GERÇEK ZAMANLI BİR GÖRÜNTÜ İŞLEME SİSTEMİNİN MODEL TABANLI OLARAK TASARLANMASI

YÜKSEK LİSANS TEZİ

Mustafa Yusuf DEMİRCİ

Danışman
Dr. Öğr. Üyesi İsmail YABANOVA

ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

Haziran 2018

AFYON KOCATEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

GERÇEK ZAMANLI BİR GÖRÜNTÜ İŞLEME SİSTEMİNİN MODEL TABANLI OLARAK TASARLANMASI

Mustafa Yusuf DEMİRCİ

Danışman
Dr. Öğr. Üyesi İsmail YABANOVA

ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Haziran 2018

TEZ ONAY SAYFASI

Mustafa Yusuf DEMİRCİ tarafından hazırlanan "Gerçek Zamanlı Bir Görüntü İşleme Sisteminin Model Tabanlı Olarak Tasarlanması" adlı tez çalışması lisansüstü eğitim ve öğretim yönetmeliğinin ilgili maddeleri uyarınca 25/06/2018 tarihinde aşağıdaki jüri tarafından **oy birliği** ile Afyon Kocatepe Üniversitesi Fen Bilimleri Enstitüsü **Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Dr. Öğr. Üyesi İsmail YABANOVA

		İmza
Başkan	: Prof. Dr. Ahmet ALTUNCU Dumlupınar Üniversitesi Mühendislik Fakültesi	.....
Üye	: Prof. Dr. Hasan ÇİMEN Afyon Kocatepe Üniversitesi Teknoloji Fakültesi	.....
Üye	: Dr. Öğrt. Üyesi İsmail YABANOVA Afyon Kocatepe Üniversitesi Teknoloji Fakültesi	.....

Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...../...../..... tarih ve
..... sayılı kararıyla onaylanmıştır.

.....
Prof. Dr. İbrahim EROL
Enstitü Müdürü

BİLİMSEL ETİK BİLDİRİM SAYFASI
Afyon Kocatepe Üniversitesi

Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

25/06/2018

İmza
Mustafa Yusuf DEMİRCİ

ÖZET
Yüksek Lisans Tezi

GERÇEK ZAMANLI BİR GÖRÜNTÜ İŞLEME SİSTEMİNİN MODEL TABANLI OLARAK TASARLANMASI

Mustafa Yusuf DEMİRCİ
Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi İsmail YABANOVA

Görüntü işleme sistemleri günümüzde birçok alanda sıklıkla kullanılmaktadır. Bu sistemlerin önemli bir kısmı sistemin doğası yapısı gereği gerçek zamanlı olarak çalışmak zorundadır. Bu tip sistemlerdeki gerçek zamanlı çalışma ihtiyacı yüksek işlem gücü gerektirdiğinden uygulamaya özel gömülü donanım ve yazılım tasarımları kullanma gereği doğabilmektedir.

Bu sebeplerden gerçek zamanlı çalışacak olan gömülü sistem tasarımlarda Alan Programlanabilir Kapı Dizileri (FPGA) sıklıkla kullanılmaktadır. Son yıllarda yarıiletken teknolojisinin de gelişimiyle birlikte FPGA ve mikro işlemciyi aynı tümleşik devre içerisinde bir araya getiren sistemler kullanıma sunulmuştur. Bu sayede tümüyle programlanabilir ve klasik sistemlere göre çok daha düşük güç ve alan tüketimi olan sistemler geliştirilebilmektedir. Ancak bu tip sistemlerin önemli bir dezavantajı hem donanım hem de yazılım tasarımının yapılması gerekmesi ve donanım tasarımının ileri düzeyde sayısal tasarım bilgisi ve tecrübesi gerektirmesidir.

Bu durumu aşmak için geliştirilen yüksek seviyeli sentez ve tasarım araçları ile donanım tanımlama dilleri kullanılmadan da donanım tasarımları yapılabilmektedir. Bu tez çalışmasında Xilinx Vivado Design Suite yazılımı ile birlikte yeni nesil yüksek seviyeli bir programlama aracı olan Matlab/Simulink geliştirme ortamlarında model tabanlı tasarım yöntemleri kullanılarak gerçek zamanlı bir görüntü işleme sistemi tasarlanmıştır.

Çalışmada Xilinx Zynq-7000 mimarisine sahip Zedboard geliştirme kartı kullanılmıştır. Kameradan anlık olarak alınan görüntü üzerinde kenar bulma, gürültü giderme, gri ton dönüştürme ve keskinleştirme filtreleri uygulanarak işlenmiş olan görüntü monitöre aktarılmıştır.

Çalışmada kullanılan Zynq platformu klasik FPGA mimarisinden farklı olarak mikroişlemci ve FPGA'dan oluşan tümleşik bir sistem mimarisi olduğundan öncelikle bu sistem mimarisi ve model tabanlı tasarım metotları hakkında detaylı bilgi verilmiştir. Sonraki bölümlerde uygulanacak olan görüntü işleme yöntemleri ve bunların gerçek zamanlı video üzerinde uygulanışı hakkında gerekli bilgiler verilmiştir. Daha sonra bu görüntü işleme ve filtre algoritmaları Matlab/Simulink ortamında model tabanlı olarak tasarlanıp simülasyonu yapılmış, ardından Xilinx geliştirme ortamında tasarlanan görüntü işleme sisteminin donanım entegrasyonu ve yazılım tasarımı yapılarak Zedboard üzerinde sistem gerçeklenmiş ve kaynak kullanımı izlenmiştir. Yapılan çalışmalar neticesinde sistemin gerçek zamanlı olarak başarıyla çalıştığı gözlemlenmiştir.

2018, xiii + 75 sayfa

Anahtar Kelimeler: Görüntü İşleme, Gerçek zamanlı sistemler, FPGA, Zynq, Model Tabanlı Tasarım, Gömülü Sistemler

ABSTRACT
M.Sc. Thesis

MODEL BASED IMPLEMENTATION OF A REAL-TIME IMAGE PROCESSING SYSTEM

Mustafa Yusuf DEMİRCİ
Afyon Kocatepe University
Graduate School of Natural and Applied Sciences
Department of Electrical and Electronics Engineering
Supervisor: Asst. Prof. İsmail YABANOVA

Image processing systems are frequently used in many fields today. A significant portion of these systems have to work in real time due to the nature of system. Because of the need for real-time operation on such systems requires high processing power, it is possible to use application-specific embedded hardware and software designs.

For these reasons, the Field Programmable Gate Arrays (FPGA) are often used in embedded system designs that will work in real-time. In recent years, with the development of semiconductor technology, the use of FPGA and microprocessor systems in the same integrated circuit has been presented. Systems that are fully programmable and have much lower power and space consumption than classical systems can be developed in this way. However, a major disadvantage of such systems is the need for both hardware and software design, and advanced design knowledge and experience in hardware design.

High-level synthesis and design tools are developed to overcome this situation which can be used to design hardware without using hardware description languages. In this thesis study, a real time image processing system is designed by using model based design methods in Matlab / Simulink environment which is a new generation high level programming tool with Xilinx Vivado Design Suite.

Zedboard development board with Xilinx Zynq-7000 architecture is used in the study. The processed image is transferred to the monitor by applying edge detection, noise reduction, grayscale conversion and sharpening filters on the image taken from the camera instantaneously.

The Zynq platform used in this study is an integrated system architecture which consists both microprocessor and FPGA what makes it different from the classical FPGA architecture. Because of that, the system architecture and the model based design methods are given in detail first. The following sections provide information on image processing methods and how they are applied to real-time video. Then, these image processing and filter algorithms were designed and simulated with model-based tools in Matlab / Simulink environment, then hardware integration and software design of the image processing system designed in Xilinx development environment were implemented and system usage was analyzed on Zedboard. As a result of the studies done, it has been observed that the system worked successfully in real time.

2018, xiii + 75 pages

Keywords: Image Processing, Real Time Systems, FPGA, Zynq, Model Based Design, Embedded Systems

TEŞEKKÜR

Bu çalışmanın baştan sona her aşamasında vizyon ve tecrübesi ile bana yol gösteren, sürekli desteği ve yakınlığı ile çalışmalarımı tamamlamam noktasındaki yapmış olduğu büyük katkılarından dolayı tez danışmanım Sayın Dr. Öğr. Üyesi İsmail YABANOVA' ya, araştırma ve yazım her konuda öneri ve eleştirileriyle yardımlarını esirgemeyen diğer hocalarıma ve arkadaşlarıma teşekkür ederim.

Bu günlere gelmemde emeği büyük olan değerli aileme ömür boyu süren destekleri için, hayatımın her aşamasındaki verdiği sonsuz destek, gösterdiği sabır ve anlayış için sevgili eşim Sümeyra DEMİRCİ'ye teşekkür ederim.

Mustafa Yusuf DEMİRCİ
AFYONKARAHİSAR, 2018

İÇİNDEKİLER DİZİNİ

Sayfa

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler

MHz	Megahertz
Ns	Nanosaniye
Mbit	Megabit
Mb	Megabyte

Kısaltmalar

AXI	Advanced eXtensible Interface (Gelişmiş genişletilebilir Arayüz)
BRAM	Block Random Access Memory (Blok Rastgele Erişimli Bellek)
CCD	Charge Coupled Device (Yük Birleştirilmiş Cihaz)
CMOS	Complimentary Metal OxideSemiconductor (Tamamlayıcı Metal-Oksit Yarıiletken)
DDR	Double Data Rate (Çiftli Veri Hızı)
DMA	Direct Memory Access (Doğrudan Bellek Erişimi)
DSP	Digital Signal Processing (Sayısal Sinyal İşleme)
FIFO	First In-First Out (İlk Giren İlk Çıkar)
FIR	Finite Impulse Response (Sonlu Dürtü Yanıtı)
FMC	FPGA Mezzanine Card (FPGA Ara Kartı)
FPGA	Field Programmable Gate Array(Alan Programlanabilir Kapı Dizisi)
FPS	Frame Per Second (Saniyedeki Kare Sayısı)
HDL	Hardware Description Language (Donanım Tanımlama Dili)
HDMI	High Definition Multimedia Interface (Yüksek Çözünürlüklü Çoklu ortam Arayüzü)
HLS	High-Level Synthesis (Yüksek Seviyeli Sentezleme)
IP	Intellectual Property (Akıllı Özellik)
LUT	Look Up Table (Başvuru Çizelgesi)
MUX	Multiplexer (Çarpıcı)
RGB	Red Green Blue (Kırmızı Yeşil Mavi)
RTL	Register Transfer Level (Kaydedici Aktarımı Seviyesi)
SDK	Software Development Kit (Yazılım Geliştirme Aracı)
SXGA	Super eXtended Graphics Array (Süper Genişletilmiş Grafik Dizisi)
SoC	System On - Chip (Yonga Üzeri Sistem)
VDMA	Video Direct Memory Access (Video Doğrudan Bellek Erişimi)
VHDL	Very High –Speed Integrated Circuit Hardware Description Language (Çok Yüksek Hızlı Tümleşik Donanım Tanımlama Dili)
YUV/YCbCr	Luminance, Chrominance Blue, Chrominance Red (Parlaklık, Kırmızı Renklilik, Mavi Renklilik)

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

RESİMLER DİZİNİ

Sayfa

1. GİRİŞ

Günümüzde otomatikleştirilmiş sistemler çok geniş bir alana yayılmıştır. Konutlarda kullanılan akıllı ev sistemlerinden büyük üretim tesislerine kadar her alanda otomatikleşmiş sistemler kullanım alanı bulmuştur. Artan üretim ve sanayileşme ile birlikte otomatik sistemlere olan ihtiyaç daha da artmıştır. Özellikle yarıiletken teknolojisi ve malzeme bilimindeki gelişim ile otomatik ve akıllı sistemler çok daha karmaşık işlemleri çok kısa sürelerde yapabilecek seviyeye gelmişlerdir.

Bu sistemlerin gelişimi kontrol sistemlerinin kullanımını çok ileri seviyelere taşımış ve gerçek zamanlı kontrol sistemleri tanımı ortaya çıkmıştır. Bu sistemlerden kesin bir zaman dilimi içerisinde istenen tepkiyi göstermeleri beklenir ve zamanlamanın çok kritik olduğu uçus-sürüş kontrol sistemleri, otomotiv, fabrika otomasyonu, savunma sistemleri gibi alanlarda bu tip sistemler sıklıkla kullanılmaktadır. Gerçek zamanlı sistemlerde önemli olan hızdan ziyade zaman kararlılığıdır (Lee 2009).

Gerçek zamanlı kontrol sistemleri genellikle sadece bulunduğu sisteme özel olarak tasarlanır ve bu sistemler gömülü sistem olarak isimlendirilir. Gömülü bir sistem, uygulamaya özel yazılım ve bu yazılımın üzerinde çalışacağı donanım kısmından oluşur. Bu sistemler genel olarak sınırlı kullanıcı ara yüzüne sahiptirler, kullanıcı ile etkileşimde bulunmayan tipler de mevcuttur. Gömülü sistemler verimlilik için optimize edilirler ve genellikle zaman-kritik olarak çalışırlar (İnt. Kyn.1).

Gömülü sistem tasarımlarında mikroişlemciler, uygulamaya özel tümleşik devreler (ASIC - Application Specific Integrated Circuit) ve alan programlanabilir kapı dizisi (FPGA) sıklıkla kullanılan yapılardır. Günümüzde FPGA adı verilen yapılar, teknolojideki gelişmeler ile birlikte daha yüksek kapasiteli ve daha ucuz olarak üretilmeye başlanmış, bu sayede piyasada ve akademik alanda geniş uygulama alanı bulmuşlardır.

Gerçek zamanlı sistemlerin performanslı ve zaman-kritik çalışma koşullarını yerine getirebilmeleri için bilgisayar ile birlikte FPGA kullanımı son yıllarda ağırlık kazanmıştır. Ancak böyle bir kullanım maliyet açısından yüksek olduğu gibi enerji

tüketiminin yüksekliği ve sistemin fiziksel yapısı nedeniyle her gömülü sistemde kullanıma uygun değildir.

Bu çalışmada kullanılan Zynq-7000 platformu, Xilinx firması tarafından geliştirilen ve klasik mikroişlemci ile FPGA mimarisini aynı yonga içerisinde bir araya getiren yeni bir donanım mimarisidir. Bu mimari ile tek bir kart üzerinde hem mikroişlemci ve işletim sistemi kullanılabildiği gibi aynı sistem içerisinde FPGA yapılarının da kullanılabilmesi mümkün olmaktadır. Özellikle görüntü ve sinyal işleme gibi çok sayıda verinin kısa bir zaman dilimi içerisinde işlenmesi gereken uygulamalarda bu platform yüksek hız ve performans vaat etmektedir.

Görüntü işleme algoritmaları gerçek zamanlı kontrol sistemlerinde ve endüstriyel otomasyon alanında sıklıkla kullanılmaktadır. Özellikle son yıllarda geliştirilen insansız hava araçları, otomatik pilot sistemleri, sürücüsüz araçlar, sürüş destek sistemlerinde kullanılan şerit takip, kör nokta uyarısı, çarpışma önleyici gibi sistemler içerisinde görüntü işleme hayati bir öneme sahiptir.

Bu tez çalışmasında ilk olarak FPGA kavramı ve kullanılan Zynq-7000 platformu anlatılmış, daha sonrasında Zedboard geliştirme kartı ve kullanılan geliştirme ortamları hakkında bilgiler verilmiştir. Daha sonra model tabanlı tasarım kavramı üzerinde durulmuş ve model tabanlı tasarımın geleneksel tasarım metotlarından farkı belirtilmiştir.

Sonraki bölümde gerçekleştirilen uygulamada kullanılan görüntü işleme metotları ve video sinyalinin oluşturulması hakkında temel bilgiler verilmiştir. Uygulama aşamasında kullanılan geliştirme kartına uyumlu olan bir arabirim üzerinden yüksek çözünürlüklü görüntü aktarabilen bir kamera kullanılmıştır. Kamera üzerinden alınan görüntüye gerçek zamanlı olarak kenar bulma, gürültü giderme, gri ton dönüştürme ve keskinleştirme gibi görüntü işleme algoritmaları uygulanmıştır. Bunun için geliştirme kartı üzerinde öncelikle FPGA kısmında gerekli donanım tasarlanarak sentezlenmiş, daha sonra bu donanıma göre yine Xilinx geliştirme ortamlarında sisteme uygun yazılım yapılmıştır. Böylece FPGA ve işlemcinin tümleşik olarak kullanıldığı bir sistem tasarlanmıştır. Görüntü işleme birimindeki filtre algoritmaları MATLAB/Simulink kullanılarak

tamamen model tabanlı tasarım yöntemleriyle oluşturularak simülasyonları yapılmış, daha sonra Mathworks'ün kod üretici HDL Coder, MATLAB Coder ve Simulink Coder isimli eklentileri ile bu tasarımlar FPGA tarafından kullanılabilen akıllı özellik paketlerine dönüştürülmüş sonrasında Xilinx Vivado ve SDK geliştirme ortamlarında donanım entegrasyonu yapılarak ve donanıma uygun yazılım hazırlanarak sistem tasarımı tamamlanmıştır.

Çalışmanın amacı, temel görüntü işleme algoritmalarını gerçekleştiren gerçek zamanlı bir gömülü sistem tasarımının FPGA ve işlemciyi bir arada en verimli şekilde kullanarak aynı zamanda tasarım süresi ve zorluğunu asgari seviyede tutarak yapılabilmesidir. Ayrıca kullanılan tasarım metoduyla yapılan sistem tasarımının tekrar kullanılabilirliği çok yüksek olup, aynı akıllı özellik paketlerinin tekrar tasarlanmadan hazır fonksiyon olarak daha ileri çalışmalarda kullanılabilir hale gelmesi amaçlanmaktadır.

Tez kapsamında yapılan çalışmalarda FPGA tabanlı sistemlerin en büyük dezavantajı olan tasarım zorluğu ve tasarım süresinin model tabanlı tasarım yöntemiyle önemli ölçüde azalmış olduğu görülmekle beraber, yapılan tasarımların çalışma hızı, gecikme ve kaynak kullanımı açısından bakıldığında geleneksel yöntemlerden geride olmadığı görülmüştür.

2. LİTERATÜR BİLGİLERİ

Bölüm 1.de bahsedilen Zynq mimarisi çok genç bir mimari olup 2012 yılında duyurulmuştur, buna rağmen endüstride geniş uygulama alanı bulmuş ve hızlı bir şekilde kabul görmüştür. Bununla birlikte geliştirilen sistemlerin genellikle ticari veya askeri amaçlı olması nedeniyle açık kaynak kodlu projeler şimdilik az sayıda akademik çalışma ve eğitimden ibarettir. Ayrıca model tabanlı tasarımla yapılan çalışmaların daha da az olduğu görülmüştür. Bu çalışmaların önemli olanları aşağıda verilmiştir.

Zynq-7000 mimarisi hakkında yapılmış olan en kapsamlı çalışma; University of Strathclyde öğretim üyeleri Crockett, Elliot, Enderwitz ve Stewart tarafından 2014 yılında yazılmış olan "The Zynq Book" isimli kitaptır. Tüm mimari hakkında oldukça detaylı bilgiler verilen kitapta gömülü bir sistemin yapısı, Zynq mimarisinin klasik FPGA sistemlerinden farkı, tasarım metotları, kullanılan yazılım paketleri, kart üzerinde çalışabilecek gömülü Linux işletim sistemleri ile piyasada bulunan geliştirme kartları ile bilgilere ilaveten referans tasarımlar ve uygulamalı çalışmalar da bulunmaktadır. Bu tez çalışmasındaki referansların önemli bir kısmını oluşturan bu eserden ilerideki bölümlerde yararlanılmıştır (Crockett *et al.* 2014).

Russell ve Fischaber yaptıkları çalışmada Zedboard geliştirme kartı üzerine yerleştirilen tümleşik bir kamera sistemi ile 1920 x 1080 çözünürlükteki görüntüyü gerçek zamanlı olarak alıp trafik işaretlerinin görüntülerini tanıma ve sınıflandırma yapan bir sistem geliştirmişlerdir. Tüm projeyi FPGA ve işlemci kaynak paylaştırması ile ve OpenCV kütüphanelerini kullanarak altı hafta gibi bir sürede oluşturmuşlar ve böylece platformun hızlı tasarım yapılabilme özelliğini vurgulamışlardır. Geliştirilen sistemde Bir trafik işaretini sınıflandırmanın yaklaşık olarak 5 saniye süre aldığı belirtilmiştir (Russell and Fischaber 2013).

Monson ve arkadaşları yaptıkları çalışmada optik akış algoritmalarını optimize ederek farklı platformlarda yürütüp deneysel karşılaştırmalar yapmışlardır. Buna göre aynı algoritma Core i7 işlemcili bir bilgisayarda, Zedboard'un Zynq-7000 yongasının yalnızca ARM işlemci biriminde ve son olarak Zynq-7000 mimarisinin programlanabilir lojik birimine özel en iyileştirme yapılarak yürütülmüştür. Sistemin kodu C dilinde Vivado

HLS programı kullanılarak sentezlenmiştir. Sonuçlara bakıldığında Zynq üzerinde yürütülen algoritmanın performansı i7 işlemcili bir bilgisayarın performansına yakın olmakla birlikte güç tüketiminin yalnızca bilgisayarın 1/7 si kadar olduğu görülmüştür (Monson *et al.* 2013).

Sümer yaptığı çalışmada farklı yüz tanıma metotlarını kullanarak duygu ve yüz ifadesi analizi uygulaması gerçekleştirmiştir. Zynq-7000 platformunu kullanan Zedboard geliştirme kartında OpenCV kütüphaneleri, C++ dili ve gömülü Linux işletim sistemi kullanılarak yapılan çalışmada öfke, mutluluk ve şaşırma ifadeleri sırasıyla %97, %100 ve %97 başarıyla sınıflandırılmış ve yaklaşık saniyede 4 ile 5 kare hızında bir performans elde edilmiştir (Sümer 2014).

Altuncu, M. ve arkadaşları yaptıkları çalışmada bazı görüntü işleme algoritmalarını Zedboard geliştirme kartı üzerinde Verilog donanım tanımlama dilini kullanarak gerçekleştirmişlerdir. Usb üzerinden webcam ile alınan 256 x 256 piksel boyutundaki görüntü üzerinde gerçek zamanlı olarak filtre işlemlerini saniyede yaklaşık 40 kare hızında uygulamayı başarmışlardır (Altuncu *et al.* 2015).

Chhabra ve arkadaşları Zynq-7000 platformunu barındıran XC7Z020-1CLG484 geliştirme kartı üzerinde bir otomatik plaka tanımlama sistemi tasarlamışlardır. Sistem öncelikle alınan görüntü üzerinde plakanın bulunduğu yeri tespit edip geri kalan kısmı keserek plakanın olduğu görüntüyü almakta ve sonrasında görüntüyü üç farklı metotla işlemekte ve plaka karakterlerini çıkış olarak vermektedir. Tüm sistem MATLAB Simulink ve Xilinx System Generator geliştirme ortamı kullanılarak yüksek seviyeli dilde sentezlenmiş, gerçeklenen uygulamalar içerisinde en verimli olanı tespit edilmiş ve her üç yöntemin de %90'ın üzerinde başarılı olduğu tespit edilmiştir (Chhabra *et al.* 2016).

Shi yaptığı çalışmada Zynq-7000 mimarisine sahip Zedboard üzerinde Sobel kenar belirleme filtresini Vivado HLS aracını kullanarak gerçek zamanlı olarak gerçekleştirmiştir. Görüntü aktarımındaki gecikmeyi önlemek için OpenCV'de yazılmış olan Sobel filtre algoritmasını uyarlayarak daha hızlı çalışmasını sağlamış ve önceki incelediği tasarımlardan %75 daha az kaynak kullanımına rağmen %30 daha fazla

performans elde ederek 1080p çözünürlükte maksimum 90,1 fps hıza ulaşabilmiştir (Shi 2016).

Son olarak Al-Naqshbandi yaptığı çalışmada Zedboard üzerinde Canny kenar belirleme algoritmasını (Canny Edge Detection) uygulayıp en iyileştirmesini gerçekleştirmiştir. OpenCV kütüphaneleri kullanılarak yapılan çalışmada birçok filtre içeren Canny kenar belirleme algoritmaları başarıyla uygulanmış ve yüksek seviyeli sentez yöntemi ile donanım sentezi yapılmış ve kodlar C++ ile oluşturulmuştur. Çalışmada algoritmaların performans analizleri yapılmış olup hem çekirdek hem de uygulama düzeyinde yapılan en iyileştirmeler belirtilmiştir. Yapılan geliştirmeler sonucu çekirdek düzeyinde 3 kata kadar, uygulama düzeyinde ise 7 kata kadar daha hızlı çalışma sağlandığı görülmüştür (Al-Naqshbandi 2016).

3. MATERYAL ve METOT

3.1. Görüntü İşleme

Bir görüntü, f (x,y) biçiminde iki boyutlu bir fonksiyon olarak tanımlanabilir. Burada x ve y uzamsal koordinatları ifade eder ve herhangi bir x,y noktasında f'nin genliği o noktanın yoğunluk veya gri ton seviyesi olarak adlandırılır. Bu fonksiyonun ayrık ve sonlu değerleri tanımlı olduğunda bu görüntüye sayısal görüntü adı verilir (Gonzalez and Woods 2002).

Sayısal Görüntü işleme ise bir görüntü kaynağından alınan görüntünün sayısal formata dönüştürülüp bu görüntü üzerinde çeşitli işlemler yapılarak çeşitli bilgiler elde edilmesi sürecidir. Bilgisayar görüsü, tıbbi görüntüleme cihazları, askeri savunma sistemleri, sürücü destek sistemleri, otonom araçlar gibi birçok alanda gelişmiş görüntü işleme teknikleri kullanılmaktadır.

Görüntü işlemenin tanımı diğer kavramlardan net olarak ayrılmamakla birlikte genel olarak üç tür görüntü işleme aşaması bulunduğundan söz edilebilir. Bunlar düşük, orta ve yüksek seviye görüntü işleme aşamalarıdır. Düşük seviyeli görüntü işlemede ön işleme de denilen görüntüyü diğer işlemlere hazırlamak için kullanılan gürültü azaltma, karşıtlık düzeltme ve keskinleştirme gibi işlemler yapılır. Orta seviyeli görüntü işlemede ise görüntüyü belirli özelliklere göre bölütleme ve sınıflandırma işlemleri yapılmaktadır. Bunlara örnek olarak kontur ve kenar bulma ya da bir parçayı görüntüden çıkartma verilebilir. Yüksek seviyeli işlemede ise bir topluluk olarak görüntüden anlam çıkarma, analiz vb. işlemler kastedilir (Gonzalez and Woods 2002).

3.2 Alan Programlanabilir Kapı Dizisi (FPGA)

Bu kısımda kullanılan geliştirme platformunun en önemli iki bileşeninden biri olan Alan Programlanabilir Kapı Dizilerinin (FPGA) yapıları, çalışma prensipleri ile neden kullanılmaları gerektiği ile ilgili bilgiler verilmiştir. Ayrıca FPGA ile tasarım yapabilmek için kullanılan donanım tanımlama dilleri ve sentezleme kavramları açıklanmıştır.

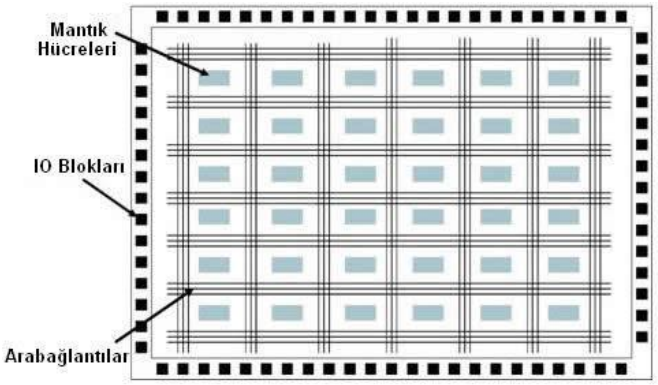
3.2.1 FPGA Mimarisi Ve Özellikleri

FPGA, istenen fonksiyona göre içyapıları kullanıcı tarafından değiştirilebilen, donanım olarak programlanabilen tümleşik devrelerdir. FPGA'i diğer mikroişlemci veya tümleşik devrelerden ayıran en önemli özelliği, içeriğinin donanım tanımlama dilleri yardımıyla istenilen biçimde programlanarak amaca özel bir yapıda çalıştırılabilmesidir (İnt.Kyn.2).

FPGA içindeki mantık kapıları, tasarımcının içerisine yazdığı programa göre gerekli biçimde çalışarak ilgili devre oluşturulmuş olur. Bunun anlamı, hemen her tür devrenin FPGA üzerinde gerçeklenebilmesinin olanaklı olmasıdır. Hatta bu yapı bilgisayar bağlantılı veya yonga üzeri sistem (SoC) tarzı sistemlerde dinamik olarak program içerisinde de değişebilmektedir. Bu da tasarımlarda çok esnek bir altyapı sunar.

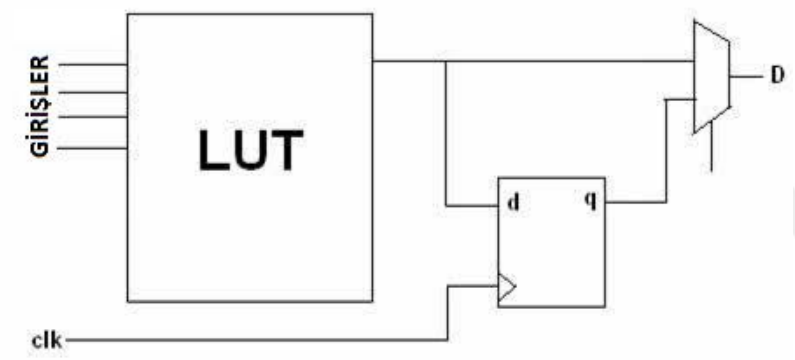
FPGA'lerin en önemli özelliklerinden birisi de paralel işlem yapabilme özelliğidir. Geleneksel bilgisayar ve mikroişlemci/mikro denetleyici sistemlerinde işlemler bir sıraya göre yapılır. Buna örnek olarak görüntü işleme sistemleri verilebilir. 640 x 480 çözünürlüğündeki görüntünün tek bir karesi üzerinde herhangi bir işlem yapılmak istendiği düşünüldüğünde bu işlemler piksel düzeyinde gerçekleşeceği için 640 x 480 = 307 200 adet işlemci tetikleme döngüsü (CPU clock cycle) gerekecektir. Elbette günümüzdeki işlemcilerin hızı çok yüksektir, ancak gerçek zamanlı sistemlerde özellikle saniyede 40-50 kare üzeri hızlar ve yüksek çözünürlükte görüntülerle çalışıldığı düşünülürse bu yüksek bant genişliğinde verilerin işlemciler ile sırayla işlenmesi sistemi büyük darboğaza sokacak ve gerçek zamandan kopmaya neden olacaktır. Burada FPGA'lerin avantajı net olarak görülür. FPGA temelli sistemler, sahip oldukları kapasiteye göre tüm pikselleri lojik yapıları içerisinde aynı anda sadece 1 saat darbesinde işleyebilirler. Bu da hız ve paralel hesaplama gerektiren görüntü - sinyal işleme gibi uygulamalarda çok önemli avantajlar sağlar.

FPGA temel olarak Mantık Hücreleri (Logic Cell), Giriş/Çıkış Blokları (IO Block) ve Ara bağlantılardan oluşur. Bir FPGA bloğunun basit yapısı Şekil 3.1.de verilmiştir.



Şekil 3.1 Bir FPGA Bloğunun Temel İç Yapısı (İnt. Kyn.15).

Mantık hücreleri, FPGA'in içerisinde gerekli devreyi kurmaya yarayan elemanlardır. Bir mantık hücresinin gösterimi Şekil 3.2.de verilmiştir.



Şekil 3.2 Bir Mantık Hücresinin Basit Devre Şeması (İnt. Kyn.15).

FPGA'in ana yapısını Mantık Hücreleri oluşturur. Bir mantık hücresi 1 adet başvuru çizelgesi (Lookup Table -LUT), 1adet D tipi Flip-Flop ve bir adet 2 x 1 çarpıcıdan (Mux) oluşur. LUT'lar, bir mantık işlemini yerine getiren küçük bellek parçalarıdır. N girişli bir LUT, 2^n elemanlı bir belleğe işaret eder. Bu mantık hücrelerinin çok sayıda bir araya gelmesiyle FPGA üzerinde devreler kurulur.

Mantık hücrelerinin ara bağlantıları matris şeklindeki veri yolları ve programlanabilir anahtarlama elemanları ile sağlanır. Bu ara bağlantılar da yapılan tasarıma göre tamamen değişebilir niteliktedir. FPGA tasarımı, her bir mantık hücresinin uygulayacağı fonksiyonu ve programlanabilir anahtarların durumunu (açık/kapalı) belirleyerek bu mantık hücreleri arasındaki bağlantıları tanımlar (İnt.Kyn.2).

3.2.2 Donanım Tanımlama Dilleri

FPGA'ler yazılım ile programlanan donanım blokları olduğundan kullanılan programlama dilleri de "Donanım Tanımlama Dili" (Hardware Description Language - HDL) olarak isimlendirilmektedir. Günümüzde FPGA'ler RTL seviyesinde HDL ile kodlanabildiği gibi C/C++ kodundan sentezlenebilir, ya da MATLAB/Simulink ve Labview gibi yüksek seviyeli dillerden model tabanlı olarak kod üretilebilir. Bu tez çalışmasının ileriki bölümlerinde model tabanlı tasarım kullanılacaktır.

HDL, bir donanım parçasını modellemek için kullanılan yazılım dilidir. Tüm dünyada en yaygın kullanılan iki HDL, VHDL ve Verilog'dur. Bu iki dilin de birbirlerine göre bazı farklılıkları olmakla beraber ikisi de aynı işlevi gerçekleştirebilmektedir. VHDL, Verilog'a göre daha kısıtlamalı (strict-typed) veri tiplerine sahiptir. Bununla birlikte özel veri tipleri oluşturma imkânı VHDL'de daha fazladır. Verilog dili ise C dili ile sözdizimi olarak daha fazla benzerlik taşır.

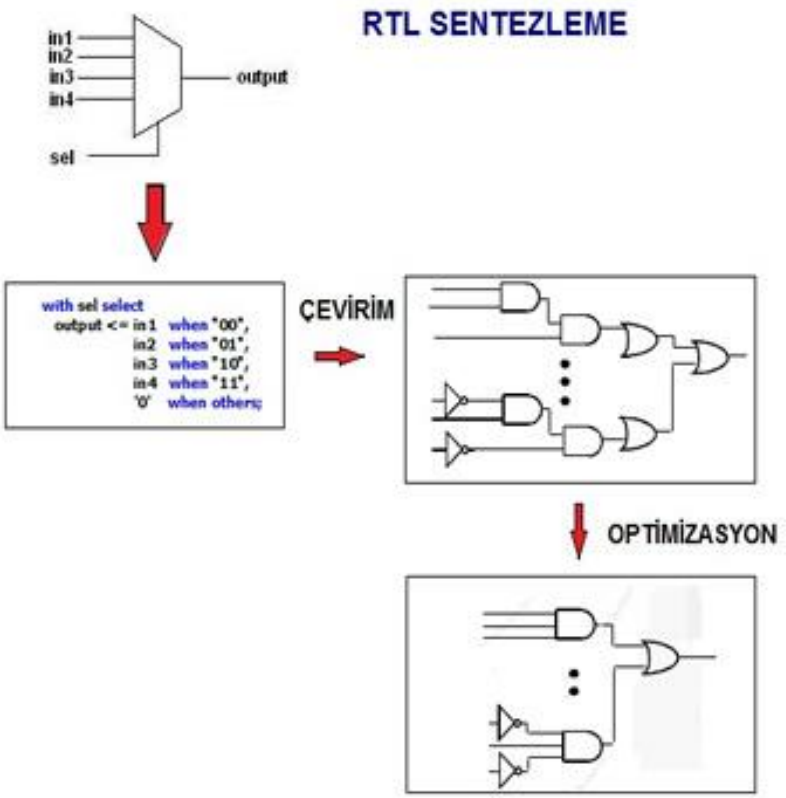
3.2.3 Donanım Sentezleme

FPGA tasarımı yaparken "sentezleme" kavramı karşımıza çıkar. Sentezleme, tanımlanan donanımın derleyici tarafından derlenerek kaydedici aktarım seviyesine (RTL- Register Transfer Level) indirgenerek ve gerekli performans geliştirmelerinin yapılarak FPGA üzerinde çalışır duruma getirilmesi işlemidir. Bu durum yazılım kodlarının derleyiciden geçirilerek makine koduna dönüştürülmesi gibi düşünülebilir.

Şekil 3.3.te RTL sentezlemenin basit bir gösterimi verilmiştir (İnt.Kyn.2).

Sentezleyiciler, çeşitli algoritmalar kullanarak aynı zamanda devreyi basitleştirme ve indirgeme işlemini de yaparlar. Böylece insandan kaynaklanan hataların en aza indirilmesi sağlanacak ve tasarımın fonksiyonelliği artacaktır.

Donanım sentezinden sonra kodun hatalara karşı tekrar kontrol edilmesi için simülasyon ve test aşamalarının da yapılması gerekmektedir. Simülasyon yazılımları üzerinde devre doğrulanmadan FPGA'e yüklenemez.



Şekil 3.3 RTL Sentezleme Prensip Şeması (İnt. Kyn.16).

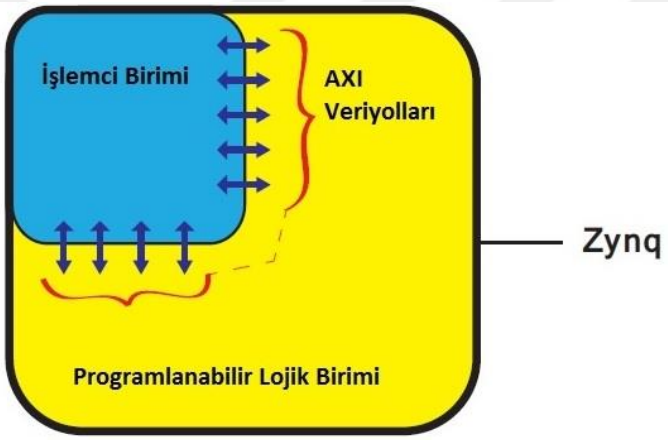
3.3 Zynq-7000 Yonga Üzeri Sistem Mimarisi

Bu Bölümde Zynq-7000 mimarisi hakkında bilgi verilecek ve bu mimarinin önemli bileşenleri ile geleneksel FPGA yapısından farklı olan kısımları üzerinde durulacaktır. Zynq-7000 bir SoC olduğundan öncelikle SoC tanımını yapmak gerekmektedir. Yonga üzeri sistem (System on Chip) genel olarak birden fazla sistem biriminin tek bir yonga içerisinde bulunduğu mimarilere verilen isimdir. Bu mimaride tek bir yonga içerisinde mikroişlemci, grafik işleme birimi, hafıza birimi, ya da Zynq üzerinde olduğu gibi programlanabilir lojik üniteler bulunabilir. Yonga üzeri sistem mimarisi son yıllarda özellikle mobil cihazlar ve mikro bilgisayarlar olmak üzere birçok alanda yaygın olarak kullanılmaktadır.

Zynq-7000, çift çekirdekli ARM Cortex-A9 mimarisine sahip işlemci ile FPGA bileşenlerini aynı yonga içerisinde barındıran Tümüyle Programlanabilir Yonga Üzeri Sistem mimarisidir (All Programmable SoC). Daha önceki bilgisayar - FPGA tabanlı birleşik sistemlere kıyasla çok daha güçlü ve pratik bir çözüm metodu olup işlemci ve FPGA arasında yüksek hızlı ve düşük gecikmeli AXI (gelişmiş genişletilebilir arabirim)

veri yolları bulunmaktadır. Böylece bu iki birimin birbirinden bağımsız olmaksızın amaca yönelik olarak bir arada kullanılması olanaklı hale gelmiştir. Bu birimlerin ayrı sistemler olarak birbirlerine bütünleşik kullanılması da pekâlâ mümkündür ancak Zynq mimarisi hem maliyet olarak daha ucuzdur, hem de kapladığı alan, genel sistem hızı, enerji tüketimi ve güvenli veri aktarımı gibi konularda çok daha üstün bir çözümdür (Crockett *et al.* 2014).

Zynq-7000 AP SoC, temel olarak işlemci birimi (PS- Processing System), FPGA sistemine karşılık gelen bir programlanabilir lojik biriminden (PL - Programmable Logic) ve ara bağlantılardan meydana gelir. Şekil 3.4.te Zynq mimarisinin basit prensip şeması verilmiştir.

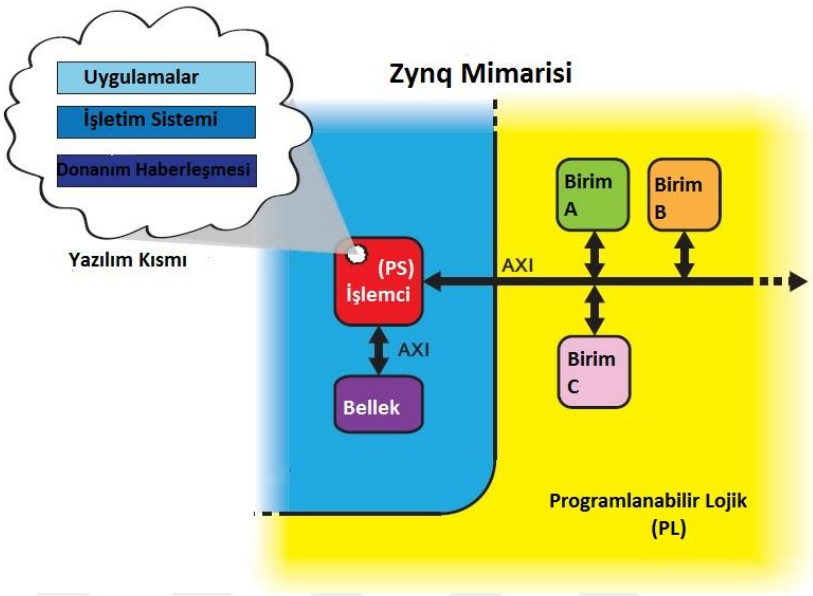


Şekil 3.4 Zynq-7000 Mimarisi Basit Prensip Şeması (Crockett *et al*. 2014).

Zynq üzerindeki işlemci birimi, standart bir ARM Cortex-A9 mimarisi işlemcidir ve uygulama düzeyinde çalışmaktadır. Bunun anlamı, sistem üzerinde ARM mimarisi ile uyumlu olan herhangi bir işletim sistemi çalıştırılabilirken aynı zamanda FPGA üzerinde istenilen işlemlerin yapılarak gerekli veri alışverişinin yapılmasının mümkün olmasıdır.

PS, normal bir işlemci olduğundan sabit bir mimaridir ve işlemci ile hafıza (RAM) elemanlarını üzerinde barındırır. PL kısmı ise tamamen esnek bir yapıdadır. Bu yapı sayesinde önceden tanımlanan akıllı özellik blokları (Intellectual Property - IP Block) kullanılabileceği gibi yeni tasarımların yapılması da mümkün olmaktadır. Bu iki birim arasındaki ara bağlantılar AXI veri yollarıyla yapılmıştır.

Şekil 3.5.te Zynq mimarisinin yazılım, donanım ve çevre birimleri arasındaki etkileşimin basit yapısı verilmiştir.

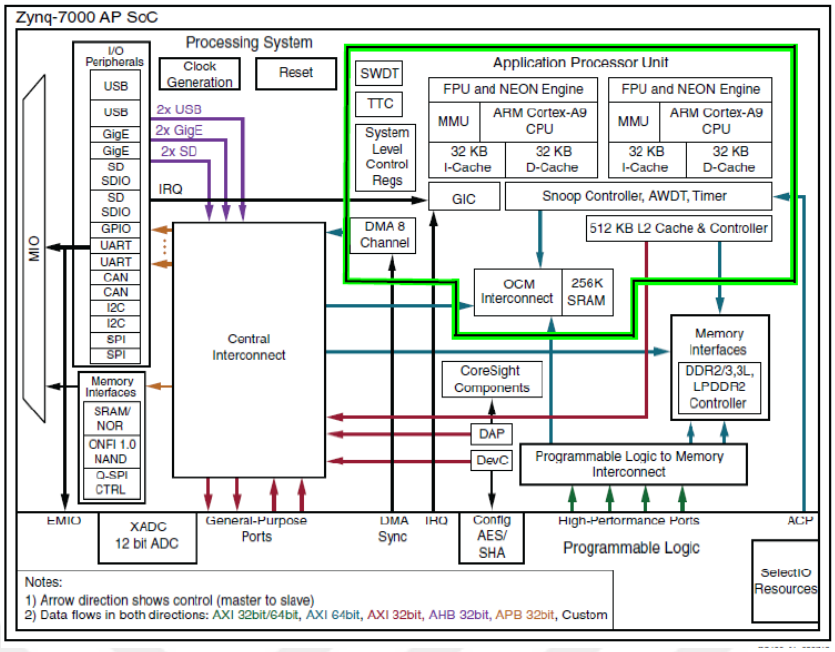


Şekil 3.5 Zynq Mimarisi ve Yazılım-Donanım İlişkisi (Crockett *et al*. 2014).

3.3.1. Zynq-7000 SoC Bileşenleri

3.3.1.1 Mikroişlemci Birimi

Zynq-7000 üzerindeki işlemci ARM-Cortex A9 mimarisine sahiptir ve 1 GHz hıza kadar ulaşan 2 adet işlemci çekirdeği bulundurur. Bu işlemci, Zynq üzerindeki fiziksel işlemcidir. Aynı zamanda programlanabilir lojik üzerinde sanal işlemci birimi oluşturmak da mümkündür. Oluşturulan sanal işlemci birimleri, yonga üzerindeki fiziksel işlemciyle bir arada da kullanılabilir. Burada belirleyici olan yapılacak işin gerektirdiği özellikleri en doğru şekilde karşılayacak tasarımı yapmaktır. Zynq mimarisinin tasarım metotlarından ileriki bölümlerde bahsedilecektir. Şekil 3.6.da işlemci biriminin blok şeması verilmiştir.



Şekil 3.6 Zynq Mimarisi ve İşlemci Sistemi Blok Şeması (Xilinx 2017).

3.3.1.2 Programlanabilir Lojik Birimi

Zynq mimarisinde kullanılan programlanabilir lojik birimi, FPGA'e karşılık gelen yapıdır ve fiziksel olarak bir FPGA'nın yapabileceği her türlü işlevi gerçekleştirebilir. Hedef platformumuz olan Zedboard üzerinde Z-7020 serisi programlanabilir yonga bulunmaktadır. Bu yonga üzerinde 1 Ghz çift çekirdek ARM işlemci ve Xilinx Artix-7 serisi FPGA bloğu kullanılmaktadır. Bu FPGA içerisinde ortalama 85 000 mantık hücresi, 53 200 LUT, 106 400 Flip-Flop bulunmaktadır. Ayrıca 220 adet 18 x 25 bit DSP48 lojik dilimi ve 140 adet 34kb Blok RAM bulunmaktadır (Xilinx 2017).

3.3.2 Dizayn Metotları

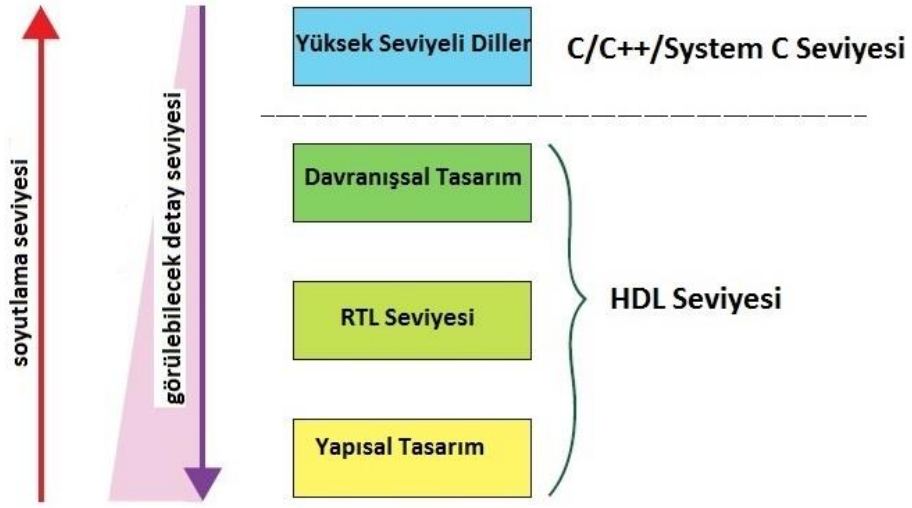
FPGA tasarımları sayısal devre, yani donanım tabanlıdır. Donanım tasarımı yapılırken kullanılan çeşitli yöntemler bulunur. Burada soyutlama kavramı ortaya çıkar. Soyutlama, tasarım dilinin fiziksel sistemden uzaklaşma seviyesini ifade eder. Sayısal tasarım metotları genel olarak dört tiptir. Bunlar donanım tanımlama dili seviyesi (HDL Level), Kaydedici transfer seviyesi (RTL), mantıksal kapı seviyesi (Gate Level) ve anahtar seviyesi (Switch Level) olarak adlandırılır (Sarıtaş ve Karataş 2013).

Fiziksel devre elemanlarını birbirlerine bağlayarak yapılan tasarımlar lojik kapı seviyesi ve anahtar seviyesi olarak adlandırılırken donanım tanımlama dilleriyle yapılan tasarım RTL, C/C++ dilleri üzerinden HDL sentezi ise yüksek seviyeli sentez (High Level Synthesis) olarak adlandırılır. En üst seviyede yer alan model tabanlı tasarımlar ise tamamen çok yüksek seviyeli programlama dilleriyle oluşturulurlar. Bu şekilde yapılan tasarımların soyutlama seviyesi diğerlerinden daha yüksektir. Bu durum donanım tanımlama dillerine hakim olmadan da tasarım yapabilme avantajını beraberinde getirir.

En düşük soyutlama seviyesi olan yapısal tasarımda, tasarımcı tüm elemanların bire bir bağlantılarını tanımlamakla yükümlüdür. Hatta bu durum, LUT ve Flip-Flop'lar düzeyinde dahi gerçekleşebilir. Bu şekilde tasarımcı tüm kontrolü elinde tutar ve gerekli en iyileştirmeleri yapabilir (Crockett *et al.* 2014).

Kaydedici transfer seviyesinde ise diğer teknolojik detaylar görünmemekle birlikte, sistemin kaydedicilerinin her işlemde aldığı değerler gözlenebilir ve böylece hata ayıklama kolaylaşır.

Donanım tanımlama dili seviyesi ise yapısal ve davranışsal seviye olmak üzere iki türe ayrılır. Soyutlama arttıkça tasarım zorluğu ve süresi azalacak, ancak hata olasılığı da artacaktır. Şekil 3.7.de soyutlama seviyeleri arasındaki ilişki şematik olarak gösterilmiştir.



Şekil 3.7 FPGA Tasarım Soyutlama Aşamaları (Crockett *et al.* 2014).

Buraya kadarki bölümlerden anlaşıldığı üzere çalışmada kullanılan Zynq mimarisi, geleneksel FPGA ve mikroişlemci sistemlerinden yapı olarak oldukça farklıdır. Bu farklılık programlama ve sistem tasarımı aşamalarında da aynı şekilde kendini göstermektedir.

Mimarinin klasik sistemlerden en büyük farkı, hem donanımı hem yazılımı bir arada barındırması ve dolayısıyla bunların her ikisinin de programlanması gerekliliğidir. Bu durum hem iyi bir bilgisayar ve sayısal elektronik bilgisi, hem de iyi düzeyde programcılık gerektirdiğinden uzun zaman alabilir ve birden fazla kişiden oluşan bir ekip ile uzun geliştirme süreçleri ihtiyacı doğurabilmektedir. Özellikle görüntü işleme gibi birçok donanım biriminin bir arada kullanılması gereken yüksek veri aktarımının gerçekleştiği projelerde bu zorluk daha da fazla hissedilmektedir.

Bu durumun önüne geçmek için Xilinx Zynq-7000 mimarisine uyumlu olan gerek Xilinx firmasının kendi ürünleri, gerekse de diğer firmaların üçüncü parti yazılım çözümleri bulunmaktadır. Bu çözümlerin öne çıkanlarından aşağıda kısaca bahsedilmiştir.

3.3.2.1 Xilinx System Generator

Xilinx System Generator For DSP, genellikle DSP tasarımlarında kullanılan MATLAB/Simulink ortamında model tabanlı tasarım yapabilmeyi sağlayan bir geliştirme aracıdır. Bu yöntemde özel hazır fonksiyon blokları ile DSP ve FPGA tasarımları Simulink ortamında gerçeklenebilmektedir (İnt.Kyn.3).

3.3.2.2 Vivado HLS

Xilinx firmasının bir diğer geliştirme aracı olan Vivado High Level Synthesis, C/C++ dillerinden HDL kod sentezi yapabilen bir yazılımdır. Bu araç kullanıcılara donanım tanımlama dili bilgisine gerek kalmadan C gibi yüksek seviyeli dilleri kullanarak FPGA programlayabilmeyi olanaklı kılmaktadır (Xilinx 2014).

Vivado HLS, temel olarak yazılan C kodlarını uygun HDL formatına çevirir ve gerekli değişken ile sinyal dönüşümlerini yapar. Geriye kalan işlemler HDL yazımındaki işlemlerden farksızdır.

3.3.2.3 Mathworks HDL Coder

Hdl Coder, Mathworks şirketi tarafından geliştirilmiş olan MATLAB isimli yazılımın bir parçası olarak sunulan otomatik kod üretici aracıdır. Bu araç ile MATLAB/Simulink içerisindeki hazır bloklar ve fonksiyonlar kullanılarak yapılan tasarımlardan direkt olarak VHDL ya da Verilog kodu üretilebilmektedir. Ayrıca geliştirilen tasarımın simülasyonunun, doğrulanmasının ve hatta sisteme yüklenmesinin Matlab ortamında yapılması mümkün olmaktadır (Mathworks 2017).

Bu tez çalışmasındaki görüntü işleyici sisteminin tasarımında da kullanılan bu araç hakkında daha detaylı bilgi Bölüm 3.7.3.te bulunabilir.

3.4 Gerçek Zamanlı Sistemler

Gerçek zamanlı sistem kavramı, işletim sistemlerinde ve özellikle gömülü sistemlerde sıkça rastlanılan bir kavram olup yeni nesil gömülü sistem örneklerinde sıkça bulunan bir sistem çeşididir. Gerçek zamanlı sistemler, bir görevi yerine getirmenin zaman-kritik olduğu sistemlerdir. Bu tip sistemlerde istenen hızdan ziyade zaman kararlılığıdır. İşin istenen zaman aralığında yapılması şarttır (Lee 2009).

Bu sistemler genel olarak "Katı Gerçek Zamanlı (Hard Real-Time)" ve "Gevşek Gerçek Zamanlı (Soft Real-Time)" olmak üzere ikiye ayrılır.

Katı gerçek zamanlı sistemlerde doğru zamanda gerçekleşmeyen bir tepki sistemin kendisini ya da çevreyi tehlikeye atabilir. Bu tip sistemlere örnek olarak sürücü destek sistemleri, silah ve füze kontrol sistemleri ile her türlü araç kontrol sistemleri verilebilir.

Gevşek gerçek zamanlı sistemlerde ise istenen tepkinin zamanında alınamaması sistemi felakete sürüklemez, ancak sistemin çalışma hızı ve kararlılığı olumsuz etkilenir. Bu sistemlere örnek olarak görüntü ve ses işleme sistemleri ile haberleşme sistemleri verilebilir (Kahraman ve Ünal 2007).

3.5 Model Tabanlı Tasarım

Günümüzde kontrol ve otomasyon sistemlerinin tasarımı yarıiletken teknolojisinin de gelişimiyle birlikte giderek daha karmaşık bir hal almıştır. Bu karmaşıklaşmanın özellikle gömülü sistemlerin tasarım süresi ve maliyetini arttırmak gibi bir etkisi olmaktadır. Bu durumun önüne geçmede model tabanlı tasarım yaklaşımı son yıllarda sıkça kullanılmaktadır.

Model tabanlı tasarım gereksinimlerin belirlenmesinden itibaren başlayıp tasarım sürecine, gerçeklemeden test edilmeye kadar tüm sistem geliştirme sürecini kapsayan ve bu geliştirme sürecini sistem modeli çerçevesinde ele alan bir sistem geliştirme yöntemidir (Canpolat ve Durak 2012).

Model tabanlı tasarımın uygulanması son ürünün istenen standartlara uymasını kolaylaştırır. Ayrıca modellerin kullanımı üretimin farklı aşamalarında çalışan geliştirme gruplarının birlikte daha etkin biçimlerde çalışmasına da katkıda bulunacaktır. Modellerin anlaşılabilirliği sayesinde tasarım grupları arasındaki iletişim artacak, böylece sorunlar daha erken aşamalarda belirlenip giderilerek ortaya daha kararlı, verimli ve yüksek kaliteli sistemler çıkabilecektir (Smith *et al.* 2007).

Şekil 3.8.de model tabanlı tasarım döngüsü gösterilmiştir.

Bu tez çalışmasında model tabanlı bir tasarım aracı olan Simulink yazılımı kullanılarak gömülü yazılımlar oluşturulmuş ve test edilerek sisteme entegre edilmiştir.

Şekil 3.8 Model Tabanlı Tasarım Döngüsü (Canpolat ve Durak 2012).

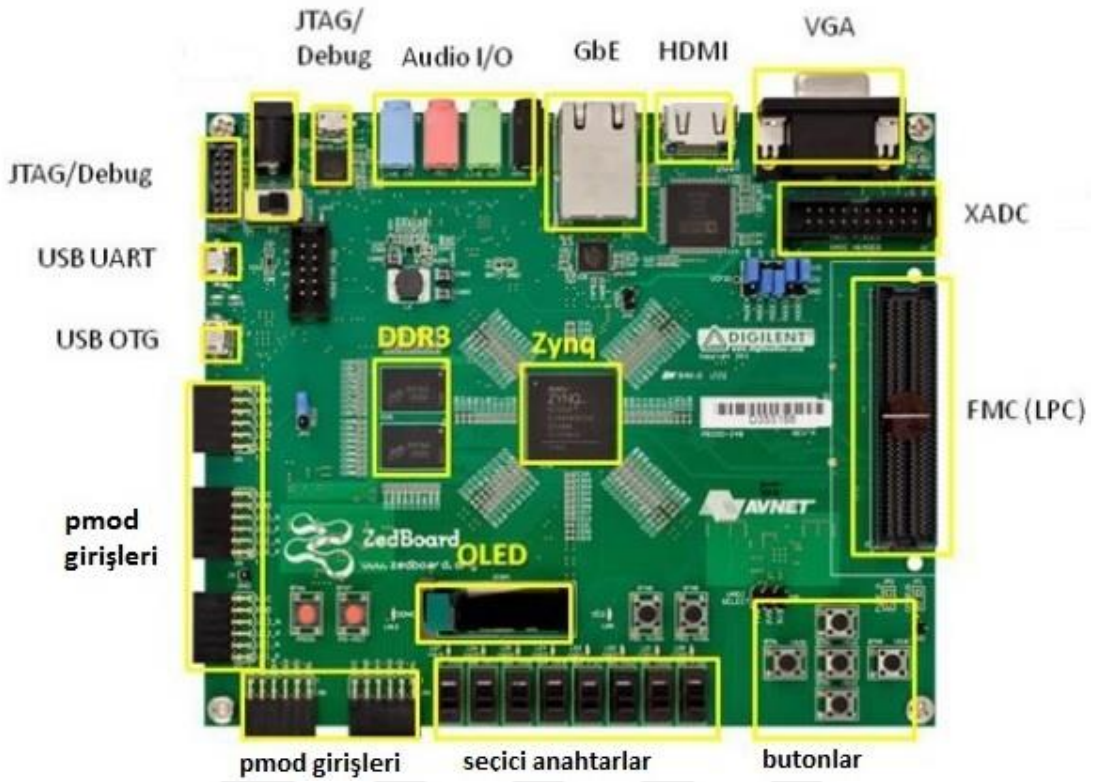
3.6. Kullanım Donanım

3.6.1 Zedboard Zynq-7000 Geliştirme Kartı

Tez çalışmasındaki tüm uygulamalar Zedboard geliştirme kartı üzerinde gerçeklenmiştir. Zedboard (Zynq Evaluation and Devalopment Board), Digilent ve Avnet firmaları tarafından Zynq-7000 platformu üzerinde geliştirilmiş, özellikle öğrenciler ve akademisyenler ve hobi elektronikçiler için üretilmiş olan düşük bütçeli bir geliştirme kartıdır. Aynı mimariyi barındıran diğer kartlara göre daha düşük bütçeli olmakla birlikte güçlü özelliklere sahip bir karttır ve gerçek zamanlı uygulamalar için uygun altyapı sunmaktadır. Zedboard'u farklı yapan bir diğer unsur ise geliştiricilerin ve akademik amaçlı kullanıcıların yardım alabileceği destek ortamlarının resmi olarak bulunmasıdır (İnt.Kyn.4).

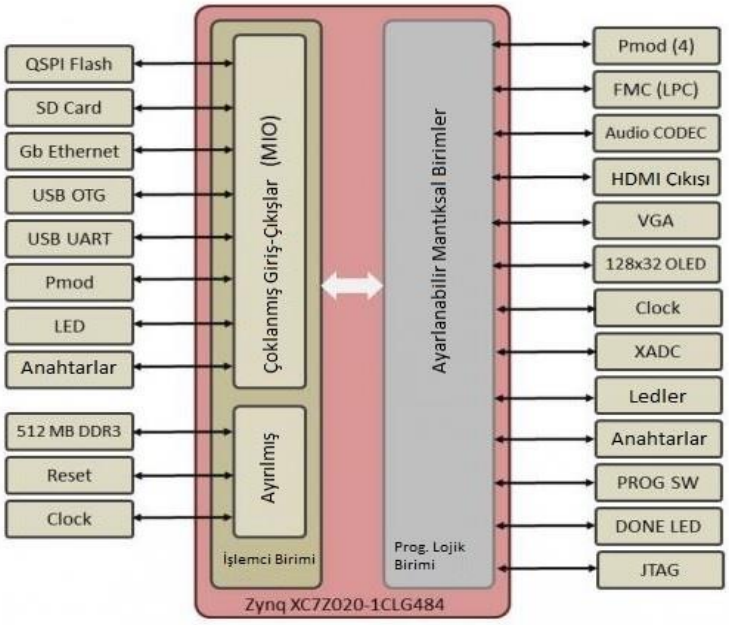
Zedboard, Xilinx firması tarafından üretilen Zynq-7000 mimarisine sahip olan XC7Z020 yonga üzeri sistem elemanı üzerinde temellenmiş bir karttır. Bu yonga üzeri sistem Artix-7 serisi FPGA ünitesi ve çift çekirdek ARM Cortex-A9 işlemciden oluşur. Kart üzerinde 256 Mbit flash hafıza ile 512 Mb DDR-3 bellek bulunmaktadır.

Resim 3.1.de Zedboard geliştirme kartı ve üzerindeki bileşenleri verilmiştir.



Resim 3.1 Zedboard Geliştirme Kartı ve Bileşenleri (İnt.Kyn.4).

Görüntü işleme uygulamaları için çok iyi bir altyapı sağlayan kartın üzerinde 9 adet genel amaçlı LED, 8 adet anahtar, 7 adet buton, OLED ekran, HDMI, VGA, USB-JTAG, OTG, UART desteği ve SD kart desteği bulunur. Şekil 3.9.da Zedboard'un blok diyagramı verilmiştir.



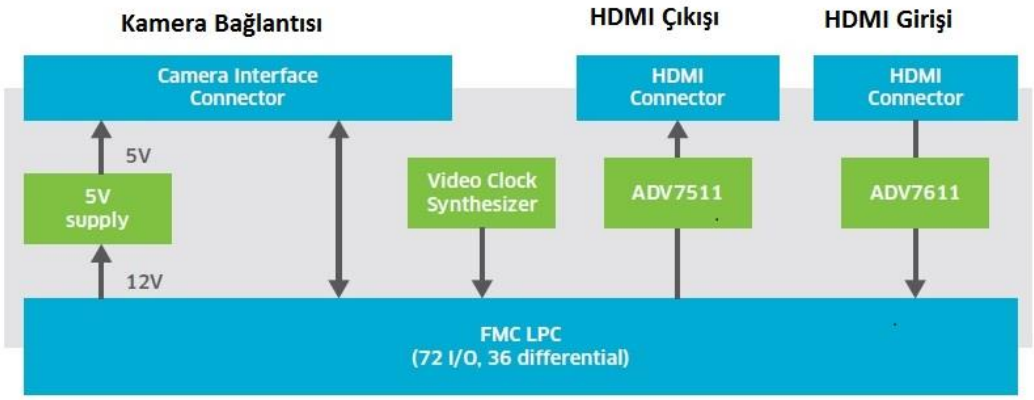
Şekil 3.9 Zedboard Sistem Blok Diyagramı (İnt.Kyn.4).

3.6.2 FMC-HDMI-CAM-G Modülü

FMC-HDMI-CAM Modülü, Avnet firması tarafından geliştirilmiş olan FMC portu bulunan Xilinx geliştirme kartlarına uyumlu bir yüksek çözünürlüklü video alıcı-verici arabirimidir. Üzerinde HDMI giriş ve çıkışlarına ilave olarak dahili kamera bağlantısının yapılabileceği bir PCI-Express portu da bulunur (Avnet 2015).

Resim 3.2 FMC-HDMI-CAM-G Modülü (Avnet 2015).

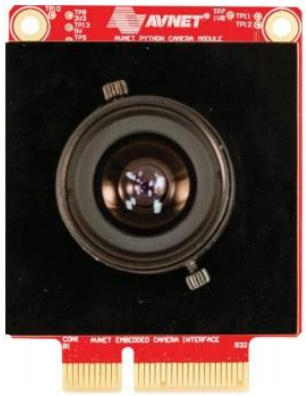
Yüksek çözünürlüklü ve gerçek zamanlı video aktarımını mümkün kılan bu modül üzerinde HDMI girişi ve çıkışı için Analog Devices firmasının ürettiği ADV7611 ve ADV7511 isimli alıcı-verici yongaları bulunmaktadır. Ayrıca video tetikleme sinyallerinin üretilmesi için de dahili bir saat darbesi üreteci de bulunan modül I2C standardı sayesinde programlanıp çalıştırılabilir. Şekil 3.10.da modülün genel yapısı görülebilir.



Şekil 3.10 FMC-HDMI-CAM Modülü Blok Şeması (Avnet 2015).

Zedboard üzerinde dahili olarak bulunan FMC portu sayesinde kolay bir şekilde bağlantı yapılıp yüksek hızlı görüntü işleme sistemleri rahatlıkla tasarlanabilir. Çalışmada ayrıca bu karta uyumlu olan Python 1300-C kamera modülü de kullanılmıştır.

3.6.3 PYTHON-1300-C Kamera Modülü



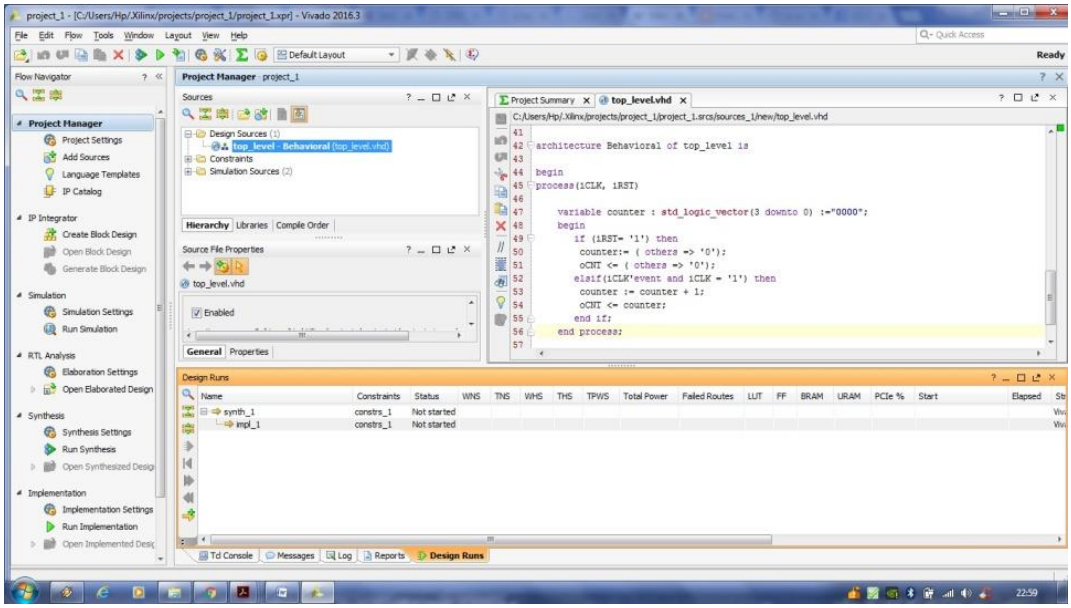
Resim 3.3 Python 1300-C Kamera Modülü (Avnet 2017).

PYTHON-1300-C Kamera Modülü, Avnet firmasının geliştirdiği ON Semiconductor PYTHON-1300 görüntü algılayıcısını üzerinde bulunduran FMC-HDMI-CAM Modülüne tam uyumlu olan kamera modülüdür. 1280 x 1024 (SXGA) çözünürlükte 210 fps'ye kadar görüntü aktarımını destekleyen modülde ayrıca dahili sabit desen gürültü düzeltme, otomatik pozlama kontrolü (Auto Exposure Control - AEC) ve yüksek dinamik aralık (High Dynamic Range - HDR) özellikleri bulunur (Avnet 2015).

3.7 Kullanılan Yazılım

3.7.1 Vivado Suite Geliştirme Ortamı

Çalışmada model tabanlı tasarım yöntemiyle oluşturulan akıllı özellik paketleri (IP Core) Xilinx firmasının Zynq mimarisi için özel olarak geliştirdiği Vivado Design Suite isimli yazılım ile donanım sistemine entegre edilerek sentezlenip gerçeklenmiştir.
Vivado Design Suite, Xilinx firmasının kendi FPGA ve SoC ürünleri için özel olarak geliştirdiği ISE Design Suite yazılımının devamı olan HDL tasarımlarını oluşturma ve bunları sentezleme, doğrulama, yerleştirme ve analiz gibi işlemleri yüksek seviyede otomatikleştirilmiş olarak yapabilen gelişmiş bir yazılım aracıdır (İnt.Kyn.5).



Resim 3.4 Vivado Suite Geliştirme Ortamı.

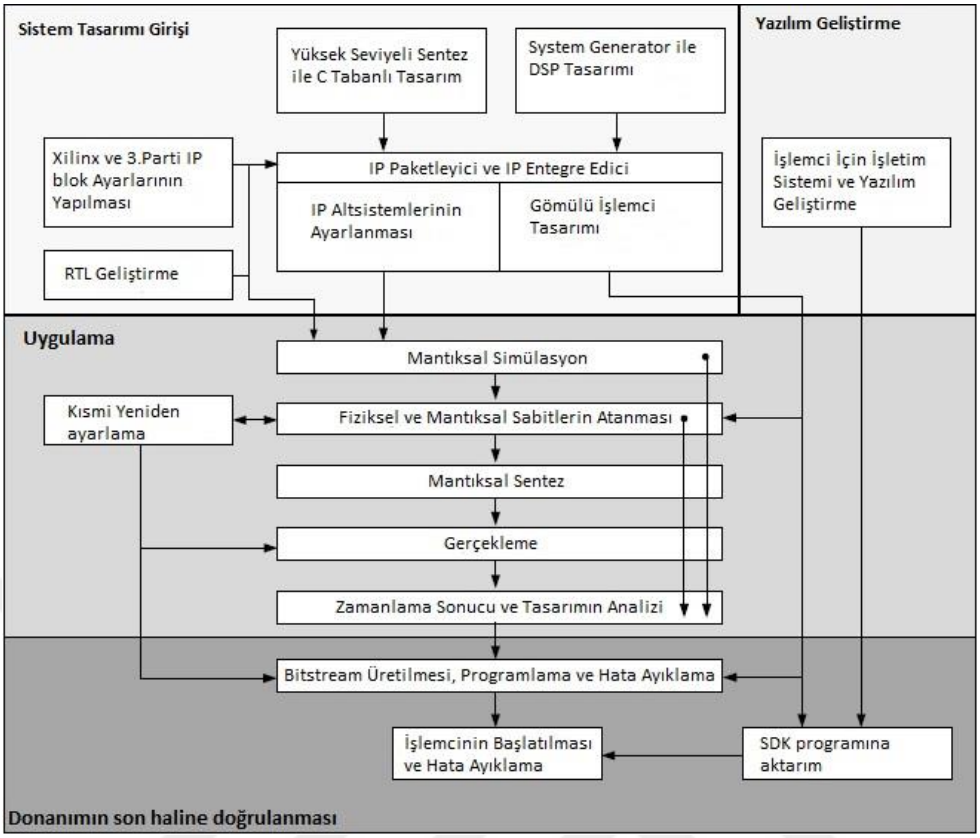
Önceki araçlardan farklı olarak gelişmiş bir dahili lojik simülatör barındıran Vivado yazılımı, aynı zamanda yüksek seviyeli sentez yapabilen HLS aracını da içinde barındırır. Sentezlenen donanım daha sonra SDK (Software Development Kit) yazılım geliştirme ortamına aktarılır ve burada donanıma uygun olarak sistemin yazılımı kodlanır. Tezde yapılan uygulamada Vivado 2016.4 sürümü ile ücretsiz olarak edinilebilen Webpack sürümü kullanılmıştır.

Vivado geliştirme ortamında geliştirme yapabilmek için öncelikle FPGA ve SoC'lerde kullanılan tasarım akışının anlaşılması gerekmektedir. Bu sistemlerde donanım geliştirme süreci Şekil.3.11.deki beş aşamadan oluşmaktadır (Xilinx 2015).

"UltraFast Degisn Methodology" adı verilen bu tasarım sürecinde sistemin planlanmasından donanımın oluşturulup buna göre yazılımın programlanarak son ürünün ortaya çıkartılmasına kadar olan tüm süreç Vivado geliştirme ortamında ve ona bağlı olan araçlarla tamamlanabilmektedir.

Şekil 3.11 SoC Donanım Tasarım Akışı (Xilinx 2015).

Resim 3.5.de sistemin yazılım geliştirme süreci de dahil olmak üzere komple tasarım akışı görülmektedir.



Resim 3.5 Vivado Suite Tasarım Akışı (Xilinx 2015).

3.7.2 SDK Yazılım Geliştirme Aracı

Deneysel çalışmalarda kullanılan yazılım Xilinx firmasının ürettiği gömülü yazılım geliştirme platformu olan SDK (Software Developlement Kit) yazılım geliştirme aracı ile tasarlanmıştır. SDK, Eclipse tabanlı bir geliştirme platformu olup, diğer Xilinx yazılımları ile en uyumlu şekilde çalışarak hızlı tasarım yapmayı sağlayan bir geliştirme platformudur. Tasarımda Xilinx Vivado 2016.4 Webpack ile birlikte gelen SDK 2016.4 kullanılmıştır.

3.7.3 MATLAB - Simulink ve HDL Coder

Matlab programı Mathworks firmasının özellikle mühendisler ve bilim adamları için geliştirdiği matris tabanlı gelişmiş bir programlama platformudur. Matlab ile kolayca nümerik analiz, model ve algoritma geliştirme yapılabilir, ya da istenen platform için kod üretilebilir (İnt.Kyn.6).

Matlabın gömülü sistemler ve FPGA tasarımlarının çok daha hızlı prototiplenip üretilmesi için çıkardığı HDL Coder ise Matlab kodlarından ve Simulink bloklarından okunabilir ve sentezlenebilir HDL kodu üretebilen bir sistem aracıdır. Ayrıca sahip olduğu iş akışı danışmanı (HDL workflow advisor) ile HDL kodu üretme sürecini tamamen otomatik hale getirmektedir (Mathworks 2017).

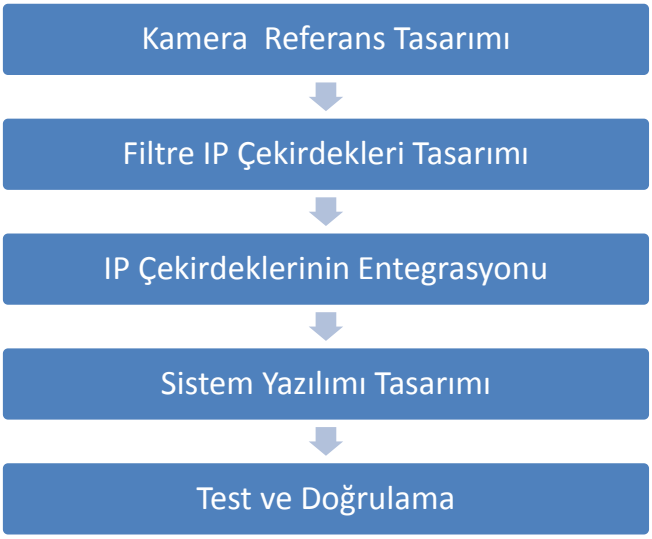
Bu araç sayesinde bir gömülü sistem mühendisi veya sistem tasarımcısı neredeyse hiç donanım tanımlama dili bilmeden komple bir sistemin tasarımını ve gerçeklemesini yapabilmektedir. Bunun yanında donanım bilgisine sahip olan mühendislerin dahi bu araçla geliştirme süresini çok daha fazla düşürmeleri olasıdır. Çünkü Matlab/Simulink hâlihazırda çok fazla kullanılmaktadır ve iyi bilinmektedir. Bu da algoritma geliştirmenin çok daha kolay ve hızlı olmasını sağlamaktadır. Bölüm 3.8.de gösterilen görüntü işleyici sistemi tamamen HDL Coder ile oluşturulmuştur.

3.8. Görüntü İşleyici Sistemi

Bu bölümde görüntü işleyici sisteminin tasarımı için kullanılan metotlar ve tasarım aşamalarından bahsedilecektir. Bunun için öncelikle kameradan görüntü alma işlemini gerçekleştiren referans tasarım oluşturulacak, daha sonra Simulink ortamında filtre algoritmalarının geliştirilmesi işlemi yapılacaktır.

Geliştirilen filtre IP paketleri Vivado Suite ortamında önceden oluşturulan kamera referans tasarımına entegre edilecektir. Son olarak sistemin sürücüleri ve ana kontrol yazılımı SDK ortamında geliştirilerek sistem çalışır duruma gelecektir.

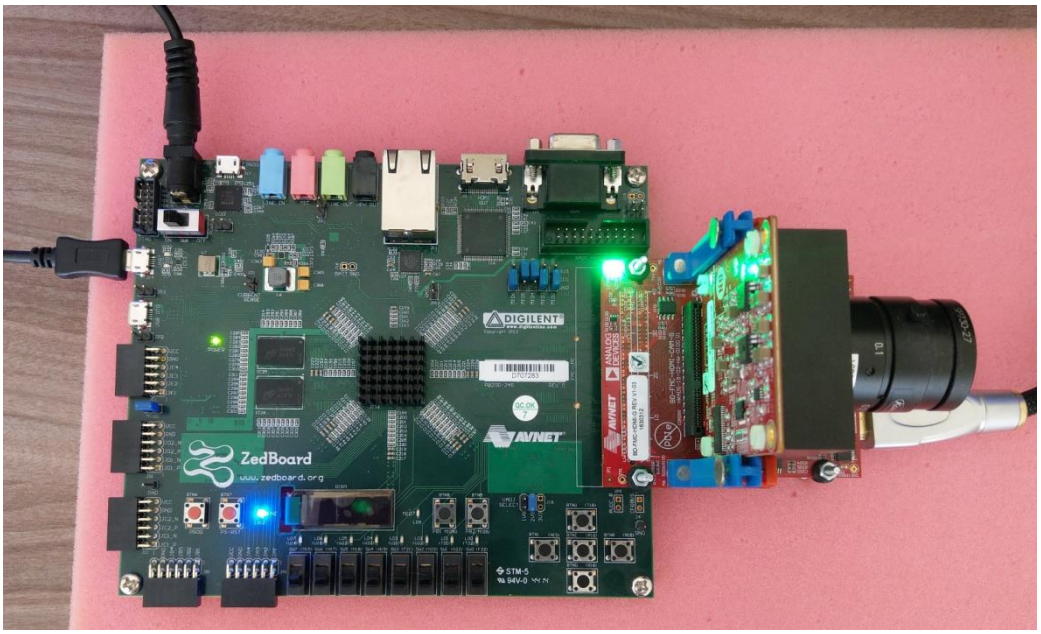
Şekil 3.12.de sistemin tasarım akış şeması verilmiştir.



Şekil 3.12 Görüntü İşleyici Sistemi Tasarım Akışı.

3.8.1 Kamera Referans Tasarımı

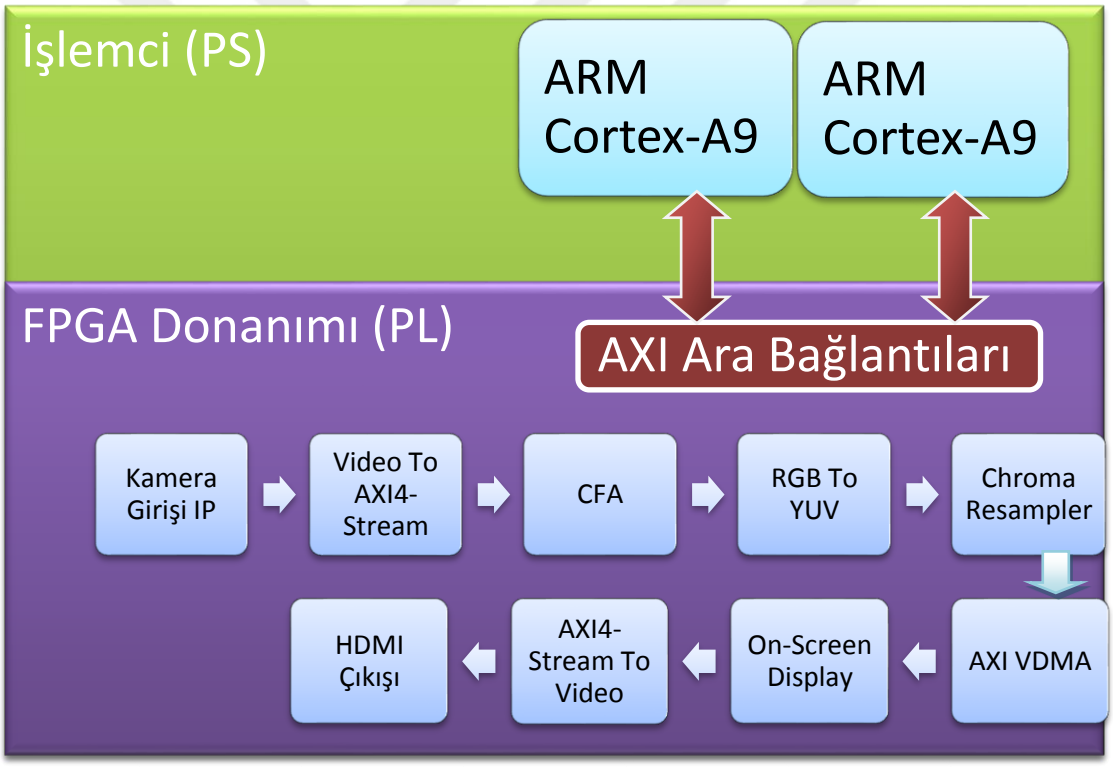
Zynq ortamında çalışır bir sistem oluşturabilmek için öncelikle temin edilen donanımların kullanılarak kamera görüntüsünü alıp çıkışı ekrana aktarabilecek bir referans tasarım oluşturulması gerekmektedir. Bunun için ilk olarak bölüm 3.6.1, 3.6.2 ve 3.6.3.de bahsedilen donanımlar uygun şekilde montaj yapılmış ve sistem Resim 3.6.daki gibi bir araya getirilmiştir.



Resim 3.6 Sistemin Genel Görünümü.

Bundan sonra Vivado üzerinde sistemin görüntüyü alabilmesi için gerekli donanımın oluşturularak sentezlenmesi gerekmektedir. Oluşturulacak tasarımda Bölüm 3.6.3.te bahsedilen özel bir kamera kullanılacağı için bu kameraya özel olarak hazırlanmış olan IP paketlerinin ve sürücülerin indirilip sisteme dahil edilmesi gerekmektedir (Avnet 2015).

Tasarımın sentezlenebilmesi için gerekli olan IP paketlerinin bir kısmı "Xilinx Video and Image Processing Pack" içerisinde bulunmaktadır ve ücrete tabidir. Deneme sürümü kullanmak için Xilinx web sitesinden ilgili paket seçilip ücretsiz deneme lisansı edinilerek IP paketleri kullanılabilir. Bu IP bloklarından bölüm 3.8.5.de daha detaylı olarak bahsedilmiştir (İnt.Kyn.7).



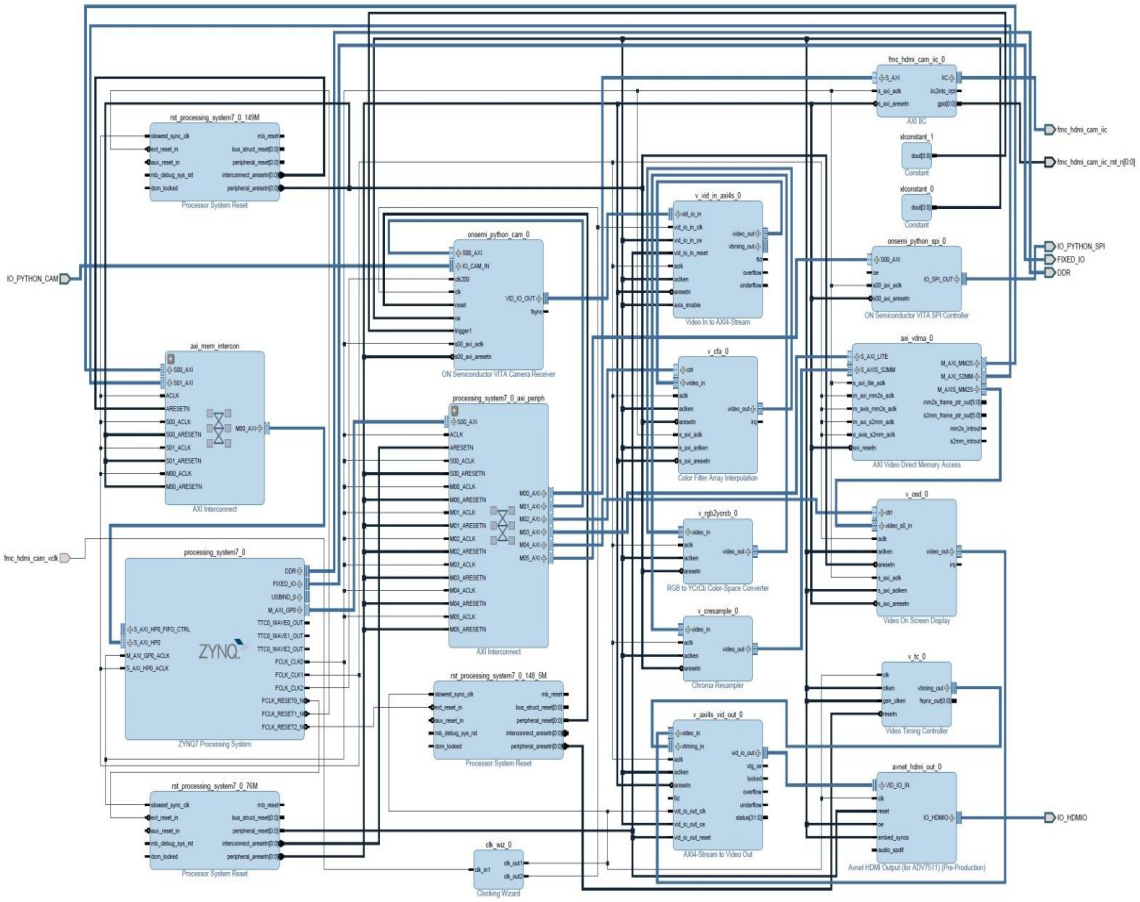
Şekil 3.13 Zynq Kamera Referans Tasarımı Şematik Gösterimi.

Referans tasarım tamamen donanım üzerinde gerçeklenmiştir. İşlemci kısmında ise ilgili sürücülerin ve çevre birimlerinin çağırılması ile parametrelerin kontrolü gerçekleştirilmektedir. Sistemin genel blok diyagramı Şekil 3.14.te gösterilmiştir.

Kullanılan hazır IP blokları sayesinde kamera donanımının tasarımı HDL kodu yazmadan hızlı bir şekilde sentezlenip gerçeklenebilmiştir. Bu sayede çok uzun sürebilecek olan tasarım ve test süreci hayli kısalmıştır.

Şekil 3.13.te görüldüğü gibi kamera görüntü aktarımı sistemi tamamen Zynq'in PL kısmında gerçeklenmiştir. FPGA'lerin paralel işlemeye uygun yapısı sayesinde sistemin gerçek zamanlı çalışması mümkün olabilmektedir.

PS kısmında ise AXI4-lite arabirimi üzerinden DDR belleğin kontrol edilmesiyle sistemin bileşenlerinin başlatılması ve sırasıyla ilgili IP'lerin devreye alınması sağlanmaktadır. Hazırlanan blok tasarım Şekil 3.14.de verilmiştir. Blok tasarımın tam boy çizimi Ek-1.de bulunabilir.



Şekil 3.14 Kamera Referans Tasarımı Blok Şeması.

Bu aşamada uygulamada kullanılan AXI veri yolunun yapısı ile VDMA çalışma prensibi ve Şekil 3.13.teki bazı önemli IP bloklarının işlevlerinin bilinmesi konunun daha iyi anlaşılması açısından önem arz etmektedir.

3.8.2 AXI Veri yolu Mimarisi

Gelişmiş Genişletilebilir Arayüz olarak adlandırılan AXI (Advanced eXtensible Interface) ARM AMBA mimarisinin bir parçası olan geliştirilebilir bir veri yolu standardıdır. İlk olarak 2003 yılında ortaya çıkan AXI, ARM firmasının mikro işlemci iletişim standardı olarak geliştirdiği AMBA 3.0 mimarisinin bir parçası olarak yayınlanmıştır. Günümüzde kullanılan sürüm olan AXI4 standardı ise 2010 yılında kullanılmaya başlanmıştır (Crockett *et al.* 2014).

AXI4 mimarisinin geliştirilmesinde ve FPGA sistemlerine entegre edilmesinde Xilinx ARM ile birlikte ortak çalışarak gerekli uyum ve entegrasyonu gerçekleştirmiştir. Bu sayede tüm Xilinx FPGA ve Zynq SoC geliştirme ortamlarında ve IP entegrasyonlarında AXI4 mimarisinin standart olarak kullanımının önünü açmışlardır. Bugün birçok IP geliştiricisi ve sistem tasarımcısı tarafından bu mimari kullanılmaktadır (Xilinx 2012).

AXI mimarisi Zynq tabanlı sistemlerde usta-köle (master-slave) ilişkisine göre işlemci ile donanım arasındaki iletişimi sağlamakla sorumludur ve AXI ara bağlantı elemanı (AXI Interconnect) vasıtasıyla kullanılır. AXI mimarisi genel olarak üç farklı tipte kullanılmaktadır. Bunlar AXI4-bellek eşlemeli (memory-mapped), AXI4-lite ve AXI4-akış (AXI4-stream) yapılarıdır.

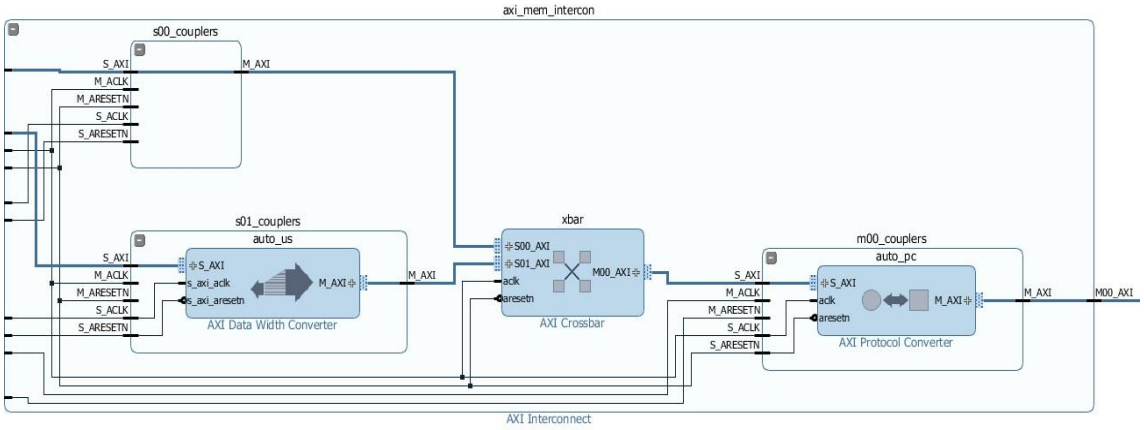
Bunlardan teorik olarak en hızlısı ancak en karmaşık olanı AXI4-bellek eşlemeli protokoldür. Bu protokolde tüm veri iletişimi bir bellek alanı içerisinde gerçekleşir. Bu yüzden hızlı olmasına rağmen kullanılabilecek kapasite kısıtlıdır. Ayrıca veri ile birlikte alıcı verisi adreslerinin öncelikle gönderilmesi gerekmektedir. AXI4-lite protokolü ise AXI-bellek eşlemeli modeli ile benzerlik taşımakla birlikte her seferinde tek bir veri kelimesi (data word) gönderimi/aktarımı sağlayabilir. AXI4-akış protokolü ise hızlı ve sürekli olarak yüksek seviyede veri aktarımının gerekli olduğu sistemlerde kullanılmak için tasarlanmış olup en önemli farklı FIFO arabelleği üzerinden verinin aktarılmasıdır.

Böylece kesintisiz ve sürekli olarak veri akışını gerçekleştirmek gereken gerçek zamanlı görüntü ve video işleme gibi uygulamalarda kullanılması çok avantajlı olmaktadır. AXI4-akış mimarisinin bir diğer önemli özelliği adresleme ihtiyacı ve gereği olmamasıdır. Bu mimaride veriler tek yönlü olarak ve usta-köle ilişkisine uygun biçimde tek yönlü olarak aktarılmaktadır (Xilinx 2017).

Çalışmada görüntü aktarımı sistemi üzerinde kullanılan IP bloklarının tamamı AXI4-stream mimarisi üzerinden birbirine bağlıdır. IP bloklarının devreye alınması ve parametrelerin ayarlanması ise AXI4-lite protokolü üzerinden işlemci üzerinde çalışan yazılım vasıtasıyla gerçekleşmektedir.

3.8.3 AXI Ara Bağlantıları

AXI ara bağlantı elemanı basitçe AXI veri yollarının birbirleri arasındaki trafiği yönlendiren bir anahtarlama mekanizmasıdır. AXI ara bağlantısı ortasında bulunan "AXI crossbar" çaprazlama elemanı ile ilgili usta ve köle birimleri arasındaki veri aktarımını gerçekleştirir. Böylece usta bağlantısından gelen bir veri köleye yönlendirilebilir ya da bunun tam tersi gerçekleşebilir. Ayrıca AXI ara bağlantısı ile AXI4-bellek eşlemeli ile AXI4-akış veri yolları arasındaki bağlantı da bu şekilde sağlanabilir (Shi 2016).



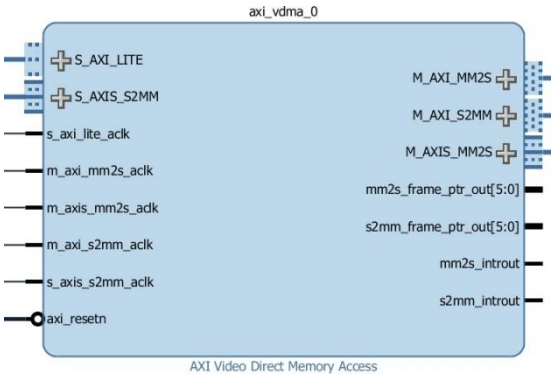
Şekil 3.15 AXI Ara Bağlantısı Blok Tasarımı.

Blok tasarımda fazla yer kaplamaması için Şekil 3.14.te bağlantılar küçültülerek gizlenmiştir. Sistemdeki AXI ara bağlantı elemanının açık şeması Şekil 3.15.te görülmektedir.

3.8.4 AXI VDMA

Doğrudan Bellek Erişimi olarak adlandırılabilecek olan DMA (Direct Memory Access) Xilinx'in AXI mimarisine adapte ettiği bir yapıdır. Bu yapının amacı sistem belleğine erişerek bellek ile AXI mimarisine sahip olan çevre birimlerinin doğrudan ve hızlı biçimde veri alışverişi yapabilmesini sağlamaktır. Çalışmada DMA motorunun video uygulamaları için özelleştirilmiş bir sürümü olan AXI VDMA (Video DMA) kullanılmıştır.

AXI VDMA motoru yüksek bant genişliği gerektiren video aktarımı ve görüntü işleme sistemlerinde bellek ile gerekli olan iletişimi sağlayan bir IP paketidir. Bir görüntü işleme sisteminin gerçek zamanlı olabilmesi için işlemci ile FPGA birimi arasındaki ilişkinin çok hızlı biçimde sağlanması gerekmektedir. VDMA motoru içerisindeki veri aktarıcı (data mover) birimi AXI4-bellek eşlemeli tip ile AXI4-akış birimleri arasındaki veri dönüştürme işlemlerini yapar. Bu sayede belleğe çok hızlı biçimde veri aktarılabilir ve bu verilere sistemin işlemci birimi tarafından DDR bellek üzerinden kolaylıkla erişilebilir (Xilinx 2017).



Şekil 3.16 AXI VDMA IP Bloğu.

Şekil 3.16.da VDMA motorunun IP blok görüntüsü verilmiştir. Şekilde de görüldüğü gibi VDMA motoru çift yönlü olarak çalışmaktadır. Veri aktarıcı birimi üzerinden

S_AXIS_S2MM girişi AXI4-akış biriminden gelen veriyi okur. Bu arada M_AXI_M2SS dönüştürülen veriyi DDR belleğe yazar. Aynı anda yine veri aktarıcıya bağlı olan M_AXI_MM2S DDR bellekten gelen veriyi okur ve dönüştürülmüş olan veriyi AXI4-akış veri yolu üzerinden görüntünün aktarılacağı OSD IP paketine yazar.

3.8.5 Xilinx Görüntü İşleme Paketi IP Blokları

Bölüm 3.8.1.de bahsedildiği üzere çalışmada kullanılan kameradan görüntüyü doğru bir biçimde alabilmek için "Xilinx Video and Image Processing Pack" paketi kapsamındaki IP bloklarının tasarımda kullanılması gerekmektedir. Bu paket içerisinde aşağıdaki beş adet IP bloğu ücretsiz deneme lisansı ile kullanılmıştır.

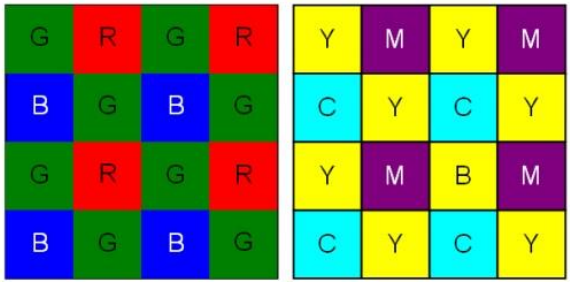
- Renk Düzen Aralığı Ekleyici - Color Filter Array Interpolation (CFA)
- Chroma Yeniden Örnekleyici - Chroma Resampler
- Video Ekranda Gösterici - Video On Screen Display (OSD)
- RGB-YCrCb Renk Uzayı Dönüştürücü -RGB to YCrCb Color-Space Converter (CSC)
- Video Zamanlayıcı Kontrolcüsü - Video Timing Controller (VTC)

3.8.5.1 Renk Düzen Aralığı Ekleyici

CFA IP bloğu kısaca kamera görüntü sensöründen gelen görüntünün renk deseni içerisindeki eksik olan renk bileşenlerini üretme görevi üstlenmektedir. CCD veya CMOS tipi algılayıcılar görüntünün yalnızca Y bileşeni adı verilen parlaklık bileşenini algılarlar. Oysaki RGB renk uzayında renkli bir sayısal görüntüden bahsedebilmek için her piksel için kırmızı, yeşil ve mavi değerleri bulunmalıdır. Kodak bilim adamı Dr. Bryce Bayer tarafından bulunan yöntemde görüntü sensörünün önüne Bayer deseni adı verilen renk desenlerinde filtreler uygulanıp böylece renkli görüntünün oluşması sağlanır (Xilinx 2015).

Bayer deseninde uygulanan filtrelerde kırmızı, yeşil ve mavinin ikili kombinasyonları bulunur. Bu kombinasyonlar sensörden gelen görüntünün ilk pikselinin değerine göre de değişkenlik gösterir.

Bu kombinasyonlar RGB renk uzayı için K kırmızıyı, Y yeşili ve M maviyi temsil etmek üzere KYKY, YKYK, YMYM, MYMY şeklindedir.



Resim 3.7 RGB ve CMY Bayer Desenleri (Xilinx 2015).

Ayrıca CFA IP bloğu sistemin çalışma anında gerçek zamanlı olarak parametreler değiştirilerek Bayer deseni belirlenebilir. Çalışmada olası dört Bayer desenini görüntüye uygulayan sistem Bölüm 4.te gerçeklenmiştir.

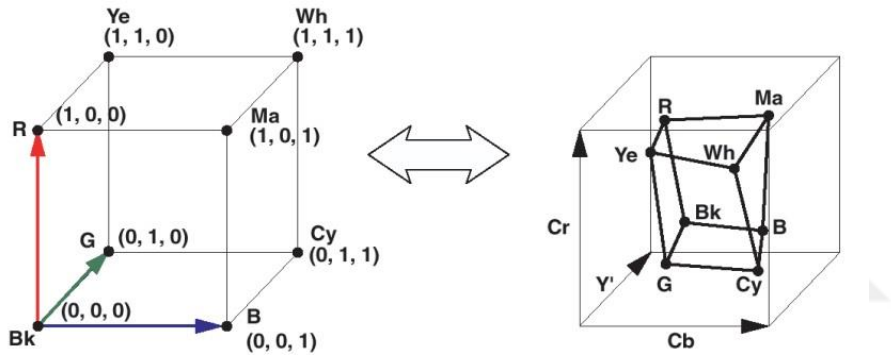
3.8.5.2 RGB-YCrCb Renk Uzayı Dönüştürücü

Renk uzayı kavramı, bir renk dizisinin matematiksel olarak gösterimini ifade eder. Bilgisayar ve sayısal görüntüleme sistemlerinde genel olarak kullanılan başlıca renk uzayları bulunur. Bunlardan günümüzde en fazla kullanılanları RGB ve YUV, YCrCb renk uzaylarıdır. Bu renk uzaylarının kullanımı her sistemde değişiklik gösterebilir. Örneğin bilgisayar sistemlerinde ağırlıklı olarak RGB renk uzayı kullanılırken video görüntüleme ve aktarım standartları genellikle YUV ve YCbCr formatlarını kullanmaktadırlar. Bu sebeplerden dolayı renk uzayları arasında dönüşüm yapılması gereği doğmuştur.

RGB renk uzayı bilgisayar sistemlerinde görüntünün oluşturulmasında oldukça fazla kullanılır çünkü sadece üç ana rengin tonları karıştırılarak istenen renkler elde edilebilir ve sistemin çalışacağı yazılım algoritmaları genellikle bu renk uzayında çalışmaya uygundur.

YUV renk uzayı ise PAL/NTSC gibi analog televizyon standartlarında kullanılmış olan bir sistemdir. Özellikle siyah-beyaz yayınların bulunduğu dönemde ilk renkli yayınlara geçişte bu standart mevcut sistem üzerinden renkli görüntü aktarımını olanaklı kılmıştır.

Burada Y parlaklık (luminance/intensity) bilgisine, U ve V bileşenleri ise renklilik (chrominance) bilgisine karşılık gelir. YCbCr formatı ise YUV formatının geliştirilmiş ve düzenlenmiş sürümüdür. Bu format uluslararası telekomünikasyon örgütünün ITU-R BT.601 kapsamında dünya çapında kullanılacak bir video aktarım standardı geliştirmesi sonucu ortaya çıkmıştır. Bu formatta Y parlaklık bilgisini, Cb (Chroma Blue) mavi renk bileşenini, Cr (Chroma Red) ise kırmızı renk bileşenini temsil eder. Buradaki renklilik asıl olarak renk farkını gösterir (Xilinx 2015).



Resim 3.8 RGB ve YCbCr Renk Uzayları Vektörel Gösterimi (Xilinx 2015).

Sistemde kullanılan görüntü işleme IP blokları YCbCr formatına göre tasarlanmıştır. Bunun iki temel sebebi vardır. Birincisi bu renk düzeni ile gri tonlamalı görüntü kolaylıkla elde edilebilir. Bunun için görüntünün sadece Y bileşenini almak yeterli olacaktır. İkincisi ise hesaplama kolaylığıdır. RGB renk uzayında görüntünün tamamı ya da bir parçası üzerinde yapılacak tüm işlemlerde tüm renklerin bilgisi okunup değiştirilerek yazılmalıdır. Bu da maliyet artışı ve hız kaybı anlamına gelecektir.

Sistemde YCbCr renk uzayı SD-ITU 601 standardına uygun olarak kullanılmıştır. Buna göre renklerin oluşturulmasında kullanılacak olan katsayılar aşağıdaki gibi hesaplanmıştır:

$$Y = R\,x\,0{,}299 + G\,x\,0{,}587 + B\,x\,0{,}114 \quad (3.1)$$

$$Cb = (B - Y)\,x\,0{,}564 \quad (3.2)$$

$$Cr = (R - Y)\,x\,0{,}713 \quad (3.3)$$

3.8.5.3 Chroma Yeniden Örnekleyici

İnsan gözü yapı itibariyle renkten ziyade parlaklığa daha duyarlıdır. Bu sebeple renk farklılığını ifade eden Cb ve Cr bileşenleri sıkıştırılarak görüntü daha hızlı ve daha az veri kullanılarak aktarılabilir ve bunun sonucunda görüntüde gözle görülebilir bir kayıp olmayacaktır (Xilinx 2015).

Bir önceki bölümde bahsedilen renk uzayı dönüştürücü IP bloğu ile YCbCr renk uzayına dönüştürülen görüntü YCbCr 4:4:4 formatındadır. Bu formattaki video sinyali 3 x 8 = 24 bit bant genişliğine sahip olacaktır. Chroma Resampler IP bloğundan geçen video sinyalinin chroma sinyalleri yatay düşük geçişli FIR filtreler kullanılarak yarıya düşürülür ve böylece video 8 + 4 + 4 = 16 bit bant genişliğine göre sıkıştırılmış olur.

3.8.5.4 Video Ekranda Gösterici

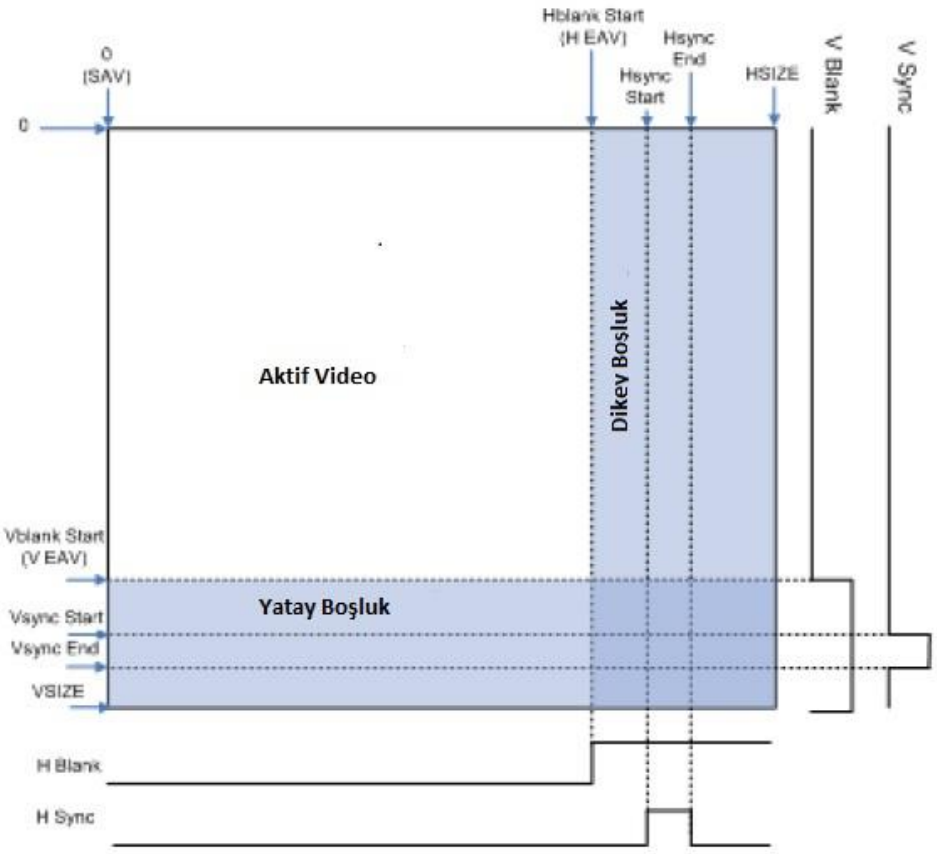
Xilinx OSD IP bloğu farklı kaynaklardan gelen video görüntülerinin ekrana yerleştirilmesi ve bunların iç içe veya karışım biçiminde gösterilmesini sağlayan bir sistemdir. Programlanabilir 8 katmana kadar videoyu ekranda gösterebilir ve bunun yanında ekrana basit karakter veya şekiller çizdirebilen bir dahili grafik kontrolcüsü de bulunmaktadır (Xilinx 2015).

Tezde önerilen tasarımda kullanılan yapıda görüntü HDMI üzerinden aktarıldığından çıkış çözünürlüğü 1920 x 1080 piksel olacaktır. Kullanılan kameranın çözünürlüğü ise 1280 x 1024 piksel olduğundan kamera görüntüsü OSP IP bloğu kullanılarak tam kare içerisine ortalanarak yerleştirilmiştir.

3.8.5.5 Video Zamanlayıcı Kontrolcüsü

Bir video karesi görüntü sinyallerinin belirli bir zamanlama ile ekrana gönderilmesi vasıtasıyla oluşur. Bu zamanlama kaynak ve alıcı birimlerde farklılık gösterdiği takdirde görüntü doğru biçimde aktarılamayacaktır. Zamanlama açısından bir video karesi bir kaç bileşenden oluşur. Temel olarak video karesi aktif video ve zamanlama sinyalinin bulunduğu boşluk kısmından oluşur (Xilinx 2017).

Resim 3.9.da örnek bir video karesi üzerinde aktif görüntü ve zamanlama sinyalleri gösterilmiştir. Resimde gösterilen "Hsync" yatay senkronizasyon sinyalini, "Vsync" dikey senkronizasyon sinyalini gösterir. "Horizontal blanking" ve "Vertical blanking" sırasıyla yatay ve dikey zamanlama sinyalleri arasındaki boşluğu tanımlar. Bu sinyallerin değerleri farklı görüntü aktarım sistemlerinde değişiklik gösterebilir. Ancak gerçek zamanlı bir görüntü işleme sisteminde bu sinyallerin birbirleriyle tam olarak uyumlu olması gerekmektedir.



Resim 3.9 Bir Video Karesinin Zamanlama Sinyalleri Gösterimi (Xilinx 2017).

Video Zamanlayıcı Kontrolcüsü IP paketi görüntüyü AXI4-akış tipine çeviren IP bloğuyla birlikte çalışarak giriş videosundan gelen zamanlama ve aktif video sinyallerini algılayarak buna göre gerekli çıkış sinyallerini senkronize biçimde üreten bir sistemdir. Böylece her video karesi için gerekli olan zamanlama değeri doğru sağlanarak video görüntüsünün kesintisiz bir biçimde bozulma olmadan aktarılması mümkün olmuştur.

Bunların dışında kameradan veri alabilmek için ve SPI iletişim için iki adet, görüntüyü AXI4-akış tipine çevirmek ve AXI4-akış tipinden tekrar dönüşüm için iki adet ve FMC-HDMI modülü üzerindeki HDMI çıkışından görüntü alabilmek için bir adet daha IP bloğu tasarımda mevcuttur. Bu IP paketleri ise üreticinin sitesinden indirilebilir ve tamamen ücretsiz olarak kullanılabilir (İnt. Kyn.8).

Buraya kadar anlatılan sistem gerçeklenerek Zedboard üzerinde çalıştırılmıştır. Tasarlanan sistemin görüntüsü Resim 3.10.da verilmiştir.

Resim 3.10 Kamera Referans Tasarımı Sistemi Görüntüsü.

3.9 Görüntü İşleyici Filtre Sistemleri

Bu bölümde kullanılan görüntü işleme ve filtre algoritmaları ile bunların Matlab ortamında gerçeklenmesi hakkında bilgi verilecek ve referans tasarım üzerine MATLAB/Simulink ortamında tasarlanan görüntü işleme filtrelerinin entegre edilmesi adımları gösterilecektir. Bu aşamada hem Xilinx hem de Mathworks araçları birlikte kullanılacaktır.

Önceki bölümlerde bahsedildiği üzere Matlab ile Zynq-7000 üzerinde çalışacak IP paketleri oluşturabilmek için bazı ek paketlerin daha kullanılması gerekmektedir. Tasarlanan uygulamada Matlab R2017b 64 Bit ve Xilinx Vivado 2016.4 sürümleri kullanılmıştır. Ayrıca Matlab üzerine aşağıdaki araçlar ve ek paketlerin kurulması gerekmektedir.

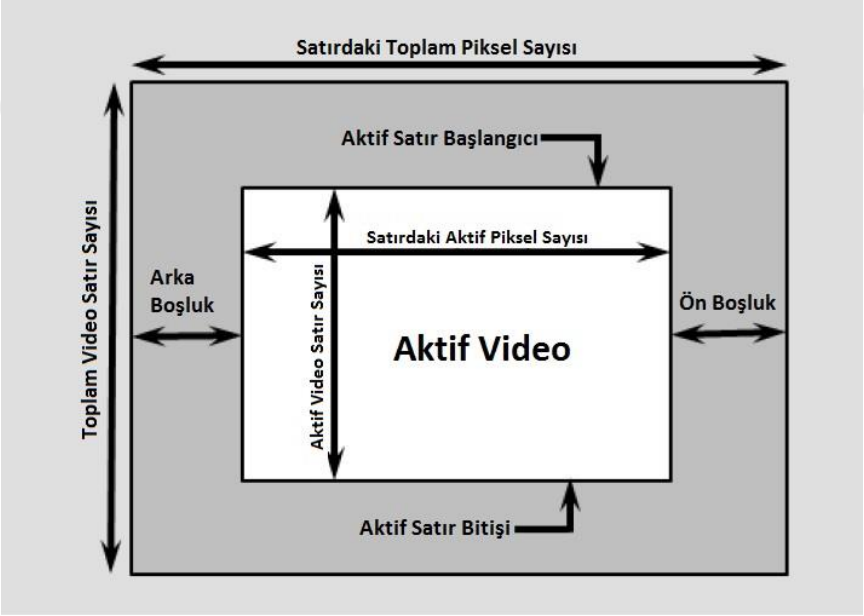
- Embedded Coder
- Computer Vision System Toolbox
- HDL Coder
- Matlab Coder
- Simulink Coder
- Vision HDL Toolbox
- Embedded Coder Support Package for Xilinx Zynq-7000 Platform
- HDL Coder Support Package for Xilinx Zynq-7000 Platform
- Computer Vision System Toolbox Support Package for Xilinx Zynq-Based Hardware

Görüntü işleyici sistemi genel olarak Simulink ortamında Vision HDL Toolbox ve HDL Coder araçları ile blok tasarım metoduyla gerçeklenmiştir. Akan piksel ara yüzü (Streaming Pixel Interface) olarak adlandırılan bu yöntem AXI4-Akış standardıyla birebir uyumludur.

Buna göre görüntü seri olarak pikseller halinde aktarılır ve veri ile kontrol olmak üzere iki tip sinyal mevcuttur. Veri sinyalleri piksel değerlerini gösterirken kontrol sinyalleri

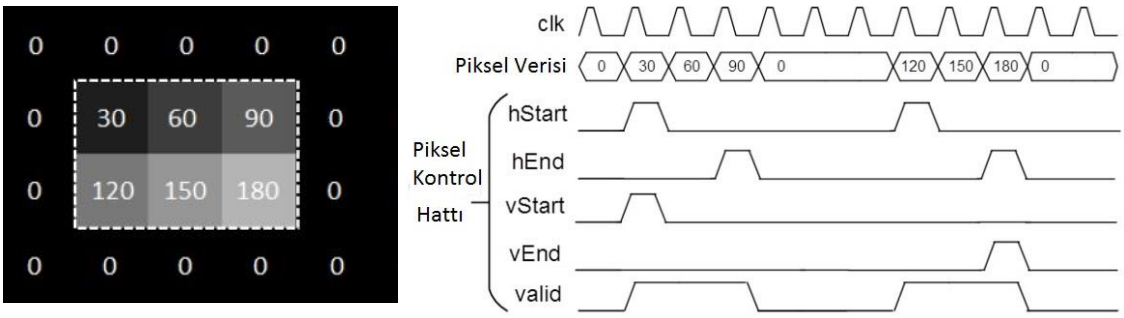
ise görüntü içerisindeki piksellerin konumunu belirtir. Böylece video boşluk sinyalleri de dahil olmak üzere video zamanlamasını birebir kopyalayarak çözünürlükten bağımsız esnek bir video platformu meydana gelir.

Video görüntüsü ekranda soldan sağa ve yukarıdan aşağıya doğru sinyaller şeklinde iletilir. Bu arada zamanlamayı ayarlamak maksadıyla bazı boşluk sinyalleri de gönderilir. Bunlardan yatay olanı genellikle iki parça halindedir. Bunlar ön boşluk (front porch) ve arka boşluk (back porch) olarak isimlendirilir. Ön boşluk aktif video satırının bitişi ile senkronizasyon sinyalinin başlangıcı arasındaki sinyallerdir. Arka boşluk ise senkronizasyon sinyali ile aktif video satırı arasındaki sinyallerdir. Dikey boşluk sinyali ise video karesinin aktif sinyalinin bitimi ile sonraki karenin aktif sinyali arasındaki aktif olmayan sinyallerdir. Sinyalin doğru aktarılabilmesi için hem dikey hem de yatay eksende başlangıç ve bitiş sinyallerinin verilmesi gerekmektedir. Resim 3.11.de bu yapı görülebilir (Mathworks 2017).



Resim 3.11 Aktif Video ve Boşluk Sinyalleri (Mathworks 2017).

Resim 3.12.de örnek bir video karesi ve zamanlama diyagramı gösterilmiştir. 2 x 3 piksel ölçülerindeki karede altı adet piksel değeri bulunmaktadır. Etrafındaki sıfır değerli sinyaller ise yatay ve dikey boşlukları belirtir.

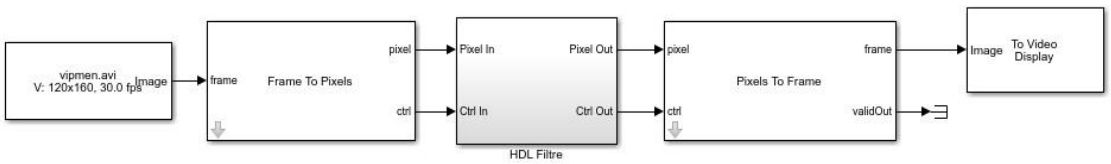


Resim 3.12 Örnek Video Karesi ve Zamanlama Diyagramı (Mathworks 2017).

Piksel kontrol veriyolu video akışı için olan zamanlama sinyallerini üretir."hStart", "hEnd", "vStart", "vEnd" ve "valid" olmak üzere beş sinyalden oluşur. Bu sinyaller video akışında piksellerin ne zaman başlayacağını ve ne zaman sonlanacağını belirtir. Burada "hStart" yatay görüntü çizgisinin başladığını, "hEnd" ise yatay görüntü hattının bittiğini bildirir. "vStart" ve "vEnd" sinyalleri de görüntü karesinin başlangıç ve bitiş değerlerini bildiren sinyallerdir. "valid" sinyali ise görüntü verinin gönderildiği zamanlarda aktif olur.

Sistemin genel blok yapısı Şekil 3.17.de gösterildiği gibidir. Simulink araçlarında bulunan blokların RGB renk uzayında işlem yapmasından dolayı video kaynağından gelen görüntü "Frame To Pixels" bloğuyla yukarıda belirtildiği yöntemle seri hale getirilerek piksel akışı formatına çevrilir ve filtre bloğuna girerek gerekli işlemlerden geçirilir.

İşlenmiş görüntü "Pixels to Frame" bloğu vasıtasıyla tekrar kare formatına dönüştürülür ve böylece sistemin tepkisi bilgisayar ortamında izlenerek gerekli değişiklikler HDL koduna dönüştürülmeden önce yapılabilir.



Şekil 3.17 Video İşleyici Sistemi Blok Diyagramı.

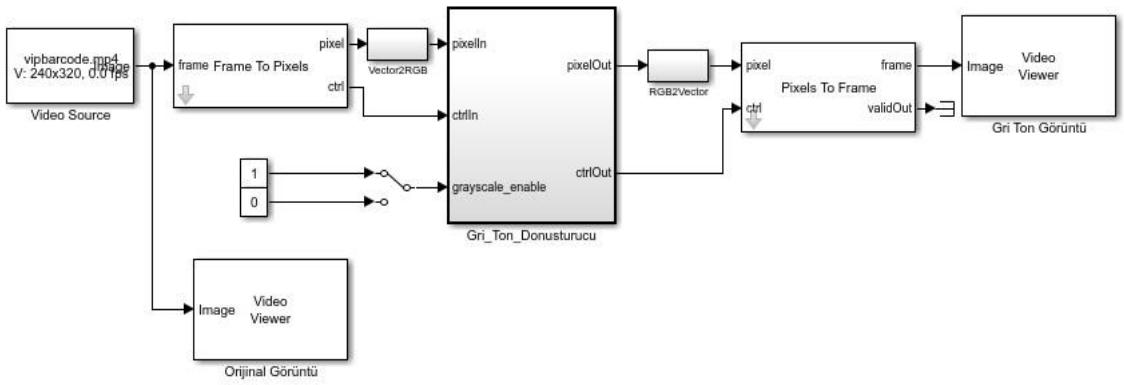
3.9.1 Gri Ton Dönüştürme

İlk olarak uygulanan işlem gri ton dönüştürmedir. Gri ton dönüştürme filtresi görüntüdeki piksel değerleri üzerinde işlem yaparak renkli görüntüyü gri tonlamalı görüntüye çevirir. Gri tonlamalı görüntüde her piksel siyah ile beyaz arasında bir değer alır. Bu değer 8 bit uzunluğa sahiptir.

YCbCr formatı gibi formatlarda gri tonlamalı görüntü sadece Y bileşeni alınarak bulunabilir. Burada tasarlanan sistem ise RGB renk uzayında çalıştığından "RGB To Intensity" bloğu kullanılarak sistem oluşturulmuştur. Buna göre kullanılan katsayılar aşağıdaki gibidir (Mathworks 2017).

$$piksel\ değeri\ =\ [0{,}299 \quad 0{,}578 \quad 0{,}114] \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.4}$$

Geliştirilen sistemin blok diyagramı Şekil 3.18.de gösterilmiştir.

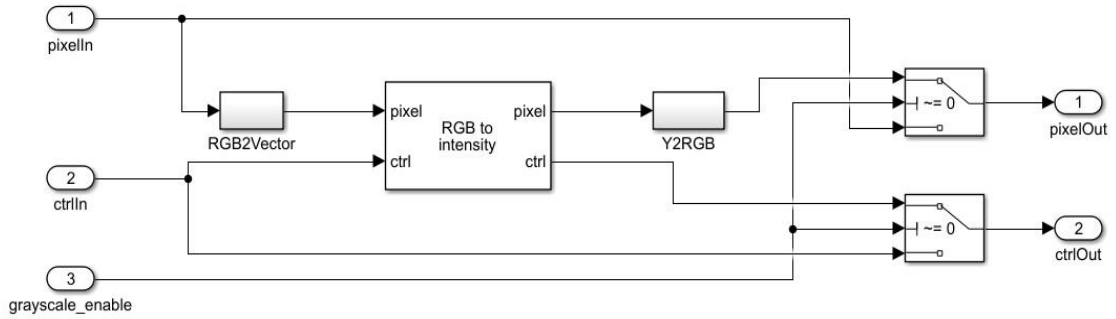


Şekil 3.18 Gri ton Dönüştürücü Blok Diyagramı.

HDL koduna dönüştürülecek olan sistemlerin Simulink ortamında bir alt sistem olarak tanımlanması gerektiğinden sistem buna uygun olarak tasarlanmıştır. Giriş videosu öncelikle AXI4-akış mimarisine uygun olarak piksel formatına dönüştürülmüştür. Ardından vektörel formatın 32 bitlik veriye dönüşümü "Vector2RGB" bloğunda yapılmıştır. Sistemin çıkışında aynı durum ters şekilde devam ederek çıkan pikseller önce vektörel tipe sonra da video karelerine dönüştürülerek ekranda gösterilmiştir. Filtrenin

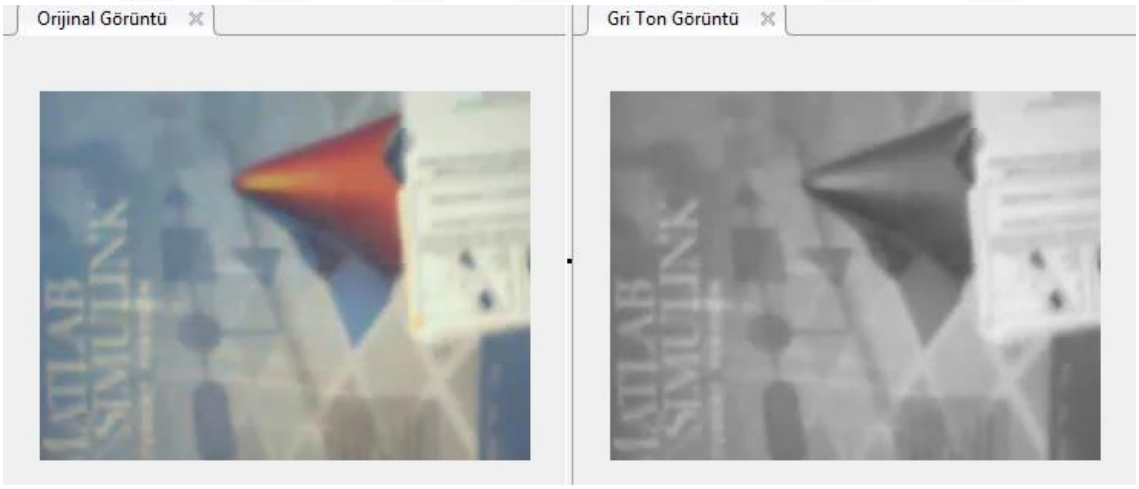
etkisini görebilmek için bir anahtar sisteme eklenmiştir.

Filtre alt sistemi ise Şekil 3.19.da verilmiştir. Sistemde tekrar vektörel dönüşüme geçilmesinin sebebi Vision HDL bloklarının vektör veri tipi ile çalışmasıdır. Bu sebepten sistem çıkışında tekrar 32 bitlik veri tipine dönüşüm yapılmıştır.



Şekil 3.19 Gri Ton Dönüştürücü IP Bloğu.

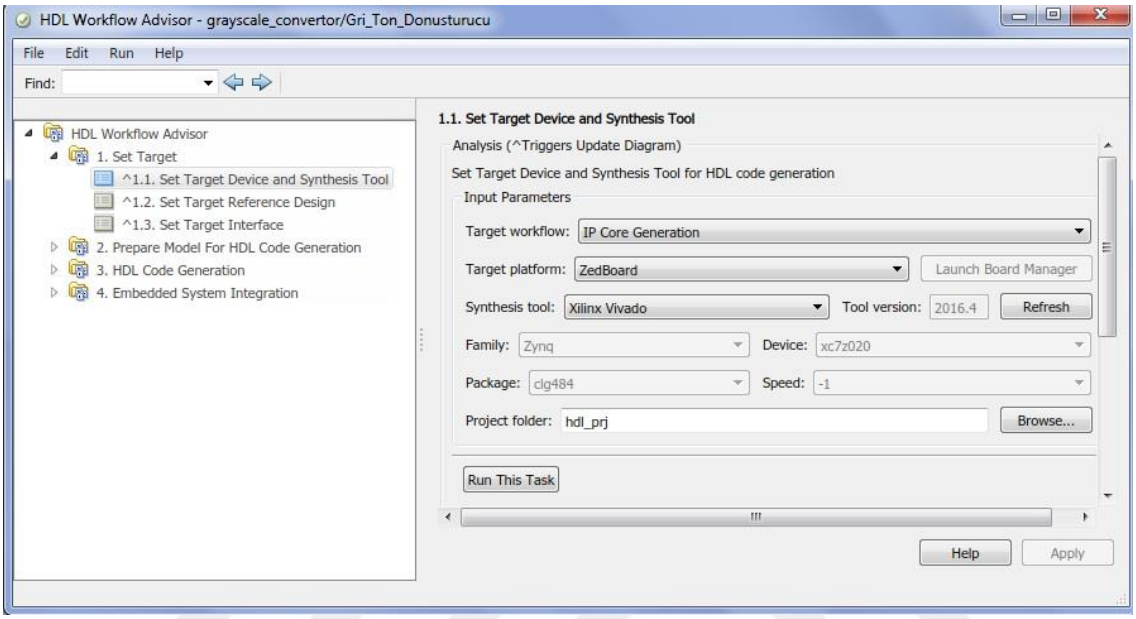
Oluşturulan sistemin simülasyon sonucunda başarılı bir şekilde gri ton dönüşümünü sağladığı görülmüştür. Sistemin çıktısı Resim 3.13.de görüldüğü gibidir.



Resim 3.13 Gri Ton Dönüştürücü Filtre Sonucu.

Sistem sorunsuz biçimde çalıştıktan sonra HDL Coder ile dönüşüm için gerekli işlemler yapılacaktır. Bunun için öncelikli olarak derleyiciyinin Matlab'a tanıtılması gerekmektedir. Çalışmada kullanılan Vivado Design Suite 2016.4 bu görevi yapmaktadır.

Bunun için Matlab komut ekranına "hdlsetuptoolpath('ToolName','Xilinx Vivado','ToolPath','C:\Xilinx\Vivado\2016.4\bin\vivado.bat')" komutu yazılarak derleyici sisteme tanıtılır. Daha sonra tasarlanan alt sistem bloğu üzerinde sağ tıklanarak "HDL Code>HDL Workflow Advisor" tıklanarak HDL iş akışı danışmanı açılır.

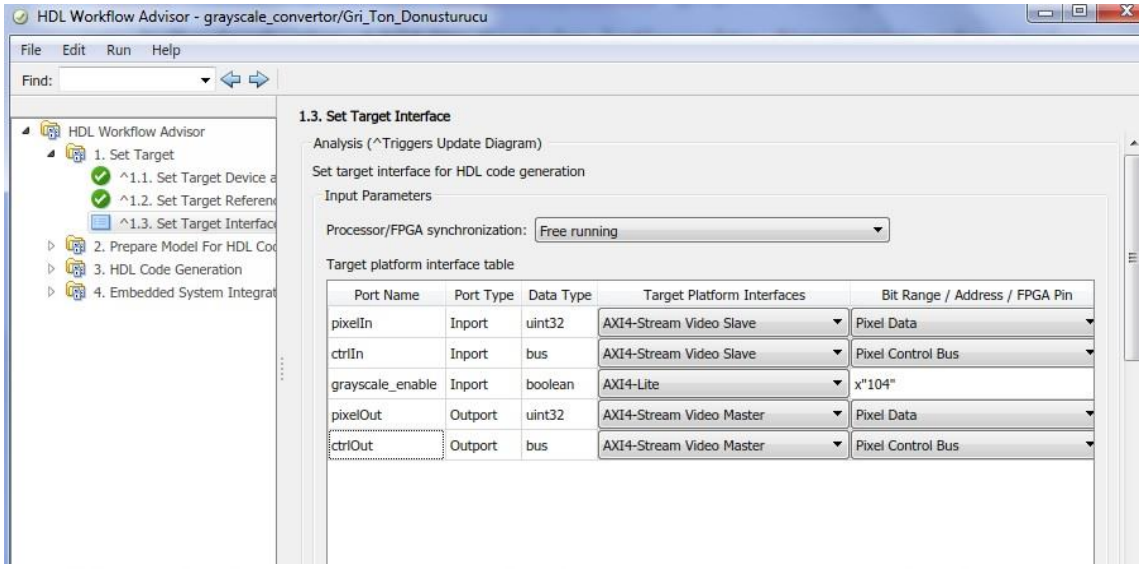


Resim 3.14 HDL İş Akışı Danışmanı.

İş akış danışmanında kod üretici için gerekli ayarlamalar yapılır. Hedef iş akışı olarak "IP Core Generation" ve hedef platform olarak kullanılan Zedboard seçilerek devam edilir. Sonraki aşamada "Default Video System" seçilerek AXI4-akış standardına uygun olarak sistem ayarlarının yapılması sağlanır.

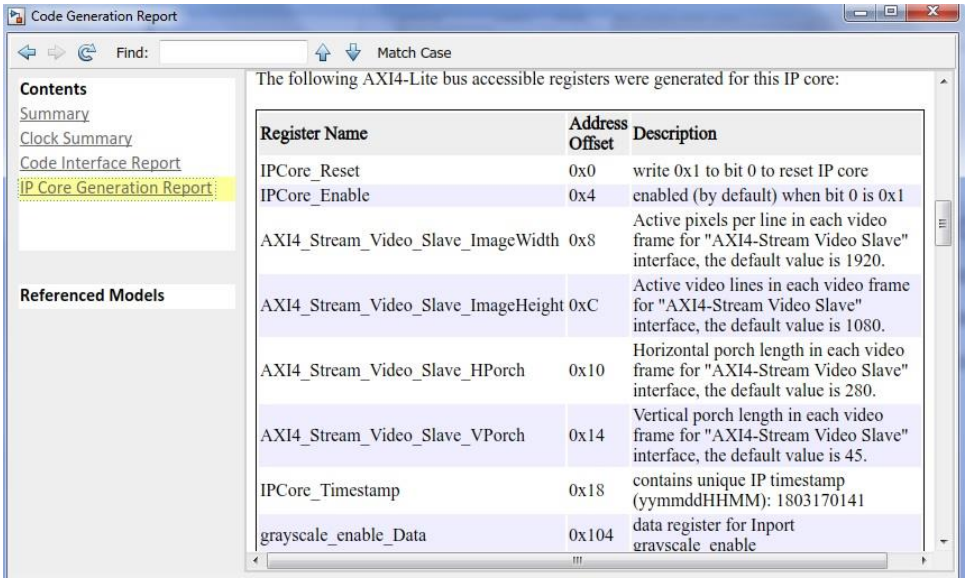
Resim 3.15.deki Aşama 1.3.de sistemin giriş ve çıkışları görünmektedir. Burada her veri yolu için doğru veri tiplerinin seçildiğinden emin olunmalıdır. Video işleme sisteminde girişler "slave" çıkışlar ise "master" olarak seçilmelidir. "Grayscale_enabled" girişi ise IP bloğunu kontrol etmek için kullanılacak olup AXI4-lite üzerinden bağlanacaktır.

Sistem doğrudan donanım üzerinde çalışacağından senkronizasyon modu "Free running" olarak ayarlanmalıdır. Sistemin adres ayarları Resim 3.15.de görüldüğü gibi yapılmıştır.



Resim 3.15 HDL IP Adres Ayarları.

İkinci bölümde model ile ilgili olan genel ayarlar değiştirilmeden devam edilir. Üçüncü bölümde kodun üretileceği dil (verilog ya da vhdl) ve optimizasyon ayarları yine varsayılan olarak bırakılır. Resim 3.15.teki 3.2.kısımda ise IP paketine isim verilerek onaylanır ve böylece IP paketi oluşturma işlemi tamamlamış olur. Dördüncü bölüm sistemde kullanılmayacağından dolayı iş akış danışmanından çıkılabilir.



Resim 3.16 HDL IP Kod Üretici Raporu Kaydedici Adresleri.

Bu adımda Resim 3.16.daki otomatik olarak üretilen kod üretici raporu incelendiğinde sistem hakkında daha detaylı bilgiye sahip olunabilir. Ayrıca sistemin kontrol parametreleri ve adresleri de bu raporda verilmiştir. Bu parametrelerin sistemin yazılımı hazırlanırken kullanılması gerekmektedir. Verilen adreslerden IP paketinin başlatılması ve yeniden ayarlanması dışında Bölüm 3.9.da bahsedilen boşluk sinyallerinin sayısının da ayarlanması bu değişkenlerden yapılmaktadır.

3.9.2 Kenar Bulma

Kenar bulma sayısal görüntü işlemede en fazla kullanılan tekniklerden birisidir. Kenar bulmanın amacı nesnelerin kenarlarını belirginleştirerek görüntüden öznitelik çıkarımı ve nesnelerin tanınması gibi işlemlerin kolaylaştırılmasıdır. Literatürde farklı kenar bulma yöntemleri bulunmakla birlikte bu çalışmada kullanılan yöntem Sobel Metodu olarak bilinen yöntemdir.

Sobel operatöründe gri ton görüntü üzerinde yatay ve dikey olmak üzere iki türevsel matris ile maskeleme yapılarak görüntünün gradyan bileşenleri elde edilir. Elde edilen bileşenlerin ağırlıkları hesaplanarak kenarları bulunmuş görüntü oluşturulur.

Bölüm 3.9.1.de olduğu gibi bu filtrede Simulink üzerinde Vision HDL Edge Detector bloğu kullanılarak gerçeklenmiştir. Bu blokta gradyan 45 ve 135 derecelerde belirlenmiştir. Gradyan hesaplama matrisleri aşağıdaki gibidir (Mathworks 2017).

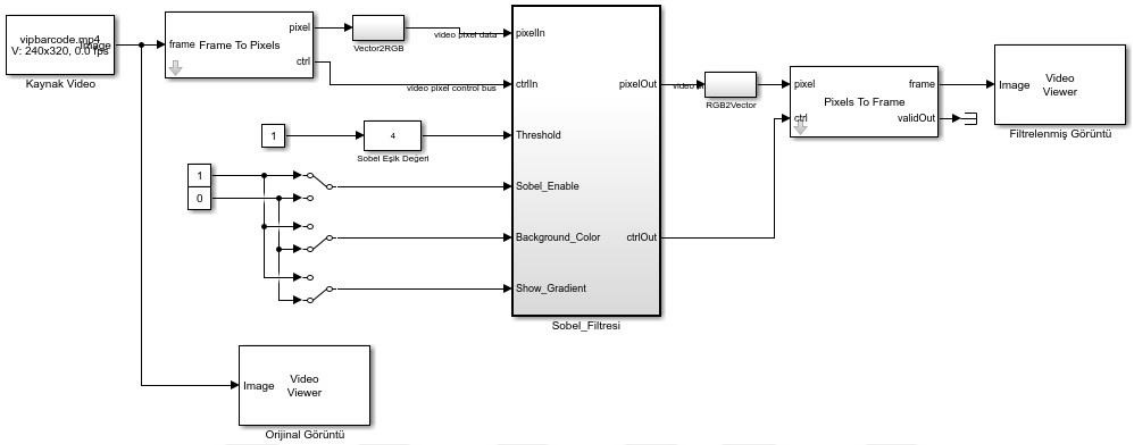
$$\mathrm{Gy} = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{3.5}$$

$$\mathrm{Gx} = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.6}$$

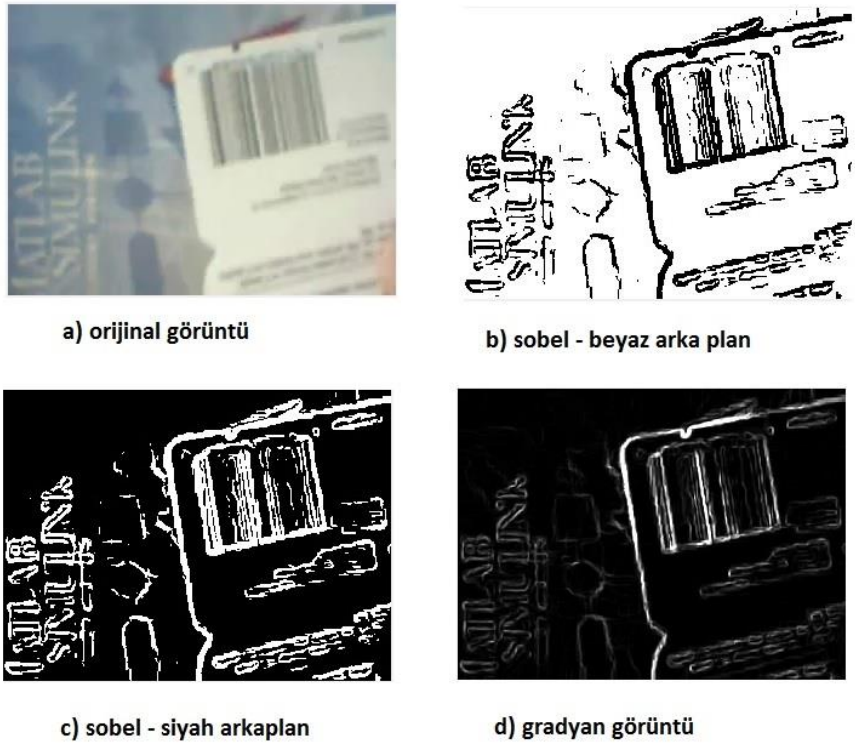
Bu değerlere göre hesaplanan yatay ve dikey gradyan değerleri Formül 3.7.ye göre bir araya getirilerek G bileşeni bulunur.

$$G = \sqrt{Gx^2 + Gy^2} \tag{3.7}$$

Tasarlanan sistemin blok diyagramı Şekil 3.20.de verildiği gibidir. Sistemde ham gradyan çıkışı gözlenebildiği gibi Sobel filtre eşiği de ayarlanabilmekte ve arka plan rengi değiştirilebilmektedir. Sistemin görüntü sonucu ise Resim 3.17.de gösterilmiştir. Simülasyondaki eşik değeri katsayısı 5 olarak seçilmiştir.



Şekil 3.20 Kenar Bulma Filtre Sistemi Blok Diyagramı.



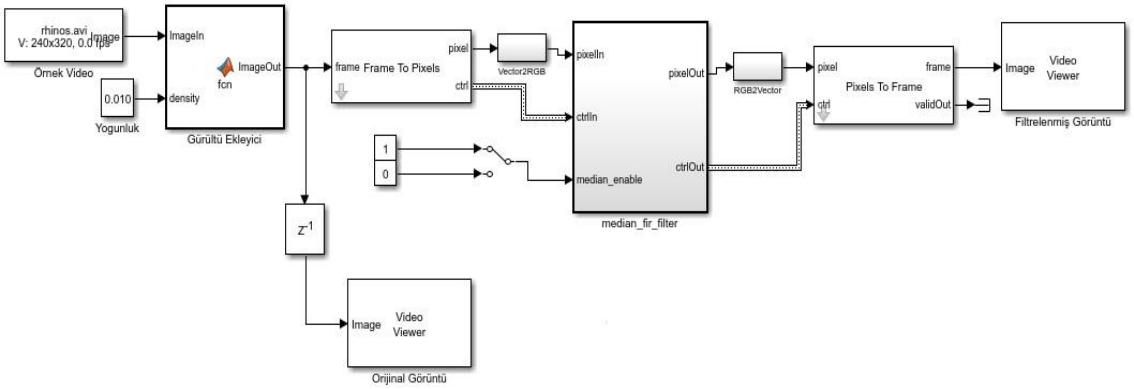
Resim 3.17 Kenar Bulma Filtresi Sonuçları.

3.9.3 Medyan Filtre

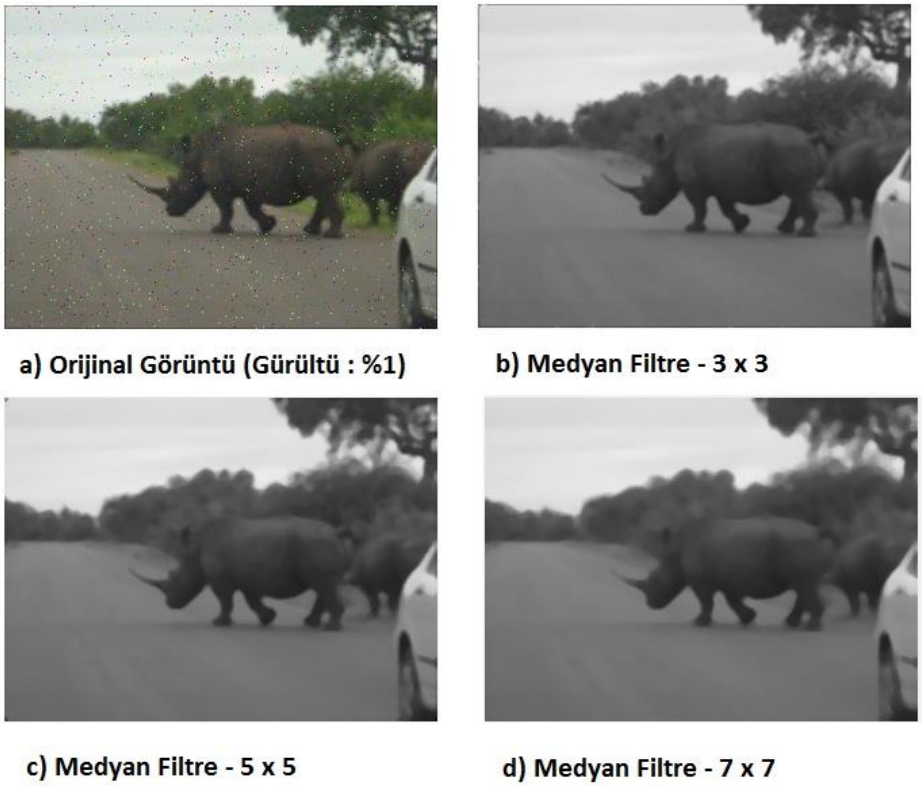
Medyan ya da diğer adıyla ortanca filtre genellikle görüntüdeki tuz-biber gürültüsü adı verilen bozulmaları gidermekte kullanılan doğrusal olmayan bir filtredir. Keskin kenarlardan ziyade ortalama değerlere daha hassas olması sebebiyle görüntüde belli ölçüde bulanıklığa sebep olur. Çalışma prensibi ise görüntünün 2 boyutlu bir komşuluk matrisi ölçüsünde piksellere ayrılarak ortanca elemanın her piksel için ayrı ayrı hesaplanarak sıralanmasına dayanır.

Komşuluk matrisi 3 x 3, 5 x 5 veya 7 x 7 boyutunda olabilir. Matrisin büyüklüğü sıralanacak piksek sayısını arttırdığından komşuluk sayısı arttıkça görüntüdeki detay azalacak ve bununla birlikte gerekli işlem gücü ciddi biçimde artacaktır. Tasarlanan sistemde filtrenin etkisinin net görülebilmesi için görüntü üzerine Simulink ortamında yapay olarak %1 oranında tuz-biber gürültüsü eklenmiştir.

Çalışmada Vision HDL aracına ait "Median Filter" bloğu kullanılmıştır. Sistemin blok diyagramı Şekil 3.21.de verilmiştir. Komşuluk matrisinin farklı değerleri için sistemin görüntü çıktısı Resim 3.18.de gösterilmiştir.



Şekil 3.21 Medyan Filtre Sistemi Blok Diyagramı.



Resim 3.18 Medyan Filtresi Sonuçları.

3.9.4 Keskinleştirici Filtre

Son olarak görüntü üzerine keskinleştirme filtresi uygulanmıştır. Bu tarz filtreler için değişik yöntemler olmakla birlikte bu tez çalışmasında Vision HDL aracı içerisindeki "Image Filter" bloğu kullanılmıştır. Bu blok basitçe katsayıları değiştirilebilen iki boyutlu sonlu dürtü cevabı tipi (FIR) filtre elemanıdır. Bu sayede görüntü üzerinde çeşitli filtreleme işlemleri yapmak mümkün olmaktadır.

Sistemde keskinleştirme için uzamsal bir filtre çeşidi olan laplasyen filtre kullanılmıştır. Bu yöntem basitçe 2. dereceden türev alınmasına dayanan doğrusal bir filtredir. Böylece görüntü üzerindeki kenarlar ve geçişler belirginleşir.

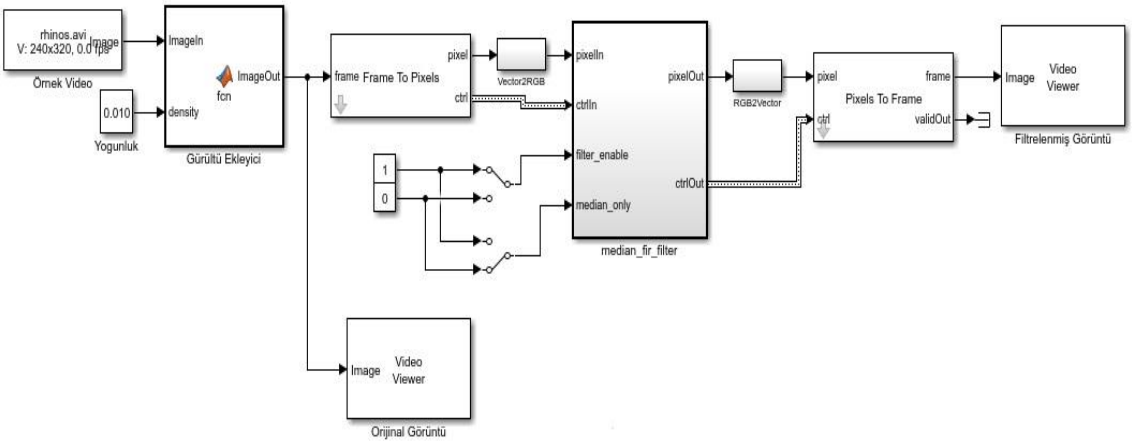
Laplasyen filtrenin genel formülü Denklem 3.8.de 3 x 3 matriste uygulanan katsayılar denklem 3.9.da ifade edilmiştir (Mathworks 2018).

$$\nabla 2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{3.8}$$

$$y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.9}$$

Yapılan tasarımda medyan filtre ve keskinleştirici filtre beraber kullanılmıştır. Bunun sebebi keskinleştirici filtrenin etkisinin daha net ortaya çıkartılmak istenmesidir. Yapılan tasarımın blok diyagramı Şekil 3.22.de verilmiştir.

3 x 3 ve 5 x 5 Medyan filtresi uygulanmış görüntünün keskinleştirme sonuçları ise Resim 3.19.da verilmiştir.



Şekil 3.22 Keskinleştirici Filtre Sistemi Blok Diyagramı.

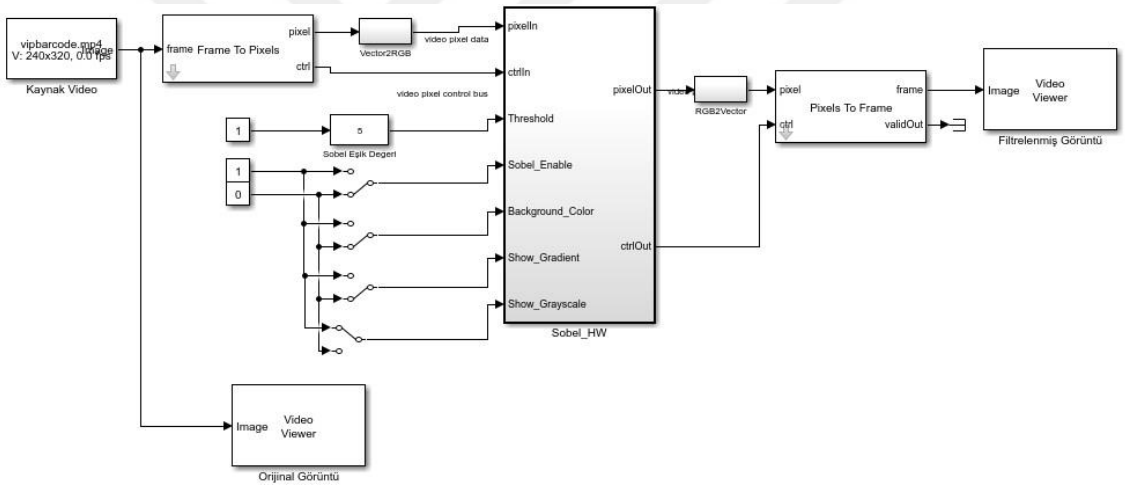
Resim 3.19 Keskinleştirici Filtre Sonuçları.

4. BULGULAR

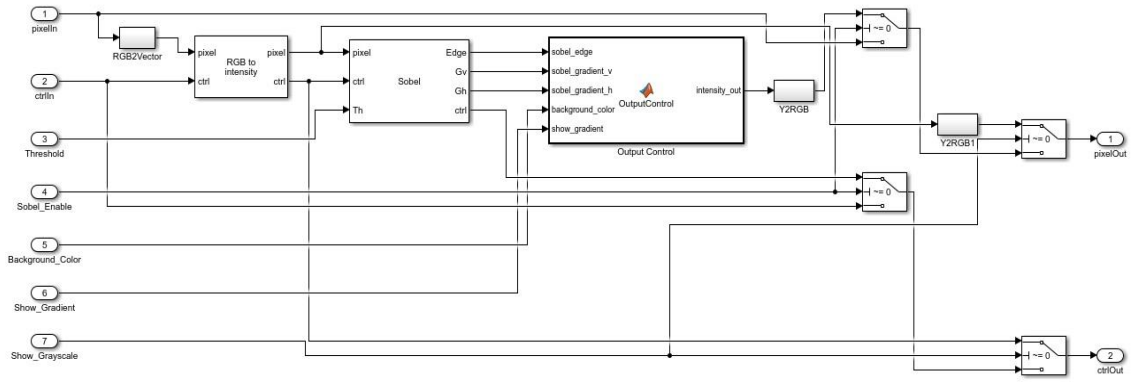
4.1 Tasarımın Sentezlenmesi

Görüntü işleyici filtre sistemlerinin tasarımı yapıldıktan sonra tüm tasarımlar Bölüm 3.9.1.deki gibi Matlab İş Akışı danışmanı vasıtasıyla IP paketi olarak sentezlenmiştir. Bu aşamada Gri ton dönüştürme işlemi kenar bulma algoritması içerisinde hali hazırda var olması sebebiyle iki sistem birleştirilmiş ve tek bir IP bloğu haline getirilmiştir.

Böylece sistemde toplam iki adet IP paketi olacak ve sistem sadeleşmekle birlikte aynı anda tüm filtreler aktif olabilecektir. Tasarlanan ikinci sistemin genel blok diyagramı Şekil 4.1.de, filtre alt sistemi blok diyagramı Şekil 4.2.de gösterilmiştir.



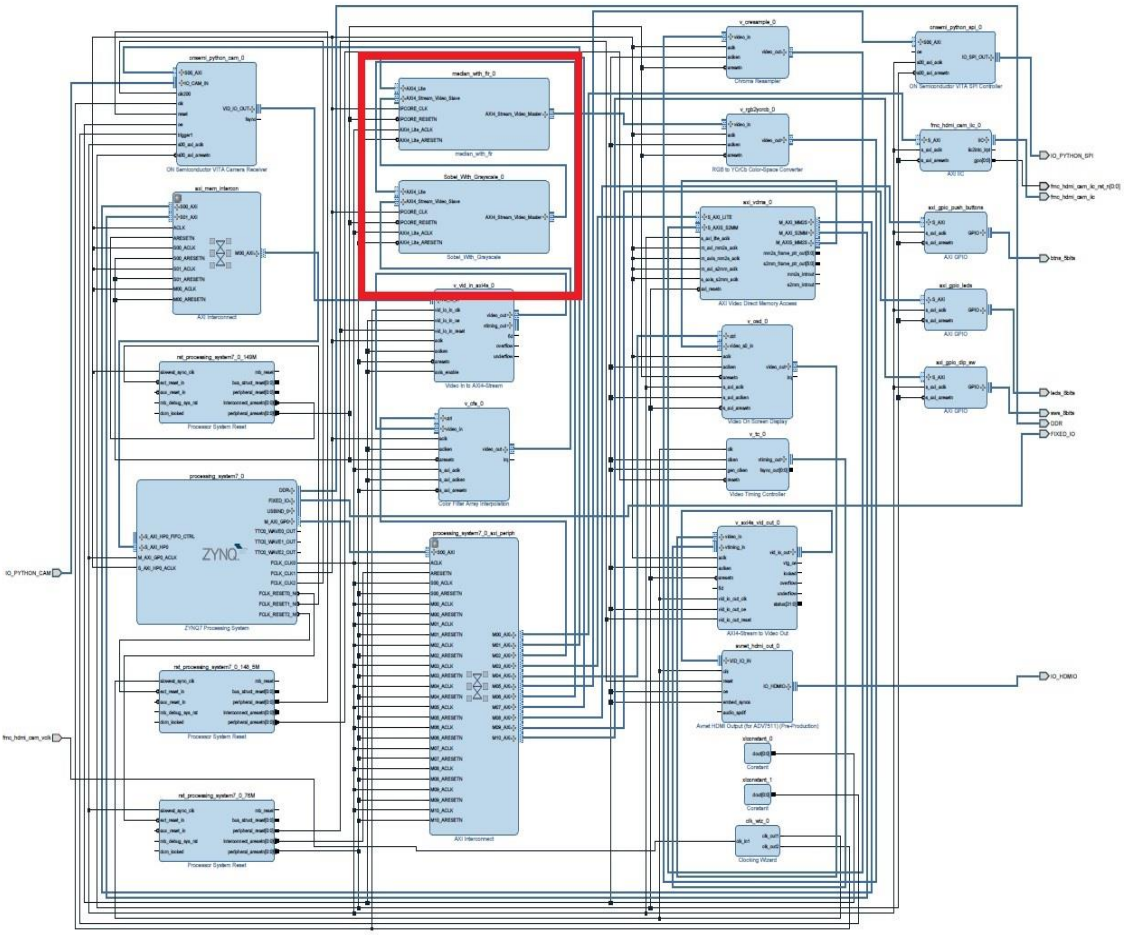
Şekil 4.1 Gri Ton ve Kenar Bulma Filtre Sistemi Blok Diyagramı.



Şekil 4.2 Gri Ton ve Kenar Bulma Filtresi IP Blok Diyagramı.

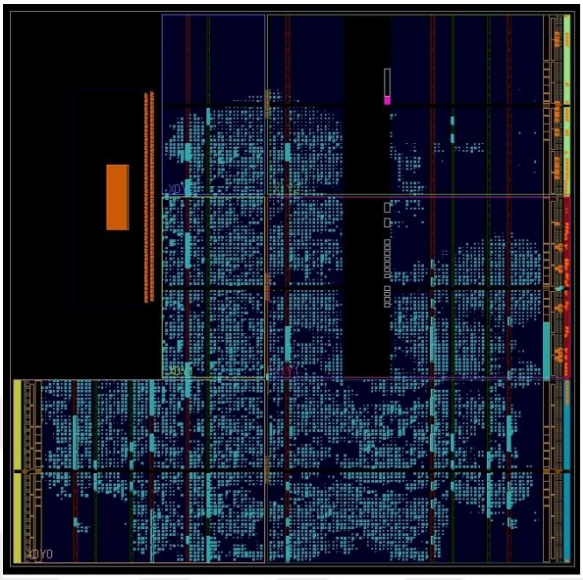
Matlab programında tamamlanan IP paketleri Vivado 2016.4 programına IP entegre edici yardımıyla eklenmiş ve Bölüm 3.8.1.deki kamera referans tasarımına entegre edilmiştir. Ayrıca sistemin bağımsız olarak kontrol edilmesi için Zedboard üzerindeki butonlar ve ledler ile filtrelerin kontrol edilebilmesi için gerekli olan donanım blokları da sisteme eklenmiştir.

Son olarak gerekli ara bağlantılar yapılarak sistem tamamlanarak sentezlenmiş ve gerçeklenmiştir. Sistemin son halinin blok tasarımı Şekil 4.3.de verilmiştir. Tasarlanan iki adet IP bloğu işaretlenerek gösterilmiştir.



Şekil 4.3 Tüm Sistemin Donanım Tasarımı Blok Diyagramı.

Blok tasarım tamamlandıktan sonra vivado programında doğrulanmış ve ardından tasarımdan donanımın sentezi yapılmıştır. Gerçeklenen donanımın tüm yonga üzerindeki yerleşimi Resim 4.1.de gösterildiği gibidir.



Resim 4.1 Gerçeklenmiş Donanımın Yonga Üzerindeki Yerleşimi.

4.2 Sentez Sonuçları

Sistemin kaynak kullanım tablosu Çizelge 4.1.de gösterilmiştir. Verilerden de anlaşılacağı üzere kullanılan çok sayıda IP bloğuna ve fazla sayıda filtrenin bir arada gerçeklenmesine rağmen sistem kaynaklarının en fazla %35'i kullanılmıştır. VDMA kullanımı sayesinde sistem doğrudan bellek erişimli olarak çalışmış ve Zynq üzerinde bulunan dahili hafıza bloklarının yalnızca %25'ini kullanarak sistem gerçeklenebilmiştir. Burada herhangi bir ekstra optimizasyon yapılmadığı da göz önünde bulundurulmalıdır.

Çizelge 4.1 Sistem Kaynak Kullanımı.

Birim	**Mevcut**	**Kullanılan**	**Kullanım Oranı (%)**
Toplam LUT	53200	18646	35,05
Mantıksal LUT	53200	17446	32,79
Hafıza LUT	17400	1200	6,90
Toplam Flip-Flop	106400	24336	22,87
Blok Hafıza	140	35	25,00
Blok DSP	220	29	13,18

Çizelge 4.2 Sistemin Saat Darbesi Kullanımı.

Saat Birimi	Periyot (ns)	Frekans (Mhz)
Clk_fpga_0	12,999	76,929
Clk_fpga_1	7,000	142,857
Clk_fpga_2	5,000	200,000
Video_clk	6,730	148,588
Clock_out_2	9,254	108,064

Sistemin çalışma hızı hesaplanırken saat darbesi kullanımı incelenmelidir. Sistemde farklı birimler için farklı saat darbeleri kullanılmaktadır. Bunlardan hesaplamada kullanılacak olan Clk_out_2 isimli saat darbesidir, kameradan alınan görüntünün işlenmesinde yaklaşık 108 Mhz'lik bu saat darbesi kullanılmaktadır. Kamera sisteminin yaklaşık çalışma hızı aşağıdaki gibi bulunabilir (İnt. Kyn.14).

$$Ekran\ Yenileme\ Hızı\ = \frac{Saat\ Frekansı\ (Hz)}{(Toplam\ Yatay\ Piksel + Toplam\ Dikey\ Piksel)} \quad (4.1)$$

Toplam yatay ve dikey piksellerin sayısı ise Bölüm 3.8.5.5 ve Bölüm 3.9.da anlatıldığı şekilde hesaplanarak Formül 4.2. ve Formül 4.3.de gösterilmiştir.

Burada h_active yatay aktif piksel sayısını, h_fporch yatay ön boşluğu, h_bporch yatay arka boşluğu, h_sync ise yatay senkronizasyon sinyalini ifade eder. Dikey sinyallerde ise aynı şekilde v_active dikey aktif piksel sayısını, v_fporch dikey ön boşluğu, v_bporch dikey arka boşluğu, v_sync ise dikey senkronizasyon sinyalini ifade eder.

$$Toplam\ Yatay\ Piksel\ = h\_active\ +\ h\_fporch\ +\ h\_bporch\ +\ h\_sync \quad (4.2)$$

$$Toplam\ Dikey\ Piksel\ = v\_active\ +\ v\_fporch\ +\ v\_bporch\ +\ v\_sync \quad (4.3)$$

Elde edilen verilere göre sistemin çalışma hızı Formül 4.4.te hesaplandığı gibidir.

$$Ekran\ Yenileme\ Hızı = \frac{108\ 064\ 000}{[(1280+29+192+85)\ x\ (1024+0+3+35)]} = 64{,}15 \quad (4.4)$$

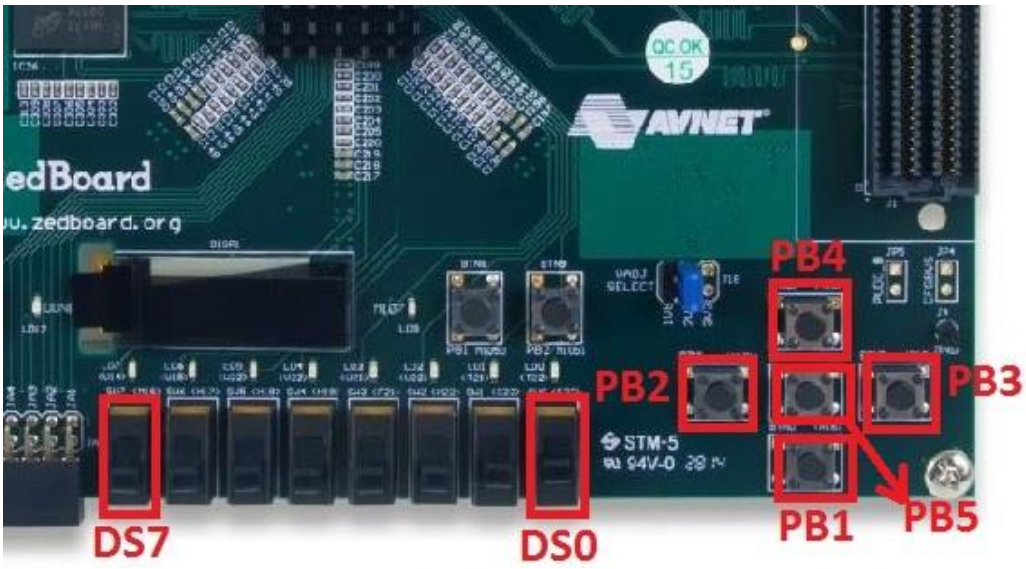
Görüldüğü üzere sistem teorik olarak saniyede yaklaşık 64 kare/saniye hızında çalışabilecek yapıdadır. Bu noktada kameradan alınan görüntü gecikme olmaksızın 60 Hz yenileme hızında monitöre aktarılarak gerçek zamanlı çalışma şartının sağlandığı söylenebilir.

4.3 Sistem Yazılımının Gerçeklenmesi

Sistemin donanım sentezlenmesinin tamamlanmasından sonra sistemin yazılımı hazırlanmıştır. Bunun için Vivado programında File menüsünden "Launch SDK" tıklanarak gerekli sürücü dosyaları otomatik olarak sisteme eklenerek ve proje yazılım geliştirme ortamına aktarılarak ilgili yazılım tasarımı yapılmıştır.

Sistemin donanımı birçok IP paketinin bir arada kullanılması ile oluşturulduğundan yazılımda IP paketlerinin sürücüleri ve genel kütüphaneler kullanılarak tasarım yapılmıştır. Ana programda ilk olarak sistemin kontrolünde kullanılacak olan Zedboard üzerindeki genel amaçlı giriş ve çıkışlar (GPIO) olan led ve butonların kullanılması için gerekli ayarlamalar yapılmıştır. Daha sonra harici olarak bağlanan FMC-HDMI-CAM modülünün ve sırasıyla kamera ve IP bloklarının ayarlamaları yapılarak başlatılmaları gerçekleştirilmiştir. Sistemin genel akış şeması Şekil 4.4.te gösterilmiştir.

Şekil 4.4 Sistem Yazılımı Akış Şeması.



Resim 4.2 Zedboard Üzerindeki Buton ve Led Kontrolleri.

Ana program döngüsünde filtrelerin seçimi ve değiştirilmesi sistemin çalışma anında gerçek zamanlı olarak gerçekleştirilebilmektedir. Bunun için Zedboard üzerindeki seçici anahtarlar ve butonlar kullanılmıştır. Butonların değerleri yine Zedboard üzerindeki ledlere yazdırılarak sistemin durumunun görüntülenmesi sağlanmıştır.

Sistemde kullanılan kontroller Resim 4.2.deki gibi adlandırılmıştır. Seçici anahtarlar DS0'dan DS7'ye doğru sağdan sola doğru numaralandırılmıştır ve ikili sayı sistemi karşılığı olarak seçim yapılmıştır. Seçici anahtar değeri hesaplanırken DS0 en düşük değerlikli bit, DS7 ise en yüksek değerlikli bit olarak kabul edilmiştir.

Görüntü üzerinde uygulanacak filtreler Çizelge 4.3 ve Çizelge 4.4.deki gibi butonların ve anahtarların değerlerine göre değiştirilebilir biçimde tasarlanmıştır. Böylece olası tüm kombinasyonların ekranda görüntülenmesi amaçlanmıştır. Ayrıca yapılan tasarım ile görüntü işleyici sisteminde kullanılan filtrelerin etkisi ayrı ayrı ve bir arada görüntülenebilecek ve Bölüm 3.1.de bahsedilen orta ve yüksek seviyeli görüntü işleme işlemleri için görüntünün hazırlanması ve temizlenmesi amacıyla hangi filtrelerin seçileceği hakkında fikir sahibi olunabileceği öngörülmektedir.

Çizelge 4.3 Görüntü İşleyici Sistemi Seçici Anahtar Durumu.

Seçici Anahtar Değeri (DS0..DS7)	Uygulanan Filtre
0	Filtre Yok
1	Yalnızca Sobel Filtre
2	Gri Ton Görüntü
4	Yalnızca Medyan Filtre
5	Sobel + Medyan Filtre
8	Medyan + Keskinleştirici Filtre
13	Sobel + Medyan + Keskinleştirici Filtre

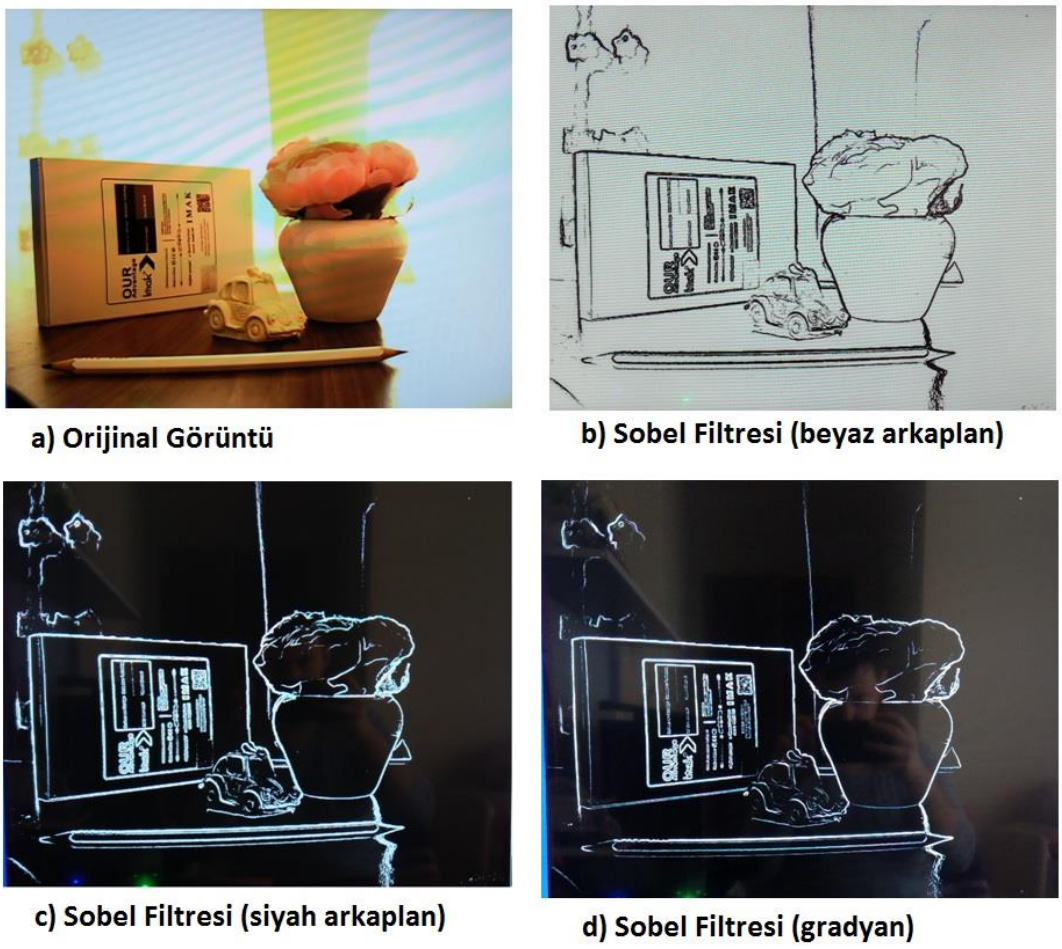
PB2, PB3 ve PB4 ile isimlendirilen butonlar ile Sobel kenar bulma filtresinin parametre kontrolü yapılabilmektedir. Bu sayede kenarları ortaya çıkarılmış olan görüntüdeki renkler terslenebilmekte veya sadece gradyan operatörlerinin sonucu görüntülenebilmektedir. Ayrıca sistemde Bölüm 3.8.5.1.de bahsedilen CFA IP paketi kullanılarak Bayer desenlerinin değiştirilmesi de PB1 butonu ile sağlanmıştır. Çizelge 4.4.te butonların durumları ve karşılık gelen parametreler verilmiştir.

Çizelge 4.4 Görüntü İşleyici Sistemi Buton Durumu.

Buton Aktifliği	Uygulanan Filtre
PB1	Bayer Deseni Değiştirici
PB2	Sobel Filtre - Beyaz Arkaplan
PB3	Sobel Filtre - Siyah Arkaplan
PB4	Sobel Filtre - Gradyan
PB5	Sobel Filtre Eşik Değeri Değiştirici

4.4 Sistemin Ekran Görüntüleri

Resim 4.3, Resim 4.4 ve Resim 4.5.te tasarlanan son sistemin ekran görüntüleri görülmektedir.

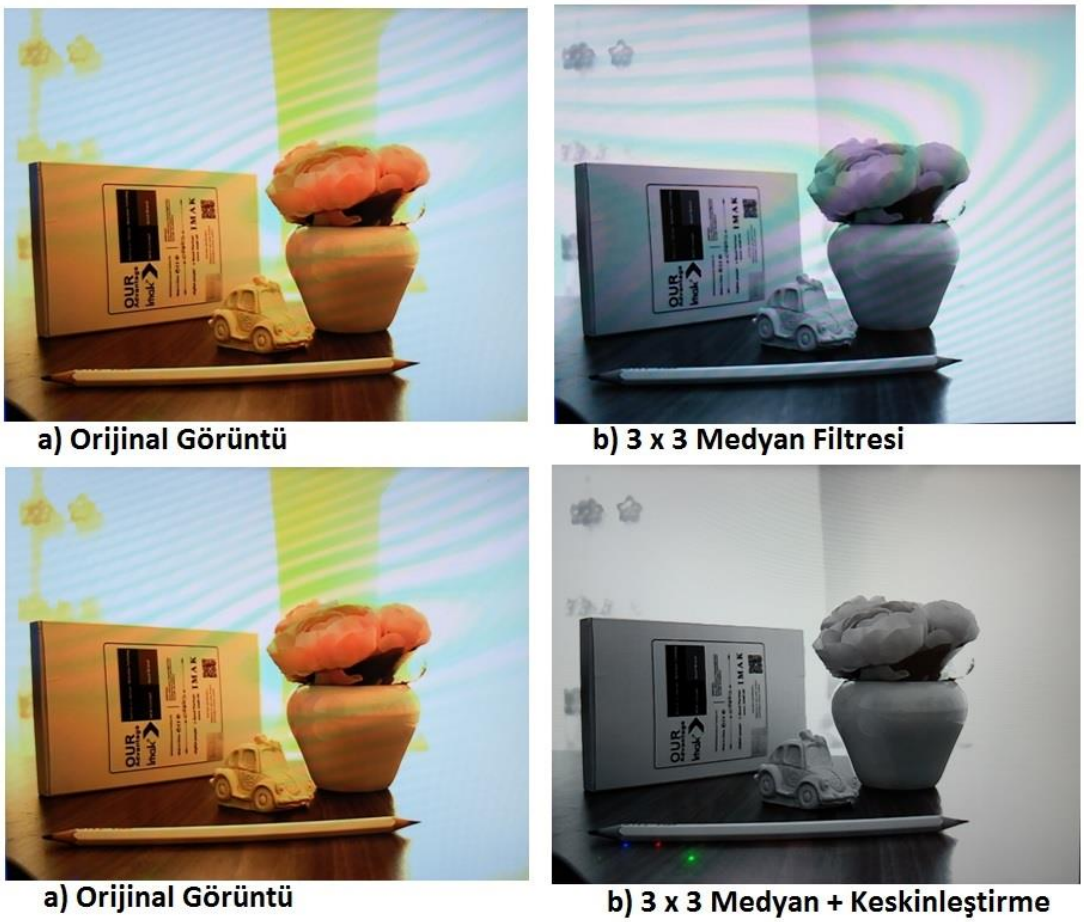


Resim 4.3 Sobel Filtresi Sonucu.

Resim 4.3.te görüldüğü gibi Sobel kenar bulma filtresi görüntüye gerçek zamanlı olarak uygulanmıştır. Orijinal görüntü referans olarak alındığında sonuç görüntülerde nesnelerin kenar hatlarının düzgün biçimde bulunduğu tespit edilmiştir. Ayrıca ilgili çalışmaların sonuçları incelendiğinde filtrelerin benzer biçimde etki gösterdiği görülmektedir (Altuncu *et al.* 2015, Russell and Fischaber 2013, Chhabra *et al.* 2016, Al-Naqshbandi 2016). Buna göre kenar bulma filtresinin verimli biçimde çalıştığı söylenebilir.

Resim 4.4 Gri Ton Filtreleme Sonucu.

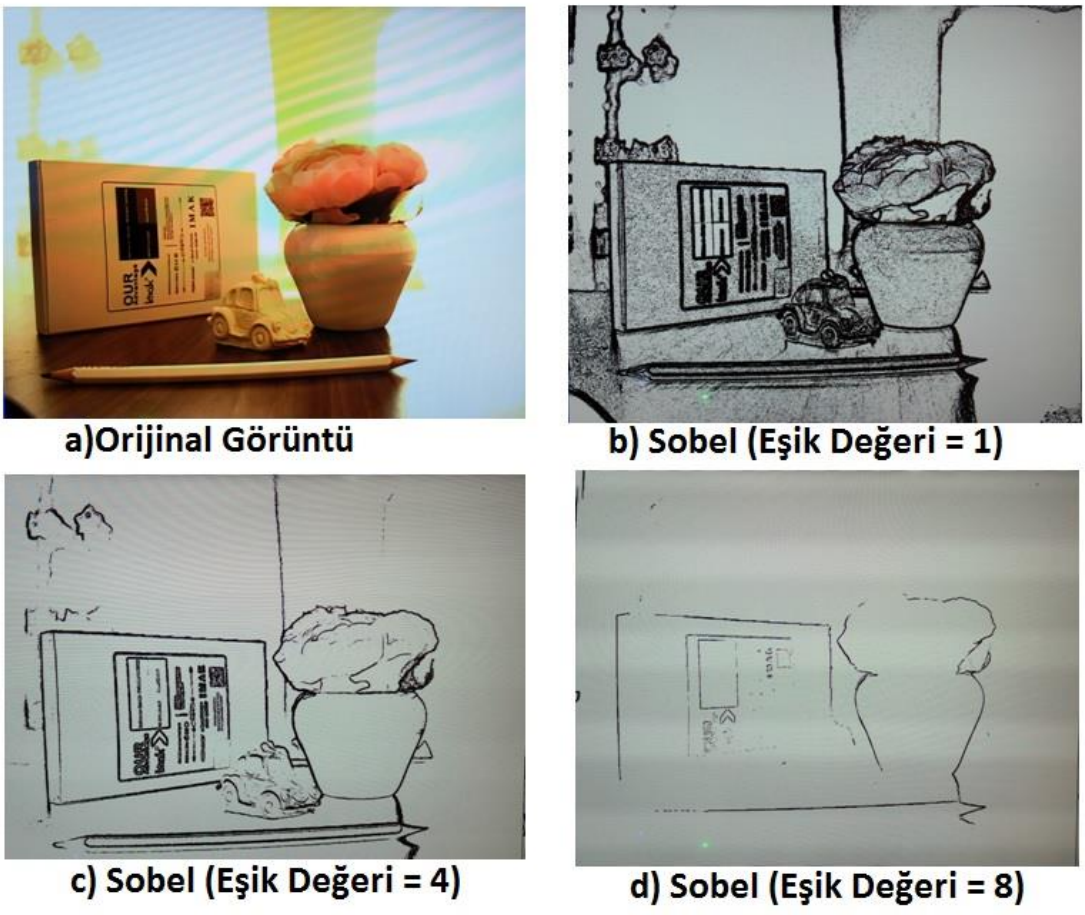
Resim 4.4.te sonuçları görülen gri ton dönüşümü Bölüm 3.9.1.de bulunan Denklem 3.4.e göre yapılmış ve sonuçların Bölüm 3.9.1.deki simülasyon sonuçları ile paralel olduğu tespit edilmiştir.



Resim 4.5 Medyan ve Keskinleştirme Filtreleri Sonucu.

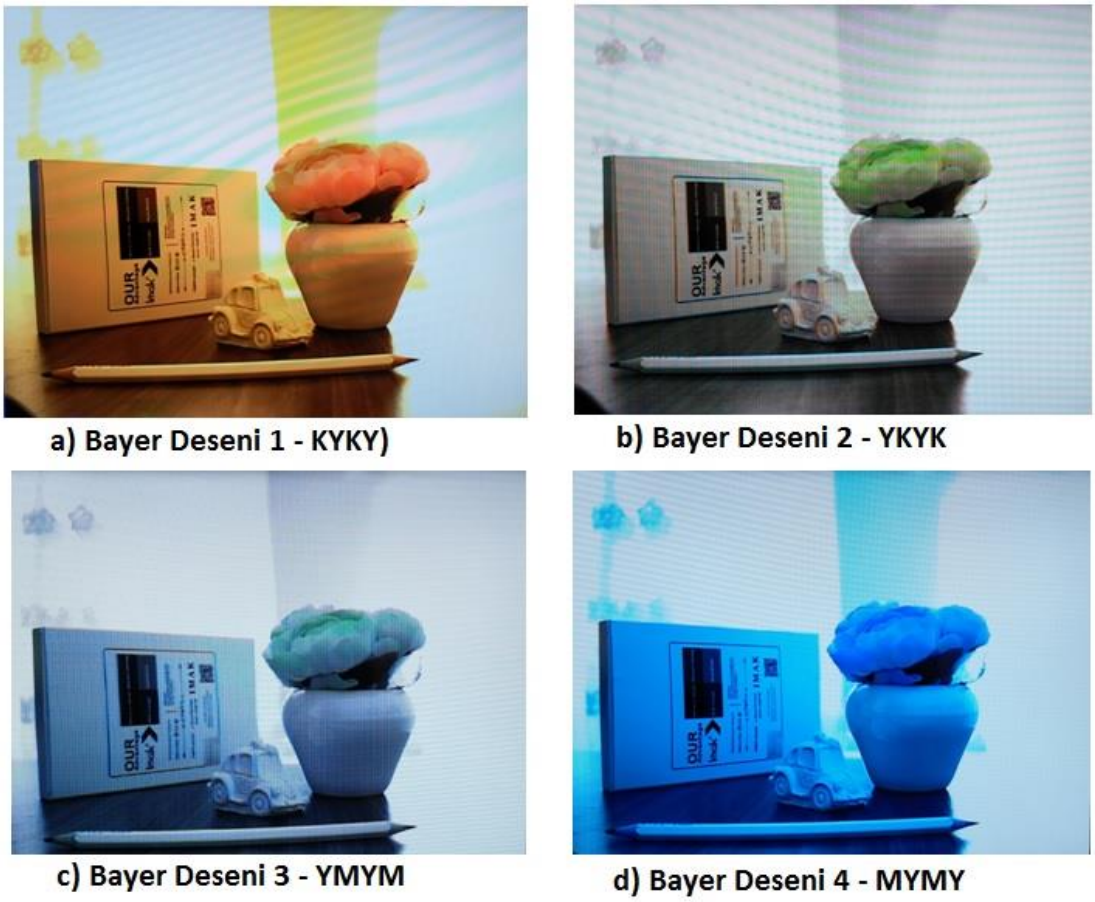
Resim 4.5.te orijinal görüntü üzerinde öncelikle 3 x 3 matris boyutunda ortanca filtre uygulanmıştır. Ortanca filtre gürültü giderme etkisine sahip olduğu gibi görüntüde belli ölçüde detay kaybına neden olabilmektedir. Resimlere dikkatli bakıldığında bu durum görülecektir. Daha sonra ortanca filtre ile yumuşatılan görüntü keskinleştirme işlemine tabi tutulmuştur. Görüldüğü üzere görüntüdeki detaylar tekrar artarak görüntü netlik kazanmıştır. Sonuçların hem Bölüm 3.9.daki simülasyon sonuçlarıyla hem de ilgili çalışmaların sonuçlarıyla benzerlik taşıdığı görülmektedir (Levent 2015, Altuncu *et al.* 2015).

Uygulanan Sobel filtresinin farklı eşik değerleri için sonuçlar Resim 4.6.da gösterilmiştir. Bayer desenlerinin farklı değerleri için sonuçlar Resim 4.7.de gösterilmiştir.



Resim 4.6 Farklı Eşik Değerleri İçin Sobel Filtresi Sonucu.

Resim 4.6.da görüldüğü üzere Sobel kenar bulma filtresi uygulanan görüntülerdeki detaylar eşik seviyesine göre değişmektedir. Eşik değeri azaldıkça daha düşük değerdeki gri ton pikseller belirgin olmakta ve böylece görüntüdeki detay seviyesi artmakta, buna bağlı olarak gürültü oranı da yükselmektedir. Aynı şekilde eşik değeri arttıkça sadece daha yüksek değerdeki pikseller aktif olmakta ve böylece görüntüdeki detay seviyesi ve buna bağlı olarak da gürültü seviyesi azalmaktadır.



Resim 4.7 Farklı Bayer Deseni Değerleri İçin Ekran Görüntüsü.

Son olarak Resim 4.7.de gerçeklenen Bayer deseni değiştiricinin sonuçları görülmektedir. Sistem Bölüm 3.8.5.1.de bahsedilen CFA IP bloğu üzerinden değiştirilerek kontrol edilmiştir. Sonuçlar incelendiğinde filtre kombinasyonlarına göre sahnedeki renklerin değiştiği gözlenmiştir. Bu sistemin renk filtreleme veya nesnelerin renklerine göre sınıflandırılması için altyapı olarak da kullanılabileceği düşünülmektedir.

5. TARTIŞMA ve SONUÇ

Günümüzde görüntü işleme sistemlerinin birçok alanda yoğun olarak kullanılması ve bu konuda gerçek zamanlı sistemlerde ihtiyaç duyulan hız ve yoğun kaynak ihtiyacı sebebiyle FPGA ve işlemciyi bir arada bulunduran sistemlerin bir arada kullanılması yaygın bir metot haline gelmiştir. Bu prensipten hareketle yapılan tez çalışmasında kullanmak üzere Zynq-7000 yonga üzeri sistem platformu seçilmiştir. Bu tümleşik sistem ile hem donanım hem de yazılım tasarımı tek bir kart üzerindeki sistemde gerçeklenmiş ve sistemin gerçek zamanlı çalışma ihtiyacına cevap verebildiği gözlemlenmiştir.

FPGA içeren sistemlerin en büyük dezavantajı olarak değerlendirilen tasarım zorluğu ve süresini azaltmak için model tabanlı tasarım yöntemi önerilmiş ve görüntü işleyici filtre tasarımları Mathworks şirketinin Matlab ve Simulink ürünleri ile HDL Coder, Vision HDL Toolbox ve Embedded Coder gibi yüksek seviyeli sentez araçları kullanılarak elle HDL kodu yazılmadan gerçeklenmiştir.

Yapılan tez çalışmasında FPGA yapıları ve kullanılan mimari ile ilgili hem yazılım hem de donanım mimarisi hakkında detaylı açıklamalar yapılmış ve ayrıca görüntü işleyici sistemleri ve bunların bileşenleri hakkında detaylı bilgiler verilmiştir. Sistemin önce donanım tasarımı yapılmış, ardından yazılım tasarımı yapılarak kamera referans sistemi gerçeklenmiştir. Referans sistemin başarılı bir biçimde çalışmasının ardından Matlab ve Simulink programları üzerinde kenar bulma, gri ton dönüştürme, ortanca ve keskinleştirme filtrelerinin tasarımları ve simülasyonları sırasıyla yapılmış, sistemin kararlı çalıştığı görüldükten sonra bunlar alt sistem olarak tasarlanıp HDL Coder aracı ile Vivado ortamında Zynq tasarımlarına entegre edilebilir IP paketlerine dönüştürülmüştür. Daha sonra sistem üzerinde sadeleştirme çalışmaları için filtreler toplamda iki adet IP paketi oluşturulacak şekilde bir araya getirilerek tekrar tasarlanmışlardır.

Çalışmada kullanılan Zedboard geliştirme kartı üzerinde 1280 x 1024 piksel çözünürlükte 60 FPS hızında çalışabilen bir sistem geliştirilmiştir. Sistemde uygulanan Sobel kenar bulma, ortanca filtre, gri ton dönüştürücü ve keskinleştirici filtre tasarımları kart üzerindeki giriş ve çıkış birimlerinden çalışma anında kontrol edilerek parametreleri değiştirilebilen esnek bir sistem ortaya konmuştur.

Yapılan tasarımlar neticesinde sistemin ihtiyaçlara cevap verebildiği, gerçek zamanlı çalışma şartını yerine getirebildiği görülmüştür. Bununla birlikte kaynak kullanımının fazla yüksek olmadığı, ancak HDL Coder aracında ilgili en iyileştirmeler yapıldığı takdirde kullanımın daha da düşebileceği düşünülmektedir. Sistemle ilgili olabilecek en önemli zayıf nokta ise mimarinin ve tasarım metotlarının farklılığıdır. Başta Zynq mimarisi olmak üzere Vivado, Matlab ve Simulink araçlarıyla ilgili bilgi sahibi olunması gerektiği düşünüldüğü gibi ayrıca kullanılan araçların iyi tanınması ve gerekli dokümanların incelenmesi bu noktada çok önemlidir.

Sistem tasarımı tamamen yeniden kullanılabilir olarak yapılmıştır. Bu sayede daha ileri çalışmalarda ve farklı sistemlerde tasarlanan IP bloklarının ya da sistemin tamamının tekrar kullanılması mümkün olabilecektir. Gelecekteki çalışmalarda bu görüntü işleme sistemini ön işleme birimi olarak kullanarak daha gelişmiş gerçek zamanlı nesne tanıma ve takip uygulamaları yapılması düşünülmektedir.

6. KAYNAKLAR

Al-Naqshbandi S. (2016). Hardware Acceleration of Computer Vision Application. M.Sc. Thesis, Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Netherlands.

Altuncu, M.A., Guven, T., Becerikli, Y., Sahin, S. (2015). Real-Time System Implementation for Image Processing with Hardware/Software Co-design on the Xilinx Zynq Platform. *International Journal of Information and Electronics Engineering*, **5** (**6**) : 473-477.

Avnet, HDMI Input/Output FMC Module with Camera Interface Hardware Guide Rev. 1.0. Avnet Inc, Sept 14, 2015.

Avnet, FMC-HDMI-CAM + PYTHON-1300-C Frame Buffer Design Tutorial Ver. 2015.2. Avnet Inc, Dec 9, 2015.

Avnet, FMC-HDMI-CAM + PYTHON-1300-C Vivado HLS Reference Design Ver. 2015.4. Avnet Inc, May 19, 2016.

Avnet, ON Semiconductor PYTHON 1300-C CAMERA MODULE Hardware User Guide Ver. 1.0 Avnet Inc, Jan. 20, 2015.

Avnet, ON Semiconductor PYTHON 1300-C CAMERA MODULE Product Brief Avnet Inc, 2017.

Avnet, ZedBoard Configuration and Booting Guide Ver. 14.1 Avnet Inc, Aug. 17, 2012.

Canpolat, M., Durak, U. (2012). Model Tabanlı Sistem Simülasyon Yazılımı Geliştirme. 6. Ulusal Yazılım Mühendisliği Sempozyumu, Ankara.

Chhabra, S., Jain, H., Saini, S. (2016). FPGA based Hardware Implementation of Automatic Vehicle License Plate Detection System. (2016). Intl. Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, Sept. 21-24, 2016, 1181-1187.

Crockett, L.H., Elliot, R.H., Enderwitz, M.A., Stewart, R.W. (2014). The Zynq Book : Embedded Processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC. Department of Electronic and Electrical Engineering University of Strathclyde, 1st Edition, Glasgow, Scotland, UK.

Gonzalez, R.C., Woods, R.E. (2002). Digital Image Processing. Prentice Hall, 2nd Edition, Upper Saddle River, New Jersey, USA.

Kahraman, E., Ünal, V. (2007). Gerçek Zamanlı Gömülü Sistem ve Yazılım Tasarımı'nda ASELSAN Yaklasımı. III.Ulusal Yazılım Mühendisliği Sempozyumu UYMS, EMO Ankara.

Lee, I. (2009). Real -Time Systems CIS 480, Spring 2009 : 3-4.

Levent V.E. (2015). Video İşleme İçin FPGA Tabanlı Bir Donanım Platformu. Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

Mathworks, Image Processing Toolbox Reference version 10.2 R2018a. Mathworks Inc, March 2018.

Mathworks, HDL Coder Getting Started Guide version 3.11 R2017b. Mathworks Inc, Sept. 2017.

Mathworks, Vision HDL Toolbox User's Guide version 1.5 R2017b. Mathworks Inc, Sept. 2017.

Mathworks, Vision HDL Toolbox Reference version 1.5 R2017b. Mathworks Inc, Sept. 2017.

Monson, J., Wirthlin, M., Hutchings, B.L. (2013). Implementing High-Performance, Low-Power FPGA-based Optical Flow Accelerators in C. NSF Center for High-Performance Reconfigurable Computing (CHREC).

Russell, M., Fischaber, S., (2013). OpenCV based road sign recognition on Zynq. 2013 IEEE 11th International Conference on Industrial Informatics. (INDIN)

Sarıtaş, E., Karataş, S. (2013). Her yönüyle FPGA ve VHDL. Palme Yayıncılık.

Shi Z. (2016). Rapid Prototyping of an FPGA-Based Video Processing System. M.Sc. Thesis, Virginia Tech- Virginia Polytechnic Institute and State University, Computer Engineering, Blacksburg, Virginia.

Smith, P.F., Prabhu, S.M., Friedman, J.H. (2007). Best Practices for Establishing a Model-Based Design Culture. SAE World Congress & Exhibition, 10.4271/2007-01-0777.

Sümer Ö. (2014). An Embedded Design And Implementation Of A Facial Expression Recognition System. Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

Xilinx, AXI Reference Guide. UG761 (v13.4). Xilinx Inc, Jan. 18, 2012.

Xilinx, AXI Video Direct Memory Access v6.2 LogiCORE IP Product Guide. PG020. Xilinx Inc, Nov 30, 2016.

Xilinx, Chroma Resampler v4.0 LogiCORE IP Product Guide. PG012. Xilinx Inc, Nov 18, 2015.

Xilinx, Color Filter Array Interpolation v7.0 LogiCORE IP Product Guide. PG002. Xilinx Inc, Nov 18, 2015.

Xilinx, Video On-Screen Display v6.0 LogiCORE IP Product Guide. PG010. Xilinx Inc, Nov 18, 2015.

Xilinx, RGB to YCrCb Color-Space Converter v7.1 LogiCORE IP Product Guide. PG013. Xilinx Inc, Nov 18, 2015.

Xilinx, Video Timing Controller v6.1 LogiCORE IP Product Guide. PG016. Xilinx Inc, Oct 4, 2017.

Xilinx, Zynq-7000 All Programmable SoC Data Sheet: Overview. DS190 (v1.11). Xilinx Inc, June 7, 2017.

Xilinx, Zynq-7000 All Programmable SoC Software Developers Guide. UG821 (v12.0). Xilinx Inc, Sept. 30, 2015.

Xilinx, Vivado Design Suite User Guide: High-Level Synthesis. UG902 (v2014.1). Xilinx Inc, May 30, 2014.

Xilinx, Vivado Design Suite User Guide: Design Flows Overview. UG892 (v2017.4). Xilinx Inc, Dec 20, 2017.

Xilinx, Vivado Design Suite User Guide: Getting Started. UG910 (v2017.4). Xilinx Inc, Dec 20, 2017.

Xilinx, UltraFast Design Methodology Guide for the Vivado Design Suite UG949 (v2015.3). Xilinx Inc, Nov 23, 2015.

Xilinx, Vivado Design Suite AXI Reference Guide UG1037 (v4.0). Xilinx Inc, July 15, 2017.

İnternet Kaynakları

1) http://encyclopedia2.thefreedictionary.com/Real-time+systems, 25.03.2018
2) http://www.fpganedir.com/FPGA/index.php, 25.03.2018
3) https://www.xilinx.com/products/design-tools/vivado/integration/sysgen.html, 25.03.2018
4) http://zedboard.org/product/zedboard, 25.03.2018
5) https://www.xilinx.com/products/design-tools/vivado.html#overview, 25.03.2018
6) https://www.mathworks.com/discovery/what-is-matlab.html, 25.03.2018
7) https://www.xilinx.com/products/intellectual-property/ef-di-vid-img-ip-pack.html, 25.03.2018
8) https://github.com/Avnet/hdl/tree/master/IP, 30.03.2018
9) https://www.mathworks.com/help/supportpkg/xilinxzynq7000/ug/model-design-for-axi4-stream-video-interface-generation.html, 25.03.2018
10) https://www.mathworks.com/help/hdlcoder/examples/getting-started-with-axi4-stream-video-interface-in-zynq-workflow.html#hdlcoder_product-hdlcoder_ip_core_axi4_video, 25.03.2018
11) https://www.mathworks.com/help/visionhdl/gs/design-video-processing-algorithms-for-hdl-in-simulink.html, 25.03.2018
12) https://www.mathworks.com/help/hdlcoder/examples/getting-started-with-hardware-software-codesign-workflow-for-xilinx-zynq-platform.html#hdlcoder_product-hdlcoder_ip_core_tutorial_zynq, 25.03.2018
13) https://www.fujitsu.com/downloads/MICRO/fme/displaycontrollers/an-mb86r01-display-timing-calc-rev1-20.pdf, 30.03.2018
14) http://www.epanorama.net/faq/vga2rgb/calc.html, 30.03.2018
15) http://www.fpganedir.com/FPGA/fpga_yapisi.php, 03.04.2018
16) http://www.fpganedir.com/VHDL/vhdl_terminoloji.php, 03.04.2018

ÖZGEÇMİŞ

Adı Soyadı : Mustafa Yusuf DEMİRCİ

Doğum Yeri ve Tarihi : İzmit - 08.10.1988

Yabancı Dili : İngilizce

İletişim (Telefon/e-posta) : 05555643580 / mfecidpc@gmail.com

Eğitim Durumu (Kurum ve Yıl)

Lise : Konya Meram Atatürk Anadolu Meslek Lisesi, Bilgisayar Bölümü, (2002-2006)

Lisans : Mersin Üniversitesi, Elektronik ve Bilgisayar Eğitimi A.B.D, Kontrol Öğretmenliği, (2006-2010)

Yüksek Lisans : Afyon Kocatepe Üniversitesi, Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği Anabilim Dalı, (2016-2018)

Çalıştığı Kurumlar ve Yıl : Milli Eğitim Bakanlığı - Teknik Öğretmen, Orhan Abalıoğlu M.T.A.L, Endüstriyel Otomasyon Teknolojileri Alanı, Pamukkale/DENİZLİ, (2012-2017)

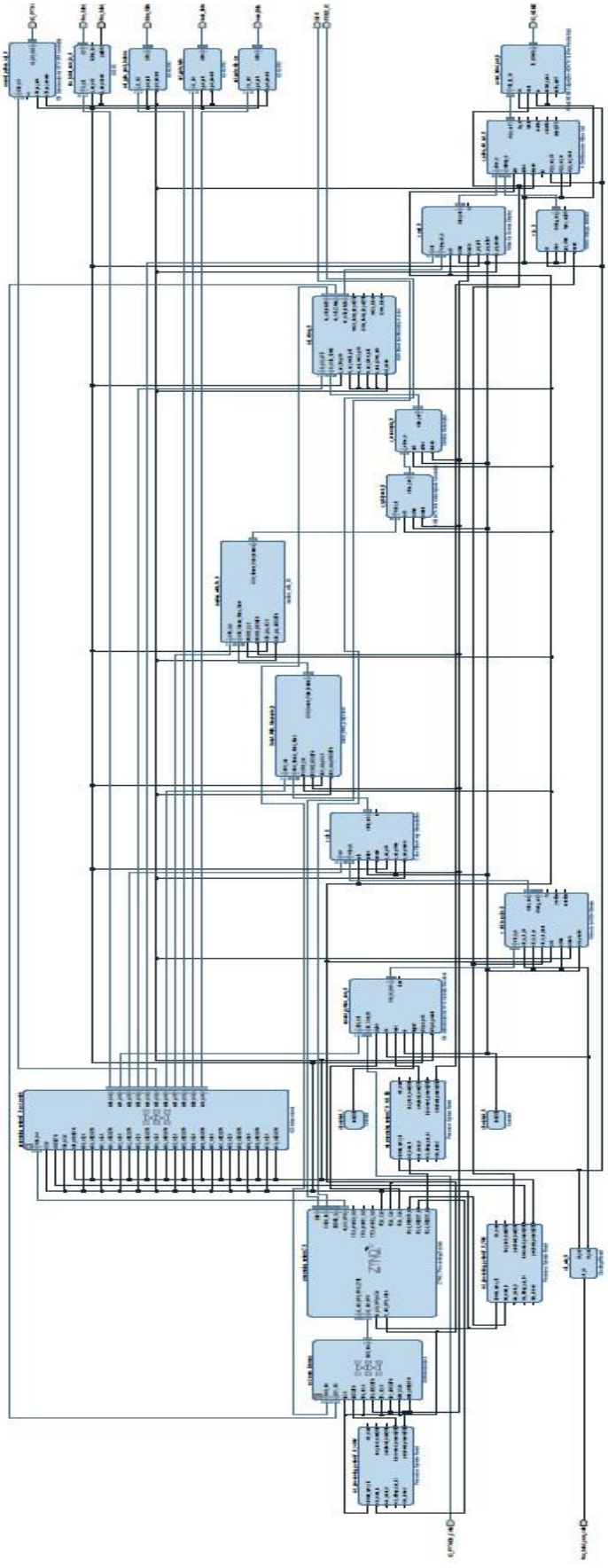
Milli Eğitim Bakanlığı - Teknik Öğretmen Fehim Adak M.T.A.L, Endüstriyel Otomasyon Teknolojileri Alan Şefi, Artuklu/MARDİN, (2017-Halen)

EKLER

EK 1. Görüntü İşleyici Sistemi Donanım Blok Şeması

EK 2. Sistemin Ana Program Kodu

EK 1. Görüntü İşleyici Sistemi Donanım Blok Şeması



EK 2. Sistemin Ana Program Kodu

```
#include "demo.h"
#include "console.h"
#include "xparameters.h"
#include "xgpio.h"
#include "onsemi_python_sw.h"

#define GPIO_LEDS_ID  XPAR_AXI_GPIO_LEDS_DEVICE_ID
#define GPIO_DIPS_ID  XPAR_AXI_GPIO_DIP_SW_DEVICE_ID
#define GPIO_PUSHBUTTONS_ID  XPAR_AXI_GPIO_PUSH_BUTTONS_DEVICE_ID

XGpio led,dip,push;
int sw_value,push_value;
int bayer_value=1;
int sobel_thres=4;

demo_t demo;
demo_t *pdemo;

int main()
{
XGpio_Initialize(&led, GPIO_LEDS_ID);
XGpio_Initialize(&dip, GPIO_DIPS_ID);
XGpio_Initialize(&push, GPIO_PUSHBUTTONS_ID);

XGpio_SetDataDirection(&led, 1, 0x00);
XGpio_SetDataDirection(&dip, 1, 0xFF);
XGpio_SetDataDirection(&push, 1, 0xFF);

        xil_printf("\n\r");
        xil_printf("-----------------------------------------------------------------------\n\r");
        xil_printf("--            AFYON KOCATEPE UNIVERSITESI       --\n\r");
        xil_printf("--            YUKSEK LISANS TEZ CALISMASI       --\n\r");
        xil_printf("--                MUSTAFA DEMIRCI               --\n\r");
        xil_printf("-----------------------------------------------------------------------\n\r");
        xil_printf("\n\r");

        pdemo = &demo;
        demo_init( pdemo );
        demo_init_frame_buffer(pdemo);
        pdemo->cam_alpha = 0xFF;
        pdemo->hdmi_alpha = 0x00;
        demo_start_cam_in(pdemo);
        demo_start_frame_buffer(pdemo);
        print_console_serial_app_header();
        start_console_serial_application();

        while (1)
        {
                sw_value=XGpio_DiscreteRead(&dip, 1);
                push_value=XGpio_DiscreteRead(&push, 1);
                if (transfer_console_serial_data()) {
                        break;
                }
                /* filtre yok */
                if (sw_value==0)
                        {
                        Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x104 , 0);
                        Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x110, 0);
                        Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x100 , 0);
                        Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x104, 0);
```

EK 2. (Devam) Sistemin Ana Program Kodu

```
XGpio_DiscreteWrite(&led,1,sw_value);
            }
/* sadece sobel */
else if (sw_value==1)
{
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x104 , 1);
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x110, 0);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x100 , 0);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x104, 0);
XGpio_DiscreteWrite(&led,1,sw_value);
}
/* sadece gri ton */
else if (sw_value==2)
{
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x104 , 0);
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x110, 1);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x100 , 0);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x104, 0);
XGpio_DiscreteWrite(&led,1,sw_value);
}
/* sadece medyan */
else if (sw_value==4)
{
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x104 , 0);
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x110, 0);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x100 , 1);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x104, 0);
XGpio_DiscreteWrite(&led,1,sw_value);
}

/* medyan + fir */
else if (sw_value==8)
{
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x104 , 0);
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x110, 0);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x100 , 0);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x104, 1);
XGpio_DiscreteWrite(&led,1,sw_value);
}
/* medyan + sobel */
else if(sw_value==5)
{
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x104 , 1);
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x110, 0);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x100 , 1);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x104, 0);
XGpio_DiscreteWrite(&led,1,sw_value);
}
/* sobel + median + fir */
else if(sw_value==13)
{
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x104 , 1);
Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x110, 0);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x100 , 0);
Xil_Out32(XPAR_MEDIAN_WITH_FIR_0_BASEADDR + 0x104, 1);
XGpio_DiscreteWrite(&led,1,sw_value);
}
/* varsayılan */
else
{
```

EK 2. (Devam) Sistemin Ana Program Kodu

```
                        XGpio_DiscreteWrite(&led,1,sw_value);
                        }

                        //sobel eşik değiştirici
                        if (push_value==1)
                        {
                        Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x100,
                        sobel_thres);
                        millisleep(500);
                        sobel_thres++;
                                if(sobel_thres>20)
                                {
                                sobel_thres=1;
                                }
                        }
                        // bayer deseni değiştirici
                        else if (push_value==2)
                        {
                        XCfa_SetBayerPhase(pdemo->pcfa, bayer_value);
                        XGpio_DiscreteWrite(&led,1,push_value);
                        millisleep(500);
                        bayer_value++;
                                if(bayer_value>3)
                                {
                                bayer_value=0;
                                }
                        }
                                        // beyaz arkaplan
                        else if (push_value==4)
                        {
                        Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x108, 0);
                        Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x10C, 0);
                        XGpio_DiscreteWrite(&led,1,push_value);
                        }
                        // siyah arkaplan
                        else if (push_value==8)
                        {
                        Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x108, 1);
                        Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x10C, 0);
                        XGpio_DiscreteWrite(&led,1,push_value);
                        }
                        // sadece gradyan
                        else if (push_value==16)
                        {
                        Xil_Out32(XPAR_SOBEL_WITH_GRAYSCALE_0_BASEADDR + 0x10C, 1);
                        XGpio_DiscreteWrite(&led,1,push_value);
                        }
        }
        return 0;
}
```