T.C.
BALIKESİR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

FPGA KONTROLLÜ ROBOTİK GÖZ

YÜKSEK LİSANS TEZİ

MUSTAFA TAŞCI

BALIKESİR, ARALIK - 2011

T.C.
BALIKESİR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

FPGA KONTROLLÜ ROBOTİK GÖZ

YÜKSEK LİSANS TEZİ

MUSTAFA TAŞCI

BALIKESİR, ARALIK - 2011

KABUL VE ONAY SAYFASI

Mustafa TAŞCI tarafından hazırlanan **"FPGA KONTROLLÜ ROBOTİK GÖZ"** adlı tez çalışmasının savunma sınavı 19.12.2011 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği / oy çokluğu ile Balıkesir Üniversitesi Fen Bilimleri Enstitüsü Elektrik Elektronik Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Danışman Yrd. Doç. Dr. Davut AKDAŞ	.....
Üye Yrd. Doç. Dr. Bayram ESEN	.....
Üye Yrd. Doç. Dr. Altuğ YAVAŞ	.....

Jüri üyeleri tarafından kabul edilmiş olan bu tez BAÜ Fen Bilimleri Enstitüsü Yönetim Kurulunca onanmıştır.

Fen Bilimleri Enstitüsü Müdürü

Doç. Dr. Hilmi NAMLI .....

Bu tez çalışması BALIKESİR ÜNİVERSİTESİ BİLİMSEL ARAŞTIRMA PROJELERİ BİRİMİ tarafından 45 nolu proje ile desteklenmiştir.

ÖZET

FPGA KONTROLLÜ ROBOTİK GÖZ
YÜKSEK LİSANS TEZİ
MUSTAFA TAŞCI
BALIKESİR ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI: YRD. DOÇ. DR. DAVUT AKDAŞ)

BALIKESİR, ARALIK - 2011

Bu tezde robotik sistemler için gerçek zamanlı bir görüntü işleme ve tanıma sistemi geliştirilmektedir. Görüntü üzerinde uygulanacak olan tüm işlemler FPGA kontrolü ile yapılacaktır. FPGA'ler paralel olarak görüntüyü aynı anda işleyebilme yetenekleri olan sistemlerdir. Bu nedenle sıralı işlemler yapan işlemci ve mikro denetleyicilere göre çok daha hızlı bir şekilde görüntü işleyebilirler.

Robotlar ve robotik sistemler için etrafındaki görüntüyü alma ve bu görüntüdeki nesnelerin algılanması büyük önem taşımaktadır. Robotlar ve robotik sistemlerin görme yeteneği, görüntüyü alma, işleme ve çıkışa aktarma hızı ile orantılıdır.

Görüntü işleme ile ilgili yapılan çalışmalar sistemlerin daha hızlı ve daha güvenilir sonuç vermesine yöneliktir.

Bu çalışma görüntü işleme teknolojisinde yeni bir çığır açan FPGA sistemleri kullanılarak yapılacak olan yeni araştırmalara temel oluşturacaktır. Tasarlamış olduğumuz sistemden elde edilen verilerle robotik uygulamalar geliştirilecektir.

ANAHTAR KELİMELER: Görüntü İşleme, Robotik Göz, Nesne Tanıma, Hedef Tanıma

ABSTRACT

FPGA CONTROLLED ROBOTIC EYE
MSC. THESIS
MUSTAFA TAŞCI
BALIKESIR UNIVERSITY INSTITUTE OF SCIENCE
ELECTRICAL AND ELECTRONICS ENGINEERING

(SUPERVISOR: ASSIST PROF. DR. DAVUT AKDAŞ)

BALIKESİR, DECEMBER 2011

In this thesis a real time image processing and recognition system is developed for robotic sytems. All the process which is to be executed on the image is carried out by FPGA . FPGAs are the systems which have the ability to process the image simultaneously . Therefore, FPGAs can process images much faster than the microprocessors and microcontrollers, which execute serial processor.

It's important for robot and robotic systems to capture images and recognise object on the images. Robot and robotic system's ability of vision is measured by its speed of capturing the image, processing and conveying it to output.

The aim is to process images faster and more reliable.

This study is going to from a bas efor the new researches which may will be using FPGA systems that will make great breaktrough in the vision processing Technologies.

KEYWORDS: Image Processing, Robotic Eye, Object Recognition, Target Recognition

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

SEMBOL LİSTESİ

V	**:** Volt
mW	**:** mili Watt
nm	**:** nanometre
s	**:** saniye
us	**:** mikro saniye
Hz	**:** Hertz
KHz	**:** Kilo Hertz
Mhz	**:** Mega Hertz
Mbps	**:** Mega bit per second (saniye başına mega bit)
Fps	**: F**rame per second (saniye başına kare)
Kbayt	**:** Kilo Bayt
Mbayt	**:** Mega Bayt
Kbit	**:** Kilo bit
Mbit	**:** Mega bit
RAM	**:** Random Access Memory (Rastgele Erişimli Bellek)
SRAM	**:** Statik RAM (Statik Rastgele Erişimli Bellek)
SDRAM	**:** Synchronous Dynamic RAM (Senkron Dinamik Rastgele Erişimli Bellek)
SD	**:** Secure Digital
CMOS	**:** Complementary Metal–Oxide–Semiconductor (Bütünleyici Metal Oksit Yarıiletken)
CCD	**:** Charge-Coupled Device (Yüklenme iliştrilimiş Araç)
LED	**:** Light Emitting Diode (ışık saçan diyod)
MUX	**:** Multiplexer (Çoklayıcı)
LUT	**:** Look-Up Table (BaŞvuru Tablosu)
LEs	**:** Logic Elements
PLL	**:** Phase-Locked Loop
RS232	**:** Recommended Standard 232
VGA	**:** Video Graphics Array (Video Grafik Dizisi)
RGB	**:** Red-Green-Blue (Kırmızı-YeŞil-Mavi)
YUV	**:** Y: Luminance U: Chrominance1 V: Chrominance2
ADC	**:** Analog Digital Converter (Analog Sayısal Dönüştürücü)
DAC	**:** Digital Analog Converter (Sayısal Analog Dönüştürücü)
CS	**:** Chip Select (Çip Seçim)
G/Ç	**:** Giriş/Çıkış
DSP	**:** Digital Signal Processing (Dijital Sinyal İşleme)
RADAR	**:** Radio Detecting And Ranging (Radyo Algılama ve Menzil Tayini)
LADAR	**:** Laser Radar (Lazer Radar)
PLA	**:** Programmable Logic Array (Programlanabilir Lojik Dizileri)
PAL	**:** Programmable Array Logic (Programlanabilir Diziler)
LCM	**:** Liquid Crystal Display Modul (Likit Kristal Ekran Modülü)
TFT	**:** Thin Film Transistor (İnce Film Transistör)
FPGA	: Field-Programmable Gate Array (Alanda Programlanabilir Kapı Dizileri)

ÖNSÖZ

Bu çalışmada Robotlar ve Robotik sistemlerde görme işlemini yerine getirmek üzere bir Robotik göz tasarlanması amaçlanmıştır. Sistem temel görme işlevlerini karşılamakta ve belirtilen nesnelerin takibini gerçekleştirmektedir.

Çalışmamız FPGA donanımı üzerinde gerçekleştirilmiştir. FPGA ile ilgili yapılan çalışmaların az olması ve FPGA'ların henüz ülkemizde yaygın olarak kullanılmaması karşılaştığımız problemlerdendir.

FPGA'ler ve programlanması konusundaki problemler, ALTERA firmasının Üniversite eğitim programları ve Çizgi-TAGEM firmasınca verilen FPGA eğitimleri ile aşılmıştır.

Bu çalışma sürecinde beni sabırla destekleyen aileme, elektronik eğitimine verdiği destekler nedeni ile Çizgi-TAGEM ekibine ve kurucusu sayın Niyazi SARAL'a, bu konuda çalışmaya beni teşvik eden ve desteklerini esirgemeyen danışman hocam Yrd. Doç. Dr. Davut AKDAŞ'a, hocalarım Yrd. Doç. Dr. Metin DEMİRTAŞ, Yrd. Doç. Dr. Ayhan İSTANBULLU ve Yrd. Doç. Dr. Bayram ESEN'e teşekkürlerimi bir borç bilirim.

1. GİRİŞ

Robotlarda ve robotik sistemlerde görme yetisi çoğu zaman yürüme, konuşma, dokunma ve diğer mekanik eylemler kadar hatta daha fazla önemlidir. İnsansı Robotlar tasarlanırken insanların yürüyüşleri konuşmaları vb. eylemleri incelenir ve benzer sistemler tasarlanır. Günümüzde duyu organlarının öğrenme sürecindeki etkisi; görme için %83, işitme için %11, koklama için %3,5, dokunma %1,5 ve tat alma için ise %1 şeklinde belirtilmektedir (1). Bu noktadan hareket ederek, görerek öğrenmenin "en etkili öğrenme biçimi" olduğunu söyleyebiliriz.

İnsanlarda görme işlevi nesnenin şekli, rengi, konumu, uzaklığı, hammaddesi, hacmi ve yaklaşık olarak ağırlığı hakkında beyine bilgiler verir.

Robot tasarımlarında genellikle insan eylemleri taklit edilir. Buradan yola çıkarak Projemizde tasarlanan sistemin görüntü işleme algoritmaları ile gördüğü nesnenin rengini belirlemesi, konumunu bulması ve kenar algılama işlevini yerine getirmesi beklenmektedir. Böylece işlenen ve sistem çıkışına aktarılan verilerin, ileride yapılacak çalışmalarda nesne ve şekil tanıma algoritmaları ile öğrenebilen robotların yapımında temel oluşturması hedeflenmektedir. Tasarlamış olduğumuz Robotik Göz sistemi ile robotların engel ve nesne algılayarak belirli ortamlarda hareket kabiliyetlerini artırabilir ve robotun görmüş olduğu nesneleri öğrenerek kendi kütüphanesinde tutabilme yeteneği de geliştirilebilir.

Literatürde gerçek zamanlı görüntü işleme, robotik göz ve hedef bulma sistemleri ile ilgili yapılan çalışmalar incelenmiştir. Sistemimizde kullanacağımız donanımı belirlemek için görüntü işlemede kullanılan donanımlar ve bu donanımların sistem gerçekleme hızları araştırılmıştır. Aynı zamanda sistemlerin kullandıkları görüntü işleme yöntemleri ve filtreler incelenerek sistemimizde kullanacağımız görüntü işleme metotları belirlenmiştir.

Rajesh Mehra, Rachna B. Sardana ve Rupinder Verma Xlinx FPGA ve DSP blokları ile FPGA tabanlı görüntü işleme uygulaması olarak etkin kenar algılama filtresi çalışması yapmışlardır. Bu çalışmada MATLAB'ta oluşturulan gri imge

bilgileri FPGA üzerinde 5x5 lik Prewitt filtresinden geçirilerek kenar algılama işlemi gerçekleştirilmiştir (2).

Ch'ng Siew SIN, Altera DE2 FPGA board üzerinde sobel filtresi kullanarak kenar algılama sistemi çalışması yapmıştır. SIN bu çalışmasında aynı resme yazılımsal ve donanımsal olarak ayrı ayrı sobel filtresi uygulayarak sistemin gerçekleme süresini hesaplamıştır. Bu test görüntü işleme uygulamalarında neden donanımsal sistemlerin tercih edilmesi gerektiği konusunda fikir vermektedir (3).

J. Hammes, A.P.W. B¨ohm, C. Ross, M. Chawathe, B. Draper ve W. Najjar ise FPGA üzerinde yüksek performanslı görüntü işleme çalışması yapmışlardır. Bu çalışma FPGA'lerde filtrelerin kullanımı ile ilgili örnek bir çalışmadır (4).

Yeni Zellanda Massey Üniversitesinde C. T. Johnston, K. T. Gribbon ve D. G. Bailey tarafından yapılan çalışmada ise FPGA üzerinde görüntü işleme algoritmaları hakkında bilgiler verilmiştir (5).

Stephen Brown and Jonathan Rose, FPGA ve CPLD'lerin prensibi ve tarihsel gelişimi konusunda bir çalışma yapmışlardır (6).

Ramazan Yeniçeri, cmos görüntü sensörü ve FPGA ile sayısal fotoğraf makinesi gerçeklenmesi konusunda bir çalışma yapmıştır (7).

Ahmet Remzi Özcan FPGA tabanlı gerçek zamanlı görüntü işleme çalışması yapmıştır. Bu çalışmada görüntü üzerinde gerçek zamanlı konvolüsyon işlemi gerçekleştirilmiş ve sistemin bilgisayarla haberleşmesinde MATLAB yazılımı kullanılmıştır (8).

Yapmış olduğumuz araştırmada görüntü işleme sistemlerinde bilgisayarlar, mikro işlemciler, mikro denetleyiciler, DSP ve FPGA gibi donanımların sıklıkla kullanılmakta olduğu görülmüştür. Bilgisayar, mikro denetleyici ve mikro işlemcili sistemler tek bir karelik görüntüde başarılı olmaktadır ancak, gerçek zamanlı peş peşe gelen görüntülerin işlenmesi uygulamalarında yetersiz kalmaktadır. Görüntünün her bir karesindeki tüm piksellerin tek tek işleme tabi tutulması sistemin yavaş ve düşük çözünülürlüklerde çalışmasına neden olmaktadır.

Sistemimizi sıralı işlem yapan bir donanım ile gerçekleştirince bir kare görüntü işleme için gereken sistem hızı şu şekilde hesaplanır;

Piksel Sayısı Tek Renk = 640x480 = 307.200 adet

Toplam Piksel Sayısı RGB = 307200 x 3 = 921.600 adet

Görüntü işleme aşamasında her piksel 20 çarpma işlemine, bir karekök alma işlemine ve birde mutlak değer işlemine tabi tutulacaktır. Bu işlemler en iyi ihtimalle 22 saat sinyalinde gerçekleştirilebilir.

Bir Kare İçin Saat Sinyali Frekansı = 921600 x 22 = 20.275.200 ≅ 20 MHz

Gerçek zamanlı bir görüntü işleme için saniyede en az 25 kare işlemek gerekmektedir. Dolayısıyla sistemin çalışması gereken minimum saat frekansı şu şekilde olmalıdır.

Sistem Saat Sinyali Frekansı = 20.275.200 x 25 = 506.880.000 ≅ 500 MHz

Sıralı işlem yapan işlemciler ile çalışırsak sadece görüntü işleyebilmek için en az 500 MHz lik bir işlemci kullanmamız gerekmektedir. Oysa tasarlamak istediğimiz sistem görüntü işleme yanında hedef algılama, hedef takip ve robotun diğer donanımları ile haberleşme yeteneğine sahip olmalıdır. Oysa projemizde de kullanacağımız FPGA sistemleri paralel komutlar işleyebilme yetisine sahip olduğundan girişindeki bir kareyi alırken, aynı anda bir kareyi işleyip, bir önceki kareyi de çıkışa aktarabilmektedir. Bu özelliği ile FPGA sistemler görüntü işleme teknolojisine büyük bir ivme kazandırmıştır. Son yıllarda yapılan görüntü işleme çalışmalarında FPGA'lar sıklıkla kullanılmaktadırlar. Ayrıca FPGA kullanarak tasarlayacağımız sistemin herhangi bir işletim sistemine ihtiyaç duymadan çalışabilmesi, küçük boyutlarından dolayı robota kolayca montaj edilebilmesi ve düşük güç tüketimi tercih sebeplerimizdendir.

Sistemimizde kullanacağımız hedef tanıma ve takip metotlarını belirlemek için yapılan yayın taramalarından bazıları aşağıda belirtilmiştir.

Thomas G. Allen, Mark R. Luettgen, Alan S. Whisky'in hareketli görüntü üzerinde hedef tespiti isimli çalışmalarında belirtilen nesnenin konum tespiti ile ilgili algoritmalar denemiştir (9).

Dimitris Manolakis, David Marden, and Gary A. Shaw isimli bilim adamları hedef tespiti için görüntü işleme uygulamaları adlı bir çalışma yapmışlardır. Bu çalışmalarında güneş ışınlarının atmosferi geçip hedeften yansıyarak uydudaki sensörlere ulaşması ile ilgili askeri çalışmalar gerçekleştirilmiştir (10).

Vladimir I. Ovod, Christopher R. Baxter, Mark A. Massie'nin FPGA taban işlemci üzerinde yüksek hızda hedef tespiti çalışması yapmışlardır. Kullanılan fonksiyonlar görüntü işleme aşamasından sonra Teta açılarının tespiti ve bu açılarla hedef yönü hakkında edinilen bilgiler bakımından faydalı bir çalışma gerçekleştirilmiştir (11).

Tezimizin içeriği FPGA kontrollü görüntü işleme ve sisteme tanımlanan hedefi takibi konusundadır.

Robotik Göz sistemi önce Altera DE2_115 FPGA platformunda D5M kamera ve VGA monitör ile gerçeklenmiş, sonra boyutları robota uygun bir platform olan DE0_NANO üzerine aktarılmıştır. Yeni tasarlanan sistemde VGA monitörün yerini LCM mini ekran almıştır. Sisteme dışarıdan müdahale için ek bir anahtar modülü tasarlanmış ve sisteme monte edilmiştir. Ayrıca sistem tarafından konumu tespit edilen nesnenin koordinatlarını sistem dışına aktaran bir çıkış modülü tasarlanmıştır.

Sistem üzerinde çeşitli görüntü işleme filtre uygulamaları yapılmış ve ihtiyacımıza cevap verecek olanlar sistem üstünde bırakılmıştır. Tasarlanan donanım ile hedef tespiti uygulamaları yapılmış ve sonuçlar incelenmiştir.

İkinci bölümde sayısal görüntü işleme hakkında genel bilgiler verilmiş, kullanılan filtreler örneklendirilmiştir. Gömülü sistemler ve FPGA'ların yapısı hakkında temel bilgiler verilmiştir.

Üçüncü bölümde tasarlanan sistemde kullanılan donanım hakkında temel bilgiler verilmiş, bu donanımları neden ve nasıl kullanıldığı konusunda açıklamalar yapılmıştır.

Dördüncü bölümde sistemi tasarlamak için kullanılan yazılımlar ve kullanım şekilleri ile ilgili temel bilgiler verilmiştir. Quartus programında kod yazımı derlenmesi, kodların FPGA'e yüklenmesi ve test programının kullanımı kısaca anlatılmıştır

Beşinci bölümde FPGA üzerinde tasarlanan sistem blokları açıklanmış, çalışma şekilleri, yaptıkları iş ve hangi kodlarla oluşturulduğu anlatılmıştır.

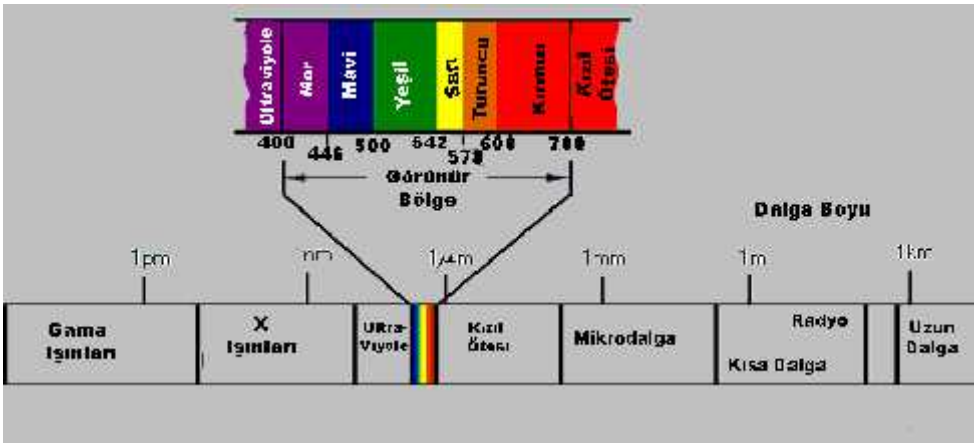
Altıncı bölümde sistemin çalıştırılması, gerçek görüntünün oluşturulması, filtrelenmiş görüntülerin çıkışa aktarılması, hedef takip modunun çalıştırılması ve çıkışının gözlenmesi kısımları anlatılmıştır.

2. GENEL BİLGİLER

2.1 Dijital Görüntü İşleme

İnsan görme sistemi gözlerimizle başlar. Işığın çok kanallı dalga boyları her biri birer algılama sistemi olan gözlerimiz yardımı ile algılanır. Görülebilen spektrum tanımı insan gözünün görebileceği elektro manyetik dalga boyu aralığını tanımlar. Buna karşın bir arının görebildiği spektral aralık ultraviyole bölgede başlar ve yeşil dalga boylarında sona erer (12).

Spektrum uzunluk ölçme birimleri ile ölçülebilen periyodik davranış sergileyen enerji dalgalarını temsil eder. Görülebilen alana ait dalga boyları 0.4μm-0.7μm arasındadır (12).



Şekil 1: Elektromanyetik Dalga Boyları

Gözlerimizle görülebilen alandaki elektromanyetik dalgaları algılar ve beynimiz yardımı ile yorumlayabiliriz. Gözün ana bileşenleri; kornea, göz bebeği, mercek, retina ve optik sinirlerdir. Kornea gözün dış kısmındadır. Geçirgen, kubbe formunda olup, ışığa odaklama fonksiyonuna sahiptir. Göz bebeği kendisini tutan kaslar yardımı ile ışık göze ulaştığında gözün açılıp kapanmasına yarar. Göz bebeği

göz merceğini örter. Kaslar yardımı ile mercek göze giren ışığın şiddetine göre kalınlaşır veya incelir.

Gözlerin farklı kontrastlara adapte olabilme yeteneği ***parlaklık adaptasyonu*** (brightness adaptation) olarak adlandırılır. İki parlaklık düzeyleri arasında ayırım yapabilme yeteneğine ise ***kontrast duyarlılığı*** adı verilir. Bu da gözün etrafını çevreleyen parlaklık düzeylerine bağlıdır. Güneşli bir günde farları yanan bir aracın farlarını görmek güçtür, fakat gece değildir (12).

Özet olarak sayısal görüntü işleme için görme sistemlerimizin altında yatan temel mekanizmaların bilinmesi oldukça önemlidir. Kısaca göz bir fotoğraf makinesi gibi düşünülebilir ve beynin görme bölümleri de karmaşık bir sayısal görüntü işleme sistemi olarak düşünülebilir (12).

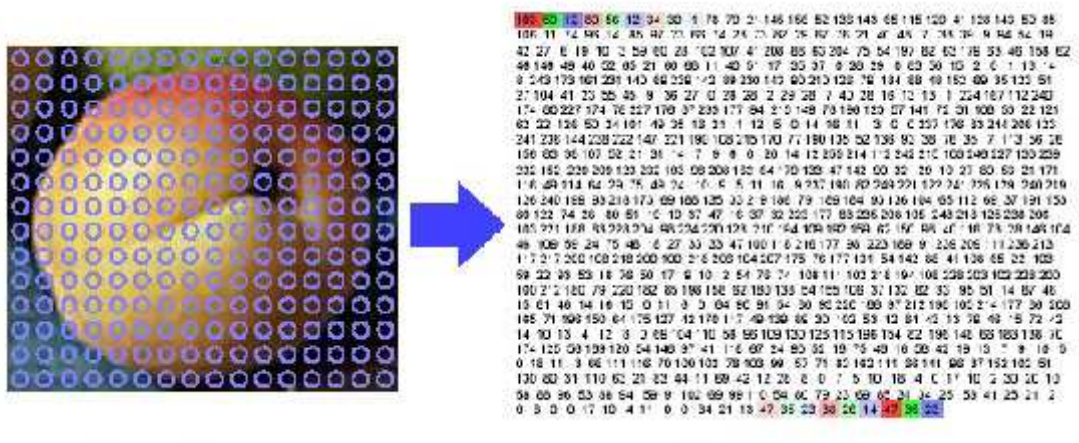
Görüntü işleme, yaşam var oldukça söz konusu olmuştur. İnsanlar ve hayvanlar gözleri ile analog temele dayanan görüntü işleme yapmaktadırlar. Bu olay beyin yardımı ile akıllı sistem (on-line), paralel ve çok spektrumlu (multispektral) olarak oluşmaktadır (13).

Resimlerin bilgisayar ortamında değerlendirilebilmeleri için veri formatlarının bilgisayar ortamına uygun hale getirilmeleri gerekmektedir. Bu dönüşüme sayısallaştırma (digitizing) adı verilir. Bir resmin sayısal forma dönüştürülmesi çeşitli şekillerde olanaklıdır. Buna farklı teknikler kullanılarak resmin sayısallaştırıldığı tarayıcılar örnek olarak verilebilir. Analog/Sayısal dönüşümün kullanılarak resmin sayısal hale dönüştürüldüğü sistemlere uzaktan algılamada uçak ya da uydulara yerleştirilen çok kanallı tarayıcılar yine örnek olarak verilebilir (14).

Sayısal bir resim denince akla analog bir sinyalin sayısal bir sinyale dönüştürülmesi gelmelidir. Bu da obje tarafından yayılan enerjinin (analog sinyal) bir algılayıcı tarafından öngörülen elektromanyetik aralıkta algılanarak sayısal sinyal haline dönüştürülmesi ile olanaklıdır.

Dijital bir resim haline getirilmiş olan gerçek yaşamdaki görüntülerin, bir girdi resim olarak işlenerek, o resmin özelliklerinin ve görüntüsünün değiştirilmesi sonucunda yeni bir resmin oluşturulmasına ***Görüntü İşleme*** denir.

Dijital görüntü sayısal değerlerden oluşur. 1 ve 0'lardan oluşan sayısal görüntü yapımız a[m,n], 2 boyutlu dünyadan elde edilen a(x,y) fonksiyonundan örnekleme tekniği kullanılarak oluşturulur.



Şekil 2: Gerçek Görüntü ve Dijital Görüntü

Sayısal görüntümüz M ve N sayılarında satır ve sütunlardan oluşur ve satır ve sütunların kesiştiği her bölgeye piksel denir. O pikseldeki değer ise derinlik (z) , renk (λ) ve zamanın (t) bir fonksiyonudur.

MxN boyutunda bir matris haline gelen görüntümüz filtrelerden geçirilerek, istenilen özellikleri belirgin hale getirilerek ya da gizlenerek yeni bir resim haline getirilir.

2.1.1 Filtreler

Filtreler görüntü zenginleştirme amacı ile de uygulanan, adından da anlaşılacağı gibi görüntüde belirli ayrıntıların ayıklanması ya da daha belirgin hale getirilmesi vb. gibi operasyonları gerçekleştiren operatörlerdir.

Farklı amaçlar için farklı filtreleme operatörleri vardır. Bunlara;

- Kenar keskinleştirme
- Kenar yakalama
- Görüntü yumuşatma

gibi daha bir çok amaçla kullanılan filtreler örnek verilebilir.

Bilindiği gibi görüntüyü oluşturan pikseller konumları ve gri değerleri ile tanımlanabilmektedir. Daha doğrusu bir görüntü matris formuna sahiptir.

Bu matrisin üzerinde şablon matris gezdirilerek her piksel değeri yeniden hesaplanır. Filtreler sayesinde girdi resminden yeni resim değişik efektler verilerek elde edilir. Filtreleme işlemi şu formülle elde edilebilir:

$$f'(x,y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} h(i,j)x\, f(x-i, y-j)$$

Burada ***h*** fonksiyonu filtreyi temsil eder.

Özel Durumlar

Filtrenin resmin kenar pikselleri için taşma durumunda kenar piksellerin aynen bir sonraki kenar varmış gibi tekrardan kenar piksel değerleri alınarak işleme dahil edilmesi ile çözülür.

2.1.1.1 Highpass Filtresi

Resimdeki az sayıda tekrar eden pikselleri elemek için kullanılır. Blur efektini azaltan filtredir.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \text{x A}$$

Şablon Matris

Şekil 3: Highpass Filtre Uygulaması

2.1.1.2 Gaussian Lowpass Filtresi

Blur efektini artırır. Resmi yumuşatır.

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \text{x A}$$

Şablon Matris

Şekil 4: Gaussian Lowpass Filtre Uygulaması

2.1.1.3 Sinc Filtresi

Kenar belirleme filtrelerinden biridir.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} xA$$

Şablon Matris

Şekil 5: Sinc Filtresi Uygulaması

2.1.1.4 Sobel Operatör Filtresi

En önemli kenar belirleme filtrelerindendir. Şekil-6 da görüldüğü gibi görüntünün yumuşak olduğu bölgelerde gradyent değeri düşüktür yani koyu bölgeler oluşmuştur. Bir gradyent görüntünün görüntülenmesi için en iyi yol yeni eşik değeri seçilmesidir. Böylelikle büyük gradyentler girilmiş olur.

$$Gx=\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} x\ A \qquad \text{ve} \qquad Gy=\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} x\ A$$

şablon matrisleri ile x ve y yönünde filtre yapıldıktan sonra,

$G = \sqrt{Gx^2 + Gy^2}$ Geometrik ortalaması alınır.

$$I(p) = \begin{cases} Imax, & G \geq d \\ 0, & other \end{cases}$$

d = Eşik değeri

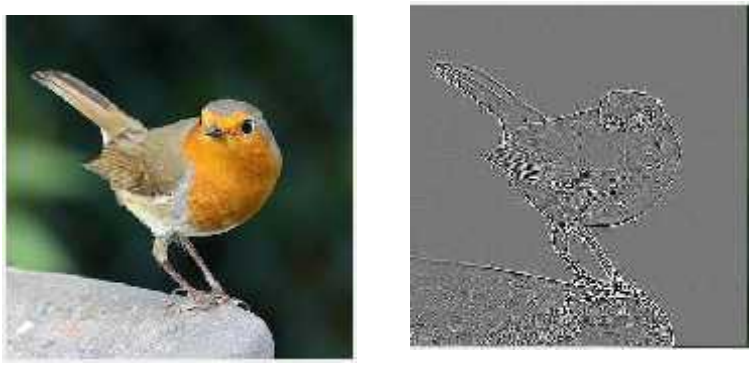
Hesaplanan değer eşik değerinden büyük ise en yüksek renk değerine, küçük ise sıfıra eşitlenir.

Şekil 6: Sobel Filtresi Uygulaması

2.1.1.5 Laplace Filtresi

Kenar belirleme filtrelerinden biridir. Sobel operatörü yatay ve düşey yönde keskinlikleri yakalamaya yararken, Laplace operatörü her yönde keskinleştirme yapmaya yarar.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{veya} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$ şablon matrisleri kullanılır.



Şekil 7: Laplace Filtresi Uygulaması

2.1.1.6 Emboss (Kabartma) Filtresi

Resme istenen yönden kabartma efekti verir.

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Doğudan Kabartma	Batıdan Kabartma	Kuzeyden Kabartma	Güneyden Kabartma

Şekil 8: Batıdan Emboss Filtresi Uygulaması

2.1.1.7 Binary İmaj Oluşturma

Gri tonları ise resimlere şu formülle dönüştürülür:

$$I_{bin}(p) = \begin{cases} 1, & Igrey \geq d \\ 0, & other \end{cases}$$

Burada d belirli bir eşik değeridir ve bu değer, çevirim için ana noktadır. Eşik (*Threshold*) noktalarının kullanılması hesaplama işlemini kolaylaştırırken bilgiyi yeterli kullanmaması ve genellikle elle girilen bir değer oluşturması bir dezavantajdır.

2.1.2 Hedef Tanıma Algoritmaları

Görüntü serileri üzerinde ilgilenilen nesnelerin efektif şekilde tespit edilmesini ve sınıflandırılmasını sağlayacak algoritmalara hedef tanıma algoritmaları denir.

Hedef tanıma çalışmaları, güvenlik sistemleri, endüstriyel uygulamalar ve savunma teknolojilerindeki gelişmelerle paralellik göstermektedir. Hedef tanıma işlemlerinde çeşitli yöntem ve algoritmalar kullanılmaktadır (15). Bu yöntemlerden bazıları şunlardır;

- Renk bileşenleri yardımıyla hedef tanıma, (15)
- Görüntünün yapısal benzerliği kullanılarak, (16)
- İki görüntünün farklı renk bileşenlerini ölçerek,(17)
- İki görüntü arasında baskın renk çifti araştırılmış ve koloni algoritmasıyla, görüntülerdeki her baskın renk çifti arasındaki mesafe ölçülerek,(18)
- İki görüntünün köşe noktaları tespit edilerek, orta noktalarının birleştirilmesi yöntemi ile görüntüler karşılaştırılmıştır.(19)

Bu çalışmada kullanılacak olan hedef tanıma algoritması ise takip edilecek olan nesnenin seçilen bir pikselinin RGB değerleri ile komşusu olan piksellerin RGB değerlerinin seçici kapılardan geçirilmesi ile yapılacaktır.

2.2 Robotik Görme

Robota dış dünyadan veri girişine duyma (sensing) ya da görme diyebiliriz. Robotlarda kullanılan tüm sensörler bu amaç için çalışırlar. Radar, ladar, sonar ve kameralardan toplanan veriler daha çok bilgi içerdiklerinden bu sinyallerin analizi robotik görmenin en popüler alanlarındandır.

Robotik görme işlemi kamera yardımıyla görüntü (image) alınması ve alınan bu görüntünün işlenerek anlamlandırılmasıyla elde edilen çıkış değeridir. Robotik görme sistemleri; CCD yada CMOS görüntü sensörleri, görüntü işlemciler, görüntü işleme yazılımları, ışık kaynakları, yansıma önleyici ekipmanlar ile geliştirilir. Bu sistemler, insanlar, konvansiyonel sistemler veya makineler tarafından gerçekleştirilen birçok uygulamada kullanılırlar. Sonucuna veya uygulama şekline göre, uygulamanın gerçekleştiği zaman çevriminin herhangi bir yerinde, hataların azalmasına, hızın ve verimin artmasına, sürekliliğe ve istikrara, giderlerin azalması, esnekliliğin ortaya çıkmasına ve daha birçok somut sonuçlara büyük katkısı vardır. Robotik görme sistemlerinin çıkışlarında analog ya da dijital olarak aşağıdaki bilgiler alınabilir;

- Koordinat bilgisi,
- Doğruluk (lojik 1-0)
- Skor,
- Kalite Kontrol,
- Registere bağlı olarak sayma işlemi, (tek veya çoklu imaj için)
- Kenar algılama,
- Renk, Kontrast bilgisi,
- Hız vs.

2.3 Gömülü Sistemler

2.3.1 Gömülü Sistem

Bir işlevin yerine getirilmesini sağlayan entegre sisteme Gömülü Sistem adı verilir. Gömülü sistemler, cep telefonu, ADSL modem, XBox, şifreli yayın alıcıları gibi günlük kullanımda olan, üretim kontrolü, kalite kontrolü sistemleri gibi endüstriyel alanda kullanılan veya uçuş kontrolü, güdüm, şifreleme, görüntüleme gibi askeri alanlarda kullanılan özel amaçlı sistemlerdir.

Genel maksatlı kişisel bilgisayar gibi sistemlerden farklı olarak, gömülü bir sistem kendisi için önceden özel olarak tanımlanmış görevleri yerine getirir. Sistem belirli bir amaca yönelik olduğu için tasarım mühendisleri ürünün boyutunu ve maliyetini azaltarak sistemi optimize edebilirler.

Gömülü sistemler genellikle büyük miktarlarda üretildiği için maliyetin düşürülmesinden elde edilecek kazanç, milyonlarca ürünün katları olarak elde edilebilir.

2.3.1.1 Gömülü Sistem Unsurları

Gömülü Sistemlerde yavaş sayılabilecek bir işlemci, bir bellek ve diğer yardımcı birimler kullanılır.

İşlemcisi olan her birim için bir de işletim sisteminden bahsetmek gerekmektedir. Sistemlerde meydana gelen gelişme, büyük oranda işletim sisteminde meydana gelen gelişme ile ilgilidir.

2.3.1.2 İlk Gömülü Sistem

İlk gömülü sistemin; MIT (Massachusetts Institute of Technology) laboratuarlarında Charles Stark Draper tarafından geliştirilen Apollo Guidance Computer olduğu bilinmektedir.

2.3.1.3 Gömülü Sistem Üretimi

İlk kitlesel gömülü sistem üretimi 1961 yılında yapılmıştır. Bundan sonra birimlerin fiyatları 1000 $ seviyesinden 3 $' a düşmüştür.

Esasında teknolojik ürünlerin çok fazla ucuzlamasının faktörlerinden biri de gömülü sistem teknolojisinde meydana gelen ucuzlama olduğu düşünülebilir.

Dünyada üretilen mikroişlemcilerin % 98'inin gömülü sistemlerde kullanıldığı ifade edilmektedir. Ancak % 2'lik bir kısım bilgisayar üretiminde kullanılmaktadır.

2.3.1.4 Gömülü Sistem Kullanımı

- Atm (Automatic Tax Machines)
- Cep Telefonları
- Network Ekipmanları
- Motor Denetleyiciler
- Abs Fren Sistemleri
- Ev Otomasyon Ürünleri
- Hava Savunma Sistemleri
- Tıbbi Ekipmanlar
- Ölçüm Sistemleri
- Ve Diğerleri

2.3.2 Programlanabilir Devreler

İlk programlanabilir aygıtlar pahalı maske programlı ROM 'lar yerine geliştirilmiştir. Bu ROM modüllerde bir değişikliğe gidilmesi gerektiğinde maske değiştirilip yeni çip'in tekrar fabrikada üretilmesi gerekiyordu. Bu özelliği nedeniyle programda yapılan bir hata ancak üretim sonucunda anlaşılmaktaydı. Daha sonra PROM modüller geliştirildi.

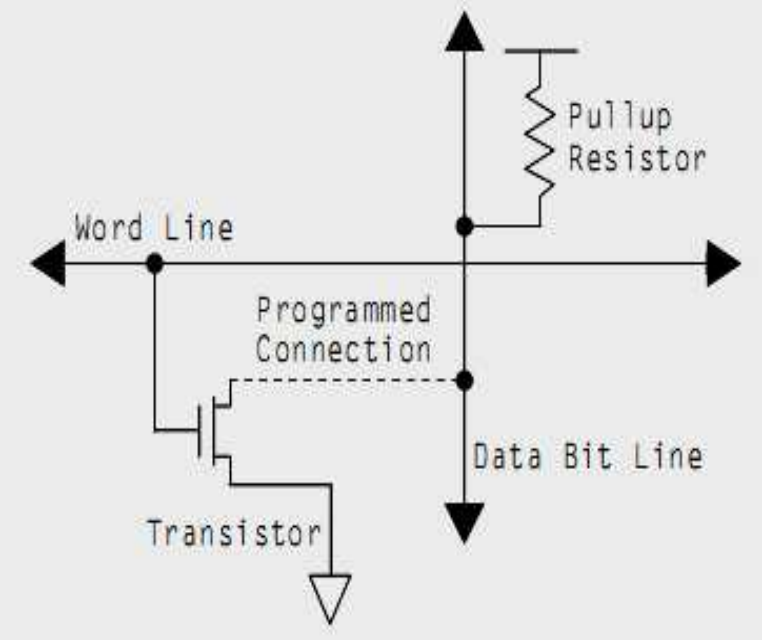
2.3.2.1 PROM

Bu modüller bir bilgisayar programı ile bir kez programlanabilmekteydi. Programlama evlerdeki sigorta sistemine benzeyen sigortalar yardımıyla yapılmaktaydı. Fazla akım geçirilen bağlantı koparılıyor geçirilmeyen kısımlar ise devreyi oluşturuyordu. PROM çipler sayesinde tasarımdaki hatalar kolayca bulunabiliyor, değişiklikler üretime gitmeden denenebiliyor ve optimizasyonlar yapılabiliyordu. Daha sonra bu çiplerin yeniden programlanabilmelerine ihtiyaç duyuldu.

Geliştirilen tekniklerle birlikte EEPROM modüller üretildi. İlk versiyonları ultraviyole ışınla silinebilmekteyken günümüzde EEPROM yani elektrikle silinebilen versiyonları kullanılmaktadır.

PROM'lar önceleri karmaşık devreleri gerçeklemek için kullanılsalar da günümüzde veri depolamada kullanılmaktadır.

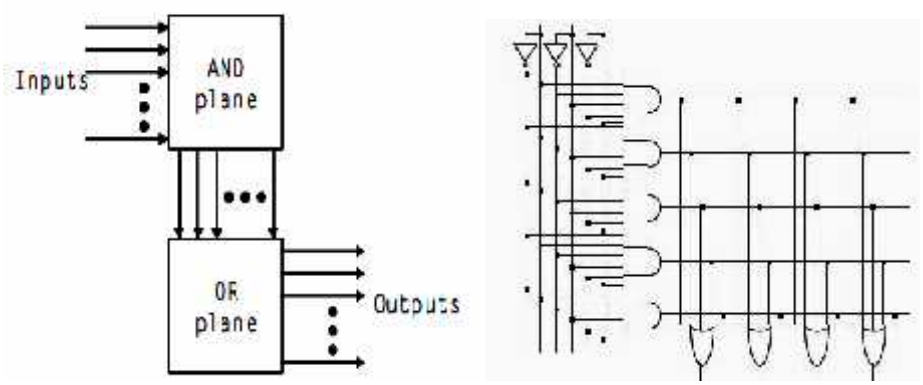
PROM'ların kullanımının bırakılmasının nedeni yeterince hızlı olmayışları ve giriş sınırlamasıdır.



Şekil 9: Prom Hücresi

2.3.2.2 PLA (Programmable Logic Array)

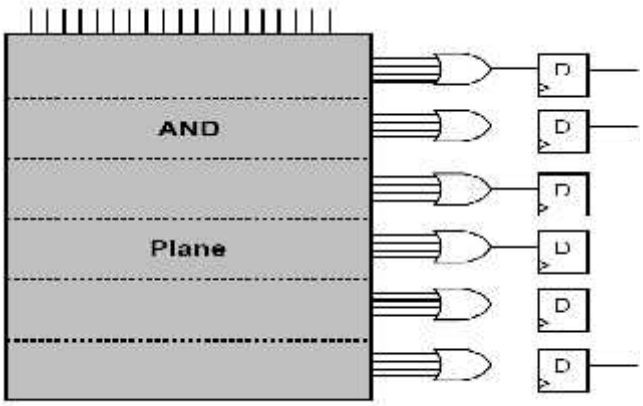
PLA (Programmable logic array) PROM'ların hız ve giriş sınırlamasına bir çözüm olarak geliştirildi. AND, OR ve NOT kapılarından oluşmaktadır. Bu nedenle PROM'lardaki gibi her olası tasarım gerçeklenemez. Tasarımın AND, OR ve NOT kapılarıyla gerçeklenmesi gerekmektedir.



Şekil 10: PLA İç Yapısı

2.3.2.3 PAL (Programmable Array Logic)

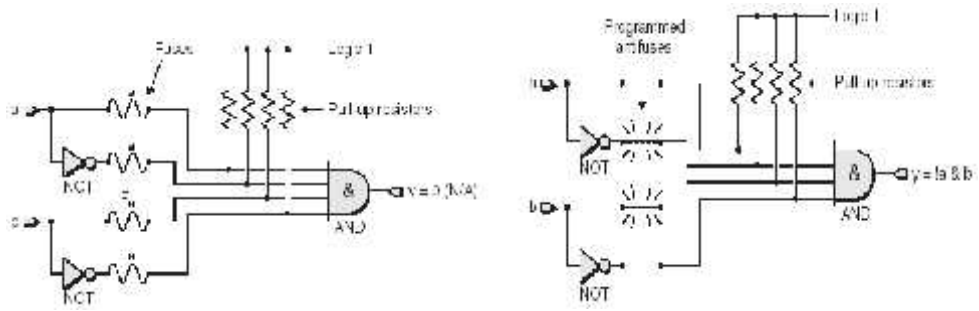
PAL çipler PLA çiplerin bir türevidir. PLA da bulunan OR kapıları azaltılmıştır. Çünkü AND ve NOT kapıları ile "de-morgan" kuralına göre OR kapısı elde edilebilmektedir. OR kapılarından boşalan alana multiplexer, XOR kapıları, Latch ve Flip-Flop yapıları konulmuştur. Böylece PLA'ya göre çok daha fazla fonksiyonu hızlı bir biçimde yerine getirebilmektedir.



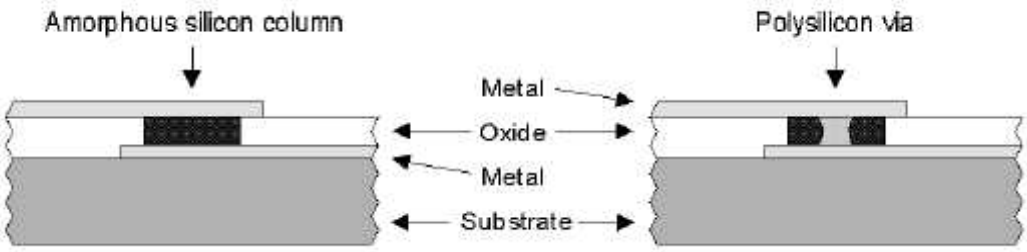
Şekil 11: PAL Şeması

2.3.2.4 SPLD (Standart Programmable Logic Device)

SPLD'ler programlanır ilk tüm olarak takdim edilirler. Eriyen sigorta teknolojileri ile bir kere programlanabilirler.



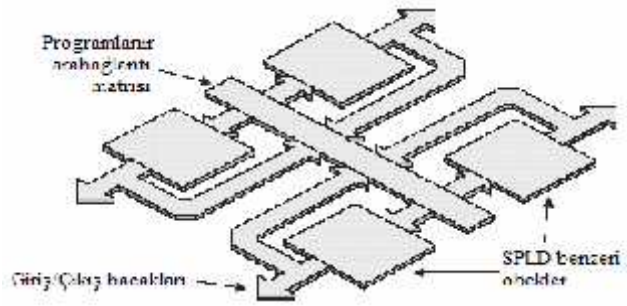
Şekil 12: Sigortaların Programlanması



Şekil 13: Eriyen Sigorta Teknolojisi

2.3.2.5 CPLD (Complex Programmable Logic Device)

Artan kapasite ihtiyacı Altera'nın EPROM ve CMOS teknolojisine dayanan CPLD'leri geliştirmesini sağladı. SPLD'ler birleştirilerek CPLD'ler meydana geldi.

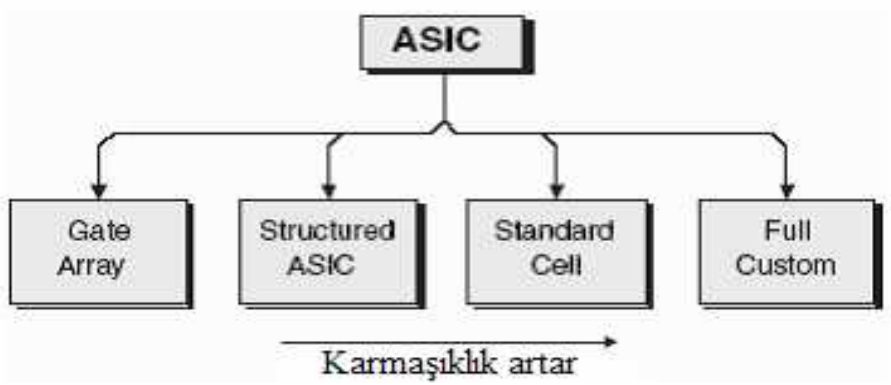


Şekil 14: CPLD İç Yapısı

2.3.2.6 ASIC (Application Specified Integrated Circuit)

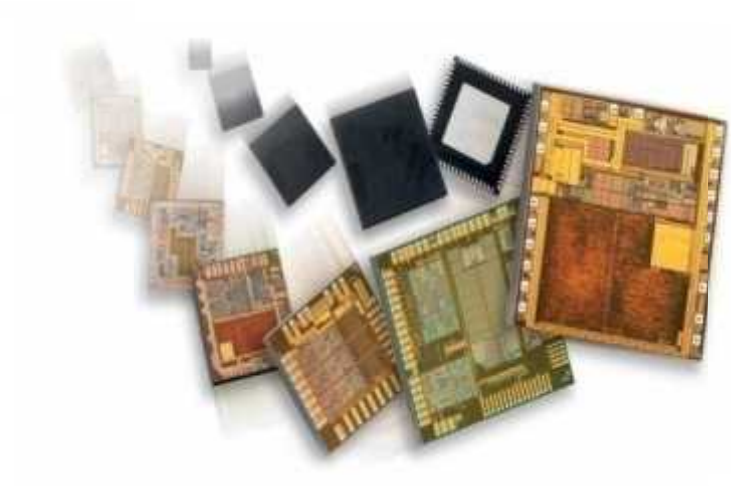
İç devreleri mekanik olarak kalıcı bir şekilde oluşturulduğundan oldukça hızlı çalışırlar. Büyük çaplı üretimlerde oldukça ucuzdur. Maske üretim yöntemiyle üretildiği için proje bazlı çalışmalarda maliyeti çok fazladır. Üretim ve test aşamaları oldukça uzun zaman gerektirmektedir.

Uygulamaya özgü tüm devreler kendi içinde dört ana grupta incelenir.



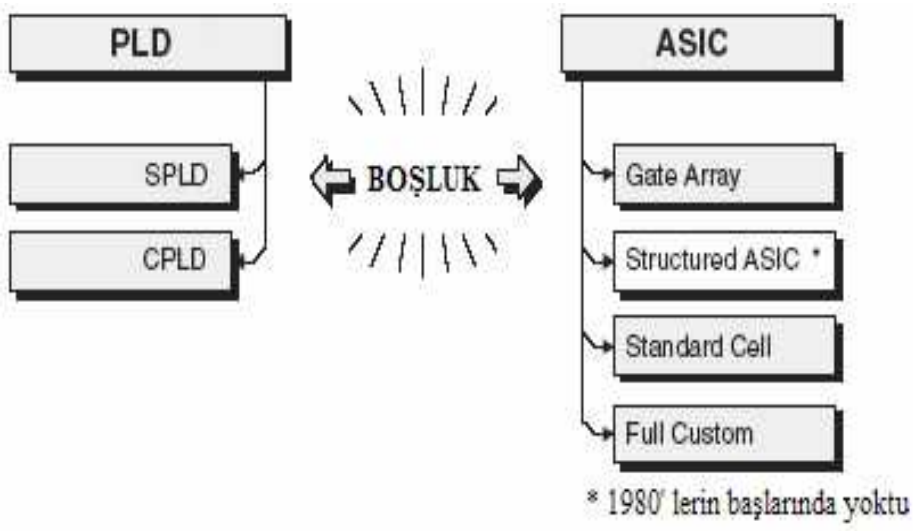
Şekil 15: ASIC Gurupları

Günümüzde üretilen ASIC çeşitlerinden bazıları şekil 16'da görülmektedir.



Şekil 16: ASIC Çeşitleri

2.3.2.7 FPGA İhtiyacı Nasıl Oluştu?



Şekil 17: ASIC ve gömülü sistemler karşılaştırma

- Yüksek yapılandırma
- Hızlı tasarım
- Değişiklik imkanı
- Geniş ve karmaşık tasarımları desteklemez
- Çok geniş ve karmaşık işlevleri destekler
- Oldukça pahalı
- Uzun süreç
- Tasarımın geri dönüşü yok

İki teknoloji arasında oluşan boşluğu doldurmak amacıyla FPGA'lar geliştirildi.

2.3.2.8 FPGA (Field Programmable Gate Array)

FPGA alanda programlanabilir kapı dizileri olarak adlandırılırlar. Çok genel bir tanımlamayla FPGA, dijital tasarımlarda kullandığımız farklı lojik kapıların milyonlarcasının tek bir entegre üzerinde toplanmasıdır.

Bu lojik kapılar istenildiği gibi programlanabilir ve istenilen her türlü dijital tasarım bu elemanlar üzerinde gerçeklenebilir.

**Sistemin işleyişinin sonucu böyle yorumlanabilir, aslında var olan şey hafıza birimleri, multiplexer'lar ve flip flop'lardır*

2.3.2.9 Neden FPGA?

FPGA'ler bize paralel işlem kabiliyeti sunan ve içyapısını istediğimiz fonksiyon ve uygulamaya göre programlayarak değiştirebildiğimiz donanımlardır.

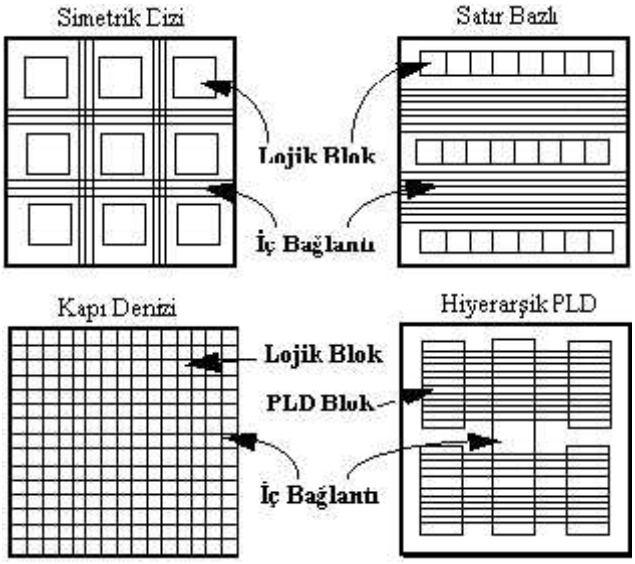
Örneğin gerçek zamanlı yüksek çözünürlüklü bir video görüntüsü üzerinde filtreleme işlemi yapmak istiyoruz. Video, peş peşe sıralanan resimlerdir ve bu resimlerin her birine "frame" denilmektedir. En basit haliyle bu işlem için videonun bir resim karesini giriş portlarından almamız, onu filtrelememiz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

2.3.2.10 FPGA Donanımı

Tasarımcı tarafından düzenlenebilecek üç ana bölümü olduğu düşünülebilir:

- Düzenlenebilir Mantık Blokları (*Configurable Logic Block*) – CLB kullanılacak mantık fonksiyonel birimleridir.
- Giriş / Çıkış Blokları (*Input / Output Blocks*) – IOB entegre devrenin paket bacakları ile iç bağlantılar arasındaki ilişkiyi oluşturur.
- İç Bağlantılar (*Interconnects*) CLB ve IOB'ler arasındaki giriş çıkışları sağlar.

2.3.2.11 FPGA Bağlantı Çeşitleri



Şekil 18: FPGA Bağlantı Çeşitleri

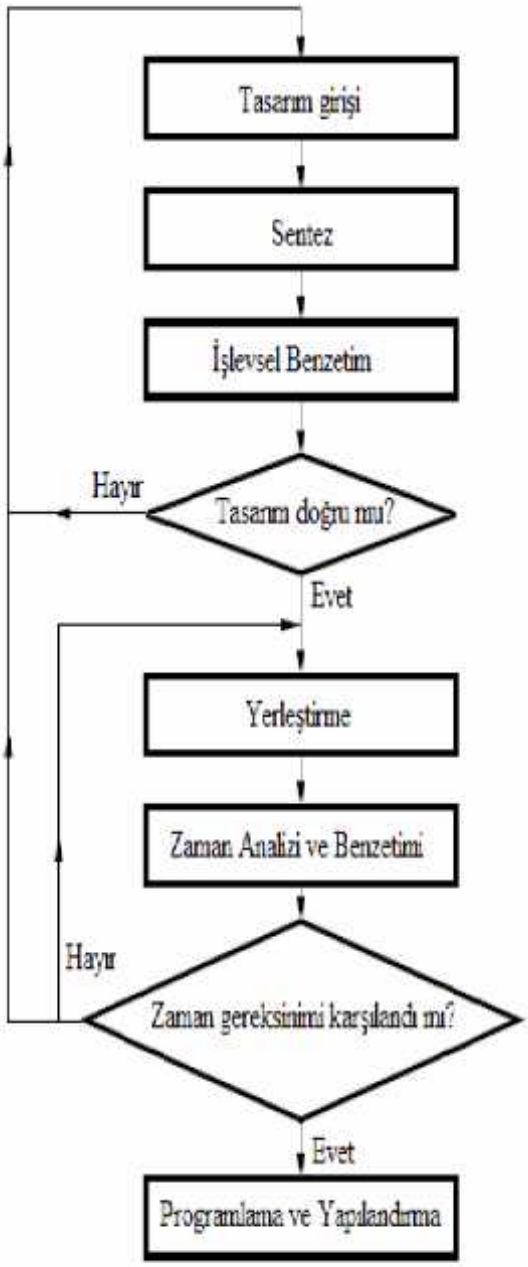
Bir formülün sonuçlarını bulmak için bilgisayarın her defasında bir hesaplama yapması yerine, bir defa hesaplanan sonuçları bir tablo halinde saklaması ve gerektiğinde hesaplama yapmadan, giriş adreslerine karşı gelen sonucun tablodan okunması yöntemine "Look – Up Table" yöntemi denir. Bu yöntem FPGA'ların çalışma hızlarını artırmaktadır.

2.3.2.12 FPGA Yapılandırılması

- "configuration file" veya "bit file" olarak isimlendirilen dosya yapılması gerekilen işlevi FPGA içinde yerine getirir.
- Yapılandırma dosyası üretici firmaların sunduğu ürün geliştirme programları tarafından üretilir.
- SRAM tabanlı FPGA'lardaki yapılandırma dosyası, yapılandırma verisi ve yapılandırma komutlarını içerir. Yapılandırma verisi, doğrudan programlanır mantık öğelerinin durumunu belirlemek için kullanılır. Yapılandırma komutları ise aygıta yapılandırma verileri ile ne yapacağını söyler.

2.3.2.13 FPGA Akış Şeması

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



Şekil 19: FPGA Akış Şeması

3. SİSTEM DONANIMI

3.1 CMOS ve CCD Kameralar

Görüntü sensörleri üzerlerine optik bir mercek yardımıyla düşürülen görüntüyü işleyerek elektriksel sinyallere dönüştüren devrelerdir. Günümüzde dijital kameralar ve diğer görüntüleme aygıtlarında bu sensörler kullanılmaktadır. Görüntü sensörleri temel yapıları itibariyle CMOS ve CCD sensörler olarak ikiye ayrılmaktadır.

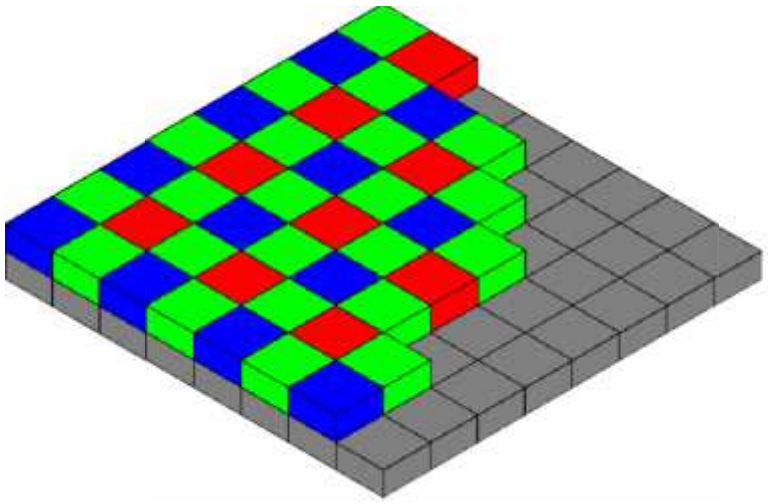
Şekil 20: CMOS Kamera

Şekil 21: CCD Kamera

CCD görüntü sensörleri bir tabakanın üstüne dizilmiş ışığa duyarlı foto diyotlardan oluşmaktadır ve bu diyotlar ışığı, düşen ışığın şiddeti oranında elektrik gerilimine çevirirler. Bu sensörler ışığa karşı CMOS sensörlerden daha duyarlıdırlar ve üretilen görüntünün niteliği daha kalitelidir. Bunun yanında daha pahalıdırlar ve daha fazla güç harcarlar.

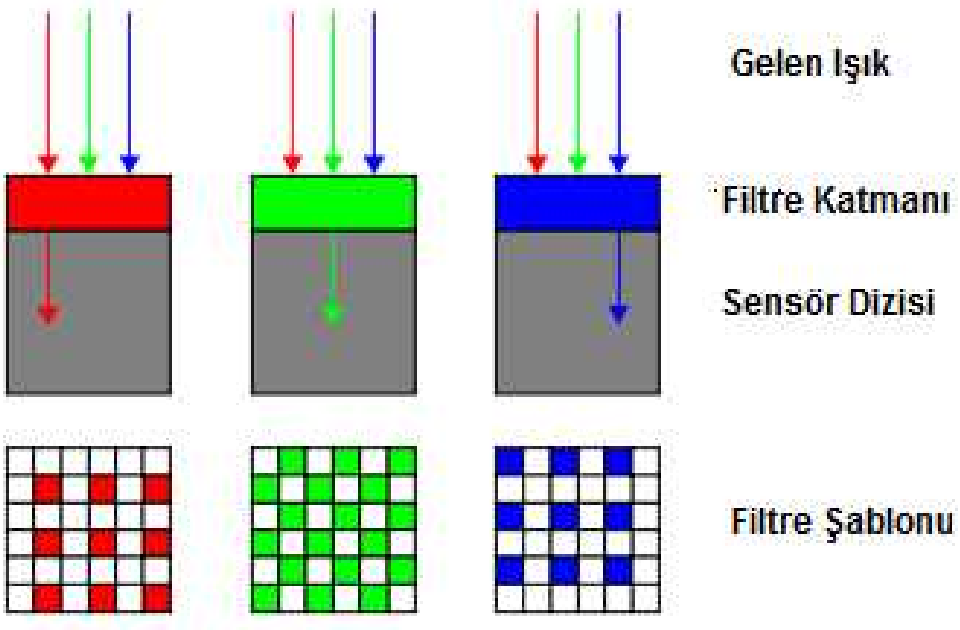
Bir CCD sensöründen alınan veri analogdur ve verinin sayısal ortamda işlenebilmesi için öncelikle bir ADC ile işlenmesi gereklidir. Buna karşılık CMOS sensörlerden alınan veri dijitaldir. Hatta çoğu CMOS sensör tümdevresinin üzerinde bir mikroişlemci bulunur ve elde edilen görüntü daha sensör tümdevresi üzerindeyken bir önişleme tabi tutulur. Projede maliyet, kullanım kolaylığı gibi nedenler dolayısıyla CMOS görüntü sensörü kullanılması kararlaştırılmıştır (8).

Görüntü sensörleri tasarlanırken bayer formatı ile renk sensörleri RGBG formunda şekildeki gibi dizilirler.



Şekil 22: Sensör Dizilimi

Nesnelerden yansıyan ışıklar Kırmızı, Yeşil ve Mavi Sensör Filtrelerinden geçerek görüntü bilgisini oluştururlar. Şekil-23 de sensörlerin çalışma mantığı ve dizilimleri ile ilgili daha detaylı bilgi vermektedir.



Şekil 23: Sensör Filtreleri

3.1.1 Altera D5M CMOS Kamera

Tasarlamış olduğumuz sistemde Altera firması tarafından üretilen 5 mega piksellik CMOS kamera kullanılmıştır. Bu kamera sistemi kullanacağımız DE2_115 ve DE0_NANO FPGA platformlarına doğrudan bağlanabildiği için tercih edilmiştir.

Şekil 24: D5M Kamera

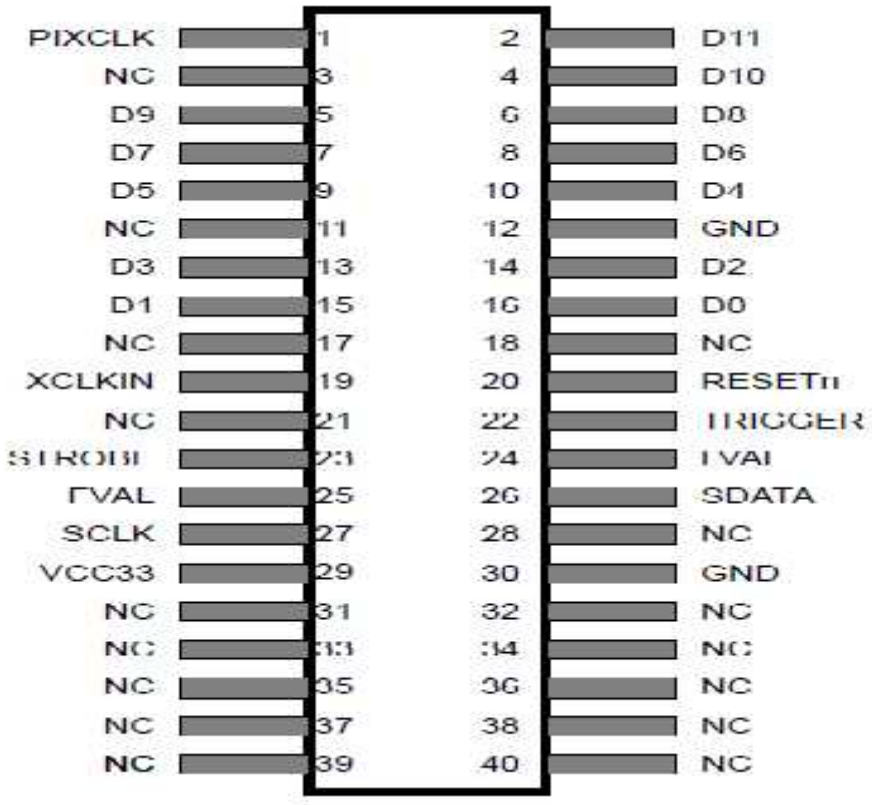
Kameranın özellikleri (20);

- Yüksek kare hızı
- Üstün düşük ışık performansı
- Düşük karanlık akımı
- Aynı anda tüm satırları sıfırlayabilme
- İsteğe bağlı anlık kare alma modu
- Yatay yada dikey tersleme modu
- Görüş alanını düşürmeden sütun ve satır atlayarak görüntü boyutunu düşürme modları (*Skipping mode*)
- Görüş alanını ve görüntü kalitesini düşürmeden sütun ve satır ortalamaları ile görüntü boyutunu düşürme modları (*Binning mode*)
- İki hat ile seri iletişim
- Programlanabilir kontroller; kazanç, kare oranı, çerçeve boyutu, pozlama

- Otomatik siyah seviye kalibrasyonu
- Tümleşik PLL özelliği

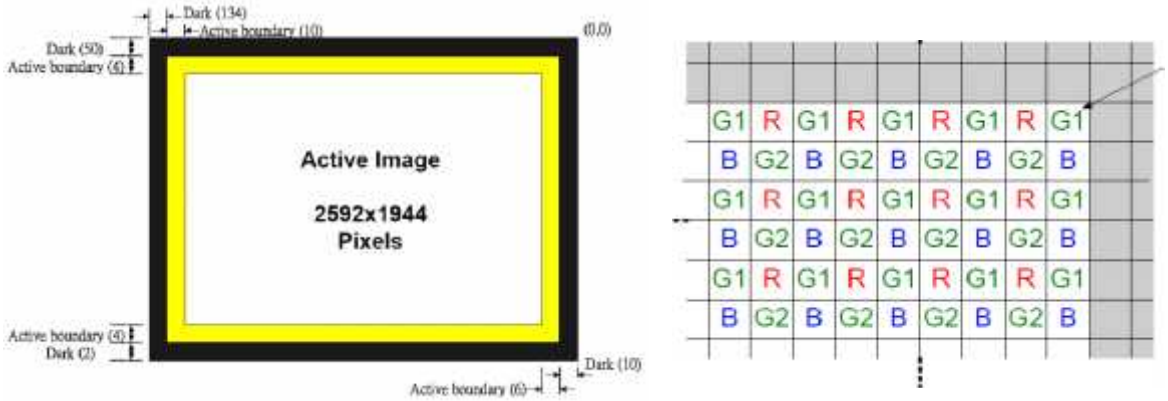
Tablo 1: D5M Teknik Özellikler

Parameter		Value
Active pixels		2,592H x 1,944V
Pixel size		2.2µm x 2.2µm
Color filter array		RGR Bayer pattern
Shutter type		Global reset release (GRR),
Maximum data rate/master clock		96 Mp/S at 96 MHz
Frame rate	Full resolution	Programmable up to 15 fps
	VGA (640 x 480)	Programmable up to 70 fps
ADC resolution		12-bit
Responsivity		1.4 V/lux-sec (550nm)
Pixel dynamic range		70.1dB
SNRMAX		38.1dB
Supply Voltage	Power	3.3V
	I/O	1.7V~3.1V



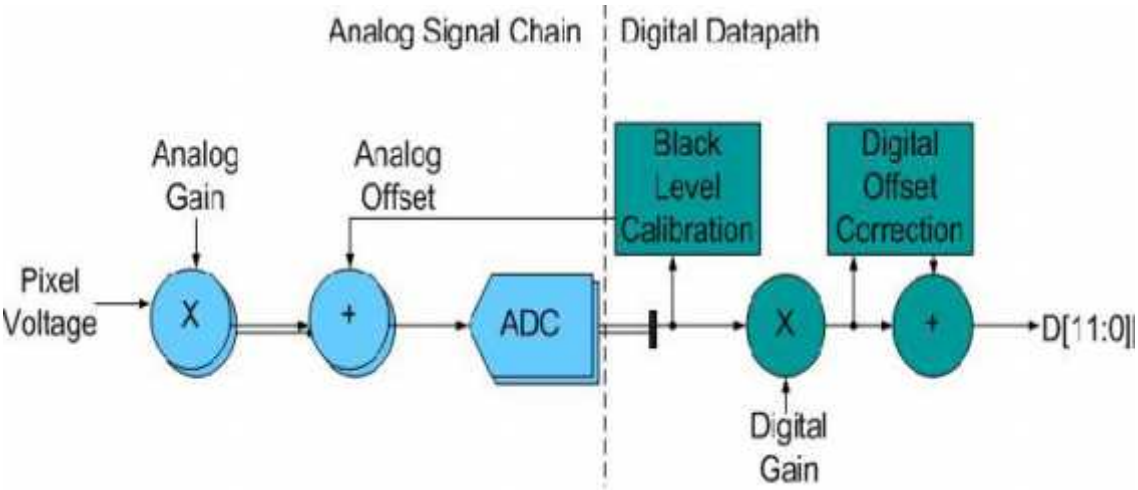
Şekil 25: D5M kamera Pin Diyagramı

Kameranın RGBG formatında sensör dizilimi ve bırakılan boşluklar şekildeki gibidir.



Şekil 26: D5M Kamera Sensör Dizilimi

Sensörlerden gelen analog bilgiler Altera D5M Kamera modülünün içinde bulunan ADC birimlerince şekil-27 de görüldüğü gibi 12 bitlik dijital bilgiye dönüştürülerek çıkışa aktarılır.



Şekil 27: ADC Çevirme İşlemi

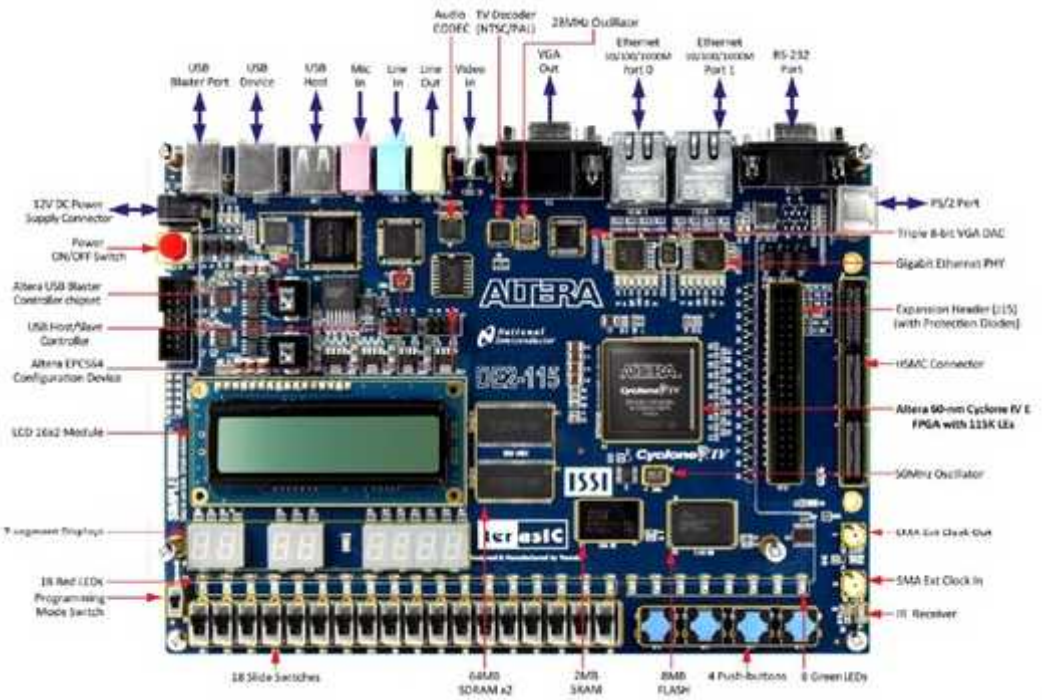
3.2 FPGA Platformu

3.2.1 Altera DE2 115 FPGA Platformu

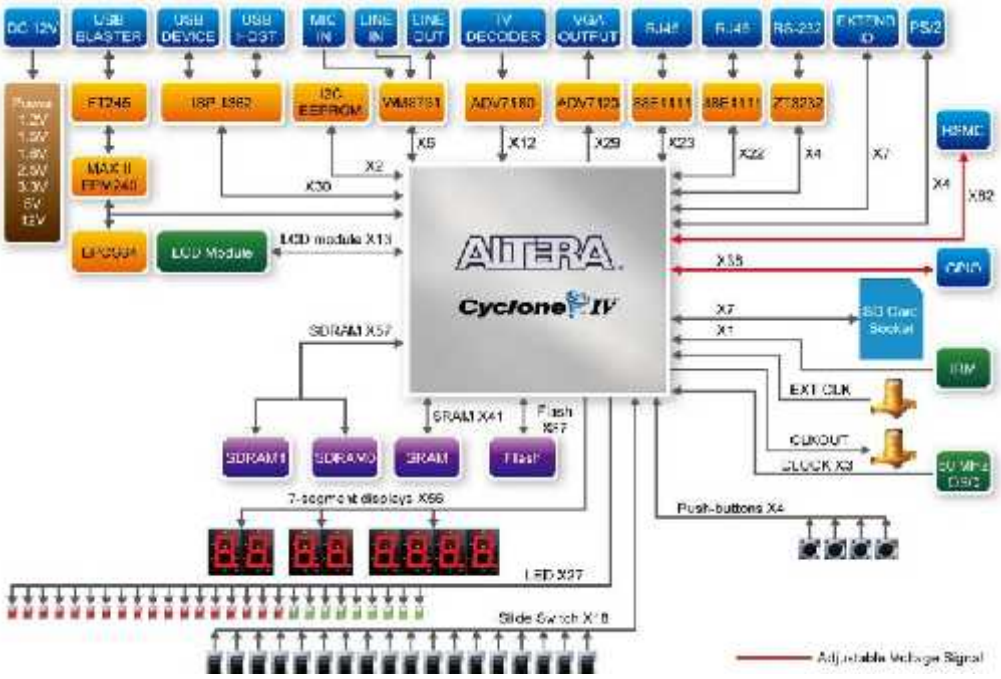
Robotik göz sistemi ilk olarak ALTERA DE2_115 FPGA platformunda tasarlandı. Bu platformun kolay temini, RAM belleğinin yüksek olması, FPGA işlemcisinin modeli, dâhili VGA çıkışlarının olması ve kullanacağımız kamera ile doğrudan bağlanabilmesi seçilmesindeki etkenlerden bazılarıdır. DE2_115 FPGA platformu üzerinde bir çok donanım ile birilikte gelmektedir. Bu donanımlar şunlardır (21);

- Altera Cyclone® IV 4CE115 FPGA işlemci
- Altera Seri Konfigürasyon Cihazı – EPCS64
- USB Blaster programlayıcı; JTAG ve AS (Active Serial)
- 2MB SRAM
- İki adet 64MB SDRAM
- 8MB Flash hafıza
- SD Kart soket
- 4 Buton
- 18 Sürgülü Anahtar
- 18 Kırmızı LED
- 9 Yeşil LED
- 50MHz Osilatör
- 24-bit CD-kalitesinde audio CODEC (line-in, line-out, ve mikrofon jakları)
- VGA DAC (8-bit yüksek hızlı üçlü DAC) ve VGAçıkış konnektörü
- TV Dekoder (NTSC/PAL/SECAM) ve TV-giriş Konnektörü
- 2 Gigabit Eternet PHY ile RJ45 konnektör
- USB kontroller konektörleri A ve B
- RS-232 seri haberleşme birimi
- PS/2 Fare ve Klavye girişleri

- IR Kumanda alıcısı
- 2 SMA harici saat sinyali giriş çıkışı
- Bir adet 40-pin genel amaçlı giriş çıkış blogu
- Bir adet HSMC (High Speed Mezzanine Card) konnektör
- 16x2 LCD modül



Şekil 28: Altera DE2_115 FPGA



Şekil 29: DE2_115 Blok Diyagram

3.2.2 Altera DE0 NANO FPGA Platformu

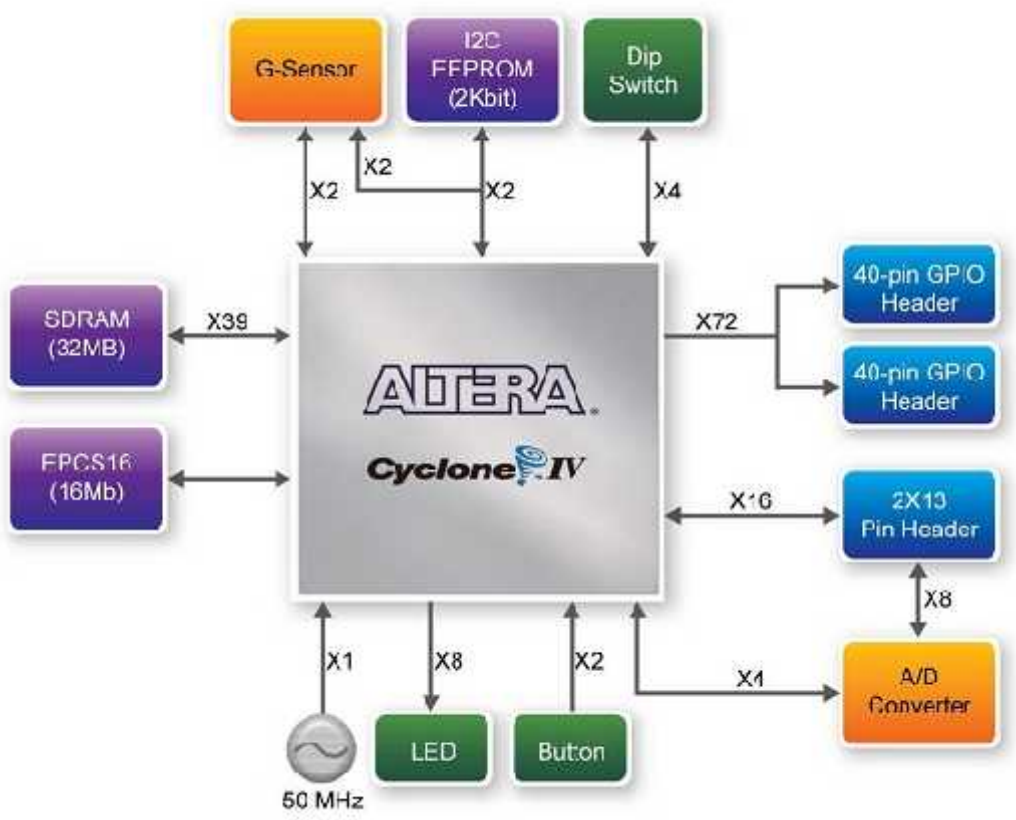
İlk olarak DE2_115 FPGA ile tasarlamış olduğumuz sistemde, robotik göz sisteminin başarılı bir şekilde çalışabilmesi için gereken temel donanımı belirledik. İkinci tasarımda ise robota montaj edilebilecek boyutlarda ve sistemin ihtiyacını karşılayacak bir FPGA platformu olan ALTERA DE0_NANO FPGA platformunu kullandık.

Şekil 30: DE0_NANO FPGA Platformu

Bu platformun sahip olduğu donanımlar şunlardır (22);

- Altera Cyclone® IV EP4CE22F17C6N FPGA işlemci
- 153 maksimum FPGA I/O pin
- Dâhili USB-Blaster programlayıcı
- Seri yapılandırma birimi – EPCS16
- İki adet 40-pin Genel amaçlı giriş çıkış birimi içinde kullanılabilir 72 I/O pini
- İki adet 5V besleme pini, iki adet 3.3V besleme pini ve dört adet toprak pini
- 32MB SDRAM
- 2Kb I2C EEPROM
- 8 yeşil LED

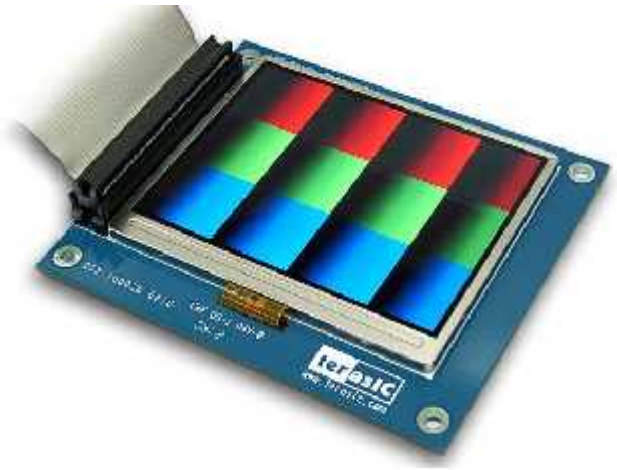
- 2 buton
- 4 DIPSWITCH
- ADI ADXL345, yüksek çözünürlükte 3-eksen ivme ölçer (13-bit)
- NS ADC128S022, 8-kanal, 12-bit A/D konvertör,
- Dâhili 50MHz osilatör
- USB Mini tip port (5V)
- 2-pin harici besleme girişi (3.6-5.7V)



Şekil 31: DE0_NANO Blok diyagram

3.2.3 TRDB LCM Ekran

Tasarlanadığımız sistemde işlenmiş görüntüyü kullanan çıkış birimidir. 320X240 piksel boyutlarında bir LCD ekran modülüdür. DE2_115 VE DE0_NANO FPGA Platformlarına doğrudan bağlanılabildiği için tercih edilmiştir.



Şekil 32: TRDB LCM Monitör

Ekran özellikleri(23) ;

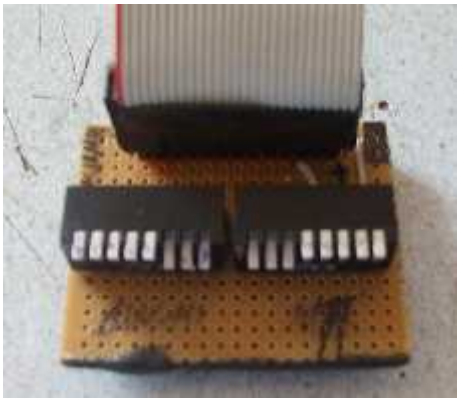
- Equipped with Toppoly TD036THEA1 compact TFT LCD module.
- 8-bit seri dijital sinyal girişi (RGB veya YUV).
- NTSC ve PAL uyumlu.
- Dahili kontrast, ışık ve gama modülasyon.
- Renk filtresi 960x240 (through mode, RGB dummy, YUV input).

Tablo 2: TRDB LCM Teknik Özellikler

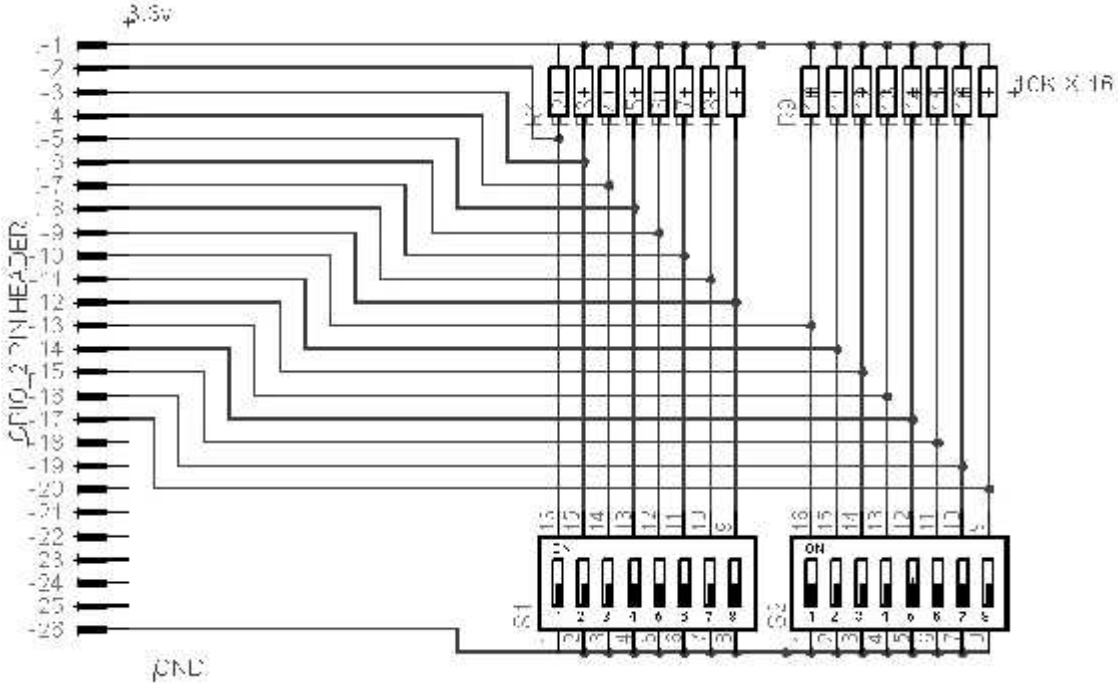
Özellik	Açıklama	Birim
Ekran Genişliği	3.6	Inch
Ekran tipi	TFT	-
Aktif Alan (HxV)	72.96 x 54.72	mm
Nokta sayısı (HxV)	320 x RGB x 240	nokta
Nokta boyutları (HxV)	0.076 x 0.228	mm
Renk modu	RGB	-
Renk sayısı	8 bit RGB (16M color)	-

3.2.4 Switch Modülü

Çalışma modlarını seçmek ve eşik (threshould) değerlerini girebilmek için tasarlamış olduğum 16 adet anahtardan oluşan modüldür. Switchler 10K lık dirençlerle 3.3 V'a pull_up yapılmıştır. DE0_NANO platformunun 26 bitlik GPIO_2 giriş çıkış birimine doğrudan bağlanacak ve besleme gerilimlerini platform üzerinden kullanacak şekilde tasarlanmıştır.



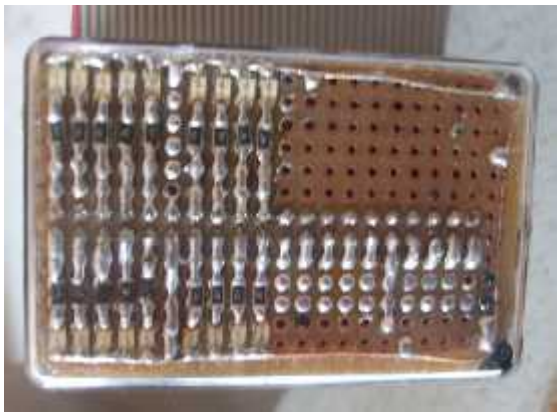
Şekil 33: Switch Modül



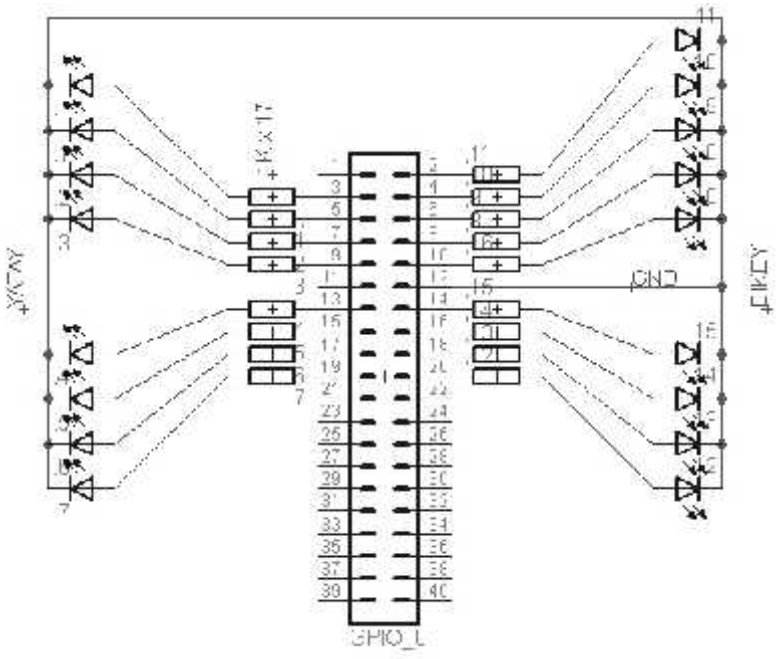
Şekil 34: Switch Modül Devre Şeması

3.2.5 Hedef Koordinat Gösterge Modülü

Hedef tanıma sisteminde bulunan hedefin koordinatlarını dış dünyaya aktarabilmek için tasarlanmıştır. Yatay kısım 9 led ile dikey kısım ise 8 adet led ile temsil edilmektedir. Koordinat bilgileri robotta başka donanımlarla da haberleşebileceği düşünülerek ikilik sayı sisteminde ledlere aktarılmıştır. Modül, DE0_NANO FPGA platformunun GPIO_0 soketine doğrudan bağlanılabilecek şekilde tasarlanmıştır. Bu soketteki diğer giriş çıkış birimlerinde LCM Monitör bağlıdır. Gösterge modülü besleme gerilimini GPIO_0 pinlerinden almaktadır.



Şekil 35: Hedef Koordinat Gösterge Modülü



Şekil 36: Hedef Gösterge Devre Şeması

4. KULLANILAN YAZILIMLAR

4.1 Quartus II Programı

Quartus II yazılımının iki farklı versiyonu bulunmaktadır. Web-Edition Quartus II nin ücretsiz ve kısıtlı versiyonu iken Quartus II Subscription - Edition ücretli ve full sürümdür. Alteranın sağladığı verilere göre web-edition full sürümün sağladığı özelliklerin %95 ini sağlayabilmektedir. Burada iki versiyon arasındaki farkları inceleyecek olursak;

Tablo 3: Quartus Sürüm Kıyaslama Tablosu

	Subscription – Edition	**Web – Edition**
İşletim sistemi	32-64 bit Windows, Linux	32 bit Windows
Desteklenen Aygıtlar	Tüm aygıtlara destek	Bazı aygıtlara destek
Multiprocessor desteği	Var	Yok
Rapid Compile	Var	Yok
Incremental Compilation	Var	Yok
IP desteği	Sınırlı Ücretsiz	Ücretli
Maliyet	Ücretli	Ücretsiz

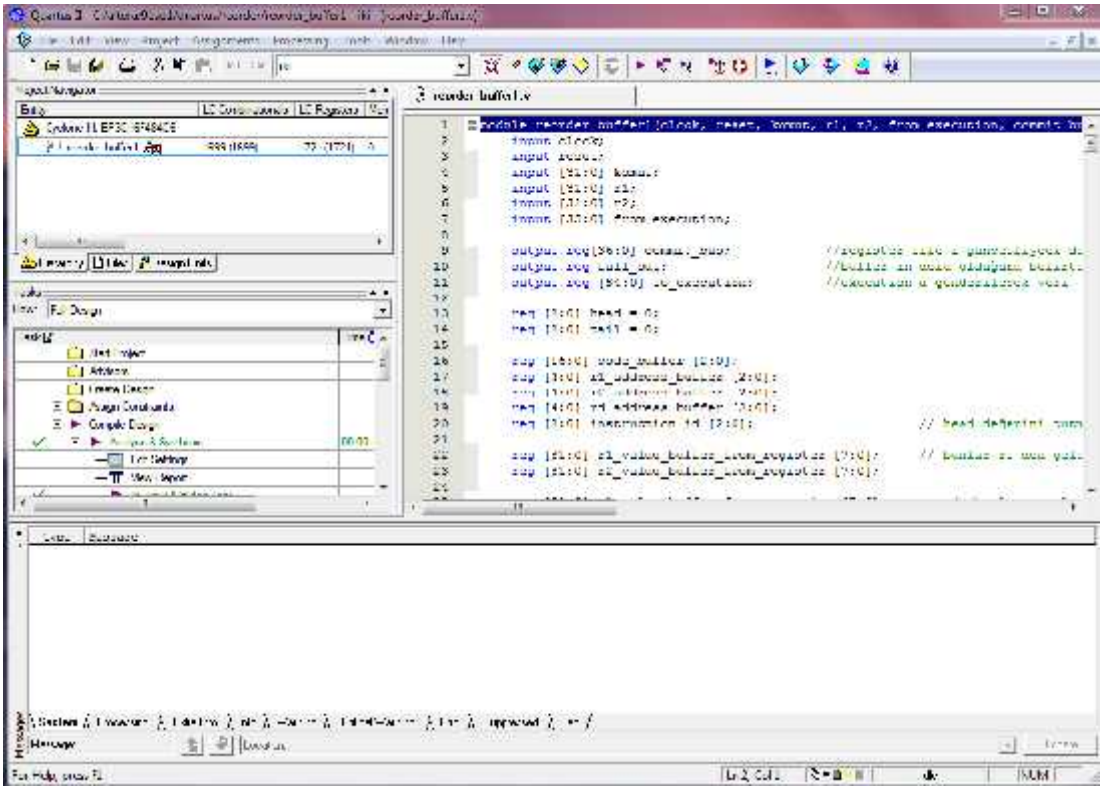
Quartus II yazılımı tamamen entegre bir yazılım geliştirme ortamıdır. FPGA programlama ile uğraşan kullanıcıların ihtiyaç duyabileceği tüm birimleri içerisinde barındırmaktadır. Bir kullanıcı temelde başka bir araca ihtiyaç duymadan şematik ve HDL dilleri ile dizayn oluşturabilir, oluşturduğu dizaynı sentezleyebilir ve bu dizaynın yerleştirme ve yönlendirme (place and route) işlemlerini gerçekleştirebilir. Ayrıca projenin doğruluğu çip üzerine yüklemeden Quartus yazılımı üzerinde simülasyonu yapılarak doğrulanabilmektedir.

Bu işlevlerinin yanı sıra Quartus II programı içerisinde zaman ve güç analiz işlevlerini de barındırmaktadır. Bu sayede oluşturduğunuz devredeki sinyallerin

zamanlamalarını önceden görebilir, kritik bir işlemin bulunduğu kısımda sınırlamaları belirterek Quartus II programının yerleştirme sırasında bu alana daha fazla önem vermesini sağlayabilirsiniz. Quartus II programı aynı zamanda içerisinde bir yükleme yazılımı da barındırmaktadır. Oluşturduğunuz çıktı bu yazılım yardımı ile çipe aktarılır.

Quartus II programı kullanıcıları 3. parti yazılımlar kullanmaları konusunda da serbest bırakmıştır. Sentezleme simülasyon ve analiz işlemleri 3. parti yazılımlar yardımı ile gerçekleştirilebilirken yerleştirme ve yönlendirme işlemlerinin Quartus programı içerisinde yapılması zorunlu kılınmıştır. Bir proje oluştururken kullanmak istediğiniz araçları programa bildirmeniz yeterli olmaktadır.

Baskı devre tasarımı yapacaksanız Orcad gibi kart tasarım programlarının kullanabileceği formatta çıktı üretmektedir.



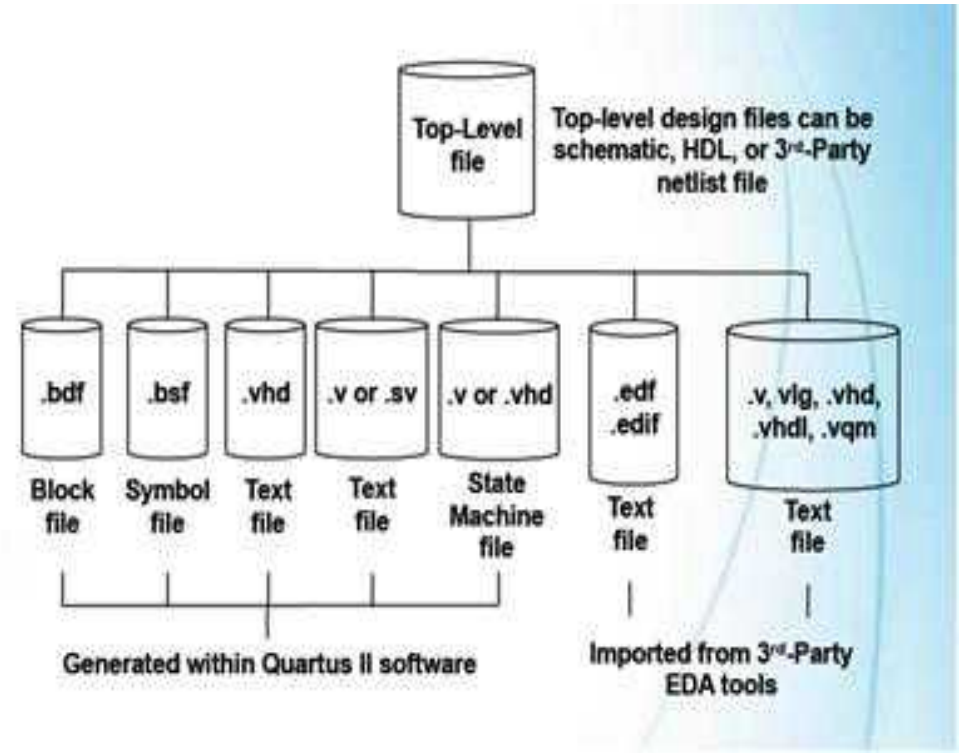
Şekil 37: Quartus II Ekran Görüntüsü

Quartus II programının ana ekranını şekil-37'deki gibidir. Açılış ekranı varsayılan olarak dört temel pencereden oluşur. Bu pencereler sol üstte project navigator, altında Task ekranı, en altta mesaj ekranı ve sağ tarafta bulunan çalışma

ekranıdır. Diğer ekranlar çok yer kaplamamaları için varsayılan olarak kapalı gelirler. Bu ekranların kullanımınıza göre boyutlarını ayarlayabilir genişletebilir ya da kapatabilirsiniz. Diğer ekranları eklemek ya da kapattığınız ekranları yeniden açmak için view menüsünde bulunan utility windows sekmesinden istediğiniz ekranı tıklamanız yeterlidir.

Quartus programı dizayn girişi için birçok dosya formatını desteklemektedir. Farklı dosya türleri aynı proje altında da kullanılabilmektedir. Böylece HDL dillerini kullanarak tasarladığınız modülleri şematik bir top modül altında birleştirebilirsiniz.

Quartus içerisinde, oluşturduğunuz dizayn dosyaları birbiri arasında da dönüştürülebilmektedir. Örneğin oluşturduğunuz bir şematik tasarımı HDL dillerinden birine dönüştürebilir ya da tam tersini yapabiliriz. Şematik tasarımdan HDL tasarıma geçerken bu oldukça eğitici olabilmektedir. File menüsünden create /update komutu ile bu dönüşümleri gerçekleştirebiliriz.



Şekil 38: Quartus II Dosya Türleri

4.2 Modelsim Programı

Yapılan tasarımlarda yazım hatası olmasa da, istediğimiz sonuçları alıp alamayacağımızı bilemeyiz. Özellikle büyük programlarda, FPGA'e yüklenmeden önce programın doğru çalıştığı kontrol edilip, mantıksal hataların giderilmesi gerekir. Aksi takdirde beklenmedik sonuçlar elde edebiliriz. Bundan dolayı programımız için bir test programı hazırlayıp, bunun herhangi bir test platformunda simülasyonunu yapmak gerekecektir.

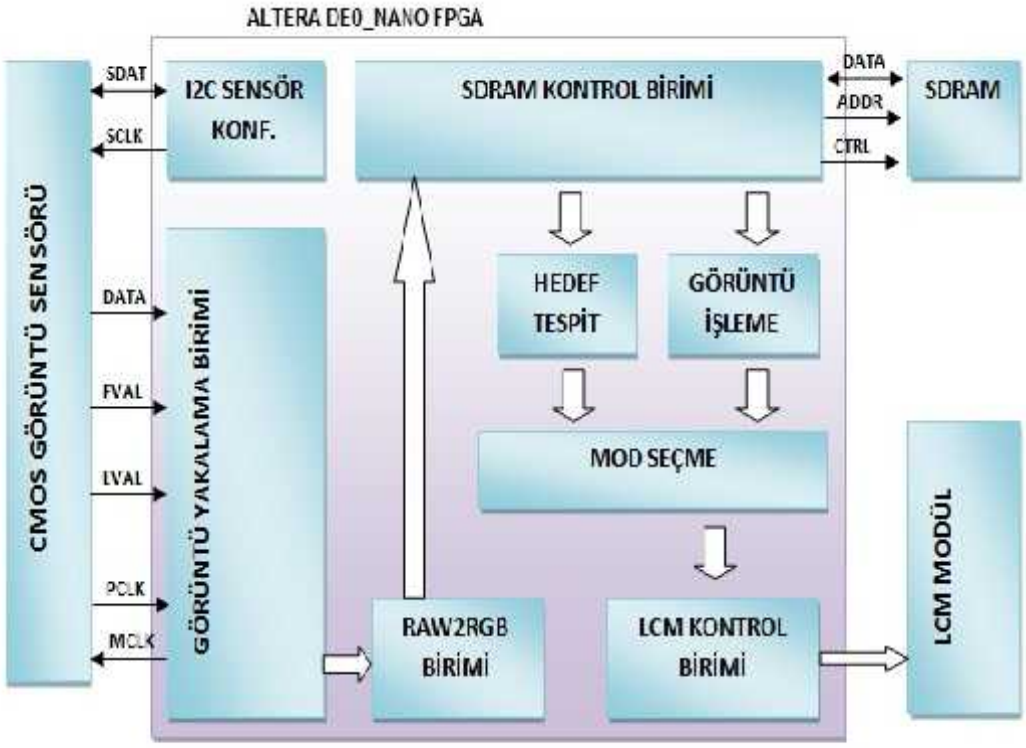
Quartus II & ModelSim ile RTL ve Gate Level Simülasyon yapılabilir. RTL simülasyonu sentezleme sonucunda elde edilecek sayısal sistemin fonksiyonel testleri yapılır. Gate Level simülasyonda ise yerleştirme (fitter) sonrasında FPGA üzerindeki yerleşimden kaynaklanan yol ve lojik gecikmelerinde dikkate alınır.

Modelsim ile bir simülasyon yapmadan önce tasarlanan sistemi çalıştırabilecek bir kod yazılır bu koda ***testbench*** denir.

Sayısal devre simülasyonu ile tasarlamış olduğumuz sayısal sistemin ve bunun alt modüllerinin gerçeklendiğinde çalışması hakkında bilgi edinilir. Bu simülasyon ile tasarlanmak istenen sistem bilinen girişler ile test edilerek beklenen sonuçlar elde edildiği doğrulanır. Test edilen modül DUT(Device Under Test) olarak adlandırılır. Testbench, test girişleri ve DUT içeren ve simülasyon sonuçlarını programsal olarak değerlendirmemize yardımcı bir modüldür. Tasarlanan tüm sistem için testbench oluşturulabileceği gibi, kullanılan her bir modülün testi için de testbench'ler oluşturulabilir.

5. GÖRÜNTÜ İŞLEME DONANIMI TASARIMI

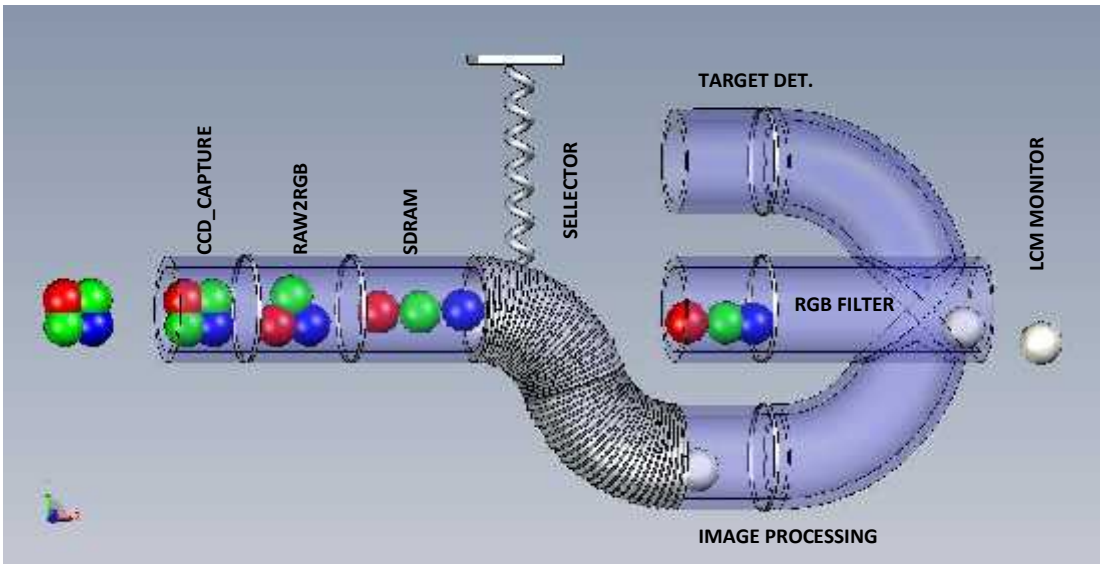
Robotik göz sistemimiz gerçek zamanlı görüntü işleyebilecek ve hedefe kilitlenen nesnenin koordinatlarını dışarıya aktarabilecek şekilde FPGA platformu üzerinde şekildeki gibi tasarlanmıştır.



Şekil 39: Sistem Blok Diyagramı

Kısaca özetleyecek olursak sistem şu şekilde çalışmaktadır. I2C sensör yapılandırma tarafından kontrol edilen kameradan gelen sinyaller bir boru içindeki misketler gibi her saat sinyalinde sırayla bir sonraki işleme tabi tutulurlar. Böylece sistem paralel olarak çalışır. Bir piksel işlenmek üzere sisteme girerken bir önceki CCD_CAPTURE birimi tarafından alınıyor, ondan önceki RAW2RGB biriminde dönüştürülüyor ve daha da öncekiler ise diğer işlemlere tabi tutuluyor olacak.

Sisteme her saat sinyali gelişinde girişinden bir piksel bilgisi girerken çıkışından işlenmiş bir piksel bilgisi çıkacak. Bu durum bir boru içindeki misketler şeklinde örneklendirilebilir. Borunun bir ucundan yeni bir misket sokunca, diğer ucundan başka bir tanesi düşmekte. Şekil-40'da bu örnek Solid Works programı kullanılarak görselleştirilmeye çalışılmıştır.



Şekil 40: Sistem Çalışması Boru Örneği

5.1 I2C CCD Birimi

Sistemde kullanılacak olan D5M kamera modülünün temel ayarlarının yazılabilmesi için FPGA platformu ile Kameranın Eprom'u arasındaki haberleşmeyi sağlayan birimdir. Bu birimde kameranın aşağıdaki özelikleri sabit olarak girilmektedir.

Sabit girilen yazmaçlar;

- Mirror özelliği aktif yada pasifleştirme,
- Exposure (aydınlık karanlık ayarı),
- H_Blanking-V_Blanking (yatay ve dikey boşlukların ayarı),

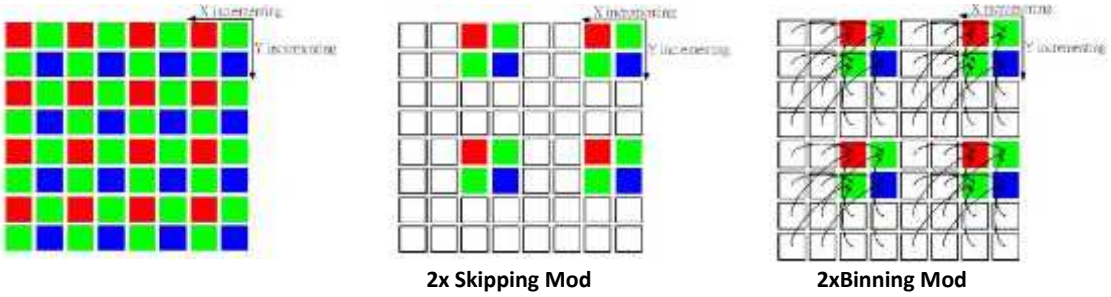
- LATCH değiştirme ayarları,
- R,G1, G2 ve B kazanç ayarları,
- Dahili PLL ayarları,
- Test Şablonları (Pattern) ayarları,
- Satır, sütun başlangıç ve boyut ayarları,
- Resim alma mod ayarları(skipping, binning mode)

Ayarlar kameranın epromuna direk yazılabildiği gibi değişkenlere atanarak sistem çalışırken de değiştirilebilir. Tasarlamış olduğumuz sistemde aydınlık karanlık ayarı (exposure), aynalama özelliği (mirror) switchlere atanarak sistem çalışırken müdahale edilecek şekilde tasarlanmıştır. Bu modülde yapılan en önemli ayarlardan biride, kamera çözünürlüğünün kullanılacak ekran boyutuna uygun olarak seçilmesidir. Kullanacağımız ekran boyutu 320x240 olduğundan kamera çözünürlüğümüzü en düşük boyut olan 640'a düşürmemiz gerekmektedir. Kameramızın normal çözünürlüğü 2592x 1944 'tür. Çözünürlüğü düşürmek için iki yöntem vardır. Bunlar skipping ve binnig modlarıdır. Tablo4'de kamera eprom'unda yapılacak ayarla ilgili teknik bilgiler mevcuttur (20).

Tablo 4: Kamera Çözünürlük Modları

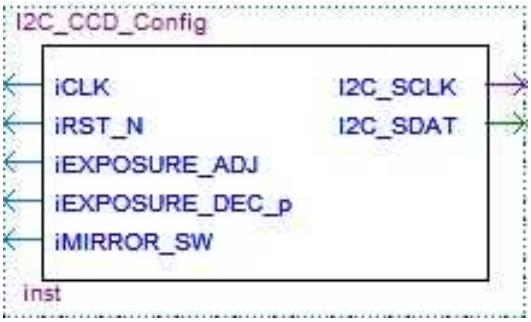
Resolution	Frame Rate	Sub-sampling Mode	Column_Size (R0x04)	Row_Size (R0x03)	Shutter_Width_Lower (R0x09)	Row_Bin (R0x22 [5:4])	Row_Skip (R0x22 [2:0])	Column_Bin (R0x23 [5:4])	Column_Skip (R0x23 [2:0])
2592 x 1944 (Full Resolution)	15.15	N/A	2591	1943	<1943	0	0	0	0
2,048 x 1,536 QXGA	23	N/A	2047	1535	<1535	0	0	0	0
1,600 x 1,200 UXGA	35.2	N/A	1599	1199	<1199	0	0	0	0
1,280 x 1,024 SXGA	48	N/A	1279	1023	<1023	0	0	0	0
	48	skipping	2559	2047		0	1	0	1
	40.1	binning	2559	2047		1	1	1	1
1,024 x 768 XGA	73.1	N/A	1023	767	<767	0	0	0	0
	73.4	skipping	2047	1535		0	1	0	1
	59.7	binning	2047	1535		1	1	1	1
800 x 600 SVGA	107.7	N/A	799	599	<599	0	0	0	0
	107.7	skipping	1599	1199		0	1	0	1
	85.2	binning	1599	1199		1	1	1	1
640 x 480 VGA	150	N/A	639	479	<479	0	0	0	0
	150	skipping	2559	1919		0	3	0	3
	77.4	binning	2559	1919		3	3	3	3

Kameramızın standart çözünürlüğü olan 2542 x 1944 lük çözünürlüğü 640x480 düşürebilmek için satır ve sütunlarda 4 te 1 oranında atlayarak görüntü almamız gerekmektedir. Kameramızın eprom'unun R0x22 ve R0x23 numaralı yazmaçlarının [2:0] bitlerine $(11)_2$ bilgisi yüklenerek kamera objektif alanından kayıp vermeden görüntü kalitesini düşürerek küçültme işlemini skipping modda gerçekleştirebiliriz. Bu da sistemin daha hızlı çalışmasını sağlar. Bu şekilde 150 Fps hızına çıkılabilir. Görüntü kalitesini düşürmeden küçültme işlemi yapmak istersek Binning modu kullanmamız gerekir. Bu modda objektif alanından ve görüntü kalitesinden kayıp olmaz ama işlem hızı 77 Fps ile sınırlı kalır. Aşağıdaki şekilde bu iki mod ile ilgili görsel açıklama yapılmıştır.



Şekil 41: Skipping ve Binning Modları

Kameranın kontrolü için tasarlamış olduğumuz sistemin blok şeması şekilde gösterilmektedir.

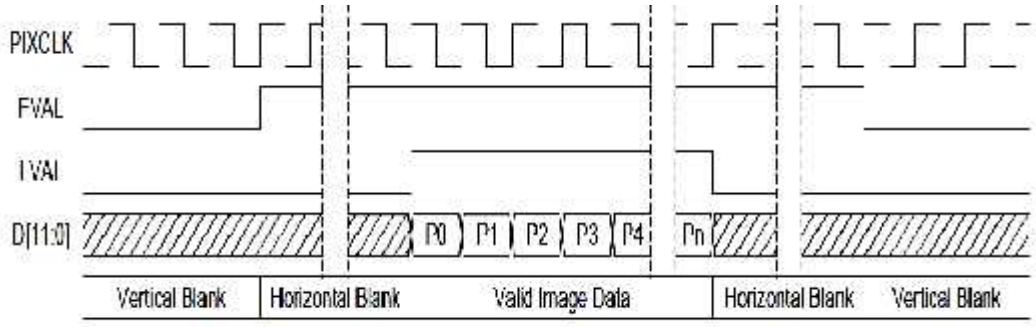


Şekil 42: I2C_CCD Birimi

5.2 CCD CAPTURE Birimi

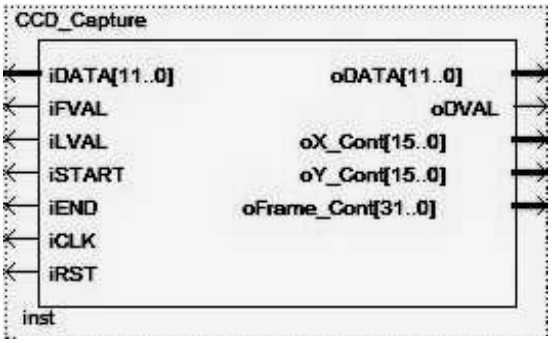
Bu birim kameradan gelen renk bilgilerinin sisteme ilk olarak alındığı birimdir. 12 bitlik renk bilgileri kamera tarafından üretilen her saat sinyalinde (Pixel_clock) sisteme alınır.

Sistem aynı anda FVAL ve LVAL değişkenlerini AND kapısı'ndan geçirerek DVAL değerini elde eder. DVAL değeri işlenen pikselin resmin içinde olup olmadığı bilgisini sisteme verir.



Şekil 43: FVAL ve LVAL Durum Göstergesi

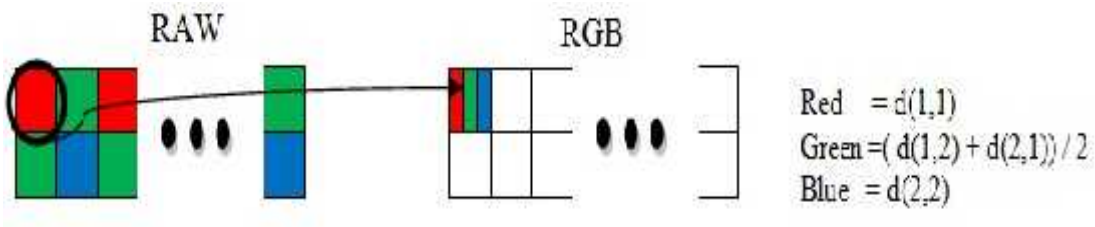
FVAL değerinin ve LVAL değerinin artıştı ile X_CONT ve Y_CONT değerleri hesaplanmaktadır. Bu değerler işlenen pikselin resim üzerindeki koordinatlarını vermektedir. Bu modül içinde Sütun genişliği bilgisi de girilmektedir. Kamerada kullanılan sütun genişliğine uygun olarak seçilmesi gerekmektedir. Bu değer projede 640 olarak seçilmiştir. Aynı zamanda FVAL değerinin 0 dan 1'e her geçişinde resim sayacı (Frame_Cont) değişkeni 1 artırılır. Tasarlanan modülün 12 bitlik, data, DVAL, x_Cont, Y_Cont ve 32 bitlik Frame_Cont hatları RAW2RGB modülünün girişlerine bağlanmaktadır.



Şekil 44: CCD_Capture Modülü

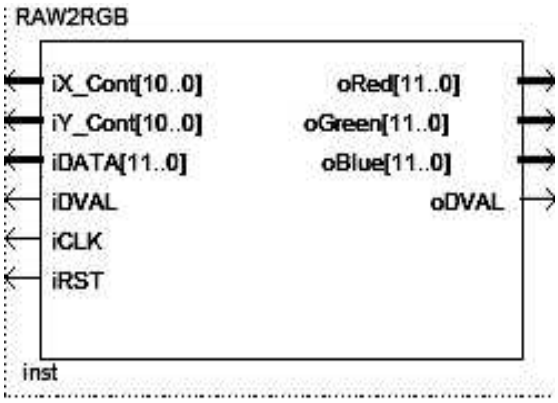
5.3 RAW2RGB Birimi

RAW2RGB modülü CCD_Capture modülünden gelen 12 bitlik renk bilgilerini, içinde tasarlanmış olan Line_Buffer birimi tarafından 2 adet tepsiye maksimum genişlik dikkate alınarak dizer. İki satır halini alan görüntüler şekildeki gibi X-Cont ve Y_Cont değerlerinin 0. bitine göre ortalamaları alarak RGB değerlerine dönüştürür.



Şekil 45: RAW - RGB Dönüştürme

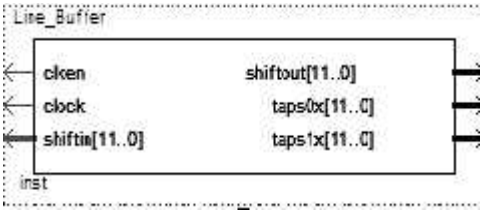
RGB formatına dönüştürülen renk bilgileri SDRAM'a yazılmak üzere SDRAM kontrol birimine aktarılır. ODVAL değeri ise RAM'de yazma ucunu aktif etmek üzere çıkışa aktarılır. Tasarlanan modülün görünümü şekil 46'da gösterilmiştir.



Şekil 46: RAW2RGB Birimi

5.3.1 Line Buffer Birimi

Kamera sensörlerinden Shift_in girişine gelen 12 bitlik renk bilgilerini iki satır halinde geçici olarak depolayan birimdir. Bu birimin boyutu kameranın sütun genişliği ile aynı boyutta seçilir. Bu proje için bu değer 640 olarak belirlenmiştir. Tasarlanan birimin blok diyagramı şekildeki gibidir.



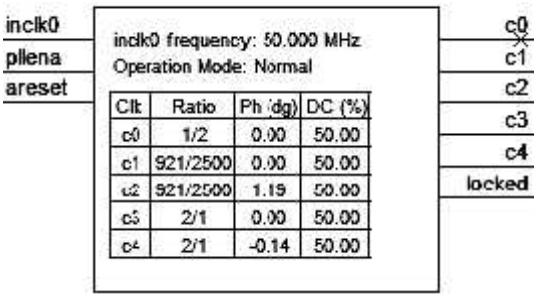
Şekil 47: LINE BUFFER Birimi

5.3.2 PLL Birimi

Tasarlanan sistemde bulunan donanımlar farklı frekanslarda çalışmaktadırlar. Kamera modülümüz 25 MHz, Ekran Modülümüz 18,42 MHz RAM'ler ise 100 MHz frekansında saat sinyalleri ile çalışmaktadır. Buna karşın FPGA donanımında 50 MHz'lik bir saat üreteci bulunmaktadır.

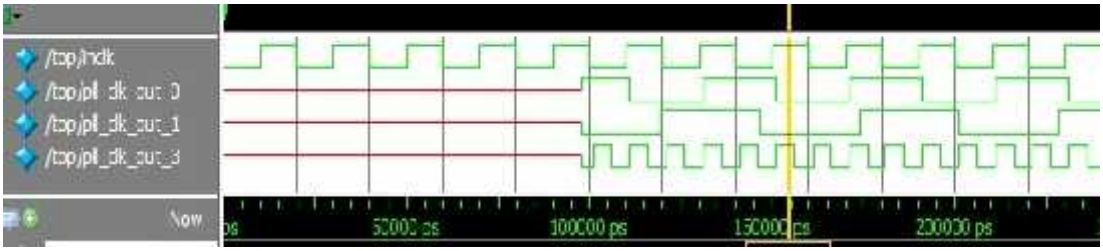
Tasarlanan bu modül, FPGA platformumuzda bulunan PLL bloklarından bir tanesini kullanarak istediğimiz frekanslardaki sinyalleri üretmemizi sağlamaktadır. Bu modülü tasarlamak için Quartus II programında Mega Function'lar arasında bulunan **alt_pll** aracı kullanılmıştır.

Sistem saat sinyallerini üretebilmek için tasarlamış olduğumuz PLL bloğu şekil 48'de görülmektedir.



Şekil 48: PLL_Bloğu

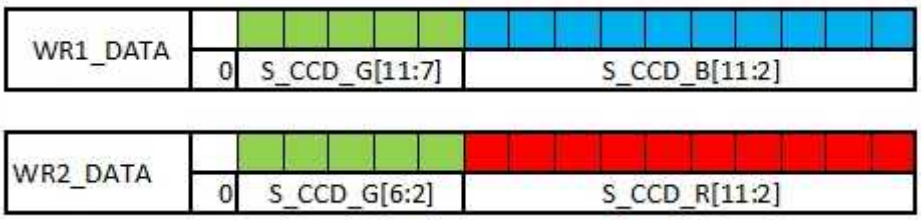
PLL bloğu Modelsim programı ile simülasyon yapıldığında elde edilen görüntü şekildeki gibidir. Yazılan testbench programında PPL bloğu ilk 100 ns pasif sonradan aktif olmaktadır.



Şekil 49: PLL Bloğu Giriş Çıkış Sinyalleri

5.4 SDRAM Kontrol Birimi

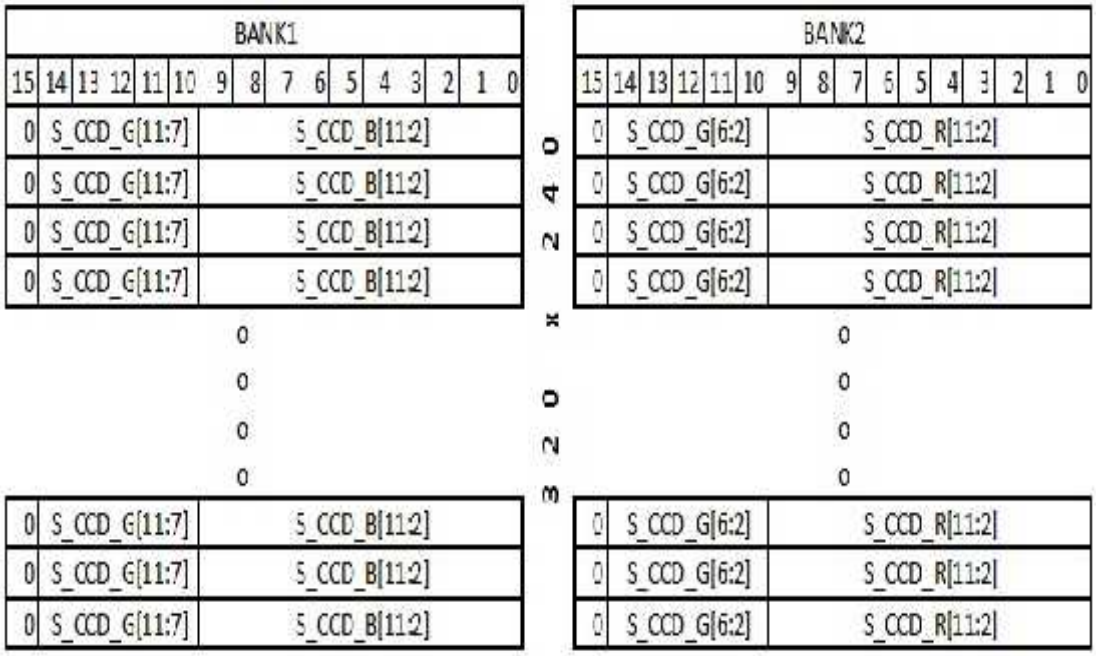
Bu birim FPGA platformu üzerinde bulunan SDRAM donanımını kontrol eden birimdir. RAM çalışma frekansı PLL bloğu tarafından üretilen 100 MHz'lik saat sinyalidir. RAW2RGB birimi tarafından RGB formatına dönüştürülen piksellerin 12 bitlik renk değerleri SDRAM yazma uçlarına transfer edilir. WR1_DATA ve WR2_DATA girişleri 16 bitlik yazmaçlardır. Renk değerlerimizin bu yazmaçlara yerleştirilmesi şekil 50'deki gibi tasarlanmıştır. 12 bitlik renk bilgilerinin üst 10 bitlik kısımları kullanılmıştır.



Şekil 50: RAM Yazma Veri Birleştirme

Yazma işlemi için kullanılan saat sinyali kamera tarafından her pikselle birlikte üretilen Pix_Clock sinyalidir. Yazma işlemi yetki ucu ise sCCD_DVAL yazmacı ile belirlenmiştir. Bu yazmaç daha öncede bahsedildiği üzere gelen pikselin kamerada aktif bölgede olup olmadığı bilgisini vermektedir.

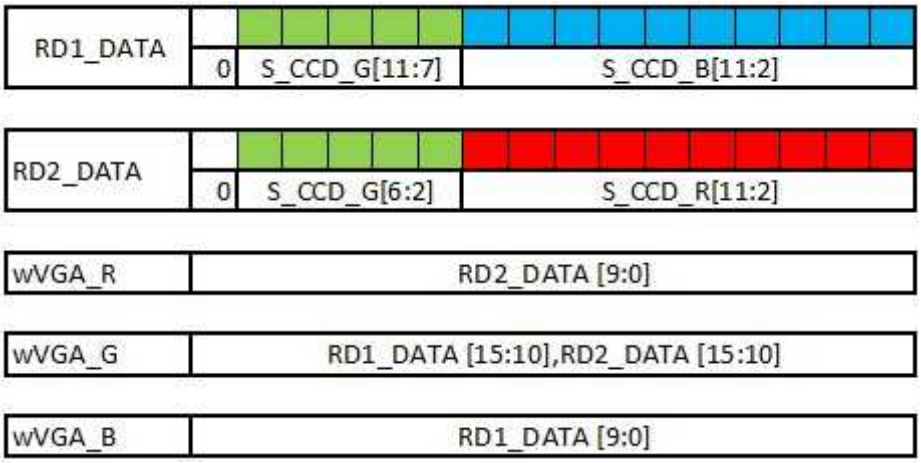
RAM' in boyutu kameradan gelen görüntü boyutuna göre ayarlanır. Kameramızın boyutu 320 x240 olduğundan RAM de maksimum yazma adresi 320 x 240 olarak belirlenir. RAM'e yazma işlemi FIFO (ilk giren ilk çıkar) sistemi ile çalıştığından maksimum adres aşıldığında ilk RAM yazmacına tekrar veri yazılır. RAM yazmaçlarının kullanımı şekil-51'de gösterilmiştir.



Şekil 51: RAM Tasarımı

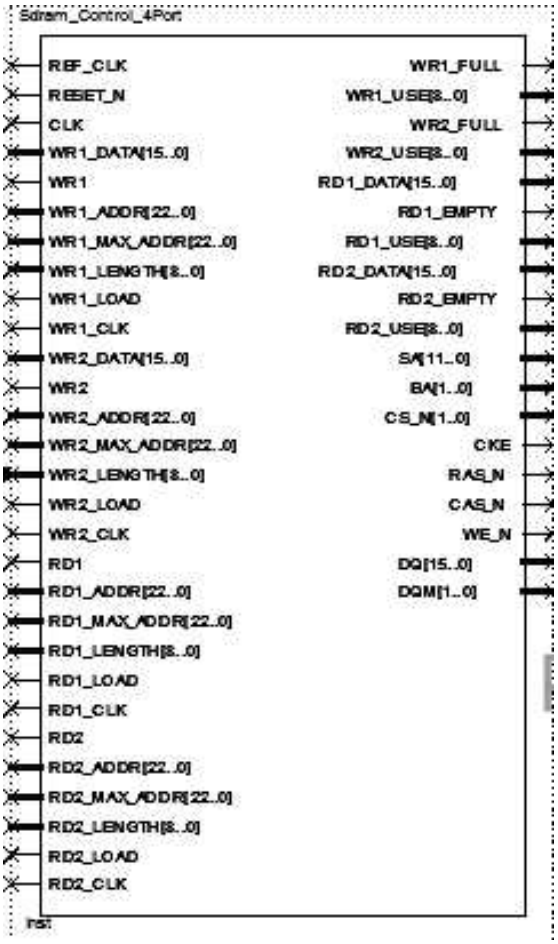
RAM'den okuma yetki ucu LCM_KONTROL modülünün READ çıkışı ile kontrol edilmektedir. Okuma bölümü LCM_KONTROL biriminin saat sinyali olan 18.42 MHz lik sinyalin 180° faz farkı ile okuma yapar. İki birim arasında senkronizasyon olduğundan RAM den 13x2 . adresteki bilgi okunuyorsa LCM ekranda da 13x2 . piksele çıkış yapılmaktadır.

RAM'den okunan veriler RD1_DATA ve RD2_DATA yazmaçlarına yüklenir. Bu yazmaçlarda birleştirilmiş olarak bulunan renk bilgileri şekil-52'deki gibi ayrılarak yazmaçlara yazılır.



Şekil 52: RAM Okuma Veri Ayırma

Tasarlanan RAM_Kontrol modülünün tüm giriş ve çıkışları şekil-53'de gösterilmiştir.



Şekil 53: SDRAM Kontrol Birimi

5.5 Görüntü İşleme Birimi

Kameradan gelen görüntü üzerinde filtreleme işlemini gerçekleştirebilmek için görüntü işleme birimi tasarlanmıştır. Bu birime gelen RGB görüntü bilgileri tasarlanan paralel toplayıcı ile her saat sinyalinde toplanır ve 3'e bölünerek Gri görüntüye çevrilir.

Filtreleme işlemlerinde 3x3 boyutunda bir şablon matris ile görüntünün konvolüsyon işlemine tabi tutulması gerekmektedir. Konvolüsyon denklemi;

$$g(x,y) = \sum_{j=-n}^{n} \sum_{i=-m}^{m} k(i,j).f(x-i,y-i) = k \times f$$

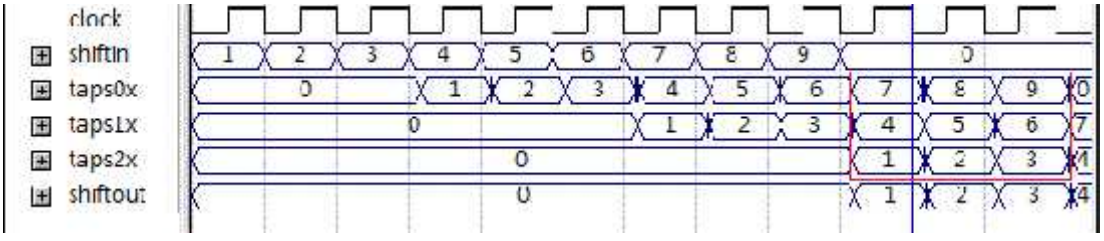
$$m = \frac{w-1}{2} \qquad n = \frac{h-1}{2}$$

k: Şablon matris ***f***: işlenecek görüntü

w , h : görüntü boyutları

Konvolüsyon, yumuşatma, keskinleştirme, kenar belirleme gibi görüntü işleme fonksiyonlarını gerçekleştirmede çok sık kullanılmaktadır. Konvolüsyonda bir pikselin yeni değeri kendisinin ve çevresindeki piksellerin ağırlıklı ortalaması ile bulunmaktadır. Piksellerin ağırlıkları konvolüsyon şablonu olarak adlandırılan bir matris ile belirlenir. Konvolüsyon şablonu uygulamaya göre farklı boyutlarda olabilmekle beraber genelde 3x3 lük bir matristir (8).

Konvolüsyon işlemi için görüntü bilgilerini üç satır halinde yazmaçlarda tutulması gerekir. Sistem üzerinde üç satırlık Line_Buffer modülü tasarlanmıştır. Line_Buffer modülü simülasyon görüntüsü şekil-54'deki gibidir.



Şekil 54: Line Buffer Simülasyon Çıktısı

Bir piksel için konvolüsyon işleminde görüntü matrisi ile şablon matriste çakışan elemanlar FPGA üzerindeki dâhili çarpma bloklarında çarpılır ve her sütun elemanları bir paralel toplayıcıda toplanır. Sonra üç sütun tekrar bir paralel toplayıcı ile toplanır. Böylece bir saat sinyalinde bir pikselin konvolüsyonu gerçekleştirilmiş olur. Bu işlemin matematiksel ifadesi aşağıdaki gibidir.

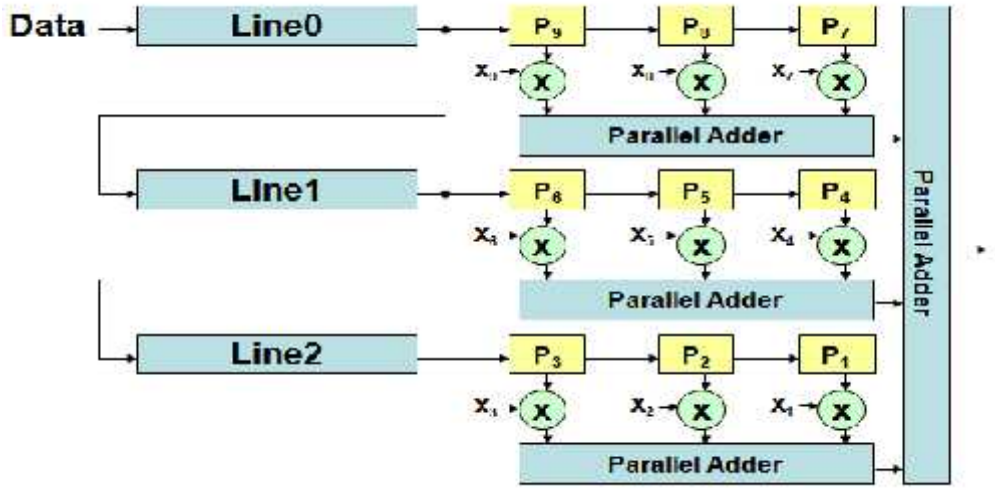
Örnek:

$$Gri\ imge = \begin{bmatrix} 18 & 34 & 12 & \cdots\cdots & 89 \\ 14 & 25 & 900 & \cdots\cdots & 560 \\ 950 & 890 & 10 & \cdots\cdots & 674 \end{bmatrix} \text{ X } şablon = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$i(1,1) = [(-18) + 0 + 12 + (-28) + 0 + 1800 + (-950) + 0 + 10] = 826$$

$$işlenmiş\ görüntü = \begin{bmatrix} 826 \end{bmatrix}$$

Bu matematiksel ifadeyi FPGA üzerinde gerçekleştirebilmek için şekil-55'deki donanım tasarlanmıştır.



Şekil 55: Konvolüsyon Birimi

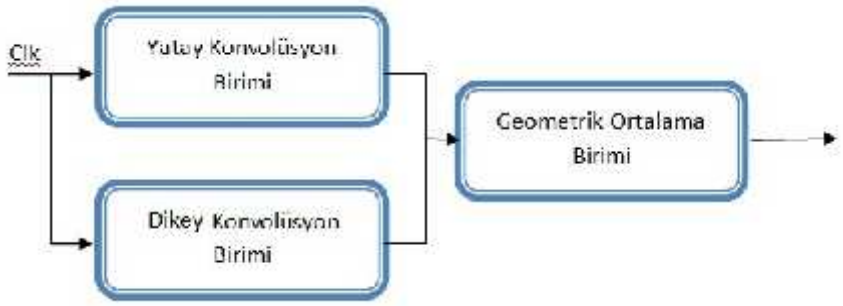
Tezimizde kullanmış olduğumuz kenar belirleme filtrelerinden biriside Sobel filtresidir. Bu filtrenin diğerlerinden farkı ise her pikseli iki kere *(yatay ve dikey kenar belirleme için)* konvolüsyon işlemine tabi tutmasıdır. Dolayısıyla bu filtrede yatay ve dikey olmak üzere iki adet şablon matris vardır. Sobel filtresinin matematiksel ifadesi aşağıdaki gibidir.

$$Gx = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} x \ imge \qquad \text{v e} \qquad Gy = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} x \ imge$$

Yatay ve Dikey şablon matrisleri ile x ve y yönünde filtre yapıldıktan sonra,

$G = \sqrt{Gx^2 + Gy^2}$ Geometrik ortalaması alınır.

$G = \sqrt{Gx^2 + Gy^2}$Sobel filtresinde kullanılan sistem tasarımı şekildeki gibidir.



Şekil 56: Sobel Filtre Birimi

5.6 Sistem Mod Seçme ve Kontrol Birimi

Bu birim sistemin hangi modda çalıştırılacağını ve bazı modlar için dışarıdan girilmesi gereken verilerin girildiği birimdir.

Tasarlanan sistem 6 moddan oluşmaktadır, bu modlar arasında geçiş switch modülde bulunan sw_0 anahtarları kullanılarak gerçekleştirilir.

Sistem çalışma modları şunlardır;

i. Kırmızı renk filtre modu,
ii. Yeşil renk filtre modu,

iii. Mavi renk filtre modu,

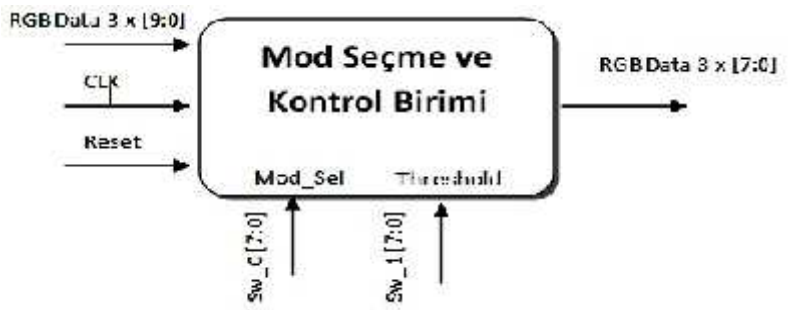
iv. Gri renk filtre modu,

v. Hedef Bulma ve Kilitlenme modu,

vi. Görüntü İşleme Modu.

- Normal görüntü modu,
- Embos Filtresi Modu,
- Laplace Filtresi modu,
- Sobel Filtresi Modu.

Sisteme kırmızı , yeşil ve mavi renk filtrelerinde ve sobel filtresinde eşik (threshold) değerleri switch modülde bulunan Sw_1 anahtarları kullanılarak girilmektedir.Tasarlanan birimin blok gösterimi şekildeki gibidir.



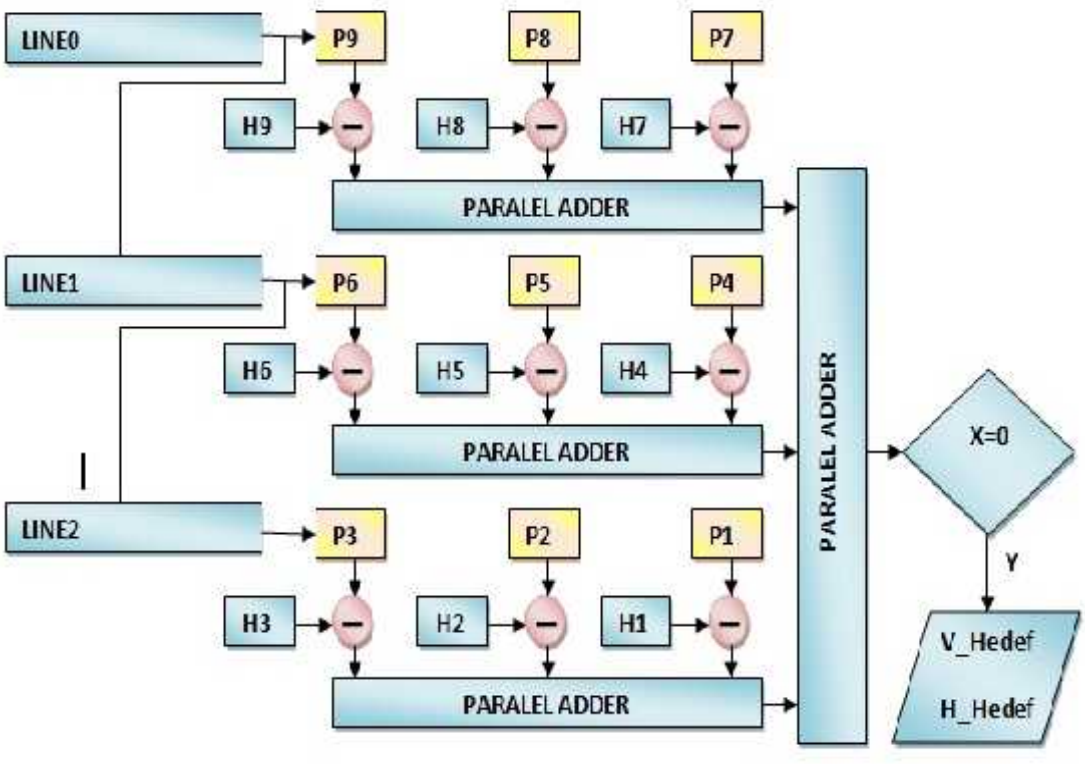
Şekil 57: Mod Seçme Birimi

5.7 Hedef Tanıma ve Kilitlenme Birimi

Bu birim robota tanımlanan nesnenin konumunu gerçek zamanlı olarak çıkışa aktaran ve LCM ekranda nesnenin merkezini yatay ve dikey bir çizgi ile gerçek zamanlı takip eden birimdir.

Hedef tanıma ve kilitlenme birimi ilk çalıştırma anında takip edilecek nesnenin kameraya tutulmasını, ekranın tam orta noktasındaki imlece sabitlenince ise Set tuşuna basmamızı bekler. Böylece hedef yazmacımıza takip edilecek pikselin ve etrafındaki 8 komşusunun RGB renk değerleri yazılır. Sistem girişine gelen RGB değeri üç satırlık Line_Buffer da tutulur. Hedef yazmacı ve komşularından oluşan 3x3 lük hedef matris sırayla Line_Buffer bulunan gerçek görüntüden çıkarılır. Sonuç

yeniden Line_buffer'a yazılmaz. Sadece sonucu 0 çıkan konum bilgileri V_Hedef ve H_Hedef yazmaçlarına yazılır. Bu yazmaçları kullanan imleç çizme birimi ekranda V_Hedef ve H_Hedef yazmaçlarındaki koordinatlara yatay ve dikey birer çizgi çizer. Böylece hedefe kilitlenen nesne ekranda da imleç yardımı ile takip edilir. Aynı zamanda V_Hedef ve H_Hedef yazmaçları Sistem veri Çıkış birimine aktarılır. Tasarlanan sistem şekil-58'deki gibi çalışmaktadır.



Şekil 58: Hedef Tanıma Birimi

5.8 Sistem Veri Çıkış Birimi

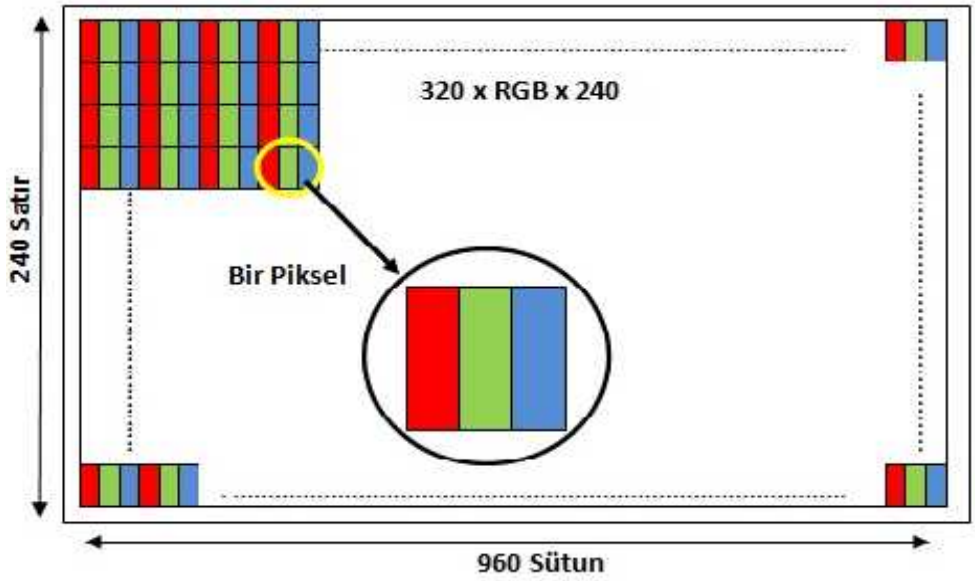
Hedef tanıma ve kilitlenme birimi tarafından tespit edilen hedefin koordinatları Sistem çıkış birimine aktarılır. V_Hedef ve H_hedef yazmaçlarındaki 9'ar bitlik konum bilgileri ikilik (binary) sistemde Hedef Koordinat Gösterge Modülüne aktarılır.

Bu modülde bulunan 17 adet led diyot hedefin koordinatlarını görsel olarak bize aktarır. Bu birime robotu kontrol eden diğer birimlerde bağlanabilir. Böylece hedef bilgilerini başka birimler tarafından kullanılabilir. Sistem çıkış birimi şekil-59'da ki gibidir.

Şekil 59: Sistem Veri Çıkış Birimi

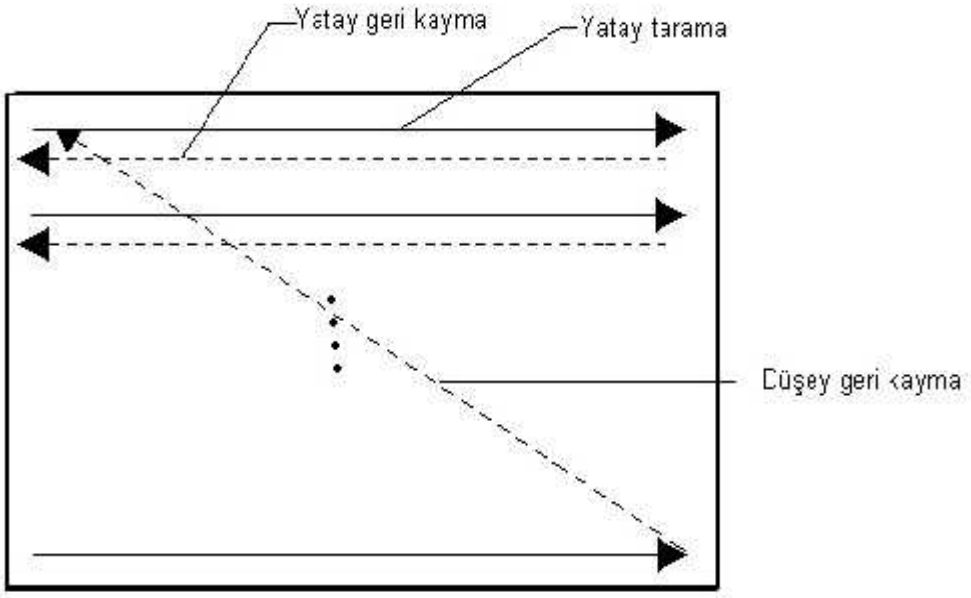
5.9 LCM Ekran Kontrol Birimi

Projemizde kullanmış olduğumuz Altera LCM TFT ekran 320 x RGB x 240 piksel boyutlarındadır. RGB formatında bir piksele ait üç renk değeri üst üste binerken TFT teknolojisinden dolayı bu ekranda üç renk değeri yan yana dizilmiştir. Şekilde TFT ekran renk dizlimi görülmektedir.



Şekil 60: TFT Ekran Piksel Dizilimi

Ekrana görüntü basma işlemi tarama şeklinde olur. İlk piksel ekranın sol kısmına basılır ve piksel basma işlemi ekranın sağ kısmına doğru devam eder. Ekranda satır sonuna gelindiğinde, bir alt satıra geçilir ve yukarıdaki satırdaki gibi pikseller soldan sağa doğru basılır. Aynı işlem Ekranın en alt kısmına gelinceye kadar tekrarlanır. Ekranın en alt kısmına gelindiğinde, tekrar ilk başlanılan noktaya geri dönülür ve aynı işlemler tekrarlanır.



Şekil 61: Ekran Tarama

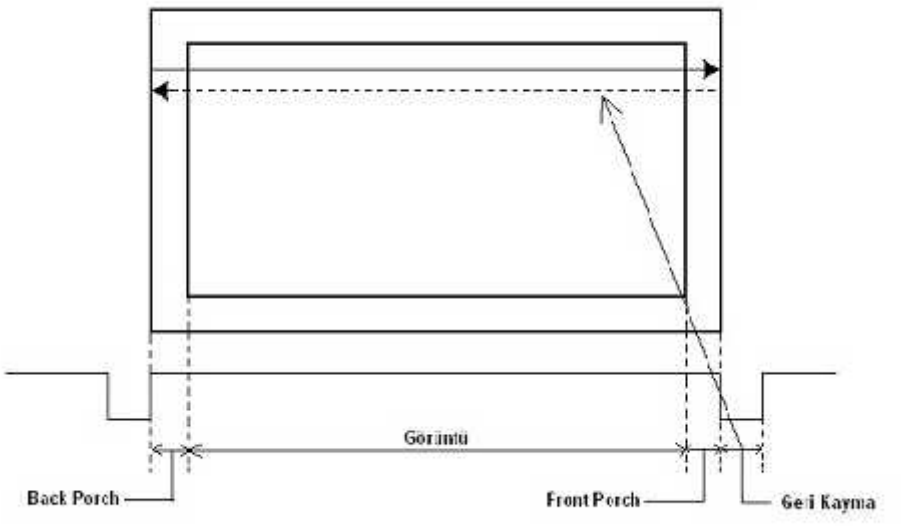
Ekran tarama işleminde satırın ya da resmin sonuna geldiğimizi anlamamıza yarayan sinyaller kullanılır. Bunlar yatay senkronizasyon (Horizontal synchronization) ve düşey senkronizasyon (Vertical synchronization) sinyalleridir.

Horizontal synchronization: Yatay konumda piksel işaretçisinin ekranın sol kısmına geldiğinde, işaretleyici bir alt satıra geçmesini sağlayan kontrol sinyalidir.

Vertical synchronization: Ekrana piksel basma olayı bittikten sonra piksel işaretçisinin tekrar ilk başlanılan noktaya geri dönmesini sağlayan kontrol sinyalidir.

5.9.1 Yatay Senkronizasyon

Ekranda piksellerin soldan sağa doğru tek satır halinde basılma işlemidir.



Şekil 62: Yatay Senkronizasyon

Görüntü: Görüntünün gösterildiği kısım.

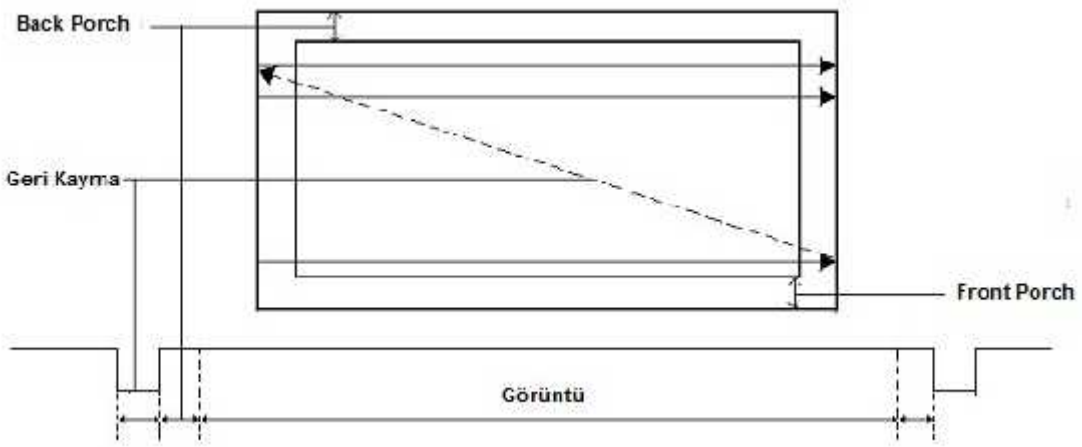
Geri kayma:Piksel işaretçisinin tekrar yatay olarak sol tarafa geldiği alan. Video sinyali bu kısımda ekrana basılmaması gerekir.

Front porch (Geri kaydırma işleminden önceki) Ekranın sağ sınır alanı olarak tanımlanır. Video sinyali bu kısımda ekrana basılmaması gerekir.

Back porch(Geri kaydırma işleminden sonraki) Ekranın sol sınır alanı olarak tanımlanır. Video sinyali bu kısımda ekrana basılmaması gerekir.

5.9.2 Dikey Senkronazisyon

Yatay tarama boyunca, piksel işaretçisi yukarıdan aşağı doğru hareket eder ve ekranın alt kısmına geldiği zaman tekrar yukarı döner. Bu işlem ile tüm ekran yenilenir.



Şekil 63: Dikey Senkronazisyon

Görüntü: Yatay çizgilerin ekranda gösterildiği kısım.

Geri kayma: Piksel işaretçisinin tekrar ekranın yukarı döndüğü kısım. Video sinyali bu kısımda ekrana basılmaması gerekir.

Front porch (Geri kaydırma işleminden önceki) Ekranın alt sınır alanı olarak tanımlanır. Video sinyali bu kısımda ekrana basılmaması gerekir.

Back porch(Geri kaydırma işleminden önceki) Ekranın üst sınır alanı olarak tanımlanır. Video sinyali bu kısımda ekrana basılmaması gerekir.

Sistemde kullanılan senkronizasyon değerleri Tablo 5'te verilmiştir.

Tablo 5: Senkronizasyon Değerleri

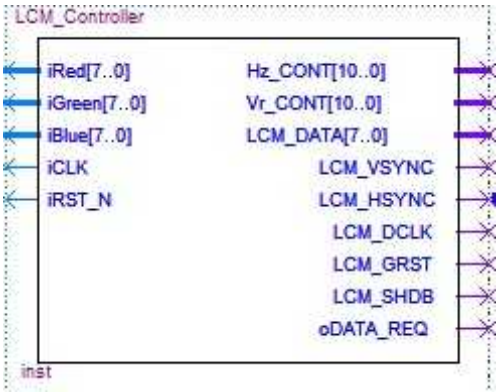
	Yatay	**Dikey**
Geri Kayma	1	1
Back Porch	151	13
Aktif Görüntü	960	240
Front Porch	59	8
Toplam	1171	262

Görüntümüzü oluşturan piksellerin RGB renk değerleri üst üste basılması gerekirken kullandığımız ekran türünden dolayı yan yana basılmaktadır. Dolayısıyla bu ekranda üç renk değeri sıra ile hızlı bir şekilde gönderilmelidir. İnsan gözü saniyede 25 kare ve üstü değişiklikleri algılayamadığı için tasarladığımız sistemin hızının bunun altına düşmemesi gerekmektedir. Sistemimiz 60 fps hızında çalıştığından ekran tekrarlama frekansını 60 Hz olarak belirledik. Ekran Saat sinyali aşağıdaki denklem ile hesaplanmıştır,

$$Toplam\ Piksel = 1171 \times 262 = 306802$$

$$Ekran\ sistem\ frekansı \quad f = 60 \times 306802 \approx 18.4\ MHz$$

Ekranda basılacak toplam piksel sayısı bir kare görüntü için gereken Saat sinyal frekansını belirler. Saniyede 60 kare görüntülemek istediğimiz için bu değeri 60 la çarparız böylece ekranımızın çalışacağı frekansı belirlemiş oluruz. Tasarlanan birim şekildeki gibidir.

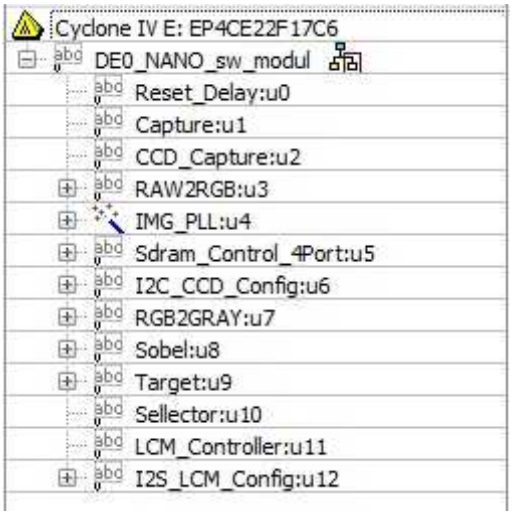


Şekil 64: LCM Ekran Kontrol Birimi

6. UYGULAMA

6.1 Kodların Derlenmesi

Verilog HDL dilinde Quartus II programı ile tasarlamış olduğumuz sistemin kod hiyerarşisi şekildeki gibidir.



Şekil 65: Kod Hiyerarşisi

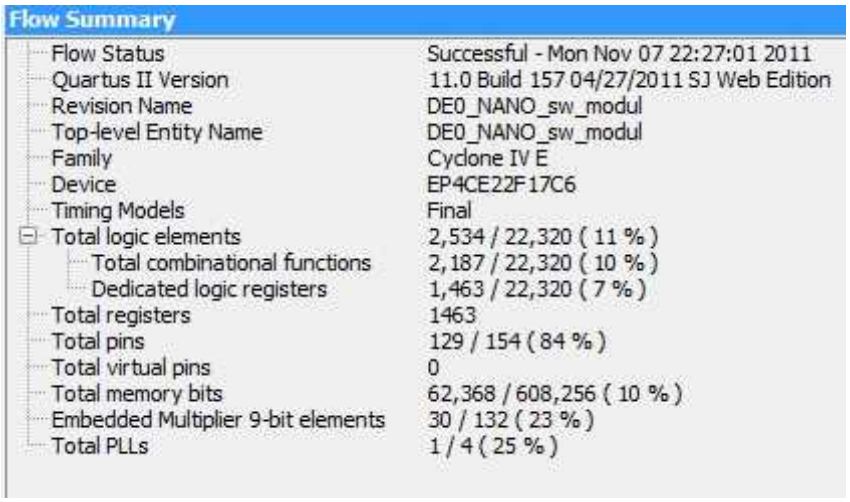
Sistem DE0_NANO_sw_modul.V isimli ana modül ve alt modüllerden meydana gelir.

Quartus II geliştirme ortamında tüm işlemler processing menüsünden erişilebilmektedir. Bazı işlemler ise kısayol olarak task ekranında da bulunmaktadır.

Projemiz üzerinde gerçekleştirebileceğimiz temel işlemler;

- Tam derleme,
- Analiz ve Detaylandırma,
- Analiz ve sentez,
- Fitter (Yerleştirme ve yönlendirme) işlemi,
- FPGA e yüklenecek dosyayı oluşturma (assembler) işlemidir.

Compile Design tuşuna tıklayarak yukarıdaki işlemlerin hepsini tek seferde gerçekleştiririz. Derleme raporu bize tasarlamış olduğumuz sistemde FPGA kaynaklarının kullanımı ve varsa hatalar ile ilgili bilgiler verir. Şekil-66'da sistem kaynaklarının kullanımı görülmektedir.



Şekil 66: Sistem Kaynaklarının Kullanımı

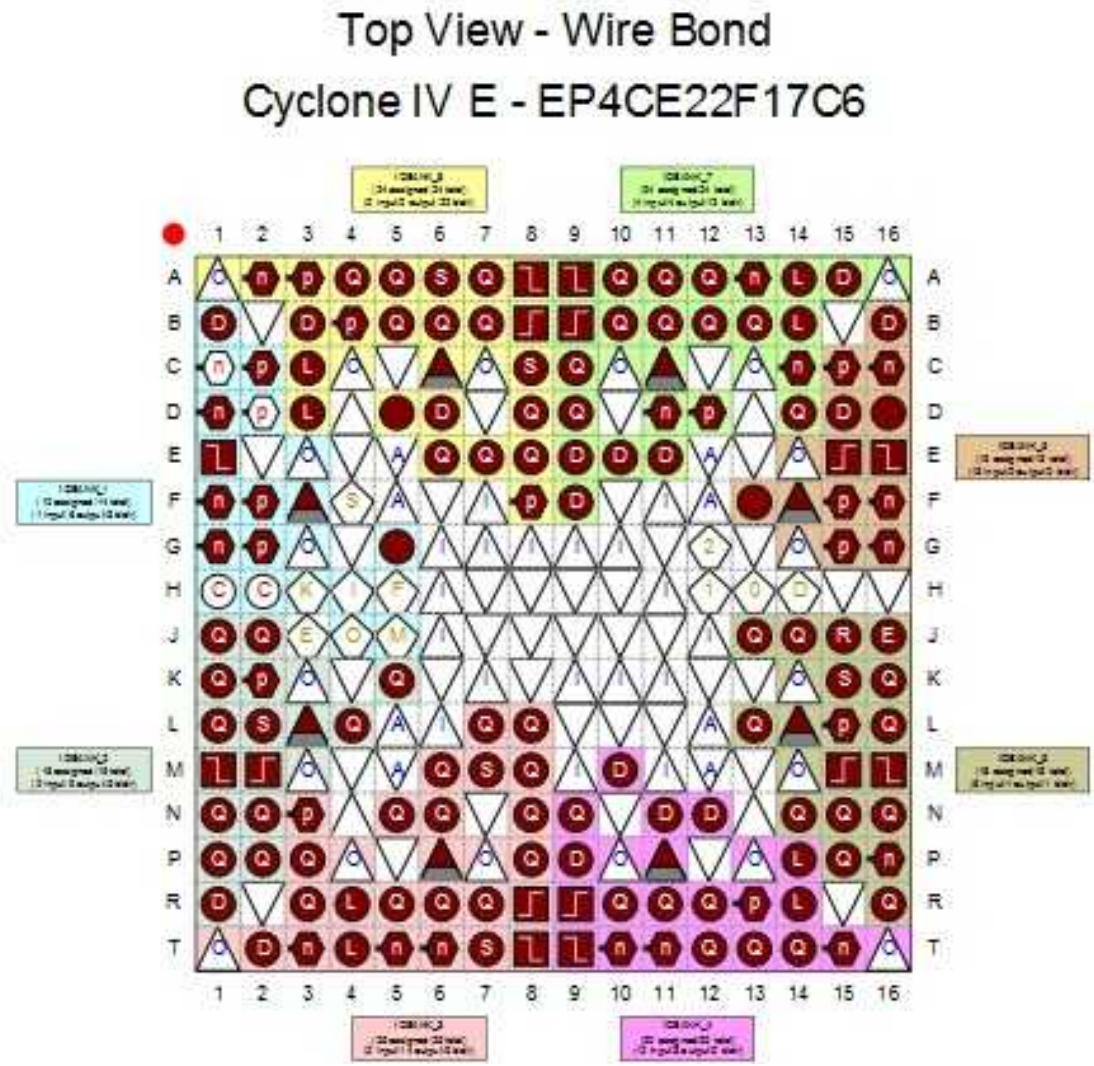
DE0_NANO FPGA platformumuz 22320 lojik elementten oluşmaktadır. Tasarlamış olduğumuz sistem 2534 lojik elementi ve 1463 hafıza yazmacını kullanmıştır. Sistem platform üzerinde hazır bulunan bir PLL bloğu ve 30 adet 9 bitlik çarpma bloğu kullanmıştır. FPGA bloğumuz genel amaçlı kullanım için 154 giriş çıkış portuna sahiptir. Biz bu portlardan 129 tanesini kullanmaktayız.

Derleme raporundan da anlaşıldığı üzere tasarlamış olduğumuz donanım sistem kaynaklarının sadece yüzde 10'unu kullanmaktadır. Bu da gösteriyor ki kullanmış olduğumuz donanımlar bu projenin daha fazla geliştirilmesi uygun olarak seçilmiştir.

6.2 Giriş Çıkış Pinlerinin Atanması

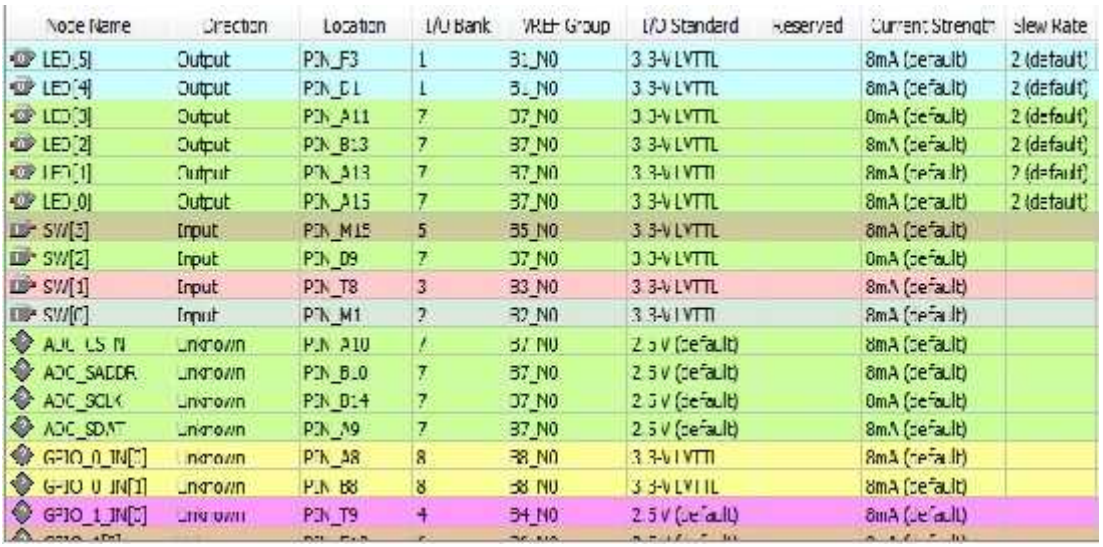
Yazmış olduğumuz kodlar derlendikten sonra FPGA üzerinde bir donanım haline geleceği için bu donanımın etrafındaki diğer donanımlarla olan bağlantılarda kullanacağı pinleri belirlememiz gerekmektedir. Bu işlemi assigment sekmesinde bulunan Import Assigment aracı ile gerçekleştiririz. Daha önceden

DE0_NANO_sw_modul.qsf dosyasına yazmış olduğumuz pin atamalarını Import Assigment aracı ile FPGA'e atıyoruz. Pin yönlendirmeleri ile ilgili değişiklikleri ya da ayarları Pin Planer aracı ile gerçekleştirebiliriz. FPGA üzerindeki pin atamalarımız şekildeki gibidir.



Şekil 67: Pin Planer Pin Atamaları

Pin atamaları ile ilgili daha detaylı bilgi Pin Planer aracının Renkli Pin listesinde bulunmaktadır. Bu liste üzerinde değişiklikte yapmak mümkündür. Sistemimizde kullanılan Pinlerle ilgili küçük bir bölümün görüntüsü şekil-68'deki gibidir.



Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate
LED[5]	Output	PIN_F3	1	B1_N0	3.3-V LVTTL		8mA (default)	2 (default)
LED[4]	Output	PIN_D1	1	B1_N0	3.3-V LVTTL		8mA (default)	2 (default)
LED[3]	Output	PIN_A11	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)
LED[2]	Output	PIN_B13	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)
LED[1]	Output	PIN_A13	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)
LED[0]	Output	PIN_A15	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)
SW[3]	Input	PIN_M15	5	B5_N0	3.3-V LVTTL		8mA (default)	
SW[2]	Input	PIN_D9	7	B7_N0	3.3-V LVTTL		8mA (default)	
SW[1]	Input	PIN_T8	3	B3_N0	3.3-V LVTTL		8mA (default)	
SW[0]	Input	PIN_M1	2	B2_N0	3.3-V LVTTL		8mA (default)	
ADC_CS_N	Unknown	PIN_B10	7	B7_N0	2.5 V (default)		8mA (default)	
ADC_SADDR	Unknown	PIN_B9	7	B7_N0	2.5 V (default)		8mA (default)	
ADC_SCLK	Unknown	PIN_D14	7	B7_N0	2.5 V (default)		8mA (default)	
ADC_SDAT	Unknown	PIN_A9	7	B7_N0	2.5 V (default)		8mA (default)	
GPIO_0_IN[0]	Unknown	PIN_A8	8	B8_N0	3.3-V LVTTL		8mA (default)	
GPIO_0_IN[1]	Unknown	PIN_B8	8	B8_N0	3.3-V LVTTL		8mA (default)	
GPIO_1_IN[0]	Unknown	PIN_T9	4	B4_N0	2.5 V (default)		8mA (default)	

Şekil 68: Pin Özellikleri

6.3 Kodların FPGA'ya Yüklenmesi

FPGA platformları bağlantıları her başlatıldıklarında yeniden gerçekleştirirler. Bu bağlantıları yapmak üzere FPGA bordu üzerinde USB Blaster isimli bir programlayıcı donanım mevcuttur. Bu işlemi yaparken üzerlerinde bulunan EEPROM da tutulan bağlantı bilgilerini kullanırlar. Sistem tasarım aşamasında ise her seferinde EEPROM'a bilgi yüklenmez. Sistem son haline gelene kadar geçici bağlantı kodları SRAM'a yüklenerek sistem çalıştırılır.

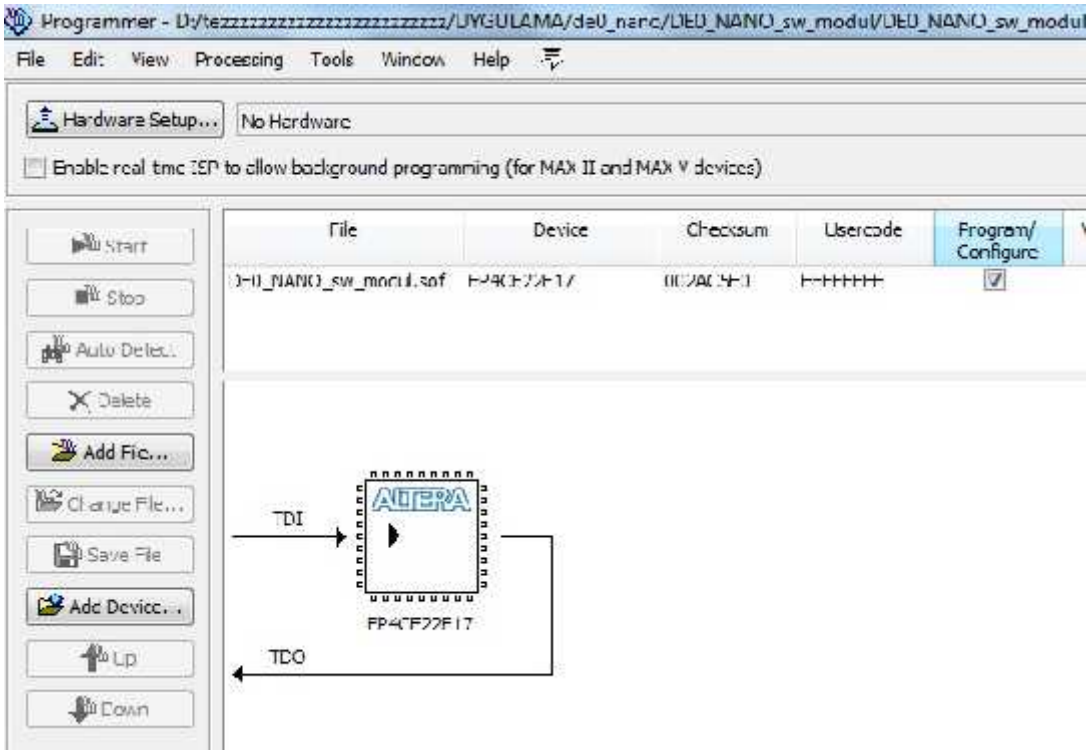
6.3.1 SRAM'a Geçici Kod Yükleme

Derleme işlemi sırasında oluşturulan SOF uzantılı kodlar Tools sekmesinde bulunan Programmer (Programlayıcı) aracı ile FPGA'de bulunan SRAM'a yüklenir.

Yükleme işlemi için öncelikle programmer aracı açılır. Add File tuşuna basılarak projemizin derlenmesi ile oluşan SOF uzantılı dosya seçilir. Alt tarafta çıkan FPGA modeli kontrol edilir, doğru ise Program/Configure kutucuğu işaretlenir. Hardware kutusu boş görünüyor ise Hardware Setup tuşuna basılarak USB Blaster

donanımı seçilir. Start tuşuna basılarak FPGA ‘e kodlar yüklenir ve FPGA içinde tasarlamış olduğumuz donanımın oluşması sağlanır.

Sistemin çalışması için herhangi bir işlem yapmaya gerek yoktur, programlama işlemi bittikten sonra sistem çalışmaya başlayacaktır. Programlama aracı ve ayarlar şekil-69’da ki gibidir.



Şekil 69: Geçici Programlama

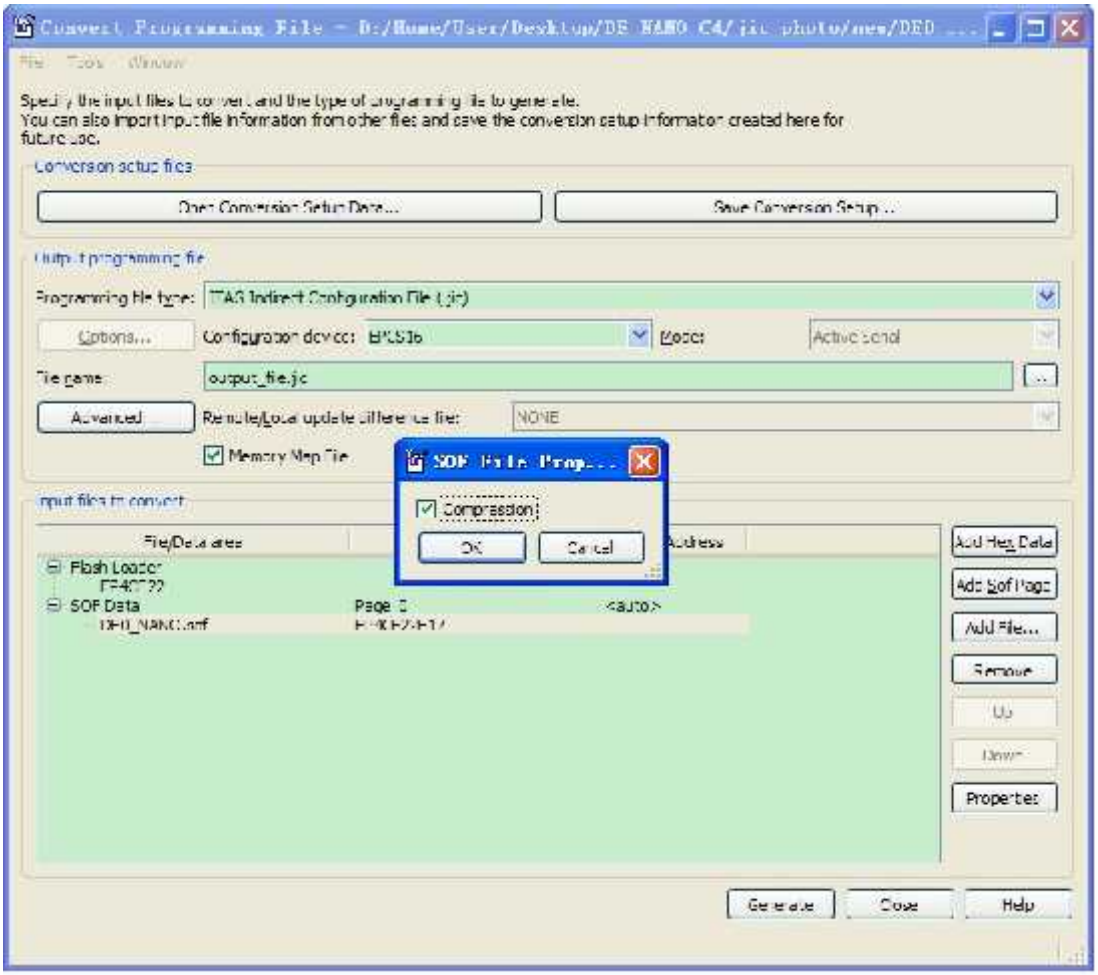
6.3.2 EEPROM’a Kalıcı Kod Yükleme

Sistem tasarımı son halini aldıktan sonra EEPROM’a yüklenecek olan kalıcı programlama dosyalarının üretilmesi gerekmektedir.

Bu işlem kullanmış olduğumuz donanım için şu şekilde gerçekleştirilir.

1. Dosya Menüsünden Convert Programming Files aracı seçilir.
2. Dosya tipi JTAG Indirect Configuration File (.jic) seçilir
3. **Configuration device** olarak EPCS16 seçilir.
4. **File Name** kısmına oluşturulacak yeni kodun ismi girilir.

5. **in the Input files to convert** alanında **SOF data** yazısına tıklayarak ADD Sof data tuşu aktif hale getilir.
6. **Add File** tıklanarak derleme işlemi sonrasında oluşturmuş olduğumuz sof uzantılı dosya eklenir.
7. **Flash Loader** yazısı tıklanarak ADD DEVICE tuşu aktif hale getirilir.
8. **ADD DEVICE** tuşu tıklanarak **EP4CE22** isimli cihaz seçilir.
9. **Sof** uzantılı dosya üzerine tıklayıp **Properties** tuşuna basılır. Gelen ekranda Compression kutucuğu işaretlenir.
10. **Generate** tuşuna basılarak **jic** uzantılı dosyamız oluşturulur.



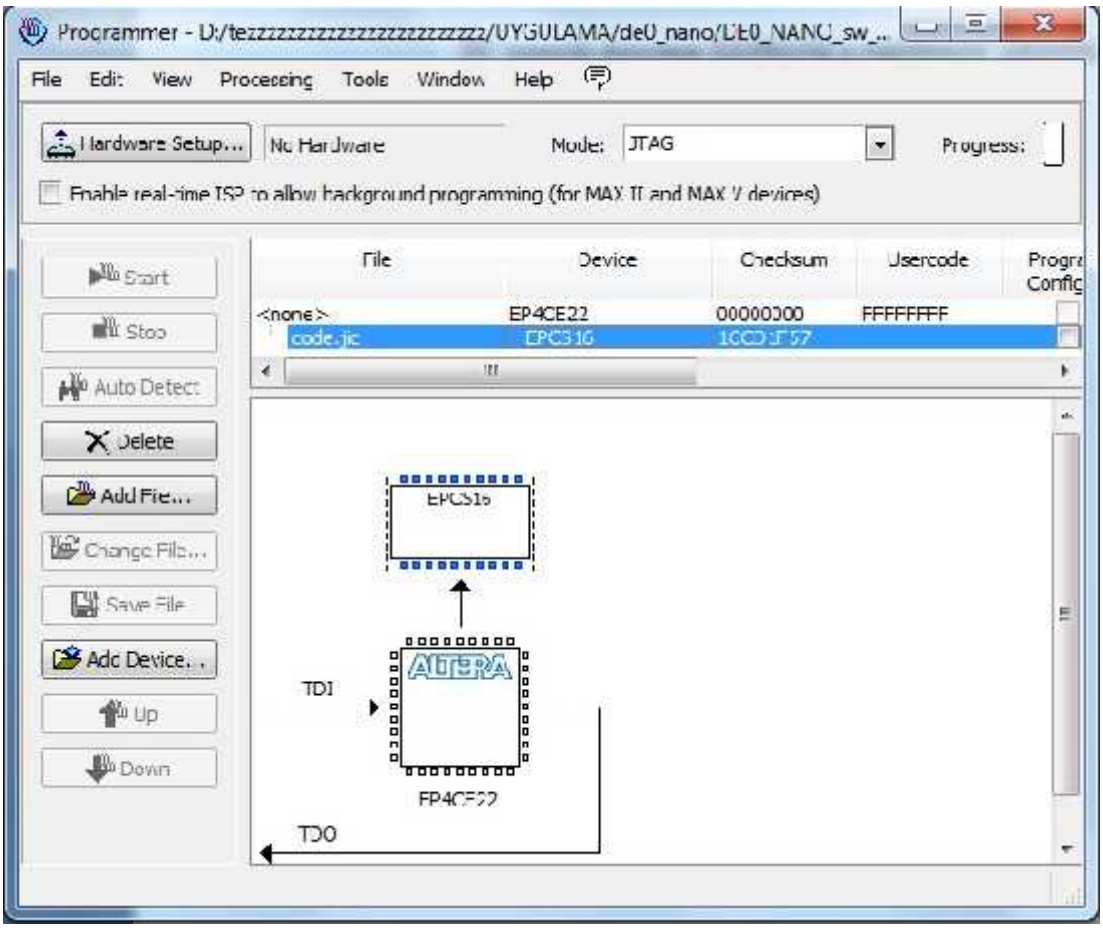
Şekil 70: Jic Uzantılı Dosya Üretme

Yukarıdaki işlemleri yaptıktan sonra FPGA platformu üzerinde bulunan EEPROM'a Jic uzantılı dosya aşağıdaki şekilde yüklenir.

Öncelikle programmer aracı açılır. Add File tuşuna basılarak oluşturduğumuz JIC uzantılı dosya seçilir. Alt tarafta çıkan EEPROM modeli kontrol edilir, doğru ise Program/Configure kutucuğu işaretlenir. DE0_NANO donanımında EPCS16 isimli EEPROM kullanılmıştır. Hardware kutusu boş görünüyor ise Hardware Setup tuşuna basılarak USB Blaster donanımı seçilir. Start tuşuna basılarak EEPROM'a kodlar yüklenir.

Bundan sonra FPGA'e her enerji verildiğinde EEPROM da bulunan kodlara göre bağlantıları gerçekleştirerek donanımın oluşmasını sağlar.

JIC uzantılı dosyanın EEPROM'a yüklenmesi şekil-71'deki gibidir.



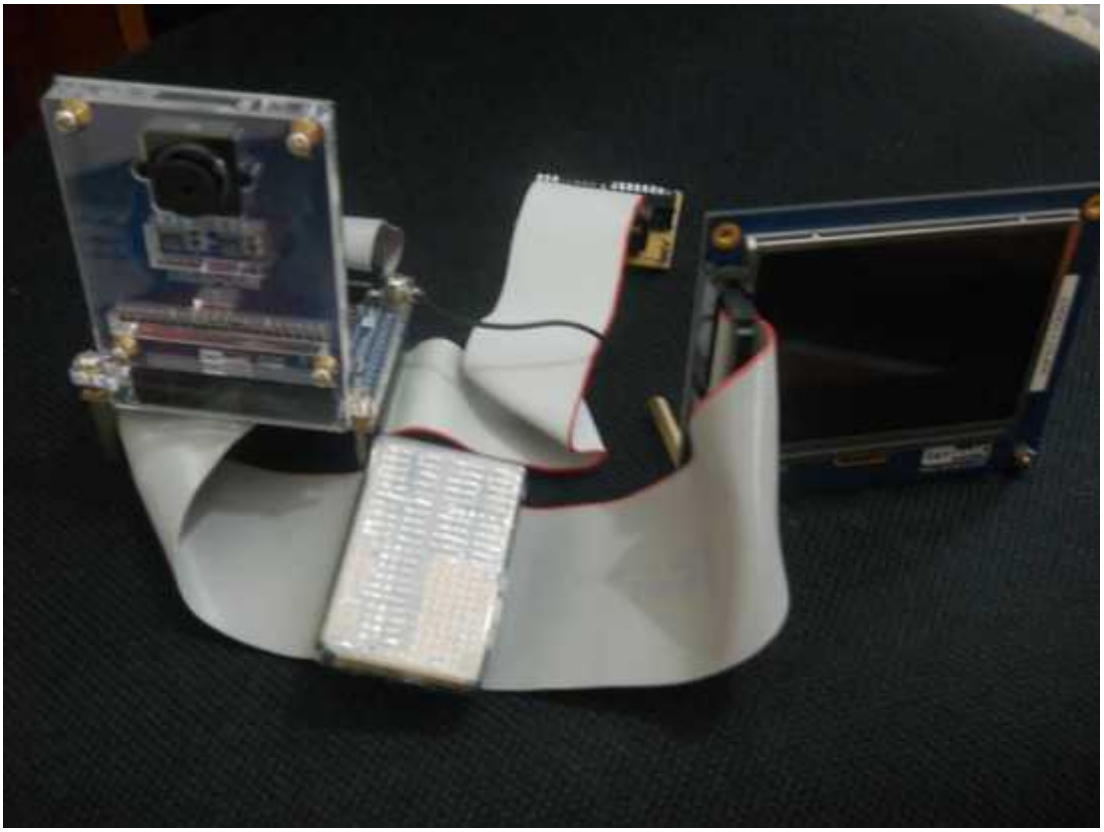
Şekil 71: EEPROM Programlama

6.4 Sistem Bağlantılarının Gerçekleştirilmesi

Kullanacağımız donanımlar sırasıyla DE0_NANO FPGA platformuna şu şekilde bağlanır.

- LCM ekran data kablosu DE0_NANO FPGA bordu üzerindeki GPIO_0 slotuna,
- D5M Kamera donanımı GPIO_1 slotuna,
- Switch_Modul donanımı GPIO_2 slotuna,
- Hedef Koordinat Gösterge Donanımı LCM ekran data kablosu üzerindeki paralel slota,
- Adaptör kablosu (-) uç kenarda kalacak şekilde FPGA bordunun besleme girişine takılır.

Sistem bağlantıları şekildeki gibidir.



Şekil 72: Robotik Göz Sistemi

6.5 Sistemin Çalıştırılması

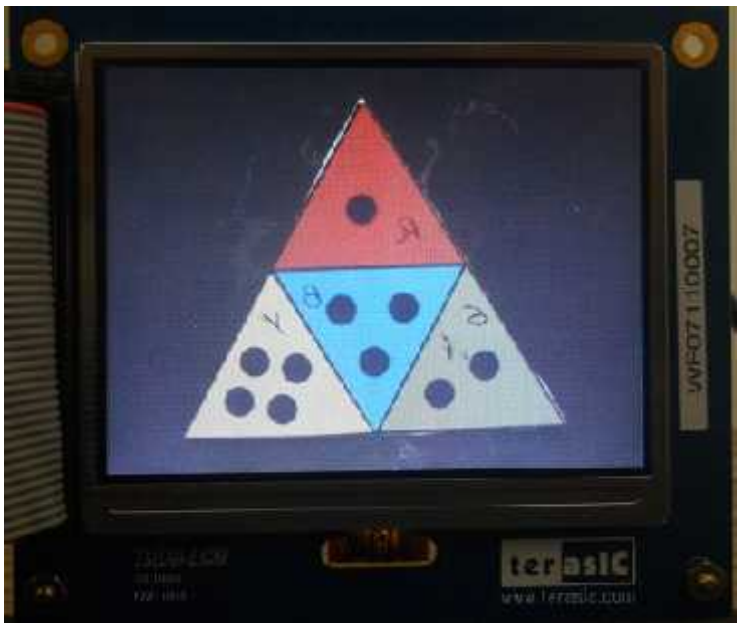
Sistemin besleme gerilimini sağlamak amacıyla tasarlanan adaptörün kablosunun uç kısmında bir adet açma kapama anahtarı ve 5V regülatör devresi bulunmaktadır. Açma kapama anahtarı ile sisteme enerji verilerek çalıştırılır. Sistemde yapılacak kontroller Switch_Modul devresinde bulunan 16 adet (SW0 ve SW1 bloğunda) anahtar ile yapılacaktır. SW0 bloğu ile çalışma modlarını seçerken SW1 bloğu ile eşik değerlerini belirleyebiliriz.

6.5.1 Gerçek Görüntü

Görüntü sensöründen alınan renk bilgilerinin RAW formatından RGB formatına dönüştürüldükten sonra hiçbir işlem yapılmadan çıkışa aktarıldığı moddur.

Bu mod için anahtar konumları şekildeki gibidir.

Şekil 73: Gerçek Görüntü Mod Seçimi



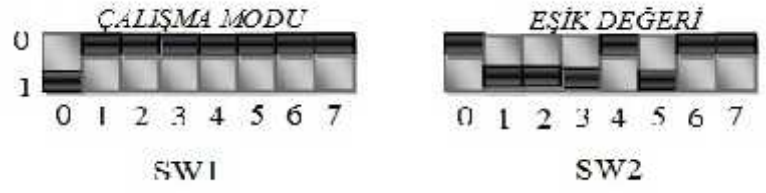
Şekil 74: Gerçek Görüntü

6.5.2 RGB Filtreleri

Tasarlamış olduğumuz sistemin robotik uygulamalarda kullanımı esnasında RGB filtrelerine ihtiyaç duyulabileceği düşünülerek kırmızı , yeşil ve mavi renk filtreleri de sistem üzerinde oluşturulmuştur.

6.5.2.1 Kırmızı Filtresi

Filtre kameradan gelen kırmızı görüntü bilgilerinde eşik değerinden büyük olan değerleri 255'e, küçük olan değerleri 0'a çeker. Böylece görüntü içinde eşik değerinden büyük kırmızı barındıran nesneler ekranda görünür diğerleri görünmez. Bu mod için gereken switch ayarları aşağıdaki gibidir. Eşik değeri istenildiği gibi girilebilir.



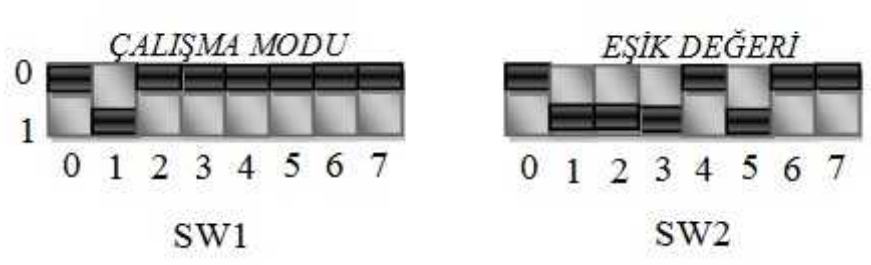
Şekil 75: Kırmızı Filtresi Mod Seçimi

Filtrelenmiş görüntü şekildeki gibidir. Örnek resimde sarı renkteki pikseller de görünmektedir. Bunun nedeni sarı rengin yeşil ve kırmızı renklerin karışımı ile oluşmasıdır.

Şekil 76: Kırmızı Filtreli görüntü

6.5.2.2 Yeşil Filtresi

Kırmızı renk filtresinde olduğu gibi kameradan gelen yeşil görüntü bilgilerinde eşik değerinden büyük olan değerleri 255'e, küçük olan değerleri 0'a çeker. Böylece ortamda içinde eşik değerinden büyük yeşil barındıran nesneler ekranda görünür diğerleri görünmez. Bu mod için gereken switch ayarları aşağıdaki gibidir. Eşik değeri istenildiği gibi girilebilir.



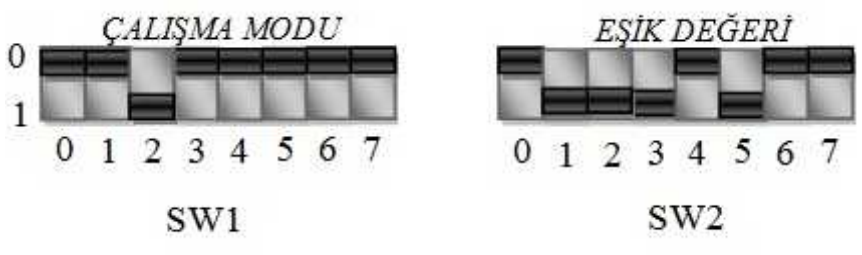
Şekil 77: Yeşil Filtresi Mod Seçimi

Filtrelenmiş görüntü şekil-77 deki gibidir. Örnek resimde sarı renkte görülmektedir. Bunun nedeni sarı rengin yeşil ve kırmızı renklerin karışımı ile oluşmasıdır.

Şekil 78: Yeşil Renk Filtreli Görüntü

6.5.2.3 Mavi Filtresi

Diğer renk filtresinde olduğu gibi kameradan gelen mavi görüntü bilgilerinde eşik değerinden büyük olan değerleri 255'e, küçük olan değerleri 0'a çeker. Böylece ortamda içinde eşik değerinden büyük mavi renk içeren nesneler ekranda görünür diğerleri görünmez. Bu mod için gereken switch ayarları aşağıdaki gibidir. Eşik değeri istenildiği gibi girilebilir.



Şekil 79: Mavi Filtresi Mod Seçimi

Filtrelenmiş görüntü şekildeki gibidir. Ekranda sarı renk görünmemektedir. Bunun nedeni sarı rengin bileşenleri arasında mavi bulunmamasıdır.

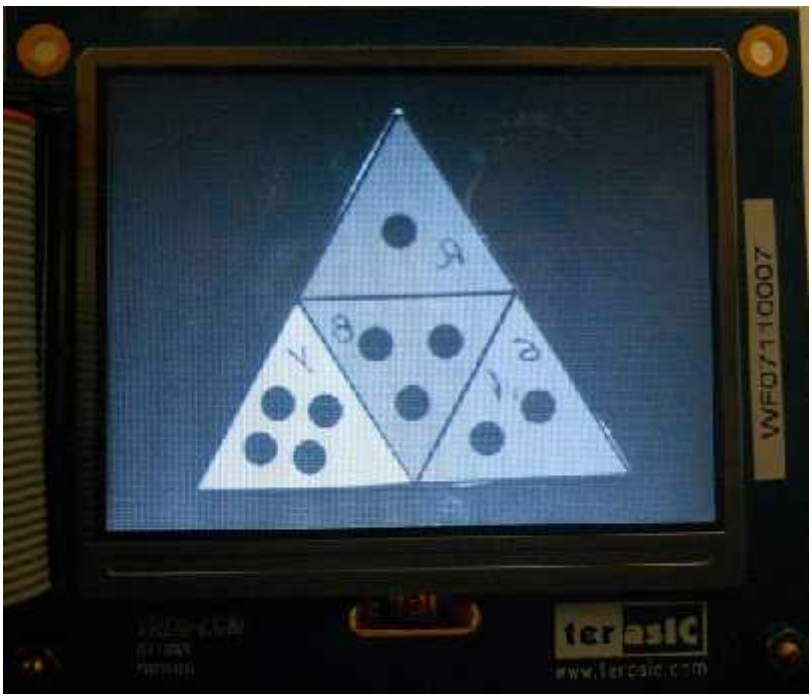
Şekil 80: Mavi Renk Filtreli Görüntü

6.5.3 Gri Filtre Uygulaması

Bu filtre kırmızı, yeşil ve mavi renk değerlerinin bir paralel toplayıcıda toplanıp 3'e bölünmesi ile oluşur. Robotik uygulamalarda kullanılabileceği düşünülerek sistem modları arasına eklenmiştir. Eşik değeri ile herhangi bir ilişkisi olmadığından sadece SW1 anahtar bloğu kullanılır.

Şekil 81: Gri Filtresi Mod Seçimi

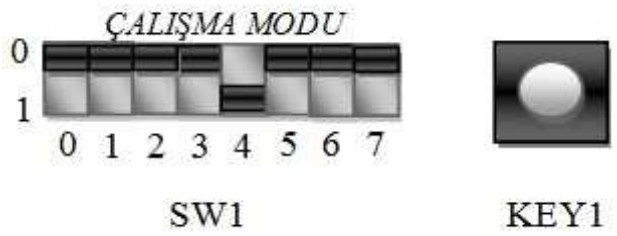
Filtrelenmiş görüntü şekil-82'deki gibidir.



Şekil 82: Gri Filtreli Görüntü

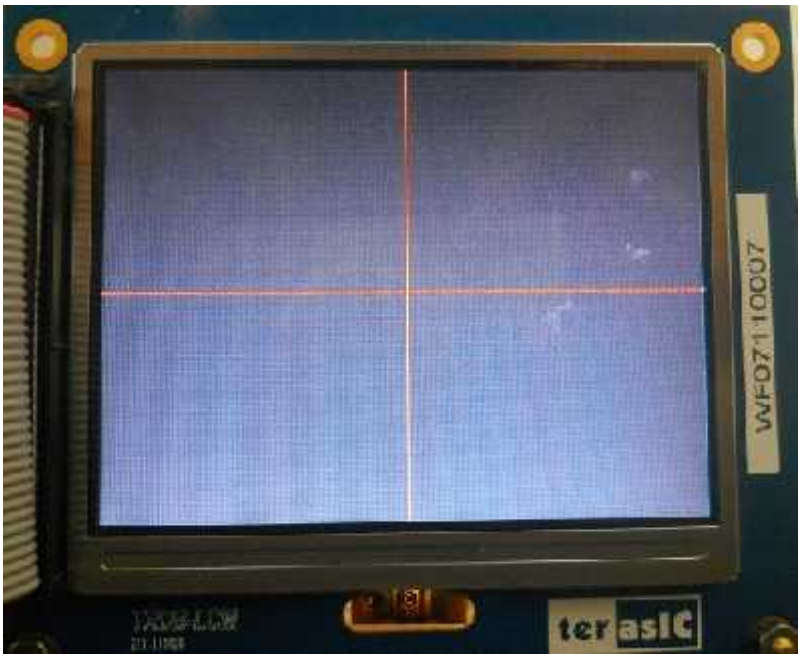
6.5.4 Hedef Algılama ve Takip Modu

Sistem bu modda kendisine tanımlanan hedefi ekrandaki imleç yardımı ile takip etmemizi sağlar ve hedefin koordinatlarını hedef gösterge modülüne gönderir. Hedef olarak belirlenen nesne ekranın tam ortasında kesişen imlece gelecek şekilde kameranın önüne tutulur. Key1 tuşuna basarak hedef hakkındaki bilgiler gerekli yazmaca yazılır. Sistem kameradan gelen veriler arasında hedefi tanımlayan veriler gelince imleci o koordinata basar ve hedef koordinatlarını Hedef Gösterge modülüne gönderir. Bu işlem her resim karesinde yeniden yapılır. Bu modla ilgili ayarlar şekilde gösterilmiştir.



Şekil 83: Hedef Algılama Mod Seçimi

Bu modun çalışma aşamaları aşağıdaki şekillerde gösterilmiştir.



Şekil 84: Hedef Bekleme

Şekil 85: Hedef Kayıt

Şekil 86: Hedef Takip

Hedef algılama ve takip modülünce led gösterge modülüne gönderilen koordinat bilgileri şekildeki gibi ledler yardımı ile görüntülenebilmektedir.

Şekil 87: Led Modül Koordinat Görünümü

Bu örnek uygulamada hedefin koordinatlarını ledlere bakarak çözersek; X ekseninin (111010101)2 = 469, Y ekseninin ise (01000010)2 = 66 olduğu görülür.

6.5.5 Konvolüsyon Uygulaması

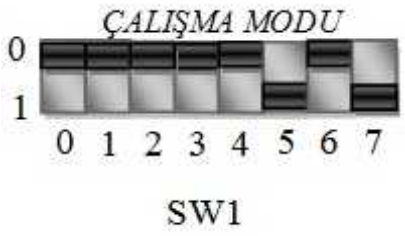
Tasarlamış olduğumuz sistemde görüntü işleme çalışmalarının en önemli noktalarından biri olan konvolüsyon işlemi de yer almaktadır.

Görüntü işleme filtrelerinden birçoğunda konvolüsyon işlemi kullanılmaktadır. Bu filtrelerde kullanılan çekirdek matrisler çok çeşitlidir. Çalışmamızda 3 adet filtreye yer ayırdık. Bunlar Laplace , Emboss ve robotik görüntü işleme uygulamalarında çok önemli bir filtre olan Sobel filtreleridir. Sistemdeki çekirdek matrisler değiştirilerek daha farklı filtrelerde tasarlanabilir.

Kullanmış olduğumuz filtrelerden Laplace ve Sobel filtreleri kenar belirleme filtreleridir. Emboss filtresi ise değişik bir filtrenin kullanımını göstermek amacı ile sisteme dahil edilmiştir.

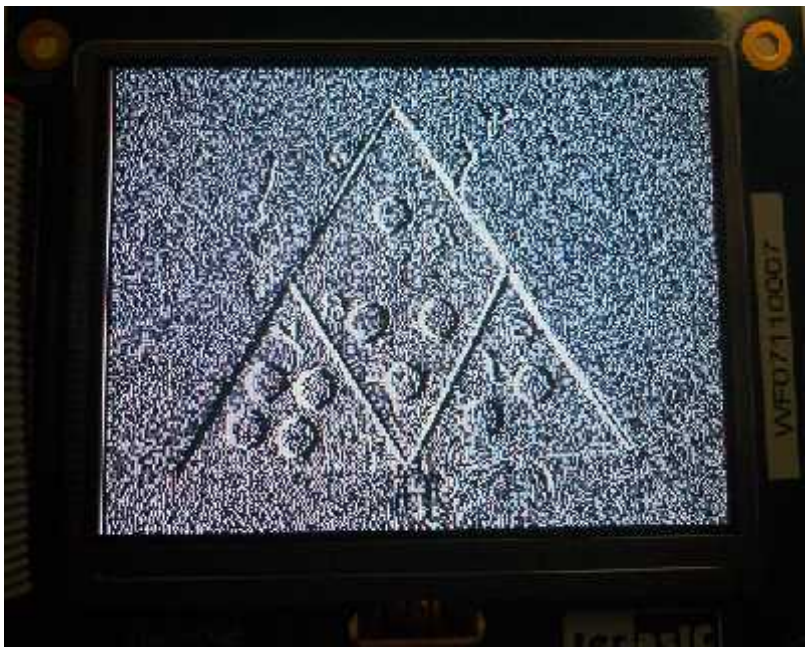
6.5.5.1 Emboss Filtresi Uygulaması

Emboss (kabartma) filtresi modu için SW1 durumları şekildeki gibi olmalıdır.



Şekil 88: Emboss Filtresi Mod Seçimi

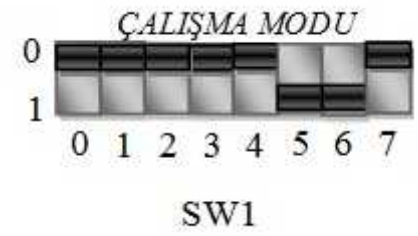
Kabartma Filtresi uygulanmış ekran görüntüsü şekildeki gibidir.



Şekil 89: Emboss Filtresi Uygulaması

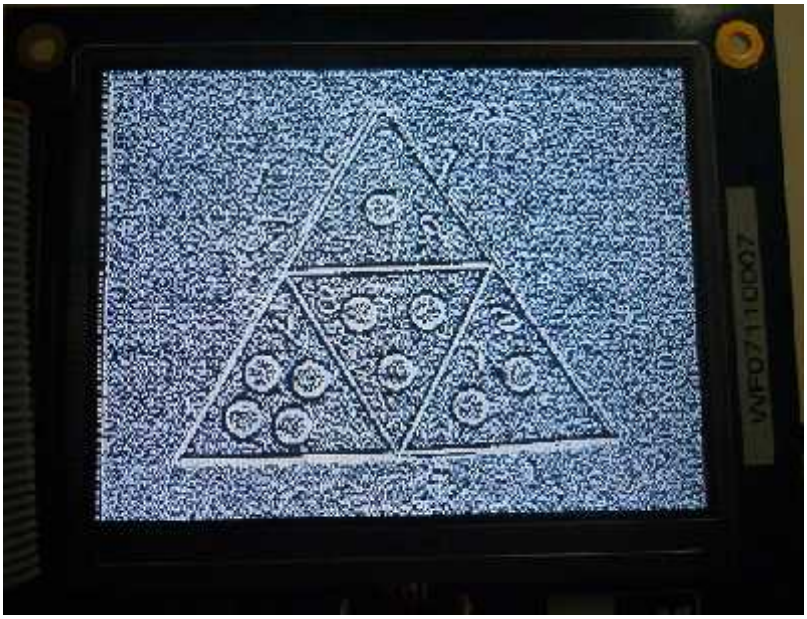
6.5.5.2 Laplace Filtresi Uygulaması

Laplace filtresi modu için SW1 durumları şekildeki gibi olmalıdır.



Şekil 90: Laplace Filtresi Mod Seçimi

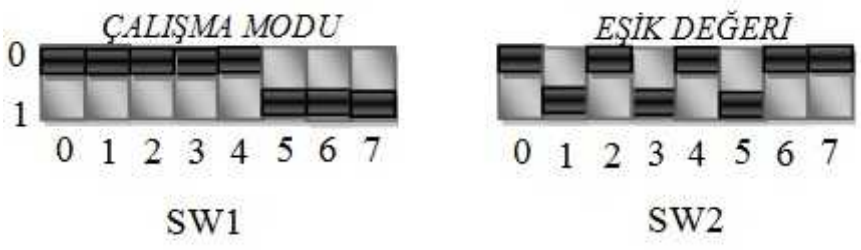
Laplace Filtresi uygulanmış ekran görüntüsü şekildeki gibidir.



Şekil 91: Laplace Filtresi Uygulaması

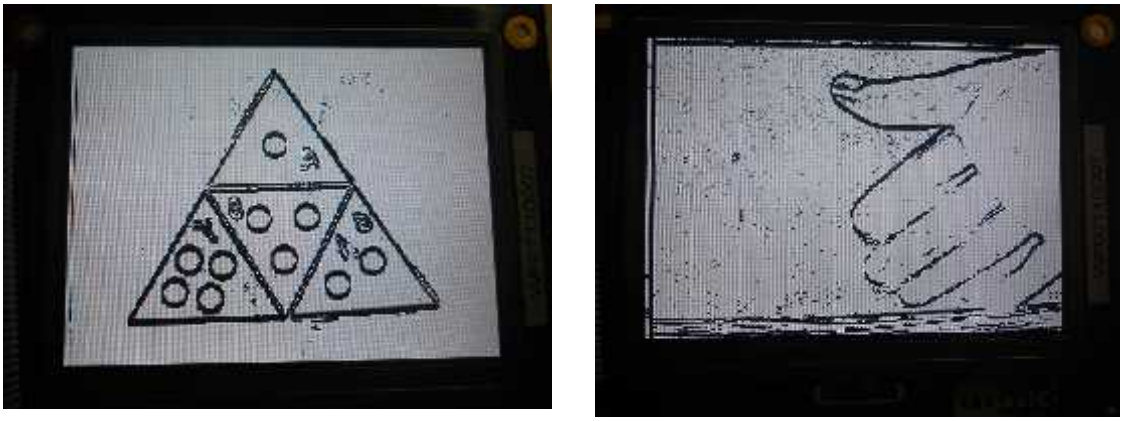
6.5.5.3 Sobel Filtresi Uygulaması

Robotik görme ve Nesne tanıma algoritmalarında en çok kullanılan filtreleme metotlarındandır. Sistem üzerinde Sobel Filtresi çalıştırma ayarları şekildeki gibidir.



Şekil 92: Sobel Modu Seçme Ayarları

Sobel Filtresi uygulanmış iki farklı görüntü şekilde gösterilmiştir.



Şekil 93: Sobel Filtresi Uygulaması

7. SONUÇ VE ÖNERİLER

Bu çalışmada görüntü sensöründen gelen renk bilgileri üzerinde gerçek zamanlı görüntü işleme, hedef tanıma ve takip algoritmaları gerçekleştiren bir donanım tasarlanmıştır. Hedef koordinatlarının farklı sistemlerde de kullanabilmesi için bir çıkış birimi tasarlanmıştır. Böylece tasarlanan donanımın robotik uygulamalardaki diğer donanım ve sistemlerle haberleşebilmesi sağlanmıştır.

Sistem robotik uygulamalarda kullanılabilecek boyutlarda tasarlanmıştır. FPGA kaynaklarının sadece %11'inin kullanması ileride yapılacak geliştirme ve yeni tasarımlara olanak sağlamaktadır.

Yapılacak çalışmalarda sistem üzerinde bulunan konvolüsyon ve kenar belirleme birimleri kullanılarak yüz tanıma, nesne tanıma, plaka ve yazı okuyabilme gibi uygulamalara altyapı oluşturmaktadır.

Bu çalışmanın devamı olarak daha önce bölüm laboratuarlarında yapılmış olan İnsansı Robot projesine montajı ve adaptasyonu planlanmaktadır.

8. KAYNAKLAR

[1] Cobun, T.C., "Media and Public School Communication" , (4 aralık 2011), http://www.khaos.info/bilimsel-mevzular/5598-nasil-ogreniyoruz/, (1998).

[2] Mehra,R., Sardana, R. B. and Verma, R.,"FPGA Based Area Efficient Edge Detection Filter for Image Processing Applications", *International Journal of VLSI and Signal Processing Applications,* 1(2) , 38-41 , (2011).

[3] SIN, C. S.," Median, Sobel, And Minimum Filter Architectures On Field", MSc THESIS, *Universiti Teknologi Faculty of Electrical Engineering*, Malaysia, (2010).

[4] Hammes, J., B¨ohm, A.P.W., Ross, C. , Chawathe, M., Draper, B. and Najjar ,W., "High Performance Image Processing on FPGAs", *Colorado State University DARPA Research Laboratory,* l, USA (2001).

[5] Johnston, C. T., Gribbon, K. T. and Bailey, D. G., "Implementing Image Processing Algorithms on FPGAs", *Massey University Institute of Information Sciences & Technology*, New Zelland , (2005).

[6] Brown, S. and Rose, J., "Architecture of FPGAs and CPLDs", *University of Toronto Department of Electrical and Computer Engineering*, l , (2000).

[7] Yeniçeri, R., "Cmos Görüntü Sensörü ve Fpga İle Sayısal Fotoğraf Makinesi Gerçeklenmesi", Lisans Tezi, *İTÜ Elektrik Elektronik Fakültesi*, İstanbul, (2007).

[8] ÖZCAN, A.R., "Gerçek Zamanlı Lineer Görüntü İşleme Algoritmalarının Fpga İle Gerçeklenmesi", Lisans Tezi, *Yıldız Teknik Üni. Elektrik Elektronik Fak* ültesi, İstanbul, (2009).

[9] Allen, T.G., Luettgen, M.R. and Whisky, A. S., "Multiscale approaches to moving target detection in image sequences", *Optical Engineering*, 33(7), (1994).

[10] Manolakis, D., Marden, D., and Shaw, G. A., "Hyperspectral Image Processing for Automatic Target Detection Applications", *Lincoln Laboratory Journal*, 14(1), (2003).

[11] Ovod, V.I., Baxter, C.R. and Massie, M. A., "Advanced Image Processing Package for FPGA-Based Re-Programmable Miniature Electronics", *Presented at SPIE*,5783-32,Orlando, (2005).

[12]Bernd ,J.," *Practical Handbook on Image Processing for Scientific Applications*", l, CRC Press, (1997).

[13] Haberaecker, P., " Digitale Bilverarbeitung" , l : Hanser, (1991).

[14] Russ, J. C., "*The Image Processing Hand Book*", l, CRC Press, (1999).

[15] Yağımlı, M., "Renk Bileşenleri Yardımıyla Hareketli Hedeflerin Gerçek Zamanlı Tespiti", *Journal of Naval Science and Engineering*, 5(2), 89-97, (2009).

[16] Sainath, S., and Sarkar, S., "An Approximate Algorithm for Structural Matching of Images", *IEEE*, 0-8186-8821-1/98, (1998).

[17] Mojsiloviç, A., Hu, J., "A Method for Color Content Matching of Images", *IEEE*, 0-7803-6536-4, (2000).

[18] Huang, X., Zhang, S., Wang, G. and Wang,H.,"Optimal Matching of Images Using Combined Color Feature and Spatial Feature", *Computational Science-ICCS PT, 1*, 411-418, (2006).

[19] Srinivasan, S. H. and Kankanhalli, M., "Wide Baseline Spectral Matching", *IEEE*, 0-7803-7965-9, (2003).

[20] Corp., Altera., "TRDB-D5M Hardware specification",(12 Aralık 2011), http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=281#section, (2009).

[21] Inc., Terasic, Technologies, "DE2_115 User Manual", (12 Aralık 2011), http://www.altera.com/products/devkits/http//de2-115.terasic.com, (2011).

[22] Corp., Altera, DE0_NANO User manual", (12 Aralık 2011), http://de0-nano.terasic.com/,(2011).

[23] Corp., Altera, "TRDB_LCM_UserGuide",(12 Aralık 2011), http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=68&No=78, (2006).

EKLER

9. EKLER

EK-A Robotik Göz Sistemi Kodları

DE0_NANO_sw_modul.V

Not: Çok yer kapladığından dolayı sadece ana program kodu burada verilmiştir. Alt modüllere ait kodlar ekteki CD de mevcuttur.

```
// Modul:  DE0_NANO_sw_modul
// ---------------------------------------------------------------------
//   Ver  :| Author              :| Mod. Date
//   V1.0 :| Mustafa TASCI       :| 07/11/11
// ---------------------------------------------------------------------
module DE0_NANO_sw_modul(
            ////////// CLOCK //////////
            CLOCK_50,
            ////////// LED //////////05426781754
            LED,
            ////////// KEY //////////
            KEY,
            ////////// SW //////////
            SW,
            ////////// SDRAM //////////
            DRAM_ADDR,
            DRAM_BA,
            DRAM_CAS_N,
            DRAM_CKE,
            DRAM_CLK,
            DRAM_CS_N,
            DRAM_DQ,
            DRAM_DQM,
            DRAM_RAS_N,
            DRAM_WE_N,
            ////////// Accelerometer and EEPROM //////////
            G_SENSOR_CS_N,
            G_SENSOR_INT,
            I2C_SCLK,
            I2C_SDAT,
            ////////// GPIO_0, GPIO_0 connect to LTM - 4.3" LCD and Touch //////////
            GPIO_0,
//GPIO_0_IN,
////////// 2x13 GPIO Header //////////
            GPIO_2,
            GPIO_2_IN,
////////// GPIO_0, GPIO_1 connect to D5M - 5M Pixel Camera //////////
            D5M_D,
            D5M_FVAL,
            D5M_LVAL,
            D5M_PIXCLK,
            D5M_RESET_N,
            D5M_SCLK,
            D5M_SDATA,
            D5M_STROBE,
            D5M_TRIGGER,
            D5M_XCLKIN
);

//=======================================================
```

```
//  PORT declarations
//=======================================================
////////// CLOCK //////////
input                       CLOCK_50;
////////// LED //////////
output          [7:0]       LED;
////////// KEY //////////
input           [1:0]       KEY;
////////// SW //////////
input           [3:0]       SW;
////////// SDRAM //////////
output          [12:0]      DRAM_ADDR;
output          [1:0]       DRAM_BA;
output                      DRAM_CAS_N;
output                      DRAM_CKE;
output                      DRAM_CLK;
output                      DRAM_CS_N;
inout           [15:0]      DRAM_DQ;
output          [1:0]       DRAM_DQM;
output                      DRAM_RAS_N;
output                      DRAM_WE_N;
////////// Accelerometer and EEPROM //////////
output                      G_SENSOR_CS_N;
input                       G_SENSOR_INT;
output                      I2C_SCLK;
inout                       I2C_SDAT;
////////// GPIO_0, GPIO_0 connect to LTM - 4.3" LCD and Touch //////////
inout           [33:0]      GPIO_0;
//inout         [1:0]       GPIO_0_IN;
////////// GPIO_0, GPIO_1 connect to D5M - 5M Pixel Camera //////////
input           [11:0]      D5M_D;
input                       D5M_FVAL;
input                       D5M_LVAL;
input                       D5M_PIXCLK;
output                      D5M_RESET_N;
output                      D5M_SCLK;
inout                       D5M_SDATA;
input                       D5M_STROBE;
output                      D5M_TRIGGER;
output                      D5M_XCLKIN;
////////// 2x13 GPIO Header //////////
input           [12:0]      GPIO_2;
input           [2:0]       GPIO_2_IN;
//////////////////////    For TFT LCD Module      //////////////////////
wire    [7:0]   LCM_DATA;                   //      LCM Data 8 Bits
wire                LCM_GRST;               //      LCM Global Reset
wire                LCM_SHDB;               //      LCM Sleep Mode
wire                LCM_DCLK;               //      LCM Clcok
wire                LCM_HSYNC;              //      LCM HSYNC
wire                LCM_VSYNC;              //      LCM     VSYNC
wire                LCM_SCLK;               //      LCM I2C Clock
wire                LCM_SDAT;               //      LCM I2C Data
wire                LCM_SCEN;               //      LCM I2C Enable
wire                CLK_18;
wire [11:0] H_hedef;
wire [8:0] V_hedef ;
wire  [10:0]    H_C;
wire  [10:0]    V_C;
//assign    GPIO_0_IN[0]    =       H_Led[0];
assign      GPIO_0[25]      =       V_Led[0];
assign      GPIO_0[0]       =       H_Led[0];
assign      GPIO_0[1]       =       H_Led[1];
assign      GPIO_0[2]       =       V_Led[1];
assign      GPIO_0[3]       =       H_Led[2];
assign      GPIO_0[4]       =       V_Led[2];
assign      GPIO_0[5]       =       H_Led[3];
assign      GPIO_0[6]       =       V_Led[3];
assign      GPIO_0[7]       =       H_Led[4];
assign      GPIO_0[8]       =       V_Led[4];
assign      GPIO_0[9]       =       H_Led[5];
assign      GPIO_0[10]      =       V_Led[5];
assign      GPIO_0[11]      =       H_Led[6];
assign      GPIO_0[12]      =       V_Led[6];
assign      GPIO_0[13]      =       H_Led[7];
assign      GPIO_0[14]      =       V_Led[7];
assign      GPIO_0[15]      =       H_Led[8];
```

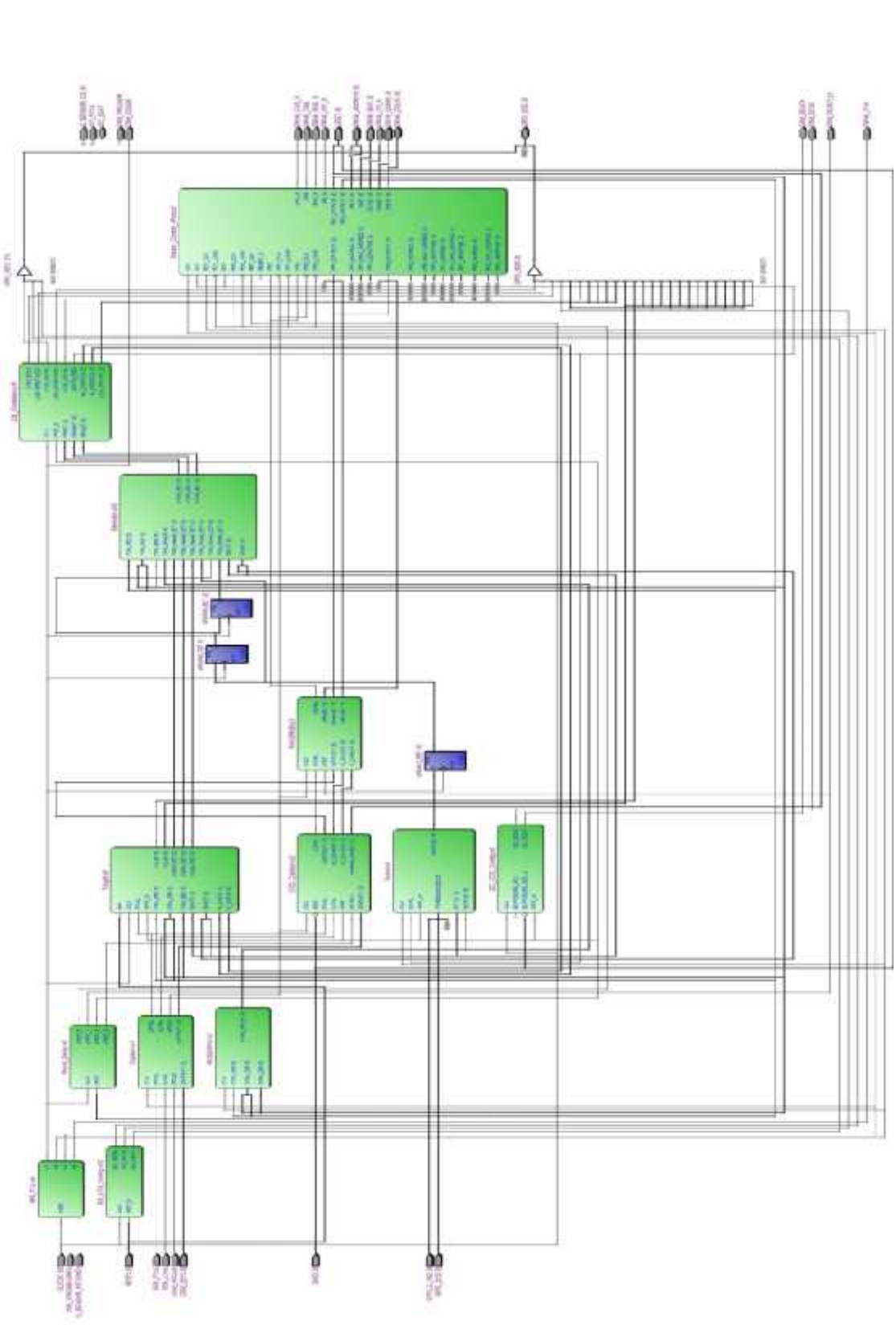
```
assign      GPIO_0[16]          =          LCM_DATA[6];
assign      GPIO_0[17]          =          LCM_DATA[7];
assign      GPIO_0[18]          =          LCM_DATA[4];
assign      GPIO_0[19]          =          LCM_DATA[5];
assign      GPIO_0[20]          =          LCM_DATA[2];
assign      GPIO_0[21]          =          LCM_DATA[3];
assign      GPIO_0[22]          =          LCM_DATA[0];
assign      GPIO_0[23]          =          LCM_DATA[1];
assign      GPIO_0[24]          =          LCM_VSYNC;
assign      GPIO_0[26]          =          LCM_SCLK;
assign      GPIO_0[27]          =          LCM_DCLK;
assign      GPIO_0[28]          =          LCM_GRST;
assign      GPIO_0[29]          =          LCM_SHDB;
assign      GPIO_0[31]          =          LCM_SCEN;
assign      GPIO_0[32]          =          LCM_SDAT;
assign      GPIO_0[33]          =          LCM_HSYNC;
//=============================================================
//  REG/WIRE declarations
//=============================================================
assign SW_M1[0] = GPIO_2[12];
assign SW_M1[1] = GPIO_2[11];
assign SW_M1[2] = GPIO_2[10];
assign SW_M1[3] = GPIO_2[9];
assign SW_M1[4] = GPIO_2[8];
assign SW_M1[5] = GPIO_2[7];
assign SW_M1[6] = GPIO_2[6];
assign SW_M1[7] = GPIO_2[5];
assign SW_M2[0] = GPIO_2[4];
assign SW_M2[1] = GPIO_2[3];
assign SW_M2[2] = GPIO_2[2];
assign SW_M2[3] = GPIO_2[1];
assign SW_M2[4] = GPIO_2[0];
assign SW_M2[5] = GPIO_2_IN[2];
assign SW_M2[6] = GPIO_2_IN[1];
assign SW_M2[7] = GPIO_2_IN[0];
wire   [7:0] SW_M1;
wire   [7:0] SW_M2;
wire         [15:0]      X_Cont;
wire         [15:0]      Y_Cont;
wire         [31:0]      Frame_Cont;
wire         [11:0]      rCCD_DATA;
wire                                 rCCD_LVAL;
wire                                 rCCD_FVAL;
wire         [11:0]      mCCD_DATA;
wire                                 mCCD_DVAL;
wire                                 mCCD_DVAL_d;
wire         [11:0]      sCCD_R;
wire         [11:0]      sCCD_G;
wire         [11:0]      sCCD_B;
wire                                 sCCD_DVAL;
wire      wDVAL_sobel;
reg                                  Pre_rCCD_FVAL;
wire         [15:0]      Read_DATA1;
wire         [15:0]      Read_DATA2;
wire                     Read;
wire                     sdram_ctrl_clk;
wire                     d5m_clk;
wire                     ltm_clk;
wire                     ltm_clk_n;
wire                     LTM_CTRL_CLK;
wire                     rCCD_PCLK;
wire                     DLY_RST_0;
wire                     DLY_RST_1;
wire                     DLY_RST_2;
wire                     DLY_RST_3;
wire [9:0] wVGA_R = Read_DATA2[9:0];
wire [9:0] wVGA_G = {Read_DATA1[14:10],Read_DATA2[14:10]};
wire [9:0] wVGA_B = Read_DATA1[9:0];
wire [7:0] yVGA_R;
wire [7:0] yVGA_G;
wire [7:0] yVGA_B;
wire [9:0] wSobel;
reg [7:0] wSobel_R;
reg [7:0] wSobel_G;
reg [7:0] wSobel_B;
wire [7:0] wDISP_R;
```

```
wire [7:0] wDISP_G;
wire [7:0] wDISP_B;
wire [9:0] wGray;
wire  [8:0] V_Led;
wire  [8:0] H_Led;
//=============================================================
//  Structural coding
//=============================================================
assign      LED[3:0]            =           SW;
assign      LED[7:4]            =Frame_Cont[5:2];
assign      D5M_XCLKIN          =ltm_clk;
assign      D5M_RESET_N         =DLY_RST_1;
assign      D5M_TRIGGER         =1'b1;
assign      LTM_NCLK            =ltm_clk_n;
assign      LTM_CTRL_CLK        =ltm_clk;
assign      LTM_GREST           =KEY[0];
assign      DRAM_CLK            =sdram_clk;
reg         gec                 = 0;
reg [15:0]  gec_say             = 0;


Reset_Delay             u0          (
            .iCLK(ltm_clk),
            .iRST(KEY[0]),
            .oRST_0(DLY_RST_0),
            .oRST_1(DLY_RST_1),
            .oRST_2(DLY_RST_2),
            .oRST_3(DLY_RST_3)
            );


Capture      u1(
            .iCLK (sdram_ctrl_clk),
            .iDATA (D5M_D),
            .iLVAL (D5M_LVAL),
            .iFVAL (D5M_FVAL),
            .iPCLK (D5M_PIXCLK),
            .oDATA   (rCCD_DATA),
            .oLVAL (rCCD_LVAL),
            .oFVAL (rCCD_FVAL),
            .oPCLK (rCCD_PCLK)
            );

CCD_Capture          u2(
            .oDATA(mCCD_DATA),
            .oDVAL(mCCD_DVAL),
            .oX_Cont(X_Cont),
            .oY_Cont(Y_Cont),
            .oFrame_Cont(Frame_Cont),
            .iDATA(rCCD_DATA),
            .iFVAL(rCCD_FVAL),
            .iLVAL(rCCD_LVAL),
//          .iSTART(!KEY[1]),
            .iSTART(DLY_RST_3),
            .iEND(!SW[0]),
            .iCLK(rCCD_PCLK),
            .iRST(DLY_RST_2)
            );

RAW2RGB             u3(
            .iCLK(rCCD_PCLK),
            .iRST(DLY_RST_1),
            .iDATA(mCCD_DATA),
            .iDVAL(mCCD_DVAL),
            .oRed(sCCD_R),
            .oGreen(sCCD_G),
            .oBlue(sCCD_B),
            .oDVAL(sCCD_DVAL),
            .iX_Cont(X_Cont),
            .iY_Cont(Y_Cont)

            );

IMG_PLL             u4(
            .inclk0(CLOCK_50),
            .c0(d5m_clk),
```

```
                .c1(ltm_clk),
                .c2(ltm_clk_n),
                .c3(sdram_ctrl_clk),
                .c4(sdram_clk)
                );
Sdram_Control_4Port                u5(
                 .REF_CLK(CLOCK_50),
                 .RESET_N(1'b1),
                .CLK(sdram_ctrl_clk),
                .WR1_DATA({1'b0,sCCD_G[11:7],sCCD_B[11:2]}),
                .WR1(sCCD_DVAL),
                .WR1_ADDR(0),
                .WR1_MAX_ADDR(320*240),
                .WR1_LENGTH(9'h100),
                .WR1_LOAD(!DLY_RST_0),
                .WR1_CLK(~rCCD_PCLK),
                .WR2_DATA(          {1'b0,sCCD_G[6:2],sCCD_R[11:2]}),
                .WR2(sCCD_DVAL),
                .WR2_ADDR(22'h100000),
                .WR2_MAX_ADDR(22'h100000+320*240),
                .WR2_LENGTH(9'h100),
                .WR2_LOAD(!DLY_RST_0),
                .WR2_CLK(~rCCD_PCLK),
                 .RD1_DATA(Read_DATA1),
                .RD1(Read),
                .RD1_ADDR(0),
                .RD1_MAX_ADDR(320*240),
                .RD1_LENGTH(9'h100),
                .RD1_LOAD(!DLY_RST_0),
                .RD1_CLK(ltm_clk_n),
                 .RD2_DATA(Read_DATA2),
                .RD2(Read),
                .RD2_ADDR(22'h100000),
                .RD2_MAX_ADDR(22'h100000+320*240),
                .RD2_LENGTH(9'h100),
                .RD2_LOAD(!DLY_RST_0),
                .RD2_CLK(ltm_clk_n),
                .SA(DRAM_ADDR),
                .BA(DRAM_BA),
                .CS_N(DRAM_CS_N),
                .CKE(DRAM_CKE),
                .RAS_N(DRAM_RAS_N),
                .CAS_N(DRAM_CAS_N),
                .WE_N(DRAM_WE_N),
                .DQ(DRAM_DQ),
                .DQM(DRAM_DQM)
                );

I2C_CCD_Config     u6(
                .iCLK(CLOCK_50),
                .iRST_N(DLY_RST_2),
                .iEXPOSURE_ADJ(1'b1),
                .iEXPOSURE_DEC_p(!SW[3]),
                .I2C_SCLK(D5M_SCLK),
                .I2C_SDAT(D5M_SDATA)
                );

RGB2GRAY  u7(
                .iCLK (ltm_clk_n),
                .iVGA_R (wVGA_R),
                .iVGA_G (wVGA_G),
                .iVGA_B (wVGA_B),
                .oVGA_GRY (wGray)
                );

Sobel u8 (
  .iCLK(ltm_clk_n),
  .iRST_N(DLY_RST_2),
  .iTHRESHOLD({SW_M2[7:0],2'b11}),
  .iFLT (SW_M1[7:6]),
  .iDVAL(Read),
  .iDATA(wGray), // gray
  .oDVAL(wDAL_sobel),
  .oDATA(wSobel)
                );
```

```
always@(posedge ltm_clk)
begin
        wSobel_B <=     wSobel_G;
        wSobel_G <=     wSobel_R;
        wSobel_R <=     wSobel [9:2];
end

Target u9(
        .iCLK (ltm_clk),
        .iRST_N(DLY_RST_2),
        .iFVAL (rCCD_FVAL),
        .VGA_R (wVGA_R),
        .VGA_G (wVGA_G),
        .VGA_B (wVGA_B),
        .Btn  (KEY[1]),
        .SW1 (SW_M1),
        .SW2 (SW_M2),
        .H_Cnt (H_C),
        .V_Cnt (V_C),
        .V_L  (V_Led),
        .H_L  (H_Led),
        .V_hedef (V_hedef),
        .H_hedef (H_hedef),
        .hVGA_R (yVGA_R),
        .hVGA_G (yVGA_G),
        .hVGA_B (yVGA_B)
        );

Sellector u10 (
        .VGA_R (wVGA_R),
        .VGA_G (wVGA_G),
        .VGA_B (wVGA_B),
        .VGA_Gray (wGray),
        .VGA_Hedef_R (yVGA_R),
        .VGA_Hedef_G (yVGA_G),
        .VGA_Hedef_B (yVGA_B),
        .VGA_Sobel_R (wSobel_R),
        .VGA_Sobel_G (wSobel_G),
        .VGA_Sobel_B (wSobel_B),
        .SW1 (SW_M1),
        .SW2 (SW_M2),
        .oVGA_R (wDISP_R),
        .oVGA_G (wDISP_G),
        .oVGA_B (wDISP_B)
        );

LCM_Controller      u11(
        .iRed(wDISP_R),
        .iGreen(wDISP_G ),
        .iBlue(wDISP_B ),
        .Hz_CONT(H_C),
        .Vr_CONT(V_C),
        .LCM_DATA(LCM_DATA),
        .LCM_VSYNC(LCM_VSYNC),
        .LCM_HSYNC(LCM_HSYNC),
        .LCM_DCLK(LCM_DCLK),
        .LCM_SHDB(LCM_SHDB),
        .LCM_GRST(LCM_GRST),
        .oDATA_REQ(Read),
        .iCLK(LTM_CTRL_CLK),
        .iRST_N(DLY_RST_2)
        );

I2S_LCM_Config    u12(
        .iCLK(CLOCK_50),
        .iRST_N(KEY[0]),
                .I2S_SCLK(LCM_SCLK),
        .I2S_SDAT(LCM_SDAT),
        .I2S_SCEN(LCM_SCEN)
        );


Endmodule
```

EK-B FPGA Sitemi Blok Şeması



Şekil 94: Sistem RTL Diyagram

EK-C Tasarlanan Sistem Görüntüleri

Şekil 95: DE2-115 FPGA ile Kırmızı Filtresi

Şekil 96: DE2-115 FPGA ile Emboss Filtresi

Şekil 97: DE2-115 FPGA ile Sobel Filtresi

Şekil 98 İnsansı Robota Bağlanmış Robotik Göz