FPGA TABANLI NESNE ALGILAMA

Alper Nabi AKPOLAT

Yüksek Lisans Tezi

Mekatronik Mühendisliği Anabilim Dalı
Danışman: Yrd. Doç. Dr. Cafer BAL

MAYIS-2015

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI NESNE ALGILAMA

YÜKSEK LİSANS TEZİ

Alper Nabi AKPOLAT

Yüksek Lisans Tezi

Mekatronik Mühendisliği Anabilim Dalı
Programı: Elektronik Sistemler

Danışman: Yrd. Doç. Dr. Cafer BAL

MAYIS-2015

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI NESNE ALGILAMA

YÜKSEK LİSANS TEZİ

ALPER NABİ AKPOLAT

Tezin Enstitüye Verildiği Tarih : 04 Mayıs 2015

Tezin Savunulduğu Tarih : 20 Mayıs 2015

Tez Danışmanı : Yrd. Doç. Dr. Cafer BAL (F.Ü)

Diğer Jüri Üyeleri : Doç.Dr. Mehmet GEDİKPINAR (F.Ü)
Yrd. Doç. Dr. Mehmet ÜSTÜNDAĞ (B.Ü)

MAYIS-2015

ÖNSÖZ

Tez çalışmamın her aşamasında literatür ve teknik çalışmalarımda desteklerini, bilgilerini ve hoşgörülerini esirgemeyen değerli Mekatronik Mühendisliği öğretim üyelerine, çalışanlarına, özellikle de danışman hocam Sayın Yrd. Doç. Dr. Cafer BAL'a ve her zaman yanımda olan aileme teşekkürlerimi bir borç bilirim.

Alper Nabi AKPOLAT
ELAZIĞ - 2015

İÇİNDEKİLER

Sayfa No

ÖZET

Örüntü, bir nesne veya olay kümesindeki elemanların art arda ve düzenli bir biçimde birbirlerini takip ederek yenilenmesidir. Nesne algılama ise bu nesne veya olayı, belirlenmiş kategorilere atamadır. Nesneler farklı yöntemler ile tanınabilir. Günümüzde gelişen sayısal yöntemler ve elektronik devre teknolojileri sayesinde bu sorunlara farklı çözümler üretilebilmektedir. Sayısal tasarım gelişiminin son noktası FPGA yongasıdır.

FPGA'ler programlanabilir komponentler (Lojik Bloklar) ve bağlantılar içeren yarı iletken cihazlardır. Lojik bloklar basit lojik kapıların (AND, XOR vs…) işlemlerini yerine getirmek için programlanabildikleri gibi, daha karmaşık fonksiyonların da (Multiplexer-Decoder-İşlemci vs…) işlemlerini yerine getirebilmektedirler.

Bu tez çalışmasında donanım tasarlamak için FPGA cihazı olarak Xilinx Spartan 3E-100 serisi ve programlama dili olarak da VHDL (Very High Speed Hardware Description Language) kullanılmıştır.

Yazılımsal olarak oluşturulan donanımları test etmek ve FPGA'e yüklemeden önce doğruluğunu denemek için Xilinx üreticisinin yazılımı olan ISE Design Suite ara yüzü kullanılmıştır. Bu yazılımın nasıl kullanılacağı ve yazılan programların cihaza gömme işlemi adım adım gerçekleştirilmiştir.

FPGA tabanlı bir nesne algılama algoritması olarak, FPGA'e bağlı bir ultrasonik sensör ile sabit bir fiziksel nesnenin algılanması ve algılanan o fiziksel nesnenin mesafesinin hesaplanması gerçekleştirilmiştir. Hesaplanan bilginin görüntülenmesi, FPGA'deki yedi parçalı göstergede sağlanmıştır.

Anahtar kelimeler: Nesne Algılama, FPGA, Ultrasonik Sensör, Yedi Parçalı Gösterge, VHDL, ISE Design Suite, Spartan 3E-100

SUMMARY

FPGA Based Object Detecting

Pattern is renewal by following each other elements consecutively and regularly in an object or set of event. Object detecting is designation this object or event for specified categories. Objects can be detected with different methods. Nowadays, different solutions can be produced by means of developing digital methods and electronical circuit technologies. The end point of development of digital design is FPGA chip.

FPGAs are programmable semiconductor devices of which consist programmable components (logic blocks) and connections. Logic blocks can be programmed to perform implementation of basic logic gates (AND, XOR, etc...), also it can be programmed in order to fulfil more complex functions of transactions (Multiplex-Decoder-Processor, etc...).

In this thesis study, Xilinx Spartan 3E-100 serial and VHDL (Very High Speed Hardware Description Language) have been used as FPGA device and software language.

The software of Xilinx's manufacturer is ISE Design Suite interface which has been used for testing hardware created by the interface and for trying the accuracy of the software prior to installation to FPGA. How to use the software and embedding process of the written program has been executed step by step.

As FPGA based an object detecting algorithm, detecting a constant physical object and calculating distance of this physical object with ultrasonic sensor which is connected to FPGA has been achieved. Visualization of calculating data has been provided with seven segment display on FPGA.

Key Words: Object Detecting, FPGA, Ultrasonic Sensor, Seven Segment Display, VHDL, ISE Design Suite, Spartan 3E-100

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

SEMBOLLER LİSTESİ

ASIC : Aplication Spesific Integrated Circuit

ASSP : Application Specific Standart Product

FPGA : Field Programmable Gate Array

MPGA : Mask Programmable Gate Array

NAND : Not And

XOR : Exclusive Or

MUX : Multiplexer

IC : Integrated Circuit

LUT : Lookup Table

LSI : Large Scale Integration

MSI : Medium Scale Integration

SSI : Small Scale Integration

VLSI : Very Large Scale Integration

PLD : Programmable Logic Devices

SPLD : Simple Programmable Logic Devices

CPLD : Complex Programmable Logic Devices

PAL : Programmable Array Logic

PLA : Programmable Logic Array

PROM : Programmable Read Only Memory

JTAG : Joint Test Action Group

HDL : Hardware Description Language

VHSIC : Very High Speed Integrated Circuit

VHDL : Very High Speed Integrated Circuit Hardware Description Language

SVGA : Super Video Graphics Array

FF : Flip-Flop

1. GİRİŞ

FPGA, Alan Programlanabilir Kapı Dizisi anlamına gelen **F**ield **P**rogrammable **G**ate **A**rray'in kısaltılmış halidir. Alan programlanabilir kapı dizileri istenen muameleye göre donanım üreten ve daha sonrada tasarımcı tarafından değiştirilebilen entegre yongalardır.

Sayısal devre tasarımı yapan yongaların gelişmesi ile analog devrelerin yapabildiği işleri kitlere yazılan birkaç kod ile gerçekleştirmek hayal olmaktan çıkmıştır.

1970'lerin başında basit programlanabilir mantık devrelerinin (SPLD-Simple Programmable Logic Device) kullanımına başlanmıştır. PLA, PAL ve PROM gibi isimler alan bu devreler sadece bir kez programlanabilen VE/VEYA kapılarından oluşmaktaydı. SPLD'ler ister sahada ister fabrikada programlansın içlerinde barındırdıkları programlanabilir mantık kapıları fiziksel olarak birbirlerine bağlıydı. İçlerindeki mantık kapılarının yerleri sabitti ve programlanmaları için bağlantı noktalarının sigortalar yardımıyla belirlenmesi gerekiyordu. Daha sonra birçok PLA'in (Programmable Logic Array) bir araya getirilmesiyle CPLD'ler (Complex Programmable Logic Device) oluşturuldu. Bu sayede sayıca daha fazla programlanabilir iç bağlantılara sahip bir tek entegre devre meydana getirilmiş oldu.

CPLD'lerden sonra, 1980'lerin sonuna doğru, FPGA'ler kullanılmaya başlanmıştır. Steve Casselman isimli bir mühendis tarafından yapılan bir çalışma ile 600000 adet yeniden programlanabilir mantık devresinden oluşan bir bilgisayar meydana getirilmesi ile FPGA'lerin temelleri atılmıştır.

Günümüzde sayısal sistemlerinin kullanıldığı alanlardaki hızlı gelişmeler, üretim tamamlandıktan sonra da esnek, genel amaçlı olacak şekilde tasarlanan işlemcileri ortaya çıkarmıştır. VLSI (Very Large Scale Integration) teknolojisi, yüksek yoğunluklu sayısal devrelerin yüksek hızlı ve kolay gerçekleştirilmesini sağlamıştır.

Sayısal devrelerin tasarımında işlevleri değiştirilebilen programlanabilir aygıtların kullanımı sayısal sistemlere oldukça üstünlük sağlamıştır. FPGA, geleneksel programlanabilir aygıtlardan daha üstün özelliklere sahiptir.

Bu özelliklerden bazıları yeniden programlanabilir yapısı ve FPGA teknolojisindeki ilerleme, tüm sistem ve alt sistemleri içeren tek aygıt haline getirilebilmesidir.

FPGA'in yapılandırılması için yüksek seviyeli donanım tanımlama dillerinden HDL (Hardware Description Language), VHDL'in (Very High Speed Integrated Circuit

Hardware Description Language) kullanımı yaygınlaşmaktadır. Bu diller hem yüksek soyutlama ile tasarım sunma hem de geleneksel metotlarla karşılaştırıldığında tasarım süresinde önemli ölçüde azaltma sağlama yeteneğine sahiptir.

Bu teknoloji günümüzde tüketici elektroniğinden, uzay ve savunma sanayi alanlarına kadar çok geniş bir yelpazede kullanılmaktadır. VHDL ise bir donanım tanımlama dili olup, FPGA'in donanım yapısının değiştirilmesi için kullanılan dillerden biridir. Dünyada uzun yıllardır kullanılan bu iki teknoloji ülkemizde de büyük bir hızla yaygınlaşmaktadır. Basit kapı seviyesi devre tasarımından, soft işlemci oluşturmaya ve farklı I/O çevre birimlerini kullanmaya kadar geniş bir alanı kapsamaktadır.

Birbirinden ayrılmaz bu iki teknoloji, tasarımcıların kolayca sayısal devre tasarlamalarına ve tasarımlarını prototip bir cihaz üzerinde gerçekleştirerek fiziksel ortamda doğrulayabilmelerine olanak sağlar.

FPGA'in daha iyi kavranması açısından için, içindeki transistörler birbirinden bağımsız ve serbest olarak üretilmiş ham bir entegre olarak değerlendirilebilir. Kullanıcının istediği fonksiyona göre FPGA içindeki bağlantılar birbirine bağlanabilir ve bu sayede istenen tasarımlar gerçekleştirilebilir.

Paralel işlem yapabilme kabiliyeti başta olmak üzere, piyasada programlanabilme, oluşturulan donanımın test edilip doğrulanabilmesi ve istenirse içerisine bir mikroişlemci sistemi dahi gömülebilmesi gibi birçok özelliği, FPGA'i günümüzde popüler kılan yönleridir [1,2].

Bu tez çalışmamızda örüntü tanıma algoritmalarının girişi sayılabilecek nesne algılama çözümü, FPGA tabanlı işlenmiştir.

Örüntü kavramı bir model, nesne, kavram ya da fikir de olabilmektedir. Bazı modeller belli bir varlığı tekrar ederek varlık gösterirler.

İstenen fonksiyon veya örüntüye göre donanım yapısı tasarımcı vasıtasıyla değiştirilen entegre devre olarak tanımlanan FPGA yongası tasarım sırasında büyük esnekliği ve paralel işlem yapabilme yetisi neden ile bu tez çalışmamızda tercih edilmektedir.

Literatürde; istenen işe göre donanım yapısı kullanıcı tarafından değiştirilebilmeye müsait olan birçok tasarım FPGA tabanlı gerçekleştirilebilir. FPGA'in bu tür tasarım süreçlerinde bu denli tercih edilmesi esnek ve paralel çalışması sonucudur.

Örneğin herhangi bir kan grubu tayininde bir görüntü işleme algoritması olan donanım uygulaması için bir görüntü işleme tekniği ile kandaki aglütinin numune içinde

kenarları ve kontrastı tespit ederek kan türünü belirlemek için aglütinasyon görünümünü kullanmaktadır. FPGA ve paralel işleme algoritmaları ile bu sistemi güvenilir hale getirmek ve çok miktardaki kan örneklerinin karakterizasyonu için görüntü işleme tekniği olarak konjugasyon kullanılmaktadır. Donanım uygulamasında güç kaynağından 2,5 V, 770 mW güç tüketildiğini göstermektedir. Bu durum FPGA'in hızı ile birleşince her yönden çekici bir hale gelmektedir [3].

Yüz tanıma sistemleri; gözetim, biyometri ve güvenlik alanları da dahil olmak üzere birçok uygulamada hayati bir rol oynamaktadır. Yüz tanıma sistemleri için bir FPGA kullanılarak alt örnekleme modülü sisteminde bir uçtan bir uca kameradan video girişi alınabilir, çeşitli algoritmalar kullanılarak yüzlerin konumları algılanabilir. Algılanan yüzler belirlenmiş kategorilere atanarak her yüz tanınır ve sonuçları verir. Saniyede birden fazla işlem yapılmasının öngörüldüğü bu sistemlerde, FPGA üreticilerinin yüksek hızlı geliştirme kitleri bu tür sistemlerin tasarımına büyük katkı sağlayacaktır [4].

Diğer örüntü tanıma algoritmalarında olduğu gibi yüz tanıma sistemlerinde de birçok teknik mevcuttur fakat bu tekniğin nasıl yapılacağı çalışmalar boyunca süregelen bir merak konusu olmuştur. Artan dünya nüfusu ile beraber artan enerji tüketimi, sistem tasarımcılarını gelişen teknolojiyi daha verimli kullanarak enerji tasarrufuna yönelik çalışmalar yapmaya yöneltmiştir. SVGA kaynaklı bir yüz tanıma sisteminde analiz ve görüntü ölçekleme faktörünün sistem hesaplama maliyeti ve seçimi arasındaki ilişkiyi deneysel sonuçlarına göre tartışmak için önceki çalışmalara bakmak gerekmektedir. FPGA tabanlı sistem, önceki çalışmalara göre toplam hesaplamaları yarı yarıya azaltır ve görüntü azaltma sürecinde durdurma eşiğini seçmek için yeni bir yöntem vermektedir. Sistemin test sonuçları, gerçek zamanlı yüz tanıma hızının ve yüz algılama oranının yüksek, güç tüketiminin diğer tekniklere göre düşük olduğunu göstermektedir [5].

Ayrıca insan-bilgisayar etkileşimi için gerçek zamanlı insan eli jestini tanıma sistemleri de FPGA tabanlı örüntü tanımaya örnek gösterilebilir. Bu sistemler makul bir doğrulukla hızlı bir oranda 10 farklı el hareketlerini tanıyabilmektedir. Jestler şekil tabanlı özellikler temelinde sınıflandırılır. Daha iyi doğruluk için, dört farklı şekil tabanlı özellik temeli kullanılmaktadır. Cilt rengi segmentasyonu ile yanlış algılama şansını en aza indirmek için yüksek hızlı işlev gören bir donanım yapısına ihtiyaç duyulduğundan dolayı FPGA yongası tercih edilmektedir [6].

Örüntü tanıma disiplinlerinden olan bir nesne izleme sistemi için çözümü yine FPGA sunar. Nesne izleme sistemleri, nesne algılama gerektiren birçok gerçek durumlarda çok

kullanışlıdır. Genel olarak, el yazısı algılama gibi nesne takip sistemi farklı olan algoritmalarla bir sürü hareketli nesneleri algılayabilir. Bu alanda gerçek nesne görüntüsü ve şekli gözlemlenmek için çalışmaların barkod okuma üzerinde yoğunlaştığını göstermektedir. Paralel ve bir o kadar hızlı işlem gerektiren bu işlem ve hesaplamaların en efektif biçimde cevap vermesi açısından FPGA yadsınamaz bir çözümdür [7].

Parmak izi tanıma sistemleri, yüz tanıma sistemleri, optik karakter okuyucular, plaka tanıma sistemleri, bina tespit sistemleri ve DNA çözümlemeleri gibi algoritmaların çözümünden örüntülerden faydalanılmaktadır. Üzerinde ölçülendirme yapılabilen ya da gözlenebilen bilgiler (algılama) örüntü olarak tanımlanabilir. Örüntü; içerisine ses, görüntü, nesne, sinyal gibi değişkenleri kapsayan bir disiplindir. Bu değişkenleri kapsayan fiziksel nesnelerin algılanması, önceden belirlenmiş kategorilere atanmasıdır [8-10].

Günümüzde imalat sonrası kalite kontrol uygulamalarının büyük bir bölümü de örüntü tanıma tekniklerini içermektedir. Enformasyon teknolojilerindeki gelişime paralel olarak fabrika proses hata denetimi, yüz tanıma, parmak izi, bina tespiti, konuşma ve konuşmacı tanım, plaka tanıma, optik karakter tanıma, DNA çözümleme (kimliklendirmesi), imza, retina ve ses gibi kişisel tanımlama sistemlerinin tasarımında nesne algılama tekniklerinden yararlanılmaktadır [11-14].

Bilinen entegrelerin büyük bir kısmı senkronize işlem yapamaz, küçük bir kısmı ise sınırlı olarak gerçekleştirebilirler. FPGA'ler kapasiteye ve uygulamaya bağlı olarak, birbiri ile eş zamanlı yüzlerce işlemi aynı anda yapabilir. Bu durum eş zamanlı işlem gerektiren uygulamaları yürütebilen bu yongalar günümüzde eşsiz ve değerli kılınmaktadır [15-17].

1960' lı yıllarda geliştirilen ve SSI (Small Scale Integrated) denen devreler aynı yonga üzerinden en fazla 100 transistör içeriyordu. 1960' lı yılların sonunda geliştirilen ve MSI (Medium Scale Integrated) devreler ise aynı yonga üzerinde birkaç yüz transistör içeriyordu. Yine TTL ailesinin o dönemde çok yoğun kullanım alanı bulan 7490 BCD sayıcı entegresi de MSI teknolojisi ile geliştirilmişti; lojik devrelerden toplayıcılar (adders), çoklayıcılar (multiplexer) devreleri de MSI idi. Entegrasyondaki gelişme 70' li yılların ortalarında LSI (Large Scale Integrated) devrelerin üretimine olanak sağlamıştır. 80' li yılların başlarında LSI entegreleri takiben üzerinde 100 K transistör içeren bu nedenle de Çok_Büyük_Skala_Entegrasyonu (VLSI) denen entegre devrelerin tasarımı mümkün olmuştur. Bu eğilim yine kronolojik olarak 80' li yılların sonralarında 1 milyon transistöre 90' lı yılların ortalarında 19 milyona transistöre, 2004 yılında 100 milyon transistöre, günümüzde ise 1 milyar transistöre ulaşmış durumdadır. Bu eksponansiyel

gelişme dijital tasarımcılar için de farklı sorunları beraberinde getirmiştir. Bu sorunların minimal seviyeye indirilmesi açısından geliştirilen en popüler yonga olan FPGA, dijital tasarımcıların yardımına yetişmiştir.[18-20]

Uygulamaya Özgü Entegre Devreler-ASIC (Application Specific Integrated Circuit) olarak adlandırılır. Bu devrelere özel bir sistemde kullanılmak üzere tasarlanan tümleşik devrelerdir denilebilir. Genel amaçlı kullanımdan ziyade belirli bir kullanım için özelleştirilmişlerdir. Uygulamaya özgü standart ürünler ASSP' ler (Application Specific Standart Product) ise daha genel donanım bileşenleridir. Çoklu tasarımlarda kullanılmak üzere üretilmişlerdir.

Sayısal mantık entegrelerinden birisi olan PLD'ler (Programmable Logic Devices) yapılandırılabilir bir mimari oluşturabilmek için üretilmiş elektronik bileşenlerdir. PLD'ler elektronik devrelerde kullanılmadan önce mutlaka yapılandırılmaları gerekmektedir.

Programlanabilir salt okunur bellek olan PROM'lar ise kullanıcı tarafından basitçe programlanabilen hafıza sistemleridir. Bir PROM içerisine, bir mikroişlemci programı, basit bir algoritma veya durum makinesi kodu yüklenebilmektedir. [21-23].

PROM'lardaki hız sorununa ve sınırlı sayıdaki giriş/çıkışlarına çözüm olarak PLA'ler geliştirilmiştir. VE kapılarından oluşan programlanabilir kısma PLA'de girişler bağlıdır. Bu kısımda tasarıma ait VE işlemleri yapılır. VE işlemlerinin çıkışları VEYA kapılarından oluşan başka bir programlanabilir yapıya sahiptir. PAL'ler ise yapı olarak PLA'lere benzerler. VE kapılarından oluşan kısımda tasarıma ait gerekli VE işlemleri gerçekleştirilir. VEYA kapılarından oluşan kısım sabittir ve programlanamazlar, bu yönüyle PLA'den ayrılırlar.

PAL ve PLA'ler küçük çaplı tasarımlar için uygulanabilir olup, daha karmaşık devrelerin tasarımları için elverişsizdirler. Bu nedenle daha karmaşık devre tasarımları için bir tek entegre devre içerisinde birbiriyle etkileşimli birden çok PAL' ın işlevi karşılayabilen Complex PLD'ler, diğer bir adıyla CPLD'ler üretilmiştir. Donanım tasarımcıları, hem ASIC'lerin esnekliğine sahip hem de programlanabilir devrelerin kullanışlılığına sahip yeni bir tümleşik devre olan FPGA'lerin gelişiminin temelleri atılmıştır. Böylece yeni bir entege devre olan FPGA sayısal tasarım dünyasına hakim olmuştur.

FPGA'ler kendilerine ayrılan pinler ile istenen fiziksel fonksiyonları gerçekleştirirler. FPGA'e ayrılan pinler başlıca; güç, konfigürasyon, clock ve kullanıcı pinleridir. HDL

dillerinden herhangi birisi ile programlanan FPGA donanımı, pinleri yardımıyla dış dünyaya açılmaktadır [23-25].

Program tasarlanırken hem grafik hem de VHDL/Verilog kodu yazılabilir. Tasarımların grafiksel olarak dizaynı için HDL ile oluşturduğumuz modellerin şematikleri çizilebilmektedir [26,27].

Tez çalışmasındaki algoritmanın oluşturulması açısından VHDL tasarım akışını sırasıyla gerçeklenir. Bu gerçeklenme sırası Kodlama, Simülasyon ve Sentezleme kısımlarıdır. Kodlama kısmında oluşturulacak algoritma için gereken kodlar yazılır. Yazılan VHDL kodlarının simülasyonu yapılarak, programın doğruluğu gözlemlenir. Simülasyon sırasında ortaya çıkan hatalar, program FPGA'e yüklenmeden önce doğrulanmaktadır. Sentezleme kısmında yazılan VHDL kod, donanım diline çevrilip RTL modeli çıkartılır. Sentezlemeden sonra kod derleyici tarafından FPGA'e yüklenmeye hazırlanacak konfigürasyon dosyasına çevrilir. Sentezleme işlemi genel olarak derleyici tarafından yürütülmektedir [28,29].

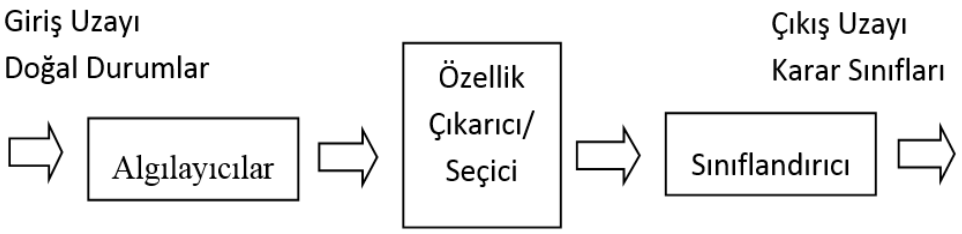
Nesne algılama algoritması ve uygulaması olarak FPGA'e bağlı bir ultrasonik sensör ile belirli durumlardaki fiziksel nesnenin algılanması gerçeklenmiştir. Algılanan fiziksel nesnenin mesafesini saptayan bir sistemin tasarımına yönelik çalışmalar bu tez çalışmasının konusunu oluşturmaktadır.

2. ÖRÜNTÜ

Örüntü kavramı, bir nesne veya olay kümesindeki elemanların art arda ve düzenli bir biçimde birbirlerini takip ederek yenilenmesidir. Schalkoff'a göre örüntü tanıma ve nesne algılama kavramları ölçülen verilerin tanımlanması veya sınıflandırılması ile uğraşan bilim dalıdır.

Örüntü kavramı çok basit bir kavrammış gibi gözükse de doğadaki varlıkların algılanmasına giden yol, örüntü tanıma ve algılama yöntemlerinden geçmektedir. Yüz tanıma sistemleri, parmak izi tanıma sistemleri, optik karakter okuyucular, plaka tanıma sistemleri, bina tespit sistemleri ve DNA çözümlemeleri gibi algoritmaların çözümünde nesne algılamalarının işini kolaylaştıran örüntüler devreye girmektedir.

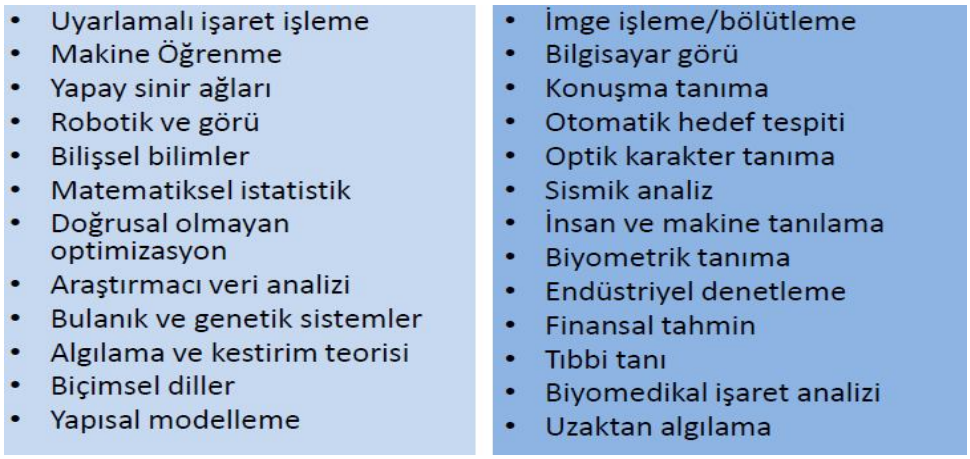
Üzerinde ölçülendirme yapılabilen ya da gözlenebilen bilgiler örüntü olarak tanımlanabilir. Örüntü; içerisine ses, görüntü, nesne, sinyal gibi değişkenleri kapsayan bir disiplindir [8,9].



Şekil 2.1. Nesne Tanıma Blok Diyagramı

2.1. Nesne Algılama ve Örüntü Tanıma Kavramı

Duda & Hart'a göre fiziksel nesne ya da olayı, önceden belirlenmiş kategorilere atama, Fukunaga'ya göre yüksek boyutlu uzayda yoğunluk fonksiyonlarını saptama, uzayı kategori ya da sınıf bölgelerine bölme, Ripley'e göre karmaşık işaretlere verilen belli örnekler ve bunlara ait doğru kararları kullanarak, gelecek yeni örnek dizileri için otomatik karar verme, Schalkoff'a göre ölçümlerin sınıflandırılması ya da tanımlanması ve kategorize edilmesi ile ilgilenen bilim, Schürmann'a göre x gözlemlerine ω ismi verme işlemidir [10].



Şekil 2.2. Örüntü Tanıma ile İlgili Alanlar ve Uygulamalar

Örüntüyü algılayarak, belirlenen kriterler doğrultusunda tanımlamayıp belirli kategorilere göre tasniflemeye nesne algılama ve örüntü tanıma denilmektedir. Örüntü tanıma ve nesne algılama sistemleri ölçülebilen ya da gözlenebilen bilgilerin tanımlanmasında birçok uygulamanın merkezi konumundadır. Örneğin Şekil 2.1.'de sıkça rastlanan bir nesne tanıma blok diyagramı vardır [11]. Algılayıcı, örüntünün fiziksel değerlerini ölçmekte, seçici; elde edilen ölçümlerden elde edilen giriş uzayından veri toplamaktadır. Sınıflandırıcı da, örüntüyü belirli özelliklere göre gruplayarak uygun kategorilere atar [10].

İnsanların günlük yaşamında, belirli bir görüntüyü veya sesi tanımak için kullandıkları kuralları belirtmek mümkün değildir. İnsanların pratikte karşılaştıkları bu örüntü tanıma olaylarını, makine tabanlı örüntü tanıma uygulamalarında belirli kriterlere oturtmak mümkündür. Bunun için, örüntülerin bir birinden bağımsız ve belirli karakteristik niteliklerinin elde edilmesi özellik çıkarım süreci ile gerçekleştirilir. Bu özellikler bir sınıflandırıcıya verilerek her bir örüntü belirli sınıflara ayrıştırılıp tanımlama yapılır. Örneğin günümüzde imalat sonrası kalite kontrol uygulamalarının büyük bir bölümü örüntü tanıma tekniklerini içermektedir. Enformasyon teknolojilerindeki gelişime paralel olarak fabrika proses hata denetimi, yüz tanıma, parmak izi, bina tespiti, konuşma ve konuşmacı tanım, plaka tanıma, optik karakter tanıma, DNA çözümleme (kimliklendirmesi), imza, retina ve ses gibi kişisel tanımlama sistemlerinin tasarımında örüntü tanıma tekniklerinden yararlanılmaktadır. Bu tekniklerden başlıcaları Şekil 2.2.'de gösterilmiştir [12].

Örüntü kavramı, öte yandan üzerinde çalışılan varlıklar ile alakalı ölçülebilir ya da gözlenebilir veriler olduğundan, insanların çeşitli ses, görüntü ve benzeri tüm örüntülerin biçimsel şekillerinden çıkardıkları dilsel şekillendirmedir [13,14].

Mevcut örüntü tanıma sistemleri istatistiksel, yapısal ve akıllı örüntü tanıma sistemleri olarak başlıca üç grupta toplanabilir.

İstatistiksel örüntü tanıma yönteminde, sınıflama algoritmaları istatistiksel analiz üzerine kurulmuştur.

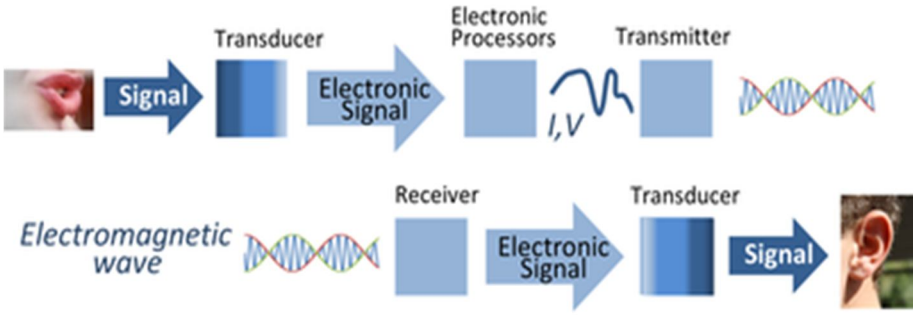
Yapısal (geometriksel, kural dizilim) örüntü tanıma yaklaşımında, verilen bir örüntü, şekilsel yapıdan temel karakteristik tanımlanmaya indirgenir. Yapısal örüntü tanımada, üzerinde çalışılan örüntünün şeklinden özellikleri çıkarılmaktadır. Tasarlanacak sistemde örüntünün alınarak rengin Gray'e çevrilmesi ve çeşitli özellikler kullanılarak nesne tanıma işlemi gerçekleştirilmesi işleminde yapısal örüntü tanıma yöntemi tercih edilmektedir.

Akıllı örüntü tanıma sisteminde ise daha önceden öğrendiklerini tutabilecek bir hafızaya sahip, çıkarım, genelleme ve belirli bir hata toleransı ile karar verebilme yeteneklerini içermekte ise bu sistem akıllı örüntü tanıma sistemi olarak değerlendirilir [8,12].

2.2. Nesne Algılama ve Örüntü Tanıma Uygulamaları

Örüntü tanıma uygulamalarında genel olarak nesne algılama-tanıma ve daha da karmaşık hali ile sinyal işleme disiplinlerine değinildiği görülmektedir. Sinyal işleme, sistem mühendisliği, elektronik mühendisliği ve uygulamalı matematik gibi dalların en önemli konularından birisidir. Genel olarak analog ve dijital sinyaller üzerinde analizler yapma, zamansal ve mekansal değişiklikleri saptayarak çeşitli sistemlere uygulama olarak tanımlanabilir.

Sinyal işleme yapılırken ses sinyalleri, elektromanyetik dalgalar, resim gibi görsel öğeler, sensörlerin algıladığı değerler kullanılabilir ve bu sinyaller kontrol sistemleri, haberleşme sistemleri, radarlar gibi birçok alana uygulanabilir hale getirilebilirler.



Şekil 2.3. İşlenen Sinyalin Geçtiği Süreçler

17. yüzyılda sinyal işlemenin temel prensipleri bulunmuş ve klasik-nümerik sinyal analiz tekniği geliştirilmiş. Oppenheim ve Schafer ise analog sinyaller için geliştirilmiş olan bu tekniği 1940-1950'li yıllarda sayısallaştırma veya dijital olarak iyileştirme adı altında geliştirmişlerdir ve böylece dijital sistemler için de sinyal analiz tekniği mevcut hale gelmiştir.

Sinyal İşlemenin amaçlarını aslında 4 ana kategoriye ayırabiliriz,

1) Sinyal Kazancı ve Yeniden Yapılanma: Sinyalin fiziksel olarak boyutunu ölçmeye, sinyali depolamaya ve daha sonra da sinyali kendi isteğimiz doğrultusunda yeniden yapılandırmaya yarar. Dijital sistemlerde örnekleme ve niceleme olarak geçer.

2) Kalite Arttırma: Gürültü azaltma, görünülebilirlik arttırma, yankı azaltma gibi işlemleri kapsar.

3) Sinyal Sıkıştırma: Ses sıkıştırma, görüntü sıkıştırma, video sıkıştırma gibi işlemlerle depolama alanından kazanç sağlar.

4) Özellik Çıkarma: Görüntü algılama ve ses tanıma özelliklerini kapsar.

Sinyal İşleme Kategorileri olarak;

Analog Sinyal İşleme: Dijital sinyale dönüştürülmemiş sinyaller için kullanılır. Eski radyolarda, televizyonlarda ve radarlarda analog sinyal mevcuttur. Pasif filtreler, aktif filtreler, geciktirici hatlar, integratörler kullanılabilir. İşlenen analog sinyalin geçtiği süreçler Şekil 2.3.'de açıkça mevcuttur.

Ayrık Zamanlı Sinyal İşleme: Analog ayrık zamanlı sinyal işleme teknolojisi zaman ayırıcı devrelerde, analog geciktirici devrelerde ve analog geri besleme devrelerde kullanılır. Dijital sinyal işleme tekniğinin öncüsü olarak bilinir ve günümüzde GHz' lik frekanslardaki sinyaller için kullanımdadır.

Dijital Sinyal İşleme: Genellikle bilgisayarlar aracılığıyla gerçekleştirilen bu teknik, bu işlem için özel geliştirilmiş entegreli devreler aracılığıyla veya dijital sinyal

işlemcileri ile de yapılabiliyor. Çeşitli aritmetik operatörler kullanılan bu teknikte, hızlı fourier dönüşümü, sınırlı sinyal tepki analizi veya sınırsız sinyal tepki analizi gibi matematiksel teknikler de kullanılıyor.

Lineer Olmayan Sinyal İşleme: Lineer olmayan sistemlerden elde edilen sinyaller üzerinde işlem yapılan bu teknikte, zaman, frekans veya geçici bölge gibi parametreler ile analiz yapılabiliyor. Ancak lineer olmayan sinyaller kullanıldığı için en karmaşık sinyal işleme tekniği olarak bilinir. Özellikle harmonikler için çeşitli sinyal filtreleri kullanılır.

Sinyal işleme alanları ve çeşitleri olarak ise;

♦ Statik Sinyal İşleme: Sinyalleri analiz etme ve bilgi edinme

♦ Spektral Kestirim: Sinyal frekansının üzerinden güç tespiti ve spektrum bilgisi edinme

♦ Ses İşleme ♦ Konuşma İşleme ♦ Görüntü İşleme ♦ Video İşleme

♦ Dizisel Sinyal İşleme ♦ Zaman-Frekans Analiziyle Sinyal İşleme

♦ Sinyal Filtreleme ♦ Sismik Sinyalleri İşleme

Bu çalışmada; fiziksel nesnenin algılaması diğer nesne algılanması yöntemlerinden farklı olarak FPGA tabanlı olarak incelenmiştir. Örüntü tanıma yönteminin FPGA' da analizi ve çözümlenmesi, FPGA tabanlı bir yazılım algoritması ile FPGA donanımının bileşiminden meydana gelen sistem ile fiziksel nesnenin uzaklığı saptanmaya çalışılmıştır [9,12,15].

3. ALAN PROGRAMLANABİLİR KAPI DİZİLERİ
(FIELD PROGRAMMABLE GATE ARRAY – FPGA)

FPGA tasarımcının istediği fonksiyon yapısına karşılık donanım üreten tümleşik devrelerdir. FPGA'i diğer tümleşik devrelerden ayıran en temel belirginlik iç konfigürasyonunun istenildiği gibi değiştirilmesidir. Programlanamayan sıradan entegrelerde, elementler arasındaki bağlantılar sabittir ve olağanüstü bir durum meydana gelmediği takdirce değişmezler.

FPGA'i, içindeki komponentleri birbirinden ayrı ve serbest olarak yapılmış bir entegre bloğu olarak ele alabiliriz. Tasarımcının isteğine göre FPGA içerisindeki bağlantılar birbirlerine bağlanır ve istenen işlev gerçekleştirilir. Teorik olarak aklımıza gelen herhangi bir entegrenin yaptığı fonksiyonları ve işleri FPGA yapabilir.

FPGA'lerin en önemli özellikleri arasında paralel ve hızlı işlem yapabilme kabiliyeti gelir. Aynı anda birden fazla işlemi yapmaya paralel işlem yapabilmek denilebilir.

Bilinen entegrelerin tamamı eş zamanlı işlem yapamazlar, bazıları sınırlı olarak gerçekleştirebilirler. FPGA ise kapasiteye ve uygulamaya bağlı olarak, birbiri ile eş zamanlı yüzlerce işlemi aynı anda yapabilir. Bu durum eş zamanlı işlem gerektiren uygulamalarda bu yongaları eşsiz ve değerli kılmaktadır.

Eş zamanlı işleme bir örnek olarak; Video Sinyali İşleme-Filtreleme işlemi için, gerçek zamanlı yüksek çözünürlüklü bir video görüntüsü üzerinde filtreleme işlemi yapmak istenirse, öncelikle videonun özelliklerini irdelemek gerekmektedir. Video, esaasında peş peşe sıralanan resimlerden oluşur ve bu resimlerin her birine "frame" denilmektedir. En basit haliyle video filtrelemesi için videonun bir resim karesini giriş portlarından almak, onu filtrelemek ve çıkış portlarından göndermek gerekmektedir. Sonra ikinci resim için de aynı işlemleri gerçek zamanlı olarak tekrarlamak gerekecektir. Standart entegreler (örneğin bir mikroişlemci) ile bu üç işlemi (alma, filtreleme, gönderme) sırasıyla yapıp bitirdikten sonra gelen ikinci resmi almak gerekecektir. Eğer bu işlemler yeterince hızlı yapılamaz ise sıradaki resmi kaçırılabilir. FPGA'de ise bu işlemler paralel olarak devam eder. İlk resmi alıp filtreleme işlemini yaparken ikinci resim alınmaya başlanır. İlk resmi gönderilirken, ikinci resmi filtrelenmeye ve üçüncü resim alınmaya başlanır. Diğer taraftan, filtreleme yapılabilmesi için yoğun olarak çarpma işlemleri gerekmektedir.

Standart bir işlemci ile bu çarpma işlemlerini de sırayla yapmak zorundadır fakat FPGA bu işlemleri de paralel olarak hızlı bir şekilde yapabilmektedir. [16].

FPGA'ler paralel işlem kabiliyetindedir. İç yapısı, arzulanan fonksiyon ve uygulamaya göre değiştirebilen entegre blokları halindedir.

FPGA'de ise donanım yapısı asla sabit değildir ve programlayıcı tarafından tanımlanır. Yazılacak HDL koduna göre istenen işlemler paralel olarak gerçekleşecektir.

FPGA ve bir mikroişlemci arasındaki bariz bir fark mevcuttur. FPGA'i istediğimiz şekli verebildiğimiz bir oyun hamuruna; mikroişlemcileri ise oyuncak bebeğe benzetilebilir. Oyun hamurundan bebek ve her oyuncak yapabilir. Oyuncak bebeği ise olduğu gibi kabul etmek ve kabul edilen fonksiyonda oynamak gerekmektedir. Oyun hamuru ile yapılan oyuncak bebek gerçek oyuncak bebeğin yerini tutmasa da aynı işlevi ve görevi görecektir. [1,17].

3.1. FPGA'in Gelişim Süreci

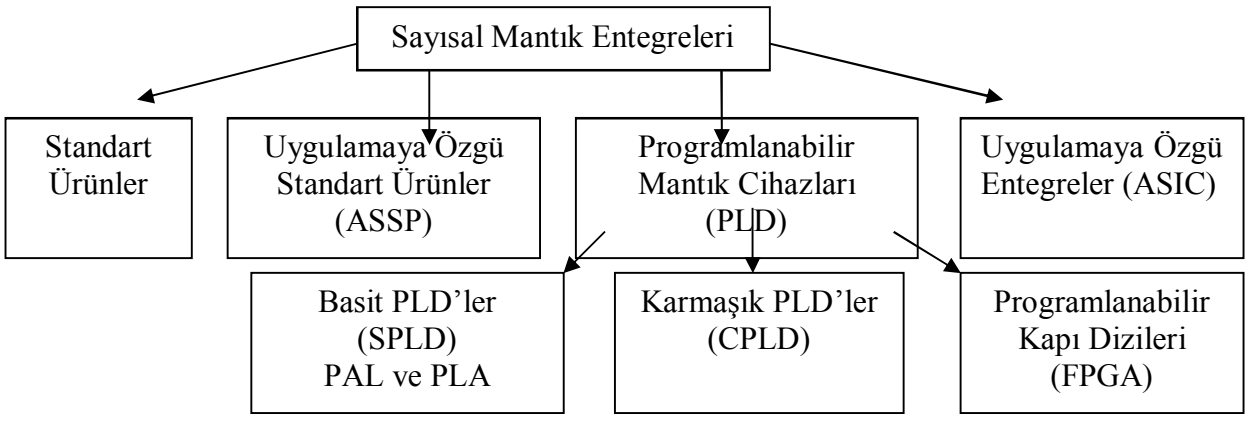
1957' de Jack Kilby' nin Amerikan Ordusu için tasarladığı ilk germanyum entegre devre ile başlayan "aynı yonga üzerine daha fazla transistör entegrasyonu" günümüzde inanılmaz boyutlara ulaşmıştır. Bu gelişme Dijital Elektronik mühendisliğindeki tasarım öğelerini ve metotlarını da değişik bir boyuta taşıdı. Öncelikle entegre devre üretimindeki teknolojik gelişmeyi kronolojik olarak özeti aşağıdadır.

60' lı yılların başında geliştirilen ve Küçük_Skala_Entegreasyon (SSI) denen devreler aynı yonga üzerinden en fazla 100 transistör içermekteydi. 7400 kodlu TTL ailesinin VE_DEĞİL (NAND) kapısı; en yoğun kullanılan SSI entegre devresidir. 60' lı yılların sonunda geliştirilen ve Orta_Skala_Entegrasyon (MSI) devreler ise aynı yonga üzerinde birkaç yüz transistör içermekteydi. Yine TTL ailesinin o dönemde çok yoğun kullanım alanı bulan 7490 BCD sayıcı entegresi de MSI teknolojisi ile geliştirilmişti; lojik devrelerden toplayıcılar (adders), çoklayıcılar (multiplexer) devreleri de MSI idi. Entegrasyondaki gelişme 70' li yılların ortalarında Büyük_Skala_Entegrasyonu (LSI) devrelerin üretimine olanak sağlamıştır. Aynı yonga üzerinde binlerce transistör içeren bu devrelere örnek olarak ilk mikroişlemciler Intel 8080 - 8085 ve Motorola 6800'ı örnek verilebilir. 80' li yılların başlarında LSI entegreleri takiben üzerinde 100 K transistör içeren bu nedenle de Çok_Büyük_Skala_Entegrasyonu (VLSI) denen entegre devrelerin

tasarımı mümkün olmuştur. Bu eğilim yine kronolojik olarak 80' li yılların sonralarında 1 milyon transistöre 90' lı yılların ortalarında 19 milyona transistöre, 2004 yılında 100 milyon transistöre, günümüzde ise 1 milyar transistöre ulaşmış durumdadır. Bu eksponansiyel gelişme dijital tasarımcılar için de farklı sorunları beraberinde getirmiştir [18].

Dijital elektronikten bilindiği üzere herhangi bir dijital devre sadece AND, OR ve NOT kapısı ile gerçekleştirilebilir. Hatta tüm lojik devreleri sadece NAND (veya NOR) kapıları ile gerçekleştirebiliriz. Bu özellik 80' li yıllarda entegre devre tasarımcılarının aynı yonga üzerinde binlerce kapı geliştirmesi ile değişik bir mimariye yol açtı. **P**rogramlanabilir **L**ojik **D**evreler (PLD) adı verilen bu entegreler aynı yonga üzerinde birbirine bağlı olmayan binlerce VE, VEYA ve DEĞİL kapısı içeriyordu. Dijital tasarımcı imalatçının öngördüğü metodu kullanarak bu kapıları istediği işlevi üretmek üzere birbirine bağlıyordu. Bağlantı genellikle bilgisayar programları yardımıyla sigorta bağlantısı kesilerek yapılıyordu. Bu teknolojideki gelişme ile birden fazla PLD'nin aynı yonga üzerine konduğu "**C**omplex **P**rogrammable **L**ogic **D**evices" yani CPLD'lerin üretimine yol açtı [1].

CPLD'lerde, ihtiyaca göre mantık bloğu sayısı artarsa, mantık bloklarının diziliş şekli ve tek bir ara bağlantının kullanılmasından dolayı, hücreler arasındaki bağlantı sayısı da artacaktır. Bu durumda çok büyük tasarımlarda bağlantı sorunlarına sebebiyet verdiğinden dolayı CPLD'leri kullanışsız hale sokarak FPGA'lerin üretilmesine temel teşkil etmiştir [14].



Şekil 3.1. Sayısal Mantık Entegrelerinin Gelişimi

Uygulamaya Özgü Entegre Devreler ASIC (Application Specific Integrated Circuit) olarak adlandırılır. Özel bir sistemde kullanılmak üzere tasarlanan tümleşik devrelerdir. Genel amaçlı kullanım aksine belirli bir kullanım için özelleştirilmiş devrelerdir.

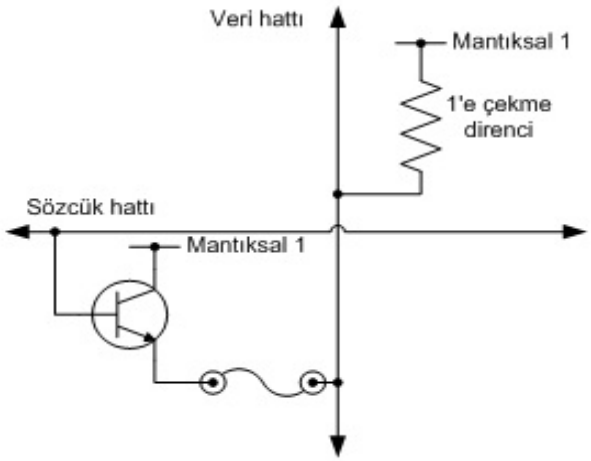
Uygulamaya özgü standart ürünler ASSP'ler (Application Specific Standart Product) ise daha genel donanım bileşenleri olmakla beraber, çoklu tasarımlarda kullanılmak üzere üretilmişlerdir [2,16, 20].

3.1.1. Programlanabilir Mantık Cihazları (PLD)

PLD'ler yapılandırılabilir bir yapı oluşturabilmek için üretilmiş elektronik bileşenlerdir. PLD'ler sabit bir işleve sahip mantık kapılarının aksine üretim esnasında tanımlı bir işleve sahip değillerdir. Bu yüzden elektronik devrelerde kullanılmadan önce mutlaka yapılandırılmaları gerekmektedir [21].

3.1.2. Programlanabilir Salt Okunur Bellekler (PROM)

PROM' lar kullanıcı tarafından basitçe programlanabilen belleklerdir. Şekil 3.2.'de gösterilen bir PROM hücresine, bir mikroişlemci programı, basit bir algoritma veya durum makinesi kodu yüklenebilmektedir. PROM' lar yavaştırlar, bu nedenle hız gerektiren tasarımlarda asla kullanışlı değillerdir [21,22].



Şekil 3.2. PROM Hücresinin Gösterimi

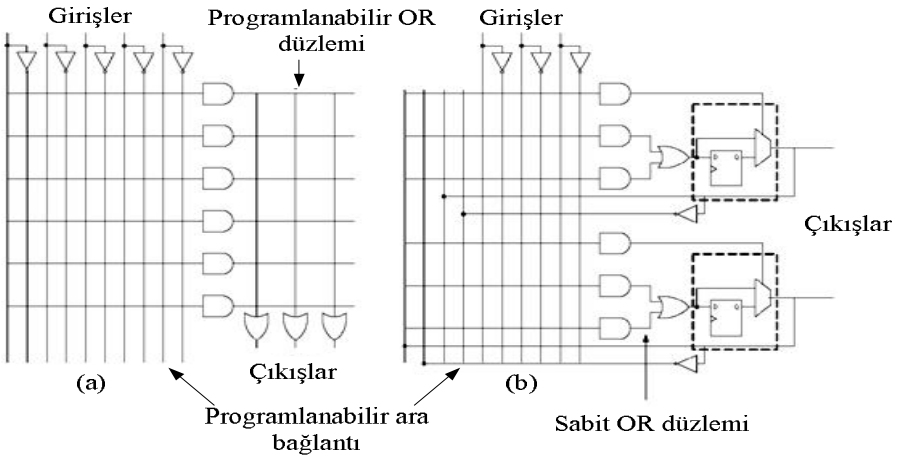
Bazı PROM'lar sadece bir defa programlanabilirken, EPROM veya EEPROM'ları ise silip tekrar programlamak mümkündür. PROM'lar sınırlı sayıda giriş/çıkışı bulunan bir kombinazonal devrelerin gerçeklenmesine elverişli cihazlardır. Ardışıl devreler yapmak için saat ile çalışan flip-flop'u veya mikroişlemcileri ayrıca eklemek gerekmektedir [16,22].

3.1.3. Programlanabilir Lojik Diziler / Dizi Lojiği (PLA / PAL)

PROM'lardaki hız sorununa ve sınırlı sayıdaki giriş/çıkışlarına çözüm olarak PLA'ler geliştirilmiştir. Bu cihazlar genel olarak PROM'lara göre fazla sayıda girişi destekleyebilir ve daha hızlı çalışabilirler.

VE kapılarından oluşan programlanabilir kısma PLA'de girişler bağlıdır. Bu kısımda tasarıma ait VE işlemleri yapılır. VE işlemlerinin çıkışları VEYA kapılarından oluşan başka bir programlanabilir yapıya sahiptir. Bu kısımda tasarıma ait VEYA işlemleri yapılır ve çıkışlara aktarılır. PLA'lerin iki adet programlanabilir alana sahip olması, fazladan kullanılan her sigorta bağlantısı daha büyük gecikmelere neden olduğu gibi, devrenin karmaşıklığını artırır.

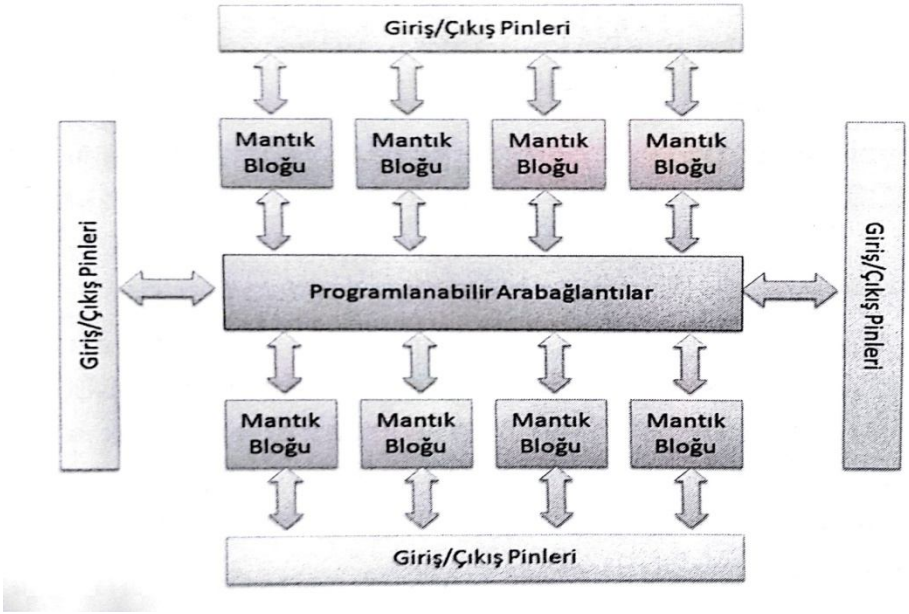
Şekil 3.3. (a)'daki PAL'ler ise yapı olarak Şekil 3.3. (b)'deki PLA'lere benzerler. VE kapılarından oluşan kısımda tasarıma ait gerekli VE işlemleri gerçekleştirilir. VEYA kapılarından oluşan kısım sabittir ve programlanamaz. Bu yönüyle PLA'lerden birbirlerinden farklıdır. Giriş veya çıkışlara çoklayıcı, latch, XOR gibi diğer basit mantık devreleri ve flip-flop gibi saatli yapılar bağlanabilmektedir [16,22].



Şekil 3.3. (a) PLA ve (b) PAL Yapıları

3.1.4. Karmaşık PLD'ler (CPLD)

PAL ve PLA'ler küçük çaplı tasarımlar için uygulanabilir olup, daha karmaşık devrelerin tasarımları için elverişsizdirler. Bu yüzden daha karmaşık devre tasarımları için bir tek entegre devre içerisinde birbiriyle etkileşimli birden çok PAL' ın işlevi karşılayabilen Complex PLD yani CPLD'ler üretilmiştir.



Şekil 3.4. CPLD İç Yapısının Gösterimi

CPLD'lerde, ihtiyaca göre mantık bloğu sayısı artarsa, mantık bloklarının diziliş şekli ve tek bir ara bağlantının kullanılmasından dolayı, hücreler arasındaki bağlantı sayısı da artacağından dolayı, büyük tasarımlarda bağlantı sorunlarına neden olmaktadır. Buna istinaden CPLD'leri kullanışsız hale gelerek FPGA'lerin üretilmesine temel oluşturulmuştur.

FPGA' da lojik blokları CPLD'lerden farklı olarak dizi şeklinde konumlanmıştır. Dizi şeklinde konumlamış bu lojik blokların aralarındaki bağlar yatay ve dikey programlanabilir ara global bağlantılar vasıtasıyla sağlanır. Şekil 3.4.'te de gösterilen bu yapı vasıtasıyla bloklar arasında ihtiyaç duyulan bağlantı sayısı, lojik blok sayısındaki artış ile doğru orantılı olarak artacaktır ki bu durum istenmeyecektir [18].

ASIC'ler ise daha büyük ve karmaşık tasarımları desteklemelerine rağmen, zaman ve maliyet açısından tasarımcı için zahmetli bir tercih olacaktır. Hatta bazı ASIC

tasarımları birkaç yıl kadar sürebilmektedir. Bu entegre devreler sadece özel tasarımlar için üretilmekte ve başka bir tasarım için kullanılamamaktadır. Donanım tasarımcılarının çalışmalarında, hem ASIC'lerin esnekliğine ve karmaşıklığına sahip hem de programlanabilir devrelerin kullanışlılığına münhasır yeni bir entegre devre olan FPGA'lerin gelişiminin temelleri atılmıştır [20].

Böylece CPLD ve ASIC ikisinin yerlerini tek başına doldurabilecek yeni bir entege devre olan FPGA sayısal tasarım dünyasına hakim olmuştur [23].

3.2. FPGA'in Önemi

FPGA'ler 500 Mhz seviyesinde yüksek hızları, çoğu tasarım türlerine hitap eden özellikleri ve bu kadar meziyete göre düşük maliyetleri ile dikkat çekmektedir. Bu yüksek seviyeli özelliklerin, teknolojinin gelişimine paralel olarak ilerleme kaydettiği yadsınamaz bir gerçektir.

Programlanabilir olması özelliği sayesinde FPGA'ler teknoloji dünyasında birçok alanda tercih edilmektedir. Gelişen FPGA teknolojisi ile FPGA kitleri üzerinde bulunabilen gömülü işlemciler, sayısal sinyal işleme (DSP) blokları, PLL'ler, yüksek hızlı paralel ve seri giriş/çıkış portları, harici bellek ara yüzleri, Gigabit Ethernet gibi alıcı/verici ara yüzleri sayesinde FPGA'lerin birçok alanda tercih edilmesini sağlamıştır.

FPGA kullanımının bu denli popüler olmasının en önemli sebeplerinden birisi de, güçlü yazılım desteğidir. FPGA üreticilerinin sundukları gelişmiş PCB tasarım, modelleme ve simülasyon yazılımları ile yapılandırılabilir hazır IP core'ler tasarımcılara kolaylık sağlamaktadır. fikri mülkiyet çekirdekleri (IP core), tasarımcıların işlerini oldukça kolaylaştırmaktadır [16-18].

FPGA'i bu denli popüler kılan tüm bu özelliklere ek olarak **teknolojideki kullanım alanları** yadsınamayacak derecede geniş yer kaplar. Bu anlamda geniş yer kaplayan bu teknolojinin sanayi alanlarını aşağıdaki gibi sınıflandırabiliriz [17].

Uzay ve Savunma Sanayi'nde; Güvenli (kriptolu) iletişim, radar & sonar, elektronik harp, aviyonik sistemler ve silah sistemleri…

Otomotiv Sektöründe; görüntü işleme, araç kontrol sistemleri, araç içi bilgi-eğlence sistemleri…

Endüstriyel Otomasyon ve Kontrol Sistemlerinde; motor kontrol, endüstriyel görüntüleme, endüstriyel ağlar…

Bilgisayar / Depolama Alanında; sunucular, disk sürücüler, yazıcı / fotokopi…

Kablolu / Kablosuz İletişimde; 3G teknolojisi, GSM, ADSL, VDSL, radyo dalgaları, optik ağlar, bilgisayar ağları…

Tıbbi Elektronikte; Ultrason görüntüleme, bilgisayarlı tomografi (CT), MRI görüntüleme, PET görüntüleme, yaşam destek üniteleri, hastane ve laboratuvar elektroniği...

TV / Radyo Yayıncılığında; gerçek zamanlı ve yüksek çözünürlüklü grafik işlemcileri, video dağıtım cihazları, video anahtarlama ve yönlendirme cihazları, profesyonel kameralar, profesyonel görüntüleme sistemleri…

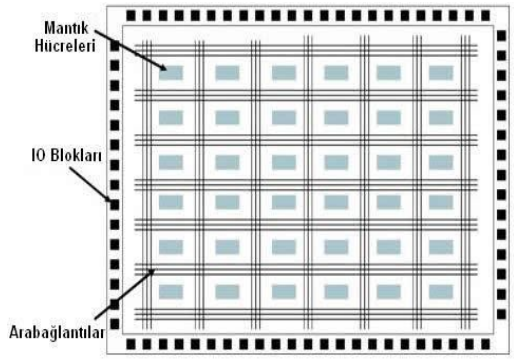
Test / Ölçüm Alanında; ağ / protokol analizörleri, spektrum analizörleri, bit-hata test cihazları, yarı iletken tabanlı test cihazları, osiloskoplar, lojik analizörler, sinyal üreteçleri…

Tüketici Elektroniğinde; akıllı telefonlar, LED-LCD-plazma TV'ler, uydu alıcılar, projektörler, multimedya cihazları, dijital kameralar, navigasyon & GPS…

Güvenlik / Şifreleme Sektöründe; kripto cihazları, askeri iletişim araçları, gömülü şifreleme, ağ şifrelemesi, datalink uygulamaları, taktik telsizler, uydu iletişimi, finans ve bankacılık, güvenli veri depolama…

3.3. FPGA İç Yapısı

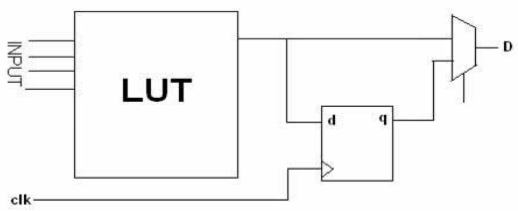
FPGA mimarisi, Şekil 3.5.'de olduğu gibi başlıca mantık elementleri-hücreleri (Logic Cells), Giriş/Çıkış Blokları (I/O Block) ve de ara bağlantılar olmak üzere 3 kısımdan oluşmaktadır.



Şekil 3.5. FPGA İç Yapısının Gösterimi

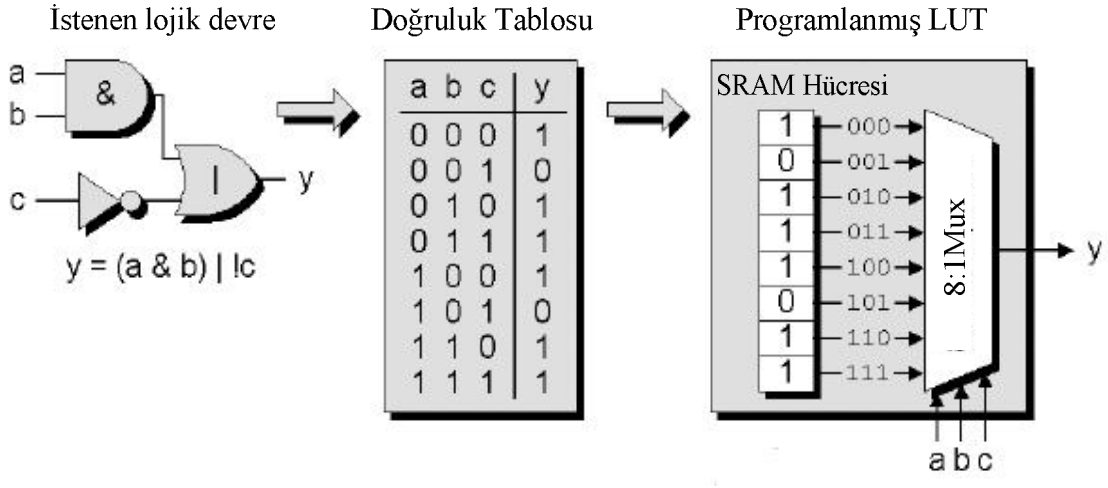
3.3.1. Mantık Hücresi (Logic-Cell)

Şekil 3.6.'da görülen bir mantık hücresi FPGA'in ana temelini oluşturmaktadır. Bir mantık hücresi bir adet **LUT**, bir adet D tipi **FF** ve bir adet **2x1 Multiplexer**'dan oluşur.



Şekil 3.6. Mantık Hücresi

LUT 'lar aslında bir mantık işlemi yerine getiren küçük belleklerdir. N girişli bir LUT, 2^N 'li bir belleğe işaret eder. Binlerce Mantık Hücrelerinin birleşimi sonucu olarak karmaşık ve büyük programlar oluşturulur. Mantık hücrelerinin ara bağlantıları matris şeklindeki veri yolları ve programlanabilir anahtarlarla sağlanmaktadır. FPGA tasarımı için Şekil 3.7.'de gösterilen LUT'un yapılandırılması ile mümkün olmaktadır. Her bir mantık hücresinin uygulayacağı fonksiyonu ve programlanabilir anahtarların durumunu (açık/kapalı) belirleyerek bu mantık hücreleri arasındaki bağlantıları tanımlar [16,17].



Şekil 3.7. LUT'un Yapılandırılması

3.3.2. FPGA Pinleri

FPGA pinleri genel olarak ayrılmış pinler ve kullanıcı pinleri olmak üzere 2 kategoriye ayrılmaktadır. Ayrılmış pinler, dedicated pins olarak, kullanıcı pinleri ise user pins olarak tabir edilebilir.

a) Ayrılmış pinler;

Gerçekleştirdikleri fonksiyonlara göre 3'e ayrılırlar.

- **Güç Pinleri:** FPGA'in enerjisi gereksinimini (güç ve ground (toprak)) sağlayan pinlerdir.
- **Konfigürasyon Pinleri:** Programın FPGA'e yüklenmesi için ayrılmış pinlerdir.
- **Clock Pinleri:** Saat sinyalleri için özel ayrılmış pinlerdir.

b) Kullanıcı Pinleri;

Bu pinler tasarımcı tarafından ayarlanabilen standart giriş/çıkış pinleridir. Input, Output, Input/Output olarak üç kategoriye ayrılır [21,24].

3.3.3. Clock ve Global Lines

FPGA tasarımları saat tabanlıdır ve FPGA içerisindeki D Flip-Floplar, saat sinyalinin yardımıyla durum değiştirirler. Senkronize olmuş tasarımlarda bir saat sinyalinin, bütün flip-flop'ları aynı anda tetiklemesi gerekir. Aksi takdirde FPGA'de elektriksel ve zamansal problemler ortaya çıkmaktadır.

3.3.4. RAM Blokları

Günümüzde FPGA'lerde bellek üniteleri yani RAM ayrılmış bir şekilde bulunmaktadır. Bunlar ayrılan bellek üniteleri mantık devrelerinin çalışması sırasında duyulan geçici hafıza gereksinimi için hazırdırlar. Bu bellek üniteleri tekli ve çoklu erişimi destekleyebilirler. Multi erişimde birçok işlem, RAM üzerinde okuma/yazma yapabilir. Örneğin 35 MHz clock sinyali ile çalışan bir veri alma ünitesinden 60 MHz ile çalışan bir veri işleme ünitesine bilgi göndermek için iki portlu bir RAM kullanılabilir. 35 MHz ile çalışan veri alma ünitesi bilgiyi RAM'e yazar ve 60 MHz de çalışan veri işleme ünitesi bilgiyi RAM'den okuyarak kullanabilir [25].

3.4. FPGA’de Programlama

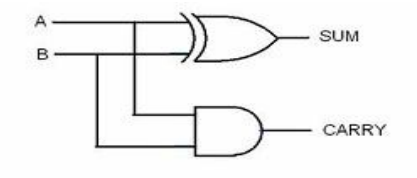
FPGA'ler HDL dillerinden biri ile programlanabilen entegre devreler olduğundan FPGA donanımına ruh verebilmek yani çalışır hale getirebilmek için programlanmaları gerekir. (Altera'da ise Quartus II ara yüzü, Xilinx'de ise ISE Design Suite derleyici programları kullanılır.)

FPGA programlanması Grafiksel Tasarım ve HDL yöntemi olmak üzere 2’ye ayrılır. Grafiksel tasarımda; tasarım, derleyici programın (ISE Design Suite veya Quartus) kütüphanesinde bulunan element ve mantık kapılarını kullanarak,

HDL(Hard. Desc. Lang.) de ise; tasarımın, Verilog veya VHDL’den bir tanesini kullanılarak programlanmasıdır. Verilog ile VHDL arasında herhangi bir üstünlük ya da yetersizlik mevcut değildir. Her iki yazılım dilinin de kendi karakteristiğine göre yapısı vardır. Kullanıcı FPGA programlamak için her iki dili de kullanabilmektedir.

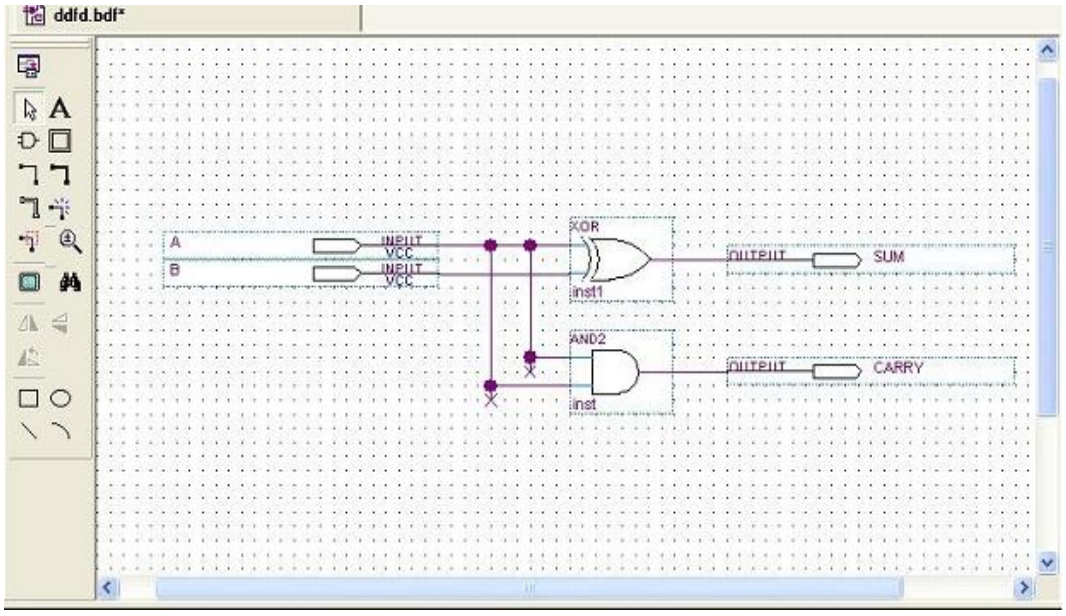
Program tasarlanırken hem grafik hem de VHDL/Verilog kodu yazılabilir. Tasarımların grafiksel olarak dizaynı için HDL ile oluşturduğumuz modellerin şematikleri çizilebilmektedir [26,27].

Örneğin Şekil 3.8.’de verilen bir adet yarım toplayıcının grafiksel tasarımı Şekil 3.9.’dadır. VHDL ile tasarımını irdelemek için ayrıca VHDL kodlarının yazılması gerekmektedir.



Şekil 3.8. Yarım Toplayıcı (Half Adder)

Schematic (şematik) modunda kullandığımız programda bir tane Ve kapısı ile bir tane Xor kapısı yandaki componentlerden seçilerek gerekli bağlantılar ve pin bağlantıları yapılarak yarım toplayıcı tasalanabilir.

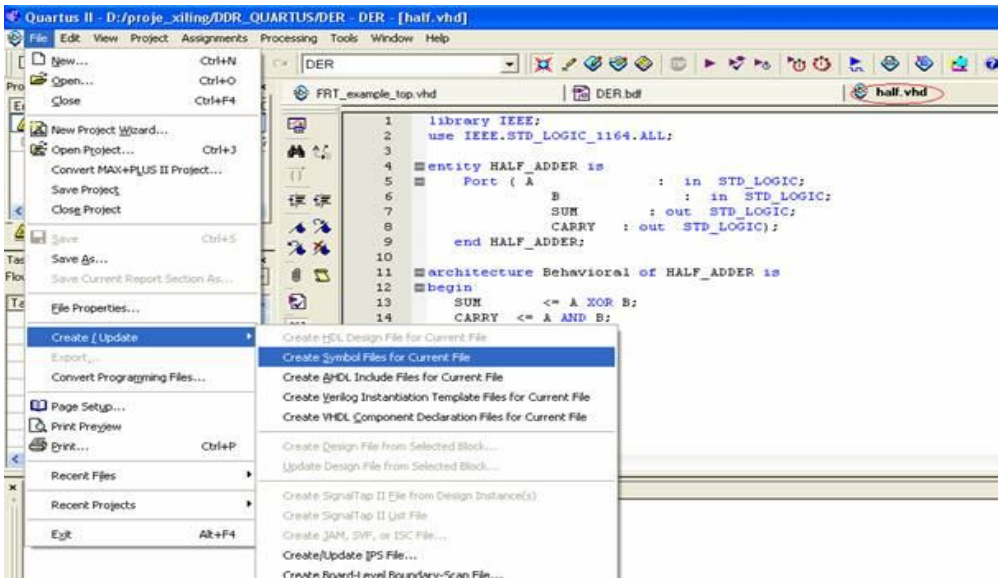


Şekil 3.9. Half Adder'ın Schematic Çizimi

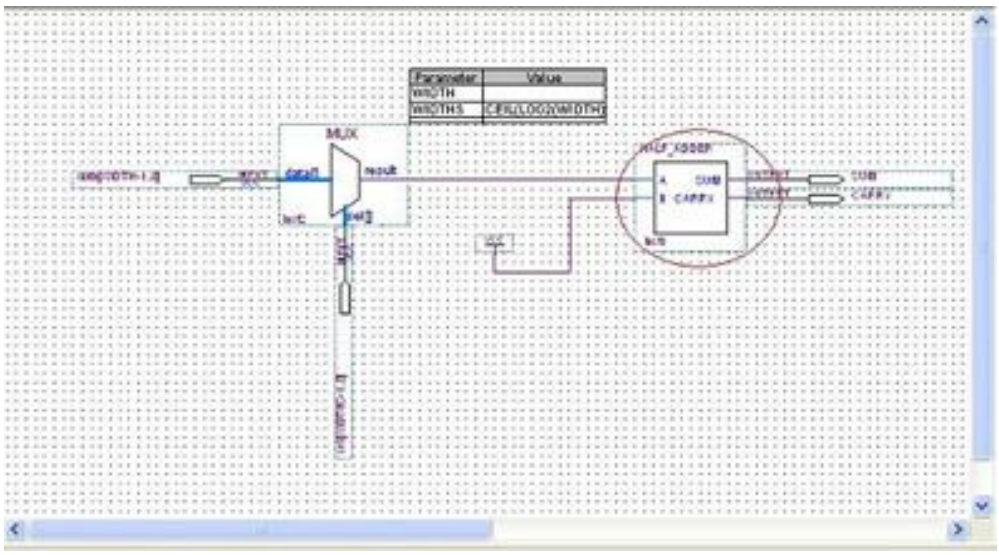
HDL dillerinden birisi olan VHDL kullanarak oluşturmak için aşağıdaki kodları yazmamız gerekecektir.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity yarim_toplayici is
Port (    x          : in  bit;
          y          : in  bit;
          toplam     : out  bit;
          elde       : out  bit);
end yarim_toplayici;
architecture Behavioral of  yarim_toplayici is
begin
toplam<= x XOR y;
elde  <= x AND y;
 end Behavioral;
```

Hem grafiksel olarak hem de VHDL kodlar ile oluşturduğumuz tasarımlar derlenirse aynı işlevi (half adder) gören bir yapılar gözlemlenecektir [26]. VHDL kodları derleme işlemini Şekil 3.10.'daki ara yüz ile, Schematic çizimi ise Şekil 3.11.'de gösterilmiştir.



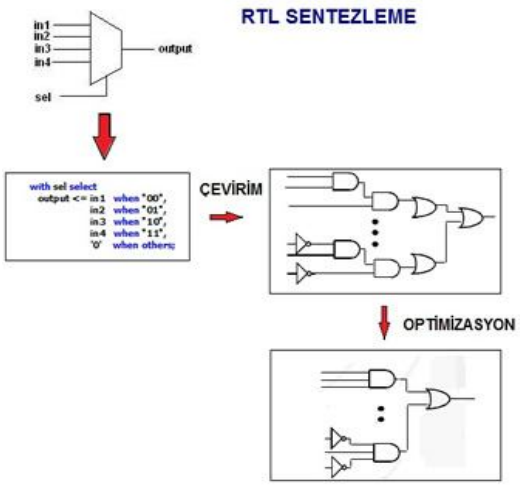
Şekil 3.10. ISE Design Suite Ara Yüzü



Şekil 3.11. Schematic Çizim

Analiz ve Sentezleme: Bu aşamada tercih edilen derleyici program, HDL dili ile yazılan kodu analiz edip, donanımsal olarak gerçekleşip gerçekleşmeyeceğine karar vermektedir. Eğer kod parçasının gerçekleşmesi mümkün ise kod sentezlenir ve Şekil 3.12.'de gösterildiği gibi Register Transfer Level şeması çıkarılır.

RTL; sentezlenen kodun, register cinsinden tasarımında belirtilmesine karşı düşmektedir. Kısacası VHDL koduna karşılık gelen lojik kapılarından oluşan devre kısmıdır.



Şekil 3.12. RTL Şeması Gösterimi

Kısıtlamaların girilmesi; program dosyasını oluşturmadan önce tasarımda kısıtlamaların girilmesi gerekir. Bu kısımda pin atamaları belirlenmektedir. Yani çıkış veya giriş olmasını istediğimiz kısmın FPGA'da hangi pine karşılık geldiği belirtilmektedir.

Yerleşim ve bağlantıların yapılması (Place ve Routing); derleyici ara yüz programı kısıtlamaları da göz önünde bulundurarak tasarıma ait programın yerleştirileceği mantık elemanları belirler. Mantık elemanları ile atanan pinler arasındaki bağlantı bu kısımda yapılmaktadır.

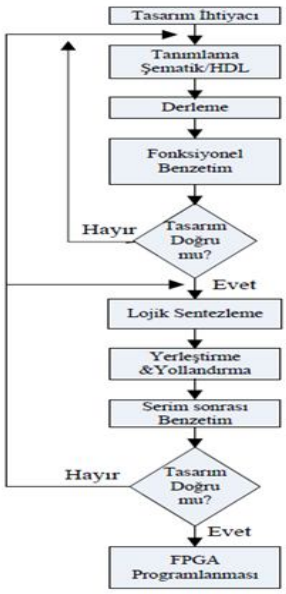
Program dosyasının oluşturulması, yerleşim ve bağlantıların yapılması aşamasından sonra derleyici ara yüz programı, programın yükleneceği FPGA'lere uygun programlama dosya formatını oluşturmaktadır. Program direkt FPGA'e yüklenecek ise JTAG uzantılı, konfigürasyon elemanı veya flash kullanılacaksa, bu cihazların desteklediği program uzantılı dosyalar oluşturulmaktadır.

Programın Yüklenmesi; yukarıdaki aşamalardan sonra program dosyası FPGA'e yongasına veya konfigürasyon elemanlarına aktarılarak işlemler tamamlanmış olur.

3.4.1. FPGA Akış Şeması

Oluşturulan tasarımlar, derlenme boyunca Şekil 3.13.'teki adımlardan geçmektedir

- Tasarım girişi
- Sentezleme
- İşlevsel benzetim
- Yerleştirme
- Zaman analizi ve benzetimi
- Programlama ve yapılandırma



Şekil 3.13. FPGA Akış Şeması Gösterimi

3.4.2. FPGA'e Program Yüklenmesi

FPGA program uçucu özellikte olduğu için programı her zaman saklayamaz. Üzerine aktarılan program elektriksel bağlantı kesildiği takdirde kaybolmaktadır. Aynı tasarım tekrar kullanılacağı zaman, yazılımın FPGA'e yeniden aktarılması gerekmektedir.

Programın her defasında uçmasını engellemek için FPGA ile birlikte konfigürasyon elemanı kullanılmalıdır. Bu konfigürasyon elemanları olarak; bilgisayarı, mikrodenetleyicileri veya Boot-PROM'u sıralayabiliriz.

•**Bilgisayar:** Çabuk ve kolay yazılım geliştirme amacı ile kullanılmaktadır. Tasarım bittikten sonra, bilgisayara gereksinim yoktur. FPGA kartının bilgisayar ile haberleşmesi için çeşitli kablo çeşitleri mevcuttur.

• **Mikrodenetleyici:** İçinde gömülü bir yazılım bulunduran mikrodenetleyici ile FPGA konfigüre edilebilir.

• **Boot-PROM:** FPGA üreten firmaların özel Boot-PROM'ları vardır. Boot-PROM'lara enerji verildiği zaman program otomatik bir şekilde aktarılmış olur.

Aslında FPGA'i programını aktarmak diğer bir deyişle konfigüre edebilmek demek cihaza sıfır ve birlerden oluşan anlamlı bilgii paketlerini göndermektir denilebilir.

FPGA'in Konfigürasyon (configuration) ve Kullanıcı (user) modu olmak üzere iki modu vardır.

Konfigürasyon modu; enerji verildiği zaman FPGA, konfigürasyon moduna geçmektedir. Bu aşamada FPGA beklemededir ve tüm pinler pasiftir. FPGA'e yazılım aktarımı bu aşamada yapılmaktadır.

Kullanıcı modu; FPGA'e yazılım yüklendikten sonra kullanıcı moduna geçilir ve çıkışları aktif olur.

Altera ve Xilinx'in konfigürasyon metotları birbirine benzerdir fakat FPGA programlanırken JTAG ara yüzü ve Synchronous Serial ara yüzü olmak üzere iki ara yüzden biri kullanılmaktadır [24].

3.5. FPGA Üreticileri

Altera ve Xilinx piyasanın en popüler iki FPGA üreticilerindendir. Diğer önemli üreticiler olarak; Lattice, Actel, Quicklogic ve SiliconBlue firmaları sayılabilir[2].

3.5.1. Xilinx

FPGA'in ilk üreten ve dünya piyasasının en büyük firmasıdır. Derleyici için ISE Design Suite ara yüz programını sunmaktadır. Virtex, Kintex ve Spartan bu firmanın sunduğu serilerdendir. Firmanın amblemi Şekil 3.14.'te verilmiştir.

Şekil 3.14. Xilinx Amblemi

3.5.2. Altera

Altera, FPGA'in mucidi olan Xilinx firmasının en hızlı rakibidir. Derleyici için Quartus II programını tasarımcılara sunmaktadır. Bu firmanın sunduğu seriler arasında Stratix, Cyclone, Arria bulunmaktadır. Firmanın amblemi Şekil 3.15'te verilmiştir.

Şekil 3.15. Altera Amblemi

4. VHDL - DONANIM TANIMLAMA DİLİ

VHDL'in ingilizce kökeni Very High Speed Integrated Circuit Hardware Description Language'den gelir. Bu dil aynı zamanda çok yüksek hızlı tümleşik devre donanım tanımlama dili olarak da tabir edilebilir. HDL dillerinden en çok kullanılanlardan biridir. Bu programlama dili 1980'lerden bu yana süregelmiş ve devamlı geliştirilmiştir. IEEE topluluğu tarafından da bir standart olarak kabul edilmiştir.

HDL donanım tanımalama dillerinden birisi olan VHDL başlıca sentezleme ve simülasyon için kullanılır.

- **Sentezleme**: Kodun FPGA'e yüklenmeden öncki halini kurgular,
- **Simülasyon**: FPGA'e yüklenecek kodun simülasyonunu gerçekleştirme,

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Ve_kapisi is
Port (     x : in bit;
y : in bit;
z : out bit);
end Ve_kapisi ;
architecture Behavioral of Ve_kapisi  is
begin
z <= x AND y;
end Behavioral;
```

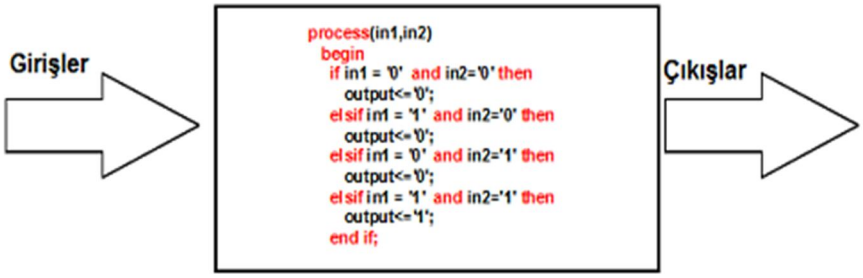
Yukarıdaki örnekteki yazılımda 2 girişli bir çıkışlı **Ve** kapısı tanımlanır [28].

4.1. Donanım Tanımlama Dili-HDL

HDL, herhangi bir donanım kısmını yapılandırmada tercih edilen bir programlama dilidir. HDL, yazılım kullanılarak donanımları modelleme ve onların davranışlarını belirleme durumu sağlamaktadır. VHDL ve Verilog dilleri, FPGA donanımının programlanmasında en çok kullanılan donanım tanımlama dillerindendir.

4.1.1. Davranışsal Modelleme

Davranışsal modelleme sistemin dış dünya ile alakalı davranışını belirleyerek Şekil 4.1.'deki gibi giriş ve çıkışa müdahale eder. Modelin giriş ve çıkış tepkimeleri davranışsal olarak tanımlanır. Modelin iç yapısına ve durumuna bakılmaz. Burada modelin işlevi ve fonksiyonu irdelenmektedir.



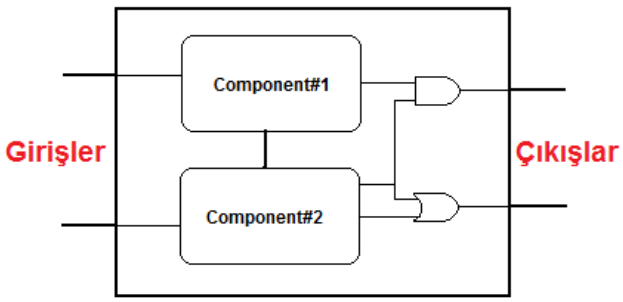
Şekil 4.1. Davranışsal Model

4.1.2. Yapısal Modelleme

Alt seviyelerdeki bileşenler ile arasındaki alakaları göstermektedir. Yapısal modelleme Şekil 4.2.'deki gibi davranışsal modelin gövdesinin yapılandırılmasıdır.

VHDL programcıya yapısal modelleme için olanak sağlamaktadır. Bu modelleme çok karmaşık ve büyük donanım parçalarında kolaylık sağlamaktadır.

VHDL tasarımlarında genellikle tercih edilen teknik, birbirinden farklı modellenen alt kısımdaki modüllerin structural modelleme ile oluşturulup; bu modüllerin üs kısımdaki yapısal modelleme ile birbirine bağlanması tekniğidir [29].

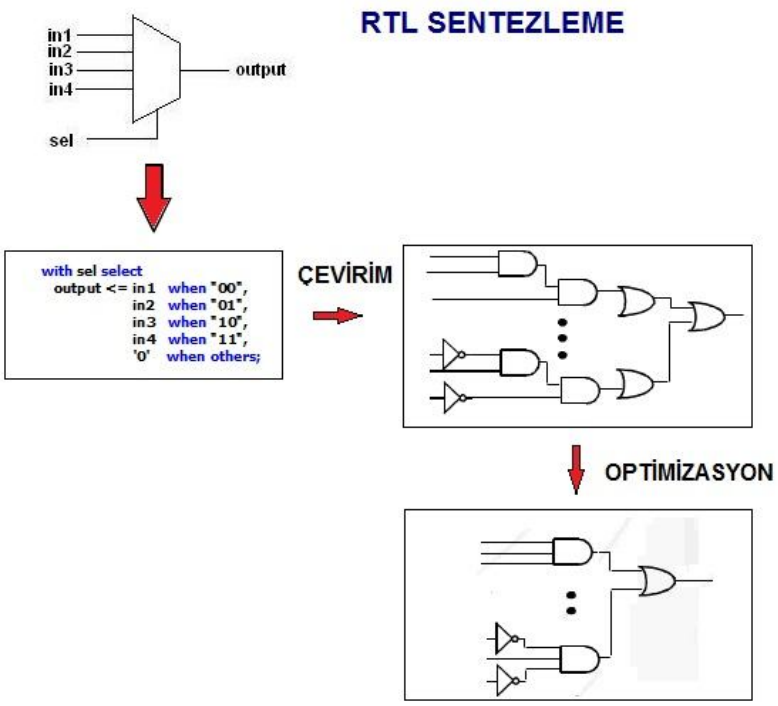


Şekil 4.2. Yapısal Model

4.1.3. Register Transfer Level (RTL)

RTL yöntemi, bir soyutlanmadır. Soyutlanma yazılımının analiz amacı gözeterek tercih edilir. Bu yöntem, tasarım için kodun register cinsinden tabir edilmesidir. VHDL koduna karşılık düşen yapıların lojik elementlerinden oluşan kısmıdır.

Şekil 4.3.'te de görüldüğü gibi RTL modeli için çevrim ve optimizasyon adımları işlenir. RTL analizde VHDL kodu ilk başta çevrim işlemi ile dijital bir bloğa dönüştürülür. Ardından da iyileştirme işlemiyle VHDL kodunun karşılığı olan mantık kapıları kısmı iyileştirme edilerek, FPGA donanımının efektif bir biçimde kullanılması sağlanır.



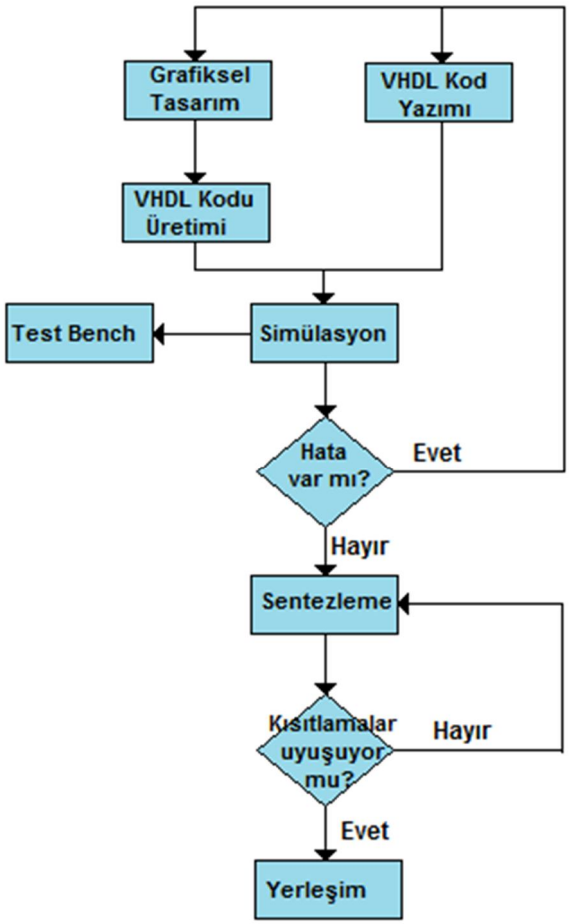
Şekil 4.3. RTL Modeli

Yukarıdaki şekilde 4 girişli tek çıkışlı bir MUX'un (veri seçici) RTL modeli görülmektedir. Tasarım VHDL olarak yazıldıktan sonra, compiler ile sentezlenir. Sentezlenen bu kısmın karşılığı olan mantık kapılarına dönüştürülür. Optimizeden sonra RTL akışı tamamlanır [30,31].

4.2. VHDL Tasarım Akışı

VHDL'de tasarım akışı sırasıyla aşağıdaki üç maddeden oluşmaktadır. Algoritması Şekil 4.4.'te görülen gibi tasarım akışının temel üç maddesi;

- Kodlama
- Simülasyon
- Sentezleme



Şekil 4.4. Tasarım Akışı Gösterimi

Kodlama: Bu kısım, programın VHDL kodunun yazıldığı kısımdır.

Simülasyon: Bu kısımda oluşturulan VHDL kodunun simülasyonu yapılır, programın doğruluğu gözlemlenir. Simülasyon sırasında ortaya çıkan hatalar, program FPGA'e yüklenmeden önce doğrulanmaktadır [16,28].

Sentezleme: Bu kısımda yazılan VHDL kod, donanım diline çevrilip RTL modeli çıkartılır. Sentezlemeden sonra kod derleyici tarafından FPGA'e yüklenmeye hazırlanacak

konfigürasyon dosyasına çevrilir. Sentezleme işlemi genel olarak derleyici tarafından yürütülmektedir. Tasarımcı tarafından da sentezlenebilir fakat kullanışlı bir yöntem asla değildir [32].

4.3. VHDL Yazım Kuralları

- VHDL kod grubu iki kısımdan oluşur.

Sentezlenebilen Kodlar; FPGA'e yüklenebilir kodlarıdır. Fakat VHDL kodların tamamı FPGA'e yüklenebilir değildirler.

Sentezlenemeyen Kodlar: Yalnızca test amaçlıdırlar. Sentezlenemez ve dolayısıyla FPGA'e yüklenemezler.

- VHDL'de kendine özgü olan kodları vardır. Örneğin case, while, if, with, and, vb...

VHDL'de ifadeler eş zamanlı(paralel) ve ardışık(sıralı) olarak ikiye ayrılır.

Eş zamanlı ifadeler; Aynı anda işlenirler.

Ardışık ifadeler; diğer dillerde de olduğu gibi sıraya göre işlenirler.

- VHDL komut satırların sonuna noktalı virgül (;) konur.

C<=5 D<=6;

"<=" sembolü küçük eşittir anlamından farklı olarak sinyal atama işine yarar. Sağdaki değerin soldakine atanmasına karşılık gelmektedir.

- VHDL kodlarında harf duyarlılığı yoktur.

Type= type=TypE

- VHDL kodunda boşlukların anlamı yoktur ve düzeni etkilemez fakat koddaki boşlukların düzenli bırakılması karmaşık kodları anlaşılır ve okunur hale getirir.

K<= M and N; ile K <= M and N; aynı anlamdadır.

- Kod kısmında herhangi bir açıklamanın yazılmak istenirse başına "--" ile başlanarak yazılmaya müsait hale getirilir. Başına tire gelen ifadeler koda dahil olmazlar [34].

```
K <= L + M;     -- K çıkışına L ve M girişlerinin toplamını at
```

4.4. VHDL Temel Bölümleri

Bir VHDL tasarımı başlıca 5 ana bölümden oluşmaktadır.

- Entity (Entity)
- Mimari (Architecture)
- Paket (Package)
- Bileşen (Component)
- İşlem (Process)

Entity;

Bir tasarımın en temel bloğudur. Verilen bir mantık fonksiyon için bütün giriş ve çıkışları yani mantık fonksiyonun dış dünya ile bağlantısını tanımlar. Her VHDL tasarım mutlaka en az bir entity içerir. Port tanımlamaları farklı biçimlerde olabilir. Burada port tanımlaması ile işaretin giriş/çıkış olduğu ve veri tipi belirtilir. Entity, tasarım ile onun dışındaki ara yüzü tanımlamaya yardımcı olmaktadır. Bu bölümde her türlü I/O portlarının tanımlanması yapılır.

Mimari (Architecture);

Tasarımın ne iş yaptığının belirlendiği kısımdır. Mimari entity'nin davranışını tanımlar. Bir entity birden fazla mimariye sahip olabilir. Bir mimari davranışsal (behavioral) tanımlama, yapısal (structural) tanımlama, veri akışı (data flow) olmak üzere üç farklı biçimde kullanılabilir.

Paket (Package);

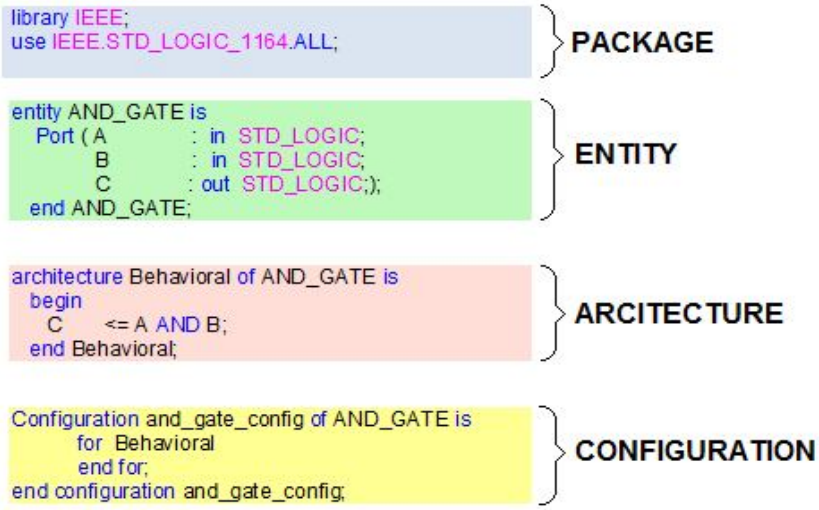
Paket; entity üniteleri tarafından kullanılan tanımlamaları bir grup haline getirir ve paylaştırır. Package ayrıca istenen yapıları farklı tasarımlarda hazır tutmak üzere tasnifleyen kısımdır.

Bileşen (Komponent);

Bileşen yapısı; devre tanımlamasında bir alt devre gibi kullanılan bileşenin adını ve ara yüzünü tanımlar.

İşlem (Process);

İşlem bloğu sıralı şekilde gerçekleştirilecek durumları içerir. Bir mimaride birden fazla işlem bloğu anlık gerçekleştirilir. Yani işlem blokları aynı anda başlar ve her bir işlem bloğu kendi içinde satır satır sıralı olarak gerçekleştirilir [35].



4.5. Veri Nesneleri

VHDL nesne özelliklere sahiptir. Nesnelerin davranış ve işlevlerine göre fonksiyonellik kazanır.

4.5.1. Sinyal (Signal)

Sinyaller entity, mimari ve paket içinde kullanılabilir. Sinyallerin paket içinde kullanımı çok önemlidir. Çünkü paket genel bir ifade olduğundan dolayı işaretler burada kullanıldığı zaman diğer entity veya mimariler tarafından çağrılarak kullanılabilir. Sinyallere başlangıç değeri atanabilir ve işlem gerçekleşirken bu ifade yenilenebilir [31].

signal sinyalin_ismi: sinyal_tipi :=değeri;

4.5.2. Değişken (Variable)

Genel olarak geçici değerleri kullanmak için tercih edilmektedir. Böylece programın daha hızlı çalışması sağlanabilir. Ortak değişken (shared variable) türü de bulunmaktadır. Eğer bir değişken birden fazla process'de ve alt programda çağrılacak ise ortak değişken olarak tanımlanma yapılmalıdır [31].

variable değişkenin_adı: değişkenin tipi := değeri;

4.5.3. Sabit (Constant)

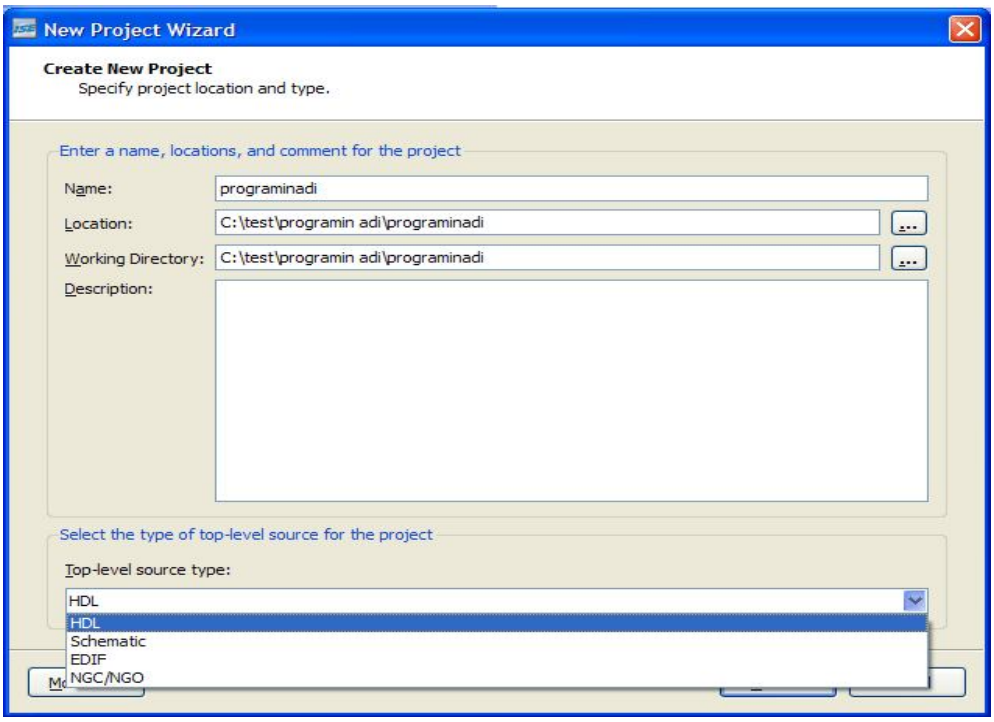
Değeri bilinen ve değişmeyen verilere ihtiyaç duyulursa sabitler kullanılır. Bu durum, tasarımın daha iyi gözlemlenebilmesi ve anlaşılabilmesi için önemli bir faktördür. Özellikle tasarım içinde sık sık kullanılan değeri aynı olan ifadeler için vazgeçilmez bir veri tipidir [31].

constant sabitin_adı: sabitin tipi := değeri;

5. XILINX ISE DESIGN SUITE YAZILIMI

Xilinx ISE Design Suite programı VHDL kodlarıyla yazılan programı derlemek için kullanılmaktadır. FPGA'in programlanması için ilk olarak Xilinx ISE Design Suite programının bilgisayara yüklenmesi gerekmektedir. Öncelikle yüklenen yazılımın simgesine çift tıklanarak program çalıştırılır. Ardından yazılmak istenen program için yeni bir proje oluşturulur.

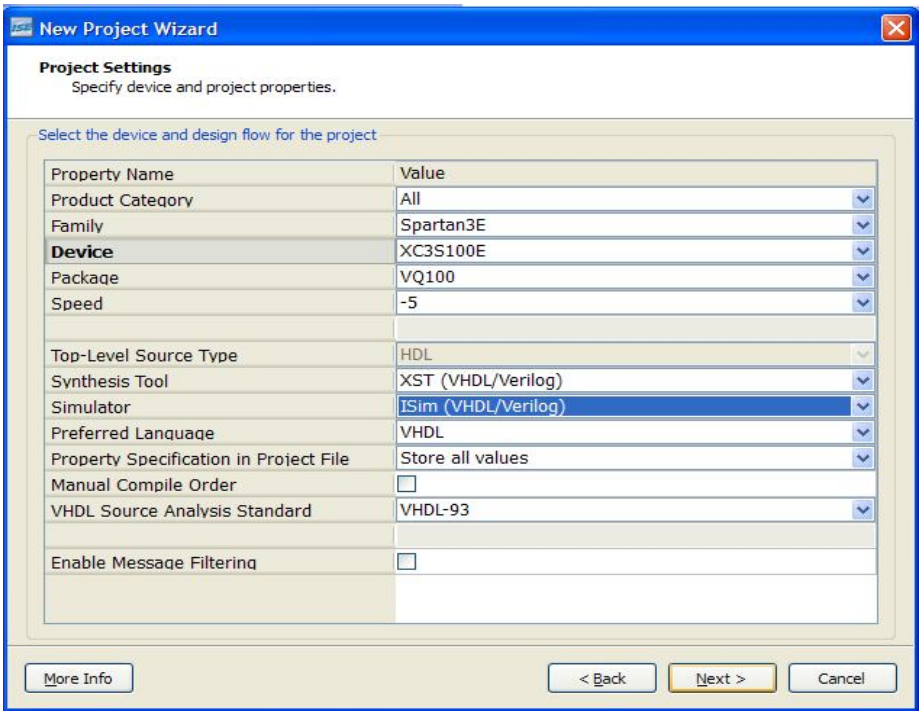
Bu işlem için **File >> New Project'e** tıklanır. Ardından gelen ekranda programın adı yazılır ve kullanılacak kaynak kod türü seçilir.



Şekil 5.1. Program Yazma Derleme-1

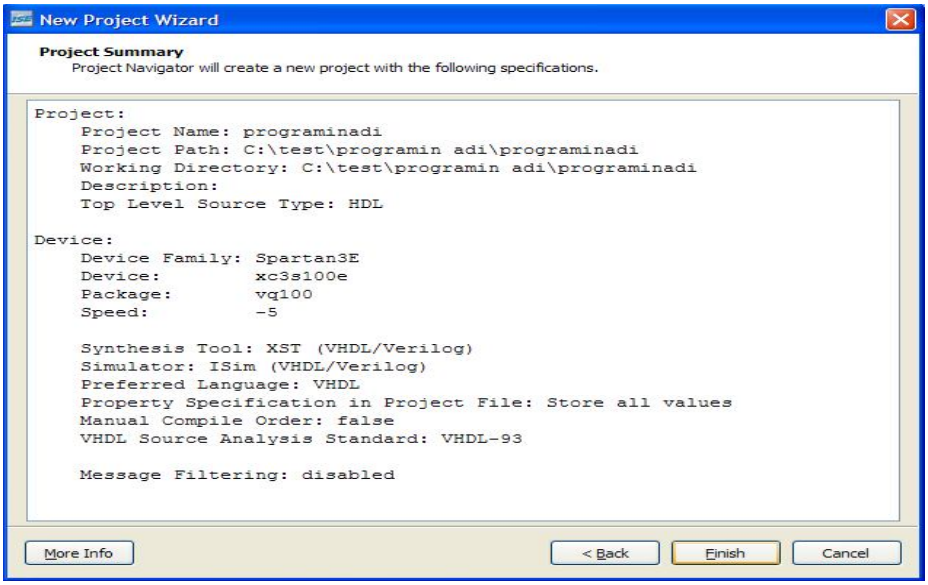
Yukarıdaki Şekil 5.1.'de HDL seçildikten sonra Next'e tıklanır.

Şekil 5.2.'deki gelen ekranda aşağıda görülen ayarlar yapılır.



Şekil 5.2. Program Yazma Derleme-2

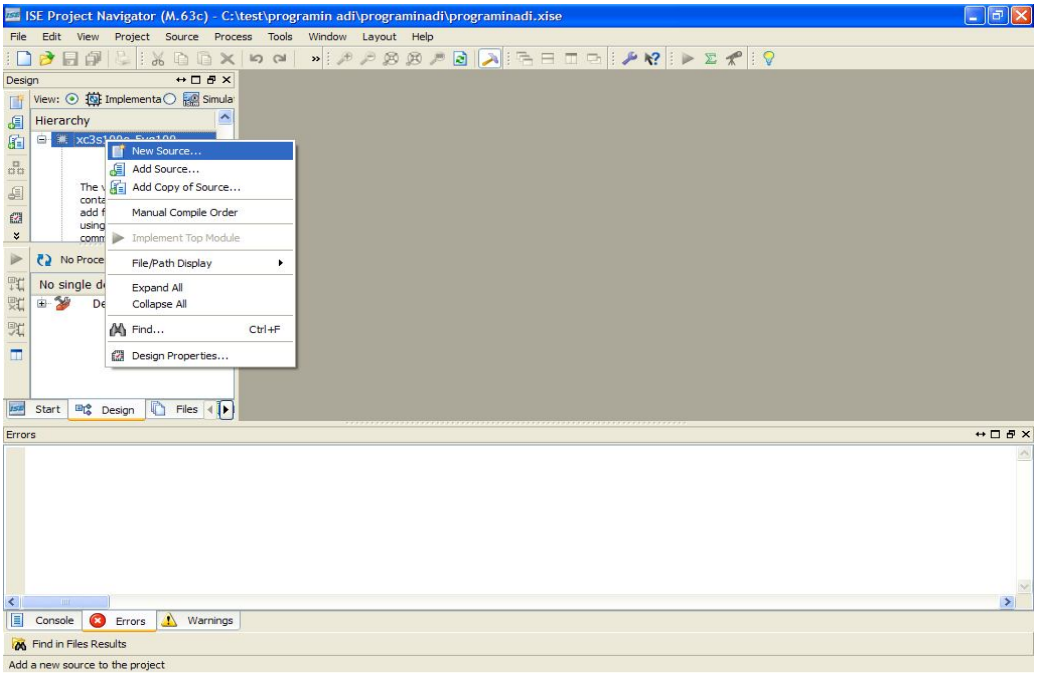
Next'e tıklanır.



Şekil 5.3. Program Yazma Derleme-3

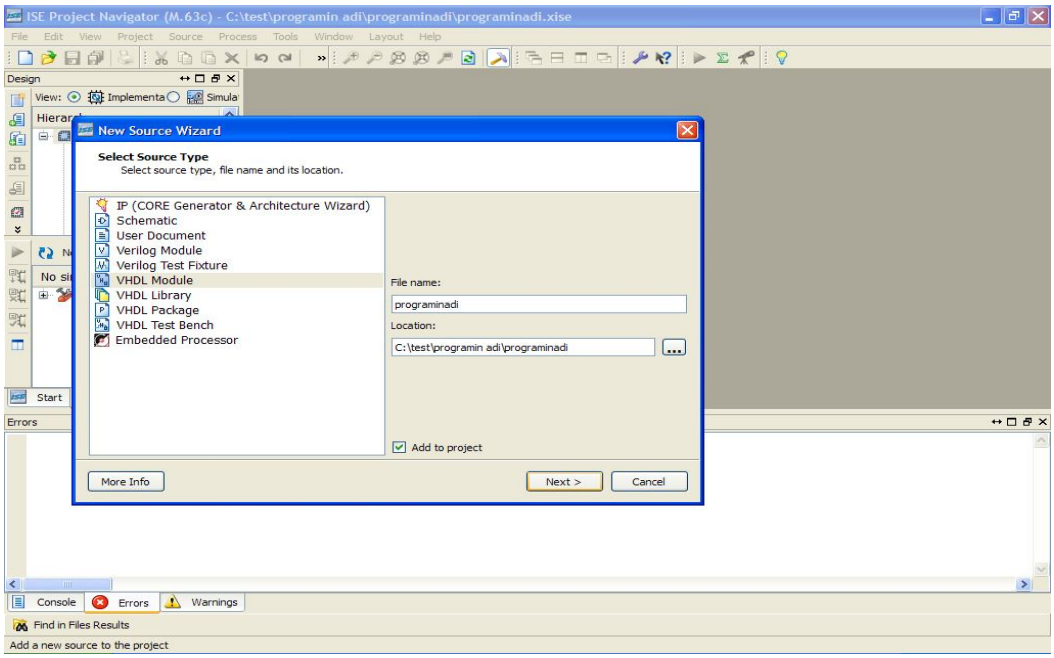
Yukarıda Şekil 5.3'te yeni hazırlanmış olan projenin bir özeti yer almaktadır. Finish'e tıklanarak işlem tamamlanır.

Daha sonra proje dâhilinde bir program oluşturmak gerekmektedir. Bunun için, Şekil 5.4.'te sol menüde hiyerarşi kısmında kullanılan kitin ismine sağ tıklanıp New Source seçilir.



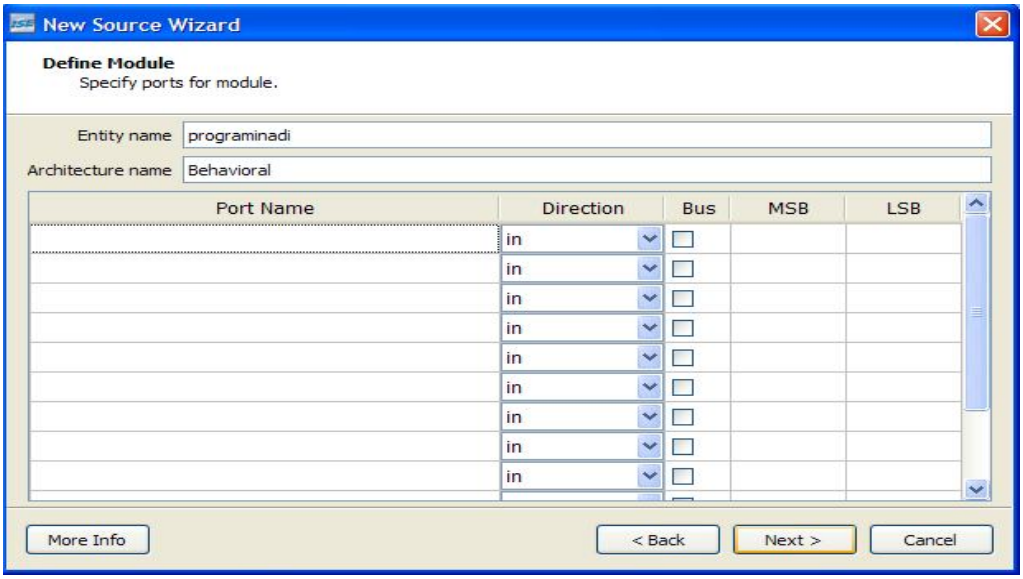
Şekil 5.4. Program Yazma Derleme-4

Şekil 5.5.'de Açılan pencerede VHDL Module'ü seçilir ve File Name'e programın adı yazılır. Ancak burada program adı ile dosya adının aynı olması ilerleyen safhalarda sorun yaşanmaması için önemlidir.



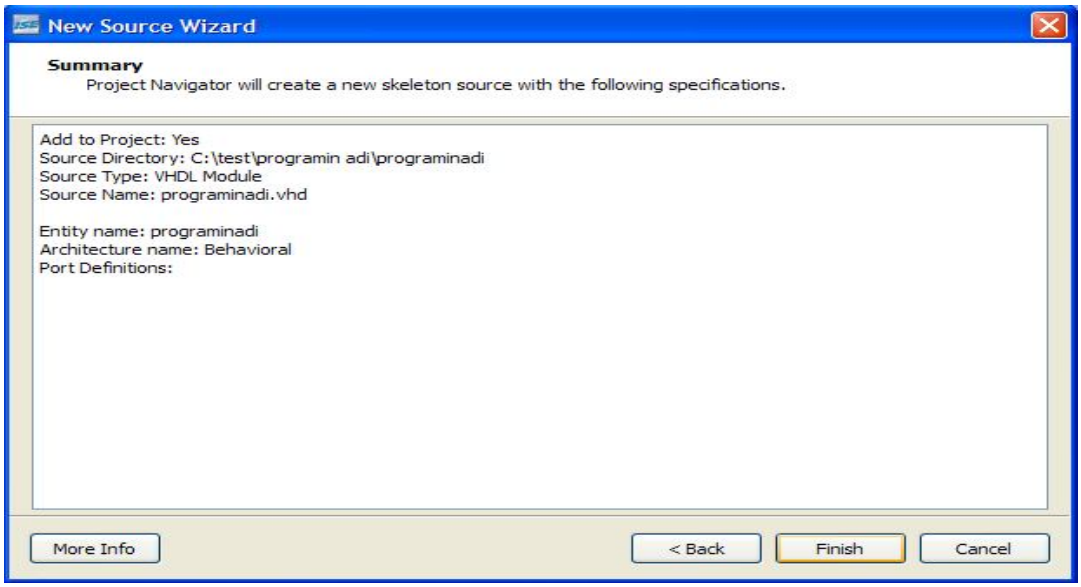
Şekil 5.5. Program Yazma Derleme-5

Next'e tıklanır ve pin numaraları ile Şekil 5.6.'daki pencere görüntülenir.



Şekil 5.6. Program Yazma Derleme-6

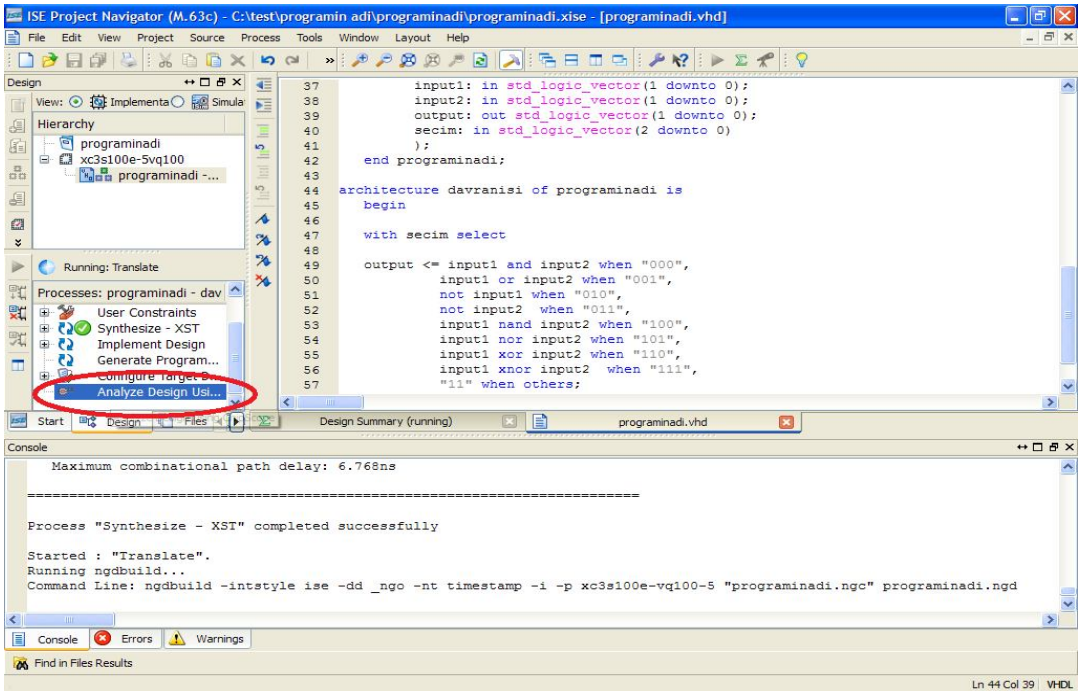
Yazılacak programa göre pin atamaları daha sonra yapılacağından bu kısım es geçilir ve Next'e tıklanır.



Şekil 5.7. Program Yazma Derleme-7

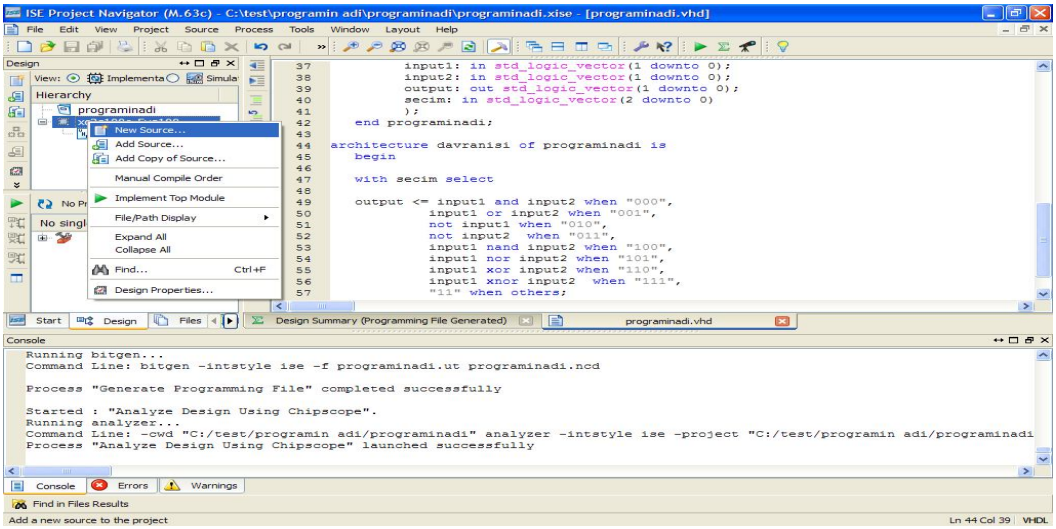
Şekil 5.7.'deki bu pencerede kullanılacak dosya ile ilgili özet bilgi yer almaktadır. Finish'e tıklanarak program ekranına geçiş yapılır.

Editörde program yazıldıktan sonra kaydedilip Şekil 5.8.'de de görülen sol alt menüde processes kısmındaki Analyze Design Using ChipScope'a çift tıklanır.



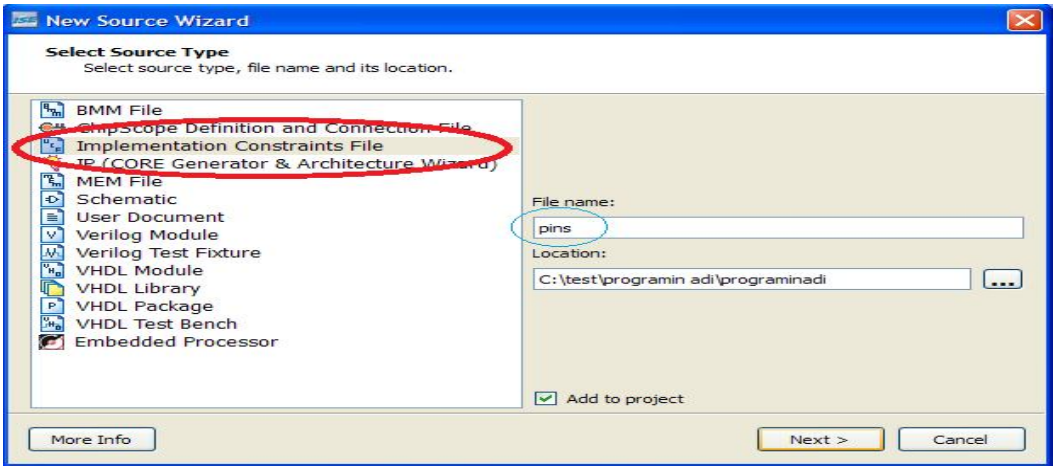
Şekil 5.8. Program Yazma Derleme-8

Böylece program analiz edilerek hata sorgulaması yapılmış olur. Alt kısımdaki console bölümünde analizin gidişatı ve sonucu gözlemlenir. Analiz işlemi sona erince, yeni bir pencerede ChipScope adındaki alt program açılacaktır. Kullanılmayacak bu pencere kapatılır. Ana ekranın sol menüsünde tekrar kullanılan FPGA seçilerek Şekil 5.9.'daki New Source'a tıklanır.



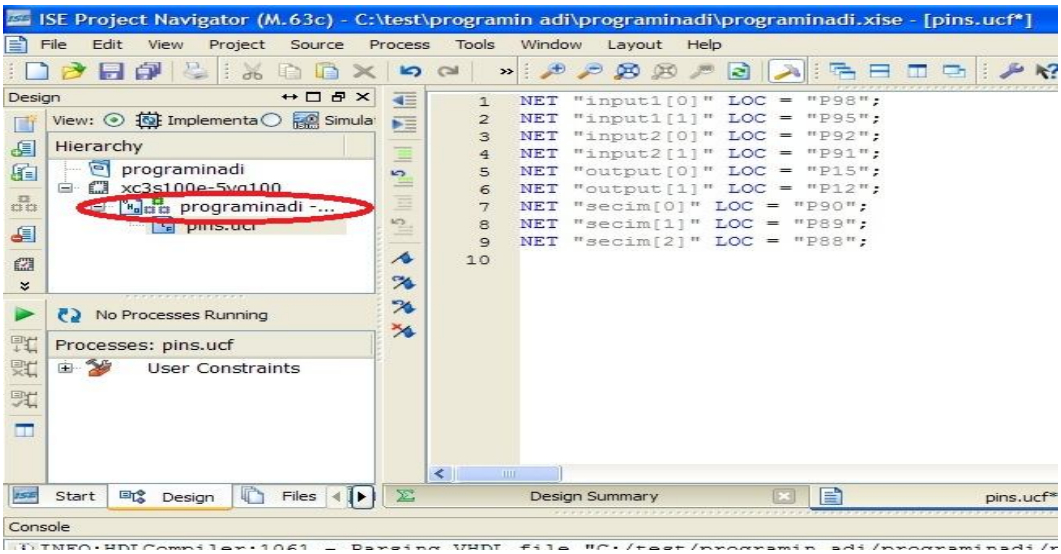
Şekil 5.9. Program Yazma Derleme-9

Pin seçimleri için Şekil 5.10.'da görülen ekranda Implementation Constraints File seçilip, File Name altına "pins" yazılır ve Next'e tıklanır.



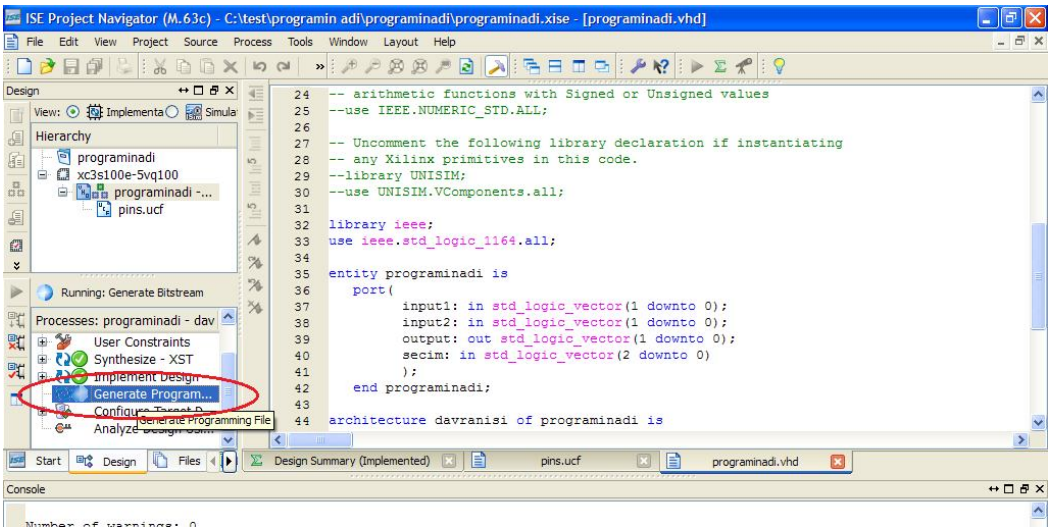
Şekil 5.10. Program Yazma Derleme-10

Şekil 5.11.'de görüldüğü üzere, açılan özet ekranında Finish'e tıklanır ve açılan boş editör sayfasına kullanılacak pinlerle ilgili kit giriş çıkış portları yazılır ve kaydedilir.



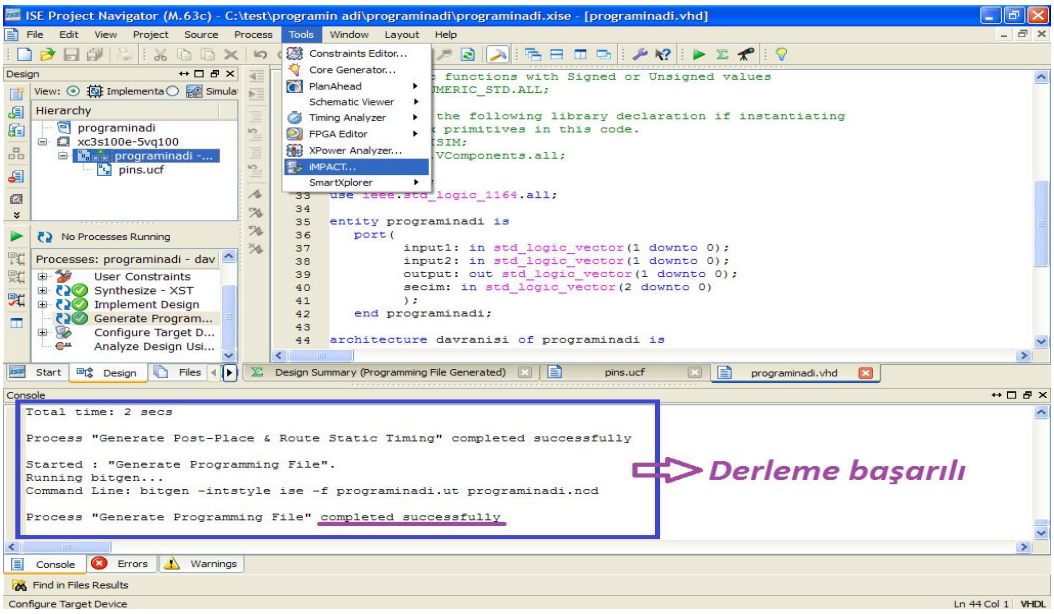
Şekil 5.11. Program Yazma Derleme-11

Programda kullanılmak istenen giriş çıkışlar, kite ait pinlerden seçilir. Bunun için Tablo 6.1.'dan yararlanılabilir. Pinler seçildikten sonra sol menüden programın adına çift tıklanır. Derleme kısmında, Şekil 5.12.'deki gibi Generate Programming File'a çift tıklanır ve derleme işlemi başlatılır.



Şekil 5.12. Program Yazma Derleme-12

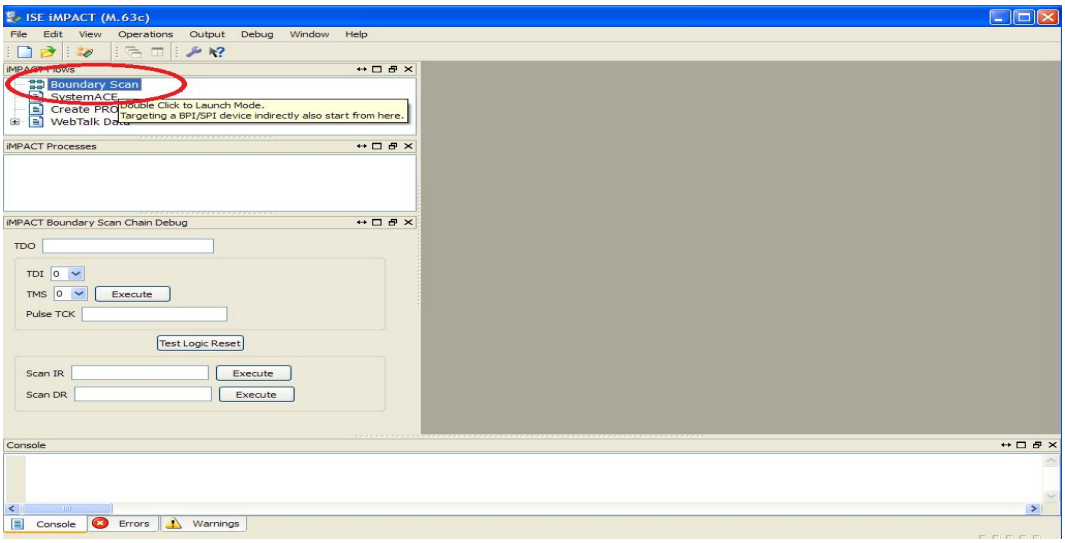
Derleme işlemi başarılı bir şekilde tamamlandıktan sonra, sırada programı FPGA'e gömme işlemi gerekmektedir. Önce Tools'tan IMPACT'e tıklanır.



Şekil 5.13. Program Yazma Derleme-13

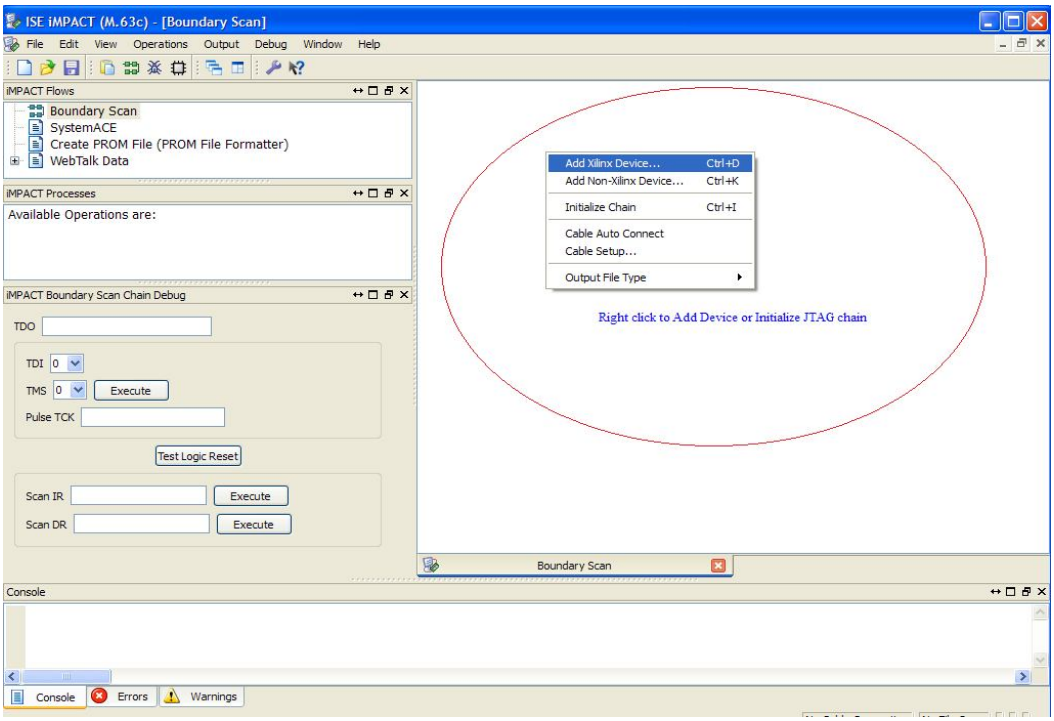
Şekil 5.13.'de çıkan uyarı penceresinde OK'a tıklanır.

ISE IMPACT, yazılan programları FPGA'ya gömme işlemi için kullanılır. Bu işlem için öncelikle bir cihaz arama işlemi yapılır. Şekil 5.14.'teki sol menüden Boundary Scan'e çift tıklanır.



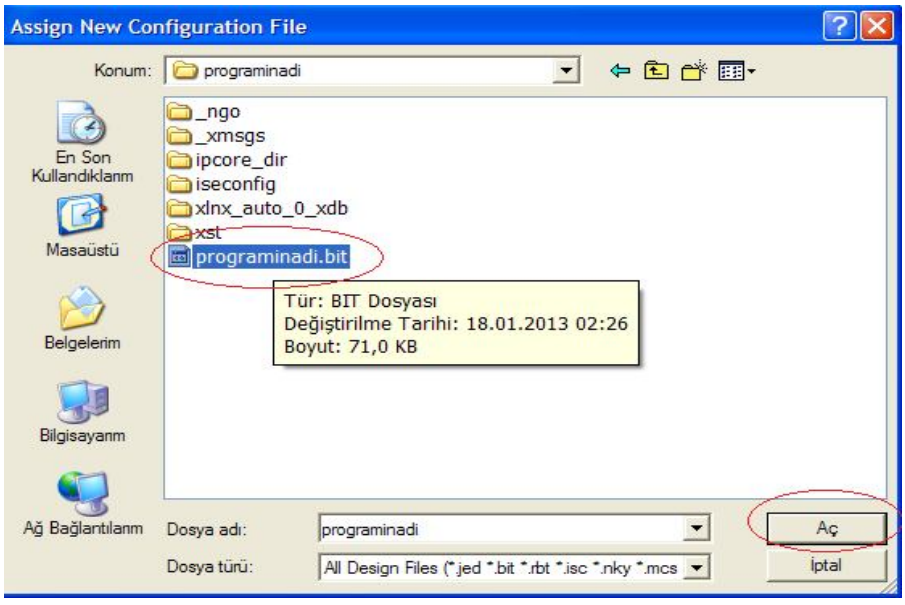
Şekil 5.14. Program Yazma Derleme-14

Arama tamamlandığında Şekil 5.15.'teki ekranda boş alana sağ tıklanıp Add Xilinx Device seçilir.



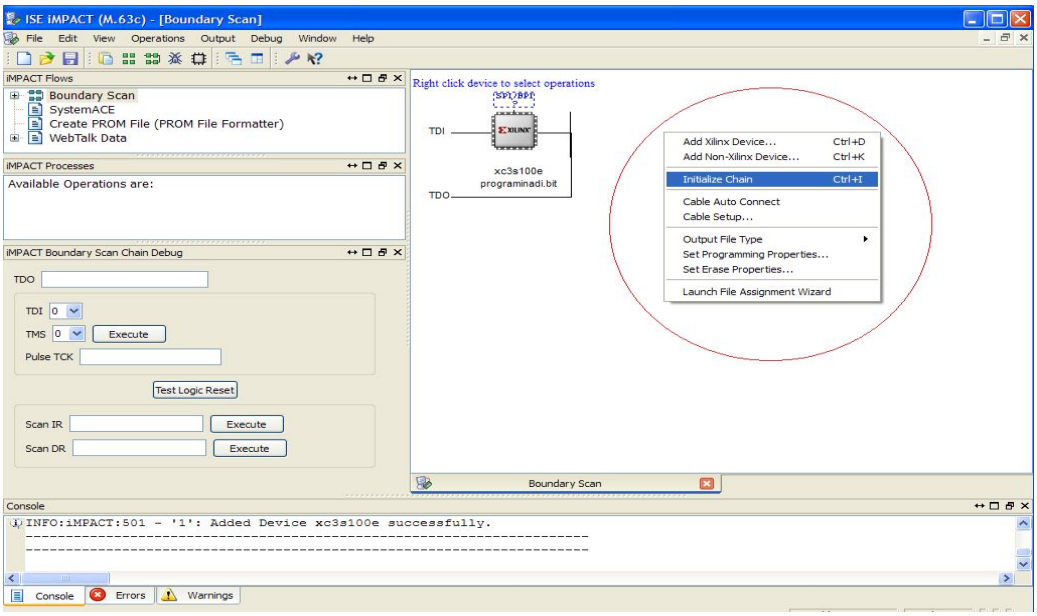
Şekil 5.15. Program Yazma Derleme-15

Daha sonrasında Şekil 5.16.'da gelecek ekranda derlenen dosya seçilip Aç'a tıklanır.



Şekil 5.16. Program Yazma Derleme-16

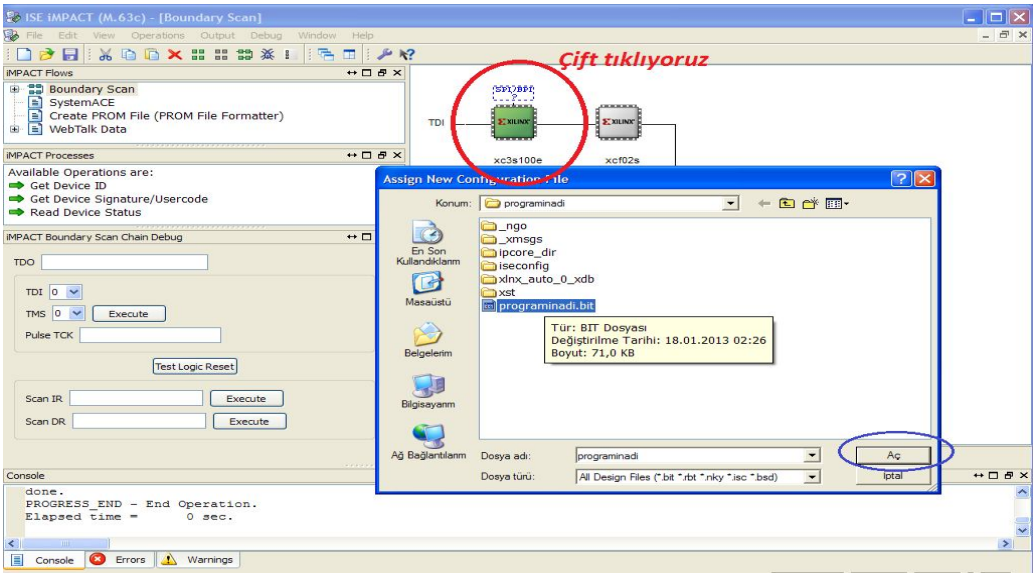
Şekil 5.17.'de boş alanda tekrar sağ tıklanarak Initiliaze Chain seçeneği seçilir.



Şekil 5.17. Program Yazma Derleme-17

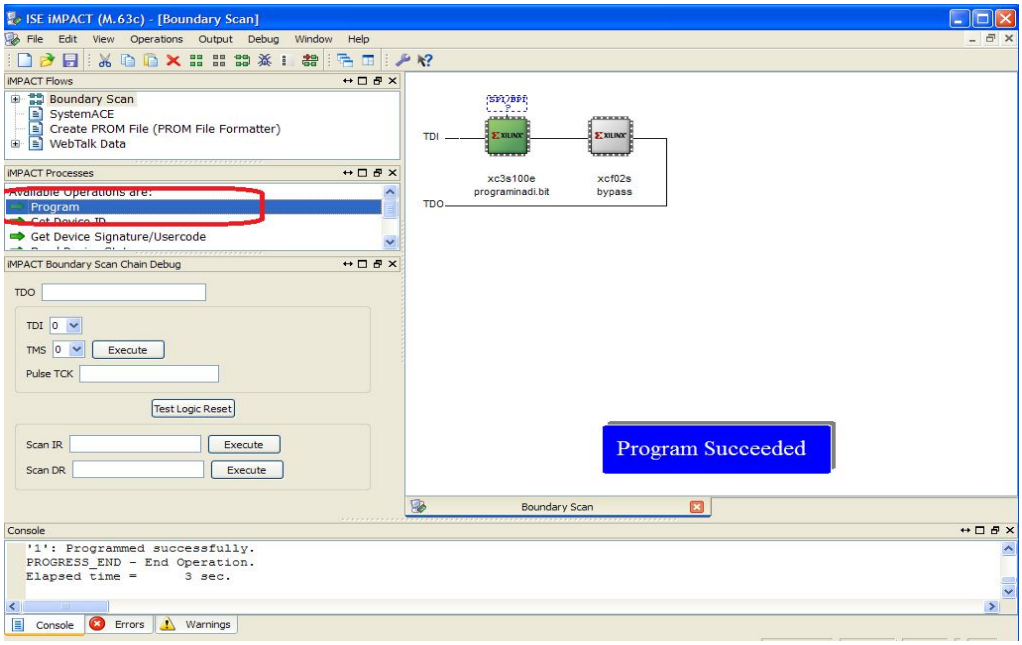
Açılan Device Programming Properties ekranı kapatılır.

Aşağıda Şekil 5.18.'de görülen kırmızı ile çevrelenmiş cihaza çift tıklanır ve derlenmiş dosya seçilip Aç'a basılır. Böylece program cihaza gönderilmeye hazır olur.



Şekil 5.18. Program Yazma Derleme-18

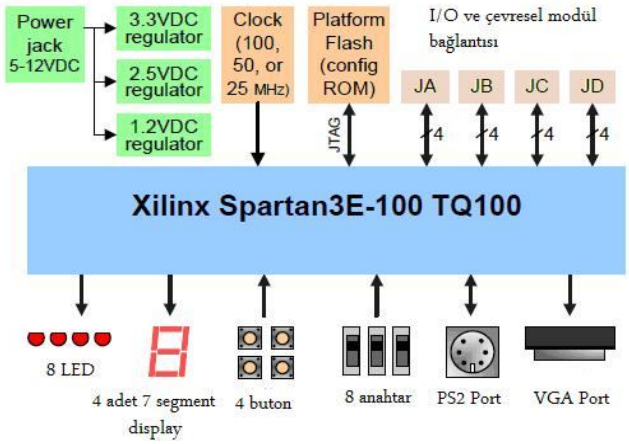
Ekrana gelen uyarıda No'ya tıklanır. Şimdi tekrar boş alandaki yeşil renkli cihaz seçilip sol menüden Program kısmına çift tıklanır. Açılan Device Programming Properties ekranı kapatılır. Ekranın altında Program Succeeded yazısı görünüyor ise program FPGA'ya doğru bir şekilde gömülmüş demektir [32].



Şekil 5.19. Program Yazma Derleme-19

6. XILINX SPARTAN 3E-100 FPGA DEVELOPMENT KIT

Piyasada birçok FPGA geliştirme kiti bulunmaktadır. Projede kullanılan FPGA geliştirme kiti Spartan 3E FPGA Geliştirme Kitidir. Bu kitin donanımları (LED, Display, VGA portu vb.) Şekil 6.1. 'de gösterilmektedir.



Şekil 6.1. Spartan 3E-100 FPGA Development Kit

Xilinx firmasına ait Spartan 3E FPGA geliştirme kartı;

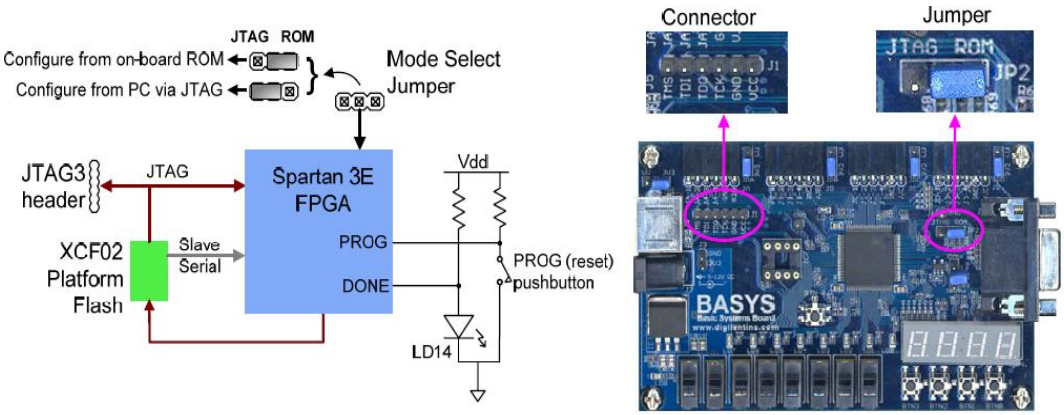
- 100000 mantık kapısı
- JTAG programlama portu
- 8 adet led
- 4 adet 7 parçalı gösterge
- 4 adet buton
- 8 adet sürgülü anahtar
- 1'er adet PS/2 ve VGA portu
- Kullanıcı tarafından seçilebilen 3 adet osilatör (25 / 50 / 100 MHZ)
- 4 adet 6 pinli giriş-çıkış portu barındırmaktadır.

Basys2 board'u (FPGA geliştirme kiti), FPGA cihazı ve modern tasarım metotları ile deneyim kazanmak isteyenler için ucuz ve kolay bir kullanım sunar. Üzerinde bulunan giriş çıkışlar ile dışarıdan herhangi bir ek donanıma gerek duyulmadan birçok devre tasarımını içeren ve bu yüzden hızlı işlem yapabilen bir geliştirme kitidir [27,36].

FPGA geliştirme kitiyle kontrol sağlamak için öncelikle programlamak gerekir. VHDL kodlarıyla yazılan program Xilinx ISE Design Suite programı ile derlenip FPGA geliştirme kitine aktarılır. FPGA kiti iki şekilde programlanır. Bunlar:

- ✓ Bilgisayara bağlıyken direkt olarak aktarılıp çalıştırılan
- ✓ Kit üzerindeki ROM'a yazılan

Tabi ki bilgisayardan direkt olarak kite atılıp çalıştırılan program, FPGA kitine enerji uygulandığı sürece çalışır. Enerji kesildiği ya da kit üzerinde bulunan "reset" butonuna basıldığı anda kit içersine yazılan program silinir. Bu bağlamda son yazılan programı sürekli hafızada tutmak için ROM'a kaydetmek gerekir. Bunun için de kit üzerinde bulunan ROM soketini JTAG konumundan ROM konumuna almak gerekir. Burada kullanılan FPGA kitinin gerçek görünümü ve giriş-çıkış bilgileri Şekil 6.2.'de gösterilmektedir.

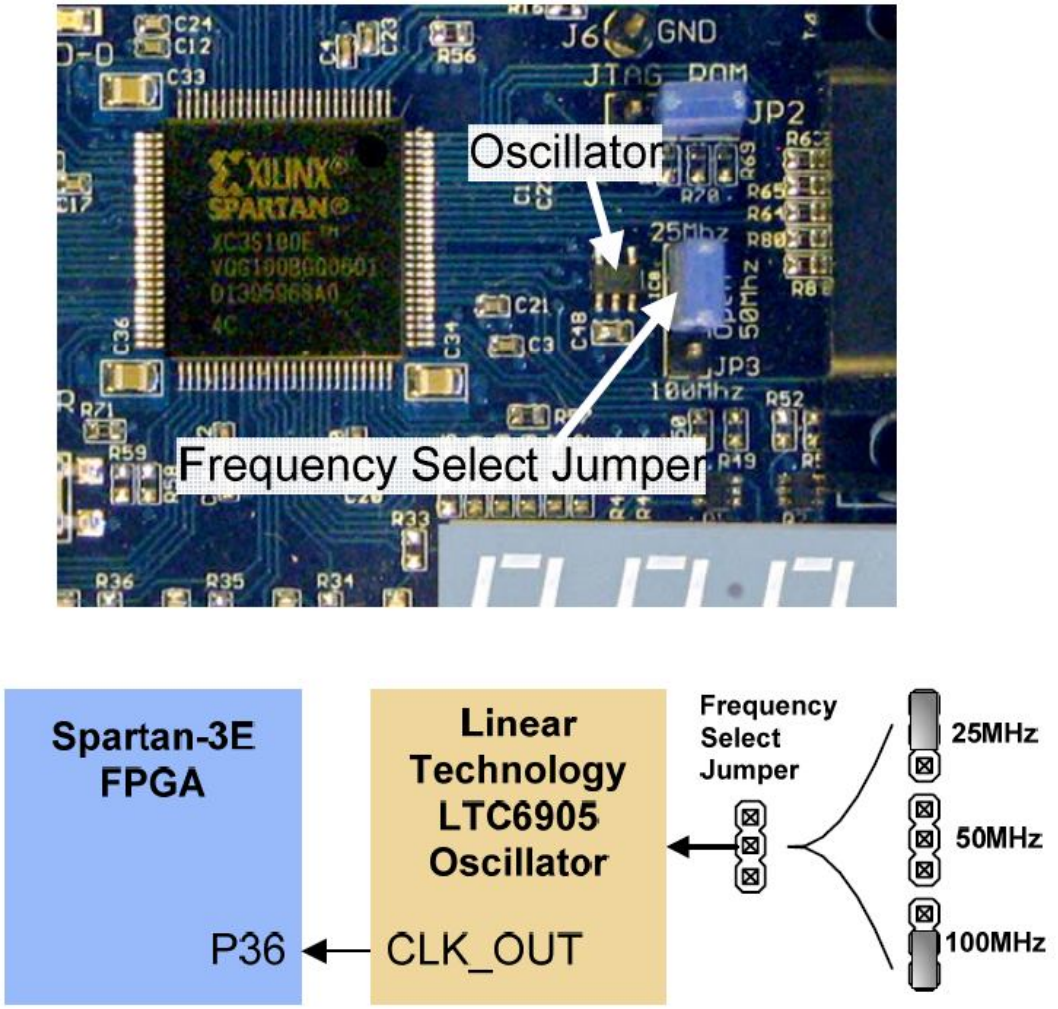


Şekil 6.2. ROM Soketi Kullanımı

Bilgisayarda yazılan programı FPGA kitine yüklemek için bilgisayar ile FPGA geliştirme kiti arasındaki bağlantı bilgisayardaki paralel porttan gelen kablo ile sağlanır. Bunun için ilk olarak paralel port kablosunun bir ucu FPGA kitine diğer ucu ise bilgisayar paralel port girişine bağlanır ve kite enerji verilir. Xilinx Ise Design Suite programı başlatılır ve FGPA kiti ile bilgisayarın birbirini tanıması için beklenir. Daha sonra bilgisayarda yazılan program direkt olarak FPGA kitine yüklenecekse klasördeki ".bit" uzantılı dosya, ROM üzerine yüklenecekse ".mcs" uzantılı dosya seçilir. Seçilen dosya ekranda bir entegre şekline dönecektir.

Bu entegre üzerine fare imleci ile sağ tıklayıp ekrana gelen seçeneklerden "program" seçeneği seçilerek program kit üzerine yada ROM üzerine yüklenmiş olur.

Osilatör seçimi kit üzerinde bulunan osilatör pinleri ile yapılır. Bunun için uygun pinlere Şekil 6.3.'de de görüldüğü gibi jumper takılır.

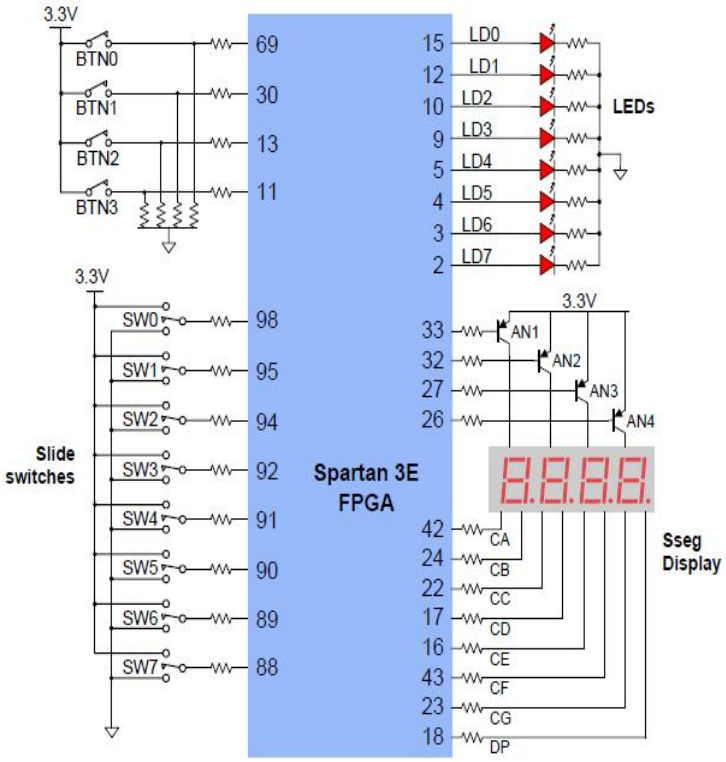


Şekil 6.3. Osilatör Seçimi

4 adet buton ve 8 adet sürgülü anahtar ile devre girişleri sağlanmaktadır. Buton girişleri normalde dijital 0 değerindedir. Butona basıldığında bu değer dijital 1 olur. Sürgülü anahtarlar pozisyonlarına göre sabit dijital 1 ya da 0 değerindedirler. Aşağı konumdayken 0 yukarı konumdayken 1 değerini alırlar. Bu buton ve sürgülü anahtarlarda kısa devreye karşı koruma için seri dirençler vardır.

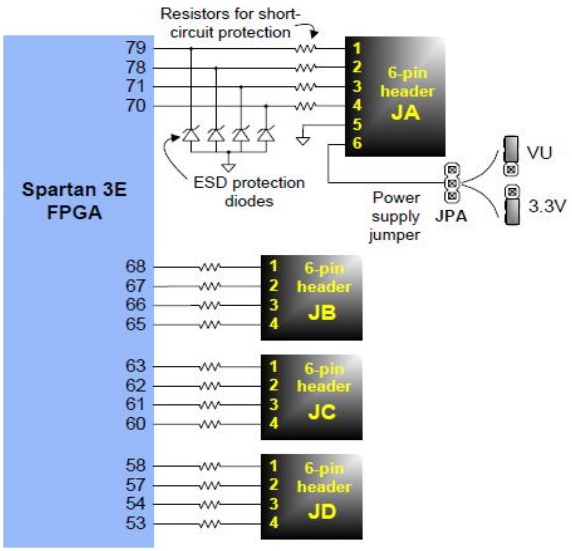
Devre çıkışları için sürgülü anahtarların karşısında 8 adet LED bulunmaktadır. Bunlar yazılan programın LED üzerinde gerçekleştirilmesi istenildiğinde kullanılır. 390 ohm dirençler ile sürülen LED'ler, dijital 1 değerini aldıklarında 3-4 mA akım ile ışık yayarlar.

FPGA üzerinde 4 adet ortak anotlu yedi parçalı LED gösterge vardır. Bu göstergelerin her biri, içerisinde 7 adet çubuk, 1 adet nokta şeklinde LED'e sahiptir. Çubuk şeklinde olanlar sönük ya da yanık durumlarına göre bir sayıyı temsil eder. Ortak anotlu gösterge olduğundan LED'lerin anotları birbirine bağlanmıştır. Bu durumda LED'in ortak olmayan diğer ucuna lojik '0' değeri verildiğinde o LED aktif olur. Tabi ki gösterge üzerinde sayıyı görmek için önce gösterge seçilmelidir. Ortak anotlu gösterge olduğu için seçilmek istenilen göstergenin anot ucuna lojik '1' verilmelidir. Ancak Spartan 3E FPGA Geliştirme Kiti'nde göstergenin ortak uçları PNP transistör ile beslendiğinden lojik '0' ile gösterge seçimi yapılır. FPGA kitinin, donanımının iç bağlantı mantığı Şekil 6.4.'de görülmektedir.



Şekil 6.4. FPGA Kitinin İç Yapısının Gösterimi

6 pinli konektörler dışarıdan FPGA'e veya FPGA'den dışarıya veri aktarımı için kullanılan portlardır. Şekil 6.5.'de konektörlerin bağlantı şeması gösterilmektedir.



Şekil 6.5. 6 Pinli Konektörler

FPGA kitinde bulunan pinlerin özellikleri Tablo 6.1.'de gösterilmektedir.

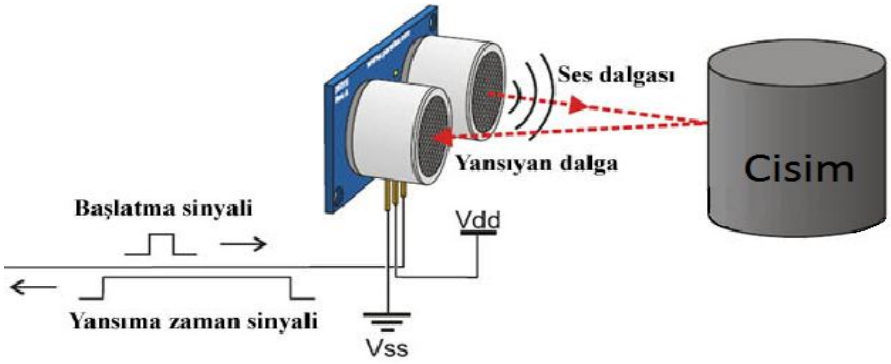
Tablo 6.1. Spartan 3E-100 FPGA PIN Tablosu

Basys2 Spartan-3E pin definitions											
Pin	**Signal**	**Pin**	**Signal**	**Pin**	**Signal**	**Pin**	**Signal**	**Pin**	**Signal**	**Pin**	**Signal**
C12	JD1	P11	SW0	N14	CC	B2	JA1	P8	MODE0	M7	GND
A13	JD2	M2	USB-DB1	N13	DP	C2	USB-WRITE	N7	MODE1	P5	GND
A12	NC	N2	USB-DB0	M13	AN2	C3	PS2D	N6	MODE2	P10	GND
B12	NC	M9	NC	M12	CG	D1	NC	N12	CCLK	P14	GND
B11	NC	N9	NC	L14	CA	D2	USB-WAIT	P13	DONE	A6	VDDO-3
C11	BTN1	M10	NC	L13	CF	L2	USB-DB4	A1	PROG	B10	VDDO-3
C6	JB1	N10	NC	F13	RED2	L1	USB-DB3	N8	DIN	E13	VDDO-3
B6	JB2	M11	LD1	F14	GRN0	M1	USB-DB2	N1	INIT	M14	VDDO-3
C5	JB3	N11	CD	D12	JD4	L3	SW1	P1	NC	P3	VDDO-3
B5	JA4	P12	CE	D13	RED1	E2	SW6	B3	GND	M8	VDDO-3
C4	NC	N3	SW7	C13	JD3	F3	SW5	A4	GND	E1	VDDO-3
B4	SW3	M6	UCLK	C14	RED0	F2	USB-ASTB	A8	GND	J2	VDDO-3
A3	JA2	P6	LD3	G12	BTN0	F1	USB-DSTB	C1	GND	A5	VDDO-2
A10	JC3	P7	LD2	K14	AN2	G1	LD7	C7	GND	E12	VDDO-2
C9	JC4	M4	BTN2	J12	AN1	G3	SW4	C10	GND	K1	VDDO-2
B9	JC2	N4	LD5	J13	BLU2	H1	USB-DB6	E3	GND	P9	VDDO-2
A9	JC1	M5	LD0	J14	HSYNC	H2	USB-DB5	E14	GND	A11	VDDO-1
B8	MCLK	N5	LD4	H13	BLU1	H3	USB-DB7	G2	GND	D3	VDDO-1
C8	RCCLK	G14	GRN2	H12	CB	B14	TMS	H14	GND	D14	VDDO-1
A7	BTN3	G13	GRN1	J3	JA3	B13	TCK-FPGA	J1	GND	K2	VDDO-1
B7	JB4	F12	AN0	K3	SW2	A2	TDO-USB	K12	GND	L12	VDDO-1
P4	LD6	K13	VSYNC	B1	PS2C	A14	TDO-S3	M3	GND	P2	VDDO-1

FPGA kiti 100 adet pine sahiptir. Bu pinler buton, anahtar, LED vb. giriş-çıkış elemanlarının kontrolü için kullanılırlar. VHDL'de yazılan program içindeki giriş-çıkışların hangi elemanlara ait olduğunu belirtmeye yarar. VHDL dilinde yazılan giriş-çıkışlar, Xilinx Ise Design Suite programının pin atamaları bölümünde pin numaralarına atanarak FPGA kontrolü sağlanır [37].

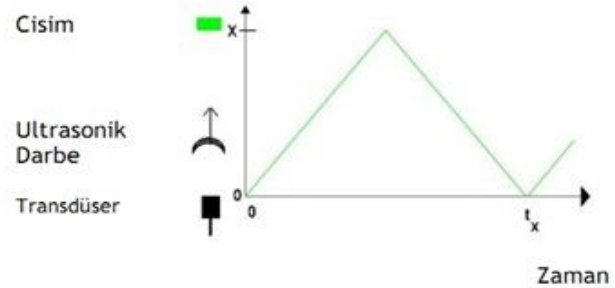
7. ULTRASONİK SENSÖRLER

Ultrasonik ses dalgaları 20 kHz ile 500 kHz arasında frekanslarda doğada bulunurlar. İnsan kulağının algılayabildiği aralık ise 20 Hz ile 20 kHz aralığındadır. Ultrasonik sensörler, 20 kHz ile 500kHz arasındaki frekanslarda ses dalgaları yayarlar. Yaydıkları bu dalgalar engellere çarparak geri yansırlar. Şekil 7.1.'de de görüldüğü gibi bu çıkan dalganın yansıyıp geri dönmesine kadar geçen sürenin hesaplanması prensibi ile aradaki uzaklığı belirlemeye yardımcı olan piezo elektronik ürünler bu sensörlerdir. Bu sensörlerde bu kadar yüksek frekanslarda ses dalgalarının yayılabilmesinin sebebi; bu frekanslardaki dalgaların düzgün doğrusal bir biçimde ilerlemeleri, genlik seviyelerinin yüksek olması ve sert yüzeylerden kolayca yansımalarıdır [38].



Şekil 7.1. Ultrasonik Sensörün Çalışması Prensibi

Transducer ultrasonik sinyali üreterek iletir. Darbe sinyali karşısına çıkan engele çarparak yansır ve sinyal transducer tarafından alınır. Darbenin gidiş geliş zamanı sensör ile engelin mesafesine göre doğru orantılıdır ki bu orantının grafiği Şekil 7.2.'de verilmiştir.

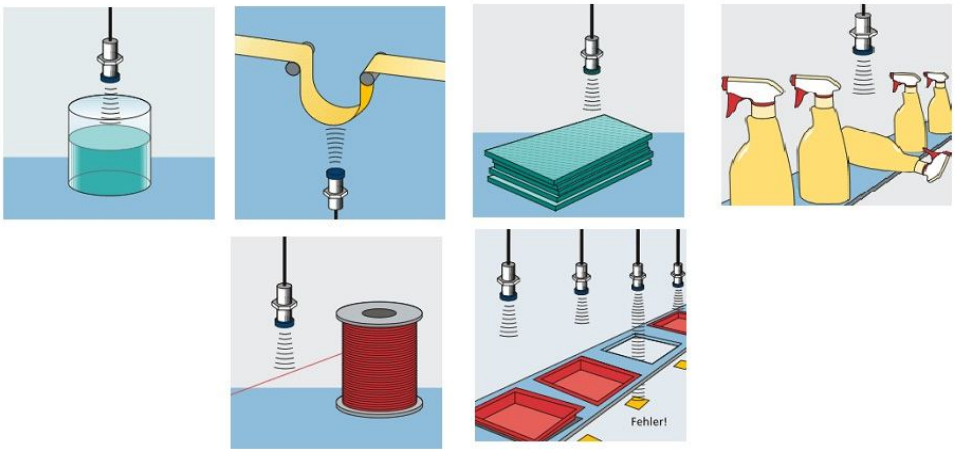


Şekil 7.2. Ultrasonik Sensörün Yol-Zaman Grafiği

Başlangıçta darbe (ultrasonik ses sinyali) iletilir, cisimden yansır, transducer tarafından algılanır ve tekrar gönderilir. Sonraki gönderilen darbe ilk darbenin ultrasonik enerjisinin hepsi absorbe edildikten sonra iletilmelidir. Bu duruma istinaden sensöre bir pals gönderilir sensör okunur ve sensörün katalogundaki yazan süre (10ms) kadar sensöre tekrar pals gönderilmez.

Katı ve sıvı cisimler ultrasonik dalgayı çok iyi oranda yansıtırlar, yaklaşık %1 kadar yansıtılamaz. Çok ufak oranlardaki enerji miktarı cisim tarafından emilir. Bundan dolayıdır ki ultrasonik sensörlerin bu özelliğinden faydalanılır ve çok çeşitli uygulamalarda sorunsuz olarak kullanabilmektedir [38].

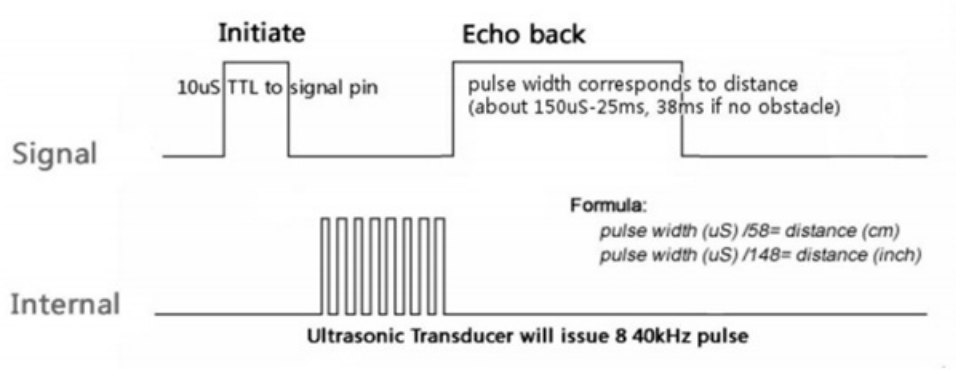
Ultrasonik sensörlerin çeşitli kullanım alanları mevcut olup, bunlardan bazıları aşağıdaki Şekil 7.3.'te verilmiştir. Ultrasonik sensör, üstteki görüntülerde; sıvı seviye kontrolü, boyut kontrolü ve nesnelerin sayılmasında kullanılmıştır. Alttaki görüntülerde ise proses hata denetimi gibi uygulamalarda kullanılmıştır.



Şekil 7.3. Ultrasonik Sensörlerin Kullanım Alanları

Bazı gelişmiş ultrasonik sensörlerin algılama menzilleri efektif koşullarda 30 metreye varabilmektedir. Ultrasonik sensörlerde iki adet transducer bulunur. Bunlardan biri ultrasonik speaker diğeri de ultrasonik mikrofondur. Elektronik devre ile ultrasonik speaker'dan ses dalgasının yayılma anı ile bu ses dalgasının engele çarpıp yansıyarak ultrasonik mikrofon tarafından algılanması arasındaki zaman ölçülür ve bu zamanın ikiye bölünüp ses hızı ile çarpılması sonucunda da engel ile ultrasonik sensör arasındaki mesafe hesaplanır. Robotlarda genellikle 40 kHz'lik ultrasonik sensörler kullanılmaktadır.

Aşağıda Şekil 7.4.'te de görüldüğü gibi sensör kendi içerisinde 40kHz frekansında bir sinyal üretip 8 adet (pulse) verici transducere gönderir. Bu ses dalgası havada, deniz seviyesinde ve 15 °C sıcaklıkta 340 m/s bir hızla yol alır. Herhangi bir engele çarparak geri sensöre yansır. Cismin sensörden uzaklığı ile doğru orantılı olarak echo pini bir süre lojik 1 seviyesinde kalır ve tekrar lojik 0 olur. Bizim bu mesafeyi ölçmek için tek yapmamız gereken echo pininin ne kadar lojik1 olduğunun süresini bulmaktır [38,39].



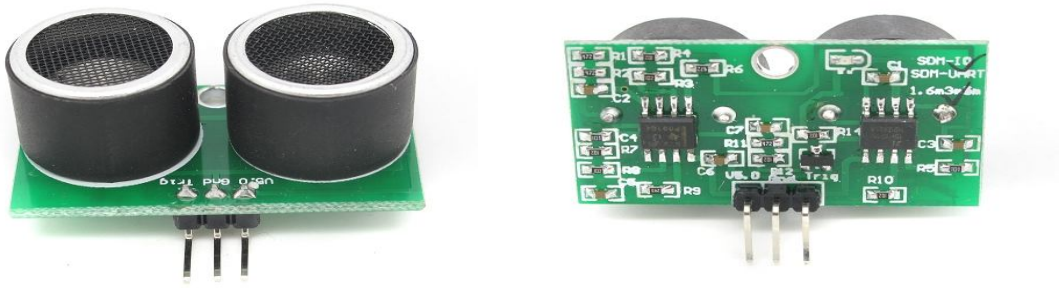
Şekil 7.4. Ultrasonik Sensör Sinyalleri

7.1. Ultrasonik Modül - SDM-IO Mesafe Sensörü

SDM-IO Mesafe Sensörü, elektronik bağlantılı temassız uzaklık ölçüm modülüdür. Şekil 7.5.'te verilen bu modül, endüstriyel uygulamalar ile kolay modüler projeler için dizayn edilmiştir. SDM-IO, kör olmayan bir alan tasarımı, hızlı cevap vermesi, üstün performansı desteklemesi ve özellikle de araç ve robot kontrolüne yatkınlığı nedeniyle tercih edilir bir modül olma özelliğindedir.

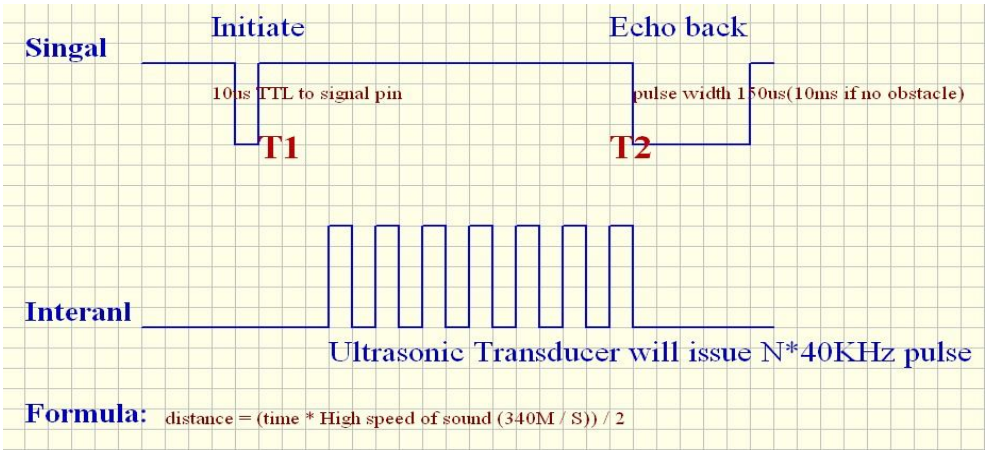
Temel karakteristik özellikleri,

- Gerilim Beslemesi:3,8- 5,5 V
- Akım Tüketimi: 8 mA
- Ultrasonik Frekans: 40 kHz
- Maksimum Uzaklık: 150 cm
- Minimum Uzaklık: 0 cm
- Çözünürlük: 3mm
- Tetikleme Darbe Genişliği: 10 µs
- Sensor Açısı: 15 derece
- Anahat Boyutları: 37x20x15 mm



Şekil 7.5. SDM-IO Mesafe Sensörü Ön ve Arka Yüzeyi

7.1.1. SDM-IO Mesafe Sensörünün Çalışma Prensibi



Şekil 7.6. Modüldeki Başlıca Sinyaller

Kısa bir ultrasonik darbe, yansıtılan bir nesne tarafından Şekil 7.6.'da da verilen T1 zamanında iletilir. Sensör bu sinyali alır ve onu elektriksel bir sinyale çevirir. Bir sonraki darbe yansıma (yankı) ortadan kaybolduğunda iletilebilir. Bu zaman aralığına devir periyodu denilebilir. Tavsiye edilen devir periyodu 10 ms'den az olmamalıdır, aksi halde modül sağlıklı olarak cevap veremez hale gelir. Eğer her 10µs'lik bir darbe tetikleme genişliği (TTL) sinyal pinine (Trig) gönderilirse, ultrasonik modül 40 kHz'lik ultrasonik sinyal çıkışı üretir ve geri yansıma (echo back) darbesini yakalar. Bu yakalanan 'echo back' darbesinin genişliği 150 µs'dir ve bundan dolayı başka bir ultrasonik modül olan HC-SR04 modülünden bu noktada ayrılmaktadır. Mesafe ölçümünde 'echo' darbe genişliği kullanılmaz. Bunun yerine aşağıdaki formülasyon ile gereken hesaplamalar yapılır [38-40].

Formül: Mesafe = (T2 – T1 – 250µs) * (340m/sn) / 2 **(7.1)**

Yukarıdaki hesaplama 250µs'lik işlem gecikmesine neden olur.

Modülün Çalışması Sırası;

- IO tetikleme sinyalleri 0'dan başlayarak TRIG sinyaline kadar sürer, en az 10 µs'dir.
- Modül otomatik olarak, geri dönen bir sinyal tespit edene kadar 8 adet 40 kHz'lik kare dalga yollar.
- Echo çıkışından geri dönen sinyale bağlı olarak, başlangıçtan geri dönüşe kadar olan zamanın ultrasonik sürecine echo denir ve bu süreçte TRIG=1 olur.

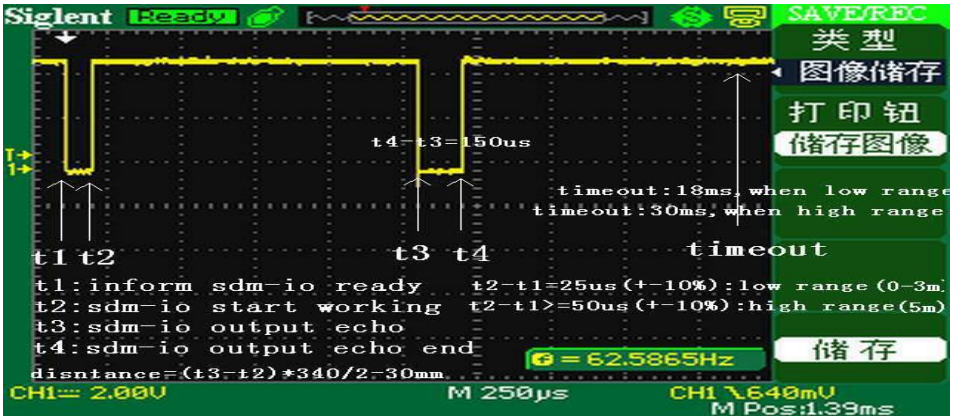
Test mesafesi = (zaman *sesin yüksek hızı (340M / S)) / 2; ifadesi ile hesaplanır.

- TRIG ucu lojik 0'dan lojik 1'e geçtiğinde ana kontrol paneli 10ms'lik zaman aşımının kontrolü için bir sayıcıyı devreye sokar. Zaman aşımı olan 10 ms'lik süreç bittikten sonra echo lojik 0 ise hala dönen sinyal yok dolayısıyla engel yok denilebilir.

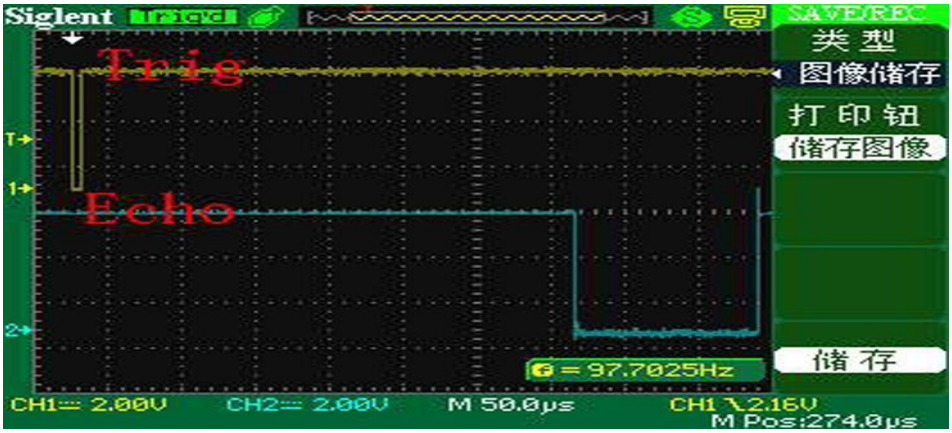
Modülün Başlıca Özellikleri;

- Ultra-minyatür, sadece alıcı ve vericiden oluşan iki adet birbirine denk yapıdan oluşmaktadır.
- Kör noktası 8 mm'lik bir üçgenimsi alandan ibarettir.
- 10 ms'lik ölçüm periyodunda hızlı cevap vermesi ve yüksek hızlı hedefleri kaybetmemesi kolay bir işlem değildir.
- Modül, test yapması ve nesne saptanmasını kolaylaştırmak için bir LED göstergeye sahiptir.

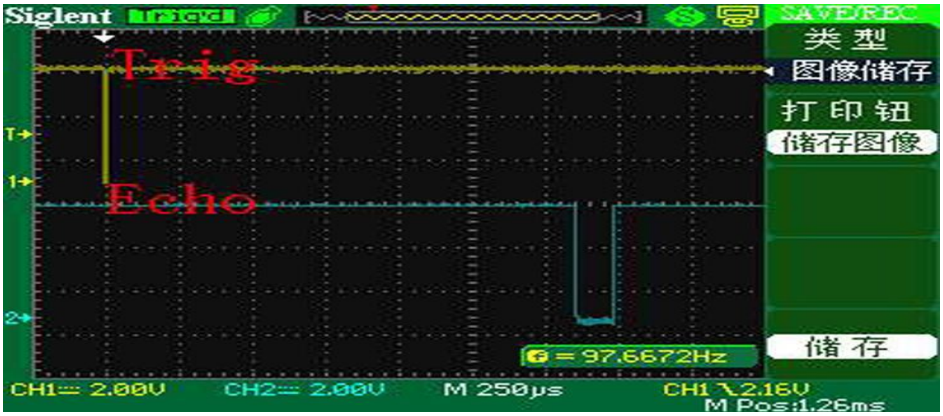
Ölçekli sinyal analizi ve ses sinyalinin aldığı kısa-uzun mesafe sinyalleri Şekil 7.7.'de, Şekil 7.8.'de ve Şekil 7.9.'da verilmiştir.



Şekil 7.7. Ölçekli Sinyal Analizi



Şekil 7.8. Kısa Mesafe Sinyalleri



Şekil 7.9. Uzun Mesafe Sinyalleri

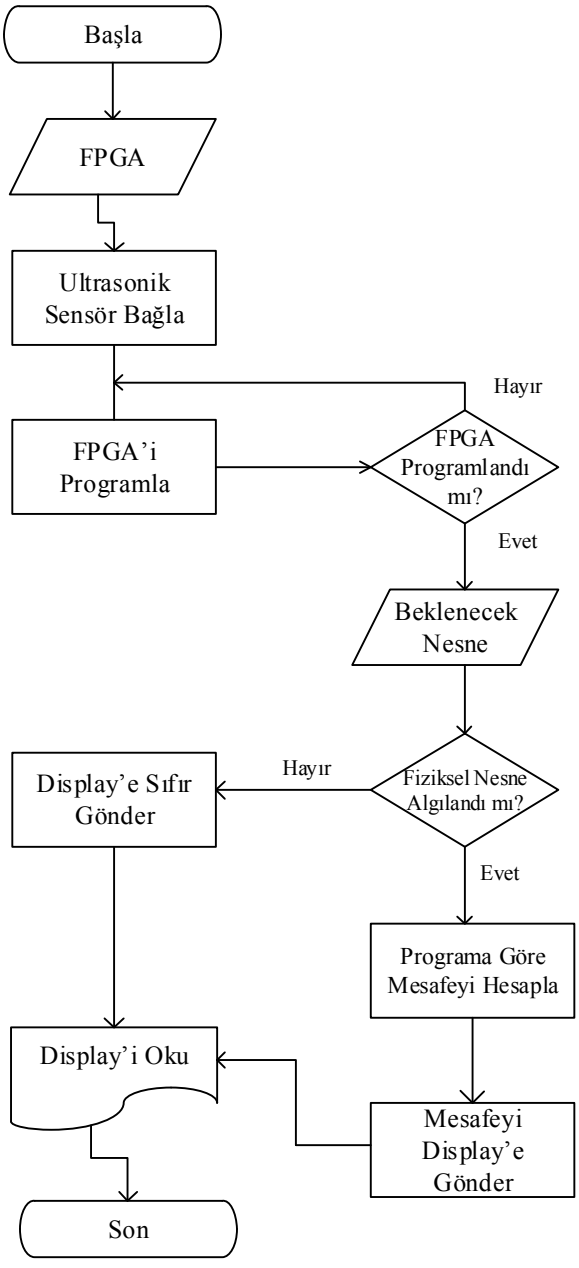
Yukarıdaki şekilleri özet olarak aşağıdaki çıkarımlarda bulunabiliriz;

- Sensör, lojik 0'da 10 µs'lik bir tetikleme darbesi ile aktif hale gelir.
- Echo pini, zaman aşımına (10ms) ya da yansıyan sinyallerin geri dönmelerine kadar lojik 1'de kalır.
- Eğer bir yankı gözlemlenirse, yankı çıkışı 150 µs kadar lojik 0 olacaktır.
- Eğer zaman aşımı meydana gelirse, Echo pini 10ms kadar daha aynı seviyede tutulur [40].

8. FPGA TABANLI NESNE ALGILAMA ALGORİTMA VE UYGULAMASI

8.1. FPGA Tabanlı Nesne Algılama Algoritması ve VHDL Kodları

Nesne tanıma algoritmalarının gerçekleştirilmesi adlı bu tez çalışmasında FPGA tabanlı nesne algılama algoritma ve uygulaması yapılmıştır. Bu gerçekleştirilen uygulamanın algoritması aşağıda Şekil 8.1.'de açıkça görülmektedir.



Şekil 8.1. Nesne Algılama Algoritması

Gerçekleştirilmesi istenen uygulamaya yönelik algoritma için ana program VHDL kodlarını ve FPGA'deki pin atama dosyasının (ucf.file) kodları aşağıda görülmektedir.

VHDL Module;

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;


entity ultrasonik is
port(
rst        :in std_logic;
clk        :in std_logic;
sig        :inout std_logic;
anot       :inout std_logic_vector (3 downto 0);
katot      :out std_logic_vector (6 downto 0)
);
end ultrasonik;

architecture Behavioral of ultrasonik is
signal okunan_deger          : std_logic_vector(19 downto 0);
signal carpim_sonucu         : unsigned(29 downto 0);
signal sayi_sonuc_bcd        : unsigned(9 downto 0);
signal sayi_sonuc_bcd_new    : unsigned(15 downto 0);

  COMPONENT ultrasonik_sensor
  port(
          rst       :in std_logic;
          clk       :in std_logic;
          gecen_zaman_deger   :OUT std_logic_vector (19 downto 0);
          sig       : inout std_logic
          );
  END COMPONENT;
```

```
  COMPONENT binarybcd
  PORT(
        b       : IN unsigned(7 downto 0);
        p       : OUT unsigned(9 downto 0)
        );
  END COMPONENT;

  COMPONENT carpma
  PORT(
clk     : IN std_logic;
        a       : IN unsigned(19 downto 0);
        p       : OUT unsigned(29 downto 0)
        );
  END COMPONENT;

  COMPONENT SevenSegmentDisplay
  PORT(
        clock           : IN std_logic;
        sayi_gelen      : IN unsigned(15 downto 0);
        anot            : INOUT std_logic_vector(3 downto 0);
        led             : OUT std_logic_vector(6 downto 0);
        dp              : OUT std_logic
        );
  END COMPONENT;

begin
  Inst_carpma: carpma PORT MAP(
        clk => clk,
        a => unsigned(okunan_deger),
        p => carpim_sonucu
  );
  Inst_binarybcd: binarybcd PORT MAP(
```

```
            b => carpim_sonucu(28 downto 21),
            p => sayi_sonuc_bcd
  );

sayi_sonuc_bcd_new<="000000"&sayi_sonuc_bcd;

Inst_SevenSegmentDisplay: SevenSegmentDisplay PORT MAP(
            clock => clk,
            sayi_gelen => sayi_sonuc_bcd_new,
            anot => anot,
            led => katot,
            dp => OPEN
            );

  Inst_ultrasonik_sensor: ultrasonik_sensor   PORT MAP(
            rst   => rst,
            clk  => clk,
            gecen_zaman_deger   => okunan_deger,
            sig=> sig
            );
end Behavioral;
```

UCF File;

```
NET"CLK"LOC="B8";
NET"rst"LOC="N3"
NET"sig"LOC="B2" ;

NET"KATOT(0)"LOC="L14";
NET"KATOT(1)"LOC="H12";
NET"KATOT(2)"LOC="N14";
NET"KATOT(3)"LOC="N11";
```

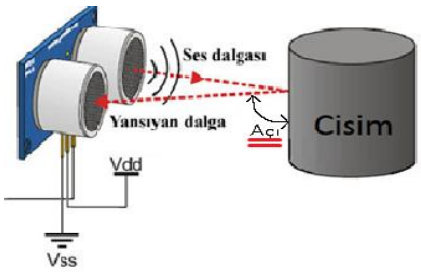
```
NET"KATOT(4)"LOC="P12";
NET"KATOT(5)"LOC="L13";
NET"KATOT(6)"LOC="M12";
NET"KATOT(7)"LOC="M13";

NET"ANOT(3)"LOC="F12";
NET"ANOT(2)"LOC="J12";
NET"ANOT(1)"LOC="M13";
NET"ANOT(0)"LOC="K14";
```

8.2. FPGA Tabanlı Nesne Algılama Uygulaması

FPGA tabanlı nesne algılama uygulamasının donanım kısmı için, FPGA olarak bir adet Xilinx Spartan 3E-100 (Basys 2) Development Kit, bir adet SDM-IO Ultrasonik Sensör ve hem FPGA'e hem de ultrasonik sensöre verilmesi gereken 5V DC gerilime gereksinim vardır.

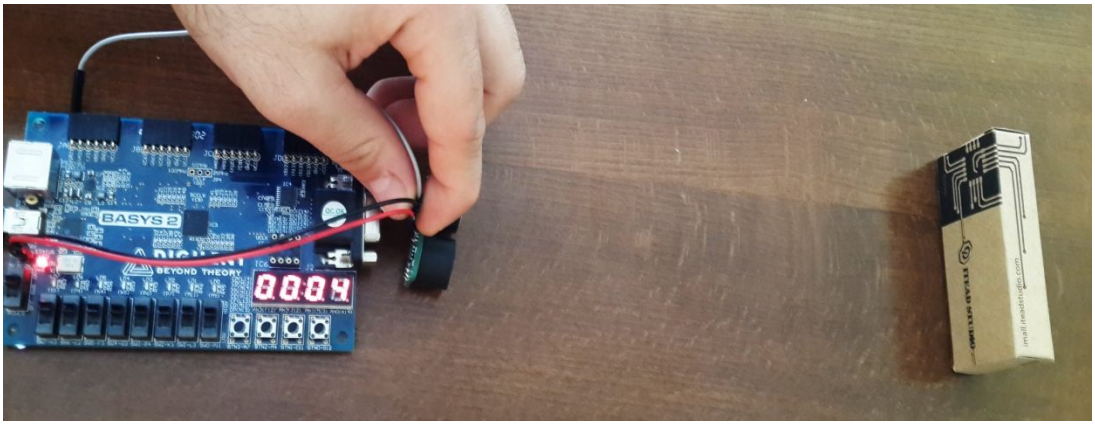
Bir bilgisayar ara yüzü olan ISE Design Suite yazılımı ile programlanmış FPGA'e jumper kablolar yardımıyla bağlı ultrasonik sensör herhangi bir nesneyi algılamak için ses sinyali göndermektedir.

Ultrasonik sensörün, konumu gereği ses sinyalini yere (yatay) paralel bir şekilde sıfır açıyla yolladığı öngörülmektedir. Bu fiziki şartlarda yollanan ses sinyalinin Şekil 8.2.'de de olduğu gibi herhangi bir yüzeye yaklaşık 97,5° açı ile çarpması ve yansımasının en doğru sonuçları verdiği tespit edilmiştir.



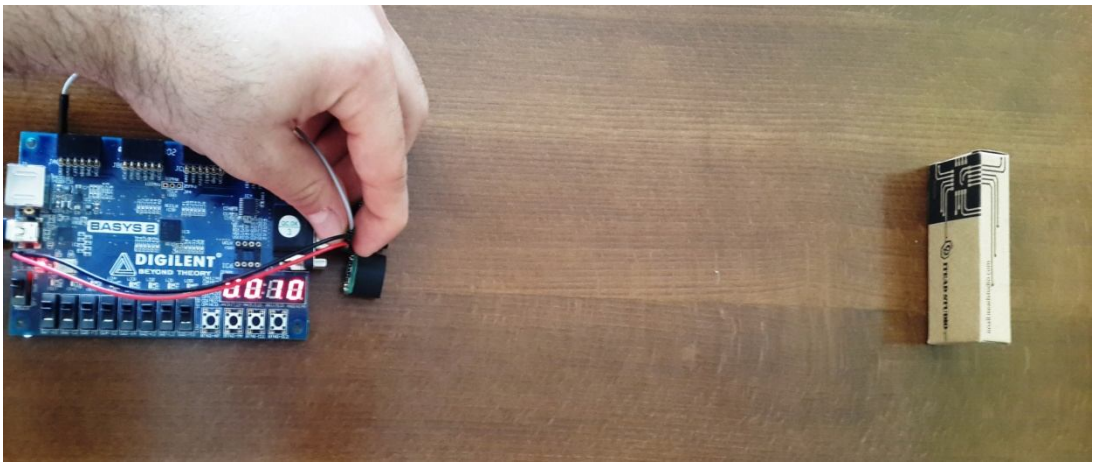
Şekil 8.2. Gönderilen ve Yansıyan Ses Sinyalinin Cisim ile Yaptığı Açı

Ultrasonik sensör sinyalinin doğası gereği ses sinyalinin gönderildiği yüzeye çarpma açısı herhangi bir konumda yaklaşık 135° ve üzerinde iken ultrasonik speakerdan yollanan sinyalin yansıdıktan sonra ultrasonik mikrofona ulaşamaması durumu gerçekleşmektedir. Bu durumda göstergede sıfır değerini okunurken diğer durumlarda ise mesafe, gerekli hesaplamalar yapılarak göstergede görüntülenmektedir.



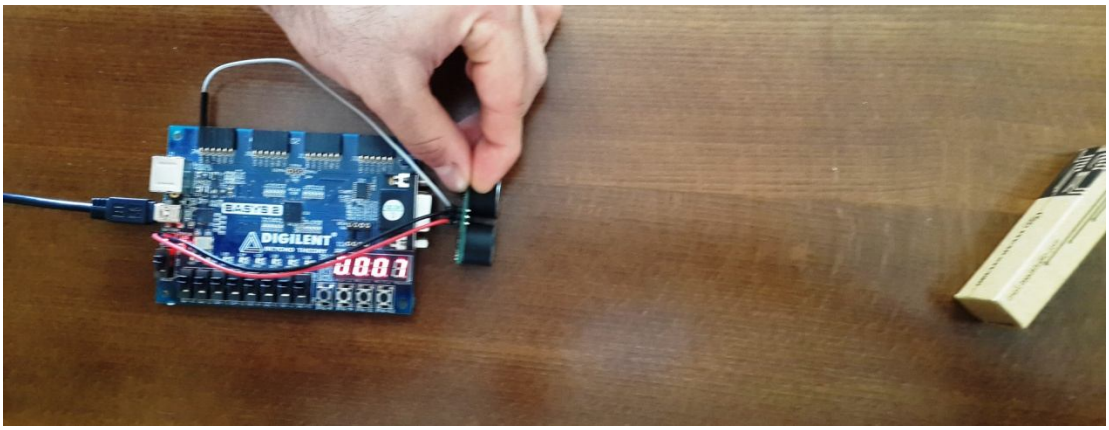
Şekil 8.3. Nesnenin Algılandığı ve Mesafenin 4 br Olduğu Durum Görüntüsü

Sistemin daha uzakta konumlandırılması ile algılanan nesnenin mesafesinin Şekil 8.3.'e göre yüksek olduğu Şekil 8.4.'de göstergede görülmektedir.



Şekil 8.4. Nesnenin Algılandığı ve Mesafenin 10 br Olduğu Durum Görüntüsü

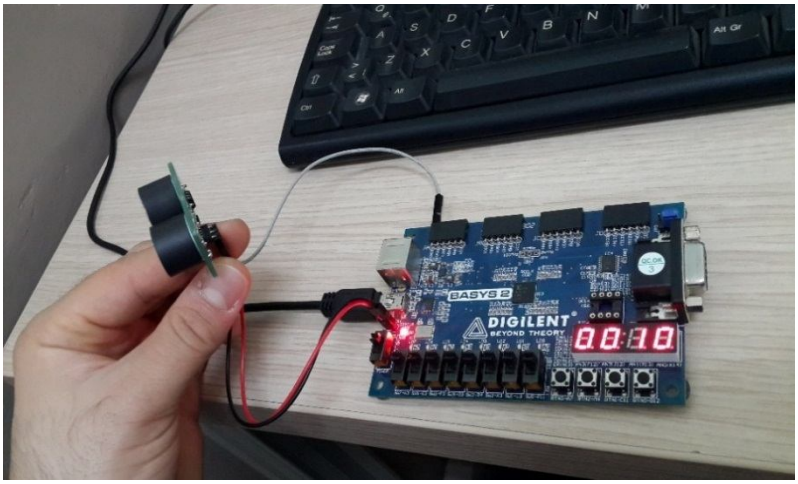
Sistemin daha yakın fakat yüzeye çarpma açısının yüksek olduğu bir konumda tutulması ile algılanan nesnenin mesafesinin cihazda okunduğu Şekil 8.5.'te görülmektedir.



Şekil 8.5. Nesnenin Algılandığı ve Mesafenin 7 br Olduğu Durum Görüntüsü

Uygulamaya ait yukarıdaki görüntülerde çeşitli uzaklıklara göre mesafelerin ölçümü gerçekleştirilmiştir. Bu üç durumda da, algılanarak mesafesi ölçülen nesnenin kutu olduğu ve ses sinyalinin bu kutunun yüzeyine algılanabilir bir açı ile çarptığı, dolayısıyla ölçümlerin güvenilir olduğu görülmektedir.

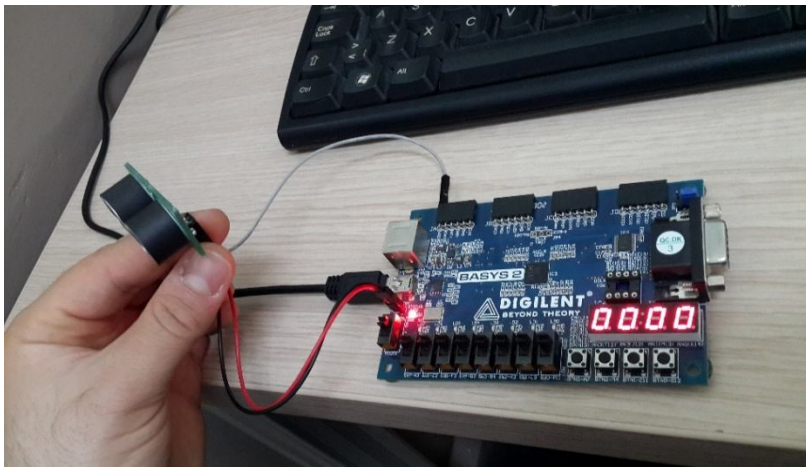
Algılanması istenen yüzeye gönderilen ses sinyalinin yüzeye çarpma açısının artırılması durumunda gerçekleşenler Şekil 8.6. ve Şekil 8.7.'deki görüntülerde gözlemlenmektedir.



Şekil 8.6. Yüzeye Çarpma Açısının Yüksek Olduğu Durum Görüntüsü

Yüzeye çarpma açısını artırıldıkça, aynı noktadan yaklaşık 97,5°'lik bir açı ile gönderilen ses sinyalindekine göre gönderilen sinyalin yüzeye çarpıp yansıması gecikmekte, dolayısıyla algılanan nesnenin mesafe de artmaktadır. Yüzeye çarpma açısını daha da artırıldığında (135° ve üzerinde), gönderilen ses sinyalinin yüzeye çarpıp geri

dönmesi mümkün olamayacağından nesne algılanmayacak ve göstergede Şekil 8.7.'deki gibi sıfır değeri görüntülenecektir.



Şekil 8.7. Nesnenin Algılanamaması Durumunun Görüntüsü

9. SONUÇ VE TARTIŞMA

FPGA (Field Programmable Gate Array – Alanda Programlanabilir Kapı Dizileri), programlanabilir mantık blokları, bu bloklar arasındaki ara bağlantılardan oluşan ve geniş uygulama alanlarına sahip olan sayısal tümleşik devrelerdir. Tasarımcının ihtiyaç duyduğu mantık işlevlerini gerçekleştirme amacına yönelik olarak üretilmiştir. Dolayısıyla her bir mantık düzenlenebilirlik özelliğinden dolayı FPGA günümüzde son derece popülerdir. FPGA'i programlamak için kullanılan donanım dilleri arasında VHDL çok karmaşık problemlerin çözümünde kolaylık sağlaması açısından sıklıkla tercih edilen bir dildir.

Çeşitli algoritmaların oluşturulup, oluşturulan algoritmaların daha da geliştirilmesi açısından VHDL terminolojisi ve tasarım akışı irdelenmiştir. Ayrıca çeşitli mantıksal işlemleri yapabilen devrelerin, gerçek bir fiziksel aygıtta uygulanabilirliğini görebilmek için VHDL kodları yazılarak bu kodlar ISE Design Suite ara yüzünde derlenerek Xilinx Spartan 3E-100 geliştirici kitine yüklenmesi gerçekleştirilmiştir.

Bu tez çalışmasında; günümüzde de kendi alanında popüler olan bu teknik dil olan VHDL ile FPGA birleştirilerek, Xilinx firmasının ürettiği ISE Design Suite ara yüzü yardımıyla sentezlenip dijital tasarımların uygulanabilirliği denenmiştir. VHDL ile birleştirilen FPGA'e bir ultrasonik sensör bağlanmıştır. Bu ultrasonik sensör; FPGA'e yüklenen VHDL yazılımı ve çalışma karakteristiği ile belirli konumlardaki nesneyi algılayan ve algıladığı nesnenin mesafesini, FPGA geliştirme kitindeki göstergeye aktaran bir sistemin parçası olma durumundadır.

Belirli konumlardaki fiziksel nesneleri algılayan bu sistem, örüntü tanıma ve algılama algoritmalarının başlangıcı niteliğinde sayılabilir. Bu sonuca varılmasının sebebi, sistemin her türlü fiziksel nesneyi algılamamasıdır. Sistemin gözü olarak tabir edeceğimiz bu ultrasonik sensörün gördüğü fiziksel nesnenin yüzeyi; sensörden yollanan ses sinyali ile maksimum limit olarak yaklaşık 135° açı yapacak şekilde konumlanmalıdır. Maksimum limit aşımındaki konumlanmalarda ultrasonik sensörden yollanan ses sinyalinin geri dönmesi mümkün olmayacağı için nesne algılanamamış ve bu algılanamama göstergeye aktarılmıştır.

Ultrasonik sensörler ile meydana getirilen bu tez çalışmasındaki sistem endüstriyel uygulamalarda, sıvı seviye algılama, akışkanlık algılamada, ahşap, kağıt, cam, şişe vb. fabrikalarda ürün algılamada ve daha bir çok uygulamada kullanılabilirler.

Kullanım alanları için; ulaşılması mümkün olmayan alanların kontrolünde, objeler veya kişiler için mevcudiyet kontrolünde, çeşitli şekillerde nesnelerin sayılmasında ve üretilmesinde, içi dolu üretilecek nesnelerin doluluğunu kontrol edilmesinde, fabrikalarda üretilen malların hata denetiminde, çok katmanlı işlemlerde gerilen materyallerin düzenlenmesinde, nakil kemerlerinin üzerindeki paketlerin seviyesinin ölçülmesinde, malların tasnif kontrolünde, taşıyıcıların üzerindeki istif yüksekliğinin ve izdüşümünün kontrolünde, hareket eden aletlerinin çarpışmasının engellenmesinde, konteynır ve siloların içindeki toplu mallar veya sıvıların seviye ölçümünde, hareket halindeki parçaların ve diğer ürünlerin aralığının ölçülmesi gibi birçok uygulama alanı sayılabilir [38].

Yukarıda görüldüğü üzere gerek sayılan gerek sayılamayan buna benzer durumlara ayak uydurabilen sistem, belirli boyutlara sahip olması önemsenen hacimsel şekillerin ortaya çıkarıldığı üretim proseslerinde kalite ve fiyat dengesi açısından ucuz bir çözüm olarak tercih edilebilir.

KAYNAKLAR

[1]. **Zeidman, B**., 2002. *Designing with FPGAs and CPLDs*, CMP Books, Kansas.

[2]. **Smith, G. R.,** 2010. *FPGAs 101*. Oxford: Elsevier.

[3]. **Zarifi, T., and Malek, M.,** 2013. FPGA implementation of image processing technique dor blood samples characterization, *Elsevier,* Islamıc Azad University, Shabestar, Iran, **40**, 1750‑1757.

[4]. **Matai, J., Irturk, A., and Kastner, R.,** 2011. Design and Implementation of an FPGA-based Real Time Face Recognition System, *IEEE International Symposium on Field-Programmable Custom Computing Machines,* Dept. Of Computer Science and Engineering, University of California, San Diego, USA, 97-100.

[5]. **Ding, Z., Zhao, F., Shu, W., Wua, M.,** 2012. Face detection system for SVGA source with hecto-scale frame rate on FPGA board, *Elsevier*, Shangai, China, **36**, 315‑323.

[6]. **Amit G., Vijay Kumar S., Mamta K.,** 2012. FPGA Based Real Time Human Hand Gesture Recognition System, *2nd International Conference on Communication*, Dr. B.R. Ambedkar National Institute of Technology Jalandhar, **6**, 98‑107.

[7]. **Husin, M., Osman, F., Sabri, M. F. M., Abidin, W. A. W. Z., Othman, K., Marzuki, A.S.W.,** 2010. Development of Shape Pattern Recognition for FPGA-Based Object Tracking System, Faculty of Engineering Universiti Malaysia Sarawak, *2010 International Conference on Computer Applications and Industrial Electronics (ICCAIE 2010), December 5-7, 2010, Kuala Lumpur, Malaysia,* 80 – 84.

[8]. **Türkoğlu, İ.,** 2003. Örüntü Tanıma Sistemleri, Ders Notları, Fırat Üniversitesi, Elazığ.

[9]. **Fukunaga, K**., 1990. *Introduction to Statistical Pattern Recognition*, Academic Press, West Lafayette.

[10]. **Güllü, K.,** 2014. Örüntü Tanıma, Ders Notları, Kocaeli Üniversitesi, Kocaeli.

[11]. Sert, E., Taşkın, D., Topçubaşı, N., Özcan, M., 2010. Görüntü İşleme Teknikleri İle Elma Tanıma, İnönü Üniversitesi, *Akademik Bilişim Konferansı*, Malatya

[12]. Theodoridis, S., Koutroumbas, K., 2006. *Pattern Recognition*, 3rd edition, Academic Press, Orlando.

[13]. Altınörs, A., Avcı, E., Biçer, Z., 2008. Sayısal Modülasyon Tanıma Sistemleri İçin Bayes Karar Kuralları Sınıflandırıcısının Kullanımı, *e-Journal of New World Sciences Academy*, **3-1**, A0056.

[14]. Türkoğlu, İ., Toraman, S., 2007. Karar Ağaçları ve Fraktal Analiz Kullanarak Histopatolojik İmgelerin Sınıflandırılması, *Gazi Üniv. Müh. Mim. Fak,*. **22,** 753-758.

[15]. Bishop, C. M., 2006. *Pattern Recognition and Machine Learning*, Springer, Singapore.

[16]. Sarıtaş, E., Karataş, S., 2013. *Her Yönüyle FPGA ve VHDL*, Palme Yayıncılık, Ankara.

[17]. Brand, W. A., 2004. *FPGAs For Dummies*, Altera, USA.

[18]. Sırmaçek, B., 2007. FPGA ile Mobil Robot İçin Öğrenme Algoritması Modellenmesi, *Yüksek Lisans Tezi*, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

[19]. Öcal, F., 2006. Güvenli iletişim için FPGA kullanarak şifreleme sistemi tasarımı ve gerçekleştirilmesi, *Yüksek Lisans Tezi*, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.

[20]. Saidani, T., Dia, D, Elhamzi, W, Atri, M and Tourki, R., 2009. Hardware Co-simulation For Video Processing Using Xilinx System Generator, *Proceedings of the World Congress on Engineering 2009 Vol I WCE 2009*, July 1 - 3, 2009, London, U.K.

[21]. Taşçı, M., 2011. FPGA Kontrollü Robotik Göz, *Yüksek Lisans Tezi*, Balıkesir Üniversitesi, Fen Bilimleri Enstitüsü, Balıkesir.

[22]. Alçin, Ö. F., 2011. Alan Programlanabilir Kapı Dizisi ile Sigma-Delta Modülatörlerin Gerçekleştirilmesi, *Yüksek Lisans Tezi*, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elazığ.

[23]. https://www.bilgiguvenligi.gov.tr/donanim-guvenligi/fpga-guvenligi.html Tübitak Bilgem.

[24]. Cofer, R.C., and Harding, B. F., 2006. *Rapid System Prototyping with FPGA's*, Elsevier Inc., Burlington.

[25]. Renovell, M., Portal, J. M., Figueras, J. and Zorian, Y., 1997. Test Pattern and Test Configuration Generation Methodology for the Logic of RAM-Based FPGA, *ATS'97, Japan*, November **199**, 254-259.

[26]. Deschamps, P. J., Sutter, G. D., & Canto, E., 2012. *Guide to FPGA Implementation of Arithmetic Functions*, Springer Science, Madrid.

[27]. Budak, C., 2011. Spartan 3E Starter Kit ile PFGA Uygulamaları, *Doktora Semineri*, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elazığ

[28]. Douglas, L. P., 1991. *VHDL*, McGraw-Hill Inc, California.

[29]. Bailey, D. G., 2011. *Design For Embedded Image Processing on FPGAs*, John Wiley & Sons Pte Ltd., Massey

[30]. Peter, J., Ashenden., 1990. *The VHDL Cookbook First Edition*: University of Adelaide, South Australia.

[31]. Enoch O. Hwang, E. O., 2006. *digital logic and microprocessor design with VHDL*, Thomson, California.

[32]. Hardi Electronics AB, VHDL Handbook (1997-2000), Lund, Sweden, 1997

[33]. Chu, P. P., 2008. *FPGA Prototyping by VHDL Examples*, Xilinx Spartan TM-3 Version, Wiley-Interscience, New Jersey.

[34]. Pedroni, V., 2008. *Electronic and Design with VHDL*, Elsevier, Burlington.

[35]. Kleitz, W., 2012. *digital electronics a practical approach with VHDL*, Pearson, New York.

[36]. http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf Spartan-3E FPGA Starter Kit Board User Guide

[37]. https://www.digilentinc.com/data/products/basys2/basys2_rm.pdf Xilinx Spartan 3E-100 User Guide

[38]. http://www.simekselektrik.com.tr/resimler/pdf/ultrasonicsensorsTR.pdf Myser Polymer Electric.

[39]. http://www.elektrikce.com/ultrasonic-sensorler-ve-kullanim-alanlari/ Ultrasonik Sensörler ve Kullanım Alanları.

[40]. http://www.biltek.tubitak.gov.tr/gelisim/elektronik/dosyalar/diger/ultrasonik.pdf Çayırpunar, Ö., Ultrasonik Uzaklık Sensörü Yapımı. Aralık 2005.

ÖZGEÇMİŞ

Doğum Tarihi	23.07.1988	
Doğum Yeri	Elazığ	
Lise	2002-2006	Balakgazi Anadolu Lisesi
Lisans	2008-2012	Fırat Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği
Yüksek Lisans	2012- Halen	Fırat Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Mühendisliği Elektronik Sistemler Anabilim Dalı

Çalıştığı Kurumlar

2012-2013	Aras Yapı Denetim LTD.ŞTİ. Test ve Kontrol Mühendisi
2013-2014	Yüzüncü Yıl Üniversitesi Mimarlık-Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği Araştırma Görevlisi
2014-Halen	Marmara Üniversitesi Teknoloji Fakültesi Elektrik-Elektronik Mühendisliği Araştırma Görevlisi