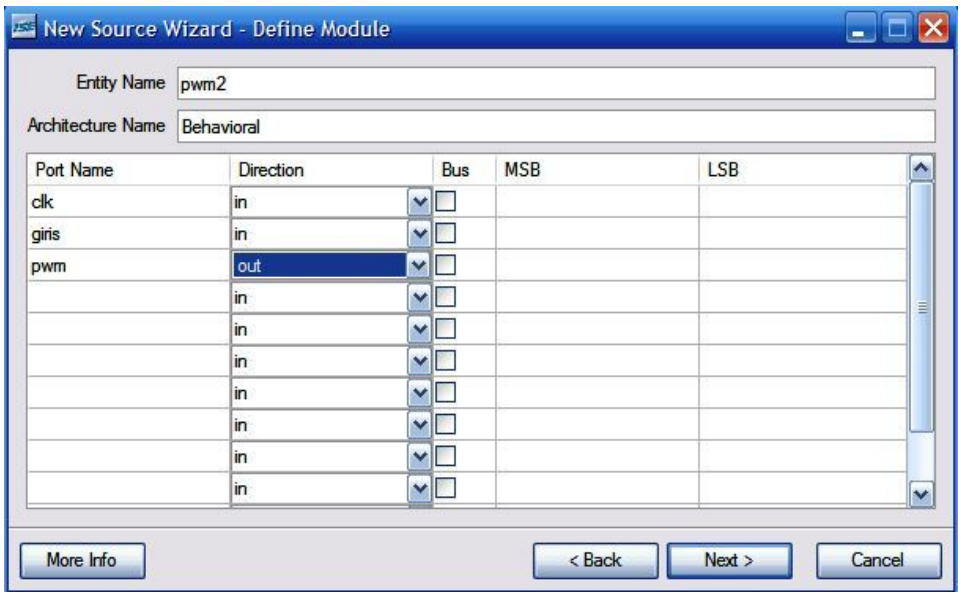
3.1. Yeni Bir VHDL Modülünün Oluşturulması

Yukarıdaki adımlardan sonra gelen kaynak tipi seçme penceresinden proje VHDL koduyla tasarlanacağı için "VHDL Model" seçilir ve proje dosyanın ismi "pwm" olarak yazılır. "Location" kısmına ise uzantılı dosyanın yolu tanımlanır. Kaynak tipini seçme ve dosya oluşturma penceresi Şekil 3.3'te verilmiştir.



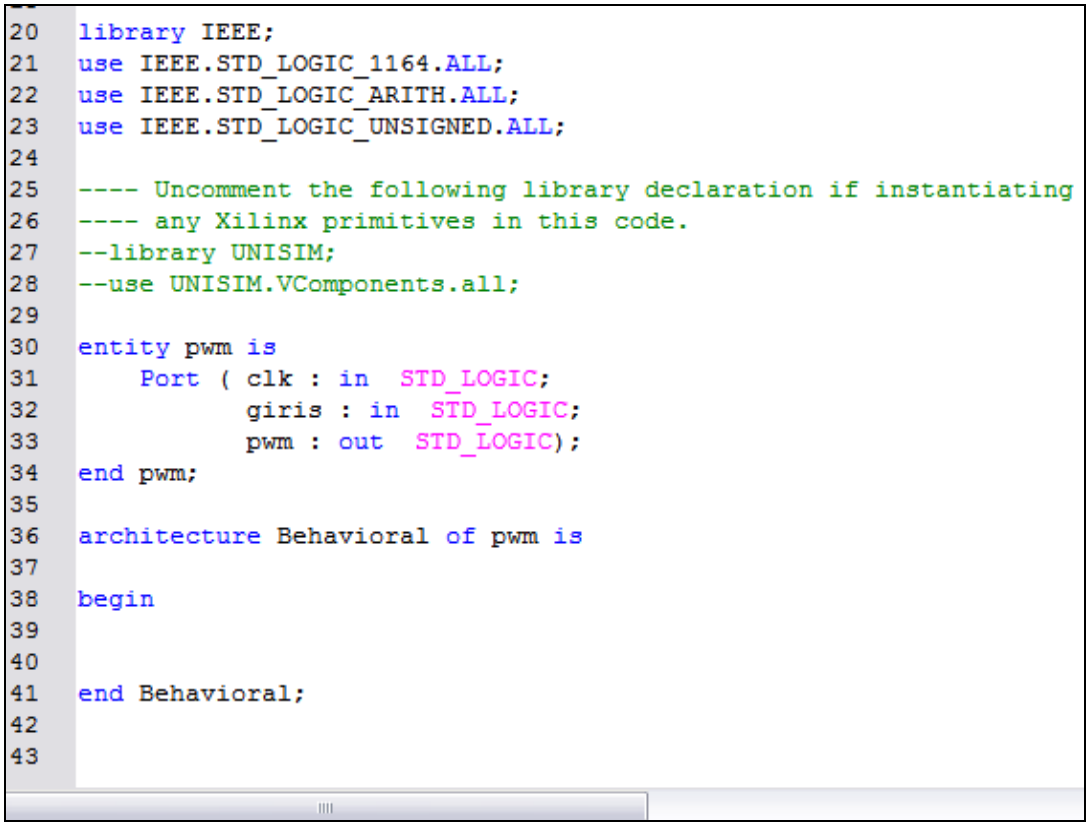
Şekil 3.3. VHDL Modülünün oluşturulması

Yukarıda görülen resimde olduğu gibi "Next" butonuna tıklanarak "Define Module" penceresine geçilerek modül giriş çıkışları tanımlanır.



Şekil 3.4. PWM modülünün tanımlanması

Modülde "clk" yerel saat darbesi ve "giris" in yani giriş olarak, "pwm" ise out yani çıkış olarak tanımlanır. "Next" butonuna basılır. Gelen "Summary" penceresinden ise "Finish" butonuna basılarak modül oluşturulur.



```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.STD_LOGIC_ARITH.ALL;
23  use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25  ---- Uncomment the following library declaration if instantiating
26  ---- any Xilinx primitives in this code.
27  --library UNISIM;
28  --use UNISIM.VComponents.all;
29
30  entity pwm is
31      Port ( clk : in  STD_LOGIC;
32             giris : in  STD_LOGIC;
33             pwm : out  STD_LOGIC);
34  end pwm;
35
36  architecture Behavioral of pwm is
37
38  begin
39
40
41  end Behavioral;
42
43
```

Şekil 3.5. Yeni proje modülü ekran görüntüsü

Şekil 3.5'te de görüldüğü gibi yeni projenin modülü oluşturulur. VHDL yüksek seviyeli donanım tanımlama dili olduğu için; kütüphane tanımlamaları, varlık tanımlamaları, mimari tasarımın yapılacağı bölüm otomatik olarak modülün içerisine yerleştirilir.

3.2. VHDL Dil Yardım Paketinin Kullanımı

Örneğin yeni oluşturulan dosyada sayıcı kodu kullanılmak isteniyorsa ve nasıl olduğu bilinmiyorsa VHDL dil yardım paketi kullanılabilir. Bunun için aşağıdaki adımlar izlenir:

1. "Edit → Language Templates" açılır

2. "+" sembolü kullanılarak kod örnekleri açılır:

VHDL → Synthesis Constructs → Coding Examples → Counters → Binary →

Up/Down Counters → Simple Counter

3. Verilen kod örneği incelenerek istenilen sayıcı programı yazılır.

3.3. Kodun Yazılması ve Simülasyon Yapılması

1. Kütüphane tanımlaması yapılır. Burada amaç veri türlerinin içinde yer aldığı kütüphanenin belirtilmesidir. Modül ilk oluşturulduğunda otomatik olarak standart ieee kütüphanesi oluşturulur. Eğer farklı veri türleri kullanıyorsak onların içinde yer aldığı kütüphanelerin tanımlanması gerekir.

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;  (standart ieee kütüphanesi)

use IEEE.STD_LOGIC_ARITH.ALL;  (lojik aritmetik işlem kütüphanesi)

use IEEE.STD_LOGIC_UNSIGNED.ALL; (işaretsiz lojik aritmetik işlem kütüphanesi)
```

2. “ENTITY” yani “varlık” kısmının tanımlanması ise devreye ait olan tüm giriş, çıkış uçlarının özelliklerinin listelendiği kısımdır.

```
entity pwm1 is

Port ( clk : in std_logic;

giris : in  STD_LOGIC_VECTOR (7 downto 0);

pwm : out std_logic);

end pwm1;
```

Kullanılacak olan PWM devresinin giriş çıkış uçları tanımlaması yukarıdaki kod parçasında gözükmektedir. “clk” devrenin yerel saat darbesidir, giriş olarak standart lojik veri türünde tanımlanmıştır. “giris” pwm’de doluluk boşluk oranını ayarlayabilmek için kit üzerinden anahtarlar yardımıyla alınan giriş işaretidir. Bu işaret 8 bitlik standart lojik vektör türündedir ve öyle tanımlanmıştır. “pwm” ise devre çıkışında oluşacak olan PWM işaretidir ve standart lojik türündedir.

3. “Architecture” yani “mimari” tanımlama kısmında devrede kullanılacak olan diğer sinyaller kullanılır bu sinyaller devre içinde yer alan global değişkenlerdir.

```
architecture Behavioral of pwm1 is

signal x : integer  :=0;

signal sayac : integer := 0;

signal clk_main : std_logic :='1' ;

begin
```

“x” sinyaline devrenin girişinden gelen 8 bitlik sayının tamsayıya çevrilmesiyle oluşan sayı atanır. Bu sayı döngü içinde karşılaştırma elemanı olarak kullanılır. “sayac” adından da anlaşılacağı üzere sayaç olarak kullanılmaktadır. “clk_main” ise FPGA yerel osilatörünün belirli bir sayıya bölünmesiyle oluşan yeni saat darbesidir, bu darbe servo motorlar için gerekli olan 20 ms.’yi elde etmede kullanılmaktadır.

4. Referans saat darbesi kullanılarak servo motorlar için gerekli olan periyodik işaret elde edilir.

```
process(clk)

variable y:integer:=0;

begin

if clk'event and clk = '1' then

y := y+1;

if y = 2000 then

clk_main <= not(clk_main);

y:= 0;

end if;
```

```
end if;

end process;
```

Bu kod parçasında process fonksiyonu, parantez içinde yer alan “clk” yerel saat darbesini referans alarak yeni bir saat darbesi üretir. “variable” yani yerel değişken olarak tanımlanan “y” kodda sayaç görevini yapmaktadır.

5. PWM işaretinin elde edilmesi için ise yeni oluşturulan “clk_main” sinyali referans olarak kullanılır.

```
(1)process (clk_main)

(2)begin

(3)   if clk_main'event and clk_main = '1' then

(4)           sayac <= sayac + 1;

(5)           x <= conv_integer(giris);

(6)           if sayac < x then

(7)                   pwm <= '1';

(8)           else

(9)                   pwm <= '0';

(10)          end if;

(11)          if 254 < sayac then

(12)                  sayac <= 0;

(13)          end if;

(14)  end if;

(15)end process;
```

(16)end Behavioral;

Girişte alınan 8 bitlik sayı tamsayıya çevrilerek elde edilen sayı sayaçta karşılaştırma elemanı olarak kullanılır. “if” deyimi koşul ifadesidir. Sayaç çevrilen sayıdan küçük olduğu sürece PWM “1” çıkışını verecektir, sayaç çevrilen sayıdan büyük olduğu sürece de “0” çıkışını verecektir. Böylece istenilen oranda PWM dalgası elde edilmiş olunur. Bu PWM dalgasının periyodunu ise sayacı 255’e kadar saydırarak elde ederiz, sonra sayacı sıfırlarız. Sayaç sıfırlanmaz ise sonsuza kadar sayma eğilimine girer.

Yazılan kodun açıklaması:

(1) Seri kod yapısı olan if deyimi için gereklidir. VHDL yapısı gereği paralel çalışır bunu seriye dönüştürmede kullanılır.

(2) Başlangıç ifadesi olup “process”in başladığını belirtmede kullanılır.

(3) Saat darbesi 1 konumundayken ilerleme yapılır.

(4) Sayacı birer adım arttırmada kullanılır.

(5) Girişteki 8 bitlik sayıyı tamsayıya çevirir.

(6) Sayaç girişten küçükse şartını gerektirir.

(7) Şart sağlanıyorsa pwm’i “1” yapar.

(8) Değilse şartını gerçekleştirir.

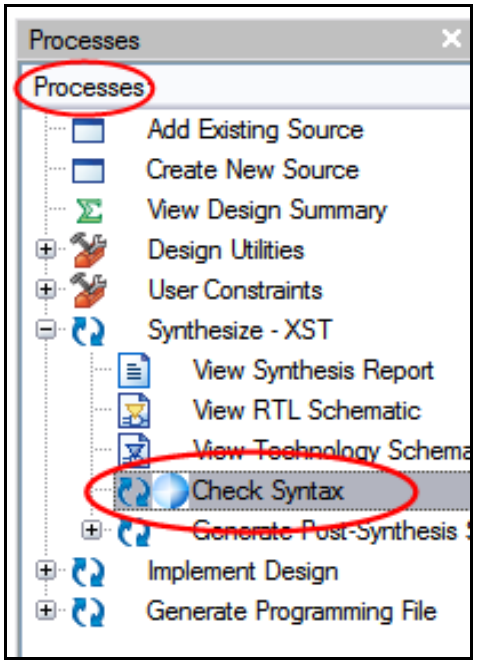
(9) Değilse pwm “0” yapılır.

(10) if deyimini sonlandırır.

(11), (12), (13), (14) PWM işaretinin periyodik olması için sayacı sıfırlamada kullanılır. Şartı ise sayaç 256 olması beklenir. 256 olunca sayaç sıfırlanır.

(15) Seri işlemi gerçekleştiren “process” deyiminin bittiğini belirtir.

6. Yazlan kodun hatalı olup olmadığını kontrol etmek için “Processes” penceresinde yer alan “Synthesize-XST” nin alt birirmi olan “Check Syntax” çift tıklanır. “Check Syntax” yazılan kodda yazım hatasının olup olmadığını kontrol etmek demektir.



Şekil 3.6. Yazım hatası kontrolü “Check Syntax”

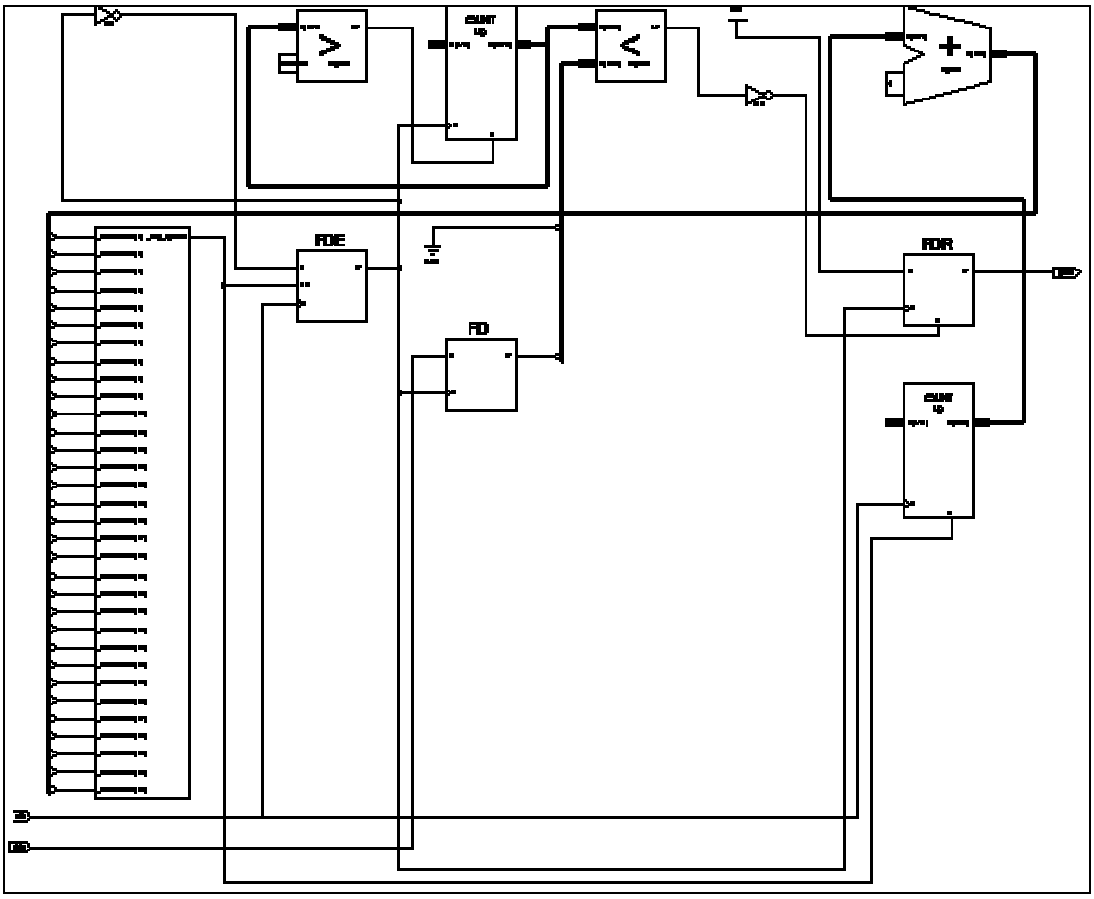
Eğer yazım hatası bulunmadıysa “Check Syntax”in yanına yeşil O.K. işareti oluşur. Eğer hata var ise uyarı vererek uyarıların ne olduğunu belirtir. Hatalar konulmayı unutulan “;” olabileceği gibi yanlış yazılan deyim ifadeleri de olabilir.

“View Synthesis Report” çift tıklanarak Syntax raporu detaylı olarak görülebilinir. Bu raporda ne kadar lojik kaynak kullanılacağı, uyarılar, hatalar, parametreler, HDL analiz gibi birçok analizin sonuçları elde edilir.

7. “View Technology Schematic” çift tıklanarak RTL devre şeması sentezlenmiş olarak elde edilir. Bu şemada LUT’lar, MUX’lar, FF’ler ve diğer lojik blokların bağlantıları yer almaktadır. Bu bağlantılar yazılan koda göre Xilinx-ISE programı tarafından otomatik yapılır. “View Technology Schematic” Xilinx-ISE programının sağlamış olduğu bir üstünlüktür; çünkü yazılan koda göre devre sentezlenmekte ve bu devre FPGA’ya yüklenmektedir. Şekil 3.7’de sentezlenen modülün blok diyagram görünümü verilmiştir.

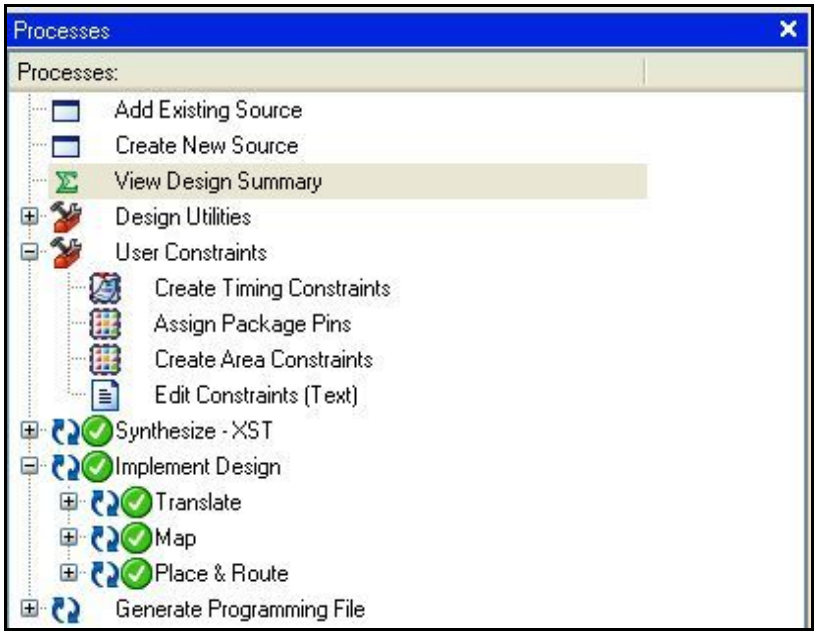
Şekil 3.7. PWM modülünün blok diyagram görünümü

İlk etapta sentezlenen devrenin genel bir blok diyagramı olarak, devrenin giriş çıkışlarının bulunduğu sembolik modül görünümü elde edilir. Bu modüle çift tıklanarak devrenin iç tasarımı ve bağlantı yolları incelenebilir. Devre içerisine yerleştirilecek lojik blokların ve ara bağlantıların şematik gösterimi Şekil 3.8'de verilmiştir.



Şekil 3.8. PWM modülünün şematik gösterimi

8. “Implement Design”a çift tıklayarak devrenin kalan kısımlarının analizleri tamamlanır bu kısımlar “Translate, Map, Place&Route”dir. Şekil 3.9’da görüldüğü gibi bu tamamlama analizinden sonra “Translate, Map, Place&Route” nın yan tarafında yeşil O.K. işareti gözükmelidir. Devre hatasız ve uyarısız, analizden çıkmalıdır. “Implement Design”dan sonra “Design Summary” kontrol edilir ve eksikliler varsa giderilmeye çalışılır.



Şekil 3.9. “Implement Design” analizi yapıldıktan sonra processes penceresinin durumu

9. Yazılımsal olarak tasarlanan devre dış dünya ile iletişiminin sağlanması için FPGA çekirdeği üzerinde giriş ve çıkışların pin atamalarının yapılması gerekir. Pin ataması yapmadan önce kullanılan giriş sinyallerinin nereden verileceğinin, çıkış sinyallerinin nereden alınacağının FPGA’nın katalog bilgisine bakılarak belirlenmesi gerekir. Pin atamaları için aşağıdaki aşamalar izlenir:

“Source” penceresinden PWM kodunun yazılı olduğu modül seçilir.

“Processes” penceresine yer alan “User Constrains” grubu içindeki “Assign Package Pins” tabına çift tıklanarak “The Xilinx Pinout and Area Constraints Editor (PACE)” açılır.

“Architecture View” tabı seçilerek kullanılan FPGA aygıtının üstten mimari görünümü elde edilir.

“Design Object List” penceresinde “Loc” kolonunda her bir giriş çıkışın karşısına gerekli pin ataması yapılır. PWM modülünde yer alan giriş çıkışların pin atamaları ise şöyledir:

Yerel saat darbesi olan “clk” karşısına “T9”,

“giris<0>” karşısına “F12” (kit üzerinde SW0),

“giris<1>” karşısına “G12” (kit üzerinde SW1),

“giris<2>” karşısına “H14” (kit üzerinde SW2),

“giris<3>” karşısına “H13” (kit üzerinde SW3),

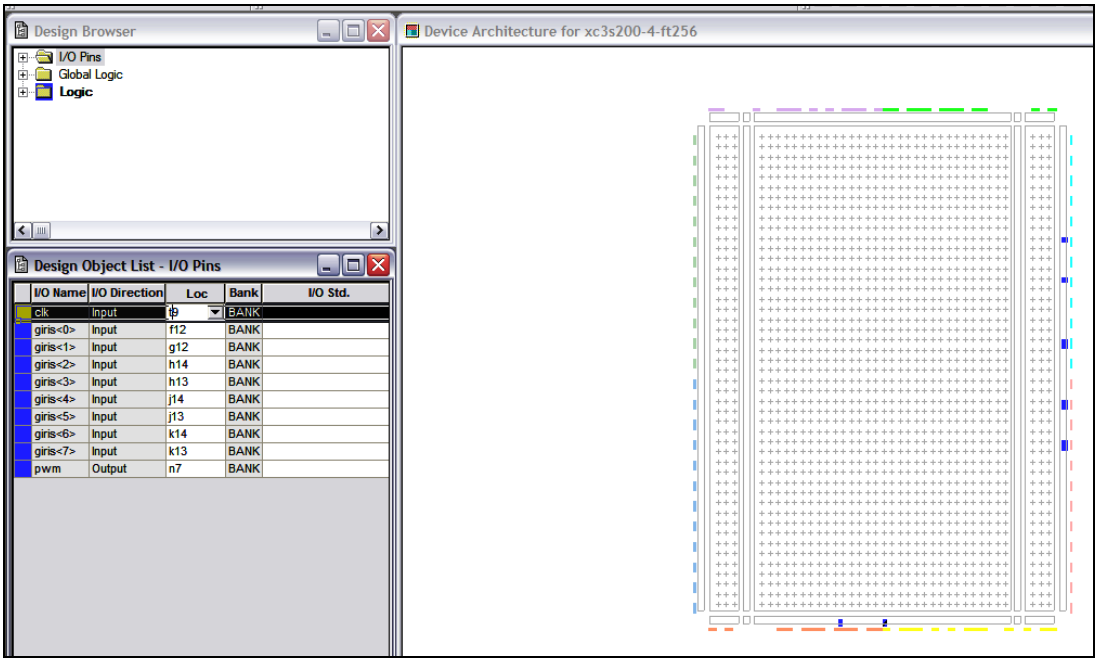
“giris<4>” karşısına “J14” (kit üzerinde SW4),

“giris<5>” karşısına “J13” (kit üzerinde SW5),

“giris<6>” karşısına “K14” (kit üzerinde SW6),

“giris<7>” karşısına “K13” (kit üzerinde SW7),

“pwm” karşısına ise “N7” (kit üzerinde IO0) yazılır.



Şekil 3.10. The Xilinx Pinout and Area Constraints Editor (PACE) penceresi

Şekil 3.10'da görüldüğü gibi pin atamaları yapılan adreslerin aygıt mimari görünümünde mavi renge döndüğü görülmektedir.

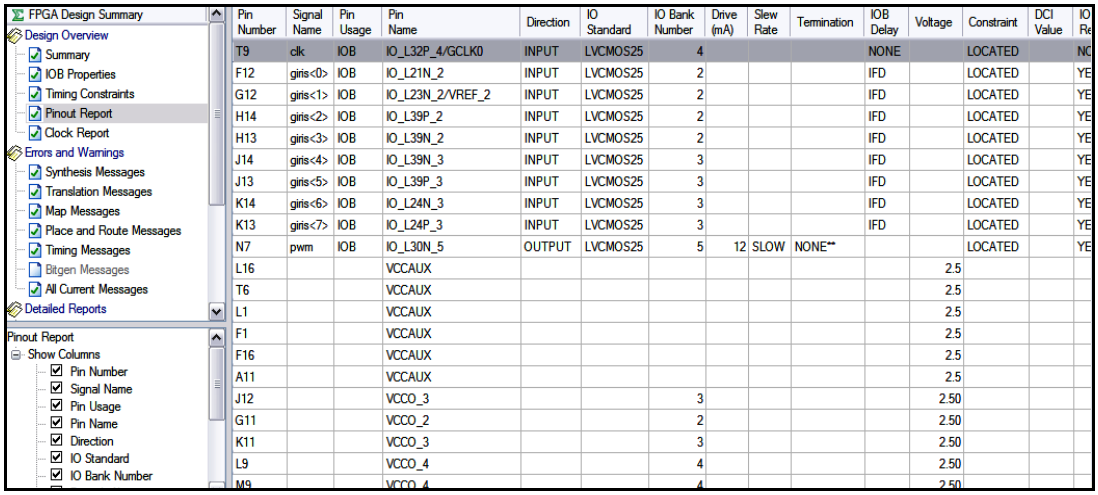
Oluşturulan pin bağlantı dosyası kaydedilir. Kaydetme esnasında karşımıza çıkan "Bus Delimiter" penceresinden "XST Default <>" seçilir ve "OK" e tıklanır. "PACE" kapatılır.

Implement Design"da turuncu soru işareti olan kısımların UCF dosyası oluşturulduğundan kalktığı görülür.

10. Pin bağlantılarının "Design Summary"den kontrol edilmesi:

Processes penceresinden "View Design Summary" çift tıklanarak "Design Summary" açılır.

"Signal Name" kolonundan "Pinout Report" seçilir ve oluşturulan pin bağlantıları rapor halinde Şekil 3.11'deki gibi ayrıntılı halde görülür.



Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number	Drive (mA)	Slew Rate	Termination	IOB Delay	Voltage	Constraint	DCI Value	IO
T9	clk	IOB	IO_L32P_4/GCLK0	INPUT	LVCMOS25	4				NONE		LOCATED		NC
F12	giris<0>	IOB	IO_L21N_2	INPUT	LVCMOS25	2				IFD		LOCATED		YE
G12	giris<1>	IOB	IO_L23N_2/VREF_2	INPUT	LVCMOS25	2				IFD		LOCATED		YE
H14	giris<2>	IOB	IO_L39P_2	INPUT	LVCMOS25	2				IFD		LOCATED		YE
H13	giris<3>	IOB	IO_L39N_2	INPUT	LVCMOS25	2				IFD		LOCATED		YE
J14	giris<4>	IOB	IO_L39N_3	INPUT	LVCMOS25	3				IFD		LOCATED		YE
J13	giris<5>	IOB	IO_L39P_3	INPUT	LVCMOS25	3				IFD		LOCATED		YE
K14	giris<6>	IOB	IO_L24N_3	INPUT	LVCMOS25	3				IFD		LOCATED		YE
K13	giris<7>	IOB	IO_L24P_3	INPUT	LVCMOS25	3				IFD		LOCATED		YE
N7	pwm	IOB	IO_L30N_5	OUTPUT	LVCMOS25	5	12	SLOW	NONE**			LOCATED		YE
L16			VCCAUX								2.5			
T6			VCCAUX								2.5			
L1			VCCAUX								2.5			
F1			VCCAUX								2.5			
F16			VCCAUX								2.5			
A11			VCCAUX								2.5			
J12			VCCO_3			3					2.50			
G11			VCCO_2			2					2.50			
K11			VCCO_3			3					2.50			
L9			VCCO_4			4					2.50			
M9			VCCO_4			4					2.50			

Şekil 3.11. Pin bağlantı raporu

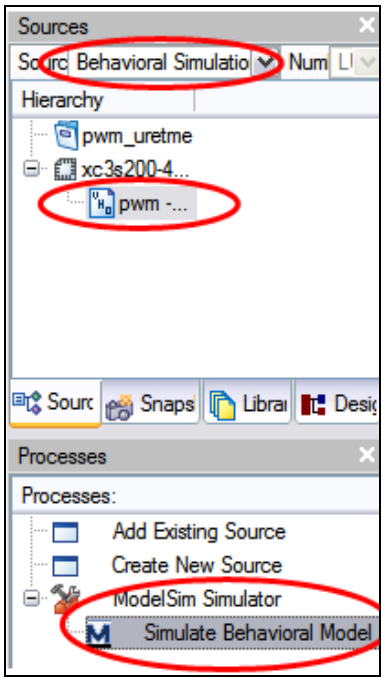
Raporda "Signal Name" tıklanarak kullanılan giriş çıkışlar üst tarafa sırayla yerleşir ve pin bağlantılarının doğruluğu kontrol edilir. Kontrol edildikten sonra pencere kapatılır.

3.4. Tasarlanan Modülün Simülasyonu

Modelsim Simulator'da kodun davranışı test edilebilir. Modelsim Simulator gelişmiş bir simülasyon programı olup birçok uygulama sonuçlarını gözlemlemede kullanılmaktadır.

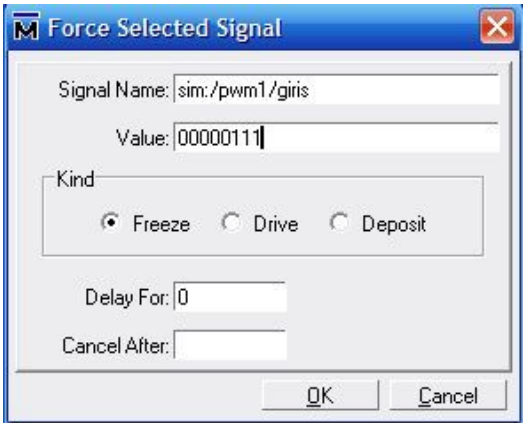
"Sources" penceresinden "Sources for:" kısmından "Behavioral Simulation" seçilir.

"Processes" penceresinden Modelsim Simulator'un altında yer alan "Simulate Behavioral Model" çift tıklanarak Modelsim açılır (bknz. Şekil 3.12).



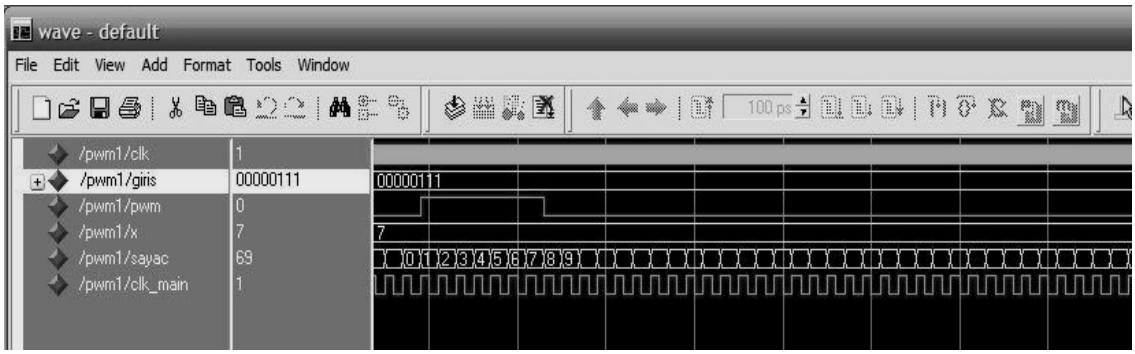
Şekil 3.12. Modelsim simülatörünün çalıştırılması

Modelsim açılınca devrenin giriş çıkışlarının verilmesi gerekir. Önce saat darbesi verilir; bunun için "clk" üzerinde sağa tıklanarak "Clock" seçilir ve test edilmek istenen saat sinyali ayarlanır. Saat sinyali ayarlanınca "giris" sinyalini de ayarlamak için "giris" üzerinde sağa tıklanır ve "Force" seçeneği seçilir. Gelen pencerede girişin 8 bitlik değeri girilir (bknz. Şekil 3.13). Böylece devrenin istenilen giriş değerleri girilmiş olur.



Şekil 3.13. Giriş değerinin girilmesi

Simülasyonda oluşan dalga şekillerini rahat bir şekilde görebilmek için "Wave" penceresi tam ekran yapılır. Simülasyon adım adım çalıştırılarak devrenin durumu gözlemlenebileceği gibi, her saat darbesinden sonraki duruma da gözlemlenebilir.



Şekil 3.14. Simülasyon sonucu oluşan dalga şekilleri

Şekil 3.14'te verilen simülasyon sonucundan anlaşılacağı gibi girilen değere göre PWM dalga şekli değişmektedir. "clk_main" oluşması için "clk" yüksek miktarlarda saydırıldığından düz bir hat gibi gözükmektedir. Kodda belirtilen ve sayaç olarak kullanılan "sayac" ve "x" in nereden nereye kadar saydığı tespit edilebilmektedir.

Gerekli simülasyon yapıldıktan sonra Modelsim kapatılabilir. Aynı anda birden fazla Modelsim çalıştırılamaz, bundan dolayı işi biten Modelsim kapatılmalıdır.

3.5. Spartan-3 Deneme Kitine Yükleme Yapılması

Son adım yazılan kodun uygulama kitine yüklenmesidir. Aşağıdaki işlemler izlenerek yükleme yapılır:

1. 5 V DC adaptör J4 bağlantısından kite bağlanır.

2. Yazılan programın kite aktarılması için ara kablo J7 bağlantısına ve bilgisayara bağlanır.

3. Sources penceresinden “Synthesis/Implementation” seçilir.

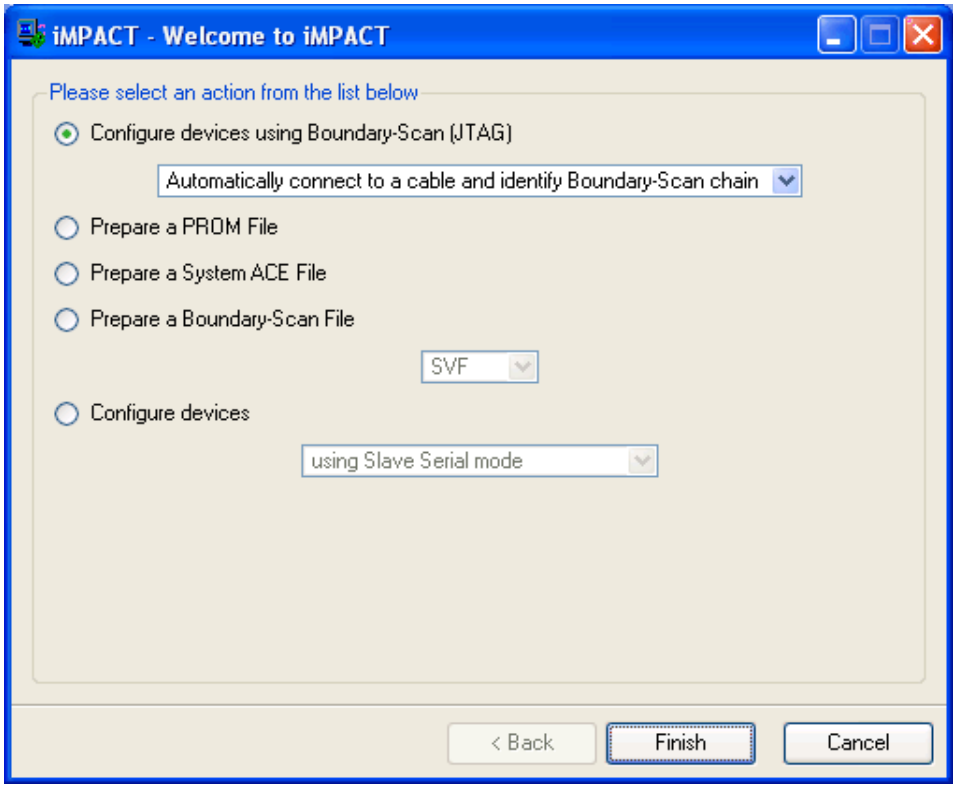
4. PWM kodunun yazılı olduğu modül seçilir.

5. Processes penceresinden “Generate Programming File” yanındaki “+” işaretine basılarak alt menü açılır.

6. “Configure Device (iMPACT)” seçilerek çift tıklanır.

7. “The Xilinx WebTalk Dialog” kutusu açılabilir bunda “Decline” basılır.

8. Gelen kutuda ise “Disable the collection of device usage statistics for this project only” seçeneği seçilir ve OK’e tıklanır.



Şekil 3.15. İMPACT Welcome Dialog kutusu

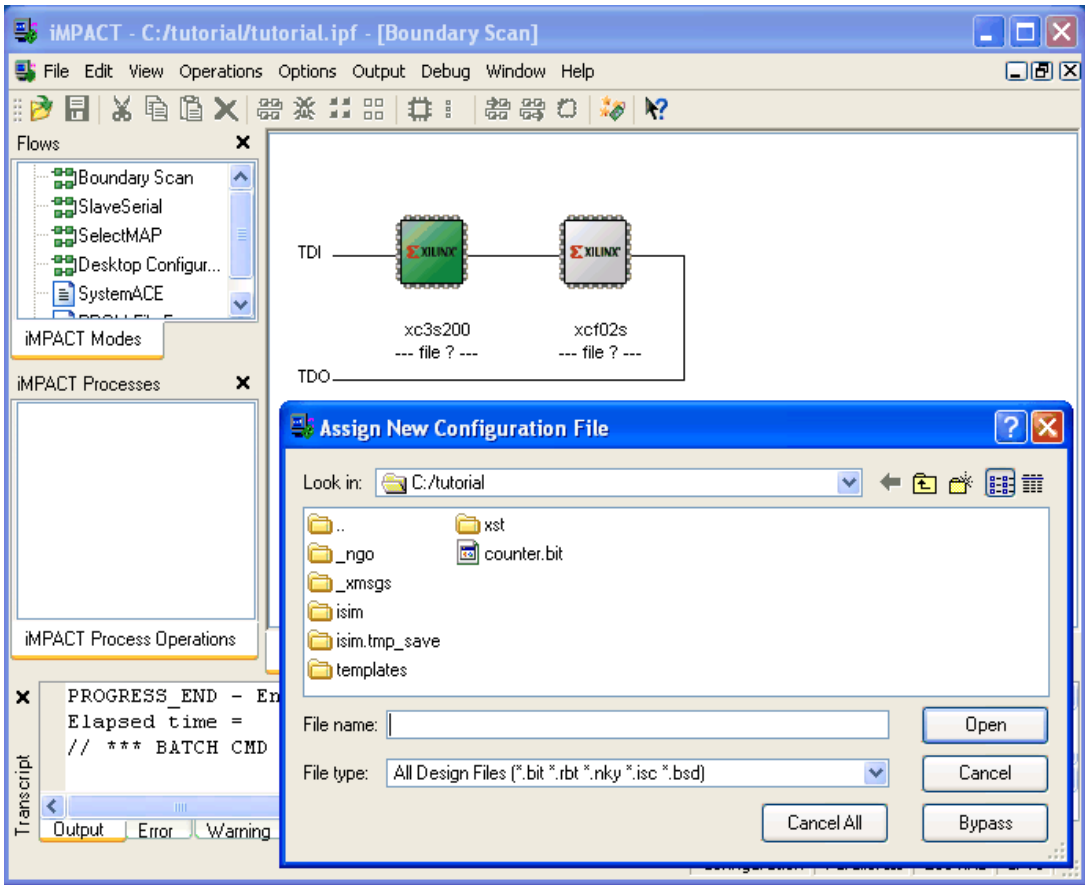
9. Şekil 3.15’teki gibi gelen İMPACT kutusunda “Configure devices using Boundary-Scan (JTAG)” seçeneği seçilir.

10. Ardından “Automatically connect to a cable and identify Boundary-Scan chain” seçilir.

11. “Finish”e tıklanır.

12. Eğer iki aygıt bulunduğuna dair mesaj gelirse “OK”e tıklanır ve devam edilir. Aygıtlar “JTAG” zinciri oluşturur ve ekranda bu zincir görülür.

13. Şekil 3.16’da verildiği gibi “Assign New Configuration File” diyalog kutusu gözükür. Konfigürasyon dosyası olarak “pwm.bit” seçilir ve OK’e tıklanır.



Şekil 3.16. Yeni konfigürasyon dosyası onaylama

14. Eğer uyarı mesajı gelirse OK’e tıklanır.

15. Daha sonra gelen kutuda sadece “Bypass”a basılır.

16. Ekranda görülen “JTAG” zincirinden xc3s200 aygıtının üzerinde sağ tıklanır ve “Program” seçeneği seçilir, gelen “Programming Properties” diyalog kutusu onaylanır.

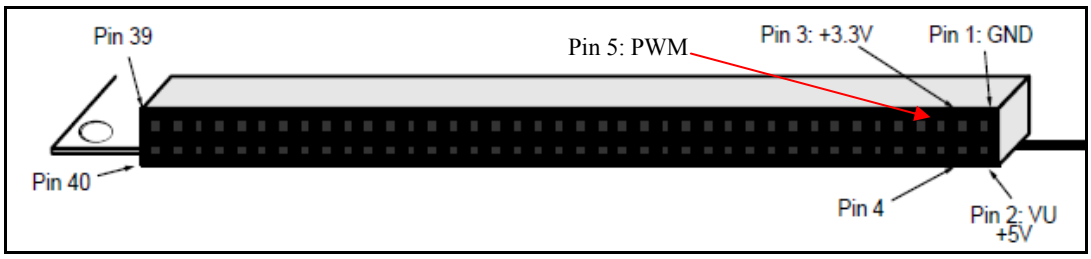
17. Program aygıtında OK'e tıklanır.

Program yüklemesi tamamlandığında "Program Succeeded" mesajı ekranda görülür.

Şekil 3.17. Yükleme mesajı

3.6. Yüklenen Programın Spartan-3 FPGA Üzerinde Çalıştırılması

Program yüklenmeden önce kite servo motor sürücü devresiyle birlikte bağlanmalıdır. Bu devre ani akım değişikliklerinin olmasına karşı FPGA çıkışlarını korumakla birlikte motorun çalışması için gerekli olan akımı sağlar. Şekil 3.18'de verildiği gibi PWM çıkışı A1 genişleme portu bağlantılarından IO0'dan alınır, IO0 ucu 5. bağlantı pinindedir. Sürücü devrenin beslemesi 3. bağlantı pininde yer alan +3.3 V'tan alınır. Devrenin FPGA kısmındaki toprak da FPGA'nın 1. bağlantı pinindeki toprağa bağlanmalıdır.

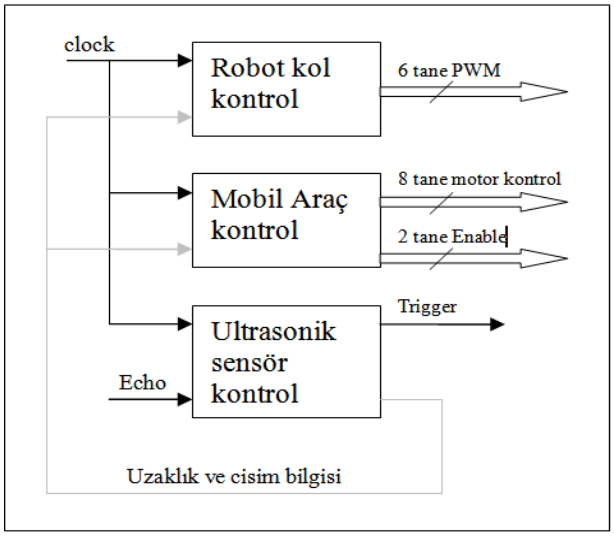


Şekil 3.18. FPGA kiti üzerinde yer alan giriş çıkış portu ve motor bağlantı pinleri

Program yüklendikten sonra FPGA kiti üzerinde yer alan 8 adet anahtarla PWM sinyalinin doluluk boşluk oranı ayarlanır ve motorun dönme açısı kontrol edilir.

4. FPGA KONTROL BLOKLARI

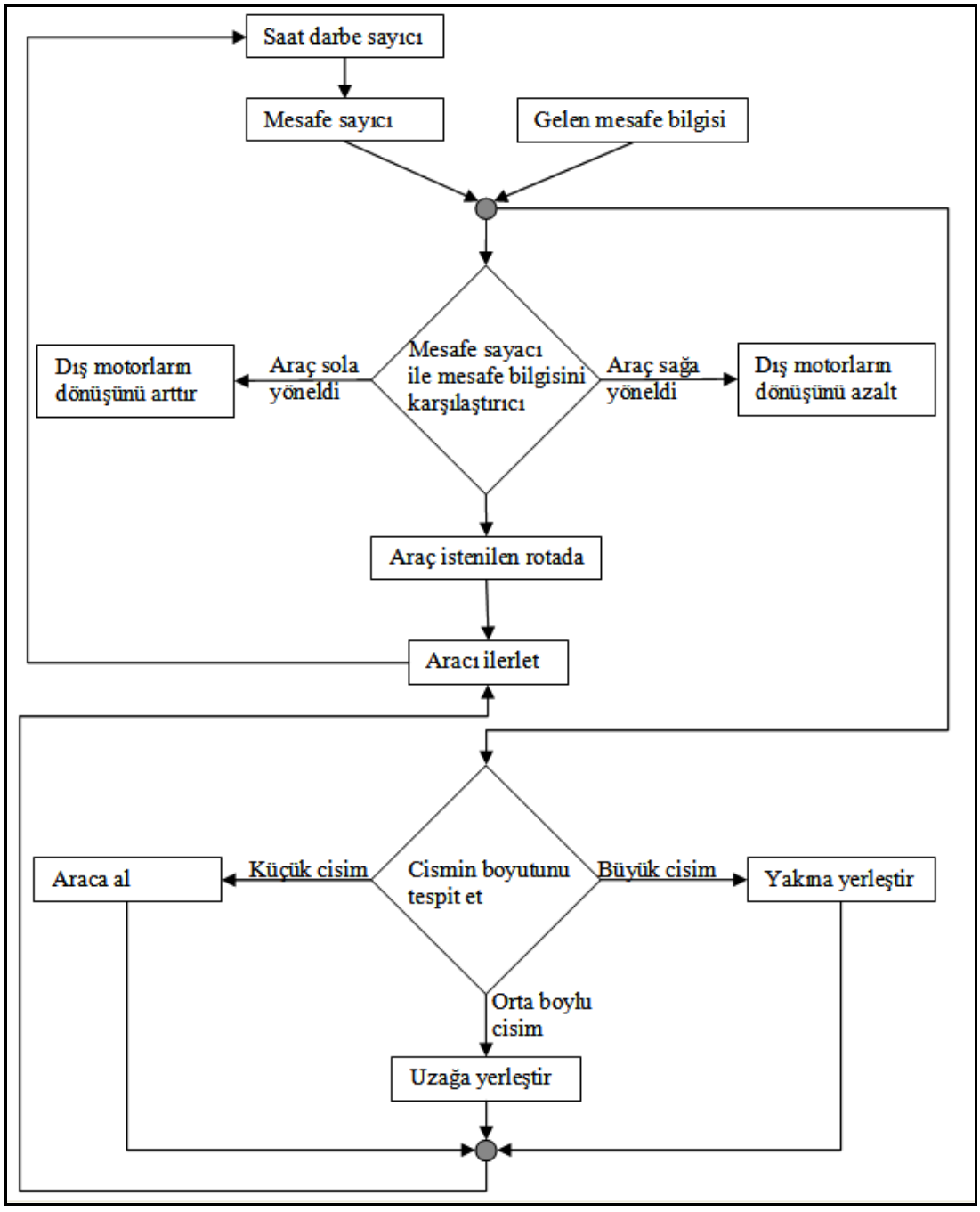
Mobil robot kolunun kontrolü için FPGA üzerinde gerçeklenen donanım tasarımı kullanılmıştır. FPGA'nın yüksek kaynak kapasitesi, hızı ve esnekliği uygulama geliştirme için oldukça elverişlidir. Bu mobil robot kolu da FPGA kullanılarak tasarlanan robotik uygulamalara güzel bir örnek oluşturmaktadır. Mikro denetleyiciler ve DSP'ler yıllarca motor kontrolünde kullanılmıştır. Bu aygıtlar tasarımla tümleşik oluyordu ve sadece yazılımsal bazı güncellemelere olanak sağlıyordu. Fakat özel uygulamalar için sınırlı olanakları vardı [10]. FPGA' ların sağladığı esneklik ile motor çeşitlerine göre kontrol stratejileri kolaylıkla tekrar ayarlanabilmektedir [11]. Motor kontrolleri için ardışıl olarak işleyen bir yazılım kodu oluşturulabilir ama FPGA'nın paralel işlem yapma özelliğinden dolayı aynı anda çok sayıda prosesi paralel olarak çalıştırabilme üstünlüğü vardır.



Şekil 4.1. Kontrol kodu blok diyagramı

Şekil 4.1'de verilen blok diyagramdan anlaşılacağı üzere ultrasonik sensör, robot kolu ve mobil araç kontrolü olmak üzere üç adet proses aynı anda çalışmaktadır. Çalışan prosesler sonucu oluşan toplam 17 adet sinyal aynı anda kontrol edilmektedir. Bu sinyallerden 6 tanesi robot kolunu oluşturan servo motorlar için PWM, 8 tanesi mobil araçtaki 4 adet DC motor için kontrol ve enable sinyali ile birlikte ultrasonik sensörün çalışması için trigger tetikleme sinyali olarak

belirlenmiştir. Tüm bloklar FPGA'dan alınan saat darbesiyle çalışmaktadır. Ultrasonik sensörden alınan uzaklık bilgisi echo (yankı) ucundan ultrasonik sensör kontrol bileşenine gelerek mesafe ve cisim boyu yorumlanarak diğer bloklara bu veriler girdi olarak gönderilir. Bu girdiler de bu bloklarda yorumlanarak kontrol çıkışları üretilir.



Şekil 4.2. Mobil robot kolu kontrol akış şeması

Mobil robot kolunun çalışması Şekil 4.2'de verilen akış diyagramından anlaşılabilir. Hareket halinde olan mobil araç, konumunu ultrasonik sensörden gelen bilgiye göre ayarlar. Tasarım mimarisi içinde yer alan mesafe sayacı sayesinde, gelen mesafe bilgisi sürekli olarak ilgili set değeriyle karşılaştırılır ve tekerleklerin dönüş hızı buna göre belirlenir. Tekerlek dönüş hızları ise üretilen PWM kontrol sinyaline göre ayarlanır.

Mobil robot hareket esnasında cismi algılarsa cismin boyu tespit edilir ve robot kolu pozisyon komutları bu bilgiye göre şekillenir. Aynı zamanda cismin arabaya olan uzaklığına göre de robot kolu cisme uzanır. Mesafe bilgisi, tasarım algoritmasında çeşitli matematiksel işlemler yardımıyla robot kolunun konumunu belirlemede kullanılır.

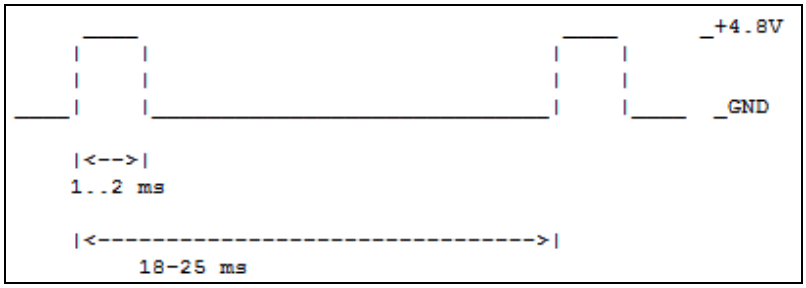
Mobil robot; robot kolu, mobil araç ve ultrasonik sensörden oluşmaktadır. Bu çalışmada tasarımın genel bileşenleri de üç ana kısımda incelenebilir:

- Robot kolu kontrol bloğu
- Mobil araç üzerindeki DC motor kontrol bloğu
- Ultrasonik sensör kontrol bloğu

Ultrasonik sensör kontrol bloğundan gelen veriye göre diğer blokların akış algoritması şekillenir. Araç mobil olarak hareket halinde iken sürekli olarak mesafe ölçümleri değerlendirilir ve cisim algılandığında cismin üzerinde işlem yapılır. Cisim mobil araca alınır veya cismin yeri değiştirilir. Mobil araç, hareket güzergâhındaki cisimler bitene kadar bu işlemleri yapmaya devam eder.

4.1. Robot Kolu Kontrol Bloğu

Robot kolunun hareketini sağlayan servo motor kontrolü için saniyede 50 defa tekrar eden ve 1 ile 2 ms arasında doluluk boşluk oranına sahip olan PWM sinyali kontrol edilir (Bknz. Şekil 4.3). 5 eksenli robot kolu için 5 farklı PWM, FPGA üzerinde kolayca üretilebilir. Servo motorlar için gerekli olan sinyalin periyot çerçevesi 20 ms olmalıdır. FPGA'nın çalışma frekansı 50 MHz olduğu için kontrol bloğu içinde sayıcı yardımıyla 20 ms'lik yerel saat darbesi oluşturulmalıdır.



Şekil 4.3. Servo motor için PWM sinyali

PWM sinyali ise 8 bitlik çözünürlüğe sahip ikili sayıcı kullanarak oluşturulur. PWM sinyali için gerekli matematiksel eşitlikler aşağıdaki gibi ifade edilir [12]:

$$Frame\_Period = T_{CLK} \times TC \tag{4.1}$$

$$Duty\_Cycle = \frac{Mark\_Value}{2^n} \tag{4.2}$$

Denklem (4.1)'de verilen, T_{CLK} 50 MHz'lik FPGA'nın sağladığı saat sinyalinin periyodudur ve bu periyot servo motoru sürmek için gerekli olan PWM sinyali için oldukça düşüktür. Frekans bölme yöntemi kullanılarak 20 ms'lik periyota sahip yeni bir saat sinyali üretilir. TC sabit sayısı ile FPGA'nın saat sinyalinin periyotu çarpılarak yeni saat sinyalinin periyotu olan $Frame\_Period$ elde edilir.

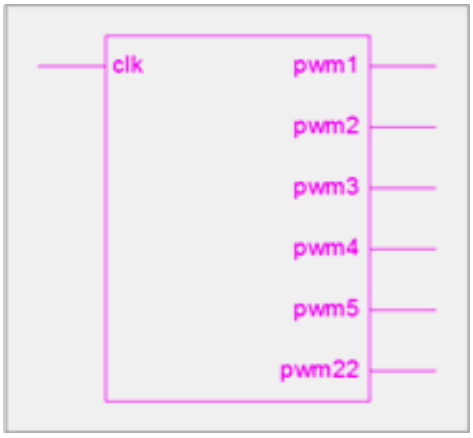
İfade de geçen $Duty\_Cycle$, PWM sinyalinin doluluk boşluk oranıdır. Bu orana göre servo motorlar konum alırlar. İfadede geçen $Mark\_Value$, PWM sinyalinin doluluk miktarını belirlemekte kullanılan, mesafe bilgisinden gelen veridir. PWM sinyalinin çözünürlüğü 8 bit olduğu için $Mark\_Value$, 256'ya bölünür. Aracın cisme olan uzaklığına göre $Mark\_Value$ değeri değişerek motorların açısı belirlenir.

Tablo 4. 1. Mark_Value değerine göre motorların açıları ve gönderilen sinyallerin süreleri

Mark_Value	8	10	12	13	14	16	18	20	22	23	24	26	28
Açı (derece)	-90	-72	-54	-45	-36	-18	0	18	36	45	54	72	90
Süre (ms)	1.0	1.1	1.2	1.25	1.3	1.4	1.5	1.6	1.7	1.75	1.8	1.9	2.0

Servo motorların robot kolunda bulunduğu yere göre açı alması gerekir. Tablo 4.1'de verilen veriler kullanılarak tasarımın içerisinde motor açılarını ve dolayısıyla robot

kolunun konumunu ayarlayan modül yerleştirilir. Modülün tasarımında ise mesafe bilgisine ve cismin boyuna göre kolun konumunun ayarlaması için bak oku tablosu oluşturulur.



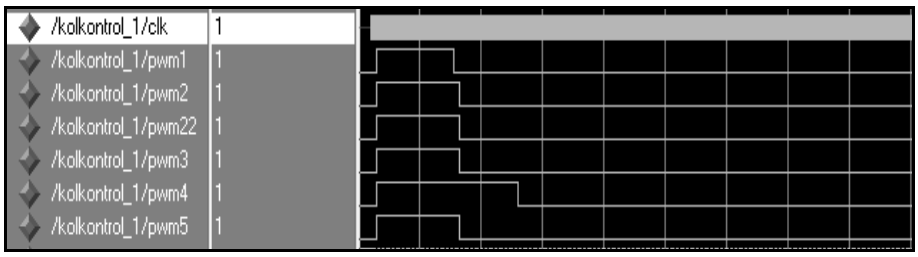
Şekil 4.4. Robot kolu
VHDL kontrol bloğu

Robot kolunun tasarım teknolojisini XILINX-ISE programının View Technology Schematic prosesi çalıştırılarak görülebilir. Şekil 4.4'te görüldüğü gibi tasarım teknolojisinin blok şeklinde görünümü bu prosesin çalıştırılmasıyla elde edilir. Devreye giren clk sinyalinden 6 adet PWM sinyali elde edilir. Robot kolunun omuz bölümüne cisim kaldırma esnasında fazla yük bindiğinden dolayı 2 adet servo motor kullanılmıştır ve bu servo motorlar da pwm2 ile pwm22 sinyalleri ile kontrol edilir. Bloktan elde edilen diğer PWM sinyalleri ise robot kolunun taban, dirsek, bilek ve tutucu kısımlarını kontrol etmektedir. PWM sinyali atama bilgilerine ilişkin tablo, Tablo 4.2'de verilmiştir.

Tablo 4. 2. Robot kolunda bulunan
motorlara giden pwm sinyalleri

Motor ismi	PWM sinyali
Taban motoru	pwm1
1. Omuz motoru	pwm2
2. Omuz motoru	pwm22
Dirsek motoru	pwm3
Bilek motoru	pwm4
Tutucu motoru	pwm5

Modelsim simülasyon programı kullanılarak, VHDL aracılığı ile yazılan donanım kodundan elde edilen sentezleme sonucunun benzetimi yapılmıştır. Şekil 4.5'te görüldüğü gibi, verilere bağlı olarak robot kolu kontrolü için gerekli PWM'ler oluşturulan bu VHDL modül üzerinde üretilebilmektedir.



Şekil 4.5. Robot kolu modelsim çıktıları

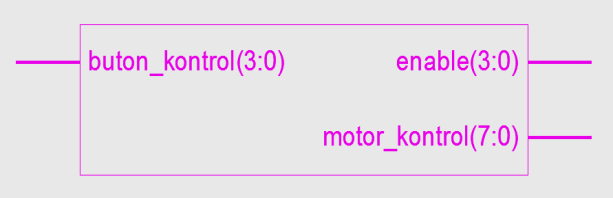
4.2. Mobil Araç Üzerindeki DC Motorların Kontrol Bloğu

Robot kolunun gezgin olmasını sağlamak için DC motorlara bağlı tekerleklerle hareket edebilen bir platform gerçekleştirilmiştir. Motorun dönme hızı ve yönü, girişlere verilen sinyale göre ayarlanabilir. Giriş sinyalleri FPGA'da tasarlanan mobil araç motor kontrol bloğu ile kolaylıkla takip edilmekte ve mesafe bilgisine göre aracın konumu ayarlanabilmektedir.

Tablo 4. 3. Mobil araç yön kontrol tablosu

Mobil araç hareketi	Sol tekerlekler	Sağ tekerlekler
Sola yönelme	Dönüş hızını arttır	Dönüş hızını azalt
Sağa yönelme	Dönüş hızını azalt	Dönüş hızını arttır
Düz ilerleme	Sabit hız	Sabit hız

Mobil aracın hareket algoritmasını oluşturmak için gerekli durum bilgileri Tablo 4.3'te verilmiştir. Mobil aracın ilerleme esnasında üç farklı durumu bulunmakta ve bu durumlara göre sağ ve sol tekerleklerin dönüş hızları ayarlanmaktadır. Ultrasonik sensörden gelen veriye göre araç herhangi bir tarafa yönelme yaparsa ilgili taraftaki tekerleklerin dönüş hızı arttırılır diğer taraftaki tekerleklerin dönüş hızı ise azaltılarak aracın düz rotada ilerlemesi sağlanır. Tekerleklerin dönüş hızına göre mobil araç yön almakta ve bu işlem sürekli yapılmaktadır. Böylece mobil araç düz bir şekilde ilerleme yapabilmektedir.



Şekil 4.6. Mobil araç motor kontrol bloğu

FPGA' lerin bize sağladığı kolaylıklardan biri de bol miktarda giriş çıkış portunun olmasıdır. Şekil 4.6'da görüldüğü gibi mobil aracın motorlarını kontrol etmek için toplam 12 tane sinyal oluşturulmuştur. Bu sinyallerden 8 tanesi motorlara giden giriş uçları, 4 tanesi ise enable yani yetkilendirme uçlarıdır. Robot kolu kontrolü için PWM sinyali oluşturulduğu gibi mobil aracın kontrolü için de PWM sinyalleri oluşturulmuştur. Ultrasonik sensörden gelen mesafe bilgisine göre motorlar, aracın konumunu belirlemek için dönüş yönünü ve hızını ayarlarlar.

4.3. Ultrasonik Sensör Kontrol Bloğu

Mobil robot kolunun ilerleme esnasındaki dengesini yani düz gitmesini sağlamak için ultrasonik sensörden gelen bilgiler kullanılır. Robot kolunun tutacağı cisimlerin boyutunu ölçmek için de aynı sensörden yararlanılır. Bu çalışmada kullanılan ultrasonik sensör modülü ortama ses dalgası gönderir, gönderilen dalga cisme çarpıp geri döner. Dalganın gidip gelme süresi yardımıyla cisimlerin uzaklığı hesaplanır. Yapılan bu çalışmada 3 cm ile 300 cm arasını sağlıklı ölçebilen, hazır bir ultrasonik sensör modülü kullanıldı. Sensörle ölçüm yapılan mesafenin hesabı Denklem (4.3)'te verilmiştir,

$$Mesafe = \frac{Toplam\_süre \times ses\_hızı}{2} \tag{4.3}$$

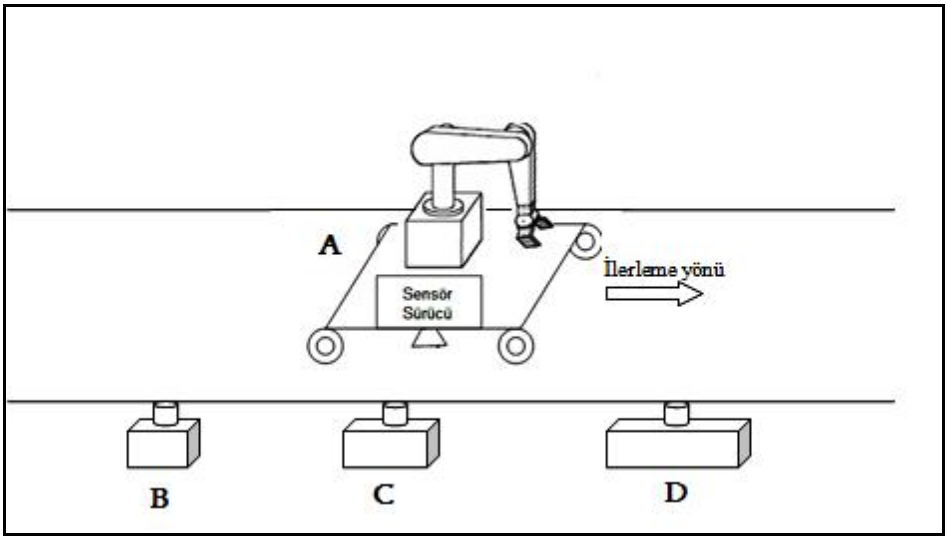
Gönderilen ses sinyali engele çarpıp döneceği için ölçülen mesafe iki kat olur. Bunun için formülde ikiye bölme yer almaktadır. Toplam süre ise FPGA'nın çalışma frekansı ile ilgili olarak değişebilir. Spartan XC3S200 FPGA'nın çalışma saat frekansı 50 MHz'dir.

Bu uygulamada DYP-ME007 ultrasonik sensörü kullanıldı. Sensörün çalışması için trigger yani tetikleme ucuna 10 µS'lik darbe uygulanır. Uyarılan sensör 8 tane ses

dalgasını gönderir ve bekler. Echo ucu yani yansıma ucu ses dalgası gönderilir gönderilmez lojik 1 olur ve dalga gelene kadar bekler, geri dönen dalga algılanınca echo ucu lojik 0 olur. Echo ucunun 1 olduğu zamanı hesaplayarak mesafenin ölçümü yapılabilir.

5. CİSİMLERİN TESPİT EDİLMESİ

Mobil robot kolunun tasarlanmasında sabit platforma bağlı olmayan robot kolu ile cisimlerin tutularak yerlerinin değiştirilmesi hedeflenmiştir. Ultrasonik sensör, mobil aracın konumunu belirlemede, cisimleri tespit etmede ve cisimlerin boylarını ölçmede kullanılmaktadır.



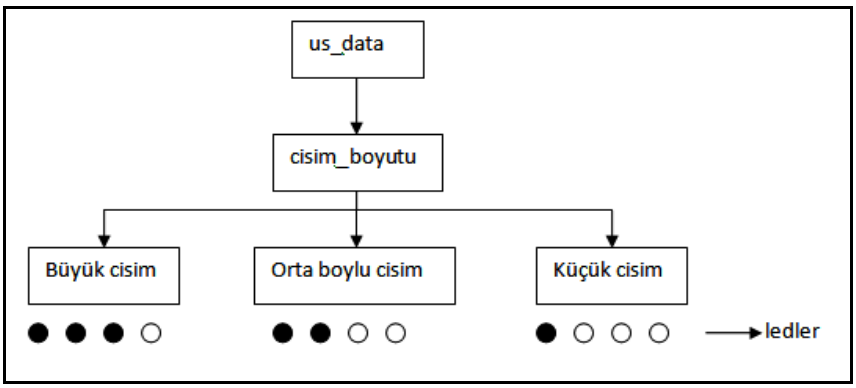
Şekil 5.1. Cisim tespit edilmesine yönelik blok diyagram

Şekil 5.1'de görüldüğü gibi mobil robot kolu (A) ilerleme yönünde ilerlerken yol üstündeki farklı boyuttaki cisimleri (B, C, D) ayırt edebilmektedir. Yol üzerine yerleştirilen üç farklı boydaki cisimler dikdörtgenler prizması şeklinde olup, üzerlerine robot kolunun tutması için silindir şeklinde tutacak monte edilmiştir. Yol üzerinde yer alan dikdörtgenler prizması şeklindeki cisimleri boylarına göre; küçük cismi aracın üstüne, orta boylu cismi diğer tarafta arabadan uzak bir yere ve büyük olanı ise diğer tarafta arabaya yakın bir yere yerleştirebilmektedir. Cisim yerleştirme durumlarını uygulamalara göre farklılıklar gösterebilir. Cisimleri ayıklama durumları Tablo 5.1'de verilmiştir.

Tablo 5. 1. Cisim tespit ve yerleştirme tablosu

Cismin şekli	Büyük cisim	Orta boylu cisim	Küçük cisim
Cismin yerleştirileceği yer	Aracın diğer yakın tarafı	Aracın diğer uzak tarafı	Araç üzeri

Mobil robot kolu akış şemasının bir bileşeni olarak ultrasonik sensörle, cisimlerin boyu Şekil 5.2'de verilen blok diyagram yardımıyla ölçülerek gruplanır.



Şekil 5.2. Cisim boyutu algılama işlemi

Spartan XC3S200 FPGA üzerinde yer alan 8 tane led kullanılarak cisimlerin boyları ölçülmüştür. Bu çalışmada sadece üç tane farklı boyutta cisim kullanıldığı için; cisimlerin boyları ledlerle ifade edildi. Cisimlerin boylarını ölçüm tekniği olarak Denklem (5.1) deki genel hız-konum eşitliğinden yararlanıldı;

$$cisim\_boyutu = araba\_hizi \times zaman \tag{5.1}$$

Bu eşitlikte arabanın hızına göre zaman ayarı yapılarak cisim boyu ölçülmektedir. Arabanın hızını tekerleklere binen yük miktarı, zeminin sürtünme katsayısı ve besleme gerilimindeki değişimler etkileyebilir. Spartan XC3S200 FPGA kitinin sağlamış olduğu çalışma frekansı kullanılarak, yeni bir sayıcı modülü tasarlanarak, bu sayıcının frekansına göre zaman ayarı yapılır.

6. SONUÇLAR ve ÖNERİLER

Xilinx-ISE paket programı yardımıyla tasarım özetini görerek kaynak kullanım sonuçları elde edildi. Tablo 6.1'de kaynak kullanım bilgileri ve oranları verilmiştir. Bu tablo sayesinde yapılan tasarımda kullanılan konfigüre edilebilir lojik blok (CLB) veya dilim (Slice) sayılarını, flip-flop sayısını, bak oku tablosu (LUT) sayılarını, giriş-çıkış port (IOB) sayısını ve bunların mevcut olan toplam kapasite miktarlarına oranlarını (% olarak) görebiliriz. Tasarım gereksinim sonuçları sayesinde yapılacak olan güncellemeler için aygıtın yeterli olup olmadığı da anlaşılabilmektedir. Bu tasarımda kullanılan Spartan XC3S200 FPGA yongasının, mobil robot kolu sayısal tasarım blokları için fazlasıyla yeterli olduğu anlaşılmaktadır.

Tablo 6. 1. Kaynak kullanım tablosu

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	348	3,840	9%
Number of 4 input LUTs	761	3,840	19%
Logic Distribution			
Number of occupied Slices	507	1,920	26%
Number of Slices containing only related logic	507	507	100%
Number of Slices containing unrelated logic	0	507	0%
Total Number 4 input LUTs	977	3,840	25%
Number used as logic	761		
Number used as a route-thru	216		
Number of bonded IOBs	26	173	15%
IOB Flip Flops	4		
Number of GCLKs	3	8	37%
Total equivalent gate count for design	10,942		
Additional JTAG gate count for IOBs	1,248		

Tasarımın FPGA üzerine gerçeklenmesi sırasında toplam 26 adet pin ataması yapılmıştır. Pin atamalarının tasarımdaki port isimleri ve giriş-çıkış durumu Tablo 6.2'de verilmiştir.

Tablo 6. 2. I/O Port isimleri ve durumları tablosu

I/O Port ismi	Port Durumu
Ackapa	input
Clk	input
echo	input
8 adet led	output
6 adet PWM	output
8 adet motor kontrol	output
Trigger	output

Bu çalışma sonunda, 5 eksenli mobil bir robot kolunun, üzerine monte edildiği mobil araçla birlikte kontrolü, VHDL donanım programlama dili yardımı ile yapılan tasarım yoluyla tek bir FPGA yongası üzerinde (Spartan XC3S200) donanımsal olarak gerçeklenmiş oldu. Sistem için gerekli diğer analog modüller (motor sürücü ve güç devreleri) ayrıca gerçekleştirildi ve aynı mobil platform üzerine monte edildi. Benzetimlerde Modelsim programı kullanıldı. Tasarlanan sistemde bileşen sayısının artmasına rağmen FPGA' in sağladığı esneklikle sistemler arası zamanlama açısından uyum kolaylıkla sağlandı. Tasarım FPGA deneme kiti üzerinde reel olarak gerçeklendi ve deneysel testler yapıldı. Mobil robot kolunun tüm beklenen işlevlerini başarıyla yerine getirdiği görüldü.

Sonuç olarak, donanım programlama dilleri yardımıyla yapılacak olan tasarımlarla, gelecekteki uygulama ve geliştirmeler için de örnek teşkil edecek çok eksenli esnek bir mobil robot kolu prototipi oluşturmuş oldu. Literatürde fonksiyonel olarak bu çalışmadaki robot kolu sistemi ile birebir örtüşen bir mobil robot kolu mekanizmasına henüz rastlanamadığından (hem FPGA tabanlı, hem mobil ve hem de 5 eksenli olması bakımından özgün bir çalışmadır), sadece performansa ve donanımsal kaynak kullanımına dayalı bir karşılaştırma yapılamamıştır. Ancak bu çalışma, tasarım teknolojisi (DSP, FPGA, veya Mikroişlemci gibi), mobil olup olmama durumu, kol serbestlik derecesi, cisim veya konum algılama metodolojisi gibi kriterler açısından, literatürdeki bazı örnek FPGA tabanlı çalışmalarla Tablo 6.5 üzerinde karşılaştırılmıştır.

Tablo 6. 3. Karşılaştırma tablosu

Yapılan Çalışmalar	Motor konumlarının belirlenmesi	Serbestlik Derecesi (DOF)	Robot Serbestliği	Kullanılan teknoloji
Bu çalışma	Ultrasonik sensör	5	Mobil	FPGA
[13]	Dönen kodlayıcı	5	Sabit	FPGA
[14]	Dönen kodlayıcı, PID, ters kinematik	4	Sabit	FPGA
[15]	Ultrasonik sensör, infrared sensör	Yok	Mobil	FPGA
[4]	Kablosuz olarak yeniden yapılandırma	4	Mobil	FPGA+ARM

KAYNAKLAR

[1] Kumar S., Shuklay A., Duttaz A., Beherax L., A model-free redundancy resolution technique for visual motor coordination of a 6 dof robot manipulator, *Intelligent Control Part of IEEE Multi conference on Systems and Control*, Singapore, 1-3 October 2007.

[2] Kung Y-S., Shu G-S., FPGA-based motion control IC for robot arm, *2nd International Conference on Information Technology*, Budapest, Hungary, 26 October 2005.

[3] Meshram U., Bande P., Dwaramwar P.A., Harkare R., Robot arm controller using FPGA, *Multimedia, Signal Processing and Communication Technologies International Conference*, Uttar Pradesh, India, 14- 16 March 2009.

[4] Xu M., Zhu W. ve Zou Y., Design of a reconfigurable robot controller based on FPGA, *Fifth IEEE International Symposium on Embedded Computing*, Beijing, China, 6-7 October 2008.

[5] Erden A., Atılım Üniversitesi Robot Teknolojileri Araştırma ve Uygulama Merkezi, http://kurumsal.library.atilim.edu.tr/pdfs/100430-sunum.pdf, (Ziyaret Tarihi: 10 Mayıs 2013).

[6] Karcı H., Tangel A. FPGA tabanlı 5-eksenli mobil robot kolu tasarımı ve uygulaması, *Elektrik-Elektronik ve Bilgisayar Mühendisliği Sempozyumu*, Bursa, Türkiye, 29 Kasım-1Aralık 2012.

[7] Arslan İ., L298 Motor Sürücü Entegresinin Kullanımı, Hacettepe Üniversitesi, Ankara, http://robot.ee.hacettepe.edu.tr/Dosyalar/makaleler/L298.pdf, (Ziyaret tarihi: 1 Mayıs 2013)

[8] Sırmaçek B., FPGA ile mobil robot için öğrenme algoritması modellemesi, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2007.

[9] URL-1: http://au.alibaba.com/product/Ultrasonic.html, (Ziyaret tarihi: 14 Nisan 2012)

[10] Monmasson E., Cirstea M., FPGA design methodology for industrial control systems a review, *IEEE Trans. on Industrial Electronics*, DOI: 10.1109/TIE.2007.898281.

[11] Meshram U. ve Harkare R., FPGA based five axis robot arm controller" *International Journal of Electronics Engineering*, 2010, **2**(1), 209-211.

[12] Lawman G., Pulse width modulation in xilinx programmable logic, *Xilinx Application Brief*, 1995.

[13] Meshram U., Bande P., Harkare R., Hardware and software co-design for robot arm position control using VHDL&FPGA" *IEEE Computer Society, International Conference on Advances in Recent Technologies in Communication and Computing*, Kerala, India, 27-28 October 2009.

[14] Cho U., Le N., Jeon W., An FPGA- based multiple- axis motion control chip, *IEEE Transactions on Industrial Electronics*, DOI: 10.1109/TIE.2008.2004671.

[15] Samuelsen D., Graven H., Low cost robots used for practical assignments in programming modules, *3rd IEEE International Conference on e-Learning in Industrial Electronics*, Porto, Potugal, 3-5 November 2009.

[16] Yılmaz U., Algılayıcılar, http://utkuonline.tripod.com/sensor.html, (Ziyaret Tarihi: 10 Nisan 2013).

EKLER

EK-1

Bu bölümde mobil robot kolu kontrolü için gerekli olan VHDL kodu verilmiştir.

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

entity mobil robot kolu is
Port ( clk : in std_logic;
       ackapa:in std_logic; --- kod yüklendikten sonra işlemin başlamı için start ---
anahtarı
       trigger:buffer std_logic;
       led:out std_logic_vector(7 downto 0);
       echo:in std_logic;
      motor_kontrol:out std_logic_vector(7 downto 0);
      pwm1 : out std_logic :='0';
      pwm2 : out std_logic :='0';
      pwm22 : out std_logic :='0';
      pwm3 : out std_logic :='0';
      pwm4 : out std_logic :='0';
      pwm5 : out std_logic :='0');
end mobil robot kolu;

architecture Behavioral of mobil robot kolu is

signal b:integer range 0 to 1000000;
signal kolkontrol :std_logic :='0';
signal d_sayac1 : integer := 0;
signal d_sayac2 : integer := 0;
signal d_sayac3 : integer := 0;
signal d_sayac4 : integer := 0;
signal d_sayac5 : integer := 0;
signal sayac1 : integer := 0;
signal sayac2 : integer := 0;
signal sayac3 : integer := 0;
signal sayac4 : integer := 0;
signal sayac5 : integer := 0;
signal asayac : integer :=0;
signal clk_main1 : std_logic :='1' ;

--------------------clk_main oluşturulur--------------------------
begin

process(clk,ackapa,echo)

   variable a:integer range 0 to 5000;
```

```
  variable temp3:integer range 0 to 1000;
  variable d:integer range 0 to 1000;

       variable temp:integer range 0 to 10000000;
       variable temp1:integer range 0 to 10000000;
       variable temp4:integer range 0 to 10000000;
       variable say:integer range 0 to 1000;
       variable say1:integer range 0 to 1000;
       variable say2:integer range 0 to 1000;
       variable say3:integer range 0 to 1000;
       variable say5:integer range 0 to 1000;
       variable say6:integer range 0 to 1000;
       variable olm:std_logic :='0';
       variable olm1:std_logic :='0';
       variable olm2:std_logic :='0';
       variable bekle:integer :=1;
       variable y:integer range 0 to 10000;
       variable bn:std_logic_vector(7 downto 0):="00000000";
       variable bn1:std_logic_vector(6 downto 0):="0000000";

      begin

      if(ackapa='1')then          --tüm sistemi açıp kapamak için

        if(clk'event and clk='1')then
             y := y+1;
                        if y = 2000 then
                              clk_main1 <= not(clk_main1);
                              y:= 0;
                        end if;

                    b<=b+1;
                    temp3:=temp3+1;
                    if(temp3=500)then
                          a:=a+1;
                          temp3:=0;
                          if(a=3999)then
                          trigger<=not trigger;
                          elsif(a=4000)then
                          trigger<=not trigger;
                          b<=0;
                          a:=0;
                          end if;

                      end if;
      end if;

if( echo'event and echo='0')then
           d := d+1;
```

```
      if(d=1)then
        d:=0;

        if (10000 < b ) and (b < 60000) and olm='0' then   ---mesafe belirlenen aralikta
ise
                olm2:='1';
                say5 := 0;
                say:=say+1;
        say1:=say;
                bn := conv_std_logic_vector(say1,8);
                bn1 := bn(7 downto 1);
                say2 := conv_integer(bn1);
-------------------------------------------------------------
                if say6 < 15 then
                        say3 := say2+6;
                else
                        say3 := say2+7;
                end if;
-------------------------------------------------------------

          else
                say:=0;

                        if say2/=0 then
                        say3:=say3-1;
                                        if say3 = 0 then
                                        motor_kontrol <= "11111111"; --arabayi durdur

                                        kolkontrol <= '1';
                                        bekle:=500;
                                        olm := '1';
                                        olm1 := '1';
                                        olm2 :='1';
                                        say1:=0;
                                        say2:=0;
                                        say6:=0;
                                        else
                                        if olm1 = '0' then
                                        motor_kontrol <= "01010101"; --arabayi gerilet
                                                kolkontrol <= '0';
                                        olm := '1';
                                                olm2 :='1';
                                                say6:=0;
                                                end if;
                                        end if;
                                else
                                        if olm2 = '0' then
                                        say5 := say5+1;
                                        say6 := say5;
```

```
                        end if;
                                bekle:=bekle-1;
                                if bekle=0 then
                                motor_kontrol <= "10101010";   --arabayi ilerlet
                                kolkontrol <= '0';
                                olm := '0';
                                olm1 := '0';
                                olm2 :='0';
                                end if;

                        end if;
                    end if;
                end if;
   end if;
else
  bekle:=1;
  temp:=0;
  temp1:=0;
  temp3:=0;
  temp4:=0;
  say:=0;
  say1:=0;
  say2:=0;
  say5:=0;
  say6:=0;
  a:=0;
  b<=0;
  motor_kontrol <= "11111111";    ---sistem kapalıyken araç hareket etmesin
end if;

led <= conv_std_logic_vector(say6,8);

asayac <= say1;

end process;
--------------------------------------------------------------------------------
-----------robot kolunun hareketlerinin sağlandığı kısım----------------
--------------------------------------------------------------------------------
process (clk_main1,kolkontrol,asayac)

variable t : integer range 0 to 100000;

variable bel1 : integer range 0 to 1000;
variable bel2 : integer range 0 to 1000;      ---bel motorunun parametresi
variable bel3 : integer range 0 to 1000;

variable omuz1 : integer range 0 to 1000;
variable omuz2 : integer range 0 to 1000;
variable omuz3 : integer range 0 to 1000;     ---omuz motorunun parametresi
```

```
variable omuz4 : integer range 0 to 1000;
variable omuz5 : integer range 0 to 1000;

variable dirsek1 : integer range 0 to 1000;
variable dirsek2 : integer range 0 to 1000;   ---dirsek motorunun parametresi
variable dirsek3 : integer range 0 to 1000;

variable bilek1 : integer range 0 to 1000;
variable bilek2 : integer range 0 to 1000;    ---bilek motorunun parametresi
variable bilek3 : integer range 0 to 1000;
variable bilek4 : integer range 0 to 1000;

variable kiskac1 : integer range 0 to 1000;
variable kiskac2 : integer range 0 to 1000;
variable kiskac3 : integer range 0 to 1000;
begin

t:=20000;
if kolkontrol='1' then

--------------------------------------------------------------------------------
------------------cimlerin boyunu ayırt etme bileşeni-----------------
--------------------------------------------------------------------------------
        if 20 <= asayac then            --büyük cisim
                bel1 := 18;
                bel2 := 1;
                bel3 := 26;

                omuz1 := 18;
                omuz2 :=10;
                omuz3 :=18;
                omuz4 :=13;
                omuz5 :=18;

                dirsek1 :=18;
                dirsek2 :=16;
                dirsek3 :=26;

                bilek1 :=2;
                bilek2 :=16;
                bilek3 :=2;
                bilek4 :=30;

                kiskac1 :=17;
                kiskac2 :=8;
                kiskac3 :=17;

        elsif 15 < asayac and asayac < 20 then    --orta cisim
                bel1 := 18;
```

```
            bel2 := 1;
            bel3 := 18;

            omuz1 := 18;
            omuz2 :=10;
            omuz3 :=18;
            omuz4 :=16;
            omuz5 :=18;

            dirsek1 :=18;
            dirsek2 :=16;
            dirsek3 :=16;

            bilek1 :=2;
            bilek2 :=16;
            bilek3 :=2;
            bilek4 :=24;

            kiskac1 :=17;
            kiskac2 :=8;
            kiskac3 :=17;
        elsif asayac <= 15 then         --küçük cisim
            bel1 := 18;
            bel2 := 1;
            bel3 := 32;

            omuz1 := 18;
            omuz2 :=10;
            omuz3 :=18;
            omuz4 :=14;
            omuz5 :=18;

            dirsek1 :=18;
            dirsek2 :=16;
            dirsek3 :=18;

            bilek1 :=2;
            bilek2 :=16;
            bilek3 :=2;
            bilek4 :=24;

            kiskac1 :=17;
            kiskac2 :=8;
            kiskac3 :=17;
        end if;
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

  if clk_main1'event and clk_main1 = '1' then
```

```
d_sayac1 <= d_sayac1 + 1;
d_sayac2 <= d_sayac2 + 1;
d_sayac3 <= d_sayac3 + 1;
d_sayac4 <= d_sayac4 + 1;
d_sayac5 <= d_sayac5 + 1;

sayac1 <= sayac1 + 1;
sayac2 <= sayac2 + 1;
sayac3 <= sayac3 + 1;
sayac4 <= sayac4 + 1;
sayac5 <= sayac5 + 1;

----------taban motoru----------
        if d_sayac1 <= t  then

                if sayac1 < bel1 then
                        pwm1 <= '1';
                else
                        pwm1 <= '0';
                end if;
                if 254 < sayac1 then
                        sayac1 <= 0;
                end if;
        elsif d_sayac1 <= 9*t then
                if sayac1 < bel2 then
                        pwm1 <= '1';
                else
                        pwm1 <= '0';
                end if;
                if 254 < sayac1 then
                        sayac1 <= 0;
                end if;
        else
                if sayac1 < bel3 then
                        pwm1 <= '1';
                else
                        pwm1 <= '0';
                end if;
                if 254 < sayac1 then
                        sayac1 <= 0;
                end if;
        end if;
----------omuz        motoru----------
        if d_sayac2 <= 2*t  then
                if sayac2 < omuz1 then
                        pwm2 <= '1';
                        pwm22 <= '1';
                else
                        pwm2 <= '0';
```

```
                pwm22 <= '0';
        end if;
        if 254 < sayac2 then
                sayac2 <= 0;
        end if;
elsif d_sayac2 <= 8*t  then
        if sayac2 < omuz2 then
                pwm2 <= '1';
                pwm22 <= '1';
        else
                pwm2 <= '0';
                pwm22 <= '0';
        end if;
        if 254 < sayac2 then
                sayac2 <= 0;
        end if;
elsif d_sayac2 <= 10*t then
        if sayac2 < omuz3 then
                pwm2 <= '1';
                pwm22 <= '1';
        else
                pwm2 <= '0';
                pwm22 <= '0';
        end if;
        if 254 < sayac2 then
                sayac2 <= 0;
        end if;

elsif d_sayac2 <= 13*t then
        if sayac2 < omuz4 then
                pwm2 <= '1';
                pwm22 <= '1';
        else
                pwm2 <= '0';
                pwm22 <= '0';
        end if;
        if 254 < sayac2 then
                sayac2 <= 0;
        end if;
else
        if sayac2 < omuz5 then
                pwm2 <= '1';
                pwm22 <= '1';
        else
                pwm2 <= '0';
                pwm22 <= '0';
        end if;
        if 254 < sayac2 then
                sayac2 <= 0;
```

```
                end if;
            end if;
-------------------------dirsek motoru----------------------
            if d_sayac3 <= 3*t  then
                if sayac3 < dirsek1 then
                    pwm3 <= '1';
                else
                    pwm3 <= '0';
                end if;
                if 254 < sayac3 then
                    sayac3 <= 0;
                end if;
            elsif d_sayac3 <= 7*t  then
                if sayac3 < dirsek2 then
                    pwm3 <= '1';
                else
                    pwm3 <= '0';
                end if;
                if 254 < sayac3 then
                    sayac3 <= 0;
                end if;
            else
                if sayac3 < dirsek3 then
                    pwm3 <= '1';
                else
                    pwm3 <= '0';
                end if;
                if 254 < sayac3 then
                    sayac3 <= 0;
                end if;
            end if;
-------------------------bilek motoru-----------------------
            if d_sayac4 <= 4*t  then
                if sayac4 < bilek1 then
                    pwm4 <= '1';
                else
                    pwm4 <= '0';
                end if;
                if 254 < sayac4 then
                    sayac4 <= 0;
                end if;
            elsif d_sayac4 <= 6*t  then
                if sayac4 < bilek2 then
                    pwm4 <= '1';
                else
                    pwm4 <= '0';
                end if;
                if 254 < sayac4 then
                    sayac4 <= 0;
```

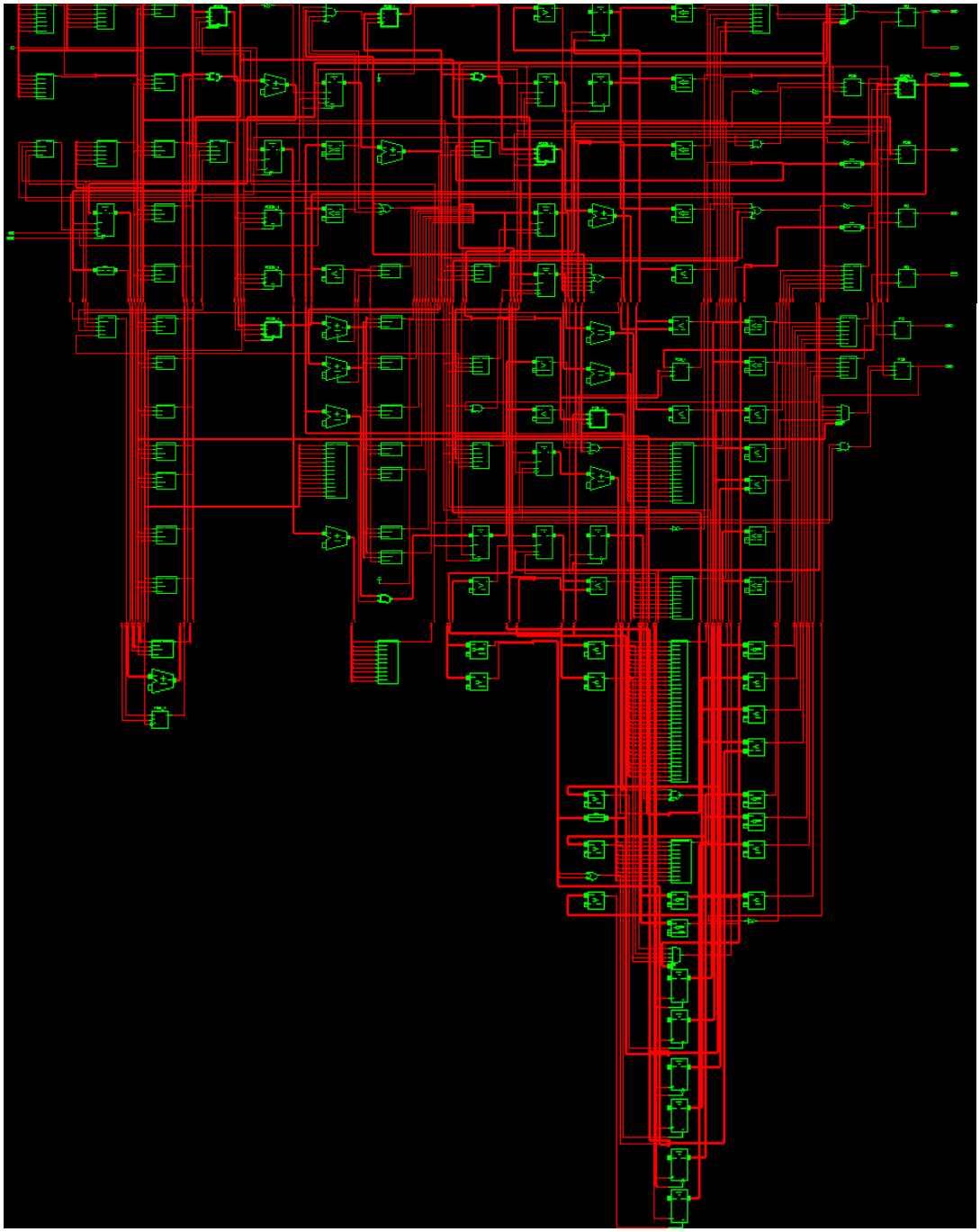
```
                end if;
            elsif d_sayac4 <= 11*t  then
                if sayac4 < bilek3 then
                    pwm4 <= '1';
                else
                    pwm4 <= '0';
                end if;
                if 254 < sayac4 then
                    sayac4 <= 0;
                end if;
            else
                if sayac4 < bilek4 then
                    pwm4 <= '1';
                else
                    pwm4 <= '0';
                end if;
                if 254 < sayac4 then
                    sayac4 <= 0;
                end if;
            end if;
----------kıskaç motoru----------
            if d_sayac5 <= 5*t  then
                if sayac5 < kiskac1 then
                    pwm5 <= '1';
                else
                    pwm5 <= '0';
                end if;
                if 254 < sayac5 then
                    sayac5 <= 0;
                end if;
            elsif d_sayac5 <= 12*t  then
                if sayac5 < kiskac2 then
                    pwm5 <= '1';
                else
                    pwm5 <= '0';
                end if;
                if 254 < sayac5 then
                    sayac5 <= 0;
                end if;
            else
                if sayac5 < kiskac3 then
                    pwm5 <= '1';
                else
                    pwm5 <= '0';
                end if;
                if 254 < sayac5 then
                    sayac5 <= 0;
                end if;
            end if;
```

```
        end if;
else
        if clk_main1'event and clk_main1 = '1' then

                d_sayac1 <= 0;       ---sayaçlar sıfırlanır.
                d_sayac2 <= 0;
    d_sayac3 <= 0;
                d_sayac4 <= 0;
                d_sayac5 <= 0;

                sayac2 <= sayac2 + 1;
                sayac3 <= sayac3 + 1;
                sayac4 <= sayac4 + 1;
                sayac5 <= sayac5 + 1;

                        if sayac2 < 18 then  --omuz
                                pwm2 <= '1';
                                pwm22 <= '1';
                        else
                                pwm2 <= '0';
                                pwm22 <= '0';
                        end if;
                        if 254 < sayac2 then
                                sayac2 <= 0;
                        end if;

                        if sayac3 < 18 then ---dirsek
                                pwm3 <= '1';
                        else
                                pwm3 <= '0';
                        end if;
                        if 254 < sayac3 then
                                sayac3 <= 0;
                        end if;

                        if sayac4 < 2 then  --bilek
                                pwm4 <= '1';
                        else
                                pwm4 <= '0';
                        end if;
                        if 254 < sayac4 then
                                sayac4 <= 0;
                        end if;
                                if sayac5 < 17 then  --kıskaç
                                pwm5 <= '1';
                        else
                                pwm5 <= '0';
                        end if;
                        if 254 < sayac5 then
```

```
                    sayac5 <= 0;
                end if;
        end if;
end if;
end process;
end Behavioral;
```

EK-2

Mobil robot kolu modülünün RTL şematik gösterimi aşağıda verilmiştir:



KİŞİSEL YAYIN ve ESERLER

[1] **Karcı H**., Tangel, A. FPGA tabanlı 5-eksenli mobil robot kolu tasarımı ve uygulaması, *Elektrik-Elektronik ve Bilgisayar Mühendisliği Sempozyumu*, Bursa, Türkiye, 29 Kasım-1Aralık 2012.

ÖZGEÇMİŞ

1987 yılında Ankara'nın Nallıhan ilçesinde doğdu. İlk, orta ve lise eğitimini Nallıhan'da tamamladı. 2005 yılında girdiği Kocaeli Üniversitesi Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği Bölümü'nden 2009 yılında Elektronik ve Haberleşme Mühendisi olarak mezun oldu. 2009 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'na Yüksek Lisans öğrencisi olarak girdi. Şu anda yüksek lisans eğitimini bitirme aşamasındadır.