

İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

**Bazı Şerit Takip Algoritmalarının SDSoc ve Vivado Platformları Kullanılarak
FPGA Üzerinde Gerçeklenmesi**

LİSANS BİTİRME TASARIM PROJESİ

Yakup GÖRÜR

Mehmet Akif AKKAYA

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

MAYIS, 2018

İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

**Bazı Şerit Takip Algoritmalarının SDSoC ve Vivado Platformları Kullanılarak
FPGA Üzerinde Gerçeklenmesi**

LİSANS BİTİRME TASARIM PROJESİ

Yakup GÖRÜR
(040130052)

Mehmet Akif AKKAYA
(040130094)

Proje Danışmanı: Doç. Dr. Sıddıka Berna ÖRS YALÇIN

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

MAYIS, 2018

İTÜ, Elektronik ve Haberleşme Mühendisliği Bölümü'nün ilgili Bitirme Tasarım Projesi yönergesine uygun olarak tamamen kendi çalışmamız sonucu hazırladığımız “Şerit Takip Algoritmalarının SDSoc ve Vivado Platformları Kullanılarak FPGA’de Gerçeklenmesi” başlıklı Bitirme Tasarım Projesi’ni sunmaktayız. Bu çalışmayı intihal olmaksızın hazırladığımızı taahhüt eder; intihal olması durumunda bitirme tasarım projesinin başarısız sayılacağını kabul ederiz.

Yakup GÖRÜR
(040130052)

.....

Mehmet Akif AKKAYA
(040130094)

.....

Proje Danışmanı : Doç. Dr. Sıddıka Berna ÖRS YALÇIN

.....

ÖNSÖZ

Bitirme projemiz boyunca bize her hafta değerli vaktini ayıran, bizi araştırmaya teşvik eden, karşılaştığımız sıkıntılarda bizden desteğini biran olsun esirgemeyen saygıdeğer danışman hocamız Doç. Dr. Sıddıka Berna ÖRS YALÇIN'a ve bitirme projemiz sürecinde bize yol gösteren Araş. Gör. Yük. Müh. Ercan Kalalı'ya sonsuz teşekkürlerimizi sunmayı bir borç biliriz.

İstanbul Teknik Üniversitesi lisans hayatımızda eğitimimize katkı sağlayan, bakış açımızı genişleten ve bize vizyon katan hocalarımıza ve kıymetli arkadaşlarımıza teşekkür ederiz.

2209-B - Sanayiye Yönelik Lisans Bitirme Tezi Destekleme Programı kapsamında bitime projemizi destekleyen TÜBİTAK'a teşekkür ederiz.

Hayatımız boyunca bizden desteklerini esirgemeyen ailelerimize de sonsuz minnettarlığımızı sunarız.

Mayıs 2018

Yakup GÖRÜR
Mehmet Akif AKKAYA

İÇİNDEKİLER

Sayfa

ÖNSÖZ	vii
İÇİNDEKİLER	ix
KISALTMALAR	xi
ÇİZELGE LİSTESİ	xiii
ŞEKİL LİSTESİ	xv
ÖZET	xvii
SUMMARY	xix
1. GİRİŞ	1
1.1 Projenin Amacı.....	2
1.2 Literatür Araştırması	2
2. ÖNBİLGİLER	5
2.1 Görüntü İşleme Kütüphaneleri	5
2.1.1 OpenCV kütüphanesi	5
2.1.2 Alternatif görüntü işleme kütüphaneleri	5
2.2 Sahada Programlanabilir Kapı Dizileri	6
2.3 Zedboard Geliştirme Kiti	8
2.4 ARM İşlemci	9
2.5 Verilog Donanım Tanımlama Dili	9
2.6 Xilinx Vivado Platformu	10
2.6.1 Proje oluşturma	11
2.6.2 Akış gezgini	11
2.6.3 Xilinx SDK ortamı	14
2.7 Xilinx SDSoc Platformu	15
2.7.1 XfOpenCV kütüphanesi	15
2.7.2 Proje oluşturma	15
2.7.3 XfOpenCV kütüphanesi bağlantılama	16
2.7.4 OpenCV kütüphanesi bağlantılama.....	16
3. ŞERİT TAKİP ALGORİTMASI YAZILIMI	19
3.1 Ön İşleme	20
3.2 Çizgi Tespiti	22

3.3 Şerit Çizimi	24
3.4 Kontrol Sınıfı.....	24
4. DONANIM VE YAZILIM BLOKLARININ AYRILMASI	25
4.1 Zaman Analizi	25
4.2 Donanım Yazılım Blokları	25
5. SDSOC PLATFORMU İLE FPGA ÜZERİNDE GERÇEKLEME	27
5.1 XfOpenCV Fonksiyonları	27
5.2 Alan Kullanımı	28
5.3 Zaman Analizi	30
6. VIVADO PLATFORMU İLE DONANIM TASARIMI.....	33
6.1 Kamera Modülü.....	34
6.1.1 Kamera bilgileri.....	34
6.1.2 Kamera devresi giriş çıkışları.....	35
6.2 Kameradan Görüntü Alınması	37
6.2.1 Capture modülü	38
6.2.2 Kamera kontrol modülü	40
6.2.2.1 SCCB protokolü	40
6.2.2.2 Modülün tasarımı	42
6.3 RGB2HLS Modülü	43
6.4 Sobel ve Bitisel Veya Modülü.....	44
6.5 Blok RAM Modülü	48
6.6 VGA Modülü.....	49
6.7 Saat Bölücü Modülü.....	53
6.8 Tasarlanan Donanımın Benzetim Sonuçları.....	53
6.8.1 Kaynak kullanımı ve güç tüketimi	53
6.8.2 Zaman analizi	55
7. SONUÇ.....	57
KAYNAKLAR.....	59
ÖZGEÇMİŞ.....	63

KISALTMALAR

ARM	: Advanced RISC Machine
ASIC	: Application Specific Integrated Circuits
AXI	: Advanced eXtensible Interface
BSD	: Berkeley Software Distribution
CGI	: Common Gateway Interface
CMOS	: Complementary Metal Oxide Semiconductor
FPGA	: Field Programmable Gate Array
Fps	: Frame Per Second
HDL	: Hardware Description Language
HDMI	: High Definition Multimedia Interface
HSYNC	: Horizontal Synchronization
I2C	: Inter Integrated Circuit
I²C	: Inter Integrated Circuits
IP	: Intellectual Property
LUT	: Look-up Table
OpenCV	: Open Source Computer Vision
RAM	: Random Access Memory
RGB	: Red Green Blue
SCCB	: Serial Camera Control Bus
SDK	: Software Development Kit
SIOC	: Serial Input Output Clock
SIOD	: Serial Input Output Data
VGA	: Video Graphic Array
VHDL	: Very High Speed Integrated Circuit Hardware Description Language
VSYNC	: Vertical Synchronization

ÇİZELGE LİSTESİ

Sayfa

Çizelge 4.1 : Macbook Pro'da şerit takip sistemi zaman analizi.	25
Çizelge 5.1 : XfOpencv ve OpenCV'de fonksiyon karşılıkları.....	27
Çizelge 5.2 : SDSoC Zedboard yazılım zaman analizi.....	30
Çizelge 5.3 : SDSoC Zedboard işlemci + donanım zaman analizi.	31
Çizelge 6.1 : Kamera giriş çıkışları.....	35
Çizelge 6.2 : VGA sinyalleri.	50
Çizelge 6.3 : Tarama için gerekli frekans ve tarama çevrimleri [39].....	52
Çizelge 6.4 : Zaman analizi.	56

ŞEKİL LİSTESİ

	<u>Sayfa</u>
Şekil 2.1 : Mantık hücresi yapısı [17].	6
Şekil 2.2 : FPGA iç yapısı	7
Şekil 2.3 : Zedboard geliştirme kiti	8
Şekil 2.4 : Arm Cortex A9 mimarisi [22]	9
Şekil 2.5 : Vivado ortamında blok tasarım	10
Şekil 2.6 : Vivado platformu	11
Şekil 2.7 : Vivado ortamı	12
Şekil 2.8 : Ayarlar seçeneği	13
Şekil 2.9 : Sentezleme stratejisi	13
Şekil 2.10 : SDSoc platformu	16
Şekil 2.11 : OpenCV ve xlopenCV kütüphanesini bağlantılama	17
Şekil 2.12 : OpenCV kütüphanesini bağlantılama	17
Şekil 3.1 : Yazılım akış grafiği	19
Şekil 3.2 : Ön işleme aşamaları	21
Şekil 3.3 : Perspektif dönüşümü	22
Şekil 3.4 : X Eksenine göre histogram alınması	23
Şekil 3.5 : İzleyen pencereler	23
Şekil 3.6 : Şerit çizimi.....	24
Şekil 4.1 : Donanım akış grafiği.....	26
Şekil 5.1 : xf::Sobel kaynak kullanımı	28
Şekil 5.2 : xf::absdiff kaynak kullanımı	28
Şekil 5.3 : xf::Threshold kaynak kullanımı	29
Şekil 5.4 : xf::bitwise_or kaynak kullanımı	29
Şekil 5.5 : Xf:warpTransform kaynak kullanımı.....	30
Şekil 6.1 : Tasarımın genel şematifi	33
Şekil 6.2 : OV7670 kamera Modülü	34
Şekil 6.3 : Kamera modülü blok şeması	34
Şekil 6.4 : FPGA-kamera bağlantısı.....	36
Şekil 6.5 : FPGA üzerinde gerçekleştirilen sistemin RTL şematifi	37
Şekil 6.6 : Capture Modülü	38

Şekil 6.7 : RGB444 formatı için sinyal diyagramı.....	39
Şekil 6.8 : SCCB haberleşmesi blok diyagramı	40
Şekil 6.9 : SCCB yazma işlemi	41
Şekil 6.10 : SCCB okuma işlemi.....	41
Şekil 6.11 : SCCB haberleşmesi sinyal diyagramı.....	42
Şekil 6.12 : Camera_controller modülü	42
Şekil 6.13 : Kamera kontrol tutucularının bir kısmı	43
Şekil 6.14 : RGB – HLS renk uzayı dönüşümü [34].....	43
Şekil 6.15 : RGB2HLS modülü	44
Şekil 6.16 : Sobel_bitwise modülü.....	45
Şekil 6.17 : Yatay sobel filtresi	45
Şekil 6.18 : Filtreleme işlemi.....	46
Şekil 6.19 : Filtre çekirdeği ve uygulanacağı piksel bölgeleri	47
Şekil 6.20 : Line buffer yapısı.....	47
Şekil 6.21 : Örnek bir “bitset veya” işlemi	48
Şekil 6.22 : Projeye istenilen özelliklerde blok RAM eklenmesi	48
Şekil 6.23 : Blok RAM’in boyutlarının ayarlanması	49
Şekil 6.24 : VGA konektör ve pin yapısı	50
Şekil 6.25 : Dikey ve yatay tarama işlemleri.....	51
Şekil 6.26 : Vsync ve Hsync sinyalleri	51
Şekil 6.27 : VGA modülü	52
Şekil 6.28 : Saat bölücü modülü	53
Şekil 6.29 : Sistemin alan kullanımı.....	54
Şekil 6.30 : Sistemin güç tüketimi sonuçları	54
Şekil 6.31 : Orijinal ve işlenmiş şerit görüntüsü.....	55

ŞERİT TAKİP ALGORİTMALARININ SDSOC VE VİVADO PLATFORMLARI KULLANILARAK FPGA'DE GERÇEKLENMESİ

ÖZET

Açık kaynak olarak gerçekleştirilen “OpenCV” kütüphanesi ile görüntü işleme algoritmalarının kullanım oranı büyük bir ivme kazanmıştır. Bir sürücü destek sistemi olan şerit takip algoritmaları da görüntü işleme teknikleri kullanılarak geliştirilen algoritmalarından biridir. Fakat görüntü işleme tekniklerinin sebep olduğu işlem yoğunluğu sebebiyle işlemciler üzerinde yavaş çalışmaktadır. Bu yavaş çalışma problemini çözmek için “Grafik İşlemci Ünitesi” (Graphics Processing Unit - GPU) gibi yüksek fiyatlı donanımlar kullanılmaktadır. Projelerde GPU kullanımı proje maliyetini artırmakta ve son kullanıcıya yüksek fiyatta ürün olarak sunulmaktadır. Bu durum görüntü işleme tekniklerinin ve uygulamalarının gündelik hayatta kullanılabilirliğini azaltmaktadır.

Bu projede bu problemin çözümü için “Xilinx” firmasının “Zynq-7000” ailesine ait Zedboard FPGA geliştirme kartı üzerinde kırmık üstü sistem (System of Chip - SoC) tanımlanmıştır. Görüntü işleme projesi olarak şerit takip sistemi algoritması geliştirilmiş ve bu geliştirilen algoritma yazılım – donanım olarak parçalandıktan sonra donanım kısmı FPGA’de gerçekleştirilmiştir. FPGA’de gerçekleştirim SDSoc platformu ile birlikte XfOpenCV kütüphanesiyle ve Vivado platformuyla yapılmıştır.

REALIZATION OF LANE DETECTION ALGORITHMS ON FPGA USING SDSOC AND VIVADO

SUMMARY

The usage rate of image processing algorithms has gained a great impetus with the "OpenCV" library, which is implemented as open source. The lane detection algorithms, which are an advanced driver assistance system, are also one of the algorithms that using image processing techniques. Unfortunately, the image processing algorithms are slow on processors because of the processing intensity of image processing techniques. High-priced hardwares such as "Graphics Processing Unit" (GPU) are used to accelerate this slow operation. But the usage of GPU in projects increases the project costs, so the end user is offered as a product at high price. This situation reduces the usability of image processing techniques and applications in daily life.

In this project, the Zedboard FPGA development card of the ZYNQ-7000 series of Xilinx Company is used as "System on Chip" (SoC) to solve this problem. A lane detection algorithm has been developed as an image processing project, and it has been implemented in the FPGA after this developed algorithm was broken down as software and hardware.

The lane detection algorithm is written in "C++" with "OpenCV" library. The algorithm is based on expressing the lines in a lane as second order polynomial equations. It has 3 main parts. The first part is "Preprocessing" function which extracts features from the image. The second part is "Detect Line" function which detects lines in a lane from the extracted features. The final part is "Draw Lane" function which is basically drawing gap between the lines. The gap represents the lane.

According to the time analysis of the lane detection algorithm, the preprocessing part was implemented in hardware. SDSoC and Vivado platforms were used to make implementation of the developed lane detection algorithm on FPGA. Low level code (Verilog) is used in Vivado platform while high level code (C++) is used in SDSoC platform. In SDSoC platform, the "xfOpenCV" library was also used. The "xfOpenCV" library is a set of 50+ kernels, optimized for Xilinx FPGAs and SoCs, based on the OpenCV computer vision library. The kernels in the xfOpenCV library are optimized and supported in the Xilinx SDSoC Tool Suit.

1. GİRİŞ

Görüntü işleme, görüntünün sayısal biçimine amaca yönelik görüntü elde etmek veya ondan bazı yararlı bilgiler çıkarmak için geliştirilmiş yöntemlerdir [1]. Bu yöntemin girdisi bir video veya yalnızca bir fotoğraf olabilir. Çıktısı ise görüntünün istenilen yada dikkat edilmesi gereken bölümüne karşılık gelir. Genellikle görüntü işleme sisteminde, önceden belirlenmiş sinyal işleme yöntemleri uygulanırken görüntü iki boyutlu sinyal olarak ele alınır. Bu iki boyutlu sinyalin her elamanı bir piksele karşılık gelir. Renkli görüntüler için ise genelde 3 boyutlu sinyaller kullanılır ve en yaygın gösterim Kırmızı Yeşil Mavi (Red Green Blue – RGB) renk uzayı formatındadır. Bu formatta her piksele ait 3 adet değer vardır ve bu değerler bağlı olduğu boyuttaki renk değerini gösterir.

Günümüzde akademinin ve işletmelerin çeşitli yönleriyle kullandıkları görüntü işleme sistemleri hızla büyüyen teknolojiler arasında yer alır ve yaygın bir kullanım alanı vardır. Bu alanlardan birisi olan Gelişmiş Sürücü Destek Sistemleri (Advanced Driver Assistance System - ADAS) görüntü işleme algoritmaları sayesinde sürücülere yardımcı olur [1]. Aynı zamanda insansız otonom arabalara geçiş evresinde önemli bir rol üstlenir. Bu projede bir ADAS olan şerit takip algoritması gerçekleştirilmiştir.

Görüntü işleme algoritmalarının kullanım oranı açık kaynak kodu olarak gerçekleştirilen “OpenCV” kütüphanesi ile büyük bir ivme kazanmıştır. Fakat görüntü işleme tekniklerinin sebep olduğu işlem yoğunluğu sebebiyle işlemciler üzerinde yavaş çalıştığı görülür. Özellikle ADAS’da algoritmanın hızlı çalışması çok önemlidir. Bu yavaş çalışma problemini çözmek için Grafik İşlemci Ünitesi (Graphics Processing Unit - GPU) gibi yüksek fiyatlı donanımlar yada yüksek işlem kapasiteli Merkezi İşlem Birimleri (Central Processing Unit - CPU) kullanılabilir. Bu donanımlar proje maliyetlerini artırmakta ve son kullanıcı yüksek fiyatlı bir ürün ile karşılaşmaktadır. Bu durum görüntü işleme tekniklerinin ve uygulamalarının gündelik hayatta yaygınlığını azaltmaktadır.

Bu projede görüntü işleme algoritmalarını düşük maliyetle kullanılabilirliğini göstermek adına “Xilinx” firmasının “Zynq-7000” ailesine ait Zedboard FPGA

geliştirme kartı üzerinden “kırmık üstü sistem” (System of Chip - SoC) tanımlanmıştır. Görüntü işleme projesi olarak şerit takip algoritması geliştirilmiştir ve bu geliştirilen sistem yazılım – donanım olarak parçalandıktan sonra donanım kısmı gerçekleşmiştir. Şerit takip algoritması “C++” ortamında “OpenCV” kütüphanesi kullanarak yazılmıştır ve 3 ana fonksiyona sahiptir. İlk fonksiyon “ön işleme” bölümünde fotoğraf çerçevesinden anlamlı bilgiler çıkarılmaya çalışılmış ikinci bölüm “şerit bulma” fonksiyonunda çıkarılan bu bilgilerden şerite ait çizgiler tespit edilmesi amaçlanmış ve son bölüm “şerit çizme” fonksiyonu ile de şeriti ifade edecek olan tespit edilen çizgilerin arası boyanmış ve görselleştirilmiştir. Yapılan zaman analizine göre ilk bölüm olan ön işleme kısmı donanımda gerçekleşmiştir. Projenin Zedboard geliştirme kartında gerçekleşmesi için SDSoC ve Vivado platformları kullanıldı. SDSoC platformunda yüksek seviyeli yazılım (C++) kullanılırken; Vivado platformunda bir donanım tanımlama dili olan Verilog kullanılmıştır.

1.1 Projenin Amacı

Bu projede ilk olarak şerit takip sistemi gerçekleşmesi amaçlanmıştır. İkincil olarak, gerçekleştirilen algoritmanın zaman analizine göre yazılım – donanım olarak bölüntülenmesi, üçüncül ve dördüncül olarak donanım kısımlarının SDSoC ve Vivado platformları kullanılarak gerçekleşmesi amaçlanmıştır. Projenin son amacı Vivado ve SDSoC platformlarında gerçekleştirilen tasarımların sonuçlarının yorumlanmasıdır.

1.2 Literatür Araştırması

Yapılan literatür araştırması şu şekildedir:

I. El Haccouji ve arkadaşları [1] tutarlı şerit tespiti ve takip sistemi için yaptıkları çalışmalarında; şeritleri bulmak için: gri seviyeli görüntüler üzerinde kenar bulmak amacıyla kullanılan “Sobel” işleci [2] ve bir görüntüdeki şekilleri saptamada kullanılan bir yaklaşım yöntemi olan “Hough Dönüşümü” (Hough Transform) [3] yöntemlerini kullanıyorlar. Şerit takibi için ise yinelemeli olarak bir durumun sürecini tahmin ederken diğer yandan yapılan hatayı minimize eden yani sürekli gerçek değere ulaşmayı hedefleyen bir formülasyon olan Kalman filtresi [4] kullanmıştır. Gerçek zamanlı olması için de FPGA tabanlı sistemde aritmetik hesaplamalar için geliştirilen etkin bir algoritma olan koordinat ve rotasyonlu sayısal bilgisayarda (Coordinate

Rotation Digital Computer - CORDIC) [5] hiperbolik ve trigonometrik fonksiyonları gerçekleştiriyor. Kullandığı FPGA sistemindeki fonksiyonlar bit kaydırma, toplama, çıkarma ve doğruluk tablosundan oluşmaktadır.

II. M. Revilloud ve arkadaşları [6] şerit tespiti ve şerit alanı tahmini için filtre methodu geliştirmiştir. Resimde, şerit olma olasılığını piksellerin gri tonlamalı yoğunluklarına göre değerlendirmişlerdir.

III. R. Fan ve arkadaşları [7] çoklu görüntü ön işleme yöntemlerine dayanan doğrusal şerit tespit sistemi için çalışmışlar ve bu çalışmalarında Hough Dönüşümü metodunu kullanmışlardır. Birden fazla platforma uyumlu "C" bazlı çalışmalarını SoC'e entegre ettiler.

IV. G. Cahple ve arkadaşları [8] gri tonlamalı görüntüdeki kenar piksellerini bulmak için "Sobel" metoduna dayalı kenar algılama algoritması tasarımı sunmaktadır. Tasarım için Xilinx ISE Design Suite-14 yazılım platformu ve donanım tanımlama dili kullanırlar. MATLAB yazılım platformundan, gri tonlamalı görüntüdeki piksel verilerini elde etmek veya tersini yapmak için yararlanıyorlar.

V. Tianchen Wang'ın [9] gerçek zamanlı ADAS (İleri Sürücü Asistan Sistemi) sistemleri için yaptığı çalışmada ADAS sistemlerinin gerçek zamanlı çalışıyor denilebilmesi için saniyede 15 çerçeve işleyebiliyor olması gerektiği belirtilmiştir.

Geliştirilen şerit takip sisteminde ikinci dereceden polinom denklem kullanılmış olup eğimli yollarda daha doğru sonuç veren bir sistem olarak literatürde yapılan çalışmalardan ayrılmaktadır. Aynı zamanda "Xilinx" firmasının "ZYNQ-7000" ailesine ait FPGA geliştirme kartında gerçekleştirilebilirliğini göstererek donanım/yazılım şeklinde SoC yapısında olan bir sistem tasarlandığı için literatürdeki diğer çalışmalardan yine ayrılmıştır.

2. ÖNBİLGİLER

2.1 Görüntü İşleme Kütüphaneleri

2.1.1 OpenCV kütüphanesi

Açık Kaynak Bilgisayarla Görü (Open Source Computer Vision - OpenCV) açık kaynak kodlu görüntü işleme kütüphanesidir [10]. OpenCV kütüphanesi ilk olarak INTEL tarafından 1999 yılında geliştirilmeye başlanmış ilerleyen zamanlarda çeşitli firmaların desteğini alarak Berkeley Yazılım Dağıtımı (Berkeley Software Distribution – BSD) lisansı [10] altında açık kaynak yazılımı olarak geliştirilmeye devam edilmiştir. BSD lisansına sahip olması OpenCV kütüphanesinin istenilen projede ücretsiz olarak kullanabileceği anlamına gelmektedir. OpenCV platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, Mac OS, iOS ve Android ortamlarında çalışabilmektedir. C++, Python ve Java arayüzlerine sahiptir. OpenCV, gerçek zamanlı uygulamalarda kullanılması için hesaplama verimliliği ön planda tutularak geliştirilmektedir.

2.1.2 Alternatif görüntü işleme kütüphaneleri

MATLAB: Matlab içerisinde görüntü işlemeye yönelik temel algoritmaları barındırmaktadır [11]. Akademik araştırmalarda, performansın önemli olmadığı durumlarda temel görüntü işlemleri için uygundur. Matlab kullanarak OpenCV Kütüphanesi ile etkileşimli uygulamalarda geliştirmek mümkündür. OpenCV gerçek zamanlı uygulamalardaki performansı ile Matlab'e göre daha fazla avantajlıdır.

Halcon: Halcon bilgisayarda görü projeleri için MvTec firması tarafından oluşturulmuş ticari bir geliştirme ortamıdır. Geliştirme ortamının yanı sıra C, C++, C# gibi programlama dilleri içinde kütüphanesi vardır [12]. İçerisinde birçok hazır fonksiyon bulundurur bu sayede hızlı uygulamalar geliştirilebilir. Daha çok endüstriyel projeler için tercih edilen ticari bir yazılımdır. OpenCV açık kaynak kodlu, ücretsiz bir kütüphanedir ve bilgisayarda görü konusuna özel olarak performans geliştirmeleri yapılmaktadır. OpenCV bu yönleri ile Halcon'dan ayrılmaktadır.

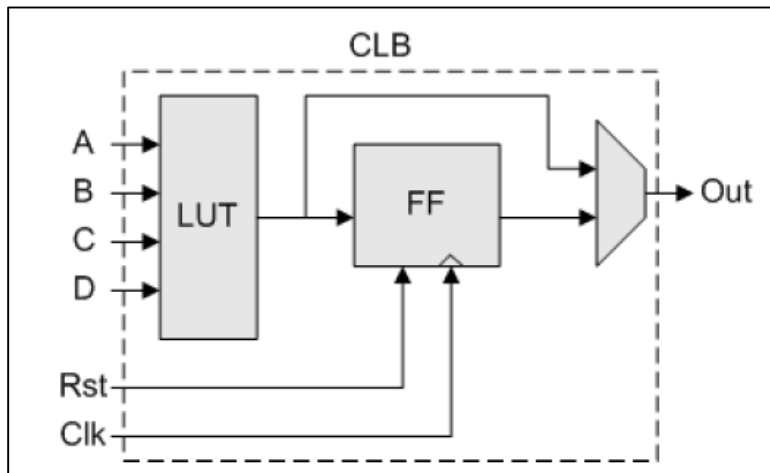
CIMG: Açık kaynak kodlu bir görüntü işleme kütüphanesidir ve Windows, Linux ve Mac OS platformları üzerinde çalışmaktadır. Sadece C++ dili için desteği bulunmaktadır fakat dönüştürücüler ile Java ve Python ile de uygulama geliştirilebilmektedir [13]. CIMG da birçok algoritmayı barındırmaktadır fakat OpenCV kadar performanslı ve geniş bir algoritma altyapısına sahip değildir.

Fiji: Java platformu için geliştirilmiş açık kaynak kodlu bir görüntü işleme kütüphanesidir. Windows, Linux ve MAC OS Intel 32-bit veya 64-bit üzerinde çalışır. Bilimsel görüntü analizi için geliştirilmiştir [14]. Genetik, hücre biyolojisi, nöro-bilim gibi alanlar için özelleştirilmiş algoritmalara sahiptir. Bulundurduğu algoritma yelpazesi açısından OpenCV daha avantajlıdır.

2.2 Sahada Programlanabilir Kapı Dizileri

Sahada Programlanabilir Kapı Dizileri (Field Programmable Gate Array - FPGA) istenilen fonksiyona göre iç yapısı programlanabilen özel kırımlardır ve üretimden sonra programlanabilirler [15]. Bu nedenle “Sahada Programlanabilir” adı verilmiştir. İç yapısı programlanabilir mantık hücreleri, giriş-çıkış birimleri ve yönetilebilir anahtarlardan oluşur.

FPGA’in ana yapısını mantık hücreleri oluşturur. Mantık hücreleri genellikle 3 temel yapının bir araya gelmesiyle oluşur. Bunlar Look-up Table (LUT), D flip-flop ve MUX (multiplexer - seçici)’dir. Genel mantık hücresi yapısı Şekil 2.1’de gösterilmiştir.



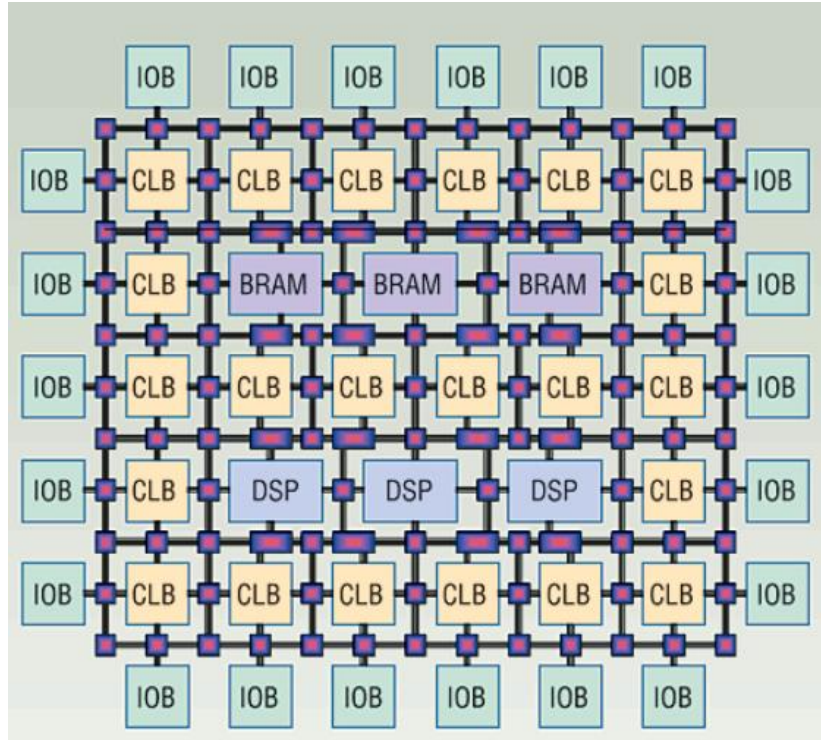
Şekil 2.1 : Mantık hücresi yapısı [17].

LUT’lar basit mantık işlemlerini yerine getiren belleklerdir. Çok sayıda mantık hücresinin birleşmesiyle daha karmaşık sistemler gerçekleştirilebilir. Mantık hücreleri

birbirlerine programlanabilir anahtarlar ile bağlantılar kurabilir. Gerçeklemek istediğimiz sayısal tasarımlar anahtarların ve mantık hücrelerinin programlanmasıyla elde edilir.

FPGA'ler üzerinde donanım gerçeklemek için Verilog [16] ve VHDL [17] gibi donanım tanımlama dilleri kullanılır. Verilog kolay söz dizimi ve C diline benzeyen yapısı sebebiyle tasarıma yeni başlayanlar tarafından sıkça tercih edilir. Verilog ile VHDL kullanılarak yazılan kodlar aslında birer özel donanıma karşılık gelirler. Bu diller kullanılarak basit flip flop devrelerinden çok daha karmaşık mikroişlemci sistemlerine kadar birçok donanım gerçekleştirilebilir. Donanım tanımlama dilleri kullanılarak yapılan tasarımlar FPGA üreticilerinin sağladığı yazılım platformları vasıtasıyla sentezlenir ve mimariye uygun hale dönüştürülür. Daha sonra bir programlama kablosu vasıtasıyla FPGA'ye gömülerek sistem gerçekleştirilir.

Giriş çıkış birimleri, programlanabilir ara bağlantılar ve mantık hücrelerinden (Configurable Logic Block – CLB) oluşan FPGA'in iç yapısı Şekil 2.2'de gösterilmiştir.



Şekil 2.2 : FPGA iç yapısı

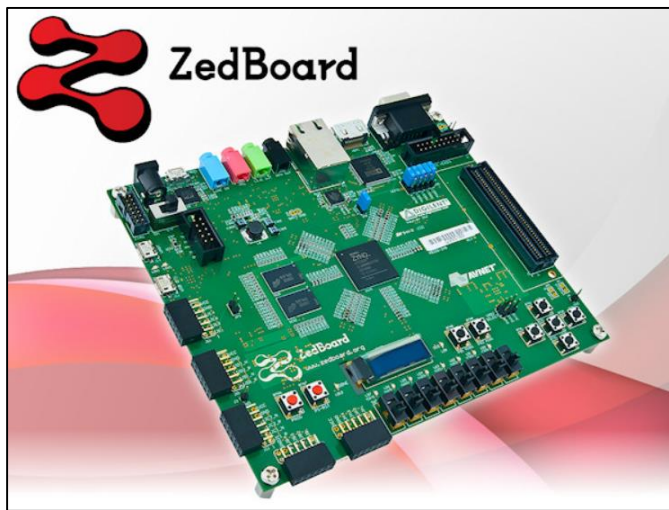
FPGA mimarisinin en önemli avantajı paralel işlem kabiliyetidir. Aynı anda birçok paralel işlemi gerçekleyebilirler ve bu sayede çok hızlı sistemlerin tasarlanmasına

olanak sağlarlar. Sinyal ve görüntü işleme uygulamalarında, mikroişlemcilerde göre çok daha üstün performans gösterdiği için FPGA'ler tercih edilmektedir. Ayrıca Uygulamaya Özgü Tümdevre (Application Specific Integrated Circuits – ASIC) [18] tasarımlarına göre maliyeti az ve piyasaya çıkma süresi daha kısadır. FPGA'lerin önemli avantajlarından bir diğeri de içerisine mikroişlemci gömebilmenin mümkün olmasıdır. Bütün bir sistemin aynı çip üzerinde yer alması sayesinde bağlantılar arası gecikmeler azaltılır ve FPGA üzerinde hızlı sistemler tasarlanabilir [19].

Yüksek paralel işlem kabiliyeti ve istenildiği zaman tekrar programlanabilmesinin sağladığı önemli avantajlar sebebiyle bu tez çalışmasında FPGA kullanılmıştır.

2.3 Zedboard Geliştirme Kiti

Bu tez çalışmasında Xilinx Zynq-7000 All Programmable SoC'yi kullanan ZedBoard geliştirme kartı kullanılmıştır. Kart, üstündeki bir çok ara birim sayesinde Windows, Linux, Android ve diğeri işletim sistemleri tabanında tasarım oluşturmaya olanak sağlar [20]. Ayrıca içerdiği farklı türden bir çok giriş çıkış portu sayesinde tasarım projelerinde büyük esneklik sağlamaktadır [21]. Üzerinde çift çekirdekli ARM Cortex-A9 [22] işlemci bulunur, bu sayede yazılım-donanım ortak tasarım yapılmasına olanak sağlar. Zynq-7000 cihazları 6.25Gb/s'den 12.5Gb/s'ye kadar haberleşme hızı sunar. Yüksek çözünürlüklü video işleme, ses işleme, yapay zeka uygulamaları, yazılım hızlandırması, gömülü ARM sistemleri geliştirme gibi bir çok gömülü sistem uygulamasının gerçekleştirilmesine olanak sağlar. ZedBoard geliştirme kitinin görünümü Şekil 2.3'teki gibidir.

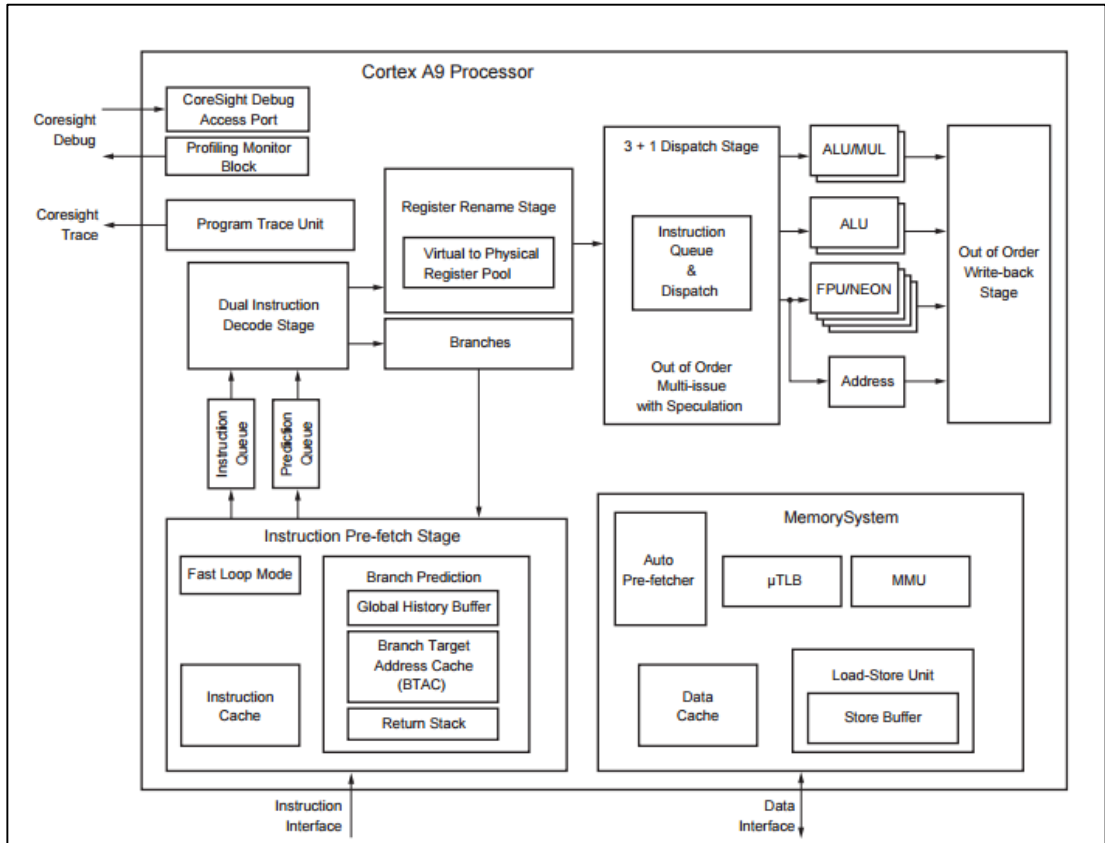


Şekil 2.3 : Zedboard geliştirme kiti

2.4 ARM İşlemci

Zynq-7000 ailesi ürünlerinde çift çekirdekli ARM Cortex-A9 işlemcileri bulunmaktadır. ARM; Acorn RISC Machine kelimelerinin kısaltılmasıdır ve 32-bit İndirgenmiş Komut Kümesi ile Hesaplama (Reduced Instruction Set Computing - RISC) [23] işlemci mimarisini temsil eder. Düşük enerji tüketimi, yüksek hız ve 32-bit mimari yapısı sayesinde gömülü sistem projelerinde sıkça tercih edilmektedir. Günlük hayatta kullanılan birçok tüketici elektroniği ürününde ARM tabanlı mikroişlemciler kullanılmaktadır.

ARM işlemci, medya işleme motoru (NEON), 32 KB L1,512 KB L2 önbellek (cache), iş hattı (pipeline) derinliği, kayan noktalı sayı birimi,bellek idare birimi gibi özelliklere sahiptir [24]. ARM Cortex A9 mimarisi Şekil 2.4'te gösterilmiştir.



Şekil 2.4 : Arm Cortex A9 mimarisi [22]

2.5 Verilog Donanım Tanımlama Dili

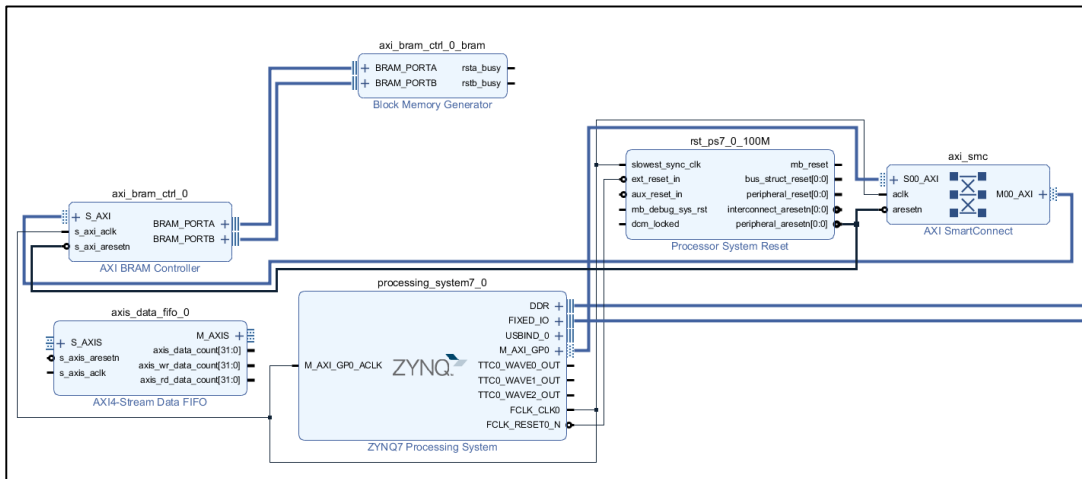
Verilog 1983/1984 yıllarında Automated Integrated Design Systems Phil Moorby ve Prabhu Goel tarafından icat edilmiştir [25]. Bu dil sayısal sitemlerin tasarlanması ve

test edilmesi için kolay bir söz dizimine sahip olacak biçimde oluşturulmuş ve yıllar içinde belli değişikliklere uğramıştır. Bu dilin tasarımcıları C diline benzer bir söz dizimi oluşturmaya çalışmışlardır. Bu sayede yeni öğrenmeye başlayanlar için kolay anlaşılır bir yapıya sahiptir [26]. Verilog kullanarak kapı seviyesinde tasarım yapılabileceği gibi davranışsal programlamaya uygun yapısı sayesinde, karmaşık sistemler rahatlıkla tasarlanabilir.

2.6 Xilinx Vivado Platformu

Vivado Xilinx firmasının geliştirdiği ve ilk olarak 2012 yılında tanıttığı bir arayüz yazılımıdır [27]. Vivado'dan önce Xilinx'in tasarımcılara sunduğu ISE programı mevcuttu. Vivado'nun yayınlanmasıyla birlikte Xilinx ISE'nin geliştirmesini durdurmuştur. Vivado, ISE'ye göre daha kullanışlı bir arayüze sahiptir bir çok özelliği tek bir çatı altında toplaması sayesinde çok daha kullanışlı bir ortam sağlar. Vivado'da Xilinx'in yeni nesil FPGA'leri programlanabilirken, eski nesil FPGA'leri desteklememektedir. Vivado platformu blok tasarım yapmayı, tasarlanan blokların paketlenmesini ve yazılım donanım olarak ortak tasarım yapılmasını oldukça pratik hale getirmiştir. Şekil 2.5'te örnek bir blok tasarım gösterilmektedir.

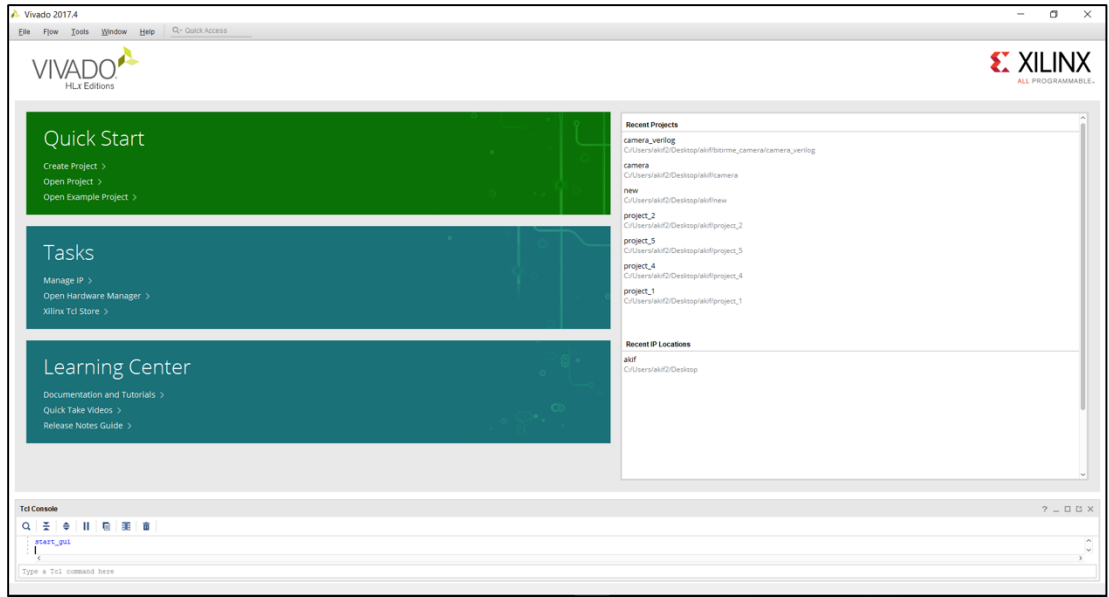
Bu projede Vivado'nun 2017.4 versiyonunun ücretsiz yayını olan WebPack Edition kullanılmıştır. Vivado Artix®-7, Kintex®-7, Kintex UltraScale™, Zynq®-7000 All Programmable SoC gibi FPGA kartlarını ücretsiz olarak desteklemektedir [21].



Şekil 2.5 : Vivado ortamında blok tasarım

2.6.1 Proje oluřturma

Vivado programı alıřtırıldıđında Őekil 2.6'daki gibi 3 ana blm kullanıcıyı karřılar. Bunlar hızlı bařlangı, grevler, đrenme merkezidir. Hızlı bařlangı blmnden yeni bir RTL projesi oluřturabilir veya nceden oluřturulan tasarımları geliřtirmeye devam edilebilir. Grevler kısmında IP blok retimi ve ynetimi iřlemleri gereklenebilir. đrenme merkezi Xilinx'in kullanıcılarına sunduđu nemli olanaklardan biridir. Burda program hakkında ve donanım tasarımı hakkında bir ok yazılı dkman ve video kaynak bulunmaktadır.



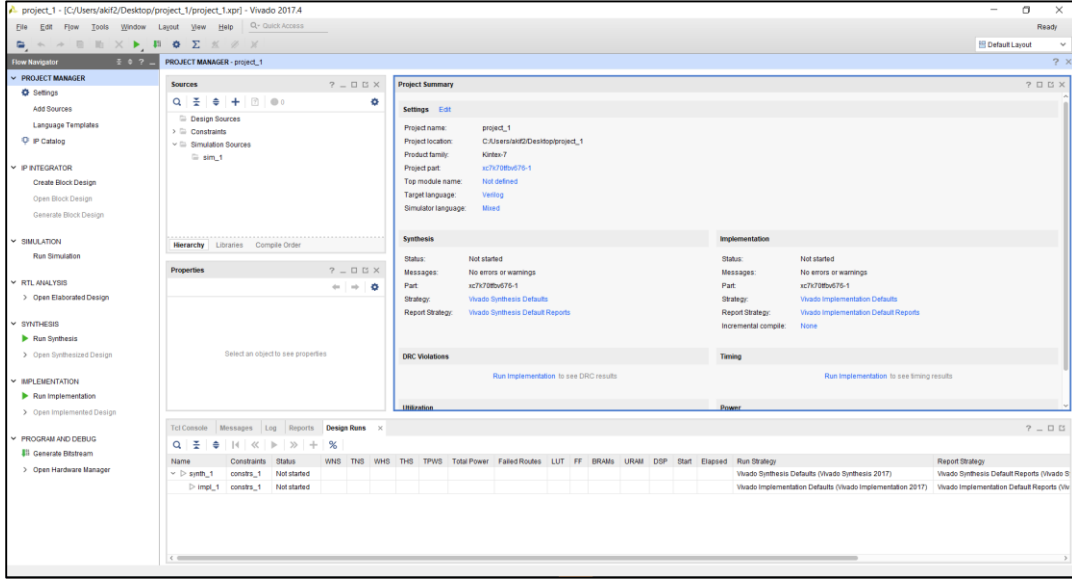
Őekil 2.6 : Vivado platformu

Oluřturulacak RTL projelerinde kullanılmak zilen dil Verilog veya VHDL olarak seilebilir. Birden fazla blođa sahip sistemlerde Verilog ve VHDL ile tasarlanan farklı bloklar birbirleriyle uyum iinde alıřır. Yani kullanıcı isterse bir projede iki donanım tanımlama dilini de kullanabilir. Bu projede bloklar Verilog dilinde yazılmıřtır. Proje oluřtururken, kullanılacak dili setikten sonra hangi geliřtirme kartının kullanılacađı seilir. Bu projede ZedBoard geliřtirme kartı kullanılmıřtır.

2.6.2 Akıř geźgini

Vivado 2017.4 srmnde kullanıcı arayzne nem vermiř nceki srmlerde yer alan gereksiz iřlem bloklarını kaldırarak grnmn basitleřtirmiřtir. Projelerde

kullanılacak işlemler Akış Gezgini'nde yukarıdan aşağıya doğru sıralanmıştır. Akış gezgininin genel görünümü Şekil 2.7'de gösterilmiştir.

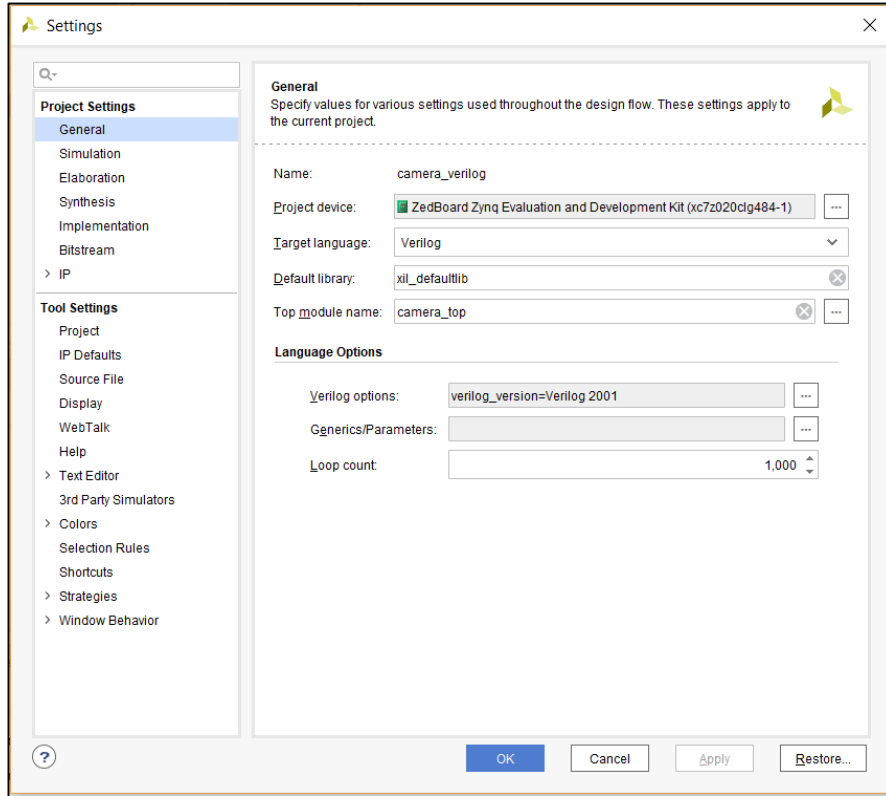


Şekil 2.7 : Vivado ortamı

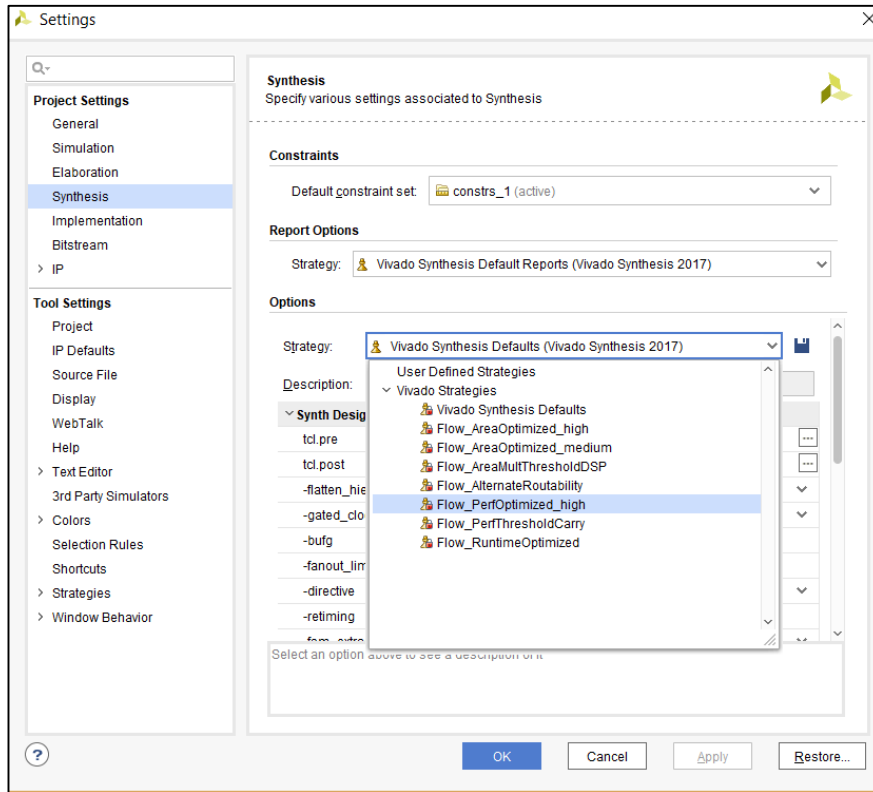
Project Manager başlığının altında “Settings” seçeneği bulunmaktadır. Bu kısımda genel tasarım ayarlarıyla birlikte, sentezleme, implemantasyon ve similasyon ayarları da bulunmaktadır. “Synthesis” sekmesinde programın sentez işleminde hangi kısıtlara göre optimizasyon yapacağı ayarlanabilir. Bu projede yüksek performans seçeneği seçilmiştir. Diğer seçenekler varsayılan ayarlarda kullanılmıştır. Şekil 2.8’de ayarlar seçeneği, Şekil 2.9’da sentezleme stratejisi gösterilmiştir.

“Project Manager” sekmesinden “Add Source” sekmesine tıklayarak yeni tasarım oluşturulabilir, var olan tasarımlar projeye dahil edilebilir. Ayrıca bu sekmeden çeşitli similasyon ve kısıt dosyaları eklenebilir. Similasyonlar projenin gerçekleşmeden önce Vivado ortamında çeşitli zaman analizlerinin yapılmasına olanak sağlar. Kısıt dosyaları ise tasarımın alan, hız, güç tüketimi gibi parametrelerin istenildiği taktirde belli ölçülerde sınırlandırılmasına olanak sağlar.

“Project Manager” IP katalog sekmesinden Xilinx firmasının kullanıcılara sunduğu hazır IP blokları bulunur. Bu blokların büyük bir kısmını kullanıcılar ücretsiz olarak kullanabilmektedir, bir kısmı ise farklı lisanslar gerektirir. Ayrıca oluşturulan projeler “IP Integrator” bölümünde paketlenerek IP blok haline getirilip bu kütüphaneye eklenebilir, daha sonra istenildiği zaman projelere dahil edilebilir. Bu kütüphane uygu-



Şekil 2.8 : Ayarlar seçeneği



Şekil 2.9 : Sentezleme stratejisi

lamalarda kullanıcılara büyük kolaylık sağlamaktadır. “RTL Analysis” sekmesinden yapılan tasarımın giriş çıkış ve ara bağlantılarla birlikte genel ve ayrıntılı şematiği görüntülenebilir, gürültü raporları incelenebilir.

“Synthesis” bölümünde tasarıma istenen kısıtlar göz önüne alınarak, yüksek seviye ve düşük seviye optimizasyonlar yapılarak sentezleme yapılır.

“Implementation” bölümünde FPGA üzerinde yerleştirme işlemi gerçekleştirilir. Yerleştirme işleminde önceden belirtilen kısıtlar göz önüne alınır. “Implementation” sonucunda program kullanıcıya, sistemin FPGA üzerinde ne kadar yer kapladığını, hangi bölgesine yerleştirildiğini ve hat gecikmelerini raporlar.

“Program and Debug” bölümünde “Generate Bitstream” ile tasarımın FPGA’ye gömülecek “bit” dosyası oluşturulur. “Open Hardware” sekmesinden FPGA kartı bağlantısı yapılarak FPGA programlanır.

2.6.3 Xilinx SDK ortamı

“SDK” açılımı Yazılım Geliştirme Kiti (Software Development Kit) olan ve Xilinx firması tarafından Vivado ortamında donanım-işlemci ortak tasarımlarda yazılım tasarımını gerçekleştirmek için geliştiren bir programdır.

Vivado platformunda oluşturulan mikroişlemcili tasarım paketlenip SDK’ya gönderilir ve mikroişlemci üzerinde yazılım geliştirmesi yapılır. SDK tarafında yer alan kaynak kütüphanelerin projeye eklenebilmesi sayesinde kullanıcılara yazılım geliştirme kısmında önemli avantaj sağlar.

- Zengin özellikli C/C++ kod editörü ve derleme ortamı
- Proje yönetimi
- Otomatik Makefile üretimi
- Hata navigasyonu
- Kaynak düzeyinde hata ayıklama ve gömülü hedeflerin görünüşü için iyi tümleştirilmiş ortam
- Kaynak kodu sürümü kontrolü

SDK tarafından kullanıcılara sunulmuş başlıca özelliklerdir [28].

SDK üzerinden yazılım geliştirme işlemi tamamlandıktan sonra donanım bilgilerini içeren “bit” uzantılı donanım dosyası ve “elf” uzantılı yazılım dosyası birleştirilerek FPGA’ye gömülür.

2.7 Xilinx SDSoC Platformu

Xilinx SDSoC platformu, Eclipse IDE üzerinde Xilinx SoC FPGA’leri için gömülü olarak C/C++/OpenCL uygulama geliştirme platformudur [29]. Bu platform, C++ ortamında yazılmış bir kodu -hızlandırmak amacıyla- FPGA’de donanım bloğu içinde kullanılmak üzere otomatik olarak donanım tanımlama diline çevirebilir. Yazılım ve donanım bloklarının otomatik olarak FPGA’ye gömülebileceği bu platform, ARM ile uyumlu gömülü yazılım ve donanım tarafı ile uyumlu bitstream (bit) dosyası oluşturur. Ayrıca SDSoC platformu, “OpenCV” kütüphanesi için donanımda gerçekleşmesi üzerine optimize edilmiş açık kaynak kod olarak paylaşılan “xfOpenCV” kütüphanesini sunar. İlk sürümü (2016.3) Aralık, 2016 yılında çıkmış olup şuanda 6. Versiyon en güncel versiyondur (2018.1).

2.7.1 XfOpenCV kütüphanesi

“XfOpenCV” kütüphanesi, “OpenCV” kütüphanesinden türetilmiş olup bir çok OpenCV fonksiyonunu Xilinx FPGA’leri için donanımda gerçekleşmesi üzerine optimize edilmiş bir kütüphanedir [30]. Xilinx SDSoC platformu ile çalışan bu kütüphane bilgisayarda görü projelerinde hızlandırıcı olarak rol oynar. İlk sürümü Haziran 2017’de çıkmış olup bu projede Aralık 2017’de çıkan üçüncü sürümü -şu anki en son sürüm- kullanılıyor.

2.7.2 Proje oluşturma

SDSoC Eclipse IDE tabanlı bir geliştirme platformu olduğundan dolayı proje oluşturma aşamalarında benzerlik vardır. Program ilk defa çalıştırıldığında ilk önce projelerin kaydedileceği “çalışma alanı” (workspace) dosya yolu oluşturulur, ardından ana bölüm kullanıcıyı karşılar. Bu alanda yeni proje oluşturulabilir, tanımlı FPGA geliştirme kartları haricinde yeni FPGA geliştirme kartlarının tanımlı olduğu dosyalar aktarılabilir, daha önceki projeler içe aktarılabilir ve SDSoC hakkında ulaşılabilecek eğitim dökümanlarına ulaşılabilir. Şekil 2.10’da bu alanın ekran görüntüsüne yer verilmiştir. Oluşturulacak SDSoC projelerinde kullanılacak olan FPGA geliştirme

kartı seçildikten sonra sistem yapılandırması başlığı altında projenin hangi işletim sisteminde çalışacağı seçilir (Linux, Standalone, FreeRTOS). Bu projede ZedBoard geliştirme kartı kullanılırken sistem yapılandırması olarak “Linux” seçilmiştir.

2.7.3 XfOpenCV kütüphanesi bağlantılama

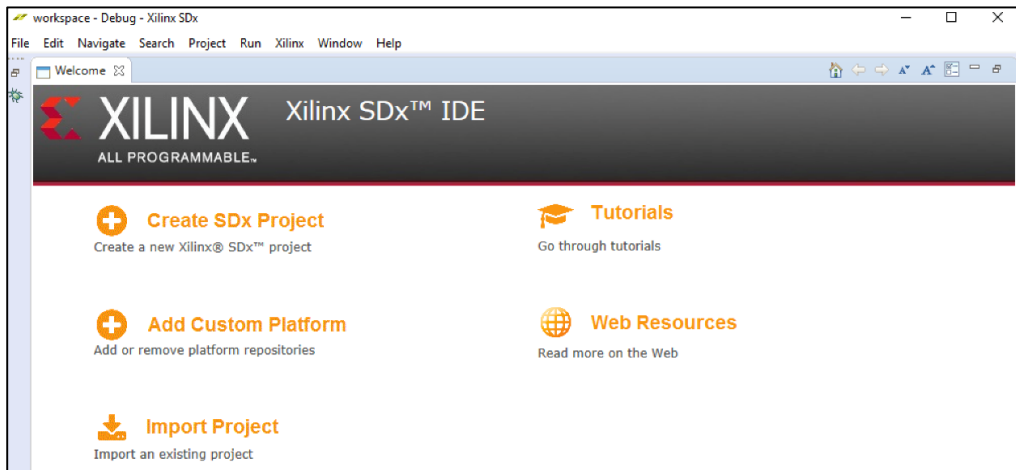
XfOpenCv kütüphanesi'nin SDSoC platformuna bağlantılanması için oluşturulan projede “C/C++ build” ayarına gelinir ve “SDS ++ Compiler” başlığı altında “Directories” sekmesine gelinir ve XfOpenCv kütüphanesi “Include Paths” yoluna eklenir. Örnek resim Şekil 2.11’de gösterilmiştir.

2.7.4 OpenCV kütüphanesi bağlantılama

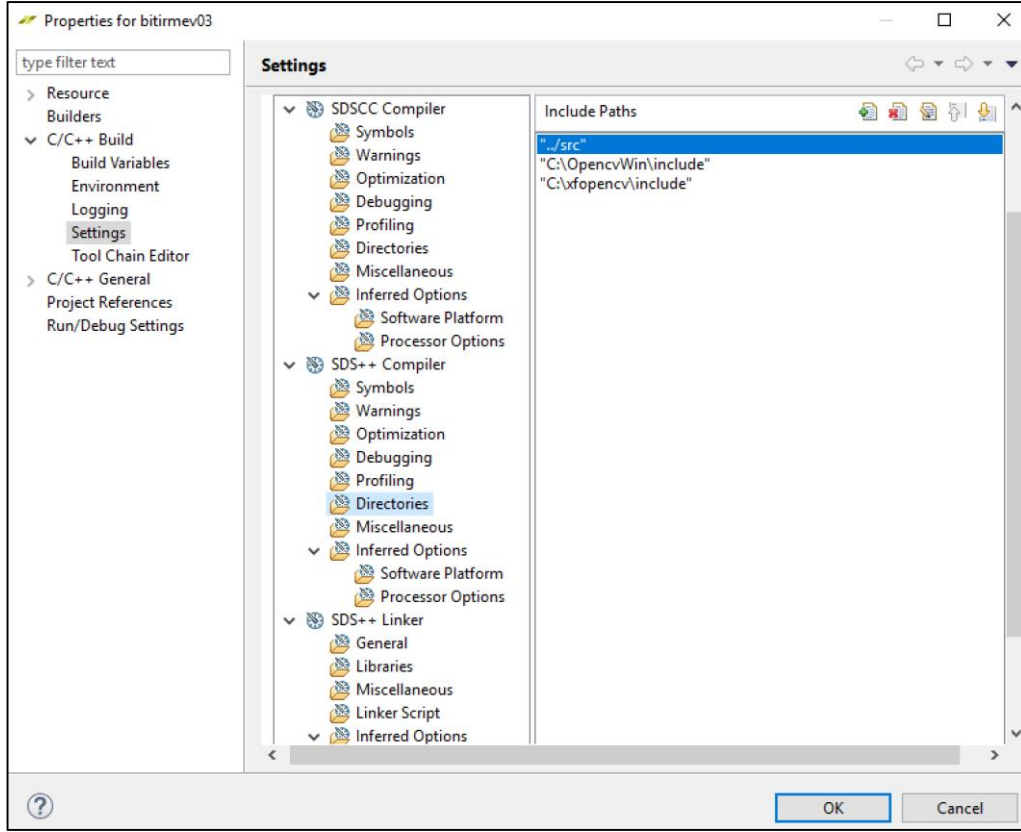
OpenCV kütüphanesinin ZedBoard Geliştirme Kartında kullanılması için OpenCV kütüphanesinin “AARCH32” mimarisine göre kurulum edilmesi gereklidir.

Kullanılan FPGA kartındaki işlemci mimarisine uygun kurulum sağlandıktan sonra projeye ait:

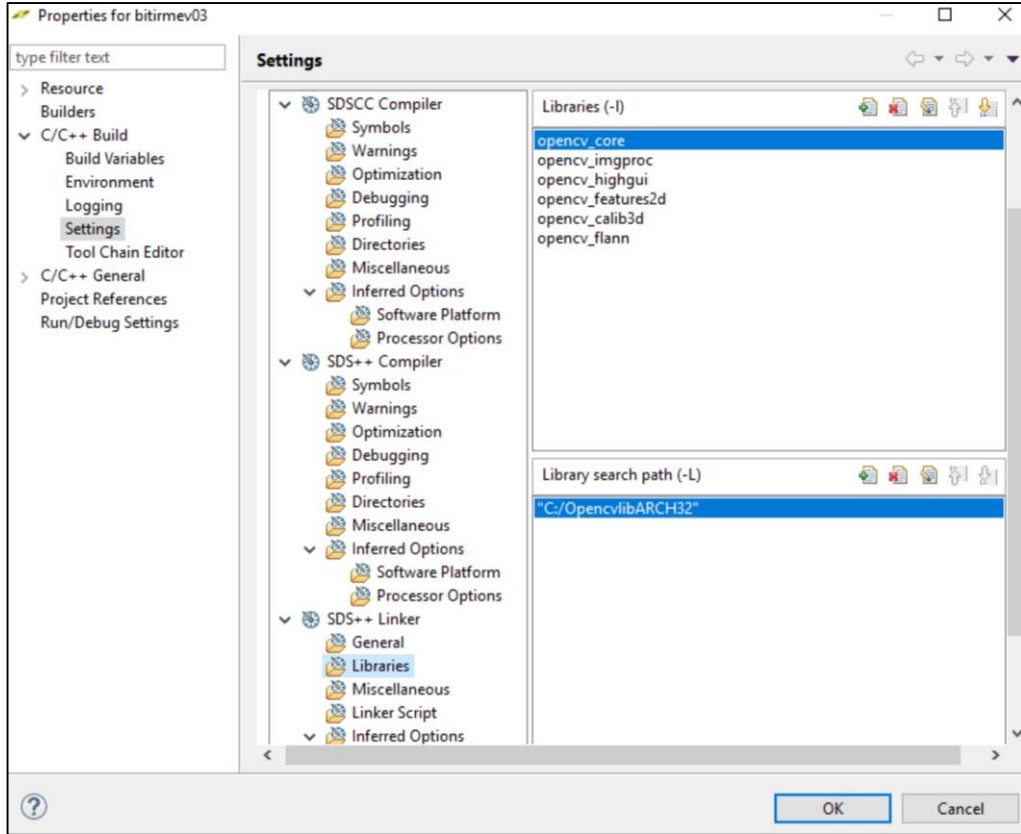
- “C/C++ build” ayarında “SDS ++ Compiler” başlığı altında “Directories” sekmesinde kullanılan bilgisayar ortam için derlenilmiş (bu projede Windows) OpenCV kütüphanesi “Include Paths” yoluna eklenir.
- Ayrıca “C/C++ build” ayarı altında “SDS ++ Linker” sekmesinde “Libraries” yoluna projede kullanılan OpenCV çekirdekleri; “Library Search Path” yoluna ise uygun hedef mimari için kurulumu yapılan (bu projede AARCH32) OpenCV kütüphanesi eklenir. Örnek bağlantılama resimlerine Şekil 2.11 ve Şekil 2.12’de ulaşılabilir.



Şekil 2.10 : SDSoC platformu



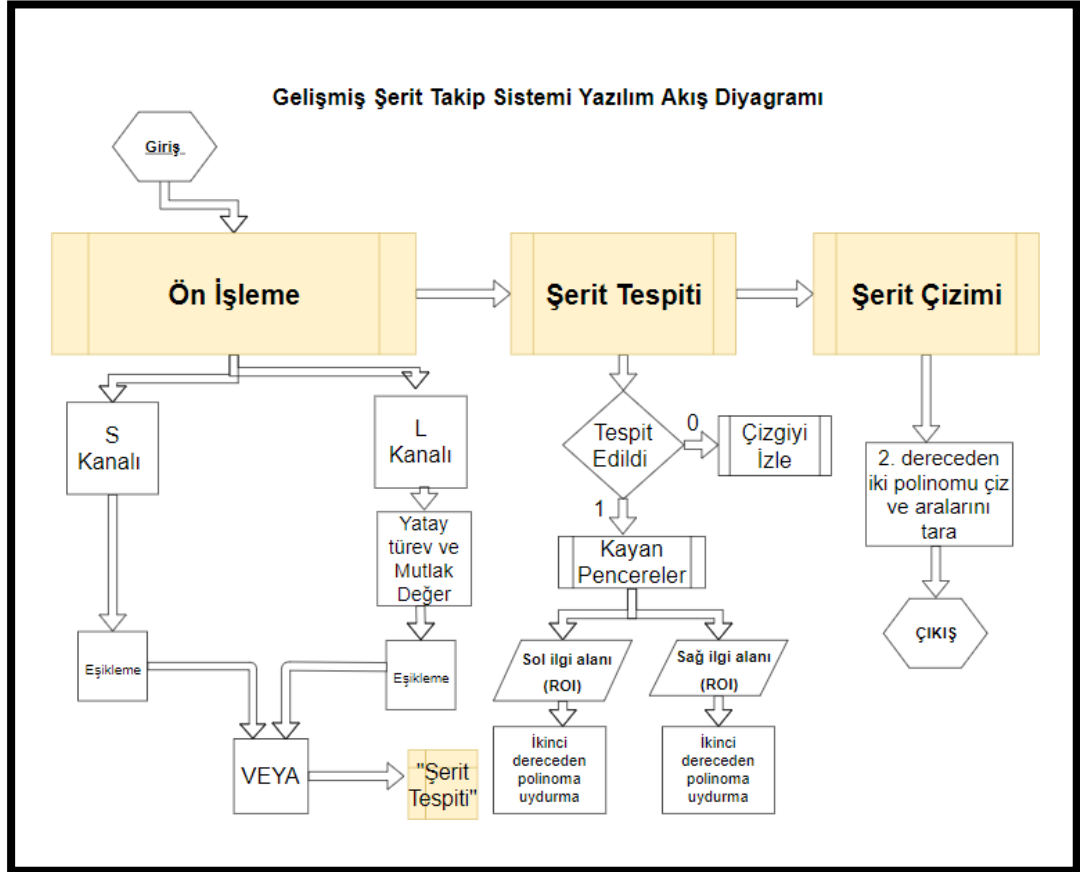
Şekil 2.11 : OpenCV ve xfoencv kütüphanesini bağlantılıma



Şekil 2.12 : OpenCV kütüphanesini bağlantılıma

3. ŞERİT TAKİP ALGORİTMASI YAZILIMI

İleri derece şerit takip algoritması kurmak için, şeride ait sağ ve sol çizgiler 2. dereceden çok terimli bir denklemi ifade eder varsayımı ile öncelikli olarak bu çizgiler bulunmaya çalışılmış daha sonra bu çizgileri ifade eden ikinci dereceden çok terimli fonksiyonun katsayıları bulunmuştur. Bulunan bu iki denklemin arasındaki alan şeridi ifade etmektedir. Şeritin anlamlı olarak görülmesi için bu alan boyanmıştır. Algoritma üç ana başlıktan oluşmaktadır; Ön İşleme, Çizgi Tespiti, Şerit Çizimi. Şekil 3.1’de bu başlıkları özetleyen bir Yazılım Akış Grafiği’ne yer verilmiştir.



Şekil 3.1 : Yazılım akış grafiği

3.1 Ön İşleme

Ön İşleme kısmında, kameradan yada videodan alınan arabanın yolda ilerlerkenki durumunu yansıtan bir görüntü çerçevesinde, çizgiler dışında bulunan diğer bilgilerin silinmesi ve çizgilerin belirgin bir şekilde elde edilmesi amaçlandı.

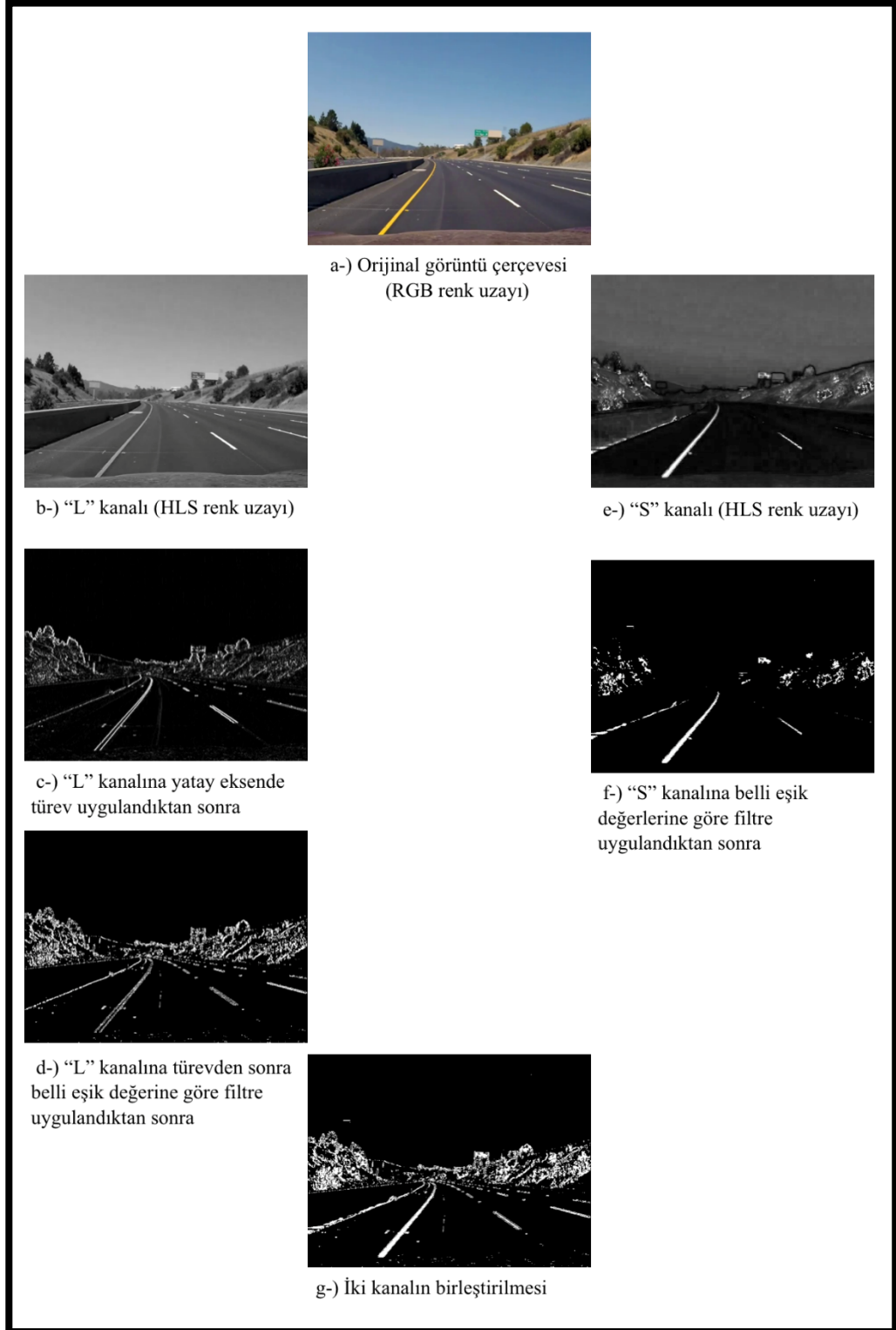
Görüntü RGB renk uzayı formatından Eşitlik 3.1'e göre Ton Doygunluk Açıklık (Hue Saturation Luminance - HSL) formatına çevrilerek renk ve ışığa bağımlılığının düşürülmesi sağlanır.

$$Vmaks = maks(R, G, B), \quad Vmin = min(R, G, B), \quad L = \frac{Vmaks + Vmin}{2}$$
$$S = \begin{cases} \frac{Vmaks - Vmin}{Vmaks + Vmin} & \text{if } L < 0.5 \\ \frac{Vmaks - Vmin}{2 - (Vmaks + Vmin)} & \text{if } L \geq 0.5 \end{cases}, \quad H = \begin{cases} \frac{60(G-B)}{Vmaks - Vmin} & \text{if } Vmaks = R \\ 120 + \frac{60(B-R)}{Vmaks - Vmin} & \text{if } Vmaks = G \\ 240 + \frac{60(R-G)}{Vmaks - Vmin} & \text{if } Vmaks = B \end{cases} \quad (3.1)$$

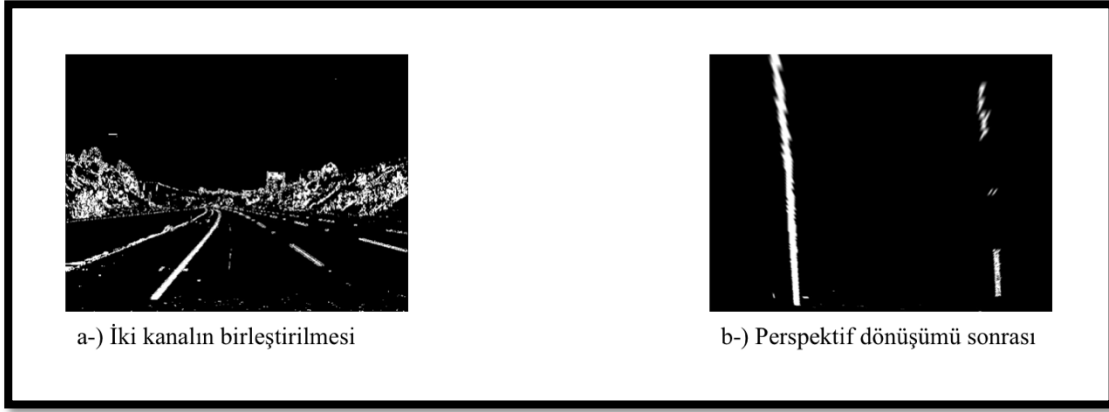
“S” kanalı, belirli eşik değerlerine göre filtrelendir. “L” kanalının ise ilk olarak yatay ekseninde türevi alınır. Görüntü çerçevesinin birinci dereceden türevinin alınmasının anlamı çerçevedeki kenarların bulunmasıdır. Şerit çizgileri görüntü çerçevesinde dikey ekseninde olacağı için yatay ekseninde türev alınması bu çizgilerin yatay eksenindeki başlangıç ve bitiş noktalarının belirginleşmesine ve çerçevede bulunan istenmeyen bilgilerin çoğunun yok edilmesini sağlar. Aracın gittiği yolda bulunan çizgiler kesin hatlarla çizildiği için resmin türevinde çizgilerin kenarı kalacaktır. Türev işlemi sonucu negatif değerler de çıkabileceği için sonucun mutlak değeri alınır. Fakat yatay ekseninde sadece şerite ait çizgiler yoktur. Türev sonrasında, yol ortamında muhtemel değişik boyutlardaki diğer nesnelere ait bulunabilecek kenar bilgileri de olacaktır. Bu bilgilerden kurtulmak için türev işleminden sonra elde edilen sonuç belirli eşik değerlerine göre filtrelendir. “S” ve “L” kanallarına uygulanan bu işlemler sonucunda şerite ait çizgiler hakkında elde edilen bilgilerin artırılması amacıyla bu iki kanal'a ait piksel değerleri “bitsel veya” işlemine tabi tutulur. Bu işlem sonucunda görüntü çerçevesinin piksel değerleri ikili sistem olarak olarak; eğer bilgi varsa 1, yoksa 0 şeklini alır.

Görüntü çerçevesinin her alanı bizim için değerli değildir. Şeriti bulmak için yol'a bakan kısım ile ilgilenilmesi kâfi olacaktır. İlgili alanının yolun olduğu kısma sabitlenmesi için perspektif dönüşümü alınır. Arabadan yol düzlemine paralel şekilde bakan kamera açısı perspektif dönüşümü sayesinde yol düzlemine tepeden bakar. Bu

dönüşüm sırasında sadece belirlenen alana ait bilgiler kalır ayrıca yol düzlemine yukardan bakıldığı için şerit çizgilerinin paralel gözükmesi sağlanır. Şekil 3.2’de ön işleme aşamaları, Şekil 3.3’te perspektif dönüşümü gösterilmiştir.



Şekil 3.2 : Ön işleme aşamaları



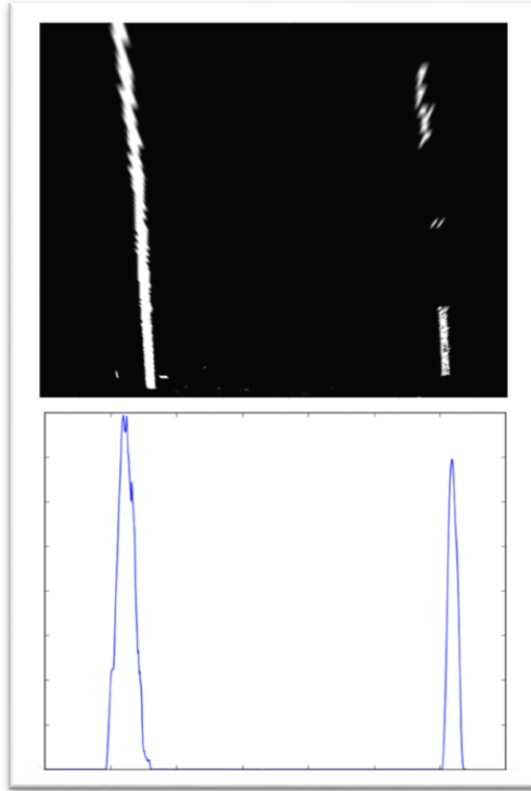
Şekil 3.3 : Perspektif dönüşümü

3.2 Çizgi Tespiti

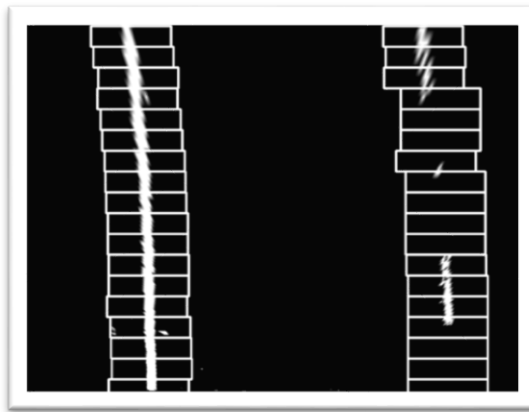
Çizgi tespitinde çizgilerin ikinci dereceden polinomial denklem olarak ifade edilebilmesi varsayımı ile hareket edildi. Bu yüzden şeritin bulunması için öncelikle şerite ait iki çizginin matematiksel olarak ifade edilmesi gereklidir. Ön İşleme fonksiyonunda şerite ait iki çizgi mümkün olduğunca belirgin kılınmaya çalışılmıştır. Şerit Tespiti fonksiyonunda bu belirlenen çizgiler ikinci dereceden polinom denklemler olarak ifade edildi ve ikinci dereceden polinom denkleme ait A, B, C katsayıları bulundu. Çizgi tespiti fonksiyonu daha önceki çerçevelerde şerit bulunup bulunmamasına göre değişim gösterir. Eğer daha önce şerit bulunmuşsa Çizgi takibi fonksiyonu çalışır ki bu fonksiyon daha önce tespit edilen şerite ait çizgilerin olduğu konuma ve belirlenmiş bir değer kadar yanlarına bakar. Eğer bu çizgilerin konumunda ve etrafında yeterince beyaz piksel değeri varsa eski bulunduğu şeritin izinden gider. Eğer yeterince beyaz piksel yoksa yada bir önceki çerçevede şerit bulunmamışsa izleyen pencereler algoritması çalışır.

Geliştirilen izleyen pencereler algoritmasında; çerçeve, y ekseninin ortasından sağ ve sol parçalara ayrılır. Sağ ve Sol parçaların x ekseninde histogramları hesaplanır ve histogramların maksimum noktalarından dikdörtgen pencere başlatılır. Sağ yada sol histogramın maksimum noktası şerite ait çizginin başlangıç noktasını vermektedir (Şekil 3.4'te bu durum gözlenebilir). Çünkü ön işleme fonksiyonu ile çerçevede ilgilendiğimiz alandaki gürültülerden ve çizgi bilgisi haricindeki diğer bilgilerden kurtulmayı amaçladık. Başarılı bir ön işleme fonksiyonu sonrasında alınan histogram ile y eksenini boyunca ilerleyen çizginin x eksenindeki histogram karşılığı maksimum

nokta olacağı varsayılıyor. Daha sonra çizginin y eksenini boyunca konumunu takip etmek için çizilen ilk dikdörtgen pencerenin içinde bulunan beyaz piksellerin orta noktası alınır ve bir sonraki dikdörtgen pencere bu nokta merkezinde ve önceki pencerenin hemen üstüne yerleştirilir. Bu işlem yinelemeli olarak tekrarlanır ve Şekil 3.5'te gösterilmiştir. Çizilen pencerelerin orta noktasına göre ikinci dereceden polinom denklem katsayıları bulunur. Sağ çerçeve parçası ve sol çerçeve parçası için ayrı ayrı bulunan bu katsayılar şerite ait sağ ve sol çizgilerin konumunu bildiren matematiksel ifadeye denktir.



Şekil 3.4 : X Eksenine göre histogram alınması



Şekil 3.5 : İzleyen pencereler

3.3 Şerit Çizimi

Şerit çizimi fonksiyonunda, daha önce çerçevenin sağ parçası ve sol parçasısı için belirlenen katsayılara göre iki adet çizgi çizilir ve bu çizgiler arası boyanır.

Ön işleme fonksiyonu sonrası alınan perspektif transformun ters perspektif dönüşümü alınır. Perspektifi alınmış bu boyanan çerçeve orijinal görüntüye eklenerek şerit çizilir. Çizilen örnek şerite Şekil 3.6'da ulaşılabilir.



Şekil 3.6 : Şerit çizimi

3.4 Kontrol Sınıfı

Kontrol Sınıfında (Class) Çizgi tespitinde bulunan katsayılar bu sınıfa ait bir nesneye eklendi ve son on çerçeveye ait bulunan katsayıların ortalaması tutuldu. Ayrıca bulunan katsayıların şerit belirtip belirtmediğini bildiren bu sınıfa ait method vardır. Bu method'dan dönecek bilgiye göre şerit çizimi için verilecek katsayılar belirlenir. Eğer geçmiş çizgilerden herhangi birisinin (sağ yada sol) mevcut bulunan bir çizgilerle oluşan kombinasyonlarından birisi şerit belirtiyorsa şerit çizimi için o katsayılar gönderilir. Eğer belirtmiyorsa şeritin bulunmadığı belirtilir.

4. DONANIM VE YAZILIM BLOKLARININ AYRILMASI

Şerit takip sisteminin daha hızlı gerçekleşmesi için sisteme ait algoritmanın bir kısmının donanımda gerçekleşmesi düşünülmüştür. Donanım ve yazılım olarak ayrılacak bu sistem için uygun olan Xilinx firmasının ZYNQ-7000 serisine ait Zedboard geliştirme kartı seçilmiştir. Yazılım bloğu Zedboard içinde mevcut ARM Cortex-A9 işlemcisinde; donanım bloğu Zedboard içerisindeki mantık hücreleri kısmında gerçekleştirilecektir.

4.1 Zaman Analizi

Şerit Takip Sistemi yazılımının Macbook Pro Retina (13-inch Late 2012 , 8 GB 1600 MHz DDR3 RAM, 2.5 GHz Intel Core i5) özellikli bilgisayarda 640 * 480 çözünürlüklü otoyol sürüş videosu için ölçülen zaman analizi Çizelge 4.1'deki gibidir:

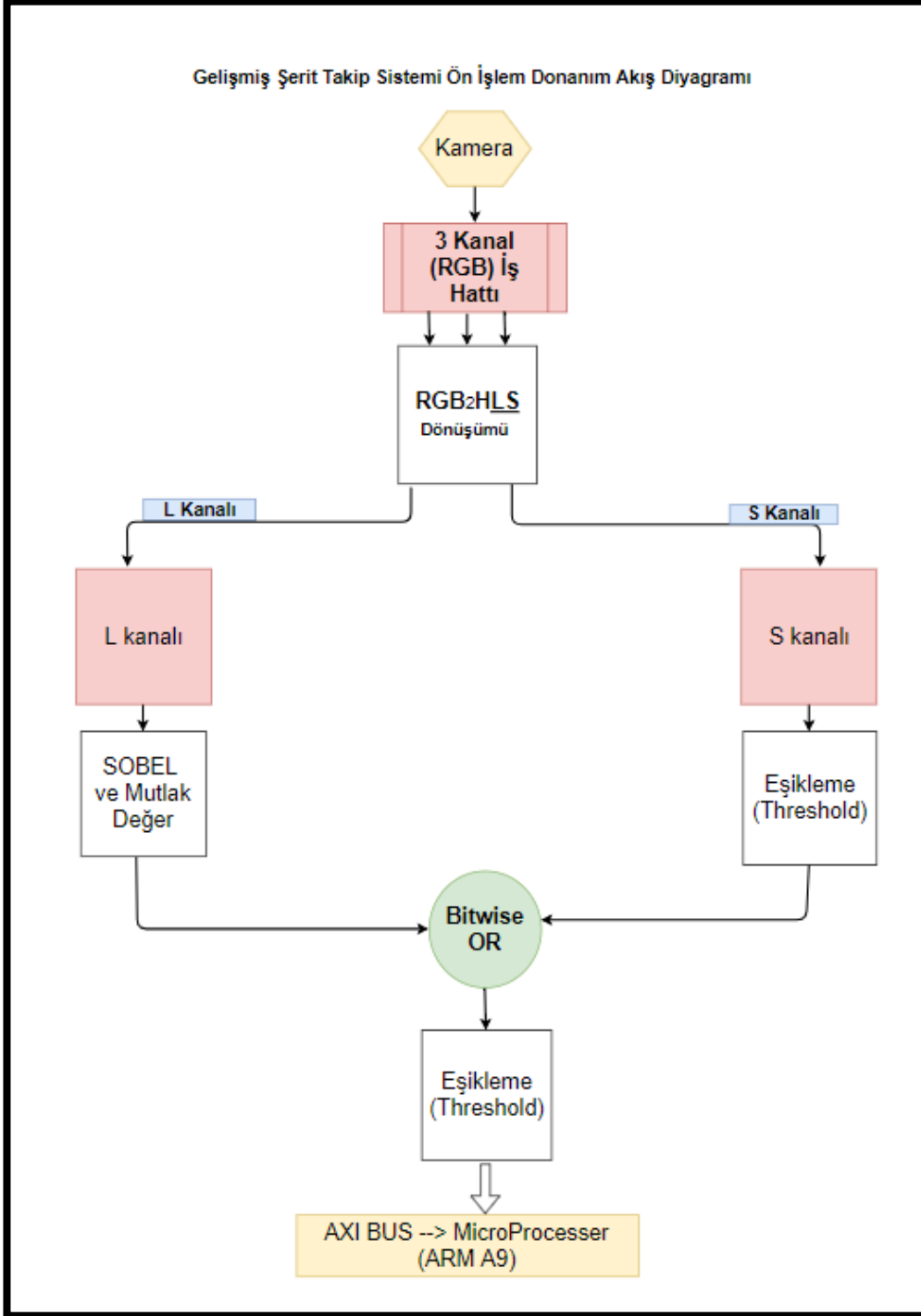
Çizelge 4.1 : Macbook Pro'da şerit takip sistemi zaman analizi.

Fonksiyon	İşlenen Çerçeve Sayısı	En büyük bekleme süresi (Bir çerçevede)	En küçük bekleme süresi (Bir çerçevede)	Ortalama bekleme süresi (Bir çerçevede)
Ön İşleme	1260	81,71 ms	8,21 ms	36,99 ms
Tüm Yazılım	1260	153,31 ms	25,47 ms	77,84 ms
Önişleme/TümYazılım	-	%53.3	%32.2	%47.5

4.2 Donanım Yazılım Blokları

Yapılan zaman analizine göre Şerit Takip Sistemi, donanım – yazılım olarak “ön işleme” fonksiyonunun donanımda gerçekleşmesi diğer fonksiyonların yazılımda gerçekleşmeleri üzerine parçalanmıştır. Ön işleme fonksiyonunun işlemsel yoğunluğundan ve donanımda gerçekleşmeye daha elverişli olduğundan dolayı

donanımda gerçeklenmesine karar verilmiştir. Ön işleme fonksiyonunun donanım akış grafiği Şekil 4.1'deki gibidir. Bu tasarım Vivado ortamında “verilog” donanım tanımla dili ile yapılan projenin tasarımıdır. SDSoC platformu için de “ön işleme” fonksiyonunun donanımda gerçeklenmesi baz alınmıştır.



Şekil 4.1 : Donanım akış grafiği

5. SDSOC PLATFORMU İLE FPGA ÜZERİNDE GERÇEKLEME

“C++” programlama dilinde “OpenCV” kütüphanesi kullanılarak geliştirilen “Şerit Takip Sistemi”nin SDSoc platformu ile gerçekleştirilmesi için Xilinx FPGA’ler için optimize edilmiş “OpenCV” kütüphanesi temel alınarak geliştirilmiş yüksek seviye programlama dilinden (C++) donanım geliştirmeye olanak veren “xfOpenCV” [30] kütüphanesinden yararlanılmıştır.

5.1 XfOpenCV Fonksiyonları

XfOpenCV kütüphanesinde 50’nin üstünde çekirdek (kernel) bulunmaktadır. Çizelge 5.1’de geliştirilen şerit takip algoritmasının ön işleme kısmında kullanılan OpenCV fonksiyonları ve onlara karşılık gelen XfOpenCV kütüphanesindeki fonksiyonlar belirtilmiştir.

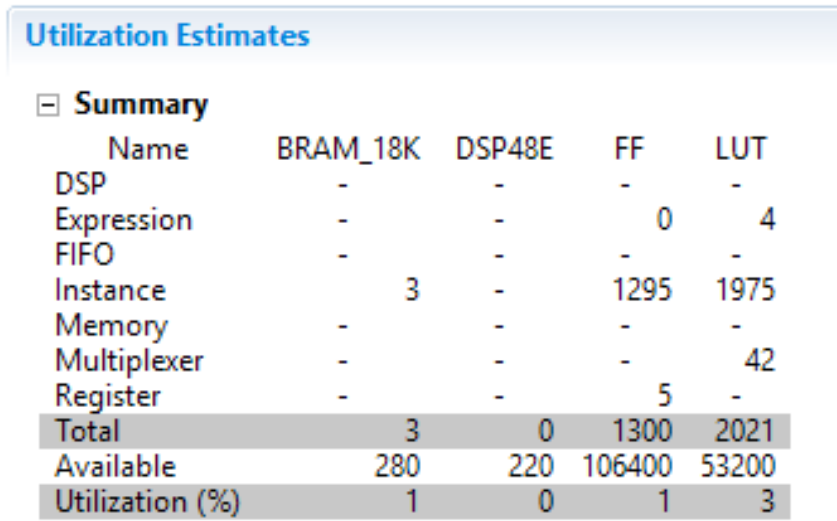
Çizelge 5.1 : XfOpenCV ve OpenCV’de fonksiyon karşılıkları.

Kütüphane Fonksiyon	OpenCV Kütüphanesi	XfOpenCV Kütüphanesi
Türev Alıcı	“cv::Sobel()”	“xf::Sobel()”
Mutlak Değer Alıcı	“cv::abs()”	“xf::absdiff()”
Eşik Değerine göre Filtreleme	“cv::inRange()”	“xf::Threshold()”
Bitsel Veya İşleci	“cv::bitwise_or()”	“xf::bitwise_or()”
Perspektif Dönüşümü	“cv::warpPerspective”	“xf::warpTransform()”

5.2 Alan Kullanımı

Bu bölümde geliştirilen Şerit Takip Sistemi için XfOpcnv kütüphanesinde kullanılacak fonksiyonların Zedboard üzerinde ne kadar donanım alanı kullandığının analizlerine yer verilmiştir.

Türev alıcı “xf::Sobel()” fonksiyonu dönüşümü sonrası Zedboard’da kullanılan alanlar Şekil 5.1’de gösterilmiştir.

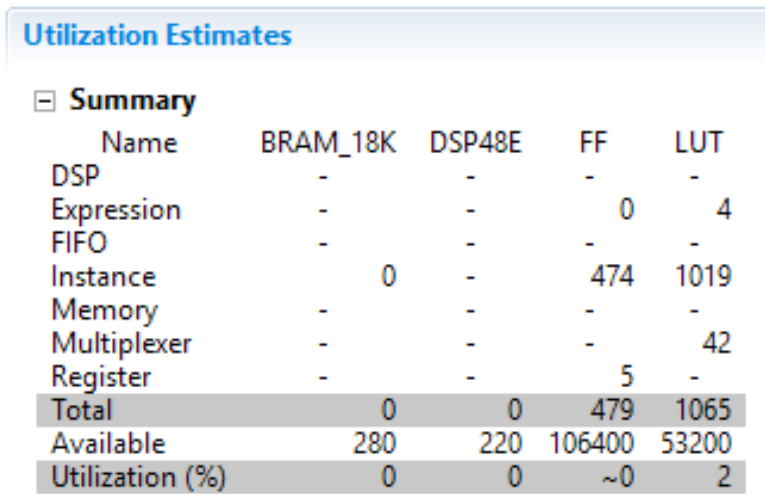


The screenshot shows the 'Utilization Estimates' window for the 'xf::Sobel()' function. It includes a 'Summary' table with the following data:

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	4
FIFO	-	-	-	-
Instance	3	-	1295	1975
Memory	-	-	-	-
Multiplexer	-	-	-	42
Register	-	-	5	-
Total	3	0	1300	2021
Available	280	220	106400	53200
Utilization (%)	1	0	1	3

Şekil 5.1 : xf::Sobel kaynak kullanımı

Mutlak değer alıcı - “xf::absdiff()” fonksiyonu dönüşümü sonrası Zedboard’da kullanılan alanlar Şekil 5.2’de gösterilmiştir.



The screenshot shows the 'Utilization Estimates' window for the 'xf::absdiff()' function. It includes a 'Summary' table with the following data:

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	4
FIFO	-	-	-	-
Instance	0	-	474	1019
Memory	-	-	-	-
Multiplexer	-	-	-	42
Register	-	-	5	-
Total	0	0	479	1065
Available	280	220	106400	53200
Utilization (%)	0	0	~0	2

Şekil 5.2 : xf::absdiff kaynak kullanımı

Eşik Değerine göre Filtreleme - “xf::Threshold()” fonksiyonu dönüşümü sonrası Zedboard’da kullanılan alanlar Şekil 5.3’te gösterilmiştir.

Utilization Estimates				
Summary				
Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	4
FIFO	-	-	-	-
Instance	0	-	475	980
Memory	-	-	-	-
Multiplexer	-	-	-	33
Register	-	-	21	-
Total	0	0	496	1017
Available	280	220	106400	53200
Utilization (%)	0	0	~0	1

Şekil 5.3 : xf::Threshold kaynak kullanımı

Bitsel Veya İşleci - “xf::bitwise_or()” fonksiyonu dönüşümü sonrası Zedboard’da kullanılan alanlar Şekil 5.4’te gösterilmiştir.

Utilization Estimates				
Summary				
Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	4
FIFO	-	-	-	-
Instance	0	-	394	942
Memory	-	-	-	-
Multiplexer	-	-	-	42
Register	-	-	5	-
Total	0	0	399	988
Available	280	220	106400	53200
Utilization (%)	0	0	~0	1

Şekil 5.4 : xf::bitwise_or kaynak kullanımı

Perspektif Dönüşümü - “xf::warpTransform()” fonksiyonu dönüşümü sonrası Zedboard’da kullanılan alanlar Şekil 5.5’te gösterilmiştir.

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	4
FIFO	-	-	-	-
Instance	676	61	43587	28038
Memory	-	-	-	-
Multiplexer	-	-	-	42
Register	-	-	5	-
Total	676	61	43592	28084
Available	280	220	106400	53200
Utilization (%)	241	27	40	52

Şekil 5.5 : Xf:warpTransform kaynak kullanımı

Şekil 5.5'te görüldüğü gibi Zedboard'ta bulunan Block RAM sayısı 280 iken “xf:warpTransform” fonksiyonu 676 Block RAM kullanması gerekmektedir. “Perspektif Dönüşümü” işlemi Zedboard üzerinde donanımda çalışmaya elverişli değildir. Bu nedenle bu işlem işlemcide gerçekleştirilmiştir.

5.3 Zaman Analizi

SDSoC ortamında OpenCV kütüphanesi kullanılarak “Zedboard” FPGA geliştirme kartı üzerinde sadece ARM Cortex A9 üzerinde gerçekleştirilen “Şerit Takip Sistemine” ait zaman analizi Çizelge 5.2 SDSoC Zedboard yazılım zaman analizi çizelgesinde yer almaktadır.

Çizelge 5.2 : SDSoC Zedboard yazılım zaman analizi.

Fonksiyon	İşlenen Çerçeve Sayısı	Bekleme süresi
Ön İşleme	1	157,07 ms
Tüm Yazılım	1	401,14 ms
Önişleme/TümYazılım	-	%39.15

SDSoC ortamında XfOpencv ve OpenCV kütüphaneleri kullanılarak “Zedboard” FPGA geliştirme kartı üzerinde donanım ve yazılım olarak işlemci + donanım üzerinde gerçekleştirilen “Şerit Takip Sistemine” ait zaman analizi Çizelge 5.3’te yer almaktadır.

Çizelge 5.3 : SDSoC Zedboard işlemci + donanım zaman analizi.

Fonksiyon	İşlenen Çerçeve Sayısı	Bekleme süresi (Yazılım)	Bekleme süresi (Donanım)	Bekleme süresi azalma oranı -Performans-
Ön İşleme (perspektif dönüşümsüz)	1	157,07 ms	113,76 ms	%28 azalma
(fps)	1	(6,37 fps)	(8,79 fps)	(1.38 kat performans kazanımı)

SDSoC ve XfOpenCV kütüphanesi kullanılarak önışleme algoritması (perspektif dönüşümsüz) 1.38 kat daha hızlı hale getirilmiştir. Performansı olumsuz etkileyen unsurlar:

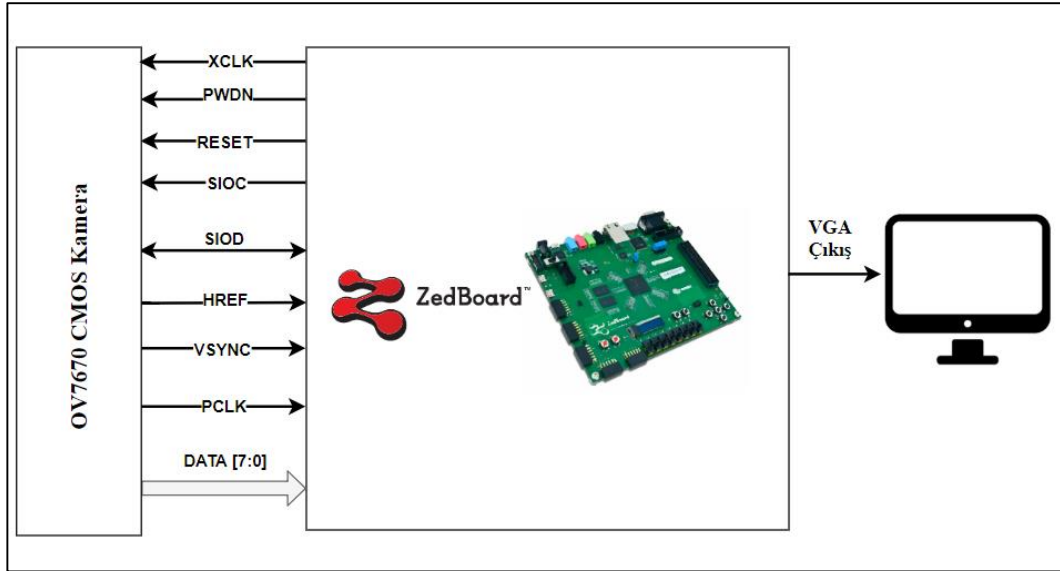
- Görüntünün işlemciden donanıma aktarılması. Zedboard Geliştirme Kartı kullanılmasından dolayı SDSoC platformu ile donanımdan input olarak alınabilecek bir HDMI girdisi bulunmamaktadır. Bu yüzden input işlemciden okunmuş donanıma aktarılmış donanımda işlem bittikten sonra yazılımın devam etmesi için işlemciye tekrar gönderilmiştir.
- Yüksek seviye dillerin esnek olmaması. Donanım tanımlama dillerine göre daha az esnek olan yüksek seviye diller ile yapılan donanım tasarımlarında mutlak optimizasyondan çok uzak kalınmaktadır.
- Zedboard Geliştirme kartında yeterli alan bulunmaması. Zedboard’un donanım kısmındaki kaynakların yetersizliği sebebiyle perspektif dönüşümü donanımda gerçekleştirilememiştir.
- SDSoC platformu ve “XfOpenCV” kütüphanesinin henüz geliştirilmekte olmalarından dolayı kaynaklanan performans düşüklükleri.

Performans hakkında çıkartılan yorumlara yukarıdaki maddelerde yer verilmiştir

6. VIVADO PLATFORMU İLE DONANIM TASARIMI

Şerit takip sisteminin, ön işlem kısmının gerçekleştirilmesi için FPGA üzerinde donanım tasarımı yapıldı. Tasarım, Vivado ortamında Verilog donanım tanımlama dili kullanılarak gerçekleştirildi. Tasarlanan alt modüller, bir tepe modül altında toplanmıştır. Tepe modülün altında yedi tane alt modül bulunmaktadır. Bu modüllerin çalışma mekanizması detaylı olarak açıklanacaktır.

Tasarımın ilk aşaması şerit görüntülerinin alınmasıdır. Bunun için OmniVision firmasının OV7670 CMOS kamera modülü [31] kullanılmıştır. Bu kamera, yurtiçi ve yurtdışı piyasasından uygun fiyata rahatlıkla temin edilebilmektedir. Bu sayede bir çok gömülü sistem projesinde kullanılmıştır. Kameradan alınan gerçek zamanlı video verileri, FPGA’de ön işlem aşamalarından geçtikten sonra sonucun gözlemlenmesi için VGA aracılığıyla monitöre yansıtılmıştır. Tasarımın genel şematığı Şekil 6.1’de gösterilmiştir.



Şekil 6.1 : Tasarımın genel şematığı

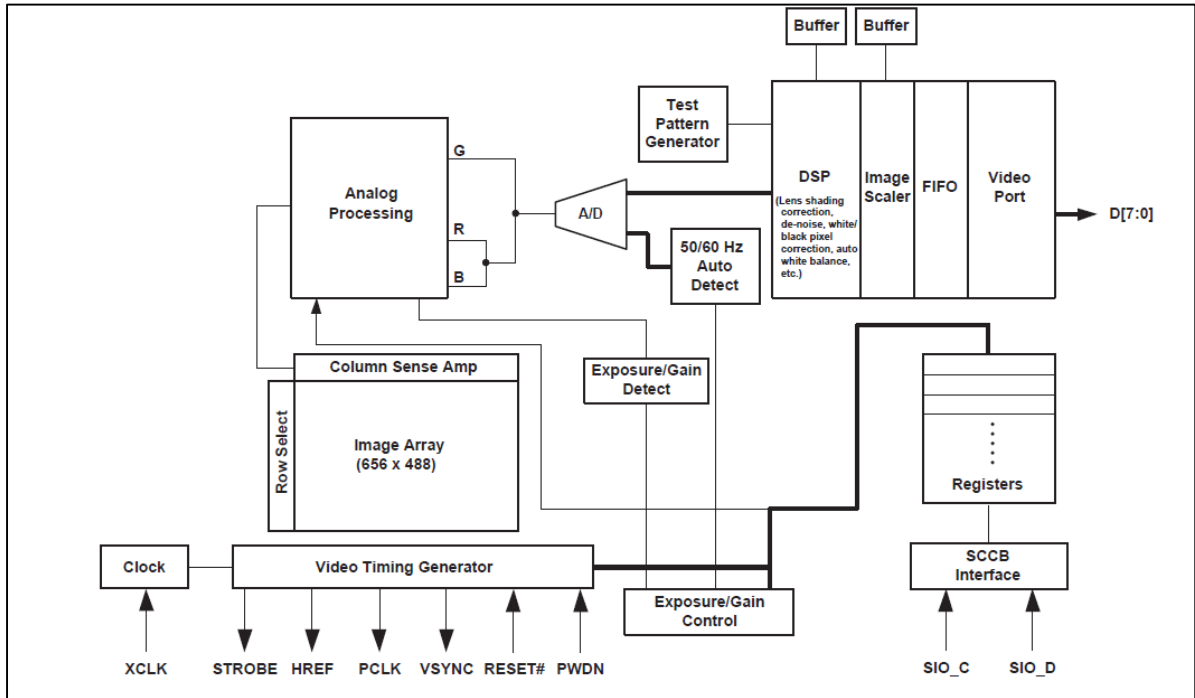
6.1 Kamera Modülü

6.1.1 Kamera bilgileri

OV7670 kamera modülü, maximum 30 Fps (saniye başına çerçeve sayısı) ile 640*480 boyutlarında görüntü verebilmektedir. Görüntü sensörü tarafından yakalanan çerçeve, kamera devresi üzerinde bulunan Sayısal İşaret İşleme (Digital Signal Processing – DSP) birimi tarafından bir takım temel ön işlemlerden geçer ve daha sonra dış birimlere gönderilir. Yine modül üzerinde bulunan analog işlemci sayesinde pozlama, kazanç, kontrast ayarları yapılabilir. Kamera devresi Şekil 6.2’de, blok şeması Şekil 6.3’te gösterilmiştir.



Şekil 6.2 : OV7670 kamera Modülü



Şekil 6.3 : Kamera modülü blok şeması

Ayrıca kullanıcı, kameranın kontrol tutucularına gerekli değerleri göndererek, kameranın çalışma modlarından çeşitli değişiklikler yapabilir, farklı ön işlemlerden geçirebilir. Bunun için Seri Kamera Kontrol Arayüzü (Serial Camera Control Interface - SCCB) haberleşme protokolü [32] kullanılır. Bu protokol bir ana-uydu haberleşme protokolüdür ve elektronik projelerinde sıkça kullanılan Tümlşik Devreler Arası (Inter Integrated Circuits – I²C) protokolüyle oldukça benzer bir yapıya sahiptir.

Kamera RGB, YUV 4:2:2 ve YCbCr 4:2:2 [31] formatlarında görüntü verebilmektedir. RGB formatı için RGB 565/555/444 seçenekleri mevcuttur. Bu formatlar arasında kırmızı,yeşil ve mavi değerlerinin ifade edildiği bit sayısı türünden farklılıklar vardır. Bu projede RGB 444 formatı kullanılmıştır.

6.1.2 Kamera devresi giriş çıkışları

Kamera modülünün üstünde 18 adet pin bulunmaktadır. Bu pinlerin listesi Çizelge 6.1’de gösterilmiştir.

Çizelge 6.1 : Kamera giriş çıkışları.

Pin	Türü	İşlevi
VDD	Supply	Güç Pini
GND	Supply	Toprak Pini
SIOC	Giriş	SCCB Saat Pini
SIOD	Giriş/Çıkış	SCCB Veri Pini
VSYNC	Çıkış	Dikey Senkronizasyon
HREF	Çıkış	Yatay Senkronizasyon
PCLK	Çıkış	Piksel Saati
XCLK	Giriş	Sistem Saati
D0-D7	Çıkış	Video Paralel Çıkış Portu
RESET	Giriş	Reset
PWDN	Giriş	Power Down

VDD: Kamera modülünün besleme pinidir. Kamera modülü 3.3 Volt besleme gerilimi ile çalışmaktadır.

GND: Kamera modülü toprak pinidir.

SIOC: Seri haberleşme için gerekli saat pinidir. Bu hat SCCB haberleşmesindeki veri senkronizasyonunu sağlar.

SIOD: Seri veri pinidir. Bu hat SCCB haberleşmesinde seri veri alışverişinde kullanılır. Gönderilen veri ve alınan veri için aynı hat kullanılır.

VSYN: Dikey senkronizasyon pinidir. Bir görüntü çerçevesinin tam olarak alınmasını ve sonraki görüntü çerçevesine geçilmesini sağlar.

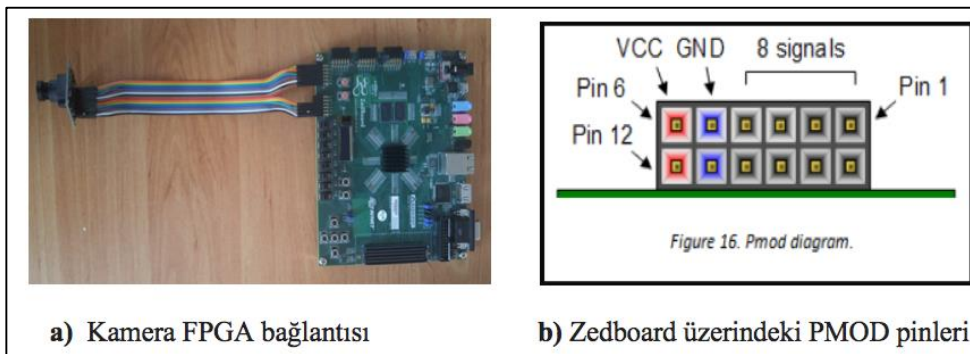
HREF: Yatay senkronizasyon pinidir. Görüntü çerçevesinin bir satırının tam olarak alındıktan sonra bir sonraki satıra geçmesini kontrol eder.

XCLK: Kameranın sistem saat girişidir. Saat kaynağı ana cihaz veya harici bir saat üretici üzerinden sağlanabilir. Kamera veri kitapçığına göre, bu frekans 10 Mhz ile 48 Mhz arasında olmalıdır.

PCLK: Piksel saat çıkışıdır. Piksel verileri bu saat işareti ile senkronize şekilde iletilir. Normal durumda XLCLK işareti ile aynı frekansa sahiptir. İstenildiği takdirde kontrol tutucularına gerekli değerler gönderilerek sistem saat işaretine göre yeniden ölçeklenebilir.

D0-D7: Piksel verilerinin iletiği paralel veri çıkışıdır. Sekiz ayrıık pinden oluşur.

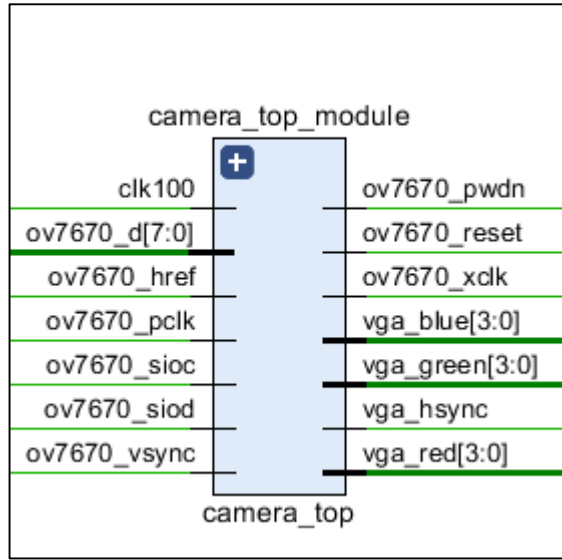
Kamera için gerekli olan sistem saati, besleme gerilimi ve diğer bağlantılar FPGA üzerindeki genel maksatlı PMOD pinleri üzerinden sağlanmıştır. FPGA kamera bağlantısı Şekil 6.4'te gösterilmiştir.



Şekil 6.4 : FPGA-kamera bağlantısı

Kameradan alınan görüntü verileri ilk olarak RGB renk uzayından HLS renk uzayına dönüştürüldü, L ve S adında iki kanala ayrıldı. Daha sonra L kanalındaki görüntü sobel filtresinden geçirilerek eşikleme uygulandı. Aynı şekilde S kanalına da eşikleme uygulandıktan sonra bu iki kanala “bitwise or (bitsel veya)” işlemi uygulandı ve görüntüler birleştirildi. Elde edilen piksel verileri blok RAM’e gönderildi ve VGA aracılığıyla ekrana yansıtıldı.

FPGA üzerinde tasarlanan her işlem bloğu iş hattı (pipeline) [33] sistemine uygun şekilde tasarlandı. Bu şekilde her bir saat periyodunda işlenmiş bir piksel verisi elde edilmiş oldu. Yukarıda anlatılan işlem blokları kameradan gelen PCLK sinyali ile senkronize şekilde çalışmaktadır.



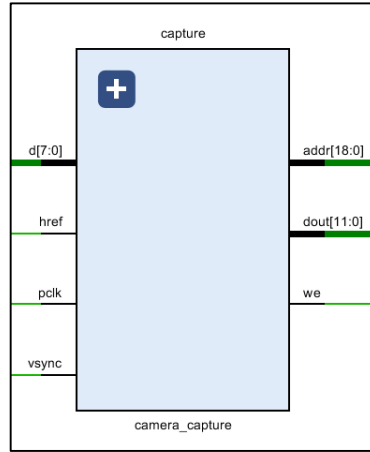
Şekil 6.5 : FPGA üzerinde gerçekleştirilen sistemin RTL şematiği

6.2 Kameradan Görüntü Alınması

FPGA ile kamera arasında iletişimi sağlayan iki adet alt modül bulunmaktadır. İlki kameradan piksel verilerinin alınmasını sağlayan “capture” modülü, ikincisi kamera ile SCCB haberleşmesini sağlayan “controller” modülüdür. Capture modülünün dört adet giriş üç adet çıkış pini bulunmaktadır. Modül kameradan gelen PCLK (Pixel Clock) sinyali ile senkronize şekilde çalışır.

6.2.1 Capture modülü

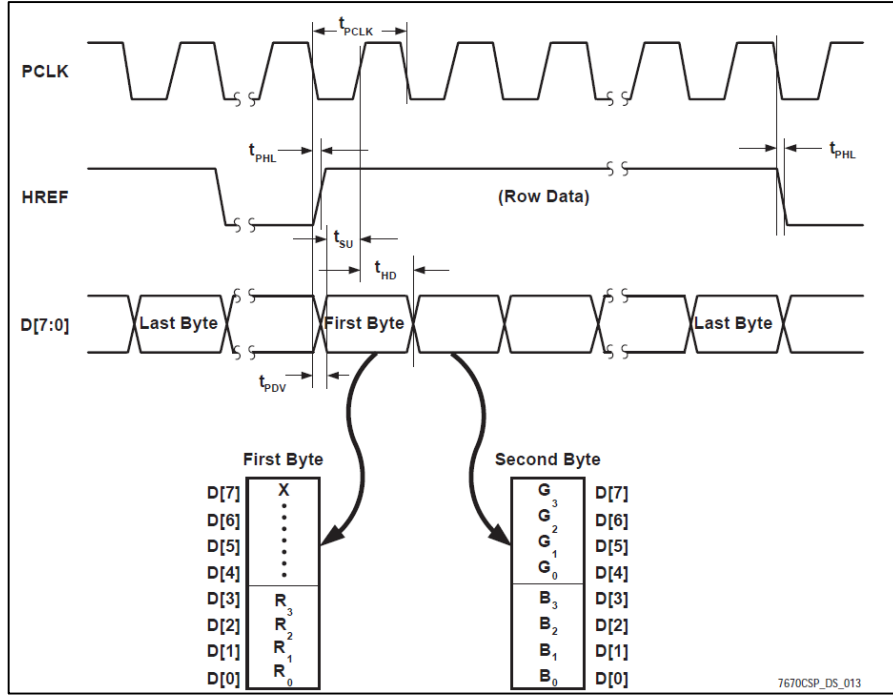
Capture modülü, kameradan gelen piksel verilerini yöneterek, görüntü verilerinin doğru bir şekilde diğer modüllere iletilmesini sağlar. RGB444 formatında kırmızı, yeşil ve mavi renklerinin her biri 4 bit ile ifade edilir. Yani bir piksel verisi 12 bit'den oluşmaktadır. Kameranın D[7:0] veri çıkışı ise her saat darbesinde 8 bitlik veri yollayabilmektedir. Capture bloğu bir piksel verisinin tam olarak elde edilebilmesi için



Şekil 6.6 : Capture Modülü

bir senkronizasyon görevi görür. 8 bit olarak aldığı piksel verilerini 12 bit olarak “dout” çıkışında diğer bloğa gönderir. Şekil 6.6’da capture modülü görülebilir.

Şekil 6.7’de kameranın RGB 444 formatında veri gönderdiği durum için sinyal zamanlaması gösterilmektedir. Kamerada çerçevenin ne zaman başladığını ne zaman bittiğini gösteren HREF ve VSYNC sinyalleri bulunur. HREF sinyali lojik 1 seviyesine yükseldiğinde piksel verileri alınmaya başlar ve o satırın son piksel verisi alınana kadar bu seviyede kalır. Her PCLK periyodunda 8 bitlik piksel verisi gönderilir. Son piksel verisi de alındıktan sonra HREF lojik 0 seviyesine düşer ve kısa bir süre bu konumda kalır. Bu süre zarfında piksel verisi gönderilmez. Daha sonra tekrar lojik 1 seviyesine yükselir ve sonraki satır bilgileri gönderilir. Bu işlem 480 satırın tamamı gönderilene kadar devam eder. Bir çerçevedeki ilk piksel verileri gönderilmeye başlandığında VSYNC dikey senkronizasyon sinyali lojik 0 seviyesine düşer ve son satırın son piksel verisi gelene kadar bu seviyede kalır. Son piksel verisi de alındıktan sonra lojik 1 seviyesine çıkar ve bir müddet bu seviyeyi korur. Bu bir görüntü çerçevesinin tam olarak alındığını gösterir. Daha sonra tekrar lojik 0 seviyesine düşer ve bir sonraki görüntü çerçevesinin piksel verileri alınmaya başlanır.



Şekil 6.7 : RGB444 formatı için sinyal diyagramı

8'er bit olarak gönderilen piksel verilerinin 12 bitlik olarak iletilmesi için "capture" modülü içinde bir zamanlayıcı yapısı kullanıldı. Bu yapı sayesinde bir pikselin kırmızı, yeşil ve mavi verilerinin bir araya gelip 12 biti tamamladıktan sonra iletilmesi sağlandı. Modül ile birlikte her 3 saat periyodunda 2 tam piksel verisi iletilmiş oldu. Bu yapı "we" (write enable) sinyali ile kontrol edilir, eğer we sinyali lojik 1 durumundaysa yakalanan çerçeve, adres bilgisi ile beraber iletilir. We sinyali lojik 0 ise adres ve piksel verisi iletilmez, bir sonraki piksel bilgisinin gelmesi beklenir ve veriler küçük bir FIFO'ya kaydedilir. Ayrıca we sinyali HREF ve VSYNC sinyallerinin durumlarına da bağlıdır. HREF sinyalinin lojik 0 olduğu, yani bir satırın sonuna geldiği durumda veya VSYNC sinyalinin lojik 1 olduğu durumda we sinyali lojik 0 seviyesindedir.

Adres bilgisi, ileride kullanılacak hafıza bloğu (block RAM) için gereklidir. Hangi piksel bilgisinin hafıza bloğunun hangi kısmına yazılacağını bildirmek için 19 bitlik adres bilgisi kullanıldı ve adres bilgisi piksel bilgisiyle aynı saat periyodu ile iletilir. Senkronizasyonun sağlanması ile birlikte 19 bitlik adres verisi ile 12 bitlik piksel verisi bir sonraki modüle iletilir.

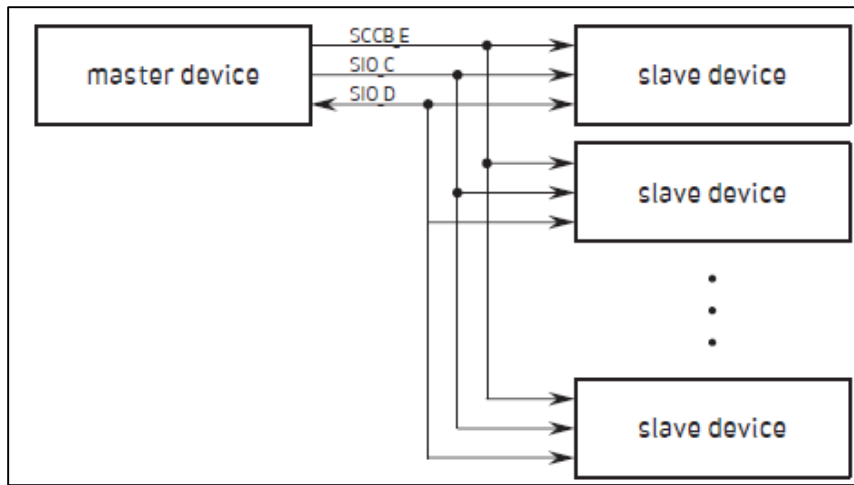
6.2.2 Kamera kontrol modülü

Kameradan görüntü alınması ve haberleşmesinin sağlanması için bir seri haberleşme modülü tasarlandı. Bu modülde SCCB seri haberleşme protokolü kullanılacaktır. Bu protokol kamera modülünün kendi özgü bir protokolüdür ve I2C protokolüne çok benzer bir yapıya sahiptir. Ana (master) – uydu (slave) ilişkisi I2C’de olduğu gibi bu protokolde de mevcuttur.

6.2.2.1 SCCB protokolü

SCCB protokolü OmniVision firmasının ürettiği sensörler için kullanılan bir seri haberleşme protokolüdür [32]. Bu protokol sayesinde sensörler ile diğer dış birimler arasında seri haberleşme sağlanır. Projede bu protokolün uygulanması için firmanın yayınladığı veri kitapçığındaki bilgilerden faydalandı.

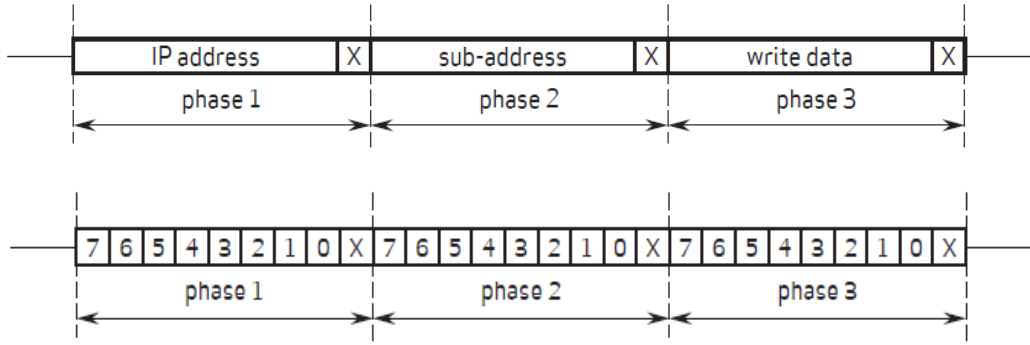
Bu protokolde haberleşme, SIOC ve SIOD adı verilen iki hat ile sağlanır. SIOD veri hattı, SIOC ise saat hattıdır. Haberleşmenin blok diyagramı Şekil 6.8’de gösterilmiştir.



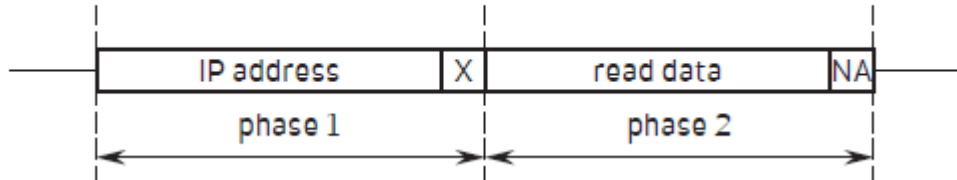
Şekil 6.8 : SCCB haberleşmesi blok diyagramı

Saat işareti olan SIOC, ana cihaz tarafından üretilir ve uyduya gönderilir. Veri hattı ise uyduya veri yazma veya uydudan veri okuma şeklinde çift yönlü olarak çalışır. Haberleşmeyi başlatan ve bitiren ana cihazdır. Saat hattı ve veri hattı pull-up dirençleri ile besleme gerilimine çekilmektedir. Bu yüzden haberleşme olmadığı durumda iki hat da lojik-1 seviyesindedir.

SCCB haberleşmesinde, yazma işlemi 3 baytlık bir yapıdan oluşur [32]. İlk bayt uydu cihazın kimliğini, ikinci bayt uydu cihazda erişilmek ve değeri değiştirilmek istenen tutucunun adresi, son bayt ise yazılmak istenen veriyi temsil eder. Okuma işlemi ise 2 baytlık yapıdan oluşur. İlk bayt uydu cihazın kimliği, ikinci bayt ise uydu cihazda okuma yapılacak tutucunun adresidir. Her gönderilen baytın sonun 1 bitlik kontrol biti bulunur ve bu bit önemsizdir. Kullanılan kameranın adresi veri kitapçığında yazma için 43h, okuma için 42h olarak verilmiştir. Şekil 6.9 ve 6.10’da okuma ve yazma işlemleri için kullanılan baytlar gösterilmektedir.

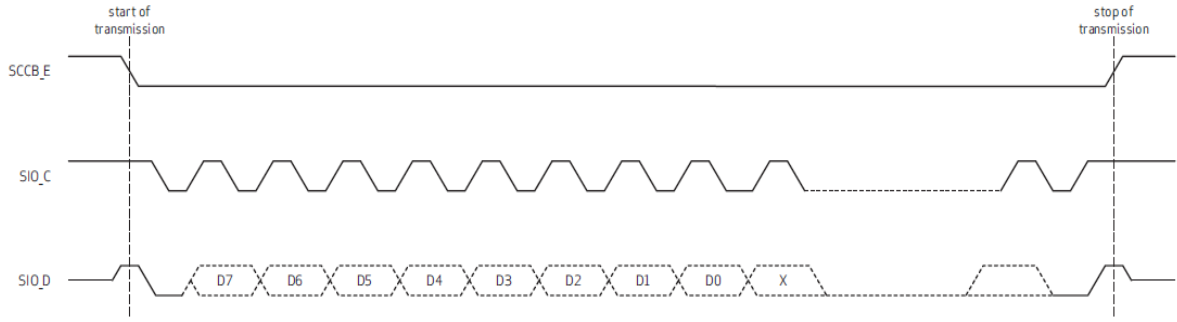


Şekil 6.9 : SCCB yazma işlemi



Şekil 6.10 : SCCB okuma işlemi

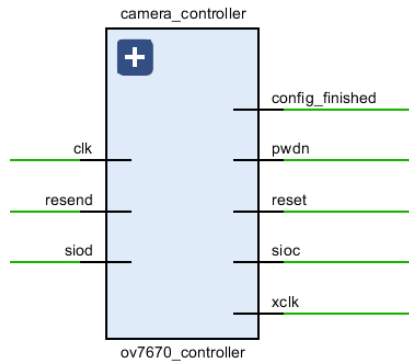
SCCB haberleşmesinde iletim başlamadan önce SIOC ve SIOD sinyalleri lojik 1 durumundadır. İletimin başlaması için ilk olarak SIOD sinyali sonra SIOC saat sinyali lojik 0’a çekilir ve SIOC saat sinyalinin yükselen kenarıyla beraber ilk baytın ilk biti gönderilir. İletim biterken ise üçüncü baytın en anlamsız biti gönderildikten sonra SIOD sinyali lojik 0 yapılır. SIOC sinyali lojik 1 yapıldıktan sonra SIOD veri sinyalinin tekrar lojik 1 yapılmasıyla iletim sonlandırılmış olur. Haberleşmenin sinyal diyagramı Şekil 6.11’de gösterilmiştir.



Şekil 6.11 : SCCB haberleşmesi sinyal diyagramı

6.2.2.2 Modülün tasarımı

Tasarlanan “camera_controller” bloğu ile kamera ile seri haberleşme sağlanır. Bu modül “SCCB_sender” ve “camera_register” adında iki alt modülden oluşur. Modülün yapısı Şekil 6.12’de gösterilmiştir.



Şekil 6.12 : Camera_controller modülü

Camera_register modülü kameranın kontrol tutucularına yazılması gereken veriler ve bu tutucuların adreslerinin yer aldığı modüldür. Kameranın veri kitapçığında tutucuların adres bilgileri ve bu adresler için çalışma modları yer alır. Bu modların bir kısmı Şekil 6.13’deki gibidir. Kontrol tutucularına yazılacak veriler ile, kameradan gönderilecek görüntünün formatı, ışık değeri, boyutları, PCLK saat işaretinin frekansı vb. bir çok özellik kontrol edilir. Tutuculara yazılacak değerler 16 bitlik “command” çıkışı ile SCCB_sender modülünü gönderilir.

Address (Hex)	Register Name	Default (Hex)	R/W	Description
00	GAIN	00	RW	AGC – Gain control gain setting Bit[7:0]: AGC[7:0] (see VREF[7:6] (0x03) for AGC[9:8]) • Range: [00] to [FF]
01	BLUE	80	RW	AWB – Blue channel gain setting • Range: [00] to [FF]
02	RED	80	RW	AWB – Red channel gain setting • Range: [00] to [FF]
03	VREF	00	RW	Vertical Frame Control Bit[7:6]: AGC[9:8] (see GAIN[7:0] (0x00) for AGC[7:0]) Bit[5:4]: Reserved Bit[3:2]: VREF end low 2 bits (high 8 bits at VSTOP[7:0]) Bit[1:0]: VREF start low 2 bits (high 8 bits at VSTRT[7:0])
04	COM1	00	RW	Common Control 1 Bit[7]: Reserved Bit[6]: CCIR656 format 0: Disable 1: Enable Bit[5:2]: Reserved Bit[1:0]: AEC low 2 LSB (see registers AECHH for AEC[15:10] and AECH for AEC[9:2])
05	BAVE	00	RW	U/B Average Level

Şekil 6.13 : Kamera kontrol tutucularının bir kısmı

SCCB_sender modülü yukarıda anlatılan SCCB haberleşme protokolünün gerçekleştiği modüldür. SIOC ve SIOD hatları bu modül üzerinden kameraya gönderilir.

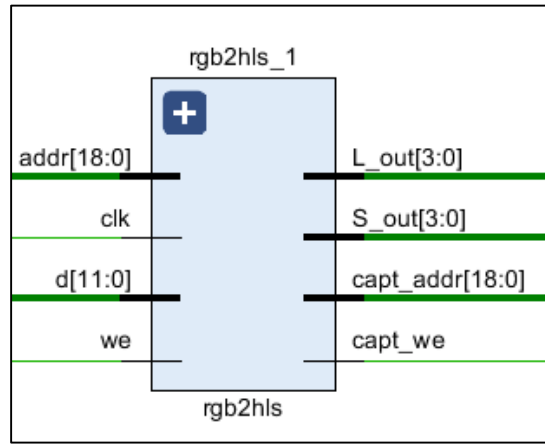
6.3 RGB2HLS Modülü

RGB2HLS modülü, görüntüyü RGB renk uzayı formatından HSL formatına çevrilerek renk ve ışığa bağımlılığının düşürülmesi sağlar. HLS uzayı 3 farklı kanaldan oluşur. Şerit takip sistemi için L ve S kanalları kullanılacaktır. Renk dönüşümü için gerekli denklemler Şekil 6.14’te gösterilmektedir.

$$\begin{array}{l}
 R=R/255 \\
 G=G/255 \\
 B=B/255
 \end{array}
 \quad
 \begin{array}{l}
 \mathbf{max} = \max(R,G,B) \\
 \mathbf{min} = \min(R,G,B)
 \end{array}
 \quad
 L = \frac{\mathbf{max} + \mathbf{min}}{2}
 \quad
 \begin{array}{l}
 \text{if } L < 0.5, S = \frac{\mathbf{max} - \mathbf{min}}{\mathbf{max} + \mathbf{min}} \\
 \text{if } L > 0.5, S = \frac{\mathbf{max} - \mathbf{min}}{2 - \mathbf{max} - \mathbf{min}}
 \end{array}$$

Şekil 6.14 : RGB – HLS renk uzayı dönüşümü [34]

Capture modülünden 12 bit olarak gelen veri 4 bitlik parçalara ayrılır. Bunlar görüntünün kırmızı, yeşil ve mavi bileşenlerine karşılık düşmektedir. Daha sonra tasarlanan karşılaştırıcı yapısıyla bu üç kanalın, en büyük ve en küçük sayısal değere sahip olanları belirlenir. L kanalı elde edilen en büyük ve en küçük değer toplamının ikiye bölünmesi ile bulunur. S kanalını hesabı L kanalının değerine bağlıdır. Kameradan gelen piksel verileri 4 bit ile ifade edilir, yani bir pikselin alabileceği en büyük sayısal değer 15 olacaktır. Bunun için burdaki koşul L'nin piksel değerinin değerinin 8'den büyük veya küçük olması durumudur. S kanalının hesaplanması için bir "look up table" oluşturulmuştur. L ve S değerleri hesaplandıktan sonra piksel değerleri tutuculara kaydedilmiştir. Bu değerler "L_out" ve "S_out" çıkışlarıyla bir sonraki modüle gönderilirler. Bu sayede seri olarak gelen piksel verileri iki kanaldan paralel olarak iletilir. Ayrıca capture modülünden gelen "we" ve adres bilgisini tutan "cap_addr" de tutuculara kaydedilir ve bu 4 sinyal aynı saat işaretiyle tetiklenecek şekilde bir sonraki modüle gönderilir. Bu şekilde, iş hattı yapısına uygun şekilde veriler iletilir. Bu modülde kullanılan saat işareti PCLK sinyalidir. Modülün görünümü Şekil 6.15'teki gibidir.

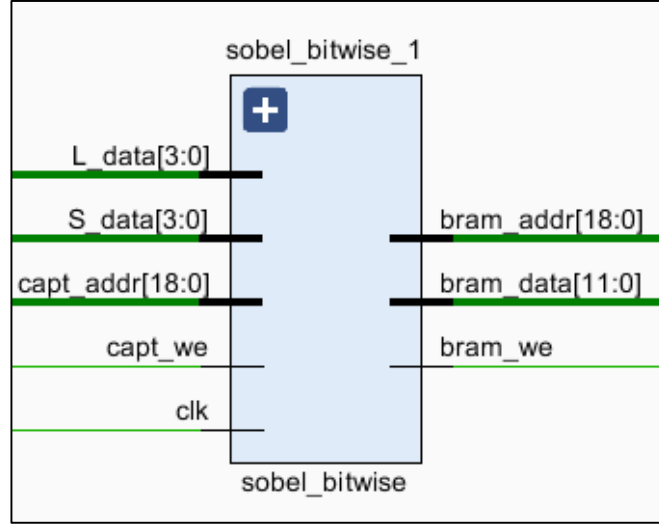


Şekil 6.15 : RGB2HLS modülü

6.4 Sobel ve Bitisel Veya Modülü

Bu modülde birden fazla işlem gerçekleştirilmiştir. İlk olarak görüntünün L kanalına sobel filtresi uygulanmıştır. Filtreden geçen görüntünün mutlak değeri alınmış, daha sonra eşikleme işlemi uygulanmıştır. İşlenen piksel verileri S kanalı piksel verileri ile "bitisel veya (bitwise or)" işlemi uygulanarak birleştirilmiştir.

Sobel modülünün 5 adet girişi 3 adet çıkışı bulunmaktadır. RGB2HLS bloğunun çıkışından L ve S kanalına ait 4'er bitlik piksel verileri gelir. Aynı zamanda capt_addr, capt_we,clk girişleri mevcuttur. Çıkışında ise bram_addr,bram_data ve bram_we sinyalleri bulunur. Modülün yapısı Şekil 6.16'daki gibidir.



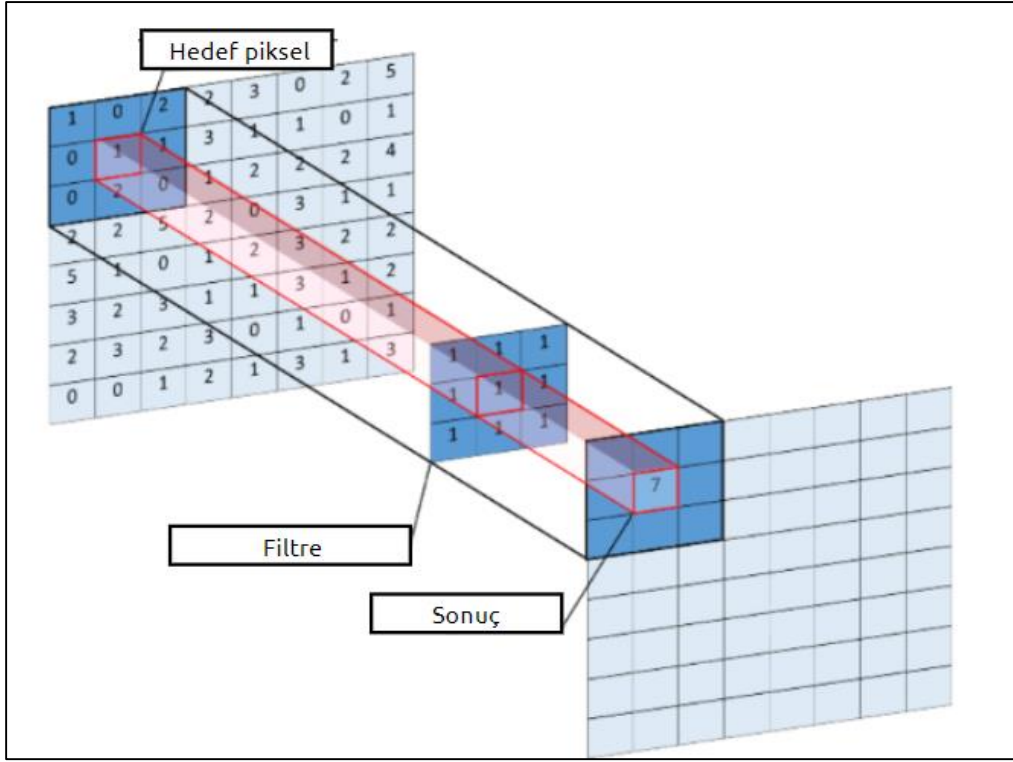
Şekil 6.16 : Sobel_bitwise modülü

Sobel filtresi görüntü işleme uygulamalarında sıklıkla kullanılan bir filtredir. Filtre renk değişiminin yoğun olduğu yerleri belirginleştirir. Kenar tespit uygulamalarında kullanılır.

Sobel modülünde, görüntünün L kanalına yatay türev uygulanır. Bunun için Şekil 6.17'deki yatay sobel filtresi uygulanacaktır. Görüntü filtreleme iki boyutlu konvolüsyon işlemidir. Konvolüsyon işlemi filtre matrisinin, görüntü üzerinde dolaştırılması anlamına gelmektedir. Görüntü ile filtrenin üst üste geldiği noktadaki katsayılar çarpılır. Daha sonra bu çarpım değerleri toplanarak merkezdeki piksele yeni sonuç yazılır.

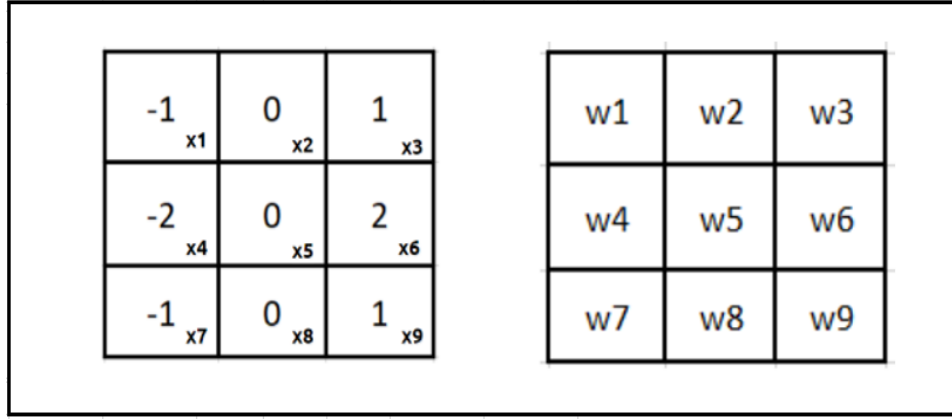
$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I$$

Şekil 6.17 : Yatay sobel filtresi

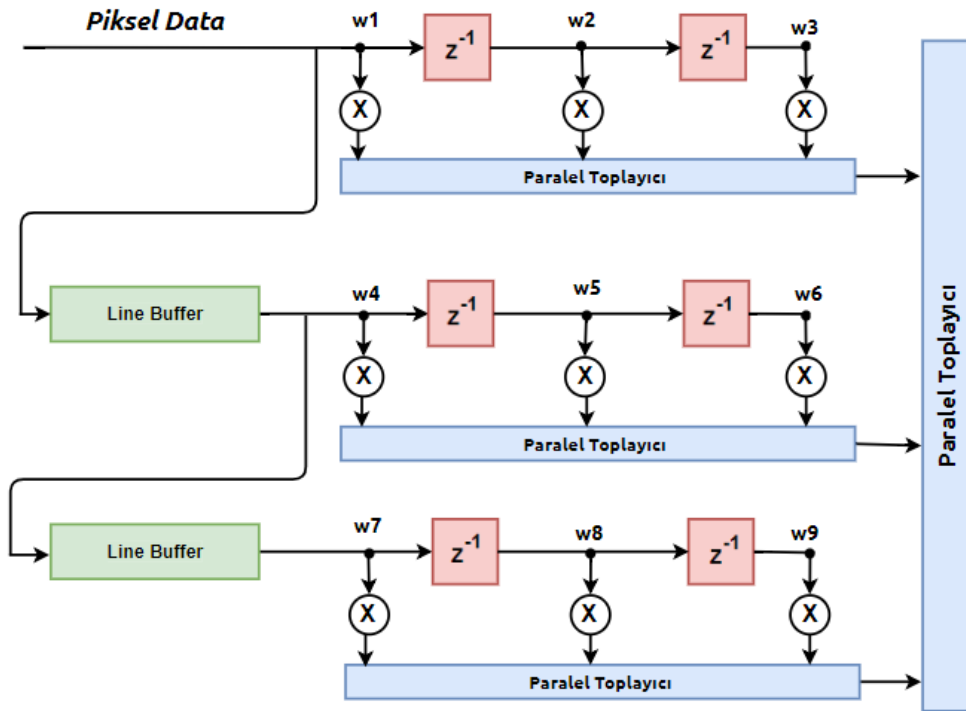


Şekil 6.18 : Filtreleme işlemi

Kameradan gelen bir tam görüntü çerçevesi 640*480 boyutlarındadır. Yani her çerçevede toplam 307.200 piksel bulunmaktadır. Kameradan sürekli olarak piksel verileri gönderilmektedir ve elimizde her an bütün bir çerçevenin görüntüsü bulunmamaktadır. Bu durum filtrenin bütün resmin üzerinde kaydırılarak uygulanmasında önemli bir problem oluşturur. Bu problemi çözmek için “line buffer [35]” yapısı kullanılır. Bu yapıda, gelen piksel verileri line buffer bloklarında bir satır uzunluğu kadar geciktirilir. Line buffer blokları dolduktan sonra, filtre resim üzerinde her saat darbesiyle beraber kayarak ilerler. Bu şekilde filtre, bütün piksellerin üzerinden doğru bir şekilde dolaşmış olur. Kullanılacak filtre boyutuna göre line buffer sayısında değişir. Biz 3x3'lük bir filtre matrisi kullanacağız ve bunun için iki adet line buffer bloğu yeterli olacaktır. Line buffer yapısının çalışma mekanizması ve filtrenin uygulanacağı piksel bölgeleri Şekil 6.19'da ve Şekil 6.20'de gösterilmiştir.



Şekil 6.19 : Filtre çekirdeği ve uygulanacağı piksel bölgeleri



Şekil 6.20 : Line buffer yapısı

Line buffer yapısı kullanılarak, gelen L kanalı verilerinin yatay türevi alınmış oldu. Daha sonra yatay türevi alınan L kanalı piksel verileri ile S kanalı verileri arasında bitisel veya işlemi gerçekleştirildi ve eşikleme uygulandı. Eşik işlemi ile belirlenen sınır değerinin üzerindeki piksellerin 1 yani beyaz, sınır değerinin altındaki piksel değerlerini 0 yani siyah olarak değerlendirilmesini sağlandı. Bu işlem ile birlikte görüntü üzerinde uygulanması gereken ön işlem aşamaları tamamlanmış oldu.

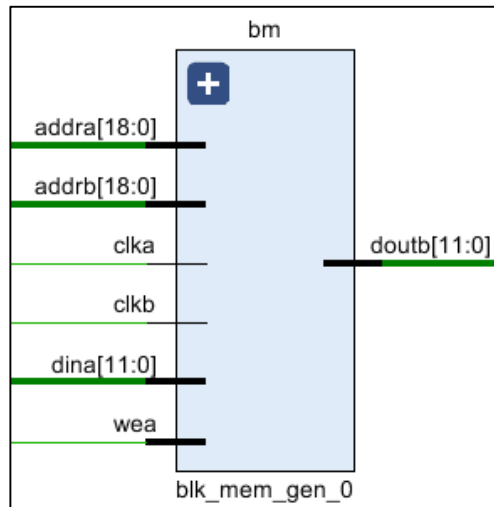
Bitsel Veya				
input1	1	0	0	1
input2	0	0	1	1
RESULT	1	0	1	1

Şekil 6.21 : Örnek bir “bitsel veya” işlemi

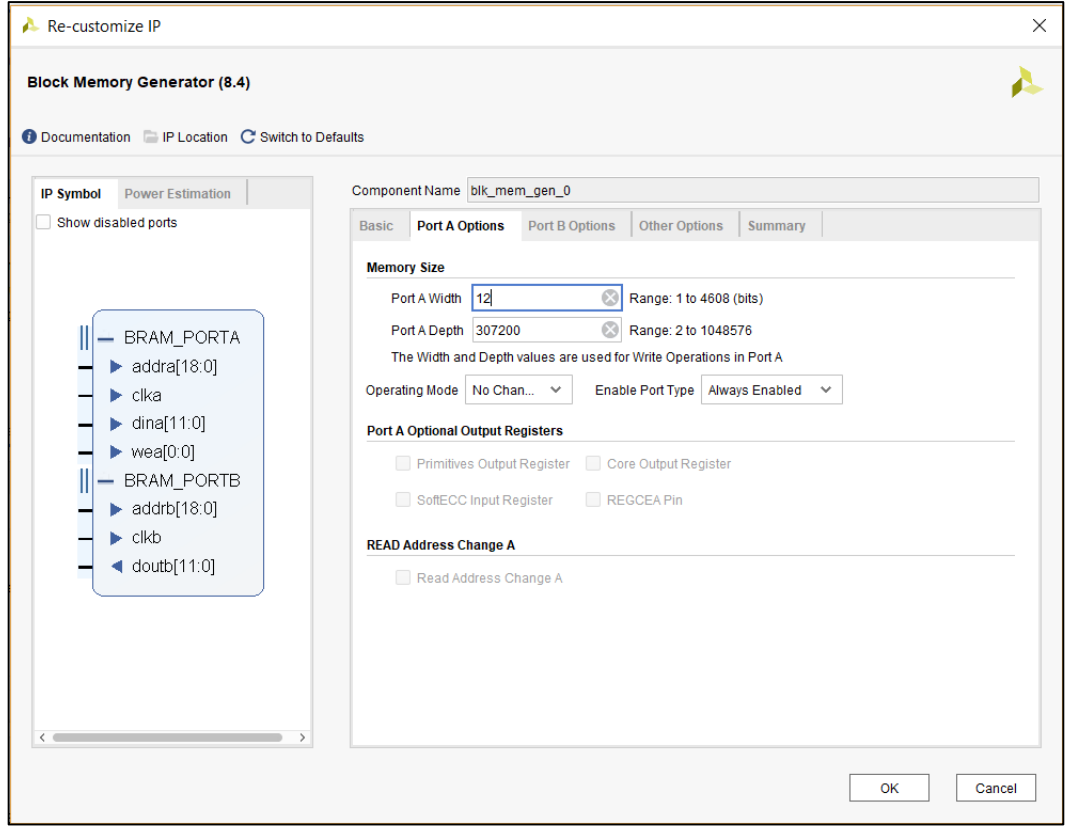
Bitsel veya işlemi sonrası elde edilen 4 bitlik veri 3 kere birleştirilerek 12 bitlik “bram_data” elde edilir. Bram_data, işlenmiş verinin hafıza bloğunun hangi adresine yazılacağını gösteren “bram_addr” bilgisi ve verinin hafızaya ne zaman yazılıp ne zaman yazılmayacağını belirten “we” sinyali ile birlikte beraber blok RAM’e gönderilir. Sobel modülü de PCLK sinyali ile senkronize şekilde çalışmaktadır ve line buffer yapısındaki gecikme haricinde her saat periyodunda bir çıkış üretmektedir.

6.5 Blok RAM Modülü

FPGA içindeki blok RAM’ler anlık hafıza ihtiyaçlarını karşılayarak performans açısından büyük avantaj sağlarlar [36]. Kullandığımız Zedboard’un 4.9 Mb’lık blok RAM kapasitesi vardır. Xilinx firmasının kullanıcılara sunduğu hazır IP kütüphanesinden bir adet blok RAM projeye eklendi. Kullandığımız RAM bloğu 12 bitlik genişliğe 307.200 bitlik derinliğe sahip olacak şekilde ayarlandı. Blok RAM’in özellikleri ile giriş çıkış bilgileri Şekil 6.22 ve Şekil 6.23’de gösterilmiştir.



Şekil 6.22 : Projeye istenilen özelliklerde blok RAM eklenmesi



Şekil 6.23 : Blok RAM'in boyutlarının ayarlanması

Blok RAM için “single port” ve “dual port” seçenekleri mevcuttur. Projede dual port blok RAM kullanıldı. Bu sayede bir taraftan veriler hafızaya yazılırken, diğer taraftan verilerin okunmasına imkan sağlandı. Aynı zamanda veriyi işleyip yazan yapı ile, veriyi okuyup dış ortama gönderen yapı da senkronize edilmiş oldu. Görüldüğü gibi blok RAM üzerinde iki farklı saat işareti bulunmaktadır. Bunlardan “clka” önceki modüllerimizde de kullandığımız PCLK sinyalidir ve verilerin hafızaya yazılmasını sağlar. Diğer saat girişi olan “clkb” ise hafızadaki bilgilerin okunmasını ve VGA modülüne gönderilmesini kontrol eder. 12 bitlik “dina” girişi sobel modülünden gelen verileri “addra”nın belirttiği adrese yazar. Doutb çıkışı ise “addrb”nin belirttiği adres bölgesindeki verileri VGA modülüne gönderir.

6.6 VGA Modülü

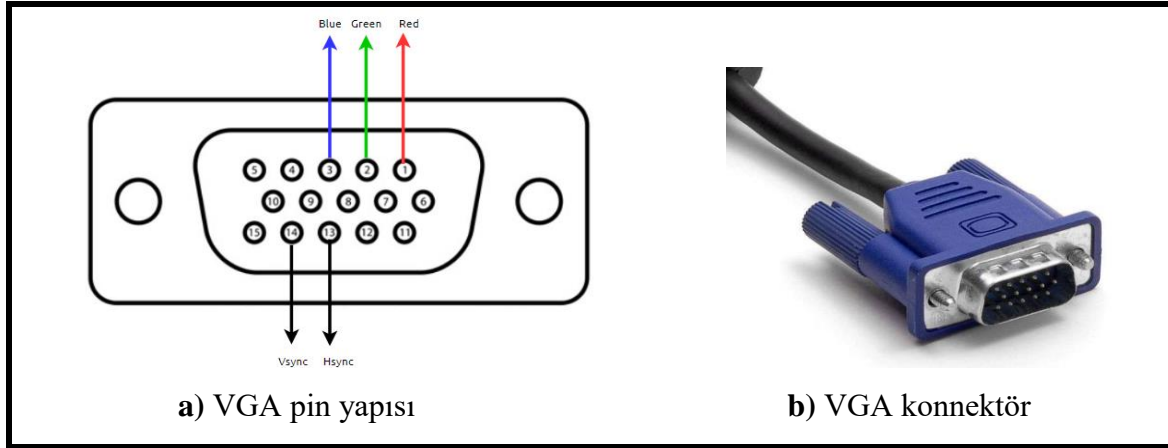
İşlenen görüntünün ekrana yansıtılıp gözlemlenmesi için Zedboard üzerindeki VGA portu kullanıldı. VGA açılımı “Video Graphics Array” (Video Grafik Dizisi) olan 1988 yılında IBM tarafından piyasaya sürülen görüntü standartıdır. 640*480'lik

görüntüyü veya 15-pin D-sub konektörü temsil eder [37]. VGA, analog monitörleri kontrol etmek için kullanılır.

VGA’de ekran piksellere ayrılır, bu pikseller farklı gerilim değerleriyle ifade edilerek kırmızı, mavi ve yeşil renklerinin kombinasyonları ortaya çıkar ve ekran görüntüsü elde edilmiş olur. Görüntünün ekrana basılması sol üst köşeden başlar, sağa doğru devam eder. Satır sonu geldiğinde bir alt satırın en başına geçilir, bütün pikseller taranır ve işlem sağ alt köşede sonlanır. Daha sonra tekrar başlangıç noktasına dönlür. VGA’de kullanılan 5 aktif sinyal Çizelge 6.2’de gösterilmiştir. VGA konektörünün görüntüsü Şekil 6.24’deki gibidir.

Çizelge 6.2 : VGA sinyalleri.

Red (R)	Kırmızı renk bilgisini taşıyan analog sinyal
Green (G)	Yeşil renk bilgisini taşıyan analog sinyal
Blue (B)	Mavi renk bilgisini taşıyan analog sinyal
Vertical Sync	Sütun eşlemesini sağlayan dijital sinyal
Horizonyal Sync	Satır eşlemesini sağlayan dijital sinyal

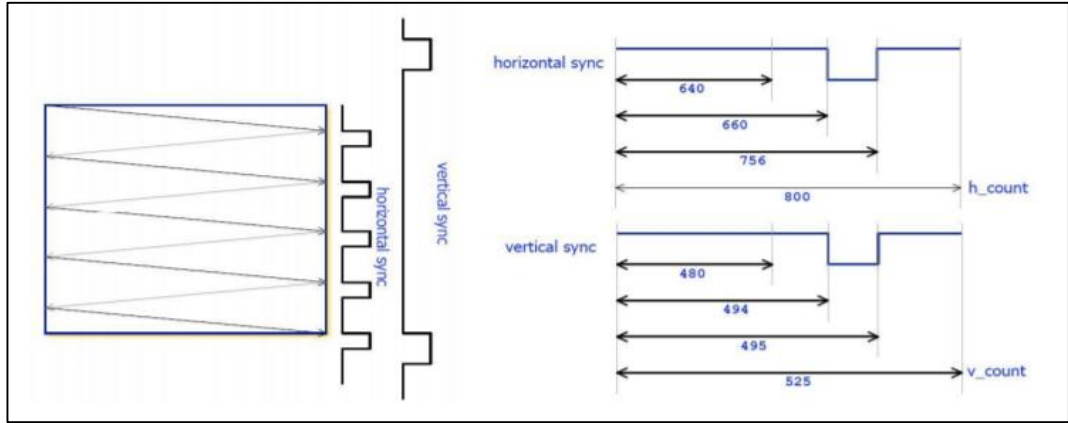


Şekil 6.24 : VGA konektör ve pin yapısı

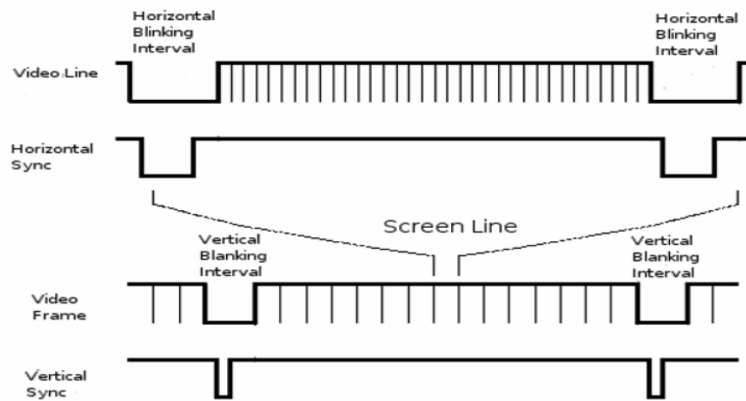
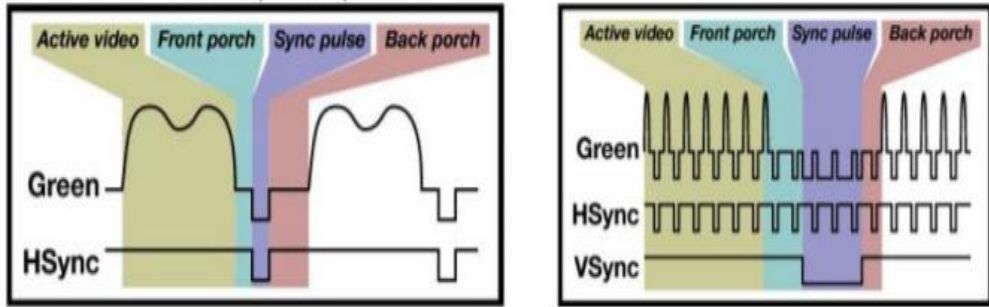
Görüntünün hareketli olarak algılanması için bir saniye içinde defalarca taranması gerekir. Bu hızı ifade etmek için tazeleme oranı kullanılır [38]. Tazeleme oranı bir saniyede yapılan tarama sayısıdır. Görüntü boyutunun 640x480 olduğunu ve 60 Hz’lik

tazeleme oranı üzerinde bir hesap yaparsak, yatay ve dikey kontrol sinyalleri ile beraber piksel periyodu yaklaşık 40 ns’de tamamlanır. Bu yüzden ekranı sürececek modül için 25 MHz’lik bir saat frekansı gereklidir. Vsync sinyali yeni görüntü çerçevesinin başladığını, hsync sinyali ise yeni satırın başladığını ekrana bildirir.

Bütün bir çerçevenin ve bir satırın doldurulmasının öncesinde ve sonrasında boş çevrimler bulunmaktadır. Bu çevrimler kontrol amaçlıdır. Sinyallerin yapısı Şekil 6.25’te ve 6.26’da gösterilmiştir.



Şekil 6.25 : Dikey ve yatay tarama işlemleri



Şekil 6.26 : Vsync ve Hsync sinyalleri

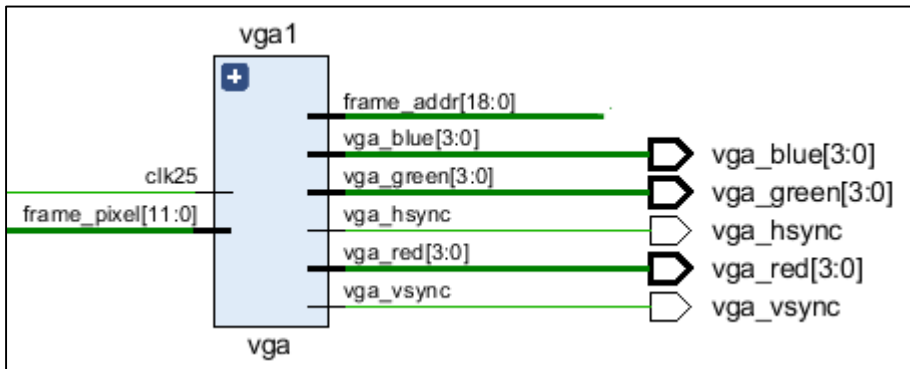
Farklı format ve tazeleme oranları için, piksel saat hızı ve tarama çevrimleri farklılık göstermektedir. Bizim sistemimiz için gerekli olan değerler Çizelge 6.3'te gösterilmiştir.

Çizelge 6.3 : Tarama için gerekli frekans ve tarama çevrimleri [39]

Format	Pixel Clock (MHz)	Horizontal (in Pixels)				Vertical (in Lines)			
		Active Video	Front Porch	Sync Pulse	Back Porch	Active Video	Front Porch	Sync Pulse	Back Porch
640x480, 60Hz	25.175	640	16	96	48	480	11	2	31

Şekil 6.27'de görüldüğü gibi tasarlanan VGA modülünün 2 giriş 6 çıkışı bulunmaktadır. Frame_pixel girişi blok RAM'den okunan verileri ifade eder ve 12 bit genişliğindedir. Clk25 girişi modülümüz için gerekli olan 25 MHz'lik saat işaretidir. Bu saat işareti saat bölücü modülünden sağlanır.

Kontrol sinyallerini de dahil edersek dikey taramada 525, yatay taramada 800 çevrim oluşur. Bunun için "hCounter" ve "vCounter" adında iki sayıcı tanımlandı. Bu sayıcılar aktif bölgelerdeyken, girişteki piksel verisini çıkışa aktarırken, diğer durumlarda lojik 0 seviyesindedir.



Şekil 6.27 : VGA modülü

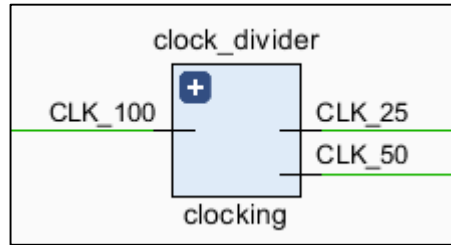
Frame_addr çıkışı blok RAM'e gider ve hafızanın hangi bölgesinden veri okunacağını belirten adres bilgisini taşır. 4'er bitlik vga_blue, vga_red, vga_green çıkışları kırmızı,

mavi ve yeşil renk bileşenlerinin değerlerini ifade eder. Diğer çıkışlar ise vsync ve hsync çıkışlarıdır.

VGA modülünün giriş çıkışları Şekil 6.27'deki gibidir. VGA modülü ilk olarak kamerayı test etmek için kullanıldı. Kameradan alınan görüntü FPGA üzerinden geçtikten sonra ekrana yansıtıldı ve kameranın sağlıklı bir şekilde çalıştığı gözlemlendi. Daha sonra kameradan alınan şerit görüntüsü gerekli ön işlem aşamalarından geçtikten sonra yine VGA modülü ile ekrana yansıtıldı.

6.7 Saat Bölücü Modülü

Saat bölücü modülü farklı frekanslarda çalışan bloklar için gerekli olan saat işaretlerinin iletimini sağlar. 25 Mhzlik saat çıkışı VGA modülüne giderken, 50 Mhzlik saat çıkışı blok RAM'in ikinci saat girişine ve camera_controller modülüne gider. Bilindiği gibi görüntünün alınması ve ön işlem aşamalarını içeren bloklar PCLK saat işaretini kullanıyordu. XCLK saat sinyali 25 MHz'dir, projede PCLK ve XCLK aynı saat frekansında çalıştırılmıştır. Modülün giriş ve çıkışları Şekil 6.28'de gösterilmiştir.



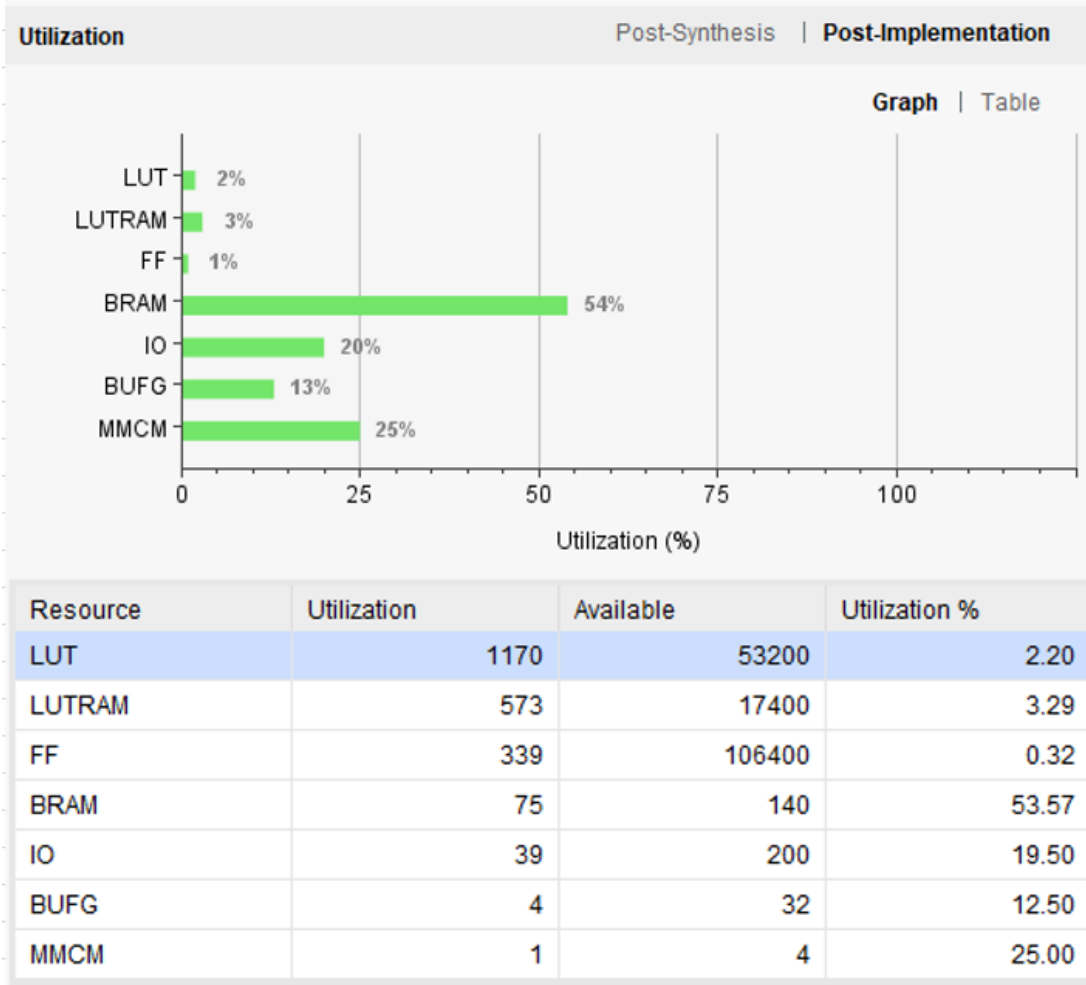
Şekil 6.28 : Saat bölücü modülü

6.8 Tasarlanan Donanımın Benzetim Sonuçları

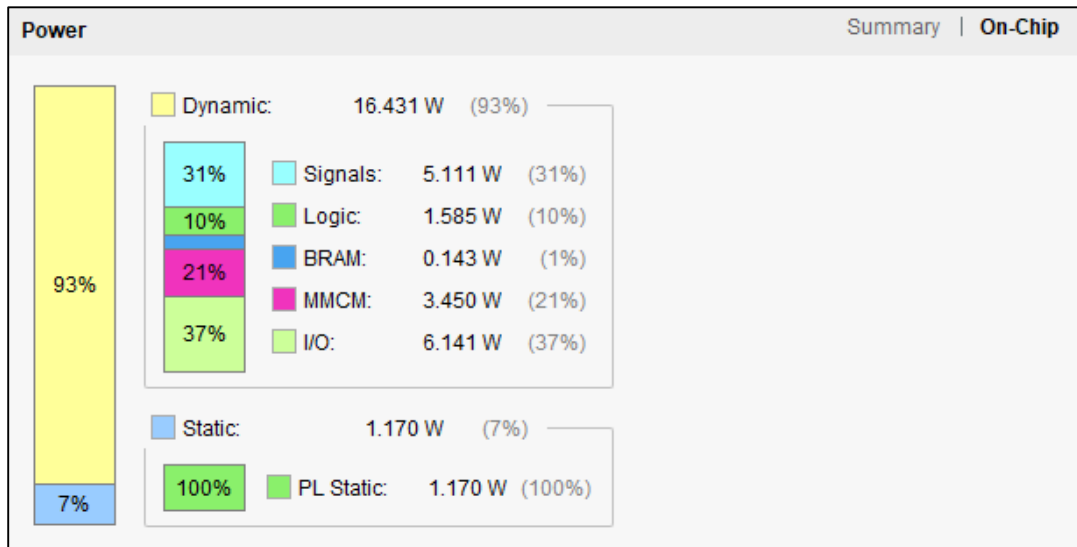
6.8.1 Kaynak kullanımı ve güç tüketimi

Verilog donanım tanımlama dili kullanılarak Vivado üzerinde sistem tasarımı tamamlandı. Kameranın bağlantıları için PMOD pinleri, ekran için de VGA pinleri kullanıldı. Projeye "XDC" dosyası eklenerek, kameranın ve VGA konnektörünün hangi giriş çıkışları kullanacağı belirtildi. Sistem sentezlenip, implemantasyon işlemi de tamamlandıktan sonra bit dosyası üretildi ve FPGA'ye yüklendi. Şekil 6.29'da sistemin alan kullanımı bilgileri gösterilmektedir. Görüldüğü üzere blok RAM dışında

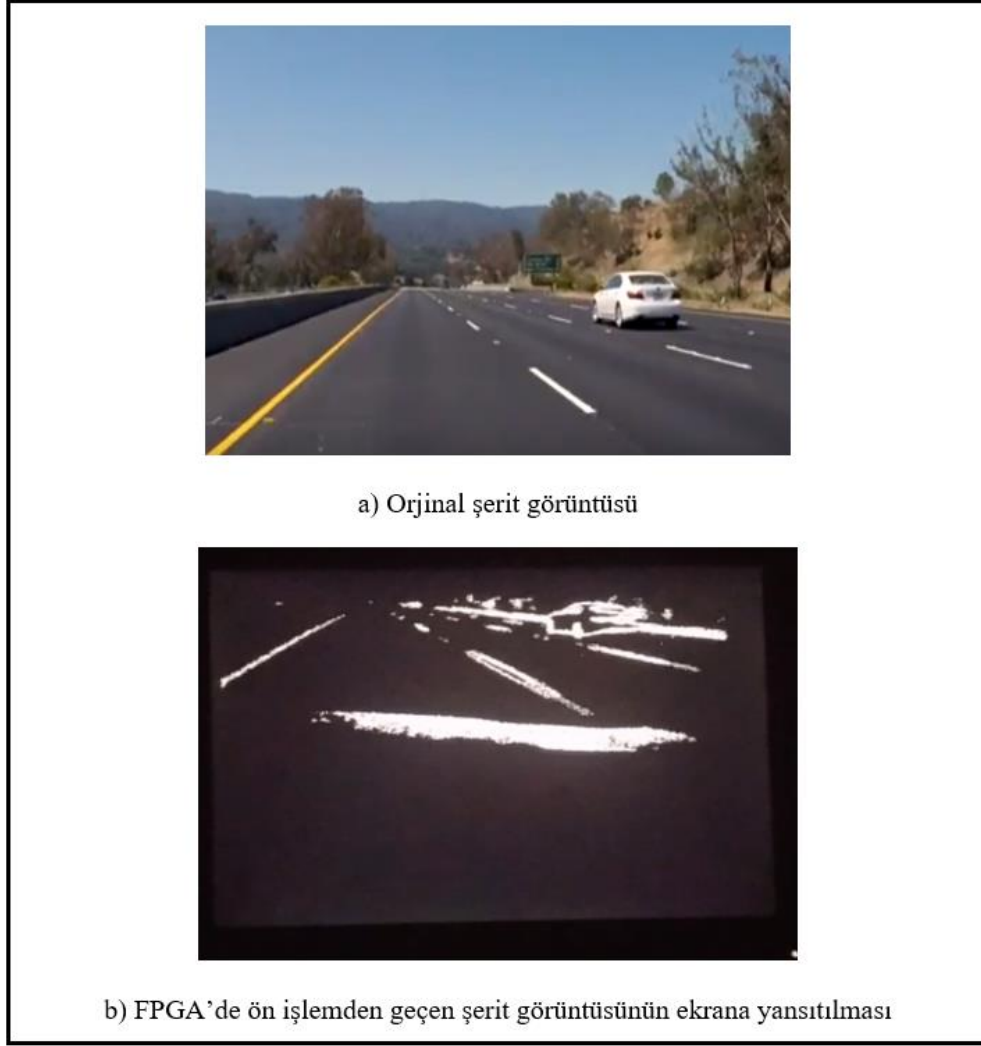
kaynak kullanımları oldukça makul seviyededir. Şekil 6.30’da sistemin güç tüketimi sonuçları gösterilmiştir.



Şekil 6.29 : Sistemin alan kullanımı



Şekil 6.30 : Sistemin güç tüketimi sonuçları



Őekil 6.31 : Orjinal ve iŐlenmiŐ Őerit grnts

6.8.2 Zaman analizi

Kameramızdan 640*480 boyutlarında ereveseler, 25 MHz saat frekansıyla gnderilmektedir. Yatay ve dikey bekleme srelerini ihmal edersek bir grnt erevesinde iŐlenmesi gereken aktif 307.200 piksel vardır. Sistem tasarlanırken iŐ hattı (pipeline) yapısına uygun Őekilde tasarlanmıŐtır. Bu sayede iŐlenebilir piksel verileri alınmaya baŐlanmasıyla beraber her saat darbesinde bir iŐlenmiŐ piksel verisi elde edilmektedir. Line buffer bloklarının dolması iin geen kısa sre de hesaba katılırsa bir tam erevenin iŐlenme sresi yaklaşık olarak 12,3 milisaniye

sürmektedir. Aktif pikseller göz önüne katılarak yapılan zaman analizi sonucu Çizelge 6.4'te gösterilmiştir.

Çizelge 6.4 : Zaman analizi.

Fonksiyon	Bir Çerçevenin İşlenme Süresi (Aktif Pikseller)
Ön İşleme	12,3 ms

FPGA üzerindeki sistemin çalışacağı maximum saat frekansını en yavaş çalışan (gecikmenin en fazla olduğu) kombinezonsal blok belirler. Ancak bu sistemde bizi sınırlayan kameranın çalışma frekansıdır. PCLK sinyalinin 25 MHz yerine 100 MHz olduğu bir kamera sistemi ile çalışılıyorsa aynı boyutlardaki bir çerçevenin işlenme süresi 3 ms seviyesinde olacaktı. Sonuçların da gösterdiği üzere FPGA ile Verilog kullanılarak gerçekleştirilen tasarımda işlem süresi önemli ölçüde kısalmıştır. 12,3 ms'de bir çerçeve işleyen sistem yaklaşık olarak saniyede 80 çerçeve işleyebilir (80 fps). Bu performans ile sistemimiz rahatlıkla gerçek zamanlı olarak çalışabilmektedir. Daha verimli sonuç elde edilmek istenildiği takdirde daha gelişmiş kamera sistemleriyle çalışılması gerekmektedir.

7. SONUÇ

Geliştirilen Şerit Takip Algoritması, şerite ait çizgileri ikinci dereceden polinomial denklemlerle ifade ettiği için virajlı yollarda başarılı olarak çalışabilmektedir. Fakat dağ yolları gibi virajların çok keskin olduğu durumlarda şerit çizimi kısmen başarılıdır. Sistem, ışık, gölge, değişik renkte çizilen şerit çizimleri gibi farklı senaryolarda başarılı olarak çalışmaktadır.

Gerçek zamanlı bir ADAS sistemi geliştirmek için saniyede 15 çerçeve işlemek gerekmektedir. Bu projede “C++” programlama dilinde “OpenCV” kütüphanesi kullanılarak yapılan Şerit Takip Sistemine girdi olarak 640 * 480 çözünürlüklü görüntü çerçevesi verilmiş ve özellikleri daha önce belirtilen bilgisayar ortamında; saniyede 12.84 çerçeve işleme hızına ulaşılmıştır. Bir görüntü çerçevesi için “önişleme fonksiyonu” 36,99 ms sürerken, tüm sistem çalıştığı takdirde 77,84 ms sürmektedir. Şerit Takip Sistemi Zedboard FPGA Geliştirme Kartında sadece ARM Cortex A9 üzerinde çalıştırılmış ve saniyede 2.55 çerçeve işleme hızına ulaşılmıştır. Bir görüntü çerçevesi için “önişleme fonksiyonu” 226,88 ms sürerken, tüm sistem çalıştığı takdirde 392,79 ms sürmektedir. Bu iki sonuç sadece işlemci üzerinde gerçekleştirildiğinden dolayı aradaki farkın sebebi bilgisayardaki işlemci (2.5 GHz Intel Core i5) ile Zedboard’taki işlemcinin (667 MHz ARM Cortex A9) performans farkıdır.

SDSoC platformu üzerinden “XfOpencv” kütüphanesi kullanılarak Zedboard’taki donanım kısmından yararlanıldığı takdirde, Önişleme fonksiyonu için (perspektif dönüşümsüz): 8.78 çerçeve işleme hızına ulaşılmıştır. Donanım bloğu kullanılmayıp işlemcide gerçekleştirildiği takdirde (perspektif dönüşümsüz) 6.37 çerçeve işleme hızındadır. Sonuç olarak SDSoC platformu ve “XfOpencv” kütüphanesi ile Zedboard üzerinden gerçekleştirilen SoC yapısı 1.38 kat daha performanslı çalışmaktadır.

Vivado platformunda, 25 MHz çıktı üreten kamera kullanılmış ve Şerit Takip Sisteminin “önişleme” fonksiyonu 12,3 ms’de gerçekleştirilmiştir. Sadece önişleme fonksiyonu olarak düşünüldüğünde 81.3 çerçeve işleme hızına ulaştığını gösterir.

Vivado platformu üzerinden düşük seviye bir dil olan “verilog” donanım tanımlama dili ile Şerit Takip Sisteminin “önişleme” fonksiyonu için yapılan tasarım ile “önişleme” fonksiyonu 2.5 GHz Intel Core i5 işlemcisine göre 3.01 kat, 667 MHz ARM Cortex A9 işlemcisine göre 18.4 kat daha performanslı çalışmaktadır.

Donanım tasarımlarında yüksek seviye dil kullanımı yaygın değildir fakat teknolojik gelişmeler ile birlikte bu alanda yeni araçlar çıkmaktadır. SDSoC platformu Xilinx firması tarafından geliştirilmiş bu araçlardan biridir. Düşük seviye dillerle yapılan tasarımların çok uzun sürmesi, kısa vadede netice alınamaması yüksek seviye dillerle geliştirme ortamı sunmaya yönelmektedir. Daha önceden işlemciler için geliştirilen yazılımları kısa vadede donanıma aktarmak düşük seviye donanım tanımlama dilleri ile mümkün değildir.

SDSoC ile Vivado platformları karşılaştırılması yapıldığında SDSoC platformunda yüksek seviye dil olan “C++” kullanılırken; Vivado platformunda donanım tanımlama dili olarak yapılandırılmış düşük seviye bir dil olan “verilog” kullanılmıştır. Yüksek seviye bir dil ile yapılan donanım tasarımında yüksek oranda performans artışı gözlemlenmemişken düşük seviye dil ile yapılan donanım tasarımında yüksek oranda performans artışı sağlanmıştır.

Bu proje sonucunda çıkarılan sonuç; kısa vade çözümler için yada referans çözümler için yüksek seviye dil kullanımı uygun olurken uzun vadede yapılacak yada performans artışının yüksek olması beklenildiği çözümlerde düşük seviye dil kullanılması uygundur.

KAYNAKLAR

- [1] **I. El Hajjouji, A. El Mourabit, Z. Asrih, S. Mars and B. Bernoussi**, "FPGA based real-time lane detection and tracking implementation," 2016 International Conference on Electrical and Information Technologies (ICEIT), Tangiers, 2016, pp. 186-190.
- [2] **Wesolkowski, S., Jerigan, M. E., Dony, R. D.**, "Comparison of Color Image Edge Detectors in Multiple Color Spaces", Proc. IEEE Int. Conf. Image Process, Vol. II, 796-799, 2000.
- [3] **Hough, P.V.C.**, (1962). Method and Means for Recognizing Complex Patterns. US Patent, 3069654.
- [4] **Wikipedia**, http://tr.wikipedia.org/wiki/Kalman_Filtresi
- [5] **Volder, J.E.**, 'The CORDIC Trigonometric Computing Technique', IRE Trans. Electronic Computers, Vol.8, September 1959, pp.330-334.
- [6] **M. Revilloud, D. Gruyer and M. C. Rahal**, "A lane marker estimation method for improving lane detection," 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, 2016, pp. 289-295.
- [7] **R. Fan, V. Prokhorov and N. Dahnoun**, "Faster-than-real-time linear lane detection implementation using SoC DSP TMS320C6678," 2016 IEEE International Conference on Imaging Systems and Techniques (IST), Chania, 2016, pp. 306-311.
- [8] **Chaple, G. and R.D. Daruwala**, "Design of Sobel operator based image edge detection algorithm on FPGA. in Communications and Signal Processing (ICCSP)", 2014 International Conference on. 2014, pp.788-792.
- [9] **T. Wang**, "Phd Forum: Real-Time Lane-Vehicle Detection for Advanced Driver Assistance on Mobile Devices," 2017 IEEE International Conference on Smart Computing (SMARTCOMP), 2017.
- [10] "About Opencv" About - OpenCV library. [Online]. Available: <https://opencv.org/about.html>. [Accessed: 29-Mar-2018].

- [11] “MATLAB Product Description,” MathWorks. [Online]. Available: https://ch.mathworks.com/help/matlab/learn_matlab/product-description.html. [Accessed: 29-Mar-2018].
- [12] “Halcon Product Information,” *MVTec*. [Online]. Available: <http://www.mvtec.com/products/halcon/>. [Accessed: 29-Mar-2018].
- [13] **D. Tschumperle**, "The CImg Library - C++ Template Image Processing Toolkit", Cimg.eu, 2018. [Online]. Available: <http://cimg.eu/>. [Accessed: 29- Mar- 2018].
- [14] "Fiji is just ImageJ", Fiji.sc, 2018. [Online]. Available: <https://fiji.sc/>. [Accessed: 29- Mar- 2018].
- [15] **Özçelik, M.F.** 2012. “Görüntü İşleme Algoritmalarının FPGA Üzerinde Gerçeklenmesi”, Yüksek Lisans Tezi, Gazi Üniversitesi Elektrik Bilişim Enstitüsü, Ankara.
- [16] IEEE Standard Verilog® Hardware Description Language. New York: IEEE Computer Society, 2001.
- [17] VHDL Reference Manual. Washington: Synario Design Automation, 1997.
- [18] What is VLSI ASIC DESIGN. Erişim: 27 Mayıs 2018, <https://www.quora.com/What-is-VLSI-ASIC-DESIGN>.
- [19] **ACAR, B.** 2017. “Hafif Bir Kripto Algoritması Olan Boron’un Fpga Üzerinde Mikroişlemci İle Beraber İlk Kez Gerçeklenmesi”, Bitirme Tezi, İstanbul Teknik Üniversitesi Elektrik Elektronik Fakültesi, İstanbul.
- [20] **Avnet** zedboard. <http://zedboard.org/product/zedboard>.
- [21] **Xilinx**, 2016. Zynq-7000 All Programmable SoC Technical Reference Manual.
- [22] ARM Cortex A9. Erişim: 27 Mayıs 2018, <https://developer.arm.com/products/processors/cortex-a/cortex-a9>.
- [23] What is RISC ? . Erişim: 27 Mayıs 2018, <https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/whatis/index.html>
- [24] Zynq-7000 All Programmable SoC Data Sheet: Overview. United States: Xilinx, 2017, p. 1.

- [25] **Wikipedia**, <https://tr.wikipedia.org/wiki/Verilog>
- [26] "What is Verilog", Doulos.com, 2018. [Online]. Available: https://www.doulos.com/knowhow/verilog_designers_guide/what_is_verilog/. [Accessed: 29- Mar- 2018].
- [27] **Xilinx**, 2017. Vivado Design Suite User Guide.
- [28] **Xilinx**, 2017. Software Development Kit User Guide.
- [29] **Xilinx**, 2017. SDSoc Environment User Guide
- [30] "Xilinx/xfopencv", GitHub, 2018. [Online]. Available: <https://github.com/Xilinx/xfopencv>. [Accessed: 29- Mar- 2018].
- [31] OV7670 Advanced Information Preliminary Datasheet, <https://www.voti.nl/docs/OV7670.pdf>
- [32] **OmniVision**, "OmniVision Serial Camera Control Bus(SCCB) Functional Specification", June 2007
- [33] **Wikipedia**, [https://en.wikipedia.org/wiki/Pipeline_\(computing\)](https://en.wikipedia.org/wiki/Pipeline_(computing))
- [34] "Math behind colorspace conversions, RGB-HSL", www.niwa.nu, 2018. [Online] Available: <http://www.niwa.nu/2013/05/math-behind-colorspace-conversions-rgb-hsl/>. [Accessed: 20- May- 2018].
- [35] "Image Filtering in FPGAs", blog.teledynedalsa.com, 2018 [Online] Available: <http://blog.teledynedalsa.com/2012/05/image-filtering-in-fpgas/>. [Accessed: 20- May- 2018].
- [36] **Xilinx**, 2012. Xilinx 7 Series FPGAs Embedded Memory Advantages.
- [37] **Wikipedia**, http://en.wikipedia.org/wiki/Video_Graphics_Array.
- [38] **Özcan,H. 2009**. "Gerçek zamanlı lineer görüntü işleme algoritmalarının FPGA ile gerçekleştirilmesi", Lisans Tezi, Yıldız Teknik Üniversitesi, Elektrik Elektronik Fakültesi, İstanbul
- [39] Ickes, Nathan, (2004) "VGA Video", Introduction to Digital Systems, <http://web.mit.edu/6.111/www/s2004/NEWKIT/vga.shtml>.

ÖZGEÇMİŞ

Adı - Soyadı : Yakup Görür
Doğum Tarihi ve Yeri : Kahramanmaraş, 1995
E-posta : gorury@itu.edu.tr
Lisans : İstanbul Teknik
Üniversitesi, Elektronik ve Haberleşme Mühendisliği,
2013-2018



ÖZGEÇMİŞ

Adı - Soyadı : Mehmet Akif Akkaya
Doğum Tarihi ve Yeri : İstanbul, 1995
E-posta : akifakkaya1@gmail.com
Lisans : İstanbul Teknik
Üniversitesi, Elektronik ve Haberleşme Mühendisliği, 2013-2018

