T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

FPGA TABANLI GERÇEK ZAMANLI GÖRÜNTÜ İŞLEME İLE BASKI DEVRE KARTI HATALARININ TESPİT EDİLMESİ

Merve AYDIN

DANIŞMAN
Prof. Dr. Fırat KAÇAR

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Elektrik-Elektronik Mühendisliği Programı

Haziran, 2022

TEZ KABUL VE ONAYI

Merve AYDIN tarafından, **Prof. Dr. Fırat KAÇAR** danışmanlığında hazırlanan **"FPGA TABANLI GERÇEK ZAMANLI GÖRÜNTÜ İŞLEME İLE BASKI DEVRE KARTI HATALARININ TESPİT EDİLMESİ"** başlıklı bu çalışma, jürimiz tarafından **07/06/2022** tarihinde yapılan sınav sonucunda **oy birliği** ile başarılı bulunarak **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Tez Jürisi

		İmza	**Sonuç**
DANIŞMAN	Prof. Dr. Fırat KAÇAR İstanbul Üniversitesi – Cerrahpaşa Elektrik-Elektronik Mühendisliği Elektronik Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Doç. Dr. Serap ÇEKLİ İstanbul Üniversitesi – Cerrahpaşa Elektrik-Elektronik Mühendisliği Telekomünikasyon Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Doç. Dr. Yunus BABACAN Erzincan Binali Yıldırım Üniversitesi Elektrik-Elektronik Mühendisliği Elektronik Anabilim Dalı		☒ Kabul ☐ Ret

Sevgili eşim Ömer'e ithaf ediyorum…

BÜTÇE DESTEKLERİ

FPGA TABANLI GERÇEK ZAMANLI GÖRÜNTÜ İŞLEME İLE BASKI DEVRE KARTI HATALARININ TESPİT EDİLMESİ

Bu tez çalışması için herhangi bir kurumdan bütçe desteği alınmamıştır.

TEŞEKKÜR

Tez çalışma sürecim boyunca karşılaştığım tüm zorlukları aşmamı sağlayan bilgi ve tecrübeleri ile bana destek olan saygıdeğer danışman hocam Prof. Dr. Fırat KAÇAR'a teşekkürlerimi sunarım.

Lisans ve yüksek lisans tez çalışmalarım boyunca yardımlarını esirgemeyen tüm hocalarıma ve arkadaşlarıma teşekkür ederim.

Tez çalışmamda beni yalnız bırakmayan her zaman maddi ve manevi destek olan aileme ve eşim *Ömer'e* sonsuz sevgi ve saygılarımı sunarım.

Haziran 2022 Merve AYDIN

İÇİNDEKİLER

Sayfa No

TEZ KABUL VE ONAYI ..... **ii**
BÜTÇE DESTEKLERİ ..... **v**
TEŞEKKÜR ..... **vi**
İÇİNDEKİLER ..... **vii**
ŞEKİL LİSTESİ ..... **x**
TABLO LİSTESİ ..... **xiv**
KISALTMA LİSTESİ ..... **xv**
ÖZET ..... **xvi**
ABSTRACT ..... **xvii**
1. GİRİŞ **1**
2. KAVRAMSAL ÇERÇEVE **4**
2.1. ALANDA PROGRAMLANABİLİR KAPI DİZİLERİ (FPGA) 4
2.1.1. FPGA Tanımı ve Özellikleri 4
2.1.2. FPGA Yapısı ve Blokları 5
2.1.3. FPGA Pinleri ve Modları 7
2.1.4. FPGA Avantajları 7
2.1.5. FPGA Programlanması 8
2.2. DONANIM TANIMLAMA DİLİ 9
2.2.1. Donanım Tanımlama Dili Seviyeleri ve Özellikleri 9
2.2.2. VHDL ve Bileşenleri 10
2.3. SAYISAL GÖRÜNTÜ 11
2.3.1. Sayısal Görüntü Formatları: RGB ve HSV (Hue-Saturation-Value) 11
2.3.2. Sayısal Görüntü İşleme Seviyeleri 12
2.3.3. Sayısal Görüntü İşleme Adımları 13
2.3.4. Sayısal Görüntü İşleme Sisteminin Elemanları 13
2.4. GÖRÜNTÜ SENSÖRLERİ 14
2.5. FİLTRELEME TEKNİKLERİ 15
2.5.1. Medyan Filtreleme 17

2.5.2. Gauss Filtreleme 18
2.6. VİDEO GRAFİK DİZİNİ (VGA) TEKNOLOJİSİ 20
2.6.1. VGA Sinyalleri 20
2.6.2. VGA Bölgeleri 21
2.6.3. VGA Çerçeve Zamanlaması 23
3. YÖNTEM 25
3.1. DONANIMSAL SİSTEM TASARIMI 25
3.2. ZEDBOARD FPGA GELİŞTİRME KARTI 27
3.2.1. ZedBoard Özellikleri 27
3.2.2. ZYNQ-7000 SoC Mimarisi 28
3.3. OV7670 GÖRÜNTÜ SENSÖRÜ 29
3.3.1. Görüntü Sensörü Özellikleri ve Pinleri 29
3.3.2. Görüntü Sensörü için SSCB (Seri Kamera Kontrol Veri Yolu) 30
3.4. VIVADO PROGRAMI 32
3.4.1. Vivado IP Blokları 32
3.4.2. Vivado Modül Sınıflandırılması 33
3.5. FPGA BAĞLANTILARI 34
3.5.1. OV7670 Görüntü Sensörü Bağlantısı 34
3.5.2. VGA Bağlantısı 36
3.5.3. LED ve Anahtar Bağlantıları 38
3.5.4. Saat Kaynağı, Reset ve Buton Pinleri Bağlantısı 38
3.6. VHDL BAĞLANTI ŞEMASI 39
3.7. DONANIM TASARIMI ÜST MODÜLÜ 40
3.8. SAAT YÖNETİCİSİ IP BLOKLARI 43
3.9. BUTON ARKI MODÜLÜ 45
3.10. KONTROLÖR MODÜLÜ 47
3.10.1. Register Modülü 49
3.10.2. I2C Modülü 52
3.11. GÖRÜNTÜ YAKALAMA MODÜLÜ 53
3.12. MEDYAN FİLTRELEME MODÜLLERİ 56
3.12.1. Satır Arabelleği IP Blokları 58
3.12.2. Kernel Matrisi ile Kayan Pencereleme İşlemi 60
3.12.3. Piksel Sayım İşlemi 61
3.12.4. Sıralama İşlemi 62

3.13. GAUSS FİLTRELEME MODÜLLERİ 64
3.13.1. Satır Arabelleği IP Blokları 65
3.13.2. Konvolüsyon Matrisi ile İşlem 66
3.14. ADRESLEME MODÜLÜ 67
3.15. ÇERÇEVE ARABELLEĞİ IP BLOKLARI 68
3.16. VGA SÜRÜCÜSÜ MODÜLÜ 69
3.17. RGB - HSV MODÜLÜ 71
3.17.1. Maksimum, Minimum Renk Değerleri ve Farkı için İşlemler 72
3.17.2. Renk Özü (H) Değeri için İşlemler 72
3.17.3. Renk Doygunluğu (S) ve Renk Parlaklığı (V) Değeri için İşlemler 73
4. BULGULAR 75
4.1. DONANIMSAL SİSTEM ÇALIŞMASI 75
4.2. GÖRÜNTÜ ALIMI DEĞERLENDİRMESİ 76
4.2.1. Görüntü Alımı Testi ve Monitör Çıktısı 76
4.2.2. Görüntü Alımı Çift Çerçeve Arabelleği (Double Frame Buffer) Sonucu 78
4.3. FİLTRELEME SONUÇLARI 79
4.3.1. Medyan Filtreleme Simülasyon Sonucu 81
4.3.2. Gauss Filtreleme Simülasyon Sonucu 86
4.3.3. Filtrelemeler Sonucu Monitör Çıktısı 88
4.4. EKSİK KOMPONENT HATA TESPİT SONUÇLARI 90
4.4.1. HSV Simülasyon Sonucu 90
4.4.2. Eksik Komponent Hata Tespiti Sonucu Monitör Çıktısı 92
4.5. KAYNAK KULLANIMI VE ZAMAN ANALİZİ 93
5. TARTIŞMA 95
6. SONUÇ VE ÖNERİLER 99
KAYNAKLAR 101
EKLER 106
EK 1. VHDL kodlarının RTL şematik gösterimi. 106
EK 2. HSV modülü eklenmiş VHDL kodlarının RTL şematik gösterimi. 112
İNTİHAL RAPORU İLK SAYFASI 116

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1: SRAM tabanlı FPGA mimarisi [15]. 6

Şekil 2.2: RTL seviye türü tasarım. 10

Şekil 2.3: RGB formatı piksel depolama. 12

Şekil 2.4: Sayısal görüntü işleme adımları. 13

Şekil 2.5: Giriş görüntüsü ve kernel matrisi. 16

Şekil 2.6: Dolgulu giriş görüntüsü ve ayna görüntülü kernel konvolüsyon gösterimi. 16

Şekil 2.7: Kernel matrisi ile konvolüsyon sonucu. 17

Şekil 2.8: 3x3 kayan pencere ile medyan değerinin bulunması [25]. 18

Şekil 2.9: Gauss kernel matrisi [26]. 19

Şekil 2.10: Gauss ayrık yaklaşım (σ=1), 3x3 ve 5x5 gauss kernel matrisleri [27]. 19

Şekil 2.11: Görüntü üzerinde gauss matrisi ile konvolüsyon işlemi [27]. 19

Şekil 2.12: VSYNC ve HSYNC sinyalleri ile görüntü üzerinde piksel taraması. 20

Şekil 2.13: VGA sinyali 640 x 480, 60 Hz endüstri standardı zamanlama. 21

Şekil 2.14: HSYNC ve VSYNC sinyal değerleri [30]. 22

Şekil 2.15: RGB 555 çıkış zamanlama şeması. 23

Şekil 2.16: VGA çerçeve zamanlama şeması. 24

Şekil 3.1: Donanımsal sistem bağlantı şeması. 25

Şekil 3.2: Donanımsal sistem bağlantı noktaları. 26

Şekil 3.3: ZedBoard geliştirme kartı [34]. 27

Şekil 3.4: Zynq mimarisi. 29

Şekil 3.5: OV7670 görüntü sensörü. 29

Şekil 3.6: Vivado programı arayüz görünümü. 32

Şekil 3.7: Vivado modül sınıflandırılması. 33

Şekil 3.8: ZedBoard PMOD bağlantısı. 35

Şekil 3.9: OV7670 bağlantısı. 35

Şekil 3.10: ZedBoard PMOD başlığı pinleri.(a) Bağlantı şeması.(b) PMOD başlığı [33]. 35

Şekil 3.11: Vivado programında görüntü sensörü pinlerinin ataması. 36

Şekil 3.12: DB15 tipi, ZedBoard 12-bit VGA konnektörü [33]. 37

Şekil 3.13: Vivado programında VGA pinlerinin ataması. 37

Şekil 3.14: Vivado programında LED ve Anahtar pinlerinin ataması. 38

Şekil 3.15: Vivado programında Saat Kaynağı, Buton, Reset pinlerinin ataması. 39

Şekil 3.16: VHDL bağlantı şeması. 40

Şekil 3.17: Üst modül giriş-çıkış portları şematik gösterimi. 41

Şekil 3.18: Karışık Mod Saat Yöneticisi (MMCM) modüllerinin şematik gösterimi. 43

Şekil 3.19: Karışık Mod Saat Yöneticisi (MMCM) IP blokları. 44

Şekil 3.20: MMCM0 IP blok giriş frekansı ayarı. 44

Şekil 3.21: MMCM0 IP blok çıkış frekansları ayarı. 44

Şekil 3.22: Debounce modülü. 45

Şekil 3.23: Controller modülü. 47

Şekil 3.24: Debounce-Controller modülü bağlantısı. 48

Şekil 3.25: Controller alt modülleri: I2C modülü ve register modülü bağlantısı. 49

Şekil 3.26: Controller alt modülü: Register modülü. 49

Şekil 3.27: Controller alt modülü: I2C modülü. 52

Şekil 3.28: Capture modülü. 53

Şekil 3.29: 640x480 görüntü boyutunun 320x240 olarak gösterimi. 55

Şekil 3.30: 1. medyan filtreleme modülü. 56

Şekil 3.31: 2. medyan filtreleme modülü. 57

Şekil 3.32: Medyan filtrelemede satır arabelleği tanımlama. 58

Şekil 3.33: Vivado programında satır arabelleği ayarlama. 58

Şekil 3.34: 320x240 görüntüde satır arabelleğinin uygulama şeması. 59

Şekil 3.35: Kernel matrisi pencereleme işlemi. 60

Şekil 3.36: 320x240 görüntüde satır arabelleği okuma için piksel sayımı. 61

Şekil 3.37: Piksel sayımı son satır pencereleme işlemi. 62

Şekil 3.38: Pencereleme işlemi başlangıcı. 62

Şekil 3.39: Örnek kernel matrisi elemanları ve piksel değerleri. 63

Şekil 3.40: Gauss filtreleme modülleri. (a) 1.gauss filtreleme. (b) 2.gauss filtreleme..... 65

Şekil 3.41: Gauss filtrelemede satır arabelleği tanımlama. 65

Şekil 3.42: Konvolüsyon uygulaması. (a) Kernel matrisi. (b) Konvolüsyon matrisi. 66

Şekil 3.43: Capture adresleme modülü. 67

Şekil 3.44: Çerçeve arabelleği IP blokları. 68

Şekil 3.45: Capture adresleme; VGA ve çerçeve arabelleği blokları bağlantısı. 69

Şekil 3.46: Vivado programında çerçeve arabelleği ayarlama. 69

Şekil 3.47: VGA sürücüsü modülü. 70

Şekil 3.48: VGA; çerçeve arabelleği blokları ve capture adresleme modülü bağlantısı. 70

Şekil 3.49: RGB - HSV modülü. 72

Şekil 3.50: ROM IP Bloğu. 73

Şekil 3.51: Bölücü IP bloğu. 74

Şekil 4.1: Donanımsal Sistem Görüntüsü. 75

Şekil 4.2: Monitörden 640x480 boyutunda RGB görüntü alımı. 76

Şekil 4.3: 640x480 boyutunda RGB görüntünün monitörde gösterimi. 76

Şekil 4.4: 320x240 boyutunda RGB görüntünün monitörde gösterimi. 77

Şekil 4.5: 640x480, RGB formatta örnek PCB görüntüsü monitör gösterimi. 77

Şekil 4.6: 640x480, RGB formatta örnek PCB görüntüsü monitör gösterimi. 77

Şekil 4.7: 640x480 tek çerçeve arabelleğe sahip görüntüde yırtılma (tearing) oluşumu. 78

Şekil 4.8: 640x480 görüntü boyutu, tek çerçeve arabelleği kaynak kullanımı sonucu. 79

Şekil 4.9: 320x240 görüntü boyutu, çift çerçeve arabelleği kaynak kullanımı sonucu. 79

Şekil 4.10: Filtreleme işlemleri için kullanıcı anahtarları ve LED'lerin gösterimi.....80

Şekil 4.11: Medyan filtre simülasyonu, kernel matris değerleri testbench gösterimi. 81

Şekil 4.12: Medyan filtre simülasyonu, medyan s0 değerleri testbench gösterimi.....82

Şekil 4.13: Medyan filtre simülasyonu, medyan s1 değerleri testbench gösterimi.....82

Şekil 4.14: Medyan filtre simülasyonu, medyan s2 değerleri testbench gösterimi.....83

Şekil 4.15: Medyan filtre simülasyonu, medyan s3, s4 değerleri testbench gösterimi. 84

Şekil 4.16: Medyan filtre simülasyonu, medyan s5, s6 değerleri testbench gösterimi. 85

Şekil 4.17: Medyan filtre simülasyonu, medyan s7,s8 değerleri testbench gösterimi. 86

Şekil 4.18: Medyan filtre simülasyonu, medyan değerinin testbench gösterimi. 86

Şekil 4.19: Gauss filtre simülasyonu, kernel matris değerleri testbench gösterimi. 87

Şekil 4.20: Gauss filtre simülasyonu, gauss değerinin testbench gösterimi. 87

Şekil 4.21: Örnek PCB üzerinde filtrelerin uygulanması.....88

Şekil 4.22: Örnek PCB üzerinde filtrelerin uygulanması.....89

Şekil 4.23: HSV simülasyonu, Cmax ve Cmin değerleri testbench gösterimi. 90

Şekil 4.24: HSV simülasyonu, fark değeri testbench gösterimi.....90

Şekil 4.25: HSV simülasyonu, left değeri testbench gösterimi.....90

Şekil 4.26: HSV simülasyonu, çarpım değeri testbench gösterimi. 91

Şekil 4.27: HSV simülasyonu, H, S, V değerleri testbench gösterimi. 91

Şekil 4.28: Örnek PCB üzerinde bir medyan ve bir gauss filtresi uygulaması. 92

Şekil 4.29: Örnek PCB üzerinde eksik komponent hatası tespiti.....92

Şekil 4.30: MATLAB ortamında lehimli alanların işaretlenmesi. 93

Şekil 4.31: Kaynak kullanımının Vivado programında gösterilmesi. 94

Şekil 4.32: Zamanlama raporunun Vivado programında gösterilmesi. 94

TABLO LİSTESİ

Sayfa No

Tablo 2.1: Görüntü işlemlerine göre örnek kernel matrisleri. 17

Tablo 2.2: Tarama piksel sayıları [29]. 22

Tablo 3.1: OV7670 Görüntü Sensörü Pinleri [31]. 30

Tablo 3.2: FPS değiştirme register ayarı [31]. 31

Tablo 3.3: Çerçeve değiştirme register ayarı [31]. 31

Tablo 3.4: PMOD-Zynq pini ile OV7670 pinleri bağlantısı gösterimi. 36

Tablo 3.5: Zynq pini ile VGA pinleri bağlantısı gösterimi [33]. 37

Tablo 3.6: Zynq pini ile LED ve Anahtar pinleri bağlantısı gösterimi [33]. 38

Tablo 3.7: COM7 değiştirme register ayarı. 51

Tablo 3.8: COM15 değiştirme register ayarı. 52

Tablo 3.9: Örnek kernel matrisi piksel değerleri sıralama işlemi. 64

Tablo 3.10: VGA zaman sabitleri piksel sayıları. 70

KISALTMA LİSTESİ

Kısaltmalar	Açıklama
ADC	: Analog Dijital Çevirici (Analog Digital Converter)
ALU	: Aritmetik Lojik Birimi (Arithmetic Logic Unit)
AOI	: Otomatik Optik İnceleme (Automated Optical Inspection)
ASIC	: Uygulamaya Özel Tümleşik Devre (App.-Specific Integrated Circuit)
ASSP	: Uygulamaya Özel Standart Ürün (Application Specific Standard Product)
BRAM	: Blok Rasgele Erişim Belleği (Block Random Access Memory)
DSP	: Dijital Sinyal İşleme (Digital Signal Processing)
FGPA	: Alanda Programlanabilir Kapı Dizileri (Field Programmable Gate Array)
FPS	: Saniyelik Görüntü Sayısı (Frame Per Second)
HDL	: Donanım Tanımlama Dili (Hardware Description Language)
HSV	: Renk, Doygunluk, Parlaklık (Hue, Saturation, Value)
I2C	: Entegre Devreler Arasında (Inter-Integrated Circuit)
IC	: Entegre Devre (Integrated Circuit)
IP	: Fikri Mülkiyet (Intellectual Property)
MMCM	: Karışık Mod Saat Yöneticisi (Mixed Mode Clock Manager)
PCB	: Baskı Devre Kartı (Printed Circuit Board)
PL	: Programlanabilir Mantık (Programmable Logic)
PLD	: Programlanabilir Mantık Cihazları (Programmable Logic Device)
PLL	: Faz Kilitli Döngü (Phase Locked Loop)
PMOD	: Çevresel Modül (Peripheral Module)
PS	: İşlemci Sistemi (Processing System)
RAM	: Rastgele Erişimli Bellek (Random Access Memory)
RGB	: Kırmızı-Yeşil-Mavi (Red-Green-Blue)
ROM	: Sadece Okunabilir Bellek (Read Only Memory)
SCCB	: Seri Kamera Kontrol Veri Yolu (Serial Camera Control Bus)
SMD	: Yüzey Montaj Elemanları (Surface Mount Devices)
SoC	: Yongada Sistem (System on Chip)
SRAM	: Durağan Rastgele Erişimli Bellek (Static Random Access Memory)
VHDL	: Çok Yüksek Hızlı Entegre Devre Donanımı Tanımlama Dili

ÖZET

YÜKSEK LİSANS TEZİ

FPGA TABANLI GERÇEK ZAMANLI GÖRÜNTÜ İŞLEME İLE BASKI DEVRE KARTI HATALARININ TESPİT EDİLMESİ

Merve AYDIN

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Elektrik-Elektronik Mühendisliği Programı

Danışman : Prof. Dr. Fırat KAÇAR

Elektronik endüstrisinde kusursuz bir PCB imalatı, tüketici elektroniği ürünlerinin kaliteli olması açısından önem taşımaktadır. Bu sebeple PCB; SMD dizgi makinesi ve reflow fırın sonrası, ürüne monte edilmeden önce denetlenmelidir. PCB denetimi sonucunda SMD dizgi esnasında PCB üzerine takılamayan ve montaj esnasında kırılan komponentler, kritik hata türü olan eksik komponent hatasına sebep olmaktadır. Bu hata türü, görüntü işleme ile tespit edilebilir. Görüntü işlemenin gerçek zamanlı olması, çok sayıda paralel işlem yapılmasını ve yüksek hızda veri aktarımını gerektirdiğinden FPGA kullanımı en iyi alternatiftir. Tez çalışması, LED halka ile oluşturulan aydınlatma koşullarında, FPGA içerisine gömülen VHDL kod sayesinde OV7670 görüntü sensörü tarafından RGB444 formatında ve 320x240 boyutunda alınan PCB görüntüsüne ait eksik komponent hatalarının gerçek zamanlı olarak kullanıcıya VGA monitörde gösterilmesini sağlamaktadır. Ayrıca görüntü alım sürecinde görüntü kalitesini düşüren gürültülerin kaldırılması amacıyla medyan ve gauss filtreleri tasarlanmıştır. FPGA üzerindeki kullanıcı anahtarları değiştirilerek PCB görüntüsünün filtreleme seçimi, filtrelemeden geçirilme sayısı ve üzerinde yer alan eksik komponent hataları monitörde gösterilmektedir. Sonuç olarak eksik komponent hatalarını göz kontrolü ile denetleyen operatör ve personele gerek kalmadan üretimde insan hatasının önüne geçilmesi, elektronik ürün üretimi esnasında yapılan elektriksel kontrollerin yapılmadan hatasız üretimin gerçekleşmesi ve üretime vakit kazandırılarak üretim maliyetinin azaltılması sağlanır.

Haziran 2022 , 137 sayfa.

Anahtar kelimeler: Gerçek Zamanlı Görüntü İşleme, FPGA, Filtreleme, PCB Hata Tespiti.

ABSTRACT

M.Sc. THESIS

DETECTION OF PRINTED CIRCUIT BOARD FAULTS WITH FPGA – BASED REAL-TIME IMAGE PROCESSING

Merve AYDIN

İstanbul University-Cerrahpaşa

Institute of Graduate Studies

Department of Electrical and Electronics Engineering

Electrical and Electronics Engineering Program

Supervisor : Prof. Dr. Fırat KAÇAR

In the electronics industry, a flawless PCB manufacturing is essential for the quality of consumer electronics products. For this reason, the PCB should be inspected before being mounted on the product after the SMD typesetting machine and the reflow oven. As a result of PCB inspection, components that cannot be mounted on the PCB during SMD typesetting and are broken during assembly cause missing component failure, which is a critical failure type. This type of failure can be detected by image processing. The use of FPGA is the best alternative since the image processing is real-time requires a lot of parallel processing and high-speed data transfer. The thesis study provides that the missing component failures of the PCB image taken by the OV7670 image sensor in RGB444 format and 320x240 size are displayed on the VGA monitor in real time to the user, thanks to the VHDL code embedded in the FPGA, in the lighting conditions created by the LED ring. In addition, median and gaussian filters were designed to remove the noise that reduces the image quality during the image acquisition process. By changing the user switches on the FPGA, filtering selection of the PCB image, the number of filtering and missing component failures on it are displayed on the monitor. As a result, it is ensured that human error is prevented in production without the need for operators and personnel who visually inspect missing component failures, faultless production is realized without electrical controls during the production of electronic products, and production cost is reduced by saving time for production.

June 2022, 137 pages.

Keywords: Real Time Image Processing, FPGA, Filtering, PCB Fault Detection.

1. GİRİŞ

Elektronik endüstrisinde tüketici elektroniği ürünlerinin içerisinde bulunan baskı devre kartları (PCB), ürün kalitesini etkilediği için önem taşımaktadır. Bu sebeple ürün içerisinde bulunan PCB'lerin imalatının kusursuz olması gerekmektedir. PCB üretiminin denetlenme süreci üç kategoride sınıflandırılabilir. Bu süreçler; baskının oluşturulduğu, komponentlerin PCB yüzeyi üzerine yerleştirildiği ve komponentlerin PCB üzerine lehimlendiği süreçlerdir [1]. Tüm süreçler tamamlandıktan sonra PCB üzerinde kritik hataların ve kalite hatalarının test edilmesi gerekir.

Üretim esnasında ve bitiminde PCB kalitesini denetlemek için göz kontrolü yapan operatörler sıkça kullanılmaktadır. Operatörler, sağlanan standartlara göre tüm üretilen PCB'leri görsel olarak denetler. Operatörlerin karmaşık bir PCB'deki çeşitli hataları bulmak için genellikle PC tabanlı mikroskop veya büyüteç kullanması gerekir. Manuel olan bu yöntem; vakit kaybı, yüksek işçilik maliyeti ve düşük verimliliği nedeniyle PCB hata tespiti için uygun değildir [2, 3]. Operatörler tarafından verilen kararlar, öznel yargı içerdiğinden farklı operatörler tarafından farklı görüşler içerebilmektedir. Ayrıca, çok katmanlı PCB'ler insan gözünün incelemesine uygun değildir ve üretim oranları çok yüksektir. Bu sebeple operatörlerin hatayı manuel denetimle kesin olarak tespit etmesi mümkün olmamaktadır. Manuel denetim yerine yapay görme teknolojisi kullanılarak denetimin yapılması, üretim sürecini hızlandırma ve verimliliği arttırma açısından uygundur [4].

Yapay görme teknolojisinde yaygın olarak kullanılan otomatik optik inceleme (AOI) sistemleri, PCB'lerin kamera ile tarandığı, otomatik görsel denetimi sağlamaktadır. AOI, üretim hatlarında endüstriyel ürünlerdeki komponentlerin yakalanması, analiz edilmesi ve sınıflandırılmasında önemli bir rol oynamaktadır [5, 6, 7]. İnsan operatörlerin öznel yargılarını ortadan kaldırdığı ve hızlı, nicel değerlendirmeler sağlayarak hataları tespit ettiği için önemlidir. Temassız bir test yöntemi olması nedeniyle üretim sürecinde yaygın olarak kullanılmaktadır. Lehim köprüsü ve eksikliği, eksik komponentler, komponente ait kalkmış bacak uçları, mezar taşı (tombstone) ve lehim topları dahil olmak üzere manuel inceleme ile aynı tipte yüzeyle ilgili hataları tespit eder. Ayrıca üzerinde komponent olmayan, boş PCB için lehim pastası denetimi (SPI), çizgi genişliği

ihlalleri, aralık ihlali, fazla bakır, devre kartında olması gereken padin eksik olması, kısa devreler, kesikler, via deliğinin dışındaki deliklerin kırık olmasını da denetlemektedir [8].

Komponentlerin PCB üzerine yerleştirildiği aşamada SMD (Yüzey Montaj Elemanları) dizgi makinesi kullanılır. Dizgi makinesinde SMD komponentleri makarasından (feeder) alarak, vakumla tutan ve PCB üzerine yerleştiren nozzle (kafa) uçları bulunur. Nozzle uçlarının, komponentleri PCB'ye doğru şekilde yerleştirememesi eksik ve yamuk komponent gibi PCB hatalarına sebep olabilir. Komponentleri doğru şekilde yerleştirememesinin sebepleri; nozzle ucunun hasarlı olması, makarada koruyucu paketin hasarlı olması, makaradaki komponentin hasarlı olması, makaraya yanlış yönde konumlanmış komponent olması, dizgi makinesine girilen bilgilerde yanlış konum girilmesi veya komponentin nozzle'a takılı kalması örnek olarak verilebilir [9]. Bunların dışında montaj esnasında komponentin kırılması sonucu eksik komponent hatası oluşabilir. Tüm bu hata tespit işlemlerinde görüntü işlemenin kullanılması işlemleri kolaylaştırmaktadır.

Görüntü işlemenin gerçek zamanlı olması, yüksek hızlarda fazla sayıda veriyi aynı anda işlemeyi gerektirmektedir. Daha hızlı ürün geliştirmek, ihtiyaçlara özel çözümler üretmek adına pek çok elektronik araç geliştirilmiştir. Bu araçlardan biri olan FPGA'ler, dijital tasarımların donanımsal olarak tek bir entegre üzerinde gerçekleştirilmesini sağlamaktadır. Aynı FPGA ile çok basit bir lojik devre tasarımından, merkezi işlem birimi (CPU) – mikrodenetleyici (MCU) gibi karmaşık tasarımlara kadar çok geniş alanda çalışma yapma imkanı mevcuttur [10]. Bu yüzden; düşük güç tüketimi, yüksek hızlarda çalışması, paralel çalışma özelliği ile fazla sayıda veriyi aynı anda işlemesi sebebiyle FPGA kullanılması avantaj sağlamaktadır.

Tez çalışmasında kavramsal çerçeve bölümünde; FPGA, donanım tanımlama dilleri, sayısal görüntü işleme yöntemleri, RGB ve HSV görüntü formatları, görüntü sensörleri, medyan filtreleme, gauss filtreleme ve VGA teknolojisinden bahsedilmiştir.

Yöntem bölümünde, PCB görüntülerini almak için oluşturulan donanımsal sistemde kullanılan ZedBoard FPGA ve OV7670 görüntü sensörü özeliklerine yer verilmiştir. Sisteme ait tüm donanımsal bağlantılara, Vivado programında yapılan pin atamalarına, yazılan VHDL (Çok Yüksek Hızlı Entegre Devre Donanım Tanımlama Dili) kodlarının modül sınıflandırılmasına, tüm bu modüllere ait kod parçaları ve kodların çalışma mantığına yer verilmiştir.

Bulgular bölümünde; görüntü alımı, medyan ve gauss filtrelemeler ve eksik komponent hata tespiti sonucunda VGA monitör çıktıları gösterilmiştir. Ayrıca medyan, gauss filtreleme simülasyon sonuçlarına yer verilmiştir. Eksik komponent tespiti için RGB (kırmızı yeşil mavi) değerlerinden HSV (renk özü, yoğunluğu, parlaklığı) değerlerinin elde edilmesi için oluşturulan simülasyon sonuçları açıklanmıştır. Kaynak kullanımları ve zaman analizi eklenmiştir.

Çalışma kapsamında, FPGA ile görüntü işleme yapılarak kritik hata türü olan eksik komponent hatasının PCB üzerinde gerçek zamanlı bir şekilde tespit edilmesi sağlanmıştır. Hata tespiti için donanımsal bir sistem oluşturulmuştur. Donanımsal sistemde; Xilinx Zynq-7000 SoC (yongada sistem) içeren AVNET firmasının ürettiği ZedBoard FPGA geliştirme kartına bağlı OV7670 kamera sensörü ile I2C (entegre devreler arasında) haberleşme sağlanarak RGB555 renk uzayına sahip görüntü gerçek zamanlı olarak alınır. Görüntü alma sürecinde görüntü kalitesini düşüren gürültü bileşenleri oluşmaktadır. Gürültü bileşenlerini kısmi olarak giderme amacıyla yatay ve dikey yönü ayarlanabilir LED aydınlatma halkası kullanılmıştır. Buna ek olarak görüntülerin filtrelenerek ön işleme aşamasından geçirilmesi ve görüntü kalitesinin iyileştirilmesi gerekmektedir [11]. Bunun için kaskad bağlı medyan ve gauss filtreleme yapılarak gürültünün kaldırılması sağlanır. PCB üzerindeki eksik komponente sahip olan kısımlar 320x240 boyutunda ve RGB444 formatında VGA monitörde kullanıcıya gösterilir. Yapılan tüm işlemlerde, VHDL kodları Vivado programında sentezlenerek FPGA içerisine PC vasıtası ile gömülmüştür.

Tez çalışması, kritik hatalı komponentler için göz kontrolü yapan personel ve operatöre gerek kalmadan üretimde insan hatasının önüne geçilmesini, ayrıca ürün üretimi esnasında yapılan kontrollerin yapılmadan üretimin hatasız gerçekleştirilmesi ve üretime vakit kazandırılarak maliyetin azaltılmasını amaçlamaktadır. FPGA ile tasarlanan bu sistem, tartışma bölümünde yer alan diğer sistemler ile karşılaştırıldığında hız, alan tüketimi ve güç açısından daha verimlidir.

2. KAVRAMSAL ÇERÇEVE

2.1. ALANDA PROGRAMLANABİLİR KAPI DİZİLERİ (FPGA)

2.1.1. FPGA Tanımı ve Özellikleri

FPGA; bir veya daha fazla mantıksal işlemi gerçekleştirmek için içerisinde bulunan mantık kapılarını birbirlerine bağlayarak kullanıcıya programlama imkanı sunan tümleşik donanım devresidir.

FPGA'lerin sahada programlanabilmesi, paralel işlem yapabilmesi, tekrar tekrar programlanabilmesi, tasarımın sentezlenerek test edilip doğrulanabilmesi, düşük güç tüketimi gibi birçok özelliği mevcuttur.

FPGA adındaki "alanda programlanabilir" ifadesi, programlamanın sahada gerçekleştiğini yani kullanıcı tarafından istenen fonksiyona göre donanım yapısının değiştirilebilir olduğunu göstermektedir [12]. FPGA, ürün sahada kurulduktan sonra bile ürün özelliklerini ve işlevlerini programlamaya, yeni standartlara uyum sağlamaya ve donanımı belirli uygulamalar için yeniden yapılandırmaya olanak tanımaktadır.

FPGA kullanımının diğer entegre devreler ile kıyaslanmasına örnek olarak araba için tasarlanmış bir arka görüş kamerasının durumu verilebilir. Görüntü sensörünün görüntüyü gördüğü andan görüntü çerçevesi ekranda görünene kadar geçen süre 250 milisaniye olabilir. Yönetmelik bu zaman aralığının 100 milisaniyeye düşürülmesini istediğinde tüm araçlar için değişiklik yapılması gerekecektir. Eğer kamera mikroişlemci tabanlı bir çözüme dayalı olarak çalışıyorsa, maliyetli ve neredeyse imkansız değişiklikler gerektirecektir. Yeniden donanım tasarlanması ve işlemcinin yenilenmesi yerine FPGA kullanılması avantaj sağlayacaktır. Böylece üretimdeki arabalar, satılmamış arabalar ve halihazırda satılan arabalar, FPGA'in basit bir yeniden programlanmasıyla güncellenebilir [13].

FPGA'in aynı saat sinyali esnasında birden fazla işlem yapması, çalışmasını paralel olarak gerçekleştirdiğini göstermektedir. Bu özelliği ile diğer entegre devrelerden daha hızlı çalışma sağlamaktadır. Tez çalışmasında görüntü işleme yapıldığından FPGA'in bu özelliği önem taşımaktadır. Gerçek zamanlı sistem tasarımında görüntü akışı sürekli olduğunda bir görüntü

çerçevesi alınırken aynı anda diğer görüntü çerçevesine filtreleme işlemi yapılması örnek verilebilir. Bu durum diğer entegre devrelerde sırayla gerçekleşmektedir [14]. Alınan görüntü çerçevesine filtreleme yapıldıktan sonra diğer görüntü çerçevesinin alınması başlamaktadır. Böylece görüntü işlemede elde edilen verilerin kısa sürede işlenmesini sağlamaktadır. Ayrıca görüntü işleme alanında örnek verilecek olursa 640x480 boyutundaki bir görüntü, 307.200 piksel sayısına sahiptir. Gerçek zamanlı bir sistem olduğunda saniyede 60 görüntü çerçevesinin gösterilmesi için işlemlerin sırayla yapılması sistemi yavaşlatmaktadır. FPGA tabanlı, donanımsal olması paralel bir şekilde tüm pikselleri bir saat sinyali ile aynı anda işleme olanağı sunmaktadır.

FPGA'ler, tekrar tekrar programlanabilmesi sayesinde tasarım geliştirilirken aynı zamanda bu tasarımı kontrol edebilme imkanı sunar. Örneğin tasarlanan donanımın FPGA'e yüklenmesi aşamasından önce bu tasarım, simülasyon ile entegre mantık analizörü (ILA) sayesinde hata ayıklanarak test edilerek doğrulanabilir.

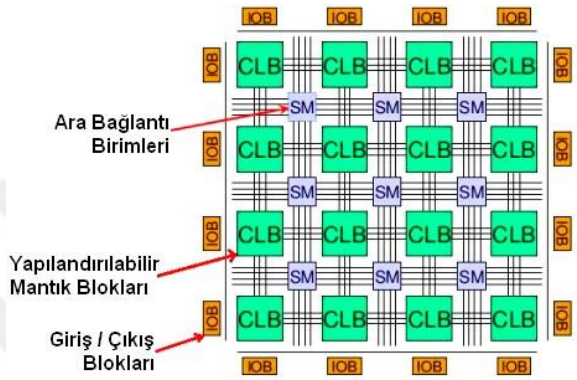
FPGA ile senkron tasarımdaki birimlerin eş zamanlı olarak çalışması, tüm sistemin bilgisayar mimarisine göre daha düşük frekanslarda çalışma imkanı sunmaktadır. Örnek verilirse 100-200 MHz civarındaki çalışma frekansında GHz seviyesindeki işlemciler ile aynı seviyede işlem gücünü göstermektedir. Böylece FPGA'in düşük güç tüketimi özelliği sayesinde enerji kaynağının sınırlı olduğu ve gerçek zamanlı olarak veri alınması gereken alanlarda tercih edilmesi doğru olur. Ayrıca dijital bir entegrenin yapabileceği tüm işlevleri yapacak biçimde tasarım imkanı sağlamaktadır. Böylece devre kartları daha küçük yapıda tasarlanabilirler [10]. Bu durum gerçekte alandan tasarruf edilmesini sağlamaktadır.

2.1.2. FPGA Yapısı ve Blokları

FPGA'ler yapı olarak Durağan Rastgele Erişimli Bellek (SRAM), giriş/çıkış blokları (I/O'lar), programlanabilir veya yapılandırılabilir mantık blokları, yönlendirilebilir ara bağlantılardan oluşan sayısal tümleşik devrelerdir.

Şekil 2.1'de gösterildiği gibi FPGA, genellikle giriş/çıkış bloklarıyla çevrelenmiş 2 boyutlu bir Yapılandırılabilir Mantık Blokları (CLB) dizisidir. Yapılandırılabilir mantık blokları, FPGA içerisindeki fonksiyonların gerçekleştirildiği yapılardır. Giriş/çıkış blokları ise dış çevre ve FPGA arasındaki bağlantıyı gerçekleştiren pinlere verilen isimdir. Tasarıma göre sadece giriş, sadece çıkış ve giriş/çıkış birlikte şeklinde yapı oluşturma imkanı sağlamaktadır. Yapılandırılabilir mantık bloklarını bağlamak için yatay ve dikey kanallar kullanılır. Yatay ve

dikey kanalların kesitlerine yerleştirilen ara bağlantı birimleri olan Anahtar Blokları (SB), gerekli yönlendirmeyi düzgün yapacak şekilde konfigüre edilebilir [15]. Kullanıcı, FPGA içerisinde bulunan transistörleri birbirlerine bağlayarak yeniden yapılandırılabilir ara bağlantılar ile belirli bir hiyerarşi sisteminde gerçekleştirmektedir. Ara bağlantıların açık veya kapalı durumda olması ile fonksiyonlar aktif edilmektedir. Kısa hatlar ile iç bağlantı yapılırken, uzun hatlar ile global olarak set/reset veya saat sinyallerinin uygulanması sağlanır.



Şekil 2.1: SRAM tabanlı FPGA mimarisi [15].

SRAM tabanlı FPGA programlanırken mantık kapılarının kontrol edilmesi için küçük statik bellekler kullanılmaktadır. FPGA'ye verilen gücün kesilmesi veya cihazın yeniden programlanmasına kadar geçen kısımda bellekte yer alan veriler sabit kalır. Bu sebeple statik bellek ismini almaktadır. Tekrar tekrar programlanma özelliği taşıyan bellek türüdür. Güç kaynağına ihtiyaç duyarak çalışmaktadırlar. Gücün kesilmesi ile FPGA içerisindeki kayıtlı program silinmektedir.

FPGA'de alıcı/verici blokları ve RAM (Rastgele Erişimli Bellek) blokları gibi çeşitli bloklar mevcuttur. Alıcı/verici blokları, kullanıcıya FPGA'lerin dış dünya ile bağlantısını yapma imkanı sunar. Örneğin paralel olarak çarpma işlemlerini gerçekleştiren DSP (Dijital Sinyal İşleme) blokları; seri-paralel dönüştürücüler; giriş saatinin ayarlanmasına bağlı olarak farklı frekansta saatlerin üretimini sağlayan, tasarımda kullanılacak olan osilatör sayısını ve saat pini sayısını azaltan faz kilitli döngü (PLL) modülleri; verileri ara belleğe alan FIFO (İlk Giren İlk Çıkar) ve saat üreteçleri ile alıcı/verici blokları sayesinde donanıma ulaşma imkanı olur. Alıcı/verici bloklar; fiber optik, Ethernet, PCI Express gibi ara yüzlerle veri alışverişi imkanı sunmaktadır. RAM blokları sayesinde bellek üzerinde okuma/yazma işlemleri gerçekleştirilir. Bu bloklar ayrıca farklı saatlerde çalışan bloklar arasındaki veri aktarımı için kullanılır.

2.1.3. FPGA Pinleri ve Modları

FPGA'in dış elemanlar ile bağlantısının sağlanması; ayrılmış pinler, kullanıcı pinleri ve özel fonksiyon pinleri ile sağlanır [16]. Tasarımda yapılan pin atamaları ile oluşturulan portlar, FPGA pinlerine bağlanır.

Ayrılmış pinler, FPGA'de tasarım için özel fonksiyonların gerçeklenmesi içindir. Bu kategoride; güç pinleri, FPGA'in programlanması için konfigürasyon pinleri, saat sinyalinin bağlantısı için kullanılan saat pinleri gibi pinler yer almaktadır.

Kullanıcı pinleri, tasarımcının isteğine bağlı giriş/çıkış olarak değişim sağladığı pinlerdir. Bu pinler, FPGA'de giriş/çıkış bloğuna bağlıdır ve güçleri VCCO pini ile sağlanmaktadır. Kullanıcının isteğine bağlı olarak voltaj aralığı 1.2, 1.5, 1.8, 2.5 ve 3.3V gibi tasarımda ayarlanmaktadır.

Saat ile ilgili beslemelerin FPGA'de yer alan saat pinleri ile yapılması gerekmektedir. Tasarımın senkron olarak gerçekleştirildiği FPGA'de mevcut flip floplar, aynı anda ve tek bir saat sinyali ile tetiklenmektedir. Bu şartın sağlanmadığı durumda tasarımda zamansal problem oluşmaktadır. Global iç bağlantı sayesinde flip flopların tamamına aynı anda ulaşılarak tek bir saat sinyali uygulanmaktadır.

FPGA mod kısmında, konfigürasyon ve kullanıcı modları olmak üzere iki çeşit mod türüne sahiptir. FPGA'e güç verildiğinde konfigürasyon modundadır. FPGA bu modda programın yüklenmesi için bekleme yapmaktadır. Bu sebeple bütün çıkışlar pasiftir. FPGA programlandıktan ve pin atamaları yapıldıktan sonra FPGA kullanıcı modu durumuna geçer. Pinler aktif haldedir. Bu mod ile FPGA tasarımdaki şartları sağlamaktadır.

2.1.4. FPGA Avantajları

FPGA, donanım tasarımı için kullanılan diğer entegre devreler olan programlanabilir mantık cihazlarına (PLD'lere), uygulamaya özel entegre devrelerine (ASIC'lere), uygulamaya özel standart ürünlere (ASSP'lere) göre daha avantajlıdır.

Programlanabilir mantık cihazları (PLD'ler), iç mimarisi üretici tarafından önceden belirlenmiş, ancak çeşitli farklı işlevleri yerine getirmek için sahadaki mühendisler tarafından yapılandırılabilecek şekilde oluşturulmuş cihazlardır. Yüz milyonlarca mantık kapısı içeren ve karmaşık işlevler oluşturmayı sağlayan ASIC ve ASSP ile karşılaştırıldığında bu cihazlar sınırlı

sayıda mantık kapısı içerir ve uygulamak için kullanılabilecekleri fonksiyonlar çok daha küçük ve basittir [12].

ASIC'ler ve ASSP'ler aynı tasarım süreçlerine ve üretim teknolojilerine dayanmaktadır. ASIC, belirli bir kullanım için özelleştirilmiş silikon veya diğer yarı iletken malzemeden oluşan bir levha üzerinde üretilen, transistörler, kapasitörler ve dirençler gibi elektrik bileşenlerinden oluşan entegre bir devredir [13]. Boyut (transistör sayısı), karmaşıklık ve performans açısından en üst düzeyi sunsalar da, tasarlamak ve inşa etmek, son tasarımın kalıcı olarak "silikon içinde donmuş" olması ve değiştirilememesi gibi dezavantajla birlikte, son derece zaman alıcı ve pahalı bir süreçtir [12]. İşlevleri üretim sırasında kablolu olduğundan, devrenin küçük bir parçasının bile işlevselliğini değiştirmek kolay değildir. Tasarımda bir işlevin değiştirilmesi gerekiyorsa, tüm çipin çıkarılarak yeniden başlanması gerekir [13].

FPGA'ler, PLD'ler gibi sahada özelleştirilebilirler. PLD'lerden farkı, milyonlarca mantık kapısı içerebilir olmasıdır. Ayrıca ASIC'ler kullanılarak gerçekleştirilebilen son derece büyük ve karmaşık işlevleri uygulamak için kullanılabilirler. FPGA tasarımının maliyeti, ASIC'den çok daha düşüktür ve tasarım değişikliklerinin uygulanması çok daha kolaydır. FPGA'ları piyasaya sürme süresi çok daha hızlıdır. Bu özellikleri ile FPGA avantaj sağlamaktadır [12].

2.1.5. FPGA Programlanması

FPGA programlama, istenen bir algoritmayı yürüten ve bu algoritmayı donanım tanımlama dilinde (HDL) tanımlayarak yeniden yapılandırılabilir bir donanım mimarisinin oluşturulmasını sağlar.

FPGA donanım tasarımı için VHDL, Verilog HDL yardımıyla programlama yapılabilir. HDL dışında C, C++, SystemC gibi programlama dilleri ile Yüksek Seviyeli Sentez (HLS) kullanılarak FPGA için donanım oluşturulabilir. Ayrıca Mathworks'ün Matlab HLS sayesinde de donanım tasarımı sağlanmaktadır.

Xilinx ve Altera gibi farklı FPGA üreticilerinin FPGA mimarileri farklı olsa da aynı HDL ile programlanabilirler. Fakat donanım tasarımı için IP (Fikri Mülkiyet) blok kullanımı gerektiren özel tasarımlarda kullanılan marka mimarisine göre IP blok tasarımının değişmesi gerekmektedir.

2.2. DONANIM TANIMLAMA DİLİ

Dijital tasarımcılar için donanım tanımlama dilleri, tasarımın oluşturulması için önemli araçlardır. SystemVerilog veya VHDL kullanılarak tüm şemaların çıkartılması ile dijital sistemler oluşturulur. Tasarımda yapılacak değişiklikler şematik üzerinden yeniden kablolama yapmak yerine kod değişiklikleri ile sağlanır. Böylece hata ayıklama döngüsünde hız kazandırmaktadır.

HDL'ler hem sentezleme hem de simülasyon için kullanılmaktadır. Simülasyon kısmı, bir sistemi donanım haline dönüştürme işleminden önce bilgisayarda test etmeyi sağlamaktadır. Simülasyon sayesinde, ölçülmesi zor olan sinyallerin değerleri kontrol edilebilmektedir. Mantık sentezi ile HDL kodu temel alınarak sayısal mantık devrelerine dönüştürülmektedir. Tasarıma başlamadan önce tasarlanacak donanım parça parça düşünülmelidir. Sistemin her bölümü için HDL kodu yazılarak sistem bütün hale getirilir. FPGA tasarımının derleyici programa girilmesi ilk aşamayı oluşturur. Bu aşamada donanım tanımlama dilleri veya şematik yöntem kullanılabilir.

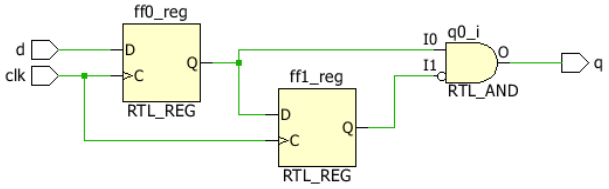
2.2.1. Donanım Tanımlama Dili Seviyeleri ve Özellikleri

HDL kodu; davranışsal seviye türü, RTL (register transfer level) seviye türü, mantıksal seviye türü ve donanım düzeni seviyesi türü olmak üzere dört şekilde yazılmaktadır.

Davranışsal seviye türü, tasarım davranışını algoritmaları kullanarak sistem veya devre davranışı açısından tanımlamaktadır. Bu sebeple kullanılan dil yapısı üst düzey yazılım programlama diline benzemektedir [17]. Zamansal süreçler kullanılmadığı için bu seviyede sentezleme işlemi yapılamaz.

RTL (register transfer level) seviye türü; fonksiyonel birimler (toplayıcı, karşılaştırıcı gibi), depolama birimleri (register gibi) ve veri seçici bileşenlerden (çoklayıcı gibi) meydana gelmektedir. Bu yapıda ortak saat sinyali kullanılarak veri sinyallerinin belirli bir saat periyodunda değer almaları sağlanır. Veri sinyalleri, saatin düşen veya yükselen kenarında örneklenebilir. Bu sebeple anlık atlamalar ve sinyal gecikmeleri sistemin çalışmasına etki etmemektedir.

Şekil 2.2’de RTL seviye türünde tasarlanan D tipi flip flop örneği verilmiştir.



Şekil 2.2: RTL seviye türü tasarım.

Mantıksal seviye türü, devre yapısını oluşturan mantık kapıları ile devre ağ listesi (netlist) oluşturması için mantık kapıları arasında ara bağlantı kabloları kullanılarak oluşturulur.

Donanım düzeni seviyesi türü, yapılan tasarımın mantık bloklarına fiziksel olarak yerleştirildiği seviye türüdür. Mantık blokları bağlantıları yapılarak doğrulanır ve donanım devresi oluşturulur.

Bir donanım tasarımındaki mantık blokları birleşerek büyük fonksiyonel blokları oluşturmaktadır. Bu fonksiyonel blokların her bir işlev için daha küçük parçalara ayrılarak parça parça modellenmesi donanım tanımlama dilinin modüler olma özelliği ile sağlanır. Bu modüler tasarım özelliği ile tasarımın karmaşıklığı azaltılmaktadır. Böylece donanımın her bir işlevi için bağımsız olarak birden fazla tasarımcı görev alabilir. Simülasyonda kolaylık sağlayan bir özelliktir. Ayrıca donanımsal tasarım bu şekilde hiyerarşik düzen sağlar. Bu düzene göre, tasarım alt modüllerden oluşur. Alt modüller ile tasarımı basitleştirilir ve daha kolay yönetilebilir. Bu düzeni oluşturmak için kütüphane, bileşen ve paket yapıları kullanılmaktadır.

2.2.2. VHDL ve Bileşenleri

VHDL, donanım parçasını modelleme amacıyla kullanılan bir tanımlama dilidir. Bu dil, IEEE standardı olarak kabul edilerek ekleme ve düzenlemeler yapılarak sürekli güncellenmektedir. Yapısal bakımdan C, C++ gibi yazılımlara benzemektedir, ancak bu dil ile elektronik devre tanımlamaktadır. Bu sebeple donanım mimarisi tam olarak anlaşılan sentezlenmesi mümkün olan bir HDL kodu yazılması gereklidir.

VHDL bileşenleri; paket bildirimleri, varlık (entity), mimari (architecture), bileşenler (component) olmak üzere dört kısımdan oluşur.

Paket bildirimleri kısmı, VHDL tasarımında kullanılacak kodların işlevsel yapılarını barındırmaktadır. Hazırlanan tasarımların başka tasarımlarda kullanılmak üzere gruplandığı

bildirimlerdir. Paketler; alt programlar, sinyaller, sabitler ve veri türleri gibi VHDL yapılarını bir arada gruplandırmaktadır. Böylece bütün alt tasarım ve modüllere dahil edilebilirler. Aynı ifadelerin tekrar yazılmadan hazır olarak kullanımını sağlarlar.

Kütüphane tanımlamaları ve paket bildirimleri yapıldıktan sonra VHDL kodu, entity (varlık) kısmı ile başlamaktadır. Bu kısımda giriş ve çıkış portları tanımlanmaktadır. Tasarımın dış çevre ile ilişkileri belirlenmiş olur. Her VHDL kodunda yalnızca bir adet entity kısmı bulunur.

Architecture (mimari) kısmında, bir modelin iç yapısını ve davranışları tanımlanmaktadır.

Bileşen (komponent) kısmı ise kullanılması şart olmayan kısımdır. VHDL'in modülerlik özelliğini kullanarak hiyerarşik düzende alt modüllerin kullanılmasını sağlar.

2.3. SAYISAL GÖRÜNTÜ

Görüntü, teoride iki boyutlu bir f fonksiyonunun (x,y) düzlem koordinatları olarak tanımlanabilir. Bu fonksiyon, f (x,y) olarak simgelenirse x ve y noktalarına karşılık gelen genlik değeri, görüntünün gri seviyesini temsil etmektedir. Fonksiyon f, x ve y'nin gri seviyesi değerlerinin ayrık ve sonlu büyüklükte olması bu görüntüyü sayısal görüntü olarak nitelendirmektedir. Video akışı ise, bu sayısal görüntü çerçevelerin ardışık halde gelmesiyle oluşur. Videoyu oluşturan bir çerçeve, bir anda çekilmiş sabit bir görüntüdür. Bu çerçeve satırlardan oluşmaktadır. Bir satır ise, satırdaki sayısı sütun sayısına eşit olan bir dizi pikselden meydana gelir. Piksel; dijital görüntünün en küçük parçasıdır, renkli bir nokta gibi görünür [18].

2.3.1. Sayısal Görüntü Formatları: RGB ve HSV (Hue-Saturation-Value)

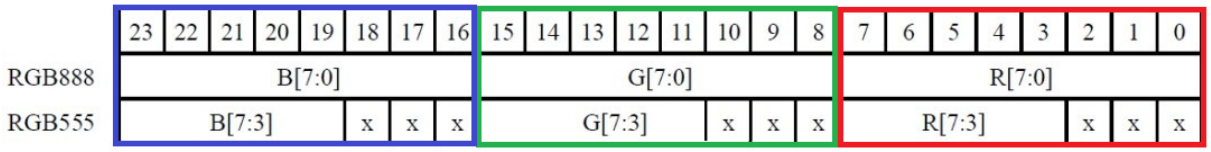
RGB renk formatına sahip bir görüntü pikseli; her bir kanalı 0-255 aralığında değere sahip kırmızı, yeşil ve mavi renk yoğunluğundan meydana gelir. En yaygın biçimi olan RGB888 formatında her piksel 8 bit olmak üzere toplam 24 bit kullanılarak depolanır.

Şekil 2.3'te RGB888 ve RGB555 formatında piksel depolama örneği gösterilmiştir. Buna göre 24 bitten oluşan 1 piksel için RGB888 formatında her bir renk 8 bit olarak gösterilirken RGB555 formatında her bir renk için en anlamlı bit (MSB) kısmından 5 bit alınmaktadır. Kırmızı kanal için 5 bit, yeşil kanal için 5 bit ve mavi kanal için 5 bit olarak depolama işlemi yapılır. Bu format, depolandığında daha az bellek kullanmaktadır, fakat depolanan bit sayısı azaldığından mevcut renk sayısı oldukça azalmış olur. RGB888 için depolanan renk sayıları denklem 2.1'de; RGB555 için depolanan renk sayısı denklem 2.2'de gösterilmiştir. Şekil 2.3'te

RGB formatı piksel depolama örneği gösterilmiştir. X ile gösterilen kısımdaki bitler, veri olarak alınmayan bitlerdir.

$$RGB888 = 2^{8+8+8} = 2^{24\,bit} = 16.777.216\,renk\,sayısı \quad (2.1)$$

$$RGB555 = 2^{5+5+5} = 2^{15\,bit} = 32.768\,renk\,sayısı \quad (2.2)$$



Şekil 2.3: RGB formatı piksel depolama.

HSV uzayı ise RGB renk uzayından daha iyi sonuç alınması, insan görü düzeyine daha yakın olması ve görüntü tanıma uygulamaları için iyi bir alternatif olması sebebiyle kullanılan bir formattır [19]. Bu renk uzayında; hue, renk özünü; saturation, renk doygunluğunu; value, renk parlaklığı gösteren sayısal değerlerdir. Renk özü, baskın olan dalga uzunluğunu yani rengin kırmızı, yeşil ve mavi gibi renklerde olup olmadığı belirlemektedir ve 0° - 360° arasında değişen açısal değere sahiptir. Renk doygunluğu ise, rengin canlı ve yoğun olmasını belirleyen 0-100 arasında sayısal değere sahip olan kısmıdır. Parlaklık, beyazlık oranını, yani renk aydınlığını belirleyen 0-100 arasında değişen değere sahiptir [20].

2.3.2. Sayısal Görüntü İşleme Seviyeleri

Piksel değerlerinin birleşiminden oluşan sayısal görüntülerin, bir sürecin giriş ve çıkışını oluşturması görüntü işleme adımını tanımlamaktadır [19].

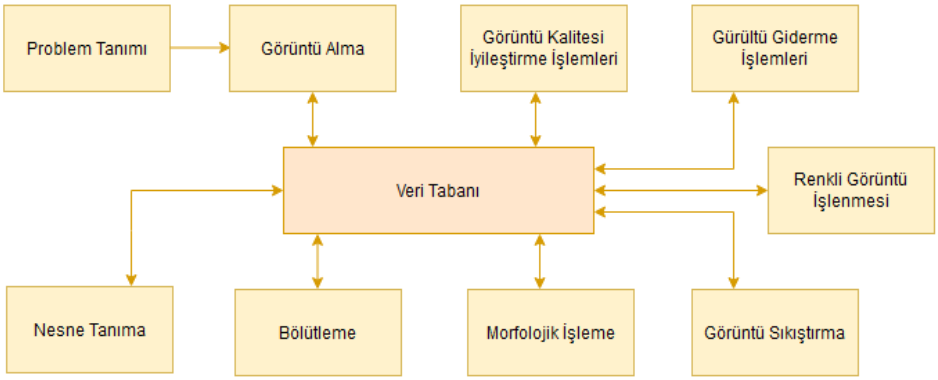
Sayısal görüntü işleme süreçleri seviyeleri; alçak, orta ve yüksek seviyeli olarak sınıflandırılabilir. Düşük seviyeli işleme süreci, görüntü üzerindeki gürültülerin azaltılması için yapılan görüntü ön işleme metodu, kontrast geliştirme ve görüntü keskinleştirme gibi görüntü ön işleme işlemlerini içerir. Bu süreçte hem girdi hem de çıktı görüntülerle karakterize edilir [19]. Orta seviyeli süreçte ise görüntü bölütlemesi ve nesne sınıflandırılması yapılmaktadır. Bu süreçte girdi görüntü olarak verildiğinde çıktı olarak bu görüntüden kenarlar ve öznitelikler elde edilir. Yüksek seviyeli süreçte sınıflandırılan nesne kümeleri ile alakalı görüntü analizi yapılarak görüntüde anlamlandırma yapılmaktadır. Tüm sayısal görüntülerin bilgisayarda verilerinin saklanması, görüntülenmesi ve iletilmesi için bilgisayarda yeterli belleğe ve hesaplama gücüne ihtiyacı vardır.

2.3.3. Sayısal Görüntü İşleme Adımları

Girişi görüntüsünden oluşan bir sistemin çıkışı, görüntü veya bu görüntüden çıkartılan özniteliklerden oluşmaktadır. Şekil 2.4'te sayısal görüntü işlemedeki adımlar gösterilmiştir. Kullanım amacına göre farklı alanlar için şekildeki her bir yöntem ayrı ayrı uygulanabilir.

Bir görüntünün kamera ile yakalanması işlemi olan görüntü alma, görüntü işlemenin ilk adımı oluşturmaktadır. Bu aşama görüntü boyutunun ayarlanması gibi işlemleri de içermektedir.

Görüntü kalitesi iyileştirme işlemi, orijinal görüntünün kontrast ayarlamaları yapılarak görüntüde iyileştirme yapma tekniğidir. Gürültü giderme işlemi, görüntüyü iyileştirme sürecini oluşturur. Filtreleme işlemleri yapılarak görüntü üzerindeki gürültü kaldırılır. Renkli görüntü işlemede renk modelleri ve temel renk işlemeleri kullanılarak sayısal görüntüler ile işlem yapılmaktadır. Görüntü sıkıştırma işlemi, görüntüyü saklama amacıyla ihtiyaç olan depolama alanını sağlama ve görüntü iletmede bant genişliğinin azaltılması için kullanılmaktadır. Morfolojik işleme, görüntü şeklinin gösterilmesi ve görüntü bileşenlerinin çıkarılması ile ilgilenmektedir. Bölütleme ile bir görüntüyü oluşturan tüm parça ve nesnelerin ayrılması yöntemidir. Nesne tanıma, nesneye bir etiket verilerek o nesneyi betimleme yöntemidir.



Şekil 2.4: Sayısal görüntü işleme adımları.

Şekil 2.4'e göre görüntü içerisindeki bilgi, veri tabanına kodlanarak alınmaktadır. Herhangi bir görüntüye blok şemasında gösterilen işlemlerin tamamının uygulanması zorunlu değildir. Görüntünün kullanım amacına yönelik blok şemasından herhangi bir işlem seçilebilir.

2.3.4. Sayısal Görüntü İşleme Sisteminin Elemanları

Sayısal görüntü işlemede kullanılan sistemin elemanları; algılama elemanları, sayısallaştırıcı, bilgisayarlar, görüntü ekranları, özel görüntü işleme donanımı, depolama elemanları, görüntü işleme yazılımı şeklinde özetlenebilir.

Görüntü alma işlemlerinde görüntü algılama elemanları veya diğer adıyla görüntü sensörleri kullanılmaktadır. Görüntülenecek nesneden yayılan enerjiye duyarlı cihaz ve bu cihaza ek olarak bu görüntü çıktısını sayısal biçime dönüştürmek için sayısallaştırıcı kullanılarak görüntü alma işlemi sağlanmaktadır.

Görüntü algılayıcı, ışığın şiddetine bağlı olarak elektriksel çıktı üretmektedir. Üretilen bu elektriksel çıktılar, sayısallaştırıcı sayesinde sayısal forma çevrilmektedir.

Özel görüntü işleme donanım elemanları; sayısallaştırıcıya ek olarak aritmetik işlemleri sağlayan donanımdan oluşabilir. Örneğin görüntü üzerindeki gürültünün kaldırılması için bir donanım tasarımı yapılacaksa görüntüdeki piksellerin ortalamasının alınması gerekebilir. Ortalama alma işlemi için aritmetik lojik birimi (ALU) kullanılarak işlem yapılabilir. ALU, görüntü üzerinde aritmetik ve lojik işlemlerini yapmaktadır. Böylece saniyelik 30 görüntü sayısı (fps) hızındaki veri akışına sahip görüntüler bu şekilde sayısallaştırılır ve ortalamasının alınması sağlanarak gürültü giderme işlemi yapılabilir.

Depolama elemanları, görüntü işlemenin en önemli elemanlarındandır. Fazla sayıda piksel verisi içeren görüntüler, depolama alanını doldurmaktadır. Örneğin piksel başına 3 RGB değeri, her RGB değeri için 8 bitten oluşan bir bayttır. Bu sebeple her bir renk yoğunluğu 8 bit olan 24 bit RGB görüntü boyutu, denklem 2.3'te gösterildiği üzere bellekte sıkıştırılmadığı zaman piksel başına üç bayt alan kaplamaktadır. 1 MB'ın, 1.048.576 bayt'a karşılık gelmesi ile denklem 2.4'te gösterildiği gibi bu görüntü, depolama alanında sıkıştırılmamış 68.7 MB'lık alan kaplamasına sebep olur [21].

$$24 \text{ MP x } 3 \text{ bayt } = 72.000.000 \text{ bayt} \tag{2.3}$$

$$1 \text{ MB x } \frac{72.000.000 \text{ bayt}}{1.048.576 \text{ bayt}} = 68.7 \text{ MB} \tag{2.4}$$

Kısa süreli depolama için çerçeve arabelleği (frame buffer) kullanılabilir. Bu arabellekler birden fazla görüntüyü saklama kapasitesine sahiptir. Görüntüyü video hızında 30 fps olarak saklayabilir.

2.4. GÖRÜNTÜ SENSÖRLERİ

Görüntü sensörü, bir görüntüyü oluşturmak için kullanılan verileri algılayan ve bu verileri elektriksel sinyallere dönüştürerek ileten sensördür. Aktif piksel sensörü (CMOS) ve şarja bağlı

cihaz sensörü (CCD) olmak üzere iki çeşittir. CCD görüntü sensörü giriş verisi analogdur ve CMOS sensörlere göre ışığa karşı daha hassas ve elde edilen görüntünün kalitesi daha iyidir. Fakat hız ve maliyet açısından bakıldığında dezavantajlıdır. Daha çok güç harcamaktadırlar ve verinin sayısal olarak işlenmesi için analog dijital çevirici (ADC) ihtiyacı vardır. CMOS görüntü sensörü verisi ise dijitaldir. CMOS sensör tümdevresinde mikroişlemci bulunur ve görüntü bu mikroişlemcide ön işleme girer. Dijital veri çıkışı olduğundan dönüştürücüye gerek yoktur [17].

CMOS sensörü, milyonlarca piksel sensörü içermektedir. Bu piksel sensörlerinin her birinde bir fotodedektör mevcuttur. Kamerada bulunan lensten ışık geçtiğinde fotodedektör, bu ışığın miktarına göre bir elektrik yükü biriktirir. Kamera bu yükü, fotodedektörde ölçülmüş olan ışığın gücüne bağlı olan bir dijital veriye çevirmektedir. Bu değerler, fotoğraf görüntülemesinde fotoğrafı oluşturan ayrı ayrı piksellere karşılık gelmektedir.

2.5. FİLTRELEME TEKNİKLERİ

Filtreleme işlemlerinin amacı bir görüntü üzerindeki gürültülerin yok edilmesidir. Gürültü, bir sinyalin aktarımı esnasında çevredeki etkilerin altında kalarak oluşan sinyal üzerindeki istenmeyen değişimler olarak ifade edilir. PCB görüntü elde etme, edinme ve iletme sürecinde; ışık yoğunluğunun dalgalanması, fotoelektrik dönüşüm sürecindeki hassas elemanların hassasiyetinin düzensizliği, dijital işlemede kuantizasyon gürültüsü, kamera titreşimi, güç kaynağındaki dalgalanma görüntü kalitesi düşecektir. PCB üzerindeki kir gibi sebeplerden dolayı da giriş dönüştürücü çevresel ortamdan etkilenerek görüntü kalitesini düşürür. Ayrıca Poisson (Shot) gürültüsü, PCB hata tespit sisteminde toplanan dijital görüntünün kaçınılmaz olarak çeşitli gürültüler ve bozulmalar içermesine neden olur. Görüntü analizini ve anlamayı kolaylaştırmak için oluşan gürültüyü ortadan kaldırmak, bozulmayı düzeltmek, görüntü kalitesini iyileştirmek ve sonraki analiz ve işlemeyi kolaylaştırmak için görüntüyü önceden işlemek gerekir [3].

Gürültü giderme; görüntüdeki gereksiz bilgileri ortadan kaldırmak, görüntünün karmaşıklığını azaltmak ve PCB deliklerinin, çizgilerin, padlerin vb. temel özelliklerini vurgulamayı sağlar [22]. Görüntü üzerindeki gürültü; parlaklık ayarlanarak ve renk tonu, keskinleştirme, filtreleme işlemleriyle azaltılabilir. Çalışmada gürültünün giderilmesi için kullanılan filtreleme teknikleri, medyan ve gauss filtrelemedir.

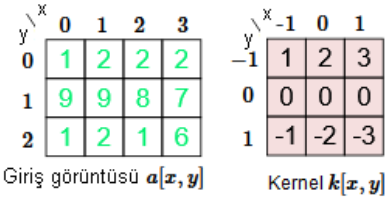
Filtreleme işlemi, görüntü çerçevesindeki her piksel ve bu pikselin çevresinde bulunan komşu pikseller için tanımlanırlar. Bu piksellerin filtreleme türüne göre bir matrisle çarpılması sonucu filtreleme işlemleri gerçekleştirilmektedir. Çarpım işleminde kullanılan matris; kernel (çekirdek), konvolüsyon matrisi veya maske ismini alır. Keskinleştirme, bulanıklaştırma, kabartma ve kenar algılama gibi işlemler için kullanılırlar. Denklem 2.5 ve denklem 2.6'da bir konvolüsyonun genel ifadesi mevcuttur. a[x,y] fonksiyonu, filtreleme işlemine alınacak orijinal giriş görüntüsünü; f[x,y] fonksiyonu, filtrelenen çıkış görüntüsünü; k[x,y] filtreleme kernelini temsil eder.

$$f[x,y] = a[x,y] * k(x,y) \tag{2.5}$$

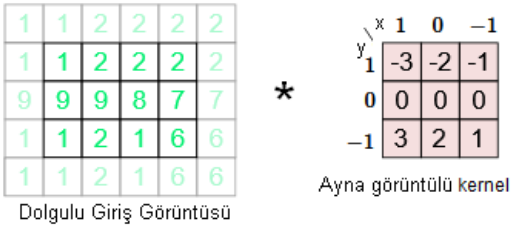
$$f[x,y] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} k\,[m,n]\,.\,a[x-m,y-n] \tag{2.6}$$

k[x,y] kernel matrisine göre görüntü büyük oranda değiştirebilir. 3x3 boyutunda bir kernel ile konvolüsyona alınan giriş görüntüsünde merkez piksel değeri ve bu merkez pikseline 8 adet komşu pikselin değerlerinin kernel matrisi ile çarpımlarının toplamı ile hesaplanmaktadır.

Giriş görüntüsü matrisinin ilk elemanı olan (0,0) elemanı, ayna görüntüsü alınmış kernel matrisinin merkez pikseline denk gelecek şekilde yerleştirilerek konvolüsyon işlemi başlatılmaktadır. Şekil 2.5'te 4x3 boyutunda giriş görüntüsü ve 3x3 boyutunda kernel matrisi gösterilmiştir. Giriş görüntüsünün dış çerçevesine çıkıldığı için dolgu malzemesi (padding) kullanılır. Şekil 2.6'da ise giriş görüntüsüne dolgu, kernel matrisinde ise ayna görüntüsü alma işlemi yapılmıştır.

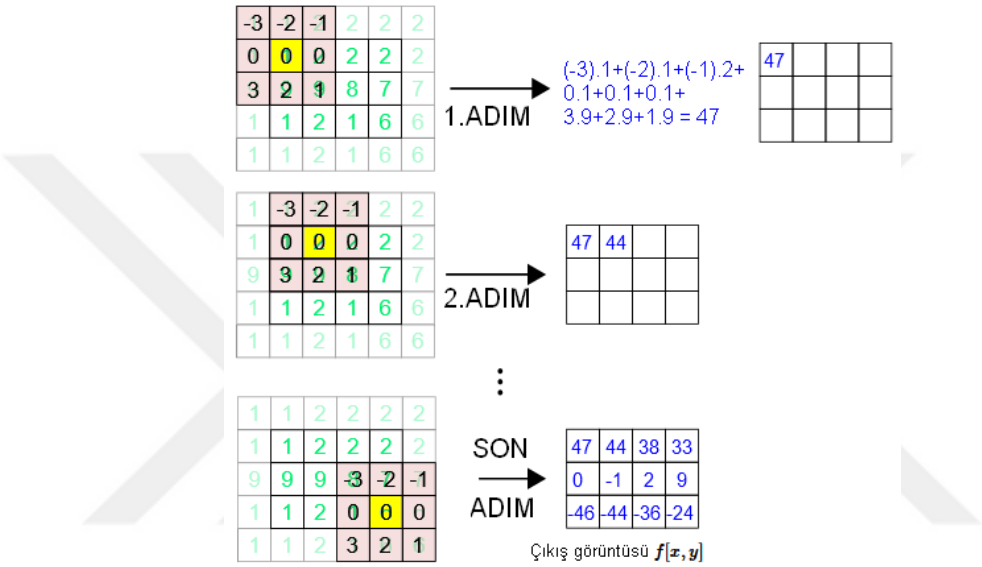


Şekil 2.5: Giriş görüntüsü ve kernel matrisi.



Şekil 2.6: Dolgulu giriş görüntüsü ve ayna görüntülü kernel konvolüsyon gösterimi.

Giriş görüntüsünün ilk elemanı olan “1” değerinin konvolüsyon matrisinin merkez pikseli olan Şekil 2.7’de sarı renk ile gösterilen “0” değerine denk gelmesi için giriş görüntüsü dolgulu olarak kullanılmıştır. Kernel matrisi kayan pencereleme işlemi ile giriş görüntüsüne denk gelen elemanlar ile çarpılır. Çarpılan elemanlar toplanarak giriş görüntüsünün ilk piksel değerinin yerine yazılır. Sağa doğru kayan pencereleme yapılarak her piksel değeri için bu işlem tekrar edilir. En son adımda tüm sonuçların yer aldığı çıkış görüntüsü elde edilerek filtreleme işlemi yapılmıştır.



Şekil 2.7: Kernel matrisi ile konvolüsyon sonucu.

Tablo 2.1’de görüntü işlemlerine göre örnek kernel matrisleri gösterilmiştir.

Tablo 2.1: Görüntü işlemlerine göre örnek kernel matrisleri.

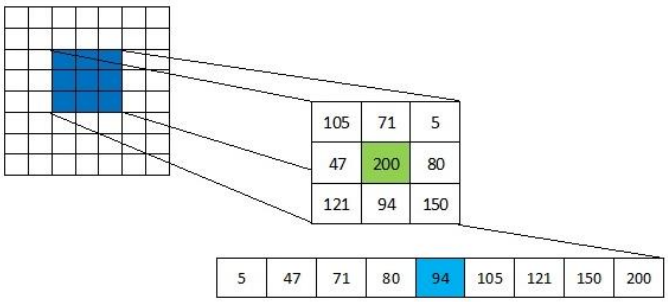
Görüntü İşlemi	Kernel Matrisi	Görüntü İşlemi	Kernel Matrisi
Filtrelenmemiş görüntü işlemi	$\begin{bmatrix}0 & 0 & 0\\0 & 1 & 0\\0 & 0 & 0\end{bmatrix}$	Kabartma işlemi	$\begin{bmatrix}-2 & -1 & 0\\-1 & 1 & 1\\0 & 1 & 2\end{bmatrix}$
Kesinleştirme işlemi	$\begin{bmatrix}-1 & -1 & -1\\-1 & 9 & -1\\-1 & -1 & -1\end{bmatrix}$	Gauss filtreleme işlemi	$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$

2.5.1. Medyan Filtreleme

Medyan filtresi, görüntü üzerinde darbe gürültüsünü bastırmak için kullanılan ve kenarları koruma yeteneği sebebiyle sıkça kullanılan doğrusal olmayan filtrelerden biridir [23]. Görüntü

üzerindeki bir pikselin ve buna komşu piksellerinin medyan değeriyle değiştirerek dürtü gürültüsünü ortadan kaldırır [24].

Çalışma mantığına göre, NxN boyutunda bir kernel matrisi, giriş görüntüsü üzerinde kayarak piksel değerlerini N-1 adet satır arabelleğinde saklamaktadır. Kernel matrisinin içerisinde bulunan piksel değerleri, küçükten büyüğe doğru sıralanır. Sıralandıktan sonra ortadaki değer medyan değeridir. Bu medyan değeri, kernel matrisinin merkez pikselinin giriş görüntüsünde denk geldiği yere atanarak medyan filtreleme gerçekleştirilir. Şekil 2.8'de 7x8 boyutunda bir görüntü üzerinde 3x3 boyutunda bir pencereleme matrisi ile medyan değerinin bulunması gösterilmiştir. Bu duruma göre 9 adet değer, küçükten büyüğe sıralanarak en ortadaki değer olan 94 sayısı medyan değeri olarak alınır ve merkez piksel olan 200 sayısının yerine 94 sayısı yazılır ve bu işlem, giriş görüntüsünün son satırının son pikseline kadar devam ettirilir.



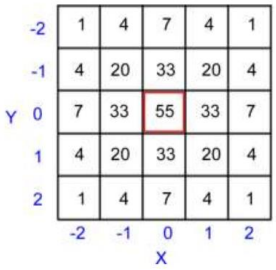
Şekil 2.8: 3x3 kayan pencere ile medyan değerinin bulunması [25].

2.5.2. Gauss Filtreleme

Gauss operatörü, görüntüdeki gürültüyü ortadan kaldırma amacıyla bulanıklaştırma yaparak konvolüsyon işlemini kullanan operatördür. Gauss fonksiyonu denklem 2.7'de gösterildiği gibi ifade edilir. Burada x, yatay eksende orijinden olan mesafe; y, dikey eksende orijinden olan mesafe olarak geçer. σ, gauss dağılımının standart sapmasıdır.

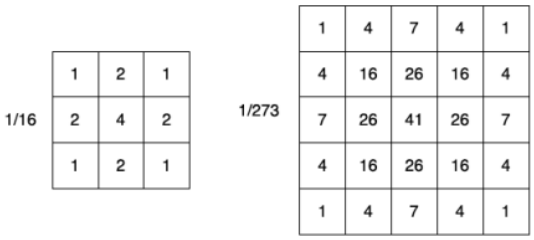
$$g(x,y) = \frac{1}{\sqrt{2.\pi.\sigma^2}} . e^{-\frac{(x^2+y^2)}{2\,\sigma^2}} \tag{2.7}$$

Gauss filtresinin görüntü düzeltme derecesi, standart sapma ile belirlenir. Standart sapma (σ=1) olarak alınır ve x, y için değer aralıkları; {-2, -1, 0, 1, 2} olarak tanımlanırsa, her aralık için gauss fonksiyonu elde edilir. Elde edilen değerler, matris köşe değerlerin '1' olması için ölçeklendirilerek en yakın tam sayıya yuvarlanırsa Şekil 2.9'da gösterilen gauss matrisi oluşturulur [26].



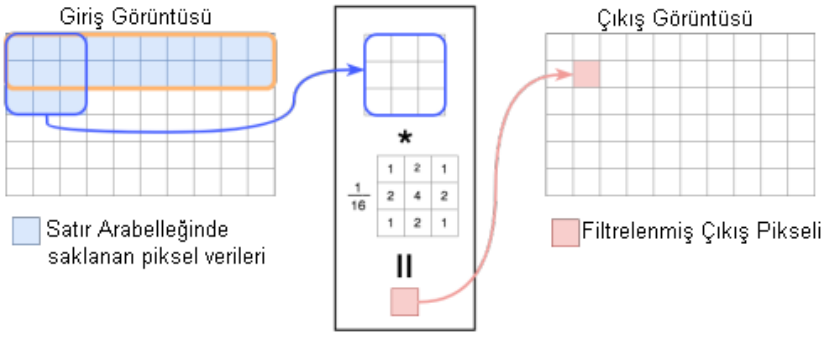
Şekil 2.9: Gauss kernel matrisi [26].

Şekil 2.10'da standart sapma değeri 1 olan, 3x3 ve 5x5 boyutlarında gauss kernel matrisleri gösterilmiştir. Konvolüsyon işlemi için büyük kernel matris (5x5) kullanmak gürültüyü daha fazla yok etmeyi sağlar. Fakat buna rağmen, daha fazla satır arabelleği kullanmaktadır.



Şekil 2.10: Gauss ayrık yaklaşım (σ=1), 3x3 ve 5x5 gauss kernel matrisleri [27].

Çalışma mantığına göre; NxN boyutunda bir kernel matrisi, giriş görüntüsü üzerinde kayarak piksel değerlerini N-1 adet satır arabelleğinde saklamaktadır. Kernel matrisinin içerisinde bulunan piksel değerleri, gauss matrisi ile konvolüsyon işlemine alınmaktadır. Konvolüsyon sonucu tek bir piksel değeri elde edilir. Bu piksel değeri, kernel matrisinin merkez pikselinin giriş görüntüsünde denk geldiği yere atanarak gauss filtreleme gerçekleştirilir. Şekil 2.11'de 10x6 boyutunda bir görüntü üzerinde 3x3 boyutunda bir pencereleme matrisi ve gauss konvolüsyon matrisi ile gauss değerinin bulunması gösterilmiştir. Bu duruma göre 9 adet değer, gauss matrisi ile konvolüsyon işlemine alınarak gauss değeri elde edilir ve bu işlem, giriş görüntüsünün son satırının son pikseline kadar devam ettirilir.



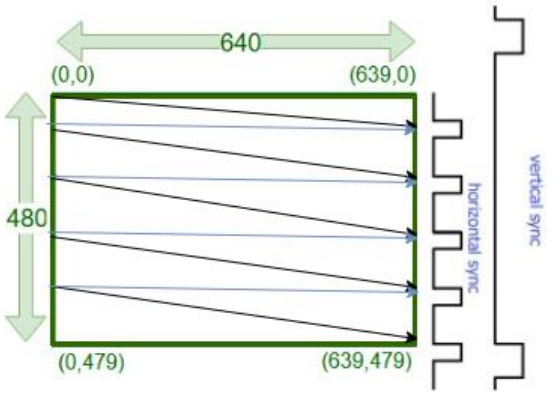
Şekil 2.11: Görüntü üzerinde gauss matrisi ile konvolüsyon işlemi [27].

2.6. VİDEO GRAFİK DİZİNİ (VGA) TEKNOLOJİSİ

2.6.1. VGA Sinyalleri

Görüntü teknolojisi olan VGA, gerçek zamanlı görüntülerin monitör üzerinden gösterimini sağlamaktadır. Görüntülerden oluşan çerçeve akışını meydana getirmektedir. Her çerçeve bir dizi yatay çizgilerden oluşmaktadır. Satır olarak adlandırılan bu yatay çizgiler ise bir dizi piksellerden oluşmaktadır. Bir görüntünün oluşabilmesi için monitörün görüntü alanındaki her çerçeve yukarıdan aşağıya doğru Şekil 2.12'deki gibi satır boyunca yatay çizgide taranmaktadır. Her satırdaki pikseller soldan sağa iletilir.

VGA monitörü sürmek için 5 aktif sinyal içeren VGA video sinyali oluşturmak gerekmektedir. Bunlardan ikisi; görüntü senkronizasyonun sağlanmasında her satırın ve çerçevenin başlangıç ve bitişinin tanımlanabilmesi için kullanılan dijital sinyaller olan yatay (HSYNC) ve dikey (VSYNC) senkronizasyon sinyalleridir. Bir görüntü çerçevesinin piksel verilerinin ne zaman iletildiğini gösteren referans noktalarını gösteren sinyallerdir.



Şekil 2.12: VSYNC ve HSYNC sinyalleri ile görüntü üzerinde piksel taraması.

Diğer sinyaller ise kırmızı, yeşil ve mavi rengi kontrol etmek için kullanılan RGB analog sinyalleridir. Her piksel konumunda, pikselin rengini kontrol etmek için video sinyalindeki RGB renk bilgisi kullanılır. Üç RGB sinyalinin analog seviyeleri değiştirilerek diğer tüm renkler üretilir.

VGA monitör ekranı her görüntü çerçevesinde yenilenmektedir. Bu yenileme işlemi en üst sol üst köşede bulunan piksel (0,0) ile başlatmaktadır. Görüntü sensörü 8 bitlik veriyi iki kez attığında bir piksel elde edilmektedir. Her piksel alımında soldan sağa doğru ilerlenir. İlk satırın sonunda, piksel (639,0) konumunda satır sayacı arttırılır ve sütun sayacı da ilk sütun için

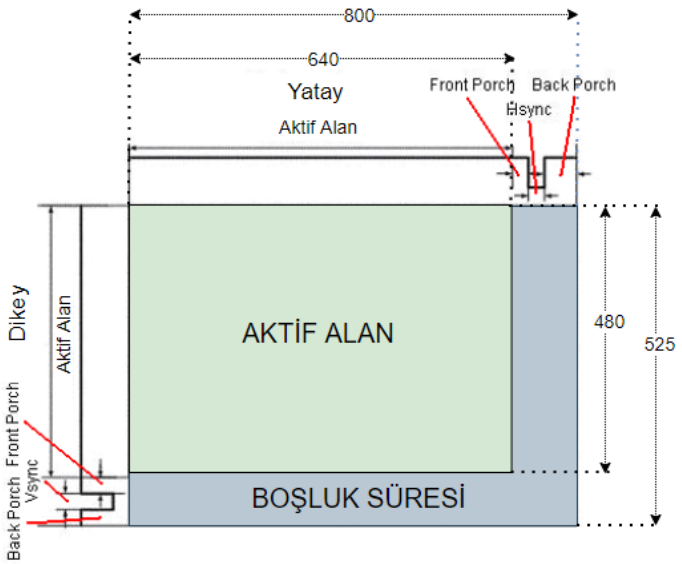
sıfırlanır. Tüm ekrana piksel geldiğinde bir görüntü çerçevesi bitmiş olur. Ardından bir sonraki görüntü çerçevesi için bu yenileme işlemi tekrarlanır.

Birden fazla görüntülerden oluşan gerçek zamanlı görüntü akışında hareketi sağlama için her bir çerçeve yeterli bir hızda gösterilmelidir. Bu yenilemenin yapıldığı süre yenileme hızı yani fps olarak geçer. Saniyede 60 kez yapılan bu işlem insan gözünün gerçek zamanlı görüntü akışını görmesi için yeterli gelmektedir. 640x480 piksellik VGA ekranı için 60 Hz yenileme hızı piksel saat sinyalinin 25.175 MHz olarak ifade edilmesini sağlar. Denklem 2.8'de gösterilen piksel başına 39.72 ns süreye karşılık gelir.

$$\frac{1}{25.175\ MHz} = 0.00003972194\ \text{s} = 39.72\ \text{ns} \tag{2.8}$$

2.6.2. VGA Bölgeleri

Şekil 2.13'te gösterildiği gibi her video satırı, satırdaki her piksel için RGB değerlerinin çıktığı aktif bir video bölgesi ile başlar. Aktif bölgeyi, siyah piksellerin iletildiği bir boşluk bölgesi takip eder. Tam ekran için tarama çizgisinin her birinin zamanlaması gösterilmiştir.



Şekil 2.13: VGA sinyali 640 x 480, 60 Hz endüstri standardı zamanlama.

Tablo 2.2'de gösterildiği gibi monitörün tam ekranı her biri 800 piksellik 525 yatay tarama satırı olarak görüntülenmektedir, ancak tarama satırlarının yalnızca 480'i ve tarama satırı başına 640 piksel gerçekte görüntüyü aktif alan olarak iletirken geri kalan boşluk süresi kısmı siyahtır.

Bir tarama satırı, ekranın sol kenarındaki boş bölüm olan 48 pikselden oluşan back porch (arka sundurma) ile başlar. Ardından aktif alanda 640 piksel değerini sağladıktan sonra devamında

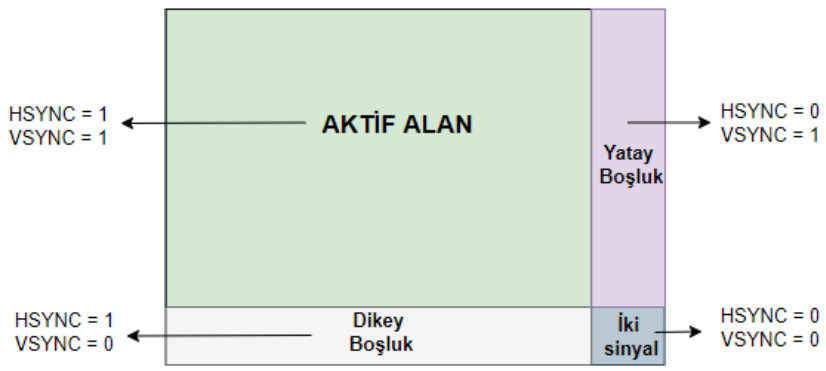
ekranın sağ kenarında boş bölüm olan 16 pikselden oluşan front porch (ön sundurma) ve taramayı hızla bir alt satıra geçirerek devam eden piksel verisini sol kenardan başlatmak için 96 pikselden oluşan HSYNC sinyali içerir [28].

HSYNC sinyali, monitöre 640 piksellik başka bir satır başlatmasını bildirir. Yatayda piksel alınan ana işaret etmektedir. VSYNC sinyali ise, monitöre yeni bir çerçeveyi görüntülemeye başlamasını bildirir.

Şekil 2.14'te HSYNC ve VSYNC sinyal değerlerinin bölgedeki değerleri gösterilmiştir. 640x480 VGA için 480 satır; 480 HSYNC sinyali ile yenilendikten sonra, VSYNC sinyali monitörü sol üst köşeye sıfırlar ve işlem yeniden başlatılır. HSYNC ve VSYNC sinyal değerleri bulunduğu bölgeye göre değer alarak yeni satır ve yeni çerçeveye geçişi başlatmaktadır. Aktif alanda bu iki sinyal değeri '1' değerini alırken, boşluk bölgelerinde yatay ve dikeyde olmasına göre '1' veya '0' değerlerini almaktadır.

Tablo 2.2: Tarama piksel sayıları [29].

Tarama Çizgisi Alanı	**Yatay Piksel Sayısı**	**Dikey Satır Sayısı**
Aktif Alan	640	480
Boşluk Zamanı: Front Porch	16	10
Boşluk Zamanı: Sync	96	2
Boşluk Zamanı: Back Porch	48	33
Tüm Ekran	800	525

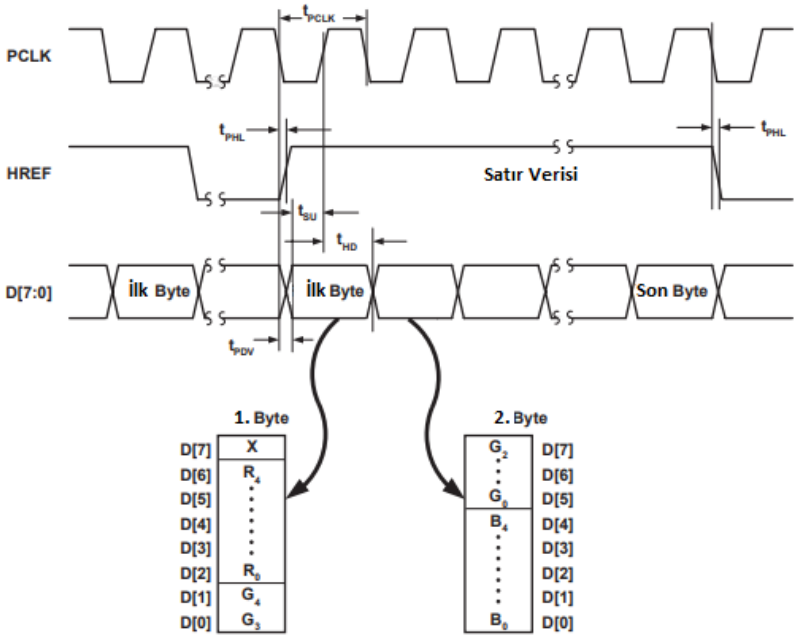


Şekil 2.14: HSYNC ve VSYNC sinyal değerleri [30].

2.6.3. VGA Çerçeve Zamanlaması

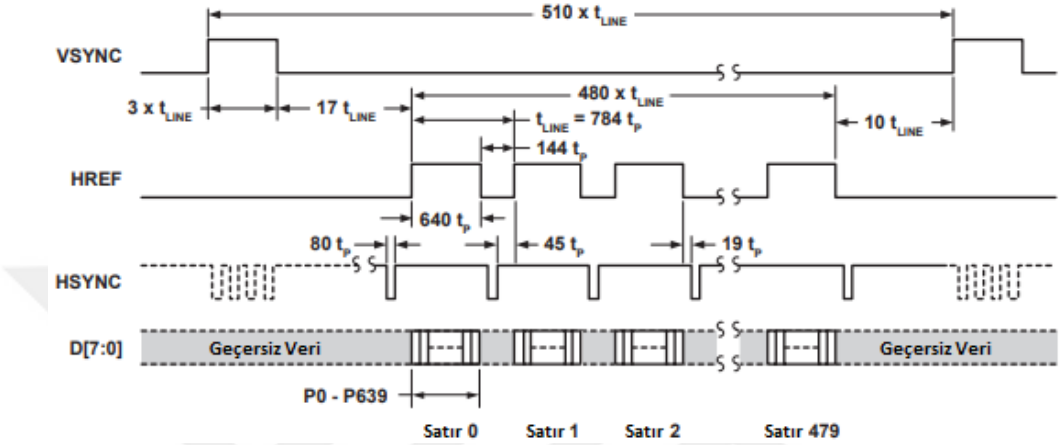
Görüntü sensörü, piksel verilerini paralel senkronize bir biçimde göndermektedir. Böylece maksimum 8 bitten oluşan piksel verisi verebilmektedir. Bu sebeple RGB555 çıktısı şeklinde alınacaksa çerçevedeki her piksel için kameranın 2 kez 8 bit veri vermesi gerekmektedir. Bu da 2 baytlık veriye karşılık gelmektedir. Piksel verisini alabilmek için görüntü sensörünün XCLK pinine bir saat sinyali gönderilir. Bu saat sinyali frekans değeri 10-48 MHz arasında olmalıdır. Genelde görüntü sensörünün 30 fps çıktı verebilmesi için bu frekans değeri 24 MHz olarak alınır [31]. PCLK ile aynı frekansa sahip XCLK pinine saat sinyali uygulandıktan sonra OV7670 görüntü sensörü; VSYNC, HREF ve D0-D7 pinlerini aktif duruma getirir. Görüntü formatından bağımsız olarak 24 MHz'lik bir PCLK 30 fps, 12 MHz'lik bir PCLK 15 fps vb. üretebilir [30].

Kamera, 8 bitlik bir piksel verisi olan D[7:0] sinyalini PCLK sinyalinin yükselen kenarında örneklenmektedir. D[7:0] verileri sadece HREF yüksek değerdeyken alınmaktadır. Ayrıca, HREF'in yükselen kenarı bir satırın başlangıcını ve HREF'in düşen kenarı bir satırın sonunu işaret eder. HREF yüksek olduğunda örneklenen tüm bu baytlar, bir satırdaki piksellere karşılık gelir. Şekil 2.15'te gösterilen RGB555 formatında 1. bayt; "X" ile temsil edilen boş veri, 5 bitlik "R" ile temsil edilen kırmızı renk, 2 bitlik "G" ile temsil edilen yeşil renkten oluşmaktadır. 2. bayt ise 3 bit yeşil renk, 5 bit "B" ile temsil edilen mavi renkten oluşmaktadır.



Şekil 2.15: RGB 555 çıkış zamanlama şeması.

Şekil 2.16'da VSYNC, HREF, HSYNC ve kameranın 8 bitlik piksel verisi olan D[7:0] sinyallerinin zamanlama şeması gösterilmiştir. HSYNC yüksek durumu sırasında, bir satır için 640 piksel yakalanmaktadır. VSYNC düşük durumunda ise bir çerçeveye eşdeğer olan 480 satır yakalanır. Bu durum, VSYNC'in düşen kenarının bir çerçevenin başlangıcını ve yükselen kenarının bir çerçevenin sonuna işaret eder.



Şekil 2.16: VGA çerçeve zamanlama şeması.

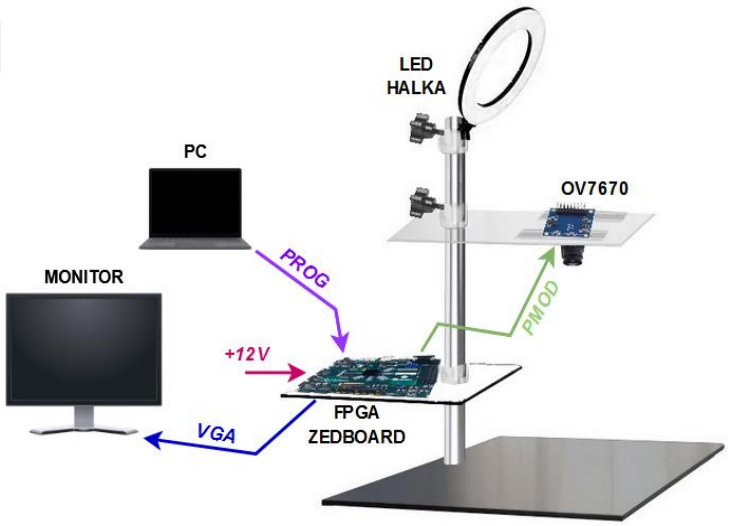
Kamera register ayarının düzgün bir şekilde yapılabilmesi için VSYNC sinyali ilk başta yeni bir çerçeve almadan '0' olarak başlatılır. 3 satır süresi boyunca VSYNC sinyali '1' değerine çekilerek yeni bir görüntü çerçevesi alınır. 17 satır sonra HREF sinyali yatayda piksel verilen anı işaret eder ve piksel alımı için '1' değerini alır. HREF '1' değerini aldığı süre boyunca kameradan 1 satırlık 640 piksellik veri çekilmektedir. 480 satır kadar veri alındığında yeni bir çerçeveye geçmek için 10 satır bekledikten sonra VSYNC sinyali '1' değerine çekilir.

3. YÖNTEM

3.1. DONANIMSAL SİSTEM TASARIMI

Gerçek zamanlı gömülü sistemler, bulunduğu sisteme özel olarak tasarlanırlar. Gömülü sistemde, uygulamaya özel olan yazılım ve donanım kısımları mevcuttur. FPGA, genelde gömülü sistemlerde bilgisayar donanımı ile kullanılır.

Görüntünün alınması, gürültülerin giderilmesi ve eksik komponent hatalarının tespiti amacıyla oluşturulan gerçek zamanlı sistem; OV7670 görüntü sensörü, FPGA, VGA monitör, PC, yatay ve dikey yönü ayarlanabilir LED halka, iki adet yatay ve dikey yönü ayarlanabilir plaka ve en alt kısımda sabit plakadan olmak üzere Şekil 3.1'de gösterildiği gibi donanımsal parçalardan oluşmaktadır.



Şekil 3.1: Donanımsal sistem bağlantı şeması.

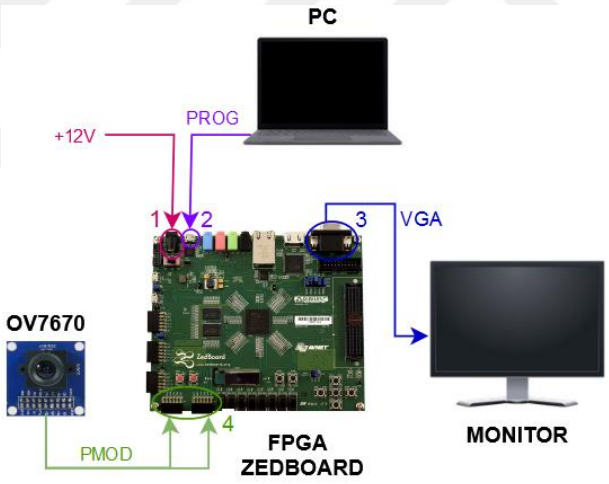
OV7670 görüntü sensörü kısmı, gerçek zamanlı görüntüyü alma amacıyla kullanılmaktadır. En üstteki plakaya OV7670 görüntü sensörü monte edilmiştir. Görüntü sensörünün görüş açısı en alttaki sabit plakadır.

En alttaki plakaya FPGA monte edilmiştir. FPGA olarak Xilinx firmasına ait Zynq-7000 XC7Z020 SoC mimarisi içeren AVNET firması üretimli Zedboard Geliştirme Kartı kullanılmıştır. 12V adaptörle güç girişi almaktadır. Görüntünün alınması, gürültülerin

giderilmesi amacıyla görüntünün filtrelenmesi, eksik komponent hatalarının tespiti ve VGA monitöre görüntü aktarımının yapılması için PC üzerinden Vivado programında VHDL programlama yapılmaktadır. FPGA içerisine PC üzerinden programlanarak oluşturulan VHDL kodları gömülür. Görüntüdeki gürültü kaldırılmak ve eksik komponentler monitörde gösterilmek istendiğinde FPGA üzerinde yer alan anahtarlar anlık olarak açılarak filtreleme işlemi yapılabilir.

Görüntüde oluşan gauss ve medyan (tuz-biber) gürültüsü dışındaki gürültü sebeplerinden biri, zayıf aydınlatma problemidir [32]. Bundan dolayı gürültüyü azaltma amacıyla led halka sisteme dahil edilmiştir.

Sonuç olarak FPGA geliştirme kartına bağlı OV7670 kamera sensörü ile I2C haberleşme sağlanarak görüntü çerçeveleri alınır. Her bir görüntü çerçevesinin verileri FPGA içerisinde geçici olarak depolanır ve bu veriler VGA monitörünü sürmek için kullanılan bir donanım tasarımını oluşturur. Şekil 3.2'de donanımsal sistem bağlantı noktaları gösterilmiştir.



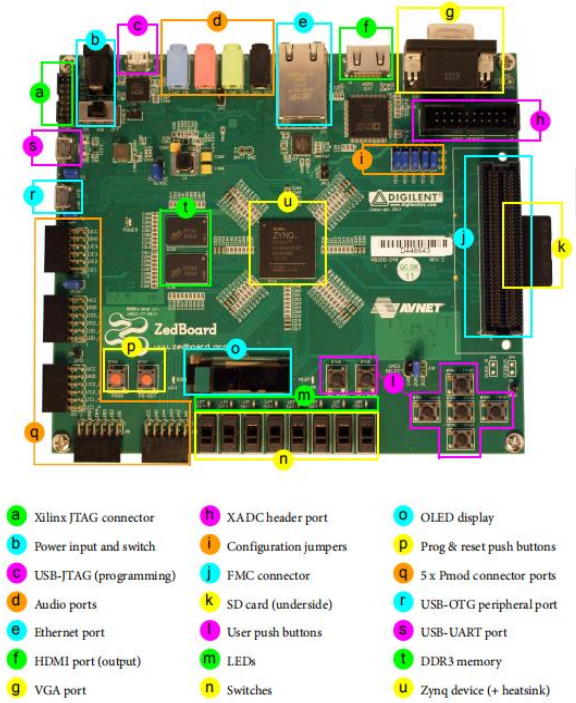
Şekil 3.2: Donanımsal sistem bağlantı noktaları.

- 1 numaralı kısımda, FPGA'in çalışması için gereken +12V güç girişi bağlantısı sağlanır.
- 2 numaralı kısımda, FPGA içerisine PC'de VHDL ile yazılmış olan kodlar; PC'nin USB girişinden FPGA'in USB-JTAG (PROG) girişine olan bağlantı ile aktarılmaktadır.
- 3 numaralı kısımda, FPGA'in VGA çıkışı, monitörün VGA girişine bağlanmaktadır. VGA pinleri için yazılan VHDL kodu sayesinde görüntü aktarımı sağlanır.
- 4 numaralı kısımda, OV7670 görüntü sensörü pinleri, FPGA'de bulunan PMOD (çevresel modül) pinleri ile bağlanmaktadır. PMOD pinleri ile görüntü sensörü pinlerine giriş/çıkış atamaları yapılır.

3.2. ZEDBOARD FPGA GELİŞTİRME KARTI

3.2.1. ZedBoard Özellikleri

ZedBoard, donanım ve yazılım geliştiricilerin Xilinx Zynq-7000 tamamen programlanabilir SoC tabanlı tasarımlar oluşturmasını ve test etmesini sağlayan FPGA kartıdır [33]. Üzerinde bulunan Zynq-7000 çipi; çift çekirdekli Cortex-A9 işlemci sistemi (PS) ve 85.000 Seri-7 programlanabilir mantık (PL) hücresiyle birleştirmektedir. Zedboard geliştirme kartı; Zynq-7000 çipi sayesinde tasarımcıların çeşitli uygulamaları tasarlanabileceği ideal bir platform haline gelmektedir. Zynq, her ikisi de ZedBoard üzerinde yer alan 512 MB DDR3 bellek ve 256 Mbit flash belleğe arayüz sağlar. 2 adet osilatör saat kaynağı mevcuttur. Bu osilatörler; PL için 100 MHz ve PS için 33.3333 MHz saat kaynağı sağlamaktadır. Şekil 3.3'te ZedBoard geliştirme kartı gösterilmiştir [34].



Şekil 3.3: ZedBoard geliştirme kartı [34].

Gösterilen birimler;

- **FPGA mikroişlemcisi**

  u: Zynq-7000 SoC XC7Z020-CLG484-1 mikroişlemcisini içermektedir.

- **Genel Amaçlı Giriş/Çıkış Arabirimleri (GPIO - General Purpose I/O)**

  m: 8 adet kullanıcı LED'i, **n:** 8 adet kullanıcı DIP anahtarı, **ı:** 7 adet basmalı düğme elemanlarını temsil etmektedir.

- **Güç Girişi ve Anahtarı**

  b: 12VDC ile beslenmektedir. Switch ile açılıp kapatılır.

- **USB-JTAG Programlama**

  c: JTAG (Joint Test Action Group), FPGA'i programlama ve donanımda bulunan hataların ayıklanması için kullanılan arabirimdir.

- **PMOD Konnektörleri:**

  q: Digilent PMOD, çeşitli harici cihazlar ile mikrodenetleyici ve FPGA geliştirme kartları arasında iletişimi sağlayan konnektörlerdir.

- **Ses Bağlantı Portları:**

  d: Analog Cihazların bağlantısı (ADAU1761), mikrofon girişi ve kulaklık çıkışına imkan veren porta sahiptir.

- **HDMI, VGA Portları ve OLED Ekranı:**

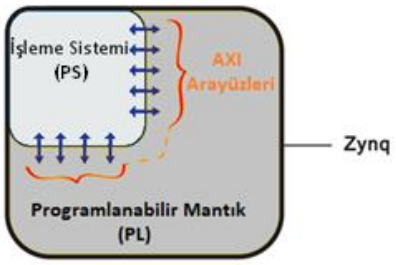
  Video görüntülemesi için 1080p HDMI, 12 bit çözünürlüklü renk ile VGA desteği ve OLED (Organik Işık Yayan Diyot) ekranı mevcuttur.

  f: HDMI portu, **g:** VGA portu, **o:** OLED ekranı

Ayrıca ethernet arayüzü, kartın alt kısmında bulunan SD kart yuvası, ADC başlığı, Xilinx JTAG, FMC (FPGA Mezzanine Card) arayüzü özellikleri mevcuttur.

3.2.2. ZYNQ-7000 SoC Mimarisi

Zynq; programlanabilir, yeniden yapılandırılabilir bir cihaz için uygulanan özel bir SoC çeşididir. İki ana bölümden oluşmaktadır. Tek bir cihazda çift çekirdekli bir ARM Cortex-A9 işlemci etrafında oluşturulmuş bir işlemci sistemi (PS) ve bir FPGA'ya eşdeğer olan Programlanabilir Mantığı (PL) birleştirmektedir [35]. Zynq-7000 SoC mimarisi içerisinde tek veya çift çekirdekli ARM işlemci ve 28 nm FPGA bulundurmaktadır. Xilinx firması, Zynq-7000 serisi ile FPGA üzerinde ARM Cortex-A9 işlemci tabanlı sistemi entegre etmiştir [36]. Ayrıca entegre bellek, yüksek hızda iletişim arayüzleri ve çeşitli çevre birimlerini içermektedir. PS kısmıyla; yazılım bölümü ve işletim sistemlerini desteklemektedir. Böylece tasarlanan sistemde genel işlevi, donanımı ve yazılımı bölümlere ayrılabilir. PL kısmıyla; aritmetik, yüksek hızda mantık ve veri akışı sistemlerini uygulamaktadır. PL ve PS arasındaki bağlantılar, Gelişmiş Genişletilebilir Arayüz (AXI) bağlantıları kullanılarak yapılır. Şekil 3.4'te Zynq mimarisinde yer alan PL ve PS kısımları gösterilmiştir.



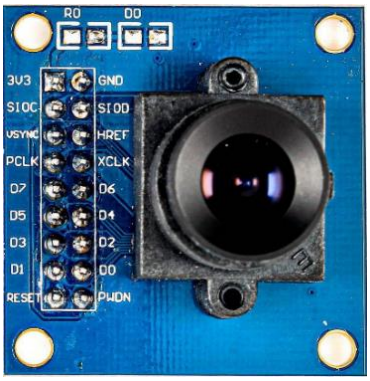
Şekil 3.4: Zynq mimarisi.

PS, işlemci ve sistem belleğini barındırmaktadır. Sabit bir mimariye sahiptir. PL, esnek olan bir mimaridedir. Tasarımcıya özel çevre birimlerini oluşturur veya standart olan birimlerin kullanımı için boş alan tanımlar.

3.3. OV7670 GÖRÜNTÜ SENSÖRÜ

3.3.1. Görüntü Sensörü Özellikleri ve Pinleri

OV7670 görüntü sensörü, düşük voltajlı CMOS cihazıdır. I2C seri haberleşme protokolü ile uyumlu Seri Kamera Kontrol Veri Yolu (SCCB) arabirimi aracılığıyla kontrol edilen RGB, YUV, YCbCr gibi formatlarda 8 bitlik piksellerden oluşan görüntü sağlar. Görüntü işleme işlevleri olan pozlama kontrolü, gama, beyaz dengesi, renk doygunluğu, ton kontrolü gibi ayarlamaların SCCB arayüzü aracılığı ile değiştirilmesini sağlar. VGA'da 30 fps hızına kadar çalışabilen bir görüntü çerçevesine sahiptir. Yatay çözünürlük, bir ekranın genişlik yönündeki piksel sayısını; dikey çözünürlük, bir ekranın yükseklik yönündeki piksel sayısını ifade eder. Görüntü sensörünün aktif bölgede 640x480 çözünürlüğü vardır ve 307.200 piksele karşılık gelmektedir. Bu da kameranın 0.3 MP çözünürlükte olduğunu göstermektedir. OV7670 tarafından RGB formatı olarak RGB565, RGB555 ve RGB444 kullanılmaktadır [31]. Şekil 3.5'te çalışmada kullanılan OV7670 görüntü sensörü ve üzerinde yer alan pinler mevcuttur.



Şekil 3.5: OV7670 görüntü sensörü.

Tablo 3.1'de OV7670 görüntü sensörü üzerinde bulunan pinlerin adına, tipine ve kullanımlarına yer verilmiştir. Bu pinlere göre görüntü sensörü programlanmaktadır.

Tablo 3.1: OV7670 Görüntü Sensörü Pinleri [31].

Pin Adı	Pin Tipi	Kullanımı
3V3	Güç	3.3V Analog Güç Kaynağı
GND	Güç	Analog Toprak Hattı
SIOC	Giriş	SCCB Seri Arayüz Clock Girişi
SIOD	Giriş / Çıkış	SCCB Seri Arayüz Verisi
VSYNC	Çıkış	Vertical sync – Dikey senkronizasyon çıkışı
HREF	Çıkış	Horizontal referance çıkışı – Yatay senkronizasyon çıkışı
PCLK	Çıkış	Pixel clock çıkışı (24 MHz)
XCLK	Giriş	System clock girişi (24 MHz)
D[7,6,…,2,1,0]	Çıkış	YUV/RGB çıkış bitleri (8 bit)
PWDN	Giriş	Power Down Mod Seçimi (Aktif Yüksek) 0: Normal mod, 1: Power down modu
RESET	Giriş	Tüm registerları siler, varsayılan değerlerine çevirir. (Aktif Düşük) 0: Reset modu, 1: Normal mod

3.3.2. Görüntü Sensörü için SSCB (Seri Kamera Kontrol Veri Yolu)

OV7670 görüntü sensöründe bulunan SSCB, görüntü gönderilmeden önce görüntüyü işleyebilen dahili DSP olarak kullanılır. DSP'ye bir SCCB arayüzü üzerinden erişilebilmektedir. FPGA'in I2C modülünün OV7670'in SCCB arayüzü ile çalışması sağlanmıştır. Görüntü sensöründe FPS değiştirmek için kameranın SSCB arayüzünün sunmuş olduğu kamera veri sayfasına (datasheet) göre PCLK frekansını değiştirmek gerekmektedir.

Çalışmada, Tablo 3.2'de SSCB'de yer alan 11. register adresinin değeri olan on altılık (hexadecimal) tabanda gösterilen '80' değeri kullanılmıştır. '80' değeri ikili tabanda '1000 0000' değerine karşılık gelmektedir. Bu da Bit [6]'nın '0' değerine eşit olduğunu gösterir. Açıklama kısmında yer alan Bit[6]'nın '0' olması durumuna göre giriş saatine ön ölçekleyici uygulanarak 24 MHz değerinde PCLK değeri alınmaktadır. 24 MHz değeri, 30 fps'ye karşılık gelmektedir. FPS değeri değiştirilmek istendiğinde Bit[6]'ya '0' değeri verilerek ön ölçekleme kullanılarak Bit[5-0] değeri değiştirilir.

Tablo 3.2: FPS değiştirme register ayarı [31].

Adres	Register Adı	Varsayılan Register Değeri	Açıklama
11 (HEX)	DAHİLİ SAAT (CLKRC)	80 (HEX)	Bit[6]: 0: Giriş saatine ön ölçekleyici uygula. 1: Doğrudan harici saati kulan. Bit[0-5]: Saat ön ölçekleyici F(dahili saat) = F(giriş saati) / (Bit[0-5] + 1) Aralık [0 0000] - [1 1111]

Görüntü sensörü register ayarlarından çerçeve formatı değiştirmek için kameranın SSCB arayüzünün sunmuş olduğu datasheet verisine göre Tablo 3.3'te gösterilen 12. register adresinin değerinin değiştirilmesi gerekmektedir. 12. register adres değeri olarak onaltılık (hexadecimal) tabanda gösterilen '00' değeri kullanılmıştır. RGB olarak değiştirmek için '04' olarak ayarlanmıştır. Bu değer ikili tabanda '0000 0100' değerine karşılık gelmektedir. Bit [2]'nin '1' değerine ve aynı zamanda Bit[0]'ın '0' değerine eşit olması RGB formatının seçildiğini gösterir.

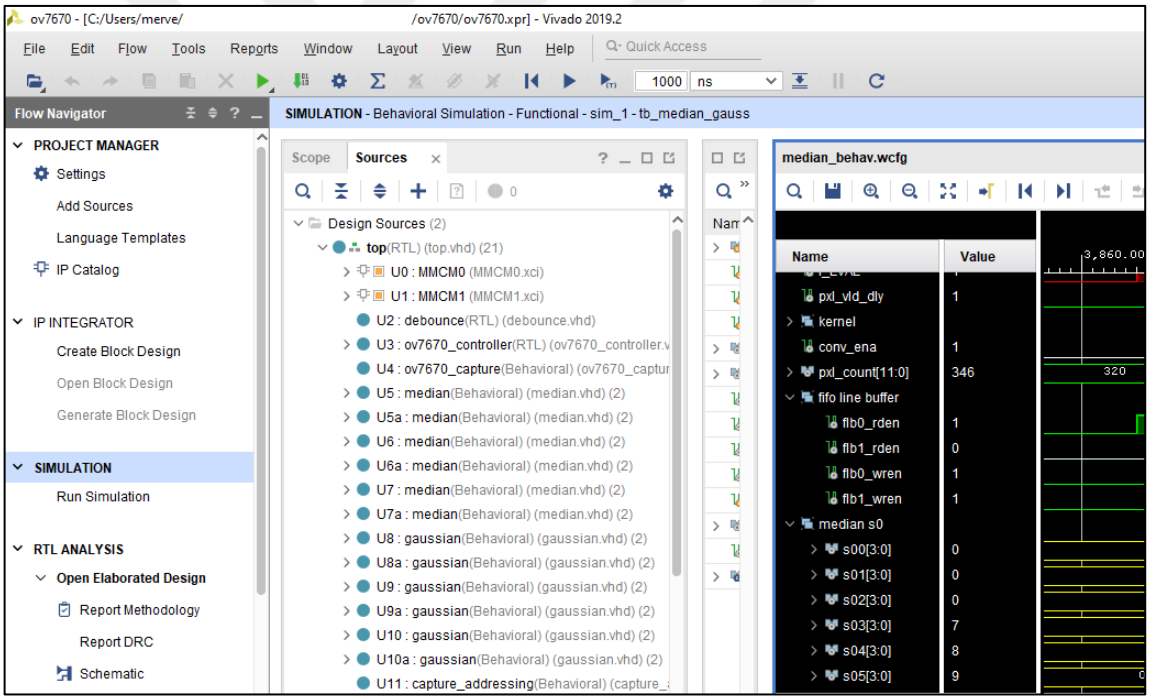
Tablo 3.3: Çerçeve değiştirme register ayarı [31].

Adres	Register Adı	Varsayılan Register Değeri	Açıklama
12 (HEX)	COM7 (Common Control 7)	00 (HEX)	Bit[7]: SSCB Register Reseti 0: Değişim yok. 1: Tüm registerları varsayılan ayarına resetle. Bit[6]: Rezerve Bit[5]: Çıkış formatı – CIF seçimi Bit[4]: Çıkış formatı – QVA seçimi Bit[3]: Çıkış formatı – QCIF seçimi Bit[2]: Çıkış formatı – RGB seçimi Bit[1]: Renk skalası 0: Reddetme 1: Seçme Bit[0]: Çıkış formatı – Raw RGB seçimi COM7[2] COM7[0] YUV formatı 0 0 RGB formatı 1 0 Bayer RAW formatı 0 1 İşlenmiş Bayer RAW 1 1

3.4. VIVADO PROGRAMI

FPGA'in programlanması, FPGA üreticilerine göre değişiklik gösteren yazılım programları kullanılarak yapılmaktadır. FPGA'i ilk üreten Xilinx firmasının sunduğu Vivado Design Suite programı, tasarımcıların donanım tanımlama dilleri ile tasarımlarını sentezlemesine ve analiz etmesine olanak sağlayan bir yazılım paketidir [37]. Donanımsal bir kodun sentezlenebilmesi, bu kodun FPGA içerisine atılması ile sağlanır. Sentezlenemeyen bir kod, FPGA içerisine yüklenemez. Yalnızca simülasyon ile sonuçları gösterilebilir.

Vivado programı, tasarımcıların soft işlemci sistemi oluşturarak projeye eklemelerine de yardımcı olmaktadır. Ayrıca C kodunu baz alarak programlanabilir mantık seviyesine dönüştürerek yüksek seviyeli sentez olanağı sunmaktadır. Şekil 3.6'da Vivado programının arayüz görünümü gösterilmiştir.



Şekil 3.6: Vivado programı arayüz görünümü.

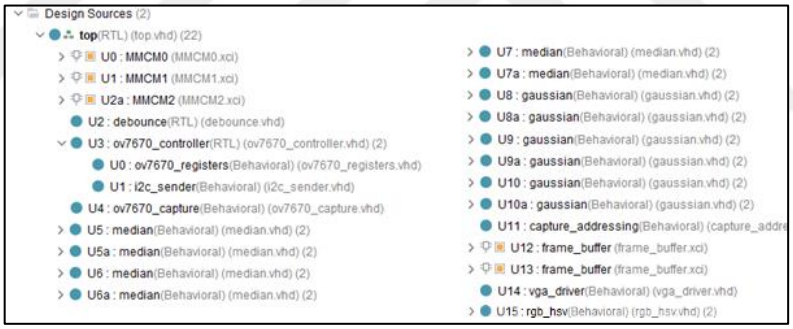
3.4.1. Vivado IP Blokları

FPGA üreticileri, Vivado içerisinde hazır olarak bulunan fikri mülkiyet (IP) bloklarını sunarak tasarımcıların gelişmiş tasarımları, modellemeyi ve simülasyonu zamandan tasarruf ederek yapmasına olanak sağlar. IP Kataloğu, içerisinde hazır Xilinx IP kütüphanesini bulundurmaktadır. Kullanıcı yapacağı tasarıma göre hazır olan bu IP bloklarını kullanabilir veya kendisi oluşturabilir. Bu kütüphaneden alınan IP'ler tasarıma eklenerek entegre edilir.

Böylece kütüphanede mevcut olan bu IP'ler, tasarımda tekrar donanım kodunun yazılmasına gerek kalmadan kullanılır. Bu sayede FPGA, güçlü yazılım desteğine sahiptir. IP çekirdeği, karmaşık sistem seviyesinde olan öncesinde test edilen mantık tasarımı işlevleridir. IP çekirdeği; geliştirme sürecinde şematik olarak sadeleştirilmiş bir tasarım, kod yazımında kısalma ve hız kazandırmayı sağlamaktadır. Böylece yazılımın derleme işlemleri azalmış olur.

3.4.2. Vivado Modül Sınıflandırılması

Büyük fonksiyonel bloklar küçük parçalara ayrılarak bu programda modellenebilir. Böylece tasarımın karmaşıklığı azaltılmış olur ve bir tasarım için birden fazla tasarımcının çalışması sağlanabilir. Hiyerarşik tasarıma göre bir üst modülün altında birden fazla alt modül mevcuttur. Böylece tasarımın daha kolay yönetilmesini sağlamaktadır. "Top" ismi verilen üst modül; FPGA'in içerisine gömülecek tüm giriş, çıkış pinleri ve port olarak clock, reset, reconfig, filtreleme ve eksik komponent anahtarları, OV7670 arayüz pinleri, VGA pinlerini içermektedir. Ayrıca çalışmanın içerdiği tüm komponentleri alt modülünde bulundurur. Şekil 3.7'de VHDL ile kodlanmış tüm modüller gösterilmektedir.



Şekil 3.7: Vivado modül sınıflandırılması.

Bu modüller;

- **U0, U1, U2a:** Karışık Modlu Saat Yöneticisi (Mixed-Mode Clock Manager - MMCM) çalışmada bulunan modüllere saat üretimini sağlayan IP bloklarıdır.
- **U2:** Debounce modülü, ZedBoard üzerinde herhangi bir butona basıldığında tuş sekmesinin oluşmasını önleyen modüldür.
- **U3:** OV7670 görüntü sensörünün i2c haberleşmesi ve register ayarlarının yapılması için "ov7670_controller" isimli 2 alt modülden oluşan kontrolör modülü kullanılmıştır.
- **U4:** Görüntü sensöründen gelen verilerin yakalanması için "ov7670_capture" modülü mevcuttur.

Çalışmada görüntü sensöründen yakalanan verilere medyan ve gauss filtreleme işlemleri 2 kez yapılabilmektedir. “a” harfi eklenmiş modüller bu işlemin 2. defa yapılmasını sağlamaktadır.

- Medyan filtreleme işlemini; kırmızı kanal için **U5, U5a;** yeşil kanal için **U6, U6a;** mavi kanal için **U7, U7a** modülleri yapmaktadır.
- Gauss filtreleme işlemini; kırmızı kanal için **U8, U8a;** yeşil kanal için **U9, U9a;** mavi kanal için **U10, U10a** modülleri yapmaktadır.
- **U11:** Görüntü sensöründen yakalanarak filtrelenen piksel verilerinin adreslenerek RAM’e yazılmasını sağlayan “capture_adressing” modülüdür.
- **U12, U13:** Ekrana basılacak olan görüntü çerçevesini kaydeden çerçeve arabelleği olarak geçen 2 adet “frame_buffer” IP bloklarıdır.
- **U14:** Frame_buffer modülüne adres göndererek ekranda piksel basılmasını sağlayan “vga_driver” modülüdür.
- **U15:** Eksik komponent hata tespiti için, RGB değerlerini alarak HSV değerlerini hesaplayan “rgb_hsv” modülüdür. Lehimli alanların HSV değerleri aralığı için ekranda kırmızı renk olarak gösterilmesini, diğer aralıklarda RGB değerlerinin ekrana basılması VGA modülü ile sağlanır.

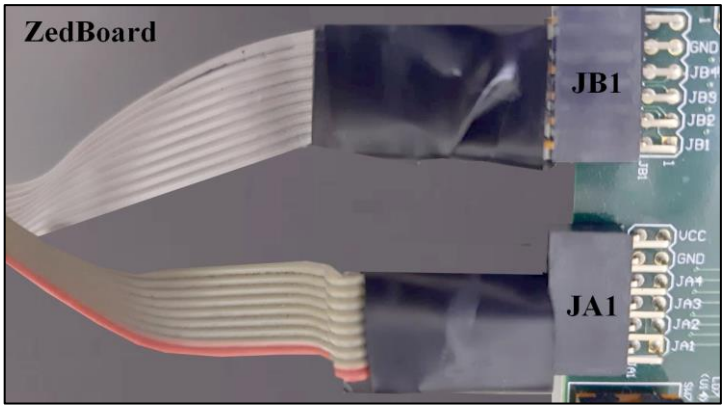
3.5. FPGA BAĞLANTILARI

3.5.1. OV7670 Görüntü Sensörü Bağlantısı

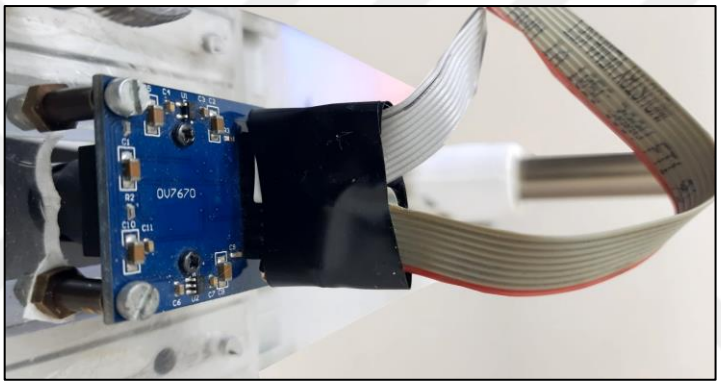
FPGA ile kameradan görüntü piksel verisinin alınabilmesi için Şekil 3.8’de gösterilen ZedBoard üzerinde bulunan 2 adet PMOD başlığı (JA1 ve JB1) kullanılmaktadır. ZedBoard PMOD başlığı ile OV7670 görüntü sensörü pinleri bağlanarak veri alımı sağlanır. Aradaki bu bağlantı flex kablo (LL83498) ile yapılmaktadır.

Şekil 3.9’da gösterilen OV7670 görüntü sensörünün flex kablo aracılığıyla ZedBoard üzerindeki PMOD’lar ile bağlantısı yapılmıştır.

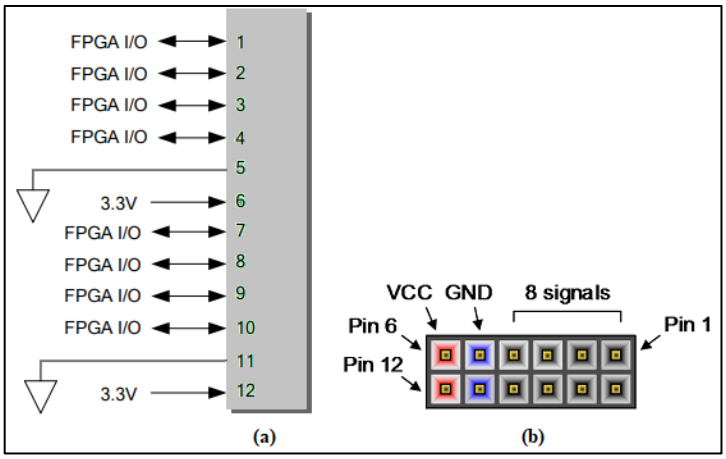
Şekil 3.10’da ZedBoard PMOD başlığı pinleri gösterilmiştir. Şekil 3.10 (a)’da bu pinlerin bağlantı şemasına, Şekil 3.10 (b)’de ise PMOD başlığının şematik olarak gösterimine yer verilmiştir.



Şekil 3.8: ZedBoard PMOD bağlantısı.



Şekil 3.9: OV7670 bağlantısı.



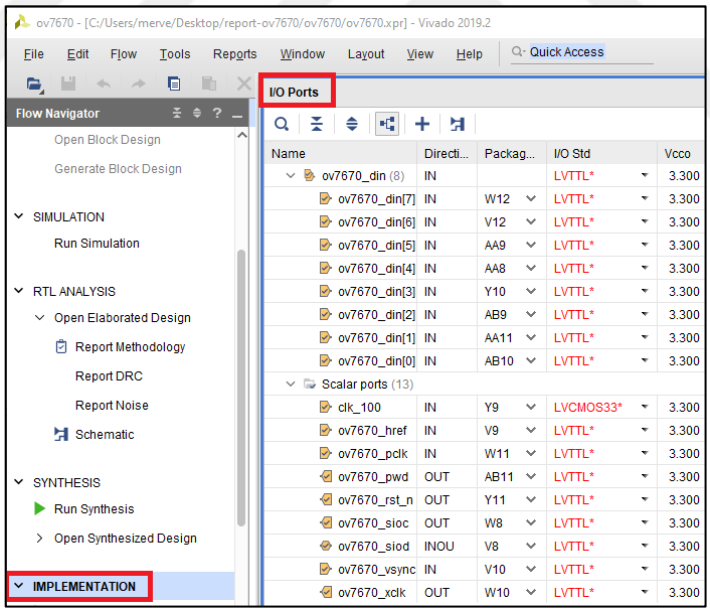
Şekil 3.10: ZedBoard PMOD başlığı pinleri.(a) Bağlantı şeması.(b) PMOD başlığı [33].

Tablo 3.4'te PMOD başlıklarının (JA1 ve JB1) fiziksel olarak bağlantısının yapılabilmesi için FPGA üzerindeki yerleşimini belirleme işlemi olan implementation (uygulama) işleminde Zynq pini karşılıkları ve OV7670 pini ile bağlantısı gösterilmiştir [33].

Tablo 3.4: PMOD-Zynq pini ile OV7670 pinleri bağlantısı gösterimi.

JA1 PMOD İsmi	Zynq Pini	OV7670 Pini	JB1 PMOD İsmi	Zynq Pini	OV7670 Pini
JA1	Y11	RESET	JB1	W12	D7
JA2	AA11	D1	JB2	W11	PCLK
JA3	Y10	D3	JB3	V10	VSYNC
JA4	AA9	D5	JB4	W8	SIOC
JA5	GND	GND	JB5	GND	GND
JA6	3V3	3V3	JB6	3V3	3V3
JA7	AB11	PWDN	JB7	V12	D6
JA8	AB10	D0	JB8	W10	XCLK
JA9	AB9	D2	JB9	V9	HREF
JA10	AA8	D4	JB10	V8	SIOD
JA11	GND	GND	JB11	GND	GND
JA12	3V3	3V3	JB12	3V3	3V3

Şekil 3.11'de Vivado programında implementation kısmında "I/O ports" (giriş/çıkış portları) sekmesinde kısmında kameraya ait 8 bitlik veri giriş pinlerinin Tablo 3.4'e uygun olarak ZedBoard PMOD pin atamaları yapılmıştır.

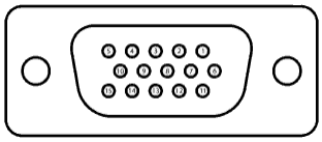


Şekil 3.11: Vivado programında görüntü sensörü pinlerinin ataması.

3.5.2. VGA Bağlantısı

Farklı FPGA geliştirme kartları, üzerinde bulunan VGA konnektörüne göre renk derinliği bit değerini almaktadır [36]. ZedBoard kartı ise monitörde gösterimi sağlamak için Şekil 3.12'de gösterilen 12 bit RGB renk derinliği video çıkışına izin veren VGA konnektörüne sahiptir.

Monitör üzerindeki her pikselin kırmızı, yeşil ve mavi renk bileşeni mevcuttur. VGA bağlantısından monitöre giden kırmızı, yeşil ve mavi renkli sinyaller analog tipte sinyallerdir. Her renk, dört PL pini direnç merdiveninden oluşturulur [33].



Şekil 3.12: DB15 tipi, ZedBoard 12-bit VGA konnektörü [33].

ZedBoard üzerinde bulunan VGA konnektörü, 12 bit renkli RGB video çıkışına izin verdiğinden, bu çalışmada RGB555 formatında kameradan alınan görüntüde her renk verisi için 4 MSB alınarak RGB444 formatında VGA monitörüne sürülmüştür. VGA monitörünün sürülebilmesi için bir saat sinyaline ihtiyaç vardır. FPGA içerisinde yer alan 100 MHz osilatör kullanılarak 25.175 MHz'lik saat sinyali ile VGA monitörü sürülmektedir. Tablo 3.5'te ZedBoard veri sayfasından alınan VGA pin karşılıkları gösterilmiştir. VGA pin karşılıkları kullanılarak Şekil 3.13'teki gibi Vivado programında pin atamaları yapılmıştır.

Tablo 3.5: Zynq pini ile VGA pinleri bağlantısı gösterimi [33].

VGA Pini	Sinyal İsmi	Zynq Pini
1	Kırmızı	V20, U20, V19, V18
2	Yeşil	AB22, AA22, AB21, AA21
3	Mavi	Y21, Y20, AB20, AB19
4	HSYNC	AA19
5	VSYNC	Y19

Name	Directi...	Packag...	I/O Std	Vcco
vga_blue (4)	OUT		LVCMOS33*	3.300
vga_blue[3]	OUT	AB19	LVCMOS33*	3.300
vga_blue[2]	OUT	AB20	LVCMOS33*	3.300
vga_blue[1]	OUT	Y20	LVCMOS33*	3.300
vga_blue[0]	OUT	Y21	LVCMOS33*	3.300
vga_green (4)	OUT		LVCMOS33*	3.300
vga_green[3]	OUT	AA21	LVCMOS33*	3.300
vga_green[2]	OUT	AB21	LVCMOS33*	3.300
vga_green[1]	OUT	AA22	LVCMOS33*	3.300
vga_green[0]	OUT	AB22	LVCMOS33*	3.300
vga_red (4)	OUT		LVCMOS33*	3.300
vga_red[3]	OUT	V18	LVCMOS33*	3.300
vga_red[2]	OUT	V19	LVCMOS33*	3.300
vga_red[1]	OUT	U20	LVCMOS33*	3.300
vga_red[0]	OUT	V20	LVCMOS33*	3.300
Scalar ports (13)				
vga_hsync	OUT	AA19	LVCMOS33*	3.300
vga_vsync	OUT	Y19	LVCMOS33*	3.300

Şekil 3.13: Vivado programında VGA pinlerinin ataması.

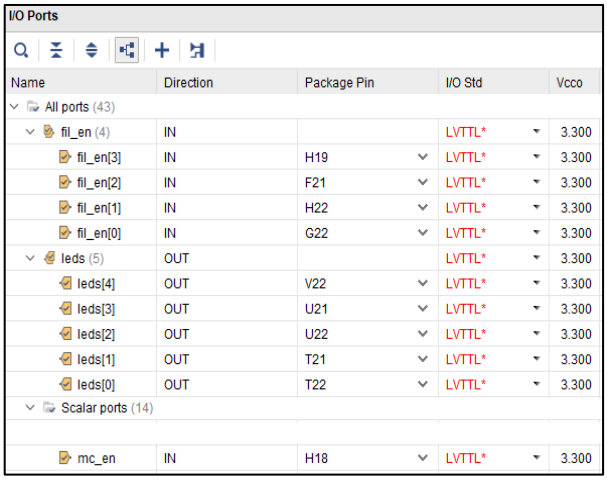
3.5.3. LED ve Anahtar Bağlantıları

ZedBoard üzerinde 8 adet kullanıcı LED'i (LD0-LD7) ve 8 adet dip anahtarı (SW0-SW7) mevcuttur. Bu çalışmada 5 adet LED, 5 adet anahtar kullanılmıştır. LED'ler, anahtarların durumunu göstermektedir. Anahtarların kullanım amaçları; kamera register ayarlama konfigürasyonun bittiğinin gösterilmesi (LD0), medyan filtreleme (SW1-LD1, SW2-LD2) ve gauss filtreleme (SW3-LD3, SW4-LD4) yapıldığının gösterilmesi, eksik komponent tespiti için lehimli alanların (SW5) gösterilmesi içindir.

Tablo 3.6: Zynq pini ile LED ve Anahtar pinleri bağlantısı gösterimi [33].

LED İsmi	Zynq Pini	Anahtar İsmi	Zynq Pini
LD0	T22	SW0	--
LD1	T21	SW1	G22
LD2	U22	SW2	H22
LD3	U21	SW3	F21
LD4	V22	SW4	H19
LD5	--	SW5	H18

Tablo 3.6'ya göre Vivado programında Şekil 3.14'teki gibi pin atamaları yapılmıştır. "fil_en" ile ifade edilen port, filtreleme için kullanılan anahtarları; "mc_en" ile ifade edilen port eksik komponent gösterimi için kullanılan anahtarı; "leds" yazan portlar ise kullanıcı LED'lerini ifade eder.



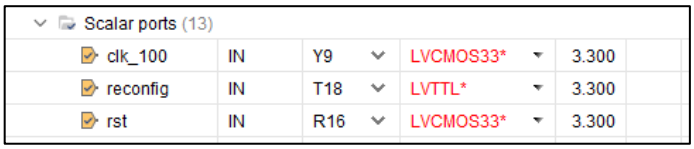
Şekil 3.14: Vivado programında LED ve Anahtar pinlerinin ataması.

3.5.4. Saat Kaynağı, Reset ve Buton Pinleri Bağlantısı

ZedBoard'un PS alt sistemi, 33.3333 MHz'lik saat kaynağı üretebilir. PL sistemi için dört adede kadar PLL tabanlı saat üretebilmektedir. Ayrıca yerleşik 100 MHz osilatör PL alt sistemi için saat girişi sağlamaktadır. Çalışmada 100 MHz'lik osilatörün sağladığı saat kaynağı

kullanılmıştır. Pin olarak Y9'a bağlıdır. Şekil 3.15'te ifade edilen "clk_100", 100 MHz'lik saat kaynağını gösterir.

ZedBoard, 7 adet kullanıcı butonuna sahiptir. PL tarafındaki 1 adet "BTNU" isimli kullanıcı butonu, basıldığında kamera registerlarını yeniden düzenleme amacıyla "reconfig" portuna T18 pini ile bağlanmıştır. Yine PL tarafındaki 1 adet "BTND" isimli kullanıcı butonu, basıldığında tüm sistemi resetleme amacıyla "rst" portuna R16 pini ile bağlanmıştır.



Şekil 3.15: Vivado programında Saat Kaynağı, Buton, Reset pinlerinin ataması.

3.6. VHDL BAĞLANTI ŞEMASI

Görüntü sensöründen gerçek zamanlı olarak görüntüyü alarak VGA monitörde göstermek, filtrelemek ve eksik komponent tespit etmek için VHDL kodlama yapılmıştır. Şekil 3.16'da Vivado modüllerinin VHDL bağlantı şeması gösterilmiştir.

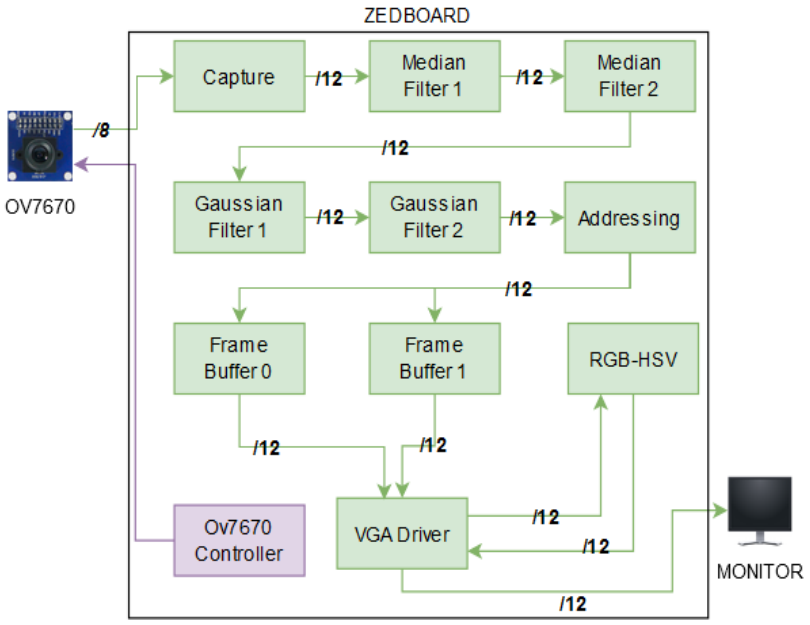
OV7670 görüntü sensörü, "OV7670 controller" (kontrolör) modülü ile kamera parametrelerini ayarlamak için SSCB protokolünü kullanır. SCCB, I2C uyumlu bir arayüzdür. Komutlar I2C arayüzü üzerinden OV7670 görüntü sensörüne gönderilir. 8 bitlik veri iki kez alındığında, 1 verisi geçersiz olmak üzere toplam 15 bitlik bir piksel elde edilmektedir. Bu 15 bitlik bir piksel verisi, RGB555 formatında alınmaktadır. Her kanal 5 biti temsil etmektedir.

"Capture" (yakalama) modülü ile piksel verileri kameradan alınır. Bu piksel verilerinin 12 bit renk çözünürlüğüne sahip olan VGA monitörde gösterilebilmesi için en anlamlı 4 biti alınarak RGB444 formatında 12 bit şeklinde kullanılmaktadır. Her RGB renk kanalı için gürültü giderme amacıyla RGB444 formatında kullanıcının isteğine bağlı olarak bir veya iki defa medyan filtreleme işlemine alınır. Medyan filtreleme işleminin devamında veya medyan filtrelemeden ayrı olarak kullanıcının isteğine bağlı, bir veya iki defa gauss filtreleme işlemi yapılabilmektedir.

"Adressing" (adresleme) modülünde her bir piksel adreslenerek iki adet "frame buffer" (çerçeve arabelleği) modülüne gönderilir. Frame buffer modülü kameradan gelen bir görüntü çerçevesini saklar. Görüntü sensörü maksimum 30 fps sahiptir. 30 fps ile çalıştığı için bir

görüntü çerçevesini alma hızı, 60 fps ile çalışan VGA'ya göre 2 kat daha yavaştır. Bu sebeple iki adet frame buffer kullanılmıştır. İki modülde de aynı okuma ve yazma portları bulunmaktadır. VGA tarafından frame buffer modülüne okuma adresi gönderilerek bir görüntü çerçevesini oluşturan piksel verileri alınarak monitöre basılmaktadır.

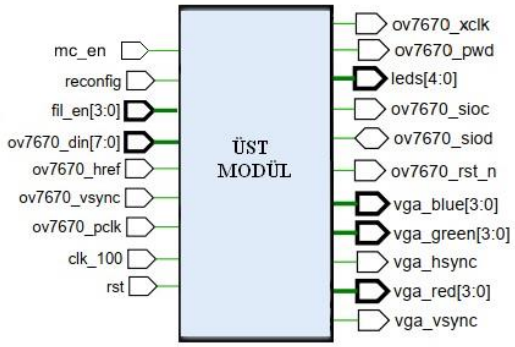
Kullanıcı isteğine bağlı olarak ekranda eksik komponentlerin tespiti için lehimli alanların gösterilmesi amaçlanmıştır. VGA sürücüsü modülü, RGB444 formatından gelen piksel verisini "RGB-HSV" modülüne göndermektedir. Bu modülde HSV değerleri hesaplanmaktadır. Hesaplanan HSV değerlerine göre lehim rengi aralığı belirlenerek bu aralıkta olan kısım, VGA monitörde kırmızı renkte gösterilmektedir. Bu aralığın dışında kalan kısım için VGA'dan alınan RGB444 verisi direkt olarak ekrana basılmaktadır. EK-1'de Vivado programında yazılmış olan VHDL kodlarının RTL şematikte gösterimi mevcuttur. EK-2'de RGB-HSV modülü eklenmiş RTL şeması mevcuttur.



Şekil 3.16: VHDL bağlantı şeması.

3.7. DONANIM TASARIMI ÜST MODÜLÜ

Üst modül, çalışmanın içerdiği tüm komponentleri bir araya getirmektedir. Bu modül, saat kaynağı, resetleme pini ve reconfig tuşu, filtreleme anahtarı, eksik komponent anahtarı, ov7670 arayüz pinleri ve VGA sinyallerini içermektedir. Şekil 3.17'de üst modüle ait tüm giriş-çıkış portları şematik olarak gösterilmiştir.



Şekil 3.17: Üst modül giriş-çıkış portları şematik gösterimi.

Her port için üst modül entity (varlık) bildiriminin altında port tanımlaması yapılmıştır. VHDL kodu ile yapılan bu port tanımlamaları açıklamalarının altına eklenmiştir.

mc_en (eksik komponent etkinleştirme) anahtarı, eksik komponentin monitörde gösteriminin seçimi için kullanılmaktadır. ZedBoard üzerinde kullanıcı anahtarına bağlıdır. mc_en portu VHDL kodu aşağıdaki gibidir:

```
mc_en           : in std_logic; --missing component enable
```

reconfig (yeniden yapılandırma) butonu ile, kamera registerlarının tekrar ayarlanmasını sağlamaktadır. ZedBoard üzerinde buton U'ya (BTNU) basıldığında kamera registerları yeniden yapılandırılır. reconfig portu VHDL kodu aşağıdaki gibidir:

```
reconfig        : in std_logic; --reconfiguration
```

fil_en [3:0] (filtre etkinleştirme) anahtarları, 4 bitten oluşan medyan ve gauss filtreleme seçimi için kullanılmaktadır. Her filtre ZedBoard üzerinde bir kullanıcı anahtarına bağlıdır. Bu anahtarlar açıp kapatılarak filtre etkinleştirme işlemi yapılmaktadır. fil_en portu VHDL kodu aşağıdaki gibidir:

```
fil_en          : in std_logic_vector(3 downto 0); --filter enable
```

ov7670_din [7:0] (veri girişi) portu, kameradan gelen piksellerin alındığı 8 bitlik veri yoludur.

```
ov7670_din      : in std_logic_vector(7 downto 0); --data in
```

ov7670_href portu, yatayda piksel verilen anı işaret eder. Aktif bölgede '1' olan, ön sundurma ve arka sundurma bölgesinde '0' olan porttur.

```
ov7670_href     : in std_logic; --horizontal ref
```

ov7670_vsync (dikey senkronizasyon) portu, kameradan yeni bir görüntü çerçevesi alınarak başladığını gösterir.

```
ov7670_vsync    : in std_logic; --vertical synchronization
```

ov7670_pclk (pixel clock) portu, kameradan gelen görüntünün senkron olduğu saat sinyalidir. Bu değer kameranın veri sayfasında belirtildiği üzere 24 MHz değerindedir. Her 2 piksel clock geldiğinde 1 piksel alınmaktadır.

```
ov7670_pclk     : in std_logic; --pixel clock: 24 MHz
```

clk_100 (clock 100 MHz) portu, ZedBoard'da yer alan osilatörden alınan 100 MHz'lik saat sinyalini kullanarak tasarımda ihtiyaç olan diğer saat sinyallerini elde etmek için kullanılmıştır.

```
clk_100         : in std_logic; --100 MHz
```

rst (reset) portu, aktif yüksek modda çalışır ve tüm sistemi resetlemek için kullanılır. Bu porta '1' verildiğinde sistem resetlenmektedir. ZedBoard üzerinde buton D'ye (BTND) bağlanmıştır. Böylece buton D'ye basıldığında sistemi resetlemektedir.

```
rst             : in std_logic; --active high
```

ov7670_xclk (X Clock) portu, dışarıdan kamera içerisindeki PLL'e giriş olarak verilen ve kameranın çalışmasını sağlayan piksel saatinin (PCLK) üretilmesini sağlayan saat sinyalidir.

```
ov7670_xclk     : out std_logic; --X Clock: 24 MHz
```

ov7670_pwd (güç kapatma) portu; aktif yüksek modda olan, kameraya güç verilmemesi için kullanılan ve kameranın PWDN pini ile bağlantısı olan porttur. Bu porta '0' verildiğinde kameraya güç verilerek çalışması sağlanır.

```
ov7670_pwd      : out std_logic; --power down          ov7670_pwd          <= '0'; --active high
```

leds [4:0] portu, bir biti ile register konfigürasyonunun tamamlandığını ve diğer dört biti ile filtreleme anahtarlamasında kullanıcıya filtrenin seçildiğini LED ile gösteren 5 bitlik porttur.

```
leds            : out std_logic_vector(4 downto 0);
```

ov7670_sioc (serial input output clock – seri giriş çıkış saati) portu, i2c haberleşme saati portudur.

```
ov7670_sioc     : out std_logic; --i2c clock
```

ov7670_siod (serial input output data – seri giriş çıkış verisi) portu, i2c haberleşme verisi için kullanılan porttur.

```
ov7670_siod     : inout std_logic; --i2c data
```

ov7670_rst_n portu, aktif düşük mod olarak çalışan kameranın reset pini ile bağlantılı olan porttur. Düşük modda çalıştığı için çalışma mantığı terstir. Buna göre bu porta '0' verilirse devre resetlenmektedir. '1' verilirse devre resetlenmemektedir.

```
ov7670_rst_n    : out std_logic; --active low       ov7670_rst_n        <= '1'; --active low
```

vga_red [3:0] kırmızı renk, **vga_green [3:0]** yeşil renk ve **vga_blue [3:0]** mavi renk için olan VGA monitörünün çıkış portları, 12 bitlik RGB444 piksel verisini temsil etmektedir.

```
vga_red         : out std_logic_vector(3 downto 0); --Zedboard VGA renk cozunurlugu 12 bit
vga_green       : out std_logic_vector(3 downto 0);
vga_blue        : out std_logic_vector(3 downto 0);
```

vga_hsync ve **vga_vsync** portları, VGA monitörü yatay ve dikey senkronizasyon portlarıdır.

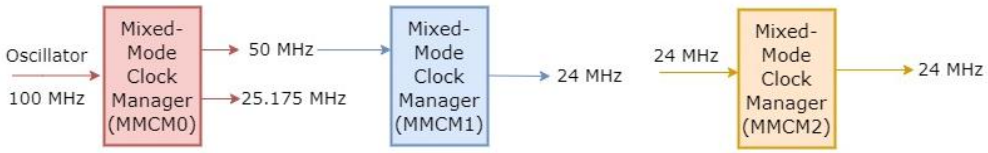
```
vga_hsync       : out std_logic; --horizontal synchronization
vga_vsync       : out std_logic  --vertical sync
```

3.8. SAAT YÖNETİCİSİ IP BLOKLARI

Tasarımda OV7670 görüntü sensörünün PCLK ve XCLK pinleri için 24 MHz saat sinyaline, VGA ve frame buffer'ın okuma kısmı için 25.175 MHz saat sinyaline ihtiyaç vardır.

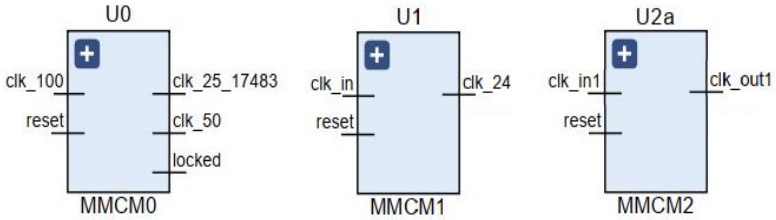
ZedBoard, osilatör ile 100 MHz üretebilen bir saat sinyaline sahiptir. Bu saat sinyalinden tasarım içerisinde kullanılacak olan diğer saat sinyalleri elde edilebilmektedir. Bunun için tasarıma Vivado programındaki Karışık Mod Saat Yöneticisi (MMCM) IP blokları eklenmiştir.

Şekil 3.18'de şematik olarak MMCM IP bloklarına yer verilmiştir. MMCM0 bloğu, 25.175 MHz ve kalan kısmı ile 50 MHz değerinde saat sinyali üretmektedir. İnce saat ayarı ile üretim sağlandığı için aynı anda hem 25.175 MHz hem de 24 MHz saat sinyali üretimine Vivado programı izin vermediği için ekstra olarak MMCM1 bloğu eklenmiştir. MMCM1 bloğu ile 50 MHz saat sinyalinden 24 MHz saat sinyali üretilmiştir. MMCM1 bloğunda yer alan 24 MHz'lik saat sinyali çok fazla modülde kullanıldığı için tasarımda zamanlama problemi oluşturduğundan ve PCLK saat sinyalinin stabilizesi için ayrıca MMCM2 bloğu eklenerek 24 MHz PCLK saat sinyali elde edilmiştir.



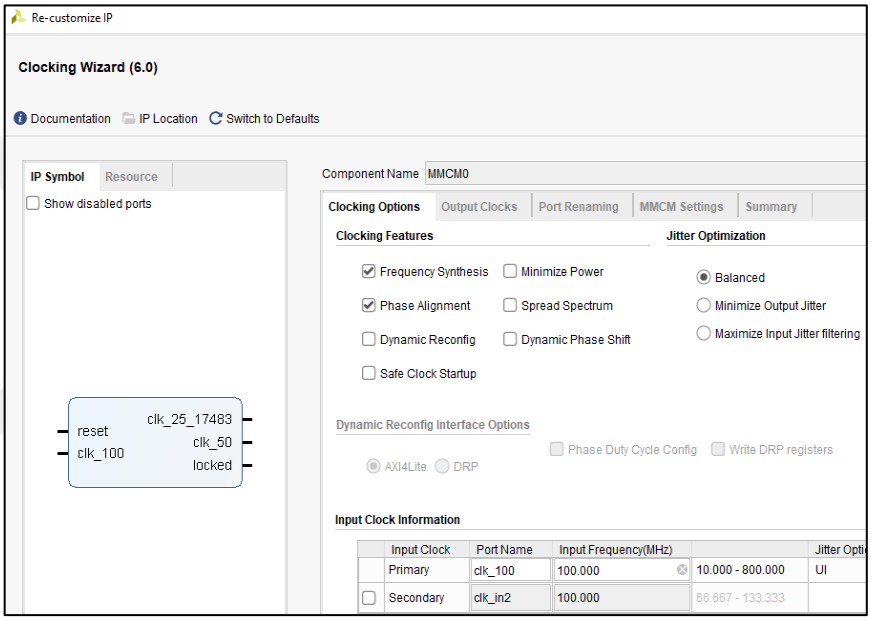
Şekil 3.18: Karışık Mod Saat Yöneticisi (MMCM) modüllerinin şematik gösterimi.

Şekil 3.19'da bu IP blokların Vivado programında RTL şematiğinde gösterimi yer almaktadır. MMCM0'da bulunan 50 MHz'lik saat sinyali, MMCM1 bloğunun girişine clk_in portu olarak verilerek clk_24 isimli 24 MHz'lik saat sinyali üretilmiştir. MMCM2 bloğunun girişinde clk_in1 portu kısmından 24 MHz verilerek stabil bir şekilde çıkışta clk_out1 portundan 24 MHz tekrar alınmıştır. MMCM0 bloğunda kullanılan locked çıkış portu, 50 MHz ve 25.175 MHz sinyallerinin aynı anda düzgün bir şekilde çıkış verdiğini gösteren 1 bitlik porttur.

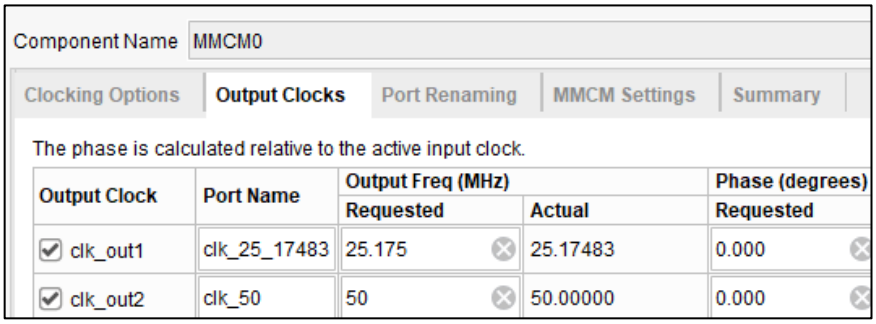


Şekil 3.19: Karışık Mod Saat Yöneticisi (MMCM) IP blokları.

Şekil 3.20'de MMCM0 IP Bloğu için giriş clocku 100 MHz olarak ayarlanmıştır. Şekil 3.21'de ise çıkış clockları ayarlanmıştır. Buna benzer ayarlamalar diğer MMCM IP bloklar içinde yapılmıştır.



Şekil 3.20: MMCM0 IP blok giriş frekansı ayarı.



Şekil 3.21: MMCM0 IP blok çıkış frekansları ayarı.

MMCM0 IP bloğu üst modülde MMCM0 komponenti olarak tanımlanmış, VHDL kodu aşağıdaki gibidir:

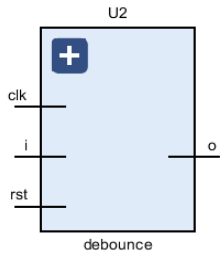
```
component MMCM0 is --Mixed Mode Clock Manager: 100 MHz --> 25.175 MHz + 50 MHz
port
(
    clk_25_17483    : out std_logic; --25.175 MHz: VGA + frame buffer
    clk_50          : out std_logic; --50 MHz: Debounce + XClk Uretimi (24 icin) + Controller
    reset           : in std_logic;  --PLL reset
    locked          : out std_logic; --Clk cikislarinin duzgun oldugu an
    clk_100         : in std_logic --Zedboard 100 MHz
);
end component MMCM0;
signal clk_25_17483 : std_logic;
signal clk_50       : std_logic;
signal m_locked     : std_logic;
signal rst_locked   : std_logic;
```

MMCM0 bloğunun portlarına aşağıdaki gibi port atamaları yapılmıştır. Örneğin clk_25_17483 MHz sahip portuna, clk_25_17483 MHz isimli sinyal ataması yapılmıştır. reset portuna, üst modülün rst portu atanmıştır. rst_locked sinyali, m_locked sinyalinin değili alınarak tasarımlarda kullanılmaktadır.

```
U0 : MMCM0
port map
(
    clk_25_17483    => clk_25_17483,--MMCM0 sinyal
    clk_50          => clk_50,      --MMCM0 sinyal
    reset           => rst,         --top port
    locked          => m_locked,    --MMCM0 sinyal
    clk_100         => clk_100      --top port
);
rst_locked          <= not m_locked; --Clk 50 ve 25'in oldugu tasarimlar icin
```

3.9. BUTON ARKI MODÜLÜ

Buton arkı (debounce) modülü, konfigürasyon butonuna ilk basıldığında ortaya çıkan dalgalanma sinyalini ortadan kaldırarak çıkışta temiz bir sinyal geçişi elde etmek için kullanılmıştır. Şekil 3.22'de debounce modülünün Vivado programında RTL şematiğinde gösterimi yer almaktadır.



Şekil 3.22: Debounce modülü.

Debounce modülü, üst modülün U2 komponenti olarak tanımlanmıştır. VHDL kodu aşağıdaki gibidir. 50 MHz'e sahip clock, bu tasarımda kullanılmıştır. 50 MHz yerine farklı saat sinyali tanımlanabilir, ancak tasarımın stabilizesi için debounce modülü çıkışına bağlı diğer modüllerin de bu saat sinyaliyle aynı saat sinyaline sahip olması gerekmektedir.

```
component debounce is --buton arki, sekme boyunca gecikme
port
(
    rst : in std_logic; --PLL reseti
    clk : in std_logic; --50 MHz
    i   : in std_logic; --butona basildiginda gelen sinyal
    o   : out std_logic --cikistaki flag / resend
);
end component debounce;
signal deb_btn  : std_logic;
```

Debounce bloğunun portlarına aşağıdaki gibi port atamaları yapılmıştır. Clk portuna, MMCM0 sinyali olan clk_50 MHz isimli sinyal ataması yapılmıştır. Reset portuna, MMCM0 sinyali olan rst_locked sinyali atanmıştır. Bir clock geciktirilecek butona basma portu olan ‘i’, üst modül ‘reconfig’ portuna; çıkıştaki sinyal ise kontrolör (controller) modülündeki resend portuna atanmıştır. Kamera registerlarının yeniden ayarlanmasını sağlayan reconfig portu, fiziksel olarak ZedBoard üzerinde yer alan buton U (BTNU) ile bağlıdır.

```
U2 : debounce
port map
(
    rst => rst_locked, --MMCM0 sinyal
    clk => clk_50,     --MMCM0 sinyal
    i   => reconfig,   --top port
    o   => deb_btn     --debounce sinyal: controller resend butonu
);
```

Debounce modülünün architecture kısmında sinyal atamaları aşağıdaki gibi yapılmıştır. Tuşa basıldığında 50 MHz ile örnekleme yapılmaktadır. Tasarım 50 MHz’de çalıştığından 1 saniye 50.000.000 kez sayım olarak tutulmaktadır. Sayaç (counter) kullanarak 24 bit ‘1’ ile saydırma işlemi yapılmıştır. Bu da decimal olarak 16.777.215 kez saymaya karşılık gelir. Yaklaşık saniyenin üçte biri kadar bekleme yapılmaktadır. Tuş 0,3 saniye boyunca bloke edilmiştir. Böylece tuşa basıldıktan 0,3 saniye sonra tekrar basmaya izin veren tasarım oluşturulmuştur. ‘i_d’ sinyali, ‘i’ giriş sinyalinin 1 clock geciktirilmiş halidir.

```
architecture RTL of debounce is

    signal counter  : std_logic_vector(23 downto 0); --tusa basildiginda 24 bit counter sayimi bitince cikisa 1 atama
    signal i_d      : std_logic;                     --i_delay: i geciktirme
    signal flag     : std_logic;                     --tusa basilan ani yakalama, tus sekmemesi icin
```

Tüm işlemler, process bloğu içerisinde saat sinyaline bağlı olarak saat sinyalinin yükselen kenarında yapılmaktadır. ‘rst’, reset portu ‘1’ alındığında tüm değerler ‘0’a çekilerek senkron resetleme yapılmaktadır.

```
begin

    PROC_DEBOUNCE: process(clk) begin
        if rising_edge(clk) then --senkron resetleme
            if (rst = '1') then
                counter <= (others => '0');
                o       <= '0';
                i_d     <= '0';
                flag    <= '0';
```

‘rst’, reset portu ‘0’ alındığında i giriş portuna, i_d sinyali atanır. (i=’1’) olması durumunda tuşa basılmıştır ve aynı zamanda i_d, 1 clock gecikmiş hali olarak (i_d=’0’) alındığında flag ‘1’ gönderilir.

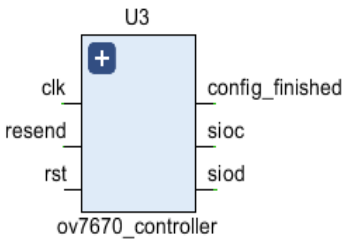
```
else
    i_d <= i;
    --tus icin i:0 i_delay:0
    if (i_d = '0' and i = '1') then --tusa basinca i='1', i_d='0'
        flag    <= '1';
    end if;
```

Flag ‘1’ atandığında onluk (decimal) sisteme göre 24 bit olan sayaç, ‘else’ bloğunda birer birer arttırılarak sayım işlemi başlatılır. Her seferinde çıkışa ‘1’ değerinde flag atanır. Bu durum if bloğunda yer alan sayaç, on altılık (hexadecimal) sisteme göre x”FFFFFF” şeklinde 24 bit ‘1’ değerine gelinceye kadar devam eder. İkili (binary) sisteme göre ‘1111 1111 1111 1111 1111 1111’ olana kadar saymaktadır. Sayaç bittiğinde sayaçtaki tüm bit değerleri sıfırlanır, çıkışa ‘1’ atanır ve process, architecture bloğu sonlandırılır.

```
if (flag = '1') then --flag=1 olunca counter ile sayma baslatma
    if (counter = x"FFFFFF") then --counter 24 kez sayma kontrolu
        counter <= (others => '0');
        flag    <= '0';
        o       <= '1'; --cikisa 1 atama
    else
        counter <= counter + '1';
        flag    <= '1';
        o       <= '0';
    end if;
else
    counter <= (others => '0');
    o       <= '0';
```

3.10. KONTROLÖR MODÜLÜ

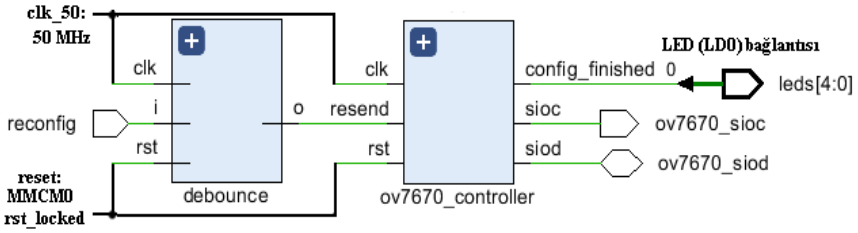
Kontrolör (controller) modülü, kamera registerlarının ayarlanmasını ve i2c haberleşmesini sağlayan modüldür. Şekil 3.23’te controller modülünün Vivado programında RTL şematiğinde gösterimi yer almaktadır.



Şekil 3.23: Controller modülü.

Şekil 3.24’te debounce ve controller modülü bağlantısı gösterilmiştir. Controller modülünün, debounce modülü ile senkron çalışması için debounce modülüne bağlı olan 50 MHz değerine

sahip clock, bu modülde de kullanılmıştır. Konfigürasyonun bittiğini göstermek için çıkışta "config_finished" isimli port mevcuttur. Bu port ZedBoard üzerinde LED0'a (LD0) bağlıdır. Böylece kullanıcıya konfigürasyonun tamamlandığı gösterilir. Kamera "sioc" ve "siod" portları i2c haberleşme için kullanılmıştır.



Şekil 3.24: Debounce-Controller modülü bağlantısı.

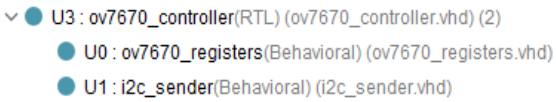
Kamera veri sayfasında (datasheet) yer alan register ayarlamalarında, yazma için adres "42", okuma için adres "43" olarak tanımlanmıştır. Kamera yazma adresi, controller modülünün varlık (entity) bildirimi altında "generic" biçiminde tanımlanmıştır. Generic biçiminde tanımlanması ile tasarımda tam sayı biçiminde olan belirli bir bilgi tanımlamasını varlığa (entity) aktarır. Üst modülde, controller modülünün generic map ataması yapılmıştır.

```
generic
(
    G_CAM_ADDR          : std_logic_vector(7 downto 0) := x"42"
);
```

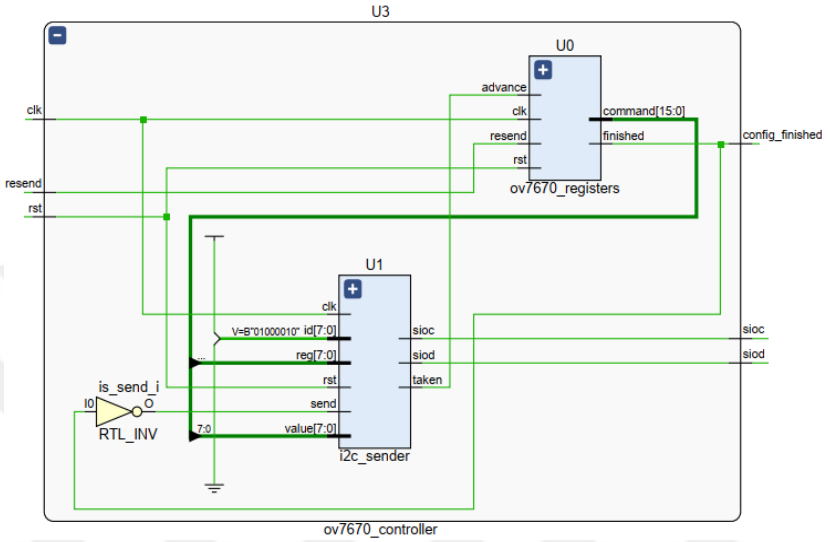
Debounce modülünün "o" çıkış portu ile controller modülünün "resend" portunu birbirine bağlamak için "deb_btn" sinyali kullanılmıştır. İki port da bu sinyale atanarak birbirine bağlanmış olur. "oco_config_finished" sinyali, "config_finished" portuna bağlanmış ve bu portun çıkışta led butonunun 0.biti olarak atanması sağlanmıştır. Port atamaları aşağıda gösterilmiştir.

```
port map
(
    rst                 =>  rst_locked,
    clk                 =>  clk_50,
    resend              =>  deb_btn,
    config_finished     =>  oco_config_finished, --controller sinyali
    sioc                =>  ov7670_sioc, --top
    siod                =>  ov7670_siod --top
);
leds(0)     <= oco_config_finished; --konfigurasyon ledi
```

Controller modülü, üst modülün U3 komponenti olarak tanımlanmıştır. VHDL kodunda U0 ve U1 olmak üzere iki alt modülü mevcuttur. U0 alt modülü register ayarı için, U1 alt modülü i2c haberleşmesi için oluşturulmuştur. Vivado programında kaynaklar (sources) kısmında gösterimi aşağıdaki gibidir:



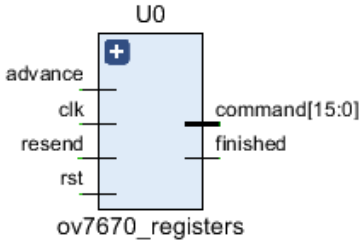
Şekil 3.25'te bu iki modülün RTL şemasında bağlantısı gösterilmiştir. I2C modülünde bulunan "taken" portu gönderilen verinin alındığını gösteren porttur. "taken" portu, '1' değerini aldığında bekleme yapılmaktadır. Bu port, register modülünün "advance" portuna bağlıdır. Böylece register durum makinesi adresi birer birer arttırılarak register ayarlamaları yapılır. "taken" portu, '0' değerini aldığında ise "send" ile register ayarı kameraya gönderilir.



Şekil 3.25: Controller alt modülleri: I2C modülü ve register modülü bağlantısı.

3.10.1. Register Modülü

Register modülü, kamera ayarlamalarının yapıldığı modüldür. I2C modülünü kullanarak kamera registerlarının ayarlamasını sağlar. Şekil 3.26'da register modülünün Vivado programında RTL şematiğinde gösterimi yer almaktadır. Saat sinyali olarak controller modülü ile aynı 50 MHz ile çalışmaktadır. "resend" portu, debounce modülünün çıkış portuna ("o") bağlıdır. "advance" portu, I2C modülünün "taken" portuna bağlanmıştır. "command" portu 16 bitten oluşur. Bu port, I2C modülünün "reg" ve "value" portlarının "&" operasyonu ile birleştirilmesiyle meydana gelir. "finished" portu ise "config_finished" portu ile çıkıştaki LED0'a (LD0) bağlıdır.



Şekil 3.26: Controller alt modülü: Register modülü.

Controller architecture bildirimi altında register komponenti port atamaları yapılmıştır. Controller modülüne ait "config_finished" portunun, register modülüne ait "finished" portu ile bağlanabilmesi için "or_finished" sinyali tanımlanmıştır. Bu sinyal controller "config_finished" portuna atanarak port map içerisinde port bağlantısı aşağıdaki gibi sağlanmıştır. "advance" portunun, I2C modülünün "taken" portuna bağlanabilmesi için "is_taken" sinyali atanmıştır. 16 bitlik "command" portuna "or_command" sinyali atanarak I2C modülünün "reg" ve "value" portları ile bağlantısı elde edilmiştir.

```
begin

    config_finished     <= or_finished; --register: or_finished, controller: config_finished

    U0 : ov7670_registers
    port map
    (
        rst         => rst, --controller top portu
        clk         => clk,--controller top portu
        resend      => resend, --controller top portu
        advance     => is_taken, --i2c sinyali
        command     => or_command, --register sinyali
        finished    => or_finished --register sinyali
    );
```

Register modülünde entity bildirimi altında port tanımlamaları yapıldıktan sonra architecture bildirimi altında 16 bitlik "sreg" ve 8 bitlik "address" sinyalleri tanımlanmıştır. "sreg" sinyali, command çıkış portuna atanmıştır.

"address" sinyali tamamen bu modüle ait durum makinesi olarak kullanılmıştır. Kullanımı sayaç gibi düşünülebilir. "address" sinyalinin her bir sayısında "sreg" sinyaline değer atanacaktır. "sreg" sinyali onluk sisteme göre 16 olduğunda (on altılık sistemde x"FFFF"), "finished" portuna '1' atanarak register ayarlamasının bittiğini gösterilir.

```
architecture Behavioral of ov7670_registers is

    signal sreg     : std_logic_vector(15 downto 0) := (others => '0'); --command cikis portu icin
    signal address  : std_logic_vector(7 downto 0)  := (others => '0'); --state machine adresi

begin
    command     <= sreg; --command cikis portuna sreg sinyal atamasi
    finished    <= '1' when sreg = x"FFFF" else '0'; --sreg 16 bit '1' oldugunda finished '1'
```

Process bloğu içerisinde clockun yükselen kenarında reset '1' atandığında adress ve sreg değerleri sıfırlanır.

```
PROC_CONFIG_REGS: process(clk) begin
    if rising_edge(clk) then
        if (rst = '1') then
            address <= (others => '0');
            sreg    <= (others => '0');
```

Reset '0' atandığında; controller'a ait ve debounce çıkışına bağlı "resend" portu '1' ise durum seçimi yapılmaz. "advance" portu '1' ise adres bir arttırılarak bir sonraki duruma geçilir.

```
        else
            if (resend = '1') then --resend gelirse adres '0'lama
                address <= (others => '0');
            elsif (advance = '1') then --advance gelirse ilerleme:
                address <= address + '1'; --adres '0'di: Adres +1
            else
                address <= address;
            end if;
```

Durum makinesi adresi “case” yapısında tanımlanmıştır. Her arttığında bir sonraki durum için bazı değerleri aşağıda gösterilmiştir.

```
case (address) is
    when x"00" => sreg  <= x"1280"; -- COM7
    when x"01" => sreg  <= x"00F0"; -- GAIN
    when x"02" => sreg  <= x"1204"; -- COM7
    when x"03" => sreg  <= x"1180"; -- CLKRC
    when x"04" => sreg  <= x"0C00"; -- COM3
    .......
    .......
```

Örneğin ilk durum makinesi adresi on altılık sistemde x”00” durumundayken sreg sinyaline x”1280” değeri atanmıştır. On altılık sistemde kamera register adresi, 12; register değeri, 80 olarak alınmıştır. Tablo 3.7’de kamera veri sayfası register ayarlamaları kısmında register adresi 12 olan COM7 için gösterilmiştir [31].

Tablo 3.7: COM7 değiştirme register ayarı.

Adres	Register Adı	Ayarlanan Register Değeri	Açıklama
12 (HEX)	Common Control 7 (COM7)	80 (HEX) 1000 0000 (BINARY)	Bit[7] = 1 , Bit[6:0]=0 Bit[7]: SSCB Register Reset Ayarı 0: Değişim yok. 1: Tüm register varsayılan değerine resetler.

Başka bir register ayarlamasına örnek verilirse RGB555 formatında görüntü almak için adres x”08” durum makinesi kullanılmıştır. Bu durumda sreg sinyali “40F0” değerini alır. 40. register adresine “F0” register değeri atanmıştır.

```
    .......
    .......
    when x"05" => sreg  <= x"3E00"; -- COM14
    when x"06" => sreg  <= x"8C00"; -- RGB444
    when x"07" => sreg  <= x"0403"; -- COM1
    when x"08" => sreg  <= x"40F0"; -- COM15
    .......
    .......
    when x"40" => sreg  <= x"0196"; -- BLUE
    when x"41" => sreg  <= x"0240"; -- RED
    when others => sreg <= x"ffff";
end case;
```

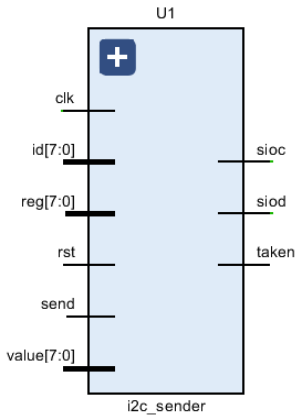
Tablo 3.8'de veri sayfasından alınan kısım incelendiğinde bit [7:6] = "11" olarak alındığından çıkış aralığı "00-FF" olarak ayarlanmıştır. Bu durumda "0-255" arasında RGB tüm renkler çıkış aralığına dahil edilmiştir. Bit [5:4] ="11" olarak alındığından RGB555 formatı seçilmiştir.

Tablo 3.8: COM15 değiştirme register ayarı.

Adres	Register Adı	Ayarlanan Register Değeri	Açıklama
40 (HEX)	Common Control 15 (COM7)	F0 (HEX) 1111 0000 (BINARY)	Bit[7:6]: Data formatı – Çıkış aralık ayarı etkinleştirme 0x: Çıkış aralığı: [10] - [F0], 10: Çıkış aralığı: [01] - [FE], 11: Çıkış aralığı: [0] -[FF] Bit[5:4]: RGB 555/565 seçimi x0: Normal RGB çıkışı, 01: RGB 565, 11: RGB 555, Bit[3:0]: Rezerve.

3.10.2. I2C Modülü

I2C (Inter-Integrated Circuit) modülü, kamera ayarlamalarının yapılmasını sağlayan modüldür. OV7670 görüntü sensörü, "OV7670 controller" (kontrolör) modülü ile kamera parametrelerini ayarlamak için SSCB protokolünü kullanır. SCCB, I2C uyumlu bir arayüzdür. Komutlar I2C arayüzü üzerinden OV7670 görüntü sensörüne gönderilir. Şekil 3.27'de I2C modülünün Vivado programında RTL şematiğinde gösterimi yer almaktadır.



Şekil 3.27: Controller alt modülü: I2C modülü.

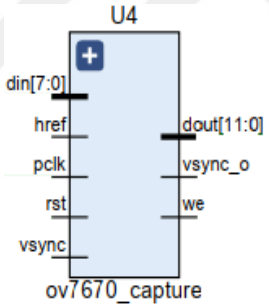
50 MHz saat sinyali ile çalışmaktadır. 8 bitlik "id" portu, kamera yazma adresi olan x"42" (binary olarak: 0100 0010) değerini alır. 8 bitlik "reg" ve "value" portları birleşerek 16 bitlik register modülünün "command" çıkış portuna bağlanır ve "sreg" ismi verilen sinyali

oluşturarak kamera register adresi ve değerini göstermiş olur. “taken” portu, register modülünün “advance” portuna bağlanmıştır. “send” portu, register modülünün “finished” ve controller modülünün “config_finished” çıkışına LED0 (LD0) ile bağlıdır. I2C modülünün, controller üst modülünde port map atamasının VHDL kodu aşağıdaki gibidir:

```
U1 : i2c_sender
port map
(
    rst     => rst,
    clk     => clk,
    siod    => siod,
    sioc    => sioc,
    taken   => is_taken, --register advance portu, is_taken sinyali ile bagli
    send    => is_send,
    id      => G_CAM_ADDR,
    reg     => or_command(15 downto 8), --command portu son 7 bit reg adresi
    value   => or_command(7 downto 0) --command portu ilk 7 bit reg value
);
is_send     <= not or_finished; --konfigurasyon bitmemisse send ile yollamaya devam et
```

3.11. GÖRÜNTÜ YAKALAMA MODÜLÜ

Görüntü yakalama (capture) modülü, kameradan gelen görüntüye ait 8 bitlik veriyi yakalayan modüldür. Şekil 3.28’de bu modüle ait Vivado programı RTL şeması yer almaktadır.



Şekil 3.28: Capture modülü.

Kameranın verebileceği maksimum hız 30 fps’dir. 24 MHz ile 30 fps hızında çalışmaktadır. Bu yüzden “PCLK” giriş portu, 24 MHz olarak tanımlanmıştır. Bu saat sinyali, MMCM2 modülü ile üretilir. Kamera veri sayfasında belirtildiği gibi kamera maksimum 8 bitlik veri alımına izin vermektedir.

“din[7:0]” (veri girişi – data in) portu, kameradan gelen 8 bitlik veri girişini temsil eder. “href” portu, yatayda piksel alımını gösteren giriş portudur. “din[7:0]” verileri geldiği sürece href değeri ‘1’e çekilir.

“vsync” giriş portu ise, görüntüde bir çerçevenin başladığını gösterir. “vsync_o” çıkış portu, görüntünün bir çerçevesinin alınma işleminin tamamlanarak medyan filtreleme bloğunun “vsync_in” girişine bağlandığı porttur.

“we” (write enable – yazma etkinleştirme) portu, ‘1’ değerini aldığında frame buffer’a yazma emrini, ‘0’ değerini aldığında ise okuma emri vermektedir. Medyan filtreleme bloklarının “pixl_vld” (geçerli piksel) portuna bağlıdır.

“dout[11:0]” (veri çıkışı – data out) portu, capture modülünden 12 bitlik piksel çıkış verilerini gösterir. Bu verilerin, sinyal aracılığıyla medyan filtrenin “pixel_in[3:0]” piksel giriş portuna aktarılması sağlanır.

Üst modülde port atamaları aşağıdaki gibi yapılmıştır. Saat sinyali bağlantısında MMCM2 modülünün 24 MHz PCLK portu bağlantısı için “cam_clk” sinyali kullanılmıştır. Diğer portlara üst modülün kamera portları bağlanmıştır. Capture modülünün “we” yazma etkinleştirme portunun, filtreleme portlarına bağlanabilmesi için “oca_we” capture sinyaline atanmıştır. Aynı durum “vsync_o” ve “dout” portları için de geçerlidir.

```
U4 : ov7670_capture
port map
(
    rst     => rst_locked,
    pclk    => cam_clk,
    vsync   => ov7670_vsync,
    href    => ov7670_href,
    din     => ov7670_din,
    vsync_o => oca_vsync_o,
    we      => oca_we(0),
    dout    => oca_dout
);
```

Görüntü; kameradan 1 piksel saati (PCLK) sinyali ile 8 bit olarak, 2 PCLK ile 16 bit olarak alınır. Bu 16 bitlik verinin 1 biti gereksiz (X), geriye kalan 15 biti RGB555 formatı ile alınmaktadır. RGB555 formatına göre 1 piksel aşağıdaki gibi ifade edilir:

1.PCLK sonrası veri girişi (1.BYTE) → **din[7:0] = [X** R_4 R_3 R_2 R_1 R_0 G_4 G_3 **]**

2.PCLK sonrası veri girişi (2.BYTE) → **din[7:0] = [** G_2 G_1 G_0 B_4 B_3 B_2 B_1 B_0 **]**

VGA, 12 bit renk çözünürlüğüne sahip olduğu için RGB555 formatında her renk için en anlamlı 4 bit alınarak RGB444 formatına çevrilmektedir. Bu çevrim, capture modülünün architecture bildiriminin altında yer alan aşağıdaki gibi tanımlanan 16 bitlik “d_latch” (data latch – veri tutucu) sinyali ile sağlanır. Bu sinyal kameradan gelen her 8 biti, 2 PCLK boyunca tutmaktadır.

```
signal d_latch      : std_logic_vector(15 downto 0) := (others => '0');
```

Process içerisinde “din[7:0]” portuna gelen ilk 8 bit “d_latch” sinyalinin ilk 8 biti olarak alınır. Sola kaydırma işlemi (left shift register) yaparak “d_latch” sinyalinin ilk 8 biti son 8 bitine eşitlenir.

```
d_latch         <= d_latch(7 downto  0) & din;
```

RGB555 formatındaki her renk için en anlamlı 4 bitin alınarak RGB444 formatına çevrilmesi için “d_latch” sinyalinde kırmızı renk için 10.bit, yeşil renk için 5.bit, mavi renk için 0.bit alınmamaktadır. “pxl” isimli 12 bitlik sinyal kullanılarak çıkış portu olan “dout” portuna atanmıştır.

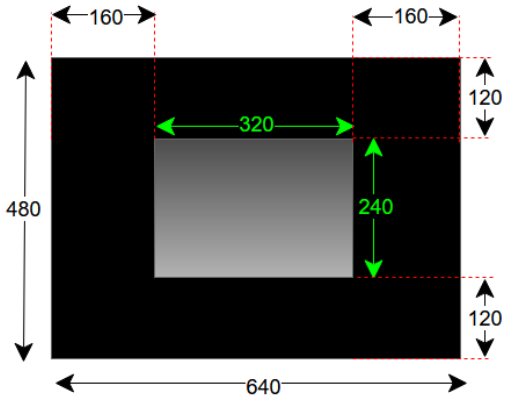
```
pxl             <= d_latch(14 downto 11) & d_latch(9 downto 6) & d_latch(4 downto 1);

dout              <= pxl;
```

Böylece çıkışta RGB444 formatına göre toplam 12 bite sahip olan 1 piksel aşağıdaki gibi ifade edilir:

dout[11:0] = [R4 R3 R2 R1 G4 G3 G2 G1 B4 B3 B2 B1 **]**

Capture modülünde ayrıca görüntü boyutunun 640x480 yerine 320x240 olarak ekranda gösterilmesi için VHDL kodlama yapılmıştır. Bu kodlamaya göre Şekil 3.29’daki gibi monitörde 320x240 boyutunda gerçek zamanlı görüntü basılmaktadır.



Şekil 3.29: 640x480 görüntü boyutunun 320x240 olarak gösterimi.

320x240 boyutuna ayarlama yapılması için 640 adet sütun sayımı için “colm_count” ve satır geçişleri için “line_count” isimli 2 adet sinyal tanımlanmıştır.

Aşağıda gösterilen iç içe if bloğunda sütun sayacı, 0.pikselden 639.piksele gelinceye kadar toplam 640 adet piksel sayımı yapar. En içteki if bloğunda sütun sayacı, 160 piksel saymışsa ve aynı zamanda 480’den küçükse; satır sayacı, 120’den büyük ve 360’tan küçükse taralı alanda

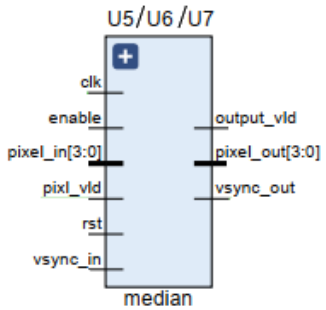
kalan bölgede piksel çıkışı alınır ve yazma portu “we”, ‘1’e çekilir. Sütun sayacı birer birer arttırılır ve 639’a eşit olduğunda en üstteki if bloğunda sıfırlanır. Satır sayısı bir arttırılarak diğer satıra geçilir. Böylece ekranın tam ortasında 320x240 boyutundaki alanda görüntü pikselleri alınmış olur.

```
if (colm_count = std_logic_vector(to_unsigned(639, colm_count'length))) then
    colm_count  <= (others => '0');
    line_count  <= line_count + '1';
    we          <= '0';

else
    colm_count  <= colm_count + '1';
    if (colm_count >= std_logic_vector(to_unsigned(160, colm_count'length)) and
        colm_count < std_logic_vector(to_unsigned(480, colm_count'length)) and
        line_count >= std_logic_vector(to_unsigned(120, line_count'length)) and
        line_count < std_logic_vector(to_unsigned(360, line_count'length))) then
        dout    <= pxl;
        we      <= '1';
    else
        we      <= '0';
```

3.12. MEDYAN FİLTRELEME MODÜLLERİ

Medyan filtreleme modülü, kameradan gelen görüntüdeki tuz-biber gürültüsünü giderme amacıyla kullanılan modüldür. Şekil 3.30’da bu modülün 1. medyan filtrelemesine ait Vivado programı RTL şeması yer almaktadır. Capture modülü çıkışından alınan 12 bitlik RGB piksel verisi, bu blokta filtrelenmektedir. RGB piksel verisi, her bir renk için 4 bit olacak şekilde filtreleme işlemine alınır. Bu sebeple bu modülden her bir renk için 3 adet kullanılmıştır. U5, kırmızı renk; U6, yeşil renk; U7, mavi renk için 1.medyan bloğudur.

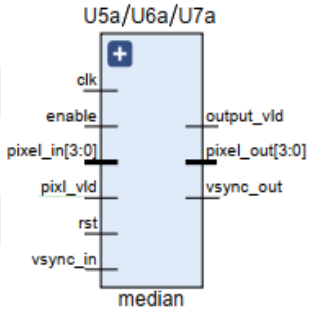


Şekil 3.30: 1. medyan filtreleme modülü.

“clk” giriş portu, capture modülü ile uyumlu çalışması için MMCM2 modülünün ürettiği PCLK 24 MHz olarak tanımlanmıştır. “enable” (etkinleştirme) portu, medyan filtre seçiminin aktif edilip edilmeyeceğinin kullanıcı tarafından ayarlamasını sağlayan porttur. ZedBoard üzerinde yer alan anahtarlara bağlanmıştır. “pixel_in[3:0]” portu, filtrelenecek 4 bitlik renk pikselini giriş olarak modüle alır. “pixel_out[3:0]” portu ise, filtrelenmiş 4 bitlik renk pikselini çıkış olarak

vermektedir. “output_vld” portu, geçerli piksel çıkış değeri alınmışsa ‘1’ değerini alır ve diğer modüllere 4 bitlik pixel_out[3:0] portunun aktarılmasını sağlar.“pixl_vld” portu, capture modülünden geçerli piksel verisi alınırsa ‘1’ değerini alır.“vsync_in” portu, VGA monitöre verilecek “vsync” sinyalinde filtrelemelerden dolayı gecikme oluşmaması için kullanılmıştır. “vsync_out” portu ise, diğer modüllere “vsync_in” girişi ile gecikme eklemeyi sağlamaktadır.

Şekil 3.31’de 1.medyan filtreleme modülü kaskadlanarak aynı VHDL kodundan 2.medyan filtre modülü oluşturulmuştur. Aynı şekilde bu medyan filtrede de RGB renk kanalları ayrı ayrı 3 adet filtreden geçirilir. U5a, kırmızı renk; U6a, yeşil renk; U7a, mavi renk için 2.medyan filtre bloğudur.



Şekil 3.31: 2. medyan filtreleme modülü.

Üst modülde, medyan filtreleme modüllerinden U5 ve U5a modüllerinin port atamaları örnek olarak aşağıdaki gibi gösterilmiştir. U5 modülü için capture modülünden gelen filtrelenecek olan son 4 bit “oca_dout” kırmızı renk pikseli, filtreleme için “pixel_in” port girişine atanmıştır. U5 modülünden filtrelenerek “pixel_out” olarak çıkan kırmızı piksel, U5a modülünün piksel girişine (“pixel_in” portuna) atanmıştır. Bu atama için “m_pxl_r” isminde 4 bitlik kırmızı renk piksel sinyali kullanılmıştır.

```
U5 : median --kirmizi 1.medyan
port map
(
    rst         => rst_locked,
    clk         => cam_clk,
    enable      => fil_en(0),
    vsync_in    => oca_vsync_o,
    pixel_in    => oca_dout(11 downto 8),
    pixl_vld    => oca_we(0),
    vsync_out   => m_vsync_or,
    pixel_out   => m_pxl_r,
    output_vld  => m_out_vld_r
);

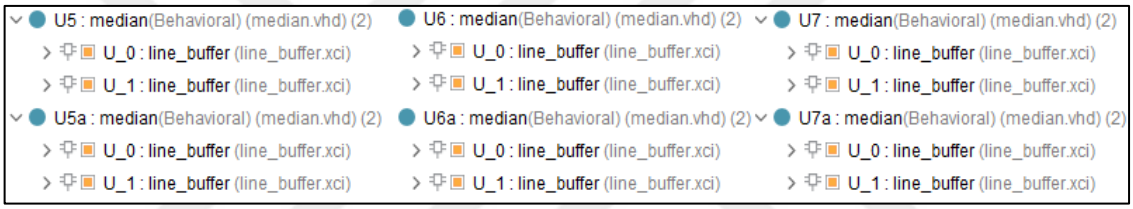
leds(1)     <= fil_en(0); --medyan 1
leds(2)     <= fil_en(1); --medyan 2
```

```
U5a : median --kirmizi 2.medyan
port map
(
    rst         => rst_locked,
    clk         => cam_clk,
    enable      => fil_en(1),
    vsync_in    => m_vsync_or,
    pixel_in    => m_pxl_r,
    pixl_vld    => m_out_vld_r,
    vsync_out   => m_vsync_or0,
    pixel_out   => m_pxl_r0,
    output_vld  => m_out_vld_r0
);
```

Medyan filtreleme modülünde kernel matrisi oluşturma, piksel sayımı işlemi ve piksellerin küçükten büyüğe sıralanması işlemleri için process blokları tanımlanmıştır. 3x3 boyutundaki kernel matrisi, 320x240 boyutundaki görüntü üzerinde kayarak her kaydığında bulunduğu satırı 2 adet satır arabellek bloğuna kaydetmektedir. Kaydedilen 9 adet piksel verisi küçükten büyüğe sıralanmaktadır. Sıralanan piksel verisinde ortadaki (medyan) değeri alınarak merkez piksel değerinin yerine yazılır.

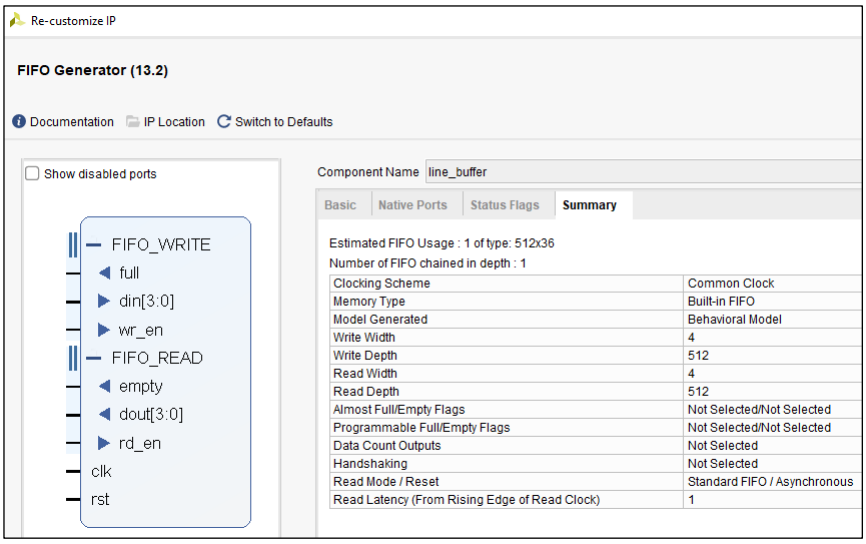
3.12.1. Satır Arabelleği IP Blokları

Bir görüntü satırı arabelleği, tek bir satır içerisinde tüm pikselleri FPGA belleğinde saklayabilmektedir. Satır arabelleğinin boyutu ayarlanabilir. Her medyan modülünün altına Şekil 3.32'de olduğu gibi 2 adet satır arabelleği eklenerek toplam 12 adet satır arabelleği kullanılmıştır.



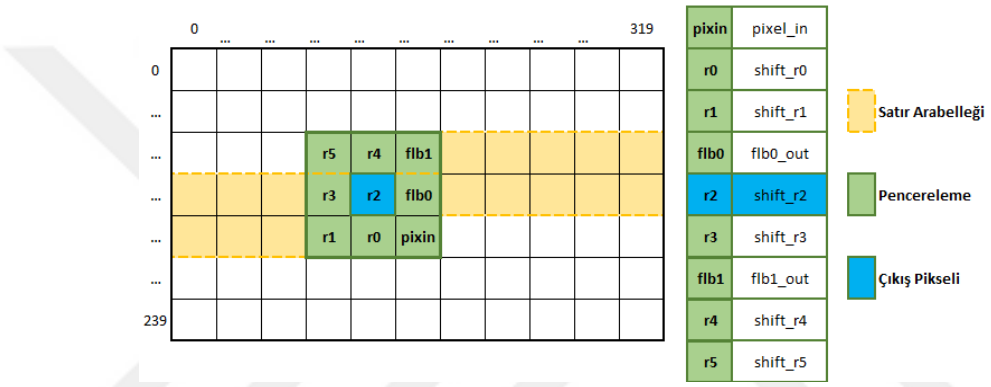
Şekil 3.32: Medyan filtrelemede satır arabelleği tanımlama.

Şekil 3.33'te Vivado programında satır arabelleği ayarlaması gösterilmiştir. Yazma genişliği (write width), 4 bit; yazma derinliği (write depth) 512 bit olarak ayarlanmıştır. Böylece 4 bitten oluşan piksel verisi giriş olarak alınır. Yazma derinliği, FIFO üreticisi IP bloğunun ayarlarında minimum 512'dir. Bir satırda 320 adet piksel olduğu için minimum ayarı seçilmiştir.



Şekil 3.33: Vivado programında satır arabelleği ayarlama.

Şekil 3.34'te gösterilen 3x3 boyutunda kernel matrisi pencereleme şeklinde görüntü üzerinde kayarak 9 adet piksel verisinin registerlara kaydedilmesini sağlar. Kernel matrisine ait her bir piksel elemanı, 4 bit boyutunda olacak şekilde "pixel_in", "shift_r0", "shift_r1", "flb0_out", "shift_r2", "shift_r3", "flb1_out", "shift_r4" ve "shift_r5" şeklinde sinyal atamaları yapılarak isimlendirilmiştir. Kaydedilen pikseller küçükten büyüğe doğru sıralanarak ortanca değer olan piksel değeri, medyan değer olarak çıkışta alınır ve merkezdeki mavi ile gösterilen çıkış pikselinin yerine bu medyan değeri yazılarak konvolüsyon işlemi devam ettirilir. 3x3 boyutunda kernel matris kullanıldığından 2 satır boyunca piksel kaydı için 2 adet satır arabelleği kullanılmıştır.



Şekil 3.34: 320x240 görüntüde satır arabelleğinin uygulama şeması.

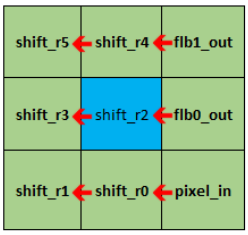
Satır arabelleklerine ait port atamaları aşağıdaki gibidir. IP bloğu, 24 MHz PCLK ile çalışmaktadır. "U0" komponenti olan ilk satır arabelleğinde; capture modülünden gelen 4 bitlik "din" veri girişi, "shift_r0" portuna atanmıştır. 1 bitlik "wr_en" (write enable) portu, satır arabelleğine piksel yazmanın etkinleştirilmesi için kullanılan porttur ve bu portta değişiklik yapılabilmesi için "flb0_wren" (fifo line buffer write enable) sinyal ataması yapılmıştır. "rd_en" (read enable) portu, satır arabelleğinden okumayı etkinleştirmek için kullanılmıştır. 4 bitlik "dout" portu, çıkış pikselidir. "full" ve "empty" portları, satır arabelleği doluysa yazmamak, boşsa okumamak için kullanılan portlardır; ancak programın IP bloğuna ait satır arabelleği yeterince bit alabildiğinden bu portlara ihtiyaç yoktur. Bu sebeple port uçları, açık (open) bırakılmıştır.

```
U_0 : line_buffer
port map
(
    clk     => clk,
    rst     => rst,
    din     => shift_r0,
    wr_en   => flb0_wren,
    rd_en   => flb0_rden,
    dout    => flb0_out,
    full    => open,
    empty   => open
);
```

```
U_1 : line_buffer
port map
(
    clk     => clk,
    rst     => rst,
    din     => shift_r2,
    wr_en   => flb1_wren,
    rd_en   => flb1_rden,
    dout    => flb1_out,
    full    => open,
    empty   => open
);
```

3.12.2. Kernel Matrisi ile Kayan Pencereleme İşlemi

Görüntü üzerindeki piksellerin medyan değerinin bulunması için kernel matrisi ile görüntünün ilk pikselinden başlanarak soldan sağa doğru taranması gerekmektedir. 3x3 boyutunda olan kernel matrisi ile görüntü üzerinde kayan pencereleme işlemi yaparak piksel değerlerinin alınması için process tanımlanmıştır. Kernel matrisini oluşturan 9 adet piksel, satır arabelleğine yazdırılır. Şekil 3.35'te gösterilen kernel matrisinin görüntü üzerinde kayan pencereleme yapması için her bir piksel değeri soldakine atanarak işlem devam eder.



Şekil 3.35: Kernel matrisi pencereleme işlemi.

Yapılan bu işlemin VHDL kodu aşağıdaki gibi tanımlanmıştır. Buna göre geçerli piksel geldiğinde (pixl_vld = '1') veya görüntü üzerinde son satıra gelinmişse (last_line_1= '1') son satır için de kayan pencereleme işlemi başlatılır. Görüntüde piksel verisi alma işlemi, "pixel_in" sinyali ile başlar. Bu piksel değeri, bir sonraki kernel matris elemanı değeri olan "shift_r0" sinyaline atanarak işlem devam eder. Buna paralel olarak aynı anda (aynı saat sinyalinde) "shift_r0" piksel değeri, "shift_r1" piksel değerine atanır.

```
if(pixl_vld = '1' or last_line_1 = '1') then
    shift_r0        <= pixel_in;
    shift_r1        <= shift_r0;
end if;
```

İlk satır arabelleği için okuma emri gelmişse (flb0_rden = '1') "flb0_out" piksel değeri, "shift_r2" değerine atanır. Aynı anda "shift_r2" piksel değeri, "shift_r3" değerine atanmış olur.

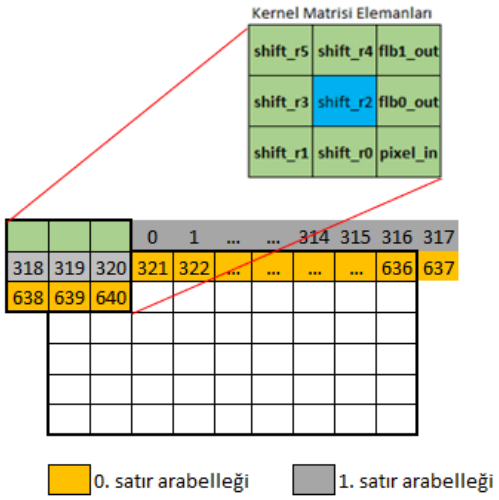
```
if(flb0_rden = '1') then
    shift_r2        <= flb0_out;
    shift_r3        <= shift_r2;
end if;
```

Aynı işlemler 2. satır arabelleği olan "U1: line_buffer" komponenti için tekrarlanır. İkinci satır arabelleği için okuma emri gelmişse (flb1_rden = '1') "flb1_out" piksel değeri, "shift_r4" değerine atanır. Aynı anda "shift_r4" piksel değeri, "shift_r5" değerine atanmış olur.

```
if(flb1_rden = '1') then
    shift_r4        <= flb1_out;
    shift_r5        <= shift_r4;
end if;
```

İlk satır arabelleğine okuma emri; geçerli piksel değeri geliyorsa ve piksel sayımı (pixl_count sinyali) 320'ye eşit ve büyükse verilir. İkinci satır arabelleğine okuma emri, geçerli piksel

değerleri geliyorsa ve piksel sayımı 640'tan büyükse verilir. Şekil 3.36'da arabelleği okuma işlemi sırasında piksel sayımı gösterilmiştir.



Şekil 3.36: 320x240 görüntüde satır arabelleği okuma için piksel sayımı.

3.12.3. Piksel Sayım İşlemi

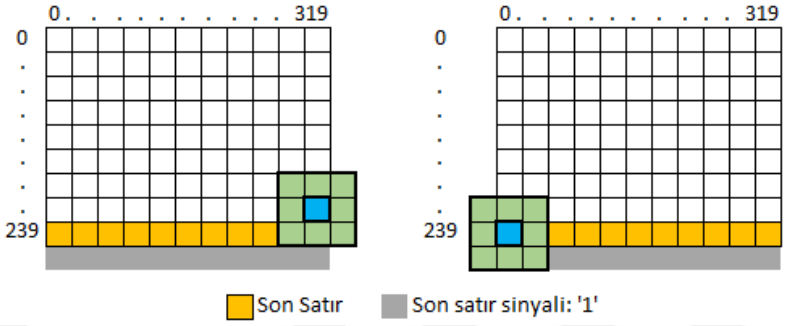
Piksel sayım işlemi için oluşturulan process bloğu; "vsync_in" portunda gecikme oluşturmak ve kernel matrisinin son satıra gelerek pencereleme işlemini sürdürmesini sağlamak için son satırı saydırma amacıyla tasarlanmıştır.

Filtrelemelerin kaskad bağlanarak kullanılması sebebiyle adresleme bloğunda oluşacak olan gecikmeden dolayı görüntü çerçevesini erkenden okuma işlemi başlamaktadır. "vsync_in" portunda gecikme yapılarak bu durum engellenmiş olur. Böylece kamera adresini arttırarak görüntü çerçevesini ekrana basmak yerine kamera adresi sıfıra çekilmektedir.

Aynı process bloğu içerisinde kernel matrisinin son satıra gelerek pencereleme işlemini sürdürmesini sağlamak için piksel sayımı yapılmıştır. Tasarımda kullanılan if bloğunda yataydaki sayaç 320, dikeydeki sayaç 240 piksel tamamlamışsa son satır 320 değerine eşitlenir. Son satır 320 değerine eşit olduğunda aşağıdaki koşul sağlanır ve son satır sayım sinyali (lst_count) sıfırlanana kadar son satırda olunduğunu gösteren son satır (last_line) sinyali '1' değerine çekilmektedir.

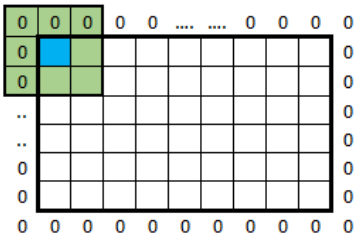
```
if (lst_count > "00000000000") then
    lst_count <= lst_count - '1';
    pxl_count <= (others => '0');
    last_line <= '1';
else
    last_line <= '0';
end if;
```

Şekil 3.37'de gösterilen durumda 239. satır olan ve turuncu ile gösterilen son satıra gelindiğinde satırda bulunan piksellerin de medyan işlemine alınması gerekmektedir. Bu sebeple gri olan kısımda son satır '1' değerine çekilerek pencereleme işleminin devam etmesi sağlanmıştır.



Şekil 3.37: Piksel sayımı son satır pencereleme işlemi.

"conv_ena" (convolution enable – konvolüsyon etkinleştirme) isimli sinyalin '1' olması durumunda kayan pencereleme başlatılmaktadır. Bu sinyalin '1' olması için 320 kez dış çerçevede piksel sayımı yapılır ve piksel sayımı 320'den büyük olduğunda pencereleme işlemi başlatılır. Bu koşulun dışında son satıra gelinmişse bu sinyal değeri '1'e çekilir. Şekil 3.38'de gösterilen görüntü çerçevesinin sol üst köşesinde yer alan mavi piksel karesi, bu işlemin başlangıcı olarak kabul edilir.



Şekil 3.38: Pencereleme işlemi başlangıcı.

3.12.4. Sıralama İşlemi

Kernel matrisinin 9 adet tüm elemanları registerlara kaydedildikten sonra sıralama işlemine alınır. Renk pikselleri, RGB444 formatında olduğu için maksimum alınabilecek renk piksel değeri ikili tabanda "1111" değerine, onluk tabanda 15 değerine karşılık gelmektedir. Kameradan filtreleme işlemine alınan görüntüdeki 0-15 arası tüm piksel değerleri için medyan değerleri alınarak gerçek zamanlı olarak ekranda filtrelenmiş görüntü gösterilmektedir. Örnek olması amacıyla eleman piksel değerleri Şekil 3.39'daki gibi olan kernel matrisi için sıralama işlemi yapılacaktır.

Kernel Matrisi Elemanları

shift_r5	shift_r4	flb1_out
shift_r3	shift_r2	flb0_out
shift_r1	shift_r0	pixel_in

Elemanların Piksel Değerleri

8	7	6
5	4	3
2	1	0

Şekil 3.39: Örnek kernel matrisi elemanları ve piksel değerleri.

Satır arabelleğinden alınan elemanlar Tablo 3.9'daki gibi if bloklarında karşılaştırılarak Toplam 9 clock sinyali sonrası küçükten büyüğe sıralanır ve en son ortadaki değer (s84), medyan değeri olarak elde edilmektedir. s0'dan s8'e kadar toplam 9 kez sıralama işlemi mevcuttur. "s00, s01,…, s08,…, s80,…s88" şeklinde tabloda belirtilen 4 bitlik piksel verileri için 81 adet sinyal VHDL'de tanımlanmıştır.

s0 durumunda, "pixel_in" piksel değeri, direkt olarak sarı ile işaretlenen s08 sinyaline atanmıştır. "conv_ena" (convolution enable - konvolüsyon seçimi) sinyali geçerli ise "v0" (valid-geçerli) değeri '1' durumuna çekilerek sıralama işlemi başlatılır.

```
s08     <= pixel_in;    v0 <= conv_ena;
```

Tüm karşılaştırma işlemleri, paralel bir şekilde aynı anda yapılmaktadır. Örnek karşılaştırma if bloğunun VHDL kodu aşağıdaki gibidir. "shift_r5" ve "shift_r4" sinyalleri karşılaştırılmıştır. "shift_r5" sinyali piksel değeri 8, "shift_r4" sinyali piksel değeri 7 olarak atanmıştır. Bu durumda "shift_r5" sinyali, "shift_r4" sinyalinden büyük olduğu için else bloğu kullanılır. "s00" değeri olarak shift_r4; "s01" değeri olarak shift_r5 sinyali atanır. Tabloda kırmızı renk ile gösterilmiştir.

```
if (shift_r5 < shift_r4) then
    s00 <= shift_r5;
    s01 <= shift_r4;
else
    s00 <= shift_r4;
    s01 <= shift_r5;
```

Her bir clock sinyalinde durumlarda karşılaştırma işlemleri yapılmıştır. flb1_out ve shift_r3 arasında; shift_r2 ve flb0_out arasında; shift_r1 ve shift_r0 arasında yapılmaktadır. s1 durumuna gelindiğinde "s00" değeri, "s10" sinyaline atanır. "s10 = 7" değerini alır. Diğer elemanlar; s11-s12 , s13-s14, s15-s16, s17-s18 karşılaştırılarak küçükten büyüğe sıralanır. s2 durumunda, "s18" değeri, "s28" sinyaline atanır. "s28 = 2" değerini alır. Diğer elemanlar için de s20-s21, s22-s23, s24-s25, s26-s27 karşılaştırılarak küçükten büyüğe sıralanır. Diğer durumlar olan s3, s4, s5, s6, s7 ve s8 durumları için de bu işlemler yapılmaktadır. Sarı ile

işaretlenen kısım tabloda bir önceki durumdaki değeri alan elemanlardır. “pixel_in” değerinin direkt olarak s08’e atanması, s00 değerinin direkt olarak s10 değerine atanması, s18 değerinin direkt olarak s28 değerine atanması bu durumlara örnek verilebilir. En son durum olan s8 durumuna gelindiğinde tüm elemanlar küçükten büyüğe sıralanmıştır. Bu durumda medyan değeri, yeşil renk ile işaretlenen “s84” değeridir.

Tablo 3.9: Örnek kernel matrisi piksel değerleri sıralama işlemi.

Kernel Matrisi		**Durum (State) Sayısı**																	
Eleman İsmi	**Piksel Değeri**	**s0**		**s1**		**s2**		**s3**		**s4**		**s5**		**s6**		**s7**		**s8**	
shift_r5	8	**s00**	**7**	s10	7	s20	5	s30	5	s40	3	s50	3	s60	1	s70	1	s80	0
shift_r4	7	**s01**	**8**	s11	5	s21	7	s31	3	s41	5	s51	1	s61	3	s71	0	s81	1
flb1_out	6	s02	5	s12	8	s22	3	s32	7	s42	1	s52	5	s62	0	s72	3	s82	2
shift_r3	5	s03	6	s13	3	s23	8	s33	1	s43	7	s53	0	s63	5	s73	2	s83	3
shift_r2	4	s04	3	s14	6	s24	1	s34	8	s44	0	s54	7	s64	2	s74	5	**s84**	**4**
flb0_out	3	s05	4	s15	1	s25	6	s35	0	s45	8	s55	2	s65	7	s75	4	s85	5
shift_r1	2	s06	1	s16	4	s26	0	s36	6	s46	2	s56	8	s66	4	s76	7	s86	6
shift_r0	1	s07	2	s17	0	s27	4	s37	2	s47	6	s57	4	s67	8	s77	6	s87	7
pixel_in	0	s08	0	s18	2	s28	2	s38	4	s48	4	s58	6	s68	6	s78	8	s88	8

VHDL kodunda “enable_d” (filtre seçimi anahtarı) aktif durumda ise medyan değeri olan “s84” değeri, “pixel_out” portu ile çıkışa verilmektedir.

```
if (enable_d = '1') then
    pixel_out   <= s84;
```

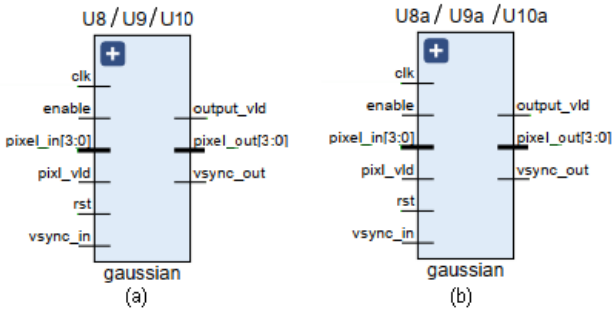
3.13. GAUSS FİLTRELEME MODÜLLERİ

Gauss filtreleme, yumuşatma işlemi ile görüntü bulanıklaştırılarak gürültünün ortadan kaldırılması için kullanılır. 2 boyuta sahip konvolüsyon (kernel matrisi) ile görüntü üzerindeki piksellerin çarpımını sağlayan modüldür. Şekil 3.40’ta bu modülün 1. gauss filtrelemesine ait Vivado programı RTL şeması yer almaktadır. Medyan modüllerinin her bir renk kanalı için ayrı ayrı alınan piksel çıkışları olan 4 bitlik RGB piksel verisi, bu modülün giriş piksel verisi olarak alınır ve filtrelenir. ZedBoard üzerinde kullanıcı anahtarları ile medyan seçimi yapılmasa bile

capture modülünden alınan piksel çıkışları medyan modülüne giriş yapar ve gecikmeli de olsa filtrelenmeden gauss modüllerine aktarılır.

Medyanda olduğu gibi bu filtrelemede de her bir renk için 3 adet modül kullanılmıştır. U8, kırmızı renk; U9, yeşil renk; U10, mavi renk için 1.gauss bloğudur. U8a kırmızı renk; U9a, yeşil renk; U10a, mavi renk için 2.gauss bloğudur. Tüm portları medyan ile aynı şekilde tanımlanmıştır. 1.gauss filtreleme modülü kaskadlanarak aynı VHDL kodundan 2.gauss filtre modülü oluşturulmuştur. Medyan ile aynı şekilde "enable" ucu, üst modül portu olan "fil_ena" portuna aşağıdaki gibi bağlıdır.

```
leds(3)     <= fil_en(2); --gauss 1
leds(4)     <= fil_en(3); --gauss 2
```

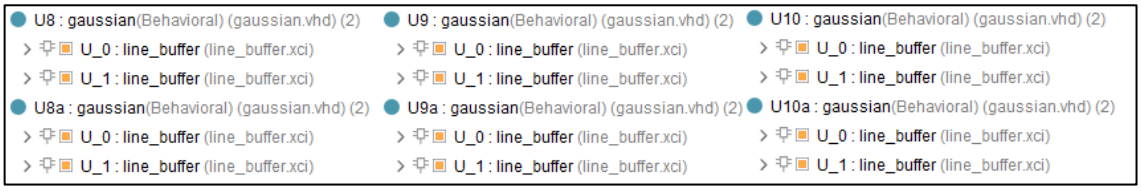


Şekil 3.40: Gauss filtreleme modülleri. (a) 1.gauss filtreleme. (b) 2.gauss filtreleme.

Gauss filtreleme modülünde kernel matrisi oluşturma, piksel sayımı işlemi ve piksellerin konvolüsyon işlemi için process blokları tanımlanmıştır. 3x3 boyutundaki kernel matrisi, 320x240 boyutundaki görüntü üzerinde kayarak her kaydığında bulunduğu satırı 2 adet satır arabellek bloğuna kaydetmektedir. Kaydedilen 9 adet piksel verisi, konvolüsyon matrisi ile konvolüsyon işlemine girmektedir. Çarpım işleminden çıkan piksel değerlerinin toplamı alınarak merkez piksel değerinin yerine yazılır ve konvolüsyon işlemi sürdürülür.

3.13.1. Satır Arabelleği IP Blokları

Her gauss modülünün altına medyan filtrelemede olduğu gibi Şekil 3.41'de 2 adet satır arabelleği eklendiği gösterilmiştir. Toplam 12 adet satır arabelleği kullanılmıştır. Medyan filtrede kullanılan satır arabellekleri ile aynı IP blokları çoğaltılmıştır.



Şekil 3.41: Gauss filtrelemede satır arabelleği tanımlama.

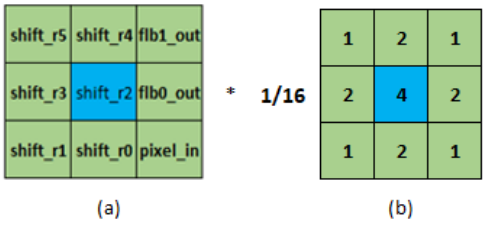
Medyan filtrelemede olduğu gibi 3x3 boyutunda kernel matrisi pencereleme şeklinde görüntü üzerinde kayarak 9 adet piksel verisinin çerçeve arabelleğine kaydedilmesini sağlar. Kernel matrisine ait her bir piksel elemanı, 4 bit boyutunda olacak şekilde "pixel_in", "shift_r0", "shift_r1", "flb0_out", "shift_r2", "shift_r3", "flb1_out", "shift_r4" ve "shift_r5" şeklinde sinyal atamaları yapılarak isimlendirilmiştir. 3x3 boyutunda kernel matris kullanıldığından 2 satır boyunca piksel kaydı için 2 adet satır arabelleği kullanılmıştır.

3.13.2. Konvolüsyon Matrisi ile İşlem

Görüntü çerçevesinden kernel matrisi ile pencerelenerek alınan piksel elemanları, aşağıda yer alan konvolüsyon matrisi ile çarpılır.

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Çarpılan değerler toplanarak gauss değerleri elde edilir. Şekil 3.42'de konvolüsyon işleminin uygulaması gösterilmiştir.



Şekil 3.42: Konvolüsyon uygulaması. (a) Kernel matrisi. (b) Konvolüsyon matrisi.

Kernel matrisi ile konvolüsyon matrisinin konvolüsyon işlemi denklem 3.1'de gösterilmiştir.

Gauss piksel değeri = (3.1)

$$\frac{1}{16}\,\text{x}\,[\,(\text{shift_r5 x 1} + \text{shift_r4 x 2} + \text{flb1_out x 1}) +$$

$$(\text{shift_r3 x 2} + \text{shift_r2 x 4} + \text{flb0_out x 2}) +$$

$$(\text{shift_r1 x 1} + \text{shift_r0 x 2} + \text{pixel_in x 1})\,]$$

Çarpma ve toplama işlemlerinin sonucunun yazılması için VHDL kodunda aşağıdaki gibi "addition" isminde 10 bitlik bir sinyal tanımlanmıştır.

```
signal addition     : std_logic_vector(9 downto 0);
```

Vivado programında çarpma işlemi için shift register kullanılmıştır. Her kernel matris elemanı, konvolüsyon matrisi elemanları ile aşağıdaki gibi çarpılmıştır. Örneğin; "1" ile çarpılan elemanlar, "00" & eleman (001) şeklinde yazılırken; "2" ile çarpılan elemanlar '0' & "eleman" & '0' (010) şeklinde yazılarak çarpım işlemine alınmıştır. En son tüm değerler toplanarak "addition" sinyaline atanmıştır. Sinyal 10 bit olarak tanımlandığından en son "shift_r5" elemanının başına 6 adet sıfır eklenmiştir.

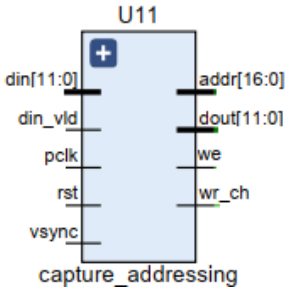
```
addition    <= ("000000" & shift_r5) + ('0' & shift_r4 & '0') + ("00" & flb1_out)
+ ('0' & shift_r3 & '0') + (shift_r2 & "00") + ('0' & flb0_out & '0')
+ ("00" & shift_r1) + ('0' & shift_r0 & '0') + ("00" & pixel_in);
```

Toplama işlemlerinden sonra matris dışında yer alan 1/16 işlemi için, 16 sayısına bölme işlemi 2^N olarak ifade edilirse, N=4 olarak alınır ve 4.bitten itibaren 4 bit alınırsa bu işlem gerçekleştirilir. Bu sebeple aşağıdaki gibi 10 bitlik addition sinyalinin 4. biti ile 7.biti arası alınmıştır.

```
pixel_out   <= addition(7 downto 4);
```

3.14. ADRESLEME MODÜLÜ

Capture adresleme modülü, kameradan gelen görüntü piksellerini adresleyerek çerçeve arabelleklerinin yazma adresine aktaran modüldür. Şekil 3.43'te bu modüle ait Vivado programı RTL şeması yer almaktadır.



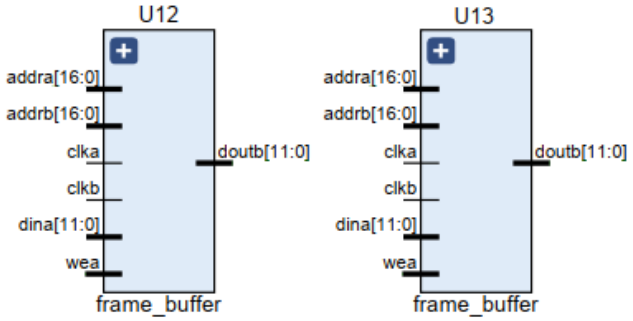
Şekil 3.43: Capture adresleme modülü.

Filtreleme modüllerinin çıkışından gelen 12 bitlik filtrelenmiş piksel verisi, bu modülün 12 bitlik "din" portuna giriş portu olarak tanımlanmıştır. 2 adet frame buffer mevcuttur. İkisine de capture adresleme aynı adresi göndermektedir. "vsync" portu 0 durumundayken geçerli piksel verileri gelirse (din_vld = '1') adres bir arttırılarak "we" yazma emri gönderilir. Çıkış pikselleri çerçeve arabelleklerine giriş pikseli olarak verilir.

```
if (din_vld = '1') then
    address     <= address + '1';
    dout        <= din;
    we          <= '1';
```

3.15. ÇERÇEVE ARABELLEĞİ IP BLOKLARI

Çerçeve arabelleği (Frame Buffer) IP blokları, bir görüntü çerçevesini saklayan IP bloklardır. Şekil 3.44'te bu modüllere ait Vivado programı RTL şeması yer almaktadır.



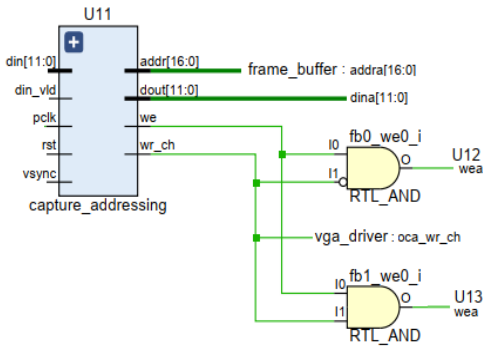
Şekil 3.44: Çerçeve arabelleği IP blokları.

Görüntü sensörü maksimum 30 fps sahiptir. 30 fps ile çalıştığı için bir görüntü çerçevesini alma hızı, 60 fps ile çalışan VGA'ya göre iki kat daha yavaştır. Bu sebeple iki adet çerçeve arabelleği bloğu kullanılmıştır. İki bloğa da aynı saat sinyali, piksel verisi ve adres gitmektedir. Okuma, yazma portları bulunmaktadır. Adresleme modülü tarafından çerçeve arabelleği bloklarına yazma adresi "addra" gönderilir. VGA tarafından çerçeve arabelleği bloklarına okuma adresi "addrb" gönderilerek bir görüntü çerçevesini oluşturan piksel verileri, adres bir bir arttırılarak "doutb" portundan VGA sürücü modülüne gönderilerek monitöre basılmaktadır.

Yazma saat sinyali "clka" 24 MHz ile çalışırken, okuma saat sinyalinin "clkb" VGA ile uyumlu olması gerektiğinden 25.175 MHz olarak alınmıştır. "wea" portu, yazma etkinleştirmesi için giriş portu olarak kullanılmıştır. "wea", '1' değerini aldığında çerçeve arabelleğine yazma işlemi yapılır. "wea" portu bağlantısı için capture adresleme modülü kullanılmıştır.

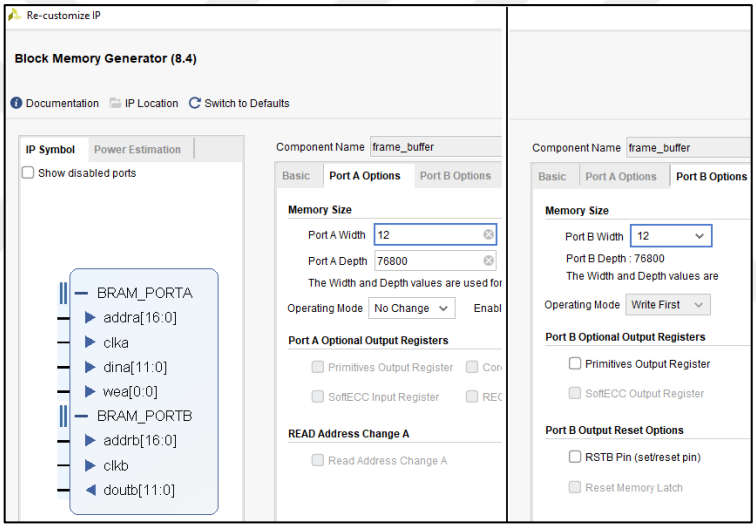
Şekil 3.45'te capture modülü ve çerçeve arabelleği blokları arasındaki bağlantı gösterilmiştir. "wr_ch" sinyali, 2 adet çerçeve arabelleğinden birine değili alınarak, diğerine değil işlemi yapılmadan direkt olarak gitmektedir. Kameradan gelen bir kare bittiğinde we sinyali sırayla U12 çerçeve arabelleği bloğuna, sonra U13 çerçeve arabelleği bloğuna dönüşümlü olarak yazılmaktadır.

Kamera, U12 çerçeve arabelleği bloğuna yazma işlemi yaparken; VGA, U13 çerçeve arabelleği bloğundan okuma işlemi yapmaktadır. Değişme işlemleri için VGA'daki boş zaman aralığı ve vsync portu kullanılır.



Şekil 3.45: Capture adresleme; VGA ve çerçeve arabelleği blokları bağlantısı.

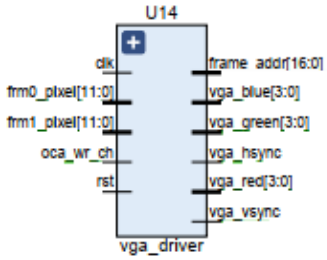
Şekil 3.46'da çerçeve arabelleği IP blokları ayarı yapılmıştır. Bu ayara göre okuma ve yazma portları için piksel verisi 12 bit olduğundan genişlik 12 bit, 320x240 görüntü boyutu olduğundan derinlik 76800 piksel olarak alınmıştır.



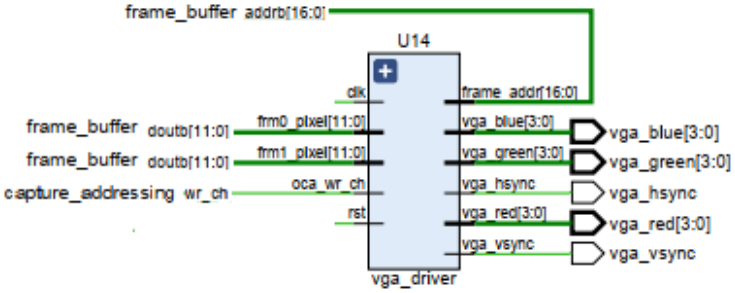
Şekil 3.46: Vivado programında çerçeve arabelleği ayarlama.

3.16. VGA SÜRÜCÜSÜ MODÜLÜ

VGA sürücüsü modülü, çerçeve arabelleğine okuma adresi gönderip pikselleri alarak monitöre basmayı sağlayan modüldür. Şekil 3.47'de bu modüle ait Vivado programı RTL şeması yer almaktadır. Şekil 3.48'de "frame_addr" adresleme portu ile çerçeve arabelleğine okuma adresi gönderilir. Hangi çerçeve arabelleğinden piksel alınacağına "actv_wr_ch" sinyali karar vermektedir. Böylece U12 veya U13 çerçeve arabelleğinin 12 bitlik "doutb" portundan alınan piksel verisi, VGA sürücü modülünün "frm0_pixel" veya "frm1_pixel" portuna gelmektedir. Çıkış portları olarak VGA'ya ait RGB kanalları, vsync ve hsync portları tanımlanmıştır.



Şekil 3.47: VGA sürücüsü modülü.



Şekil 3.48: VGA; çerçeve arabelleği blokları ve capture adresleme modülü bağlantısı.

VGA bölgeleri, zaman sabitlerine göre ayarlanmaktadır. VGA architecture bloğunda zaman sabitleri aşağıdaki gibi tanımlanmıştır. “hRez” ve “vRez”, sırasıyla yatayda aktif görüntü piksel sayısını ve dikeyde aktif satır sayısını belirtmektedir. Görüntü boyutu 320x240 olduğundan bu değerler sırasıyla 320 ve 240 olarak alınmıştır. “hStartSync” ve “vStartSync”, yatayda ve dikeyde görüntünün ön sundurma zamanının başlangıcını temsil etmektedir. “hEndSync” ve “vEndSync”, yatayda ve dikeyde görüntünün bittiği anın zaman sabitidir. Tüm VGA ekran boyutu, “hMaxCount” ve “vMaxCount” ile belirlenmiş olup 800x525 olarak alınmıştır.

```
constant hRez           : natural  := 320;
constant hStartSync     : natural  := 640+16;
constant hEndSync       : natural  := 640+16+96;
constant hMaxCount      : natural  := 800;
constant vRez           : natural  := 240;
constant vStartSync     : natural  := 480+10;
constant vEndSync       : natural  := 480+10+2;
constant vMaxCount      : natural  := 480+10+2+33;
```

Tablo 3.10’da tanımlanan sabitlerin piksel sayısı gösterimine yer verilmiştir.

Tablo 3.10: VGA zaman sabitleri piksel sayıları.

TARAMA ÇİZGİSİ ZAMAN ALANI	YATAY BÖLGEDE		DİKEY BÖLGEDE	
	VHDL Sabit Değer İsmi	**Yatay Piksel Sayısı**	**VHDL Sabit Değer İsmi**	**Dikey Satır Sayısı**
Aktif Alan	hRez	320	vRez	240
Boşluk Zamanı: Front Porch	hStartSync	640+16= 656	vStartSync	480+10 = 490
Boşluk Zamanı: Sync	hEndSync	640+16+96= 752	vEndSync	480+10+2 = 492
Boşluk Zamanı: Back Porch	-	48	-	33
Tüm Ekran	hMaxCount	800	vMaxCount	480+10+2+33=525

Capture modülünde bahsedildiği üzere, piksel sayısı 0 ile başlanarak görüntü yatayda 159. pikselde gösterilmektedir. Böylece 320 adet piksel yatayda aktif alanı başlatılır. Dikeyde ise 0 ile başlanarak 119. satırda gösterilmektedir. 240 adet piksel aktif alan olarak gösterilir. Bu başlangıç noktaları için aşağıdaki gibi sabitler tanımlanmıştır.

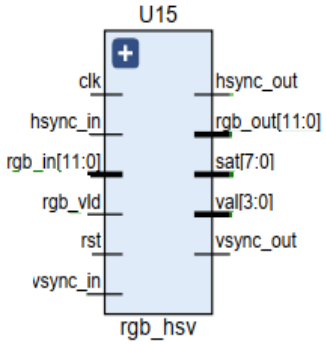
```
constant im_start_x     : natural   := 159;
constant im_start_y     : natural   := 119;
```

Yatay 640 piksel ve dikey 480 satır sayımı için iki adet sayaç sinyali tanımlanır. Bu sayaçlar, yukarıda bulunan sabitler tarafından ayarlanarak 320 piksele 240 piksel görüntü ekrana basılmış olur. Bu koşulların altında "actv_wr_ch" isimli sinyal koşulu aşağıdaki gibi tanımlanmıştır. Bu sinyal, "oca_wr_ch" portuna bağlıdır. "0" olduğunda U13 çerçeve arabelleğinden okunan piksel değerleri; "1" olduğunda ise U12 çerçeve arabelleğinden okunan piksel değerleri ekrana basılır. 12 bitlik piksel verisinin; ilk üç biti mavi renk olan "vga_blue" portu; sonraki üç biti yeşil renk olan "vga_green" portunu; en son üç biti kırmızı renk olan "vga_red" portunu temsil etmektedir. Bu atamalardan sonra okuma adresi bir bir arttırılarak gerekli koşullar sağlanıyorsa piksel basımına devam edilmektedir.

```
if (actv_wr_ch = '0') then
    vga_red     <= frm1_pixel(11 downto 8);
    vga_green   <= frm1_pixel( 7 downto 4);
    vga_blue    <= frm1_pixel( 3 downto 0);
else
    vga_red     <= frm0_pixel(11 downto 8);
    vga_green   <= frm0_pixel( 7 downto 4);
    vga_blue    <= frm0_pixel( 3 downto 0);
end if;
address     <= address + 1;
```

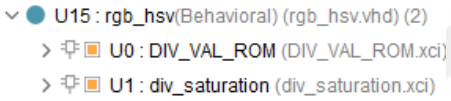
3.17. RGB - HSV MODÜLÜ

HSV renk uzayı verileri, eksik komponent hatası tespiti için lehimli alanların parlak yüzeye sahip olması sebebiyle kolaylık sağlamaktadır. VGA sürücüsünden alınan RGB piksel verileri kullanılarak RGB-HSV modülü ile PCB üzerinde bulunan lehimli alanların tespiti sağlanmıştır. Şekil 3.49'da bu modüle ait Vivado programı RTL şeması yer almaktadır. VGA sürücüsü çıkışında yer alan 12 bitlik RGB444 piksel verisi, bu modüle giriş olarak "rgb_in [11:0]" verilmiştir. Böylece piksel verisi alınarak renk özü (H), doygunluk (S) ve parlaklık (V) değerleri bulunmuştur. H, 0°-360° arasında değişen açısal değerdedir. S ve V değerleri, 0-100 arasında değişen değerdedir. RGB444 formatında olduğu için her bitin onluk tabanda alabileceği maksimum 15 değeridir. Bu sebeple H, S ve V değerleri, 16 bit ile oranlanmıştır.



Şekil 3.49: RGB - HSV modülü.

HSV modülü altında; hue değeri hesaplamada yer alan bölme işlemini tutması için "Block Memory Generator" IP bloğuna ait ROM (Read Only Memory) ve saturasyonda yer alan bölme işlemi için "Divider Generator" IP bloğu kullanılmıştır.



3.17.1. Maksimum, Minimum Renk Değerleri ve Farkı için İşlemler

HSV değerlerinin bulunabilmesi için RGB piksel verisi içerisinde maksimum ve minimum olan renk değerinin bulunması gerekmektedir. Bu sebeple renk değerleri arasında karşılaştırma yapılır. Aşağıda örnek karşılaştırma için VHDL kodu gösterilmiştir. 1.if bloğunda, 11.bit ile 8.bit arasında yer alan kırmızı renk ve 7.bit ile 4.bit arasında yer alan yeşil renk değeri karşılaştırılmıştır. Eğer kırmızı rengin bit değeri yüksekse, 2. if karşılaştırma bloğunda kırmızı renk ve 3.bit ile 0.bit arasında yer alan mavi renk karşılaştırılır.

```
if (rgb_in(11 downto 8) > rgb_in(7 downto 4)) then -- if (R1>G1)
    if (rgb_in(11 downto 8) > rgb_in(3 downto 0)) then --if(R1>B1)
        cmax    <= rgb_in(11 downto 8); --cmax=R1
    else
        cmax    <= rgb_in(3 downto 0); --cmax=B1
    end if;
```

Tüm bu karşılaştırmalar sonucunda maksimum renk "Cmax", minimum renk "Cmin" olarak belirlenir. Bu değerlerin farkı, delta olarak gösterilir. VHDL'de bu değer, "diff" sinyali olarak tanımlanmıştır.

```
diff            <= cmax - cmin;
```

3.17.2. Renk Özü (H) Değeri için İşlemler

Renk özü değerinin hesaplanması denklem 3.2'de gösterilmiştir.

$$H = \begin{cases} 0° & , \ \Delta = 0 \\ 60°x\left(\frac{G'-B'}{\Delta}\ mod6\right) & , \ C_{max} = R' \\ 60°x\left(\frac{B'-R'}{\Delta} + 2\right) & , \ C_{max} = G' \\ 60°x\left(\frac{R'-G'}{\Delta} + 4\right) & , \ C_{max} = B' \end{cases} \quad (3.2)$$

Maksimum renk değeri kırmızı renk ise, "G-B" değerinin pozitif çıkması için aşağıda verilen örnek VHDL kodundaki gibi fark hesabı yapılarak "left" sinyaline atanmıştır.

```
if (cmax_d0 = rgb_in_d1(11 downto 8)) then --cmax_d0= R1'
    if (rgb_in_d1(7 downto 4) >= rgb_in_d1(3 downto 0)) then --G2 >= B2
        left    <= rgb_in_d1(7 downto 4) - rgb_in_d1(3 downto 0); --hue: G2-B2
    else --B > G
        left    <= rgb_in_d1(3 downto 0) - rgb_in_d1(7 downto 4); --hue: B2-G2
    end if;
```

16 bit için olmak üzere (60 / Δ) işlemleri, Şekil 3.50'de gösterilen ROM IP bloğuna kaydedilmiştir.



Şekil 3.50: ROM IP Bloğu.

ROM IP bloğunun "douta" çıkışından alınan 16 bitlik "div_val" değeri ile 4 bitlik "left" sinyali çarpılarak 20 bitlik "mult" sinyali elde edilir.

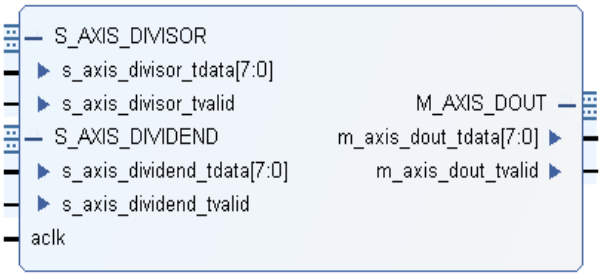
```
mult            <= left * div_val;
```

Renk özü (H) değerinde maksimum çıkan kırmızı renk için örnek VHDL kodu aşağıdaki gibidir. Farkın negatif çıkmasına bağlı olarak 360° değerinden "mult" sinyali çıkartılır.

```
if (rgb_in_d3(7 downto 4) >= rgb_in_d3(3 downto 0)) then --G3 >= B3
    hue     <= mult(18 downto 10);
else --G3<B3
    hue     <= std_logic_vector(to_unsigned(360, hue'length)) - mult(18 downto 10);
end if;
```

3.17.3. Renk Doygunluğu (S) ve Renk Parlaklığı (V) Değeri için İşlemler

Renk doygunluğu (S) değerinin hesaplanması için bölücü IP bloğu kullanılmıştır. Şekil 3.50'de bu IP bloğuna ait Vivado programı RTL şeması yer almaktadır.



Şekil 3.51: Bölücü IP bloğu.

Bölünen (dividend) girişine, fark değeri olan diff (Δ) sinyali; bölen (divisor) girişine ise Cmax sinyali atanmıştır. Çıkışta ise saturasyon değeri elde edilmiştir. Saturasyon değerinin hesaplanması denklem 3.3'te gösterilmiştir.

$$S = \begin{cases} 0 & , \ C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , \ C_{max} \neq 0 \end{cases} \tag{3.3}$$

Renk parlaklığı değeri hesaplaması denklem 3.4'te gösterilmiştir. Bu girişe direkt olarak Cmax sinyali atanmıştır.

$$\mathrm{V} = C_{max} \tag{3.4}$$

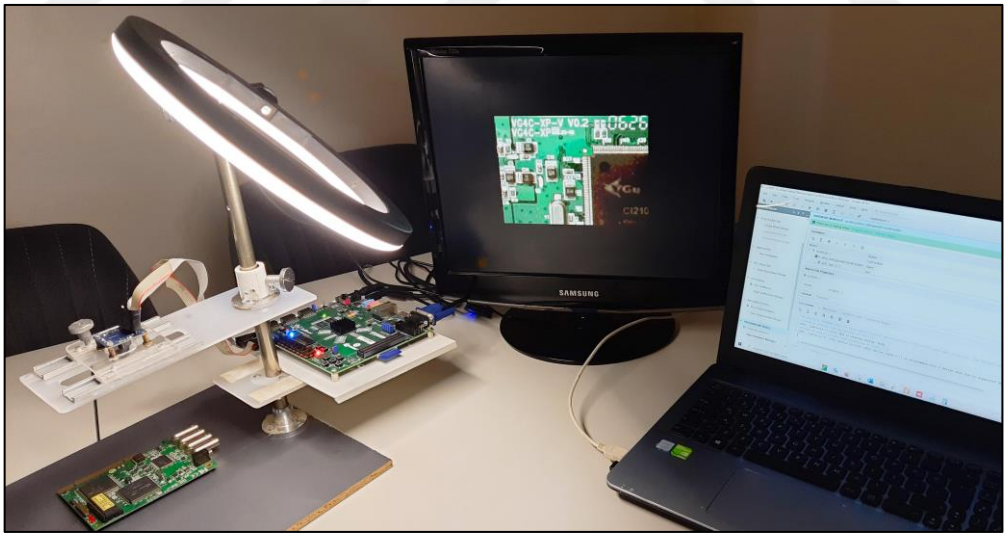
Eksik komponentlerin tespiti için lehimli alanların monitörde kırmızı renk ile gösterilmesi sağlanmıştır. Bunun için parlaklık seviyesi sayesinde lehim rengini HSV renk uzayında temsil eden gri tonlarına yakın olan renk aralığı belirlenerek test edilmiştir. Lehim rengine yakın olan kısımlar için RGB renk uzayının kırmızı kanalı verilmiştir. "mc_en" anahtarı aktifse ve S ve V değerleri lehim renk aralığını sağlıyorsa kırmızı kanala 15 değerinin atanması sağlanmıştır. Bu aralığın dışında ise RGB piksel değerleri direkt olarak monitörde gösterilir.

4. BULGULAR

4.1. DONANIMSAL SİSTEM ÇALIŞMASI

ZedBoard FPGA ile OV7670 görüntü sensöründen gerçek zamanlı olarak RGB görüntüyü alan, medyan ve gauss filtreleme işlemini gerçekleştirerek monitör üzerinde kullanıcının isteğine bağlı olarak filtreli, eksik komponente sahip kısımları veya orijinal görüntüyü gösteren tamamen donanım tabanlı olan sistem gerçekleştirilmiştir.

Şekil 4.1'de donanımsal sistemin görüntüsü bulunmaktadır. ZedBoard FPGA; bilgisayar, monitör ve OV7670 görüntü sensörü ile bağlıdır. Bilgisayar ortamında Vivado programı kullanılarak yazılan VHDL kodları, ZedBoard FPGA içerisine gömülür. OV7670 görüntü sensöründen alınan 320x240 boyutundaki renkli görüntü, VGA aracılığı ile monitöre yansıtılır. ZedBoard kartı üzerinde bulunan anahtarlar kullanılarak filtreleme ve eksik komponente sahip olan kısımlar anlık olarak gösterilmektedir.



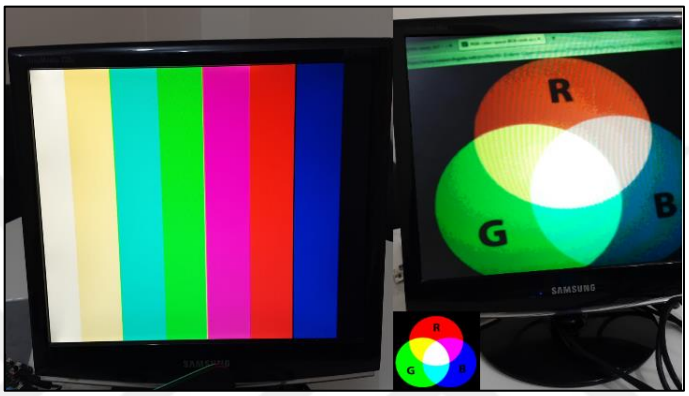
Şekil 4.1: Donanımsal Sistem Görüntüsü.

Medyan, gauss filtrelemeleri için filtreleme modüllerine ve eksik komponent lehimli alan kısımları için HSV modülüne, Vivado programında testbench kodu yazılmıştır. Simülasyon ile test edilen VHDL kodları, gerçek zamanlı sistemde denenerek başarı ile medyan, gauss filtreleme ve eksik komponent için lehimli alanların tespit işlemlerini gerçekleştirmektedir.

4.2. GÖRÜNTÜ ALIMI DEĞERLENDİRMESİ

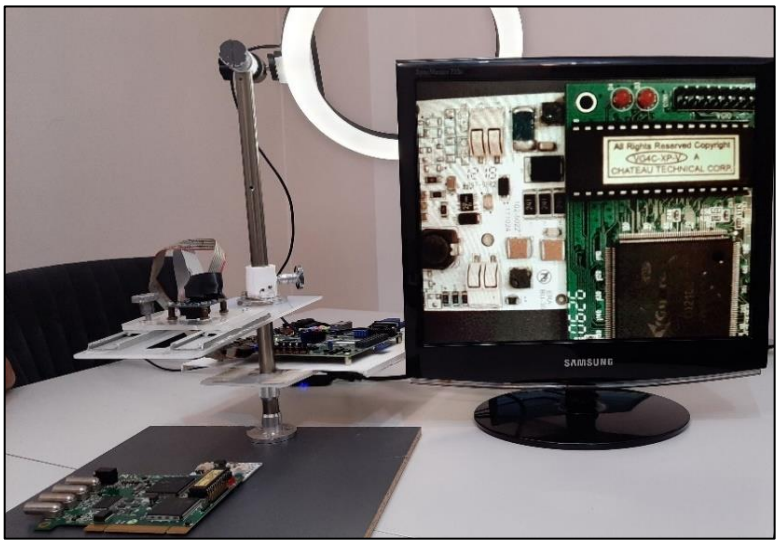
4.2.1. Görüntü Alımı Testi ve Monitör Çıktısı

OV7670 görüntü sensöründen gerçek zamanlı olarak 30 fps hızında alınan 640x480 boyutundaki görüntü, monitöre aktarılarak görüntü alımı sağlanmıştır. Kameranın test edilmesi için Şekil 4.2'de sol tarafta, kamera register ayarlamalarında ifade edilen "8-bar color bar" ayarı ile test modeli; sağ tarafta ise örnek RGB görüntü için renklerin monitörde gösterimi yer almaktadır.



Şekil 4.2: Monitörden 640x480 boyutunda RGB görüntü alımı.

Şekil 4.3'te, donanımsal sistemde gerçek zamanlı olarak OV7670 görüntü sensöründen alınan 640x480 boyutunda, RGB renk formatında örnek PCB görüntülerinin monitörde gösterimi yer almaktadır. Şekil 4.4'te ise aynı görüntünün ortalanması ile elde edilen 320x240 boyutunda görüntüsü mevcuttur.

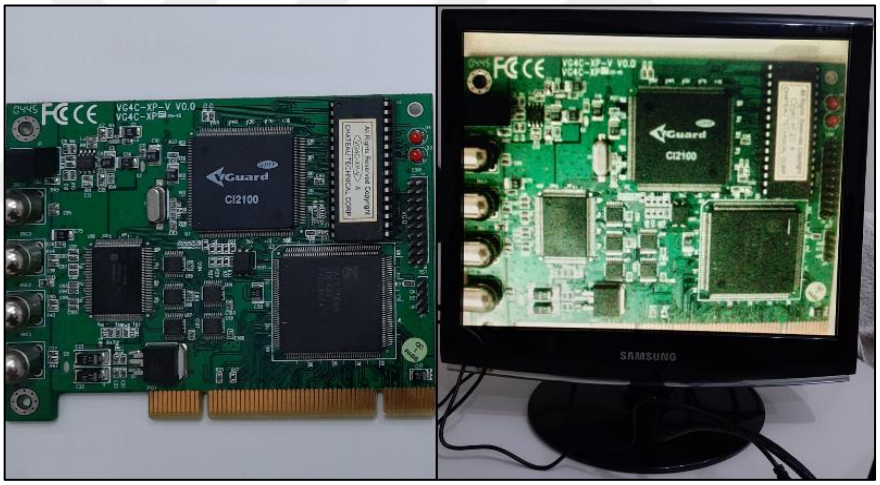


Şekil 4.3: 640x480 boyutunda RGB görüntünün monitörde gösterimi.

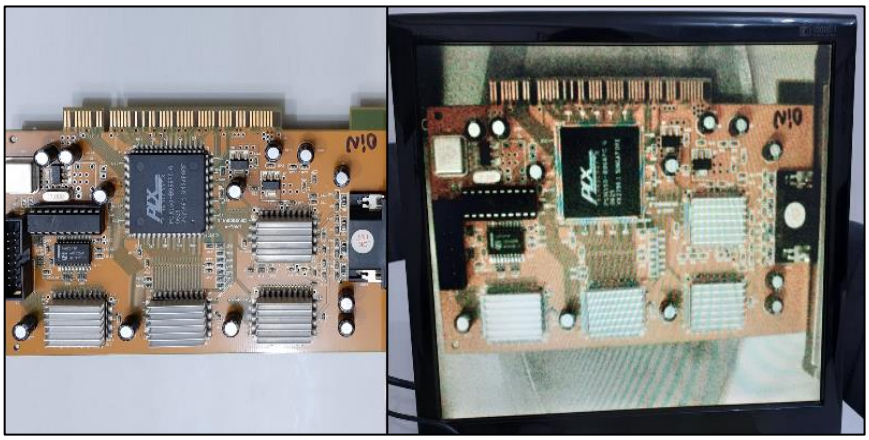


Şekil 4.4: 320x240 boyutunda RGB görüntünün monitörde gösterimi.

Şekil 4.5'te ve Şekil 4.6'da sol tarafta orijinal örnek PCB'lerin görüntüsü; sağ tarafta bu PCB'lerin gerçek zamanlı olarak OV7670 görüntü sensöründen alınan görüntülerinin RGB formatta ve 640x480 boyutunda monitörde yakından gösterimleri yer almaktadır.



Şekil 4.5: 640x480, RGB formatta örnek PCB görüntüsü monitör gösterimi.



Şekil 4.6: 640x480, RGB formatta örnek PCB görüntüsü monitör gösterimi.

4.2.2. Görüntü Alımı Çift Çerçeve Arabelleği (Double Frame Buffer) Sonucu

Görüntü sensörünün bir görüntü çerçevesini alma hızı, veri sayfasında belirtildiği üzere maksimum 30 fps hıza karşılık gelmektedir. 30 fps ile çalışan VGA mevcut değildir ve 60 fps ile çalışan VGA monitörünün görüntüyü gösterme hızına göre görüntü sensörü iki kat daha yavaştır. Bu durumda piksel verisi okumanın hızlı, yazmanın yavaş olduğu söylenebilir. Bu sorunu çözmek için çerçeve arabelleği (frame buffer) IP bloğu, bir yerine iki adet kullanılarak okunan görüntü iki kez yazdırılabilir.

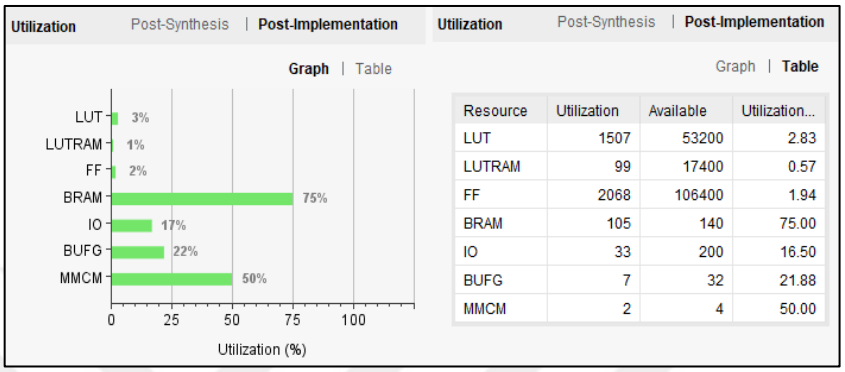
Görüntü sensörü hareketi sırasında, nesnelerin kenarları hizalanamadığı için yırtık bir görünüm oluşturabilir. Şekil 4.7'de tek çerçeve arabelleği kullanımında görüntü sensörü hızla hareket ettirilmiş ve görüntüde yırtılma (screen tearing) oluşup oluşmadığı gözlemlenmiştir. Bu durumda tek çerçeve arabelleği kullanılmış, 30 fps hız ile alınan 640x480 boyutunda görüntüde ekranın tam ortasında yırtılma oluşmaktadır. Ekran yırtılması, görüntü sensöründen VGA monitöre gelen görüntü akışında ekran yenileme hızının sensör veri aktarımının senkron hızda olmamasından kaynaklanmaktadır. Bu problemin giderilmesi için çift arabellek kullanımına karar verilmiştir.



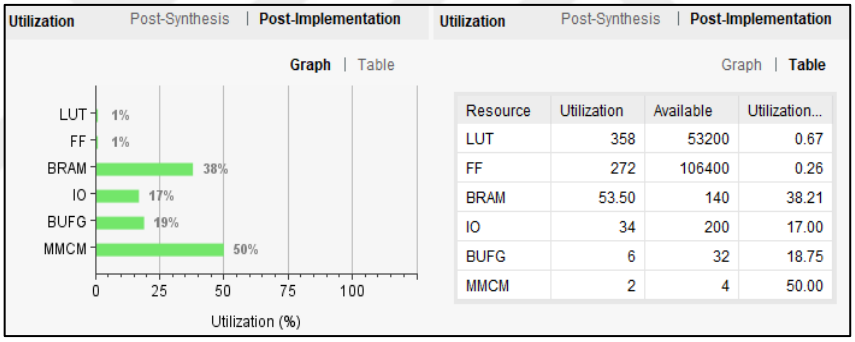
Şekil 4.7: 640x480 tek çerçeve arabelleğe sahip görüntüde yırtılma (tearing) oluşumu.

Tek çerçeve arabelleği kullanıldığında ZedBoard FPGA kartı için kaynak kullanımı sonucu Şekil 4.8'de gösterilmiştir. Bu donanımsal sisteme, birden fazla modül daha ekleneceği için ZedBoard FPGA kartının kaynak kullanımı kısıtlı kalmıştır. Bu sebeple görüntü boyutu yarı yarıya küçültülerek kaynak kullanımı azaltılmıştır. Görüntü boyutu, 640x480 yerine 320x240 boyutu kullanılarak görüntü ortalanmıştır. Tek çerçeve arabelleği yerine çift çerçeve arabelleği

kullanılmıştır. Küçültülen ve çift çerçeve arabelleği kullanılarak arabelleği arttırılan sistemin kaynak kullanımı Şekil 4.9'da sunulmuştur. Buna göre çift çerçeve arabelleği kullanılmasına rağmen görüntü boyutu küçültüldüğü için kaynak kullanımlarının azaldığı gözlemlenmiştir. ZedBoard için gösterilen kaynak limiti "Available" yazan kısmı, "Utilization" yazan kısımda uygulama sonucu kullanım miktarı, grafikte ise kullanım yüzdeleri yer almaktadır.



Şekil 4.8: 640x480 görüntü boyutu, tek çerçeve arabelleği kaynak kullanımı sonucu.



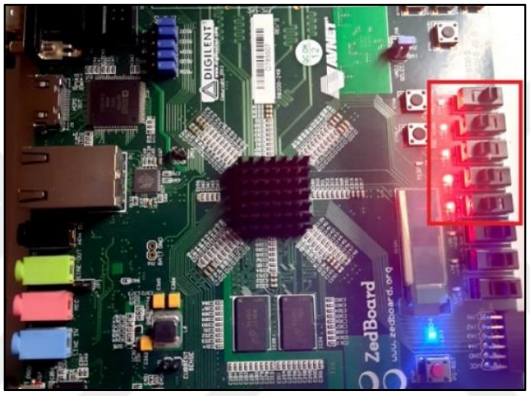
Şekil 4.9: 320x240 görüntü boyutu, çift çerçeve arabelleği kaynak kullanımı sonucu.

4.3. FİLTRELEME SONUÇLARI

Görüntü alımı sonucunda gözlemlenen gürültülerden kaynaklı olarak görüntü üzerinde filtreleme işleminin yapılması gerekmektedir. Görüntü üzerindeki gürültü boyutu fazla olduğu için filtrelemede kullanılan küçük boyutlu pencereleme matrisleri ile gürültü tam olarak kaldırılamaz. Bu sebeple daha büyük boyutta pencereleme matrisi kullanmak veya görüntüyü birkaç kez aynı filtreme işleminden geçirmek gerekmektedir. Hesaplama zamanının uzun olmaması için görüntüyü birkaç kez aynı filtreleme işleminden geçirme yöntemi tercih edilmiştir.

Görüntü verisi iletimi sırasında oluşan hatalardan kaynaklı meydana gelen görüntü sinyalindeki ani değişikliklerle tuz-biber gürültüsü meydana gelmektedir. Görüntünün piksel değerleri,

RGB444 formatında maksimum 15 veya minimum 0'a doğru yakın değer aldığında medyan gürültü çeşidi gözlemlenmektedir. Bu sebeple medyan filtreleme uygulanmıştır. 3 katmanlı RGB renk modelinde renkli görüntü sensöründen elde edilen görüntüde mavi kanaldaki sinyal seviyesinin diğer iki kanala göre fazla durumdaysa mavi kanalda daha fazla gürültü oluşmasına sebep olur. Bu sebeple de gauss filtreleme uygulanmıştır. Şekil 4.10'da filtreleme işlemleri için ZedBoard üzerinde yer alan ve kırmızı ile işaretlenen kullanıcı anahtarlarından dördü kullanılmıştır. İlk LED register ayarlamalarının konfigürasyonu için, 2. ve 3. anahtara bağlı LED'ler medyan filtrelemeler, 3. ve 4. anahtara bağlı LED'ler gauss filtrelemeler için kullanılmaktadır.

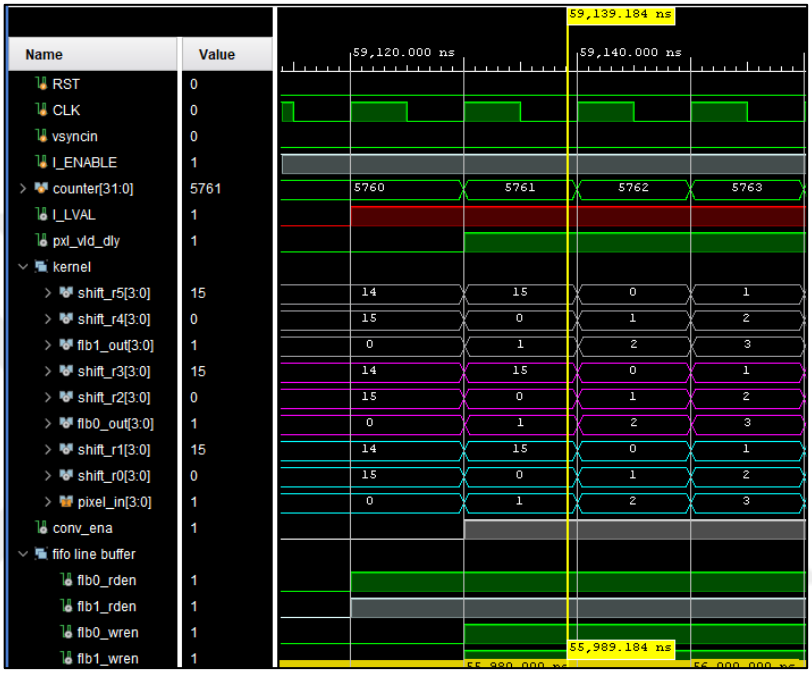


Şekil 4.10: Filtreleme işlemleri için kullanıcı anahtarları ve LED'lerin gösterimi.

Görüntü üzerindeki gürültüleri kaldırmak için uygulanan medyan ve gauss filtreleme modülleri, önce simülasyon ile denenmiş ardından mevcut görüntü alma sistemine eklenmiştir. Simülasyon için testbench kodunda 10 ns periyoda sahip saat ("CLK"), reset ("RST"), 76.800 pikseli sayan sayaç ("counter"), "vsyncin" portları tanımlanmıştır. Kameradan gelen 8 bitlik veriyi, "I_PIXEL" sinyali; satır takibi için "I_LVAL" (line valid) sinyali; "O_PIXEL" ile 4 bitlik medyan filtreleme sonucu çıkış sinyali; "OG_PIXEL" ile 4 bitlik gauss filtreleme sonucu çıkış sinyali; "I_ENABLE" ile seçim ucunun aktif olduğunu gösteren sinyaller tanımlanmıştır. "I_LVAL" 1 satır yani 320 piksel boyunca '1' değerini alır. Saat sinyalinin yükselen kenarında "I_LVAL" değeri '1' ise sayaç, pikselleri saymaktadır. "I_ENABLE" değeri '1' olduğunda ise monitöre görüntü yazılmaktadır. "conv_ena" sinyali '1' değerini aldığında ise filtreleme işlemi için konvolüsyon başlatılır. RGB444 formatında 0 ile 15 bit arasında piksel değerleri oluşturulur. Kernel matrisini oluşturan sinyaller, 0'dan 15'e kadar toplam 16 farklı değer almaktadır. Bu işlem 20 kez tekrarlandığında 320 pikselden oluşan 1 satır meydana gelmektedir. Her satır arabelleği için okuma ve yazma sinyal değerleri olan "fifo line buffer" grubu simülasyona eklenmiştir.

4.3.1. Medyan Filtreleme Simülasyon Sonucu

Şekil 4.11'de kernel matris elemanlarının seçilen bir zaman aralığında değerleri gösterilmiştir. Bu zaman aralığında conv_ena sinyali '1' değerine ulaştığında konvolüsyon işlemi başlatılmaktadır. Sayacın 5761. pikselde olduğu gözlemlenmektedir. Her satır 320 pikselden oluştuğundan bu durum, 19. satırın ilk pikseline karşılık gelmektedir. 9 saat sinyali sonrası medyan değeri elde edilir.

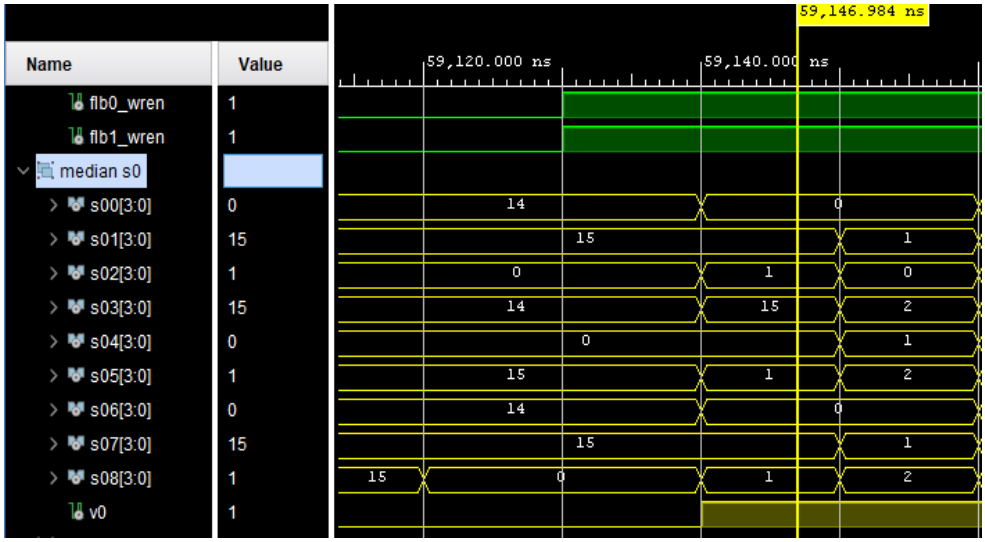


Şekil 4.11: Medyan filtre simülasyonu, kernel matris değerleri testbench gösterimi.

Kernel grubunda, kernel matrisi elemanları olan "shift_r5: 15" , "shift_r4: 0", "flb1_out: 1", "shift_r3: 15", "shift_r2: 0", "flb0_out: 1", "shift_r1: 15", "shift_r0: 0", ve "pixel_in: 1" değerlerini almıştır. Bu sinyallerin aldığı değere göre örnek konvolüsyon işlemi yapılacaktır. Şekil 4.12'de ise aynı zaman aralığında medyanın ilk durum (s0) değerleri gösterilmiştir.

"pixel_in" değeri '1', direkt olarak s08'e atanmıştır. İlk durum için aşağıdaki karşılaştırmalar yapılmıştır:

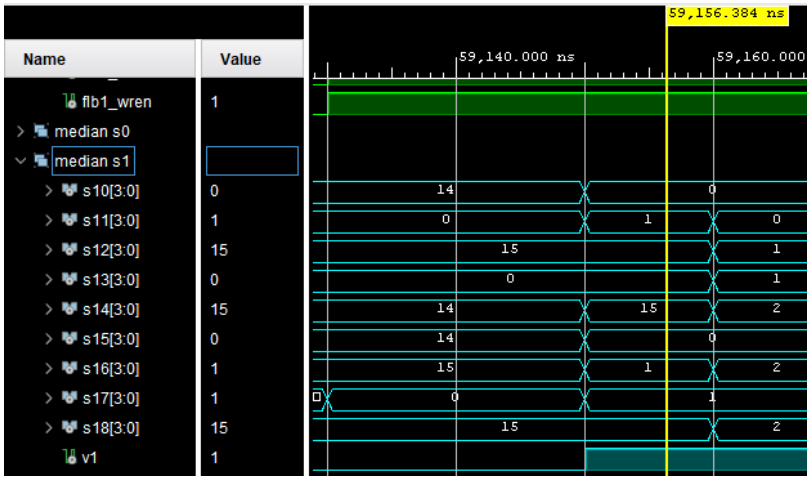
- shift_r5 , shift_r4 arasında, (15 > 0), (küçük olan s00: 0, büyük olan s01: 15 olur.)
- flb1_out ve shift_r3 arasında, (1 < 15), (küçük olan s02: 1, büyük olan s03: 15 olur.)
- shift_r2 ve flb0_out arasında, (0 < 1), (küçük olan s04: 0, büyük olan s05: 1 olur.)
- shift_r1 ve shift_r0 arasında. (15 > 0), (küçük olan s06: 0, büyük olan s07: 15 olur.)



Şekil 4.12: Medyan filtre simülasyonu, medyan s0 değerleri testbench gösterimi.

Şekil 4.13'te, bir clock sonrasında s1 durumu medyan değerleri gösterilmiştir. s1 durumuna gelindiğinde "s00: 0" değeri, direkt olarak "s10" sinyaline atanır. S1 durumu için aşağıdaki karşılaştırmalar yapılmıştır:

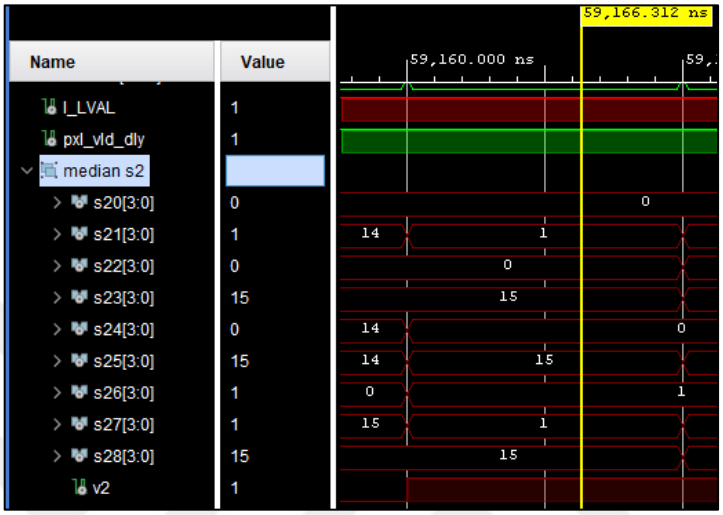
- s01 ve s02 arasında, (15 > 1), (küçük olan s11: 1, büyük olan s12: 15 olur.)
- s03 ve s04 arasında, (15 > 0), (küçük olan s13: 0, büyük olan s14: 15 olur.)
- s05 ve s06 arasında, (1 > 0), (küçük olan s15: 0, büyük olan s16: 1 olur.)
- s07 ve s08 arasında. (15 > 1), (küçük olan s17: 1, büyük olan s18: 15 olur.)



Şekil 4.13: Medyan filtre simülasyonu, medyan s1 değerleri testbench gösterimi.

Şekil 4.14'te, bir clock sonrasında s2 durumu medyan değerleri gösterilmiştir. "s18: 15" değeri, direkt olarak "s28" sinyaline atanır. s2 durumu için aşağıdaki karşılaştırmalar yapılmıştır:

- s10 ve s11 arasında, (0 < 1), (küçük olan s20: 0, büyük olan s21: 1 olur.)
- s12 ve s13 arasında, (15 > 0), (küçük olan s22: 0, büyük olan s23: 15 olur.)
- s14 ve s15 arasında, (15 > 0), (küçük olan s24: 0, büyük olan s25: 15 olur.)
- s16 ve s17 arasında. (1 = 1), (s26: 1, s27: 1 olur.)



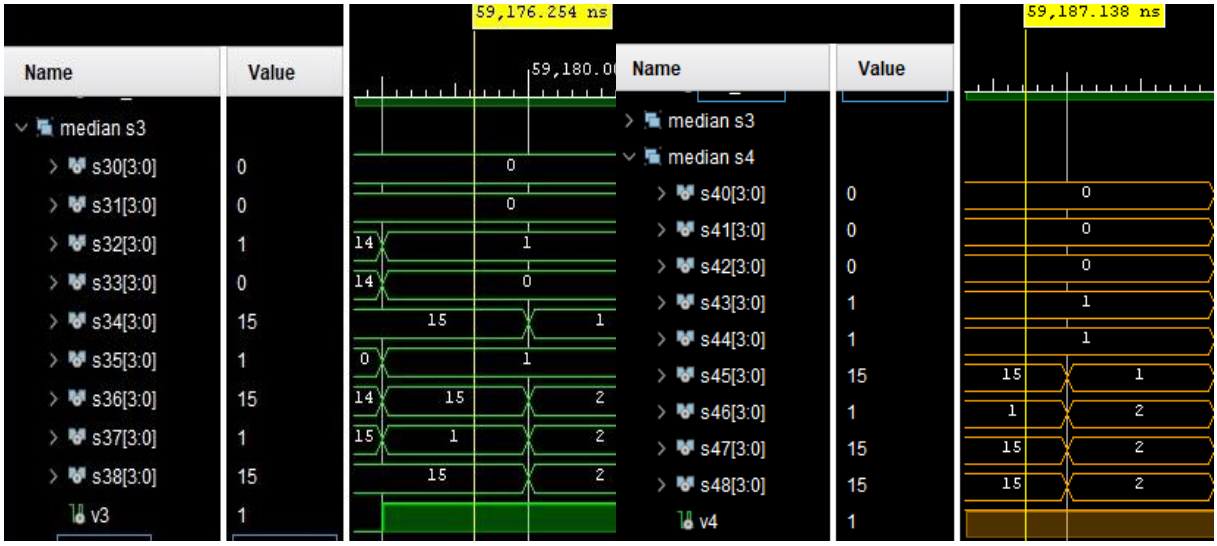
Şekil 4.14: Medyan filtre simülasyonu, medyan s2 değerleri testbench gösterimi.

Şekil 4.15'te, bir clock sonrasında s3 durumu, bir clock daha sonrasında ise s4 durumu medyan değerleri gösterilmiştir. "s20: 0" değeri, direkt olarak "s30" sinyaline atanır. S3 durumu için aşağıdaki karşılaştırmalar yapılmıştır:

- s21 ve s22 arasında, (1 > 0), (küçük olan s31: 0, büyük olan s32: 1 olur.)
- s23 ve s24 arasında, (15 > 0), (küçük olan s33: 0, büyük olan s34: 15 olur.)
- s25 ve s26 arasında, (15 > 1), (küçük olan s35: 1, büyük olan s36: 15 olur.)
- s27 ve s28 arasında. (1 < 15), (küçük olan s37: 1, büyük olan s38: 15 olur.)

"s38: 15" değeri, direkt olarak "s48" sinyaline atanır. S4 durumu için aşağıdaki karşılaştırmalar yapılmıştır:

- s30 ve s31 arasında, (0 = 0), (s40: 0, s41: 0 olur.)
- s32 ve s33 arasında, (1 > 0), (küçük olan s42: 0, büyük olan s43: 1 olur.)
- s34 ve s35 arasında, (15 > 1), (küçük olan s44: 1, büyük olan s45: 15 olur.)
- s36 ve s37 arasında. (15 > 1), (küçük olan s46: 1, büyük olan s47: 15 olur.)



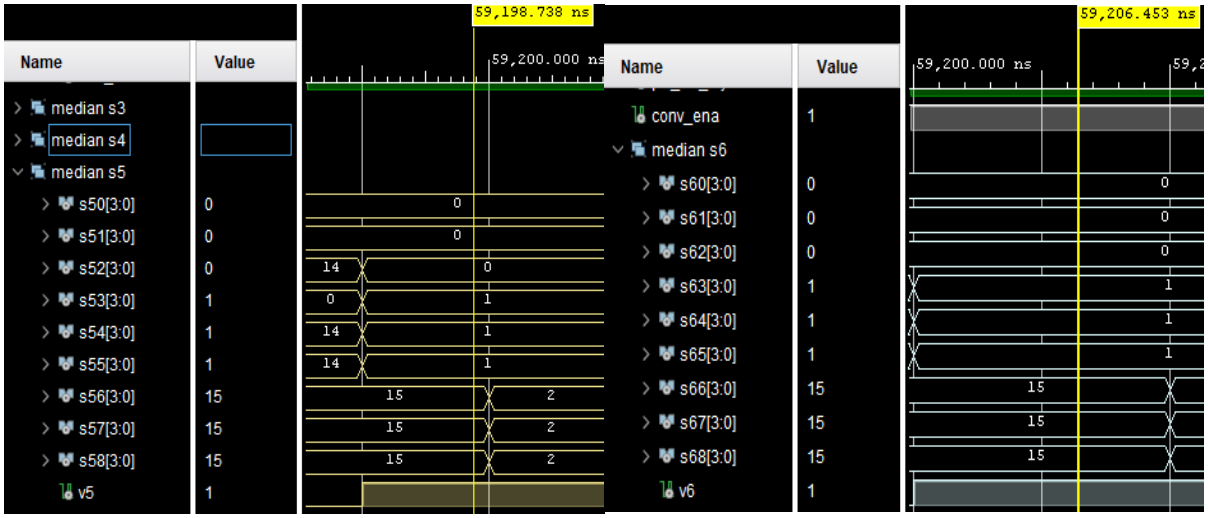
Şekil 4.15: Medyan filtre simülasyonu, medyan s3, s4 değerleri testbench gösterimi.

Şekil 4.16'da, bir clock sonrasında s5 durumu, bir clock daha sonrasında ise s6 durumu medyan değerleri gösterilmiştir. "s40: 0" değeri, direkt olarak "s50" sinyaline atanır. S5 durumu için aşağıdaki karşılaştırmalar yapılmıştır:

- s41 ve s42 arasında, (0 = 0), (s51: 0, s52: 0 olur.)
- s43 ve s44 arasında, (1 = 1), (s53: 1, s54: 1 olur.)
- s45 ve s46 arasında, (15 > 1), (küçük olan s55: 1, büyük olan s56: 15 olur.)
- s47 ve s48 arasında. (15 = 15), (s57: 15, s58: 15 olur.)

"s58: 15" değeri, direkt olarak "s68" sinyaline atanır. S6 durumu için aşağıdaki karşılaştırmalar yapılmıştır:

- s50 ve s51 arasında, (0 = 0), (s60: 0, s61: 0 olur.)
- s52 ve s53 arasında, (0 < 1), (küçük olan s62: 0, büyük olan s63: 1 olur.)
- s54 ve s55 arasında, (1 = 1), (s64: 1, s65: 1 olur.)
- s56 ve s57 arasında. (15= 15), (s66: 15, s67: 15 olur.)



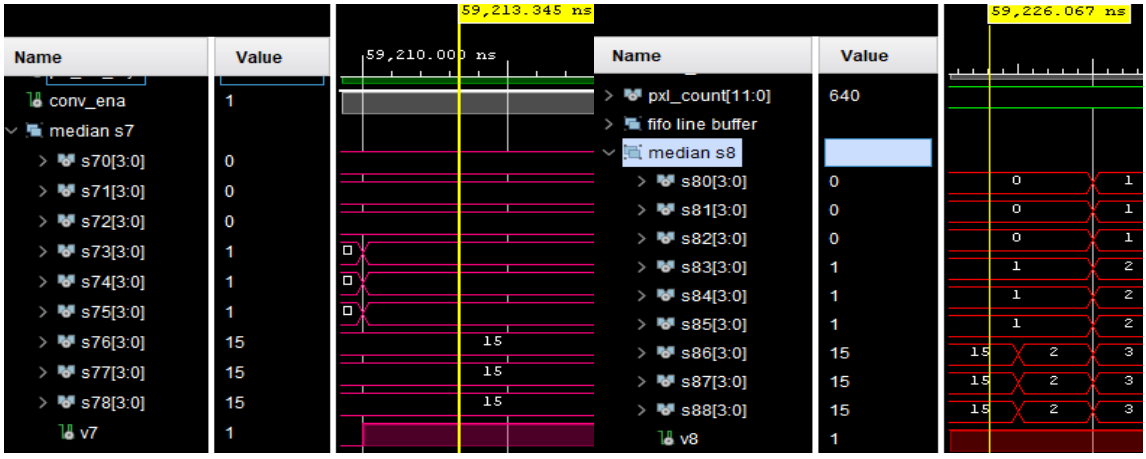
Şekil 4.16: Medyan filtre simülasyonu, medyan s5, s6 değerleri testbench gösterimi.

Şekil 4.17'de, bir clock sonrasında s7 durumu, bir clock sonrasında s8 durumu gösterilmiştir. "s70: 0" değeri, direkt olarak "s80" sinyaline atanır. S7 durumu için aşağıdaki karşılaştırmalar yapılmıştır:

- s61 ve s62 arasında, (0 = 0), (s71: 0, s72: 0 olur.)
- s63 ve s64 arasında, (1 = 1), (s73: 1, s74: 1 olur.)
- s65 ve s66 arasında, (1 < 15), (küçük olan s75: 1, büyük olan s76: 15 olur.)
- s67 ve s68 arasında. (15 = 15), (s77: 15, s78: 15 olur.)

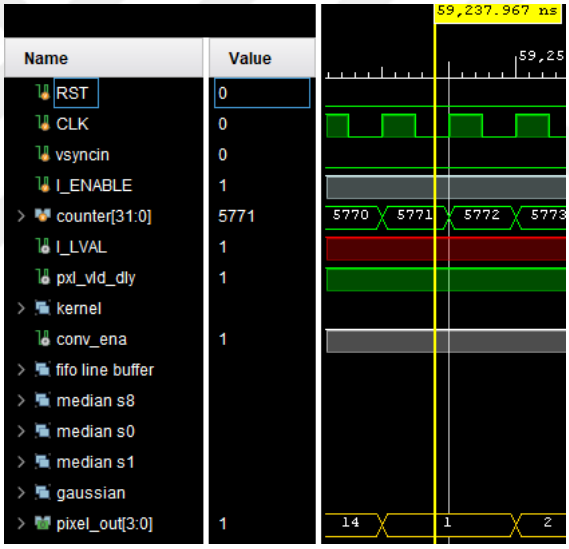
"s78: 15" değeri, direkt olarak "s88" sinyaline atanır. S8 durumu için karşılaştırma:

- s70 ve s71 arasında, (0 = 0), (s80: 0, s81: 0 olur.)
- s72 ve s73 arasında, (0 < 1), (küçük olan s82: 0, büyük olan s83: 1 olur.)
- s74 ve s75 arasında, (1 = 1), (s84: 1, s85: 1 olur.)
- s76 ve s77 arasında. (15= 15), (s86: 15, s87: 15 olur.)



Şekil 4.17: Medyan filtre simülasyonu, medyan s7,s8 değerleri testbench gösterimi.

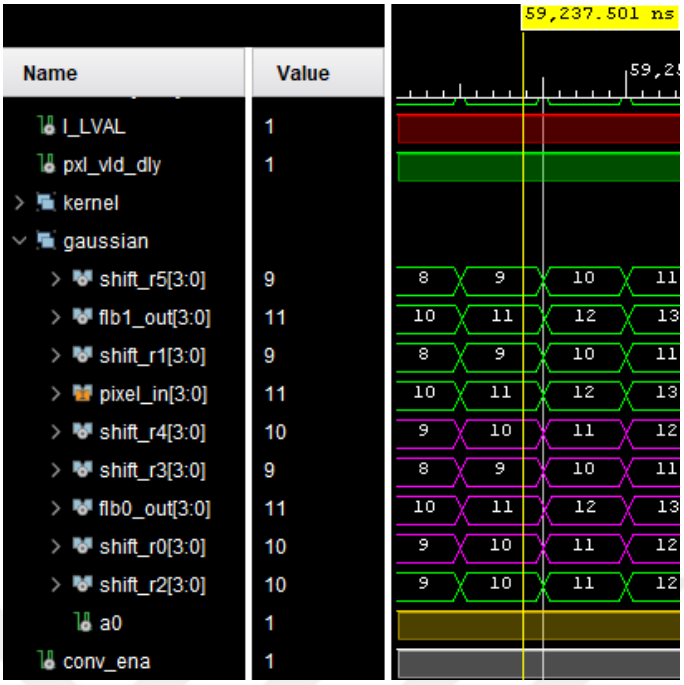
En son medyan değeri ortadaki değer olan "s84" değeridir. S8 durumundan bir sonraki clockta çıkış pikseli olarak alınmıştır. Bu değer, Şekil 4.18'de gösterilen "pixel_out" ile ifade edilir.



Şekil 4.18: Medyan filtre simülasyonu, medyan değerinin testbench gösterimi.

4.3.2. Gauss Filtreleme Simülasyon Sonucu

Şekil 4.19'da kernel matris elemanlarının seçilen bir zaman aralığında değerleri gösterilmiştir. Bu zaman aralığında conv_ena sinyali '1' değerine ulaştığında konvolüsyon işlemi başlatılmaktadır. Sayacın 5771. pikselde olduğu gözlemlenmektedir. Her satır 320 pikselden oluştuğundan bu durum, 19. satırın 12. pikseline karşılık gelmektedir. 3 saat sinyali sonra gauss filtreleme işlemi tamamlanmaktadır.



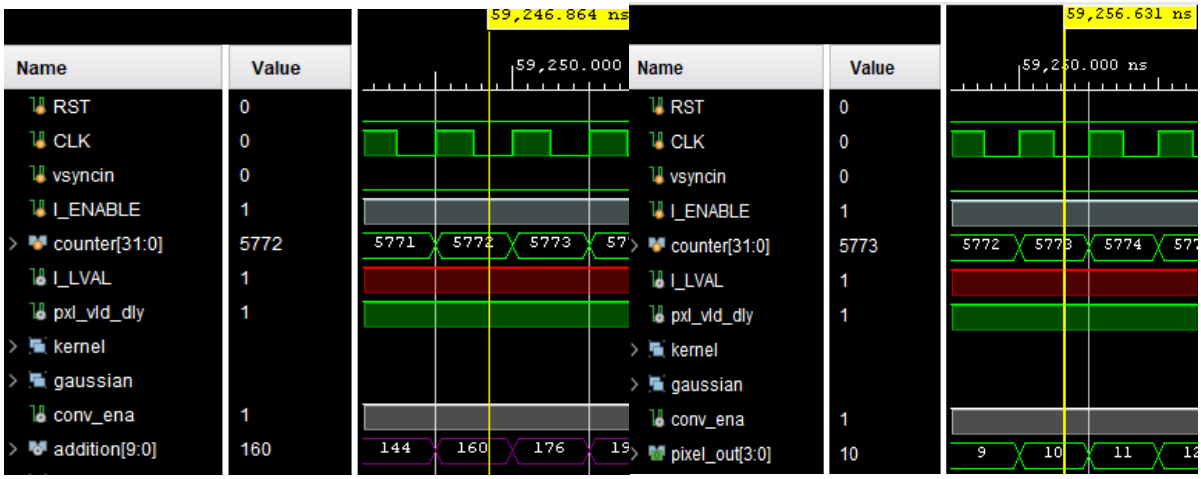
Şekil 4.19: Gauss filtre simülasyonu, kernel matris değerleri testbench gösterimi.

Gauss formülü denklem 3.1'de gösterilmişti. Çarpma ve toplama işlemlerinin sonucunun yazılması için "addition" isminde 10 bitlik sinyalin hesaplanması denklem 4.1'de gösterilmiştir.

addition= (4.1)

$$[\,(9.1 + 10.2 + 11.1) + (9.2 + 10\,.4 + 11.2) + (9.1 + 10.2 + 11.1)\,] = 160$$

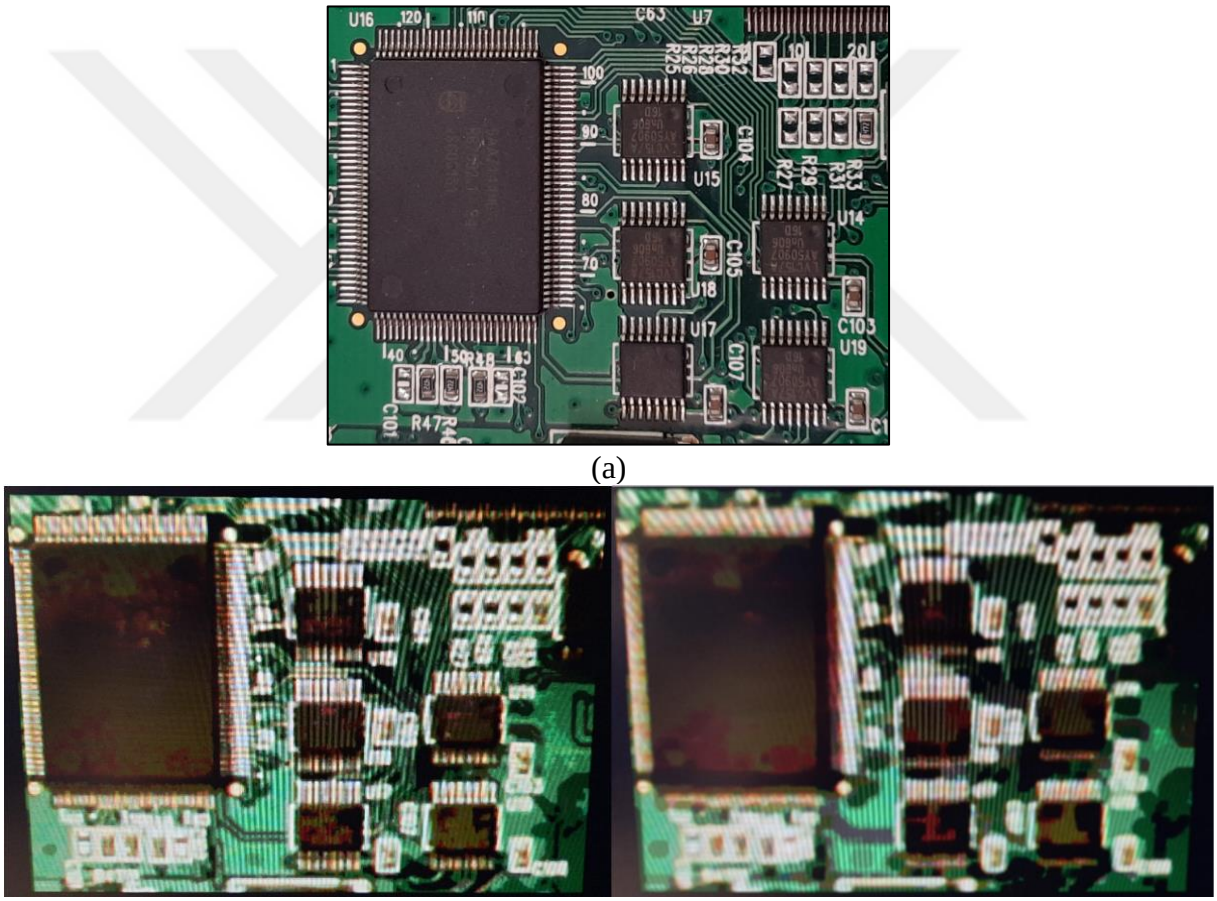
Toplam değeri 16'ya bölünerek gauss filtreleme sonucu çıkış pikseli, 10 olarak elde edilir. Şekil 4.20'de bir saat sinyali sonrası "addition" değeri ve bir saat sinyali daha sonrası için gauss filtreleme sonucu çıkış piksel değeri, "pixel_out" portunda gösterilmektedir.



Şekil 4.20: Gauss filtre simülasyonu, gauss değerinin testbench gösterimi.

4.3.3. Filtrelemeler Sonucu Monitör Çıktısı

Medyan ve gauss filtreleme uygulaması, gerçek zamanlı olarak örnek baskı devre kartlarında gerçekleştirilmiştir. Şekil 4.21'de filtrelemeler için örnek uygulama gösterilmiştir. (a) görüntüsünde PCB'ye ait orijinal görüntü gösterilmiştir. (b) görüntüsünde bu PCB'nin filtrelenmemiş 320x240 boyutunda monitördeki gösterimine yer verilmiştir. (c)'de 2 kez medyan, 2 kez gauss filtreleme için 4 adet kullanıcı anahtarı aktif duruma getirilmiştir ve filtrelemelerin tamamı uygulanmıştır. Uygulamaların tamamı nanosaniye mertebesinde gerçekleşmektedir. Uygulama esnasında herhangi bir veri kaybı, görüntüde bozulma olmamaktadır. Şekil 4.22'de başka bir PCB üzerinde filtre uygulamaları gerçekleştirilmiştir.

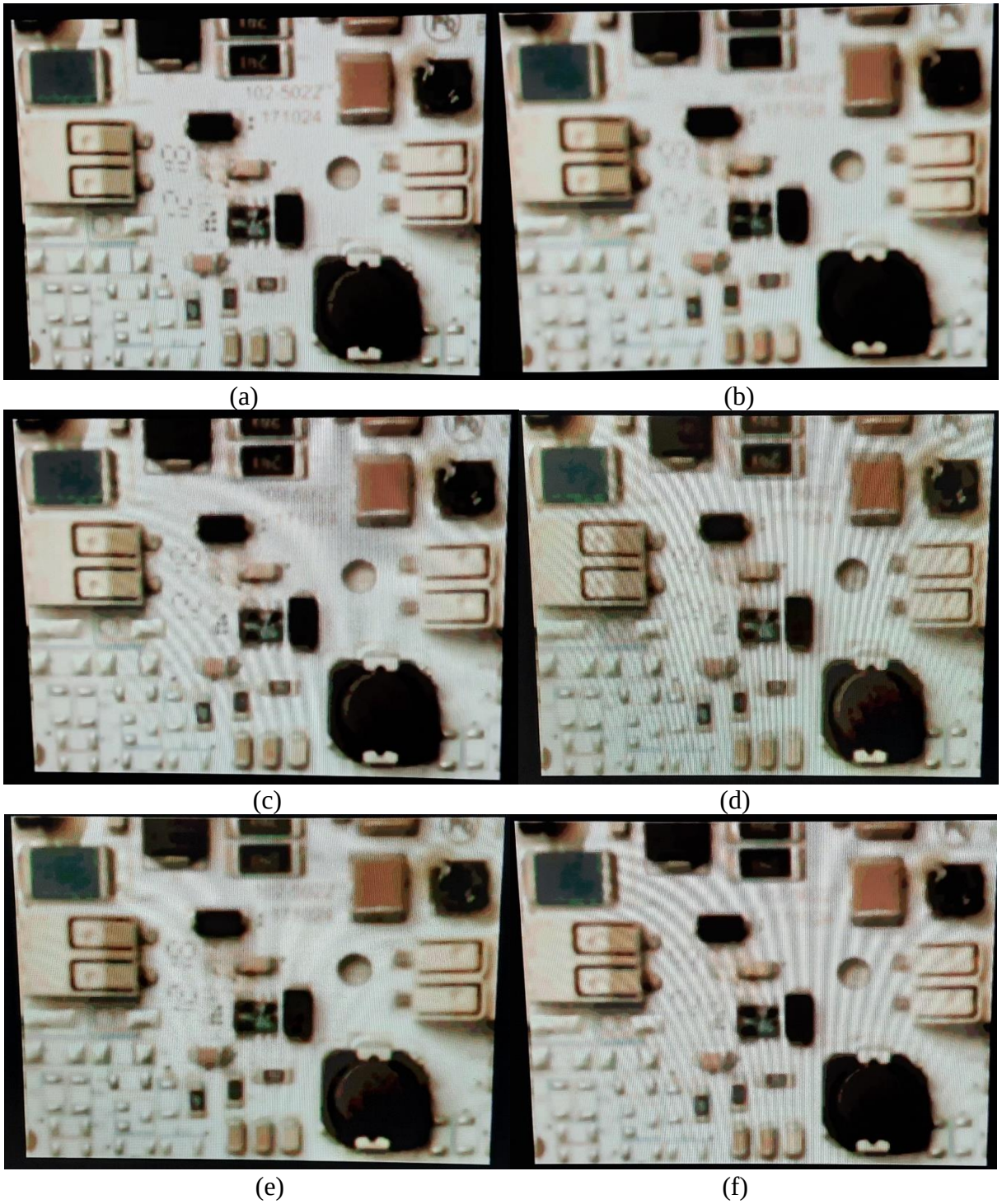


(a)

(b) (c)

Şekil 4.21: Örnek PCB üzerinde filtrelerin uygulanması.

(a) Orijinal PCB görüntüsü. (b) Monitörde filtresiz PCB görüntüsü.
(c) 2 medyan + 2 gauss filtreli PCB.



(a) (b)

(c) (d)

(e) (f)

Şekil 4.22: Örnek PCB üzerinde filtrelerin uygulanması.

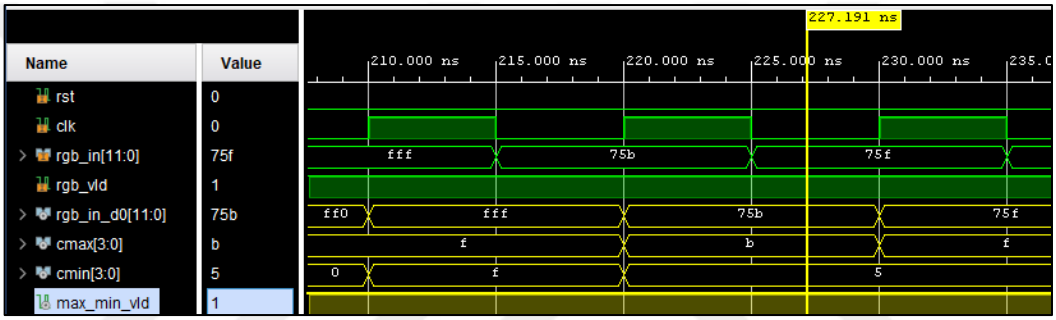
(a) Monitörde filtresiz PCB görüntüsü. (b) 2 medyan + 2 gauss filtreli. (c) 1 medyan filtreli. (d) 2 medyan filtreli (e) 1 medyan + 1 gauss filtreli. (f) 2 medyan + 1 gauss filtreli.

4.4. EKSİK KOMPONENT HATA TESPİT SONUÇLARI

4.4.1. HSV Simülasyon Sonucu

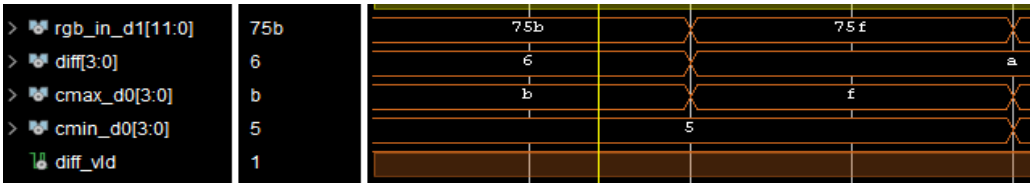
Eksik komponentlerin tespiti için lehimli alanların monitörde kırmızı renk ile gösterilmesi sağlanmıştır. Bunun için parlaklık seviyesi sayesinde lehim rengini HSV renk uzayında temsil eden gri tonlarına yakın olan renk aralığı belirlenerek test edilmiştir. RGB444 formatındaki "7,5,b" değeri, RGB888 formatında 16 ile çarpılarak "112, 80, 176" değerleri ile gösterilir.

Şekil 4.23'te simülasyonda HSV değerlerinin hesaplanması için bir zaman aralığındaki değerleri gösterilmiştir. Simülasyondaki "75b" değeri için maksimum olan değer mavi rengine karşılık gelmektedir. Cmax değeri 'b', Cmin değeri '5' olmaktadır.

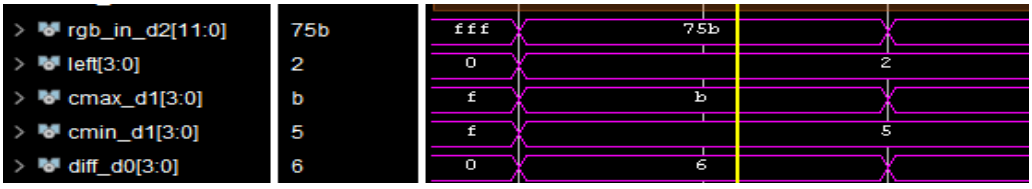


Şekil 4.23: HSV simülasyonu, Cmax ve Cmin değerleri testbench gösterimi.

"75b" değerinin olduğu durum için, Cmax - Cmin fark değerini gösteren diff değeri 'b-5' (onluk tabanda: 11-5) olur. Bu durumda diff[3:0], Şekil 4.24'te gösterildiği gibi 6 değerine karşılık gelmektedir. Şekil 4.25'te mavi maksimum olduğu için hue değerinin hesaplanması için "R" değerinden "G" değerinin çıkartılarak "left" sinyali elde edilir. 'left' sinyalinin değeri, "7-5" işleminden 2 olmaktadır.

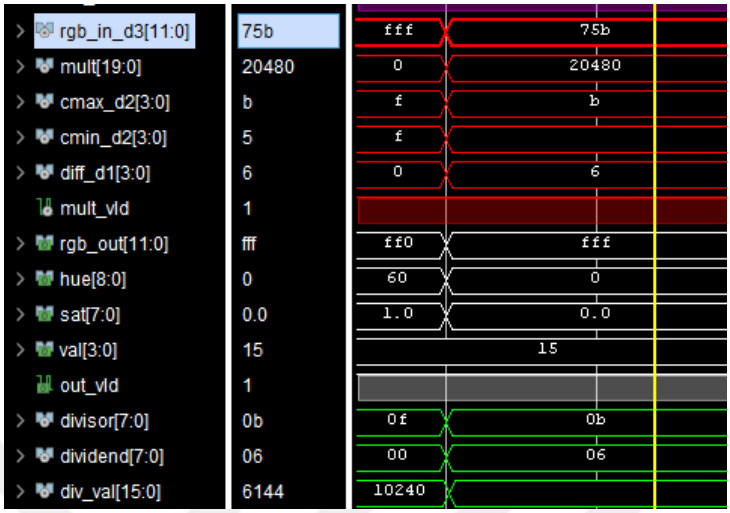


Şekil 4.24: HSV simülasyonu, fark değeri testbench gösterimi.



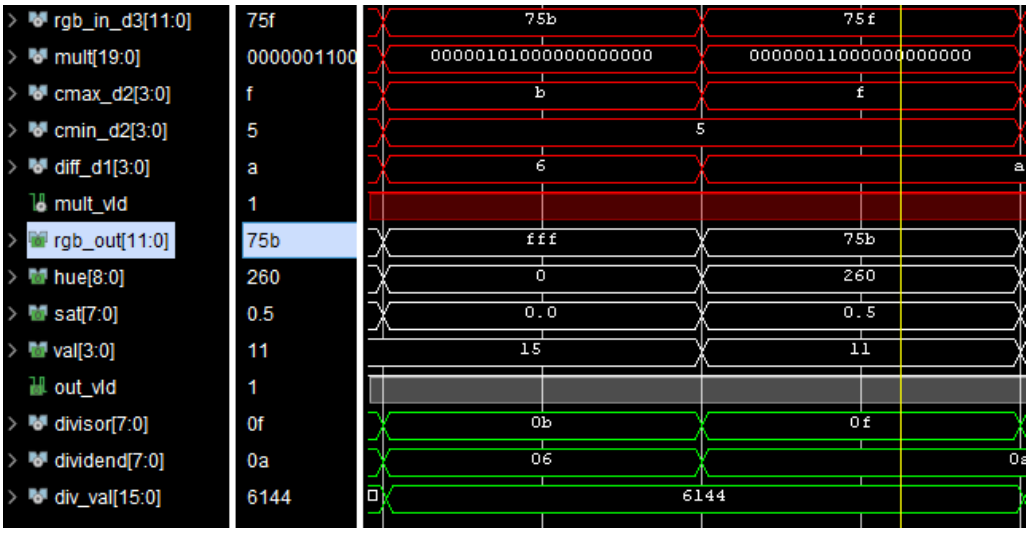
Şekil 4.25: HSV simülasyonu, left değeri testbench gösterimi.

Şekil 4.26'da gösterilen 'mult' sinyali, 'left' ile 'div_val' sinyalinin çarpımıdır. Bu çarpım değeri, 2 ve 10.240 değerlerinin çarpımı ile 20.480 olmaktadır.



Şekil 4.26: HSV simülasyonu, çarpım değeri testbench gösterimi.

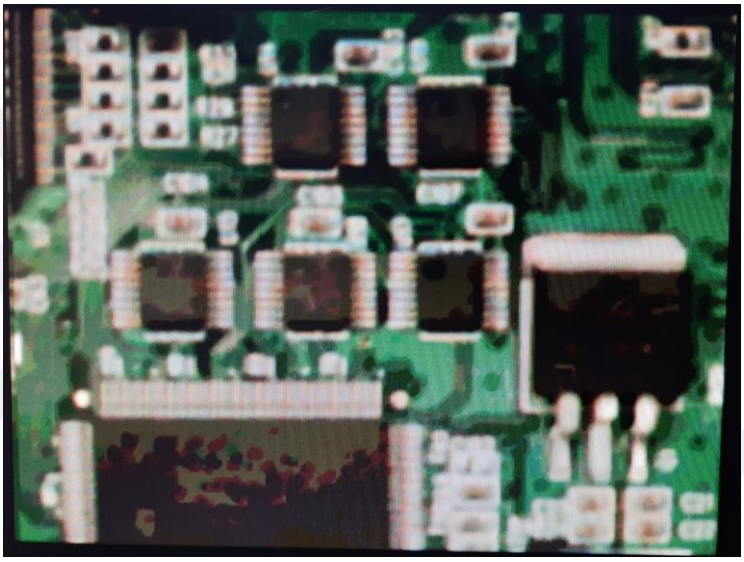
Şekil 4.27'de mult sinyali ikilik tabanda gösterilmiştir.18.bit ile 10.bit arası alınırsa '00010100' değeri elde edilir. Bu değer onluk tabanda 20 değerine karşılık gelmektedir. Bu değere, hue değeri hesaplanırken mavi değeri maksimum olduğu için formülde yer alan 60°x4 = 240 değeri eklenir. Böylece Hue değeri 260° olarak simülasyonda elde edilmiştir. Renk doygunluğu (S) değeri ise, diff '6' değerinin Cmax 'b' (onluk tabanda 11) değerine oranıdır. Böylece S değeri, 6/11 işleminden 0.54 olarak bulunur. Simülasyonda 0.5 olarak elde edilmiştir. Renk parlaklığı (V) değeri; Cmax, 11 değeri olarak gösterilmiştir.



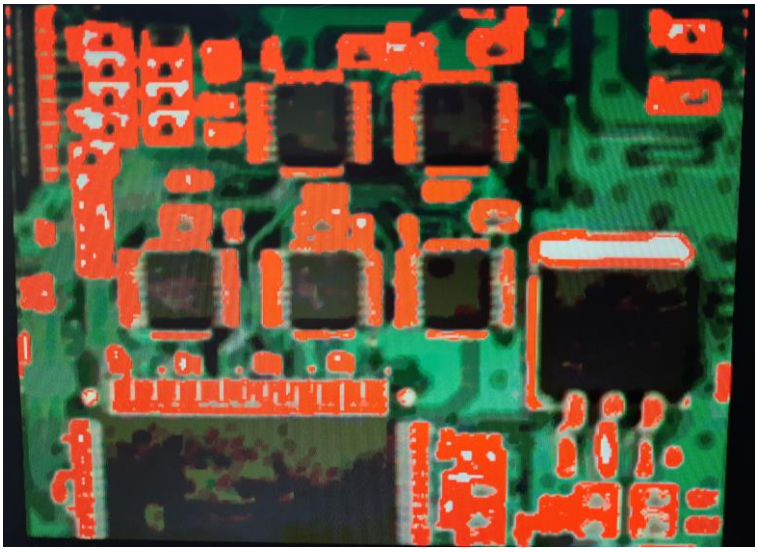
Şekil 4.27: HSV simülasyonu, H, S, V değerleri testbench gösterimi.

4.4.2. Eksik Komponent Hata Tespiti Sonucu Monitör Çıktısı

Gerçek zamanlı olarak örnek baskı devre kartında lehimli alanların gösterilerek eksik komponent hatasının tespit edilmesi gerçekleştirilmiştir. Şekil 4.28'de bir medyan, bir gauss filtreleme uygulanmış baskı devre kartı monitör çıktısı gösterilmiştir. Şekil 4.29'da ise aynı baskı devre kartının filtrelemelerden sonra lehimli alanlarının monitörde gösterimi sağlanmıştır. Görüntü, 320x240 boyutunda ve RGB444 formatındadır. Lehimli alanların gösterimi için 1 adet kullanıcı anahtarı aktif duruma getirilmiştir. Uygulama esnasında herhangi bir veri kaybı, görüntüde bozulma olmamaktadır.

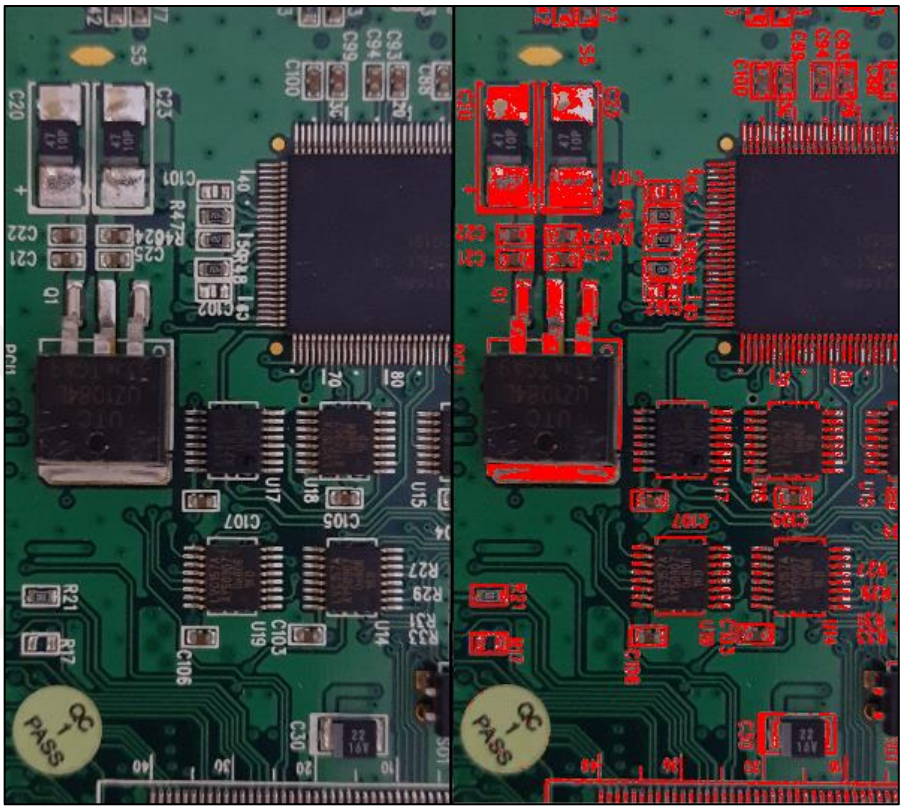


Şekil 4.28: Örnek PCB üzerinde bir medyan ve bir gauss filtresi uygulaması.



Şekil 4.29: Örnek PCB üzerinde eksik komponent hatası tespiti.

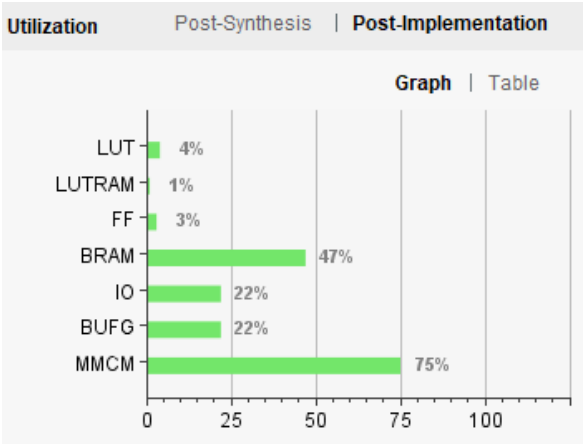
RGB444 ve RGB888 format farkının gösterilmesi için örnek PCB görüntüsü, yüksek çözünürlüklü kamera ile alınarak MATLAB ortamında lehimli alanlar işaretlenmiştir. İki format arasındaki farka bakıldığında yüksek çözünürlük kamera ile alınan görüntüde lehimli alanların gösterilmesi daha düzgün sonuç vermiştir. Şekil 4.30'da sol tarafta örnek PCB görüntüsü, sağ tarafta ise MATLAB'de lehimli alanları işaretlenen PCB görüntüsü mevcuttur.



Şekil 4.30: MATLAB ortamında lehimli alanların işaretlenmesi.

4.5. KAYNAK KULLANIMI VE ZAMAN ANALİZİ

Şekil 4.31'de çalışma sonucunda gerçekleşen ZedBoard FPGA üzerinde kaynak kullanımlarına yer verilmiştir. Flip floplar (FF), process içerisinde kullanılan sinyallerin oluşturduğu; Look-up table (LUT), kapılar ve if-else bloklarında kullanılan karşılaştırmaların; Block RAM (BRAM), kullanılan 2 adet çerçeve arabelleği IP bloğunun; LUTRAM, BRAM dışındaki hafıza elemanlarının; Global Buffer (BUFG), 3.3V ve LUT besleme amaçlı; Mixed-mode clock manager (MMCM), kullanılan 3 adet saat üreteci IP bloğunun kaynak kullanımlarını ifade etmektedir.



Şekil 4.31: Kaynak kullanımının Vivado programında gösterilmesi.

Şekil 4.32'de ZedBoard FPGA zaman analizine yer verilmiştir. Her giriş portunun, maksimum gecikmesi ve minimum gecikmesi olabilir. Maksimum gecikme için setup tarafına, minimum gecikme için hold tarafına bakılmaktadır.

"En Kötü Negatif Bolluk (WNS)" pozitifse zamanlama başarılıdır. Negatif ise, başarısız olduğu anlamına gelir.

"Toplam Negatif Bolluk (TNS)", tasarımdaki negatif bolluğun toplamıdır. 0 ise, tasarım zamanlamayı karşılar. Pozitif sayı ise, tasarımda negatif bolluk olduğu anlamına gelir, dolayısıyla tasarım başarısız olur.

"En Kötü Tutma Gevşekliği (WHS)", WNS gibi pozitifse başarılı, negatifse başarısız olduğu anlamına gelir.

"Toplam Tutma Gevşemesi (THS)", negatif tutma gevşekliğinin toplamıdır ve değeri sıfır ise tasarım başarılıdır. Oluşturulan tasarım, zamanlama şartlarını karşılamaktadır.

Timing	Setup
Worst Negative Slack (WNS):	12.067 ns
Total Negative Slack (TNS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	6639

Timing	Setup | Hold
Worst Hold Slack (WHS):	0.012 ns
Total Hold Slack (THS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	6639

Şekil 4.32: Zamanlama raporunun Vivado programında gösterilmesi.

5. TARTIŞMA

Tez çalışmasındaki sistem tasarımı, literatürdeki çoğu çalışmalar ile karşılaştırıldığında hazır şekilde sunulan araçlar kullanılmadan, saf donanım olarak tamamen VHDL ile kodlanarak Vivado programında oluşturulmuştur. Literatürde ise sıkça Matlab Simulink araçları, OpenCV kütüphanesi, C / C++ dili ile Vivado HLS tarafından donanım oluşturma araçları kullanılmaktadır. VHDL ile kodlanarak saf donanım oluşturulması; yazılım veya donanım / yazılım ortak tasarımının uygulamasından daha iyi işlem hızına sahip performans sunmaktadır, fakat daha karmaşık ve düşük esnekliktedir [11, 24, 38].

Yazılım ile programlanan mikrodenetleyici sistemler, işlemleri belirli bir sıraya göre yapmaktadır. Görüntü işleme sistemi açısından bakıldığında 320x240 çözünürlüğe sahip bir görüntü çerçevesi için 76.800 piksel işlenmesi gerekmektedir. Gerçek zamanlı bir sistem olduğunda saniyede 60 görüntü çerçevesinin gösterilmesi için işlemlerin sırayla yapılması sistemi yavaşlatmaktadır. Sistemin FPGA tabanlı, donanımsal olması sayesinde paralel bir şekilde tüm pikseller bir saat sinyali ile aynı anda işlenmektedir. Bu durum yazılıma göre avantaj sağlamaktadır.

Tasarımda, ZedBoard FPGA geliştirme kartının üzerinde bulunan Zynq-7000 işlemcisi kullanılmamıştır. Bu işlemcinin kullanılmaması sayesinde yazılan VHDL kodları, marka ve model fark etmeden tüm FPGA geliştirme kartlarında kullanılabilir.

Tez çalışmasında kameradan gelen bir görüntü çerçevesini saklayan iki adet çerçeve arabelleği mevcuttur. Bu çerçeve arabellekleri olmadan mevcut tasarımı kullanmak mümkündür. Tasarımda kullanılmasının tek sebebi, kameranın 30 fps, VGA kısmının ise 60 Hz hızında çalışmasıdır. VGA, kameradan iki kat daha hızlı çalıştığı için iki adet çerçeve arabelleğinde görüntü depolanmıştır. Kameradan yeni görüntü alınana kadar VGA diğer görüntüyü monitörde göstermektedir. Mevcut tasarımda 60 fps hızına sahip kamera kullanılırsa, VGA monitörde aynı 60 Hz hızında seçilirse çerçeve arabellekleri olmadan girişten alınan görüntü verileri işlenir ve direkt olarak VGA monitöre verilebilir. Ayrıca sistem, mevcut tasarımda yer alan iki adet çerçeve arabelleği sayesinde farklı hıza sahip VGA ve kamera için çalışabilmektedir.

Tez çalışmasında, PCB üzerinde komponent olan kısımlar görüntü üzerinde işaretlenebilir. Fakat, RGB444 formatında kart renginin monitör çıktılarında görüldüğü üzere koyu yeşil renk olarak gözükmesi, komponentlerin siyah olması sebebiyle bu iki renk arasında ayrım yapılamayacağı için elde edilen görüntü sonucu düzgün çıktı vermeyecektir. Aynı işlemler, RGB888 formatında yapılırsa komponent olan kısımlar görüntü üzerinde bulunabilir. Tez çalışmasında, RGB444 formatı kullanılmasının nedeni, görüntü sensörünün sağladığı renk çözünürlüğün maksimum 8 bit olması ve ZedBoard VGA'nın renk çözünürlüğünün 12 bit olmasıdır. RGB888 formatında monitörde gösterilebilecek renk sayısı toplam 24 bitten, 2^{24} ile hesaplandığında yaklaşık 16 milyon renk sayısına karşılık gelmektedir. RGB444 için ise toplam 12 bitten, 2^{12} ile yaklaşık 5 bin renk sayısıdır. Bu durumda gösterilebilecek renk sayısı oldukça azalmıştır.

Aşağıda yer alan tüm çalışmalar, tez çalışması ile karşılaştırıldığında, üzerinde komponent bulunmayan PCB üzerinde hata tespitlerinin yapılması, tamamının yazılımsal olması, FPGA tabanlı olmamaları ve gerçek zamanlı olmaması gibi dezavantajlara sahiptir. Genellikle gerçek zamanlı olmayan test görüntüleri alınarak referans kart ile karşılaştırılıp görüntü çıkartma algoritması kullanılmıştır. Görüntü çıkartma algoritmasının uygulanabilmesi için referans ve test görüntüsünün piksellerinin tamamının denk gelmesi gerekmektedir. Bu sebeple test görüntüsünde oluşabilecek herhangi bir açısal eğim bu algoritmanın hatalı sonuç vermesine sebep olur.

Nayak, Anitha ve diğerleri [38], üzerinde komponent bulunmayan PCB hatalarını Matlab yazılım dili ile tespit etmişlerdir. Görüntülerdeki PCB eğim açısı, kötü aydınlatma koşullarındaki görüntüler, görüntülerin çekildiği dikey mesafeyi göz önünde bulundurarak mesafe sorununu çözemeyen yöntemlerden farklı olarak test edilen görüntü ile referans ve görüntüyü yazılım ile karşılaştırmışlardır.

Sunil, Ambadas ve diğerleri [39], üzerinde komponent bulunmayan PCB görüntüsünü alarak yeniden boyutlandırır. Wiener filtresi ile gürültü giderme yapılır ve Canny Kenar Algılama Yöntemi ile PCB kenarının tespit edilmesini sağlarlar. Kenarı tespit edilen PCB'nin eğim açısı kontrol edilerek görüntüyü kırpar ve görüntü üzerinde eşikleme (thresholding) işlemi gerçekleştirerek ikili görüntü elde edilir. Referans ve test görüntüsünü karşılaştırarak görüntü çıkarma (subtraction) gerçekleştirir. Hatalar PCB üzerinde tespit edilir ve Matlab GUI kullanılarak uygulamada gösterilir. Ayrıca Shinde ve Morade [40], referans PCB ve test

edilecek olan PCB görüntülerini almışlardır. Bu görüntüler üzerinde görüntü çıkarma işlemi uygulamışlardır.

Vinita ve Kaushik [41], üzerinde komponent bulunmayan PCB'ye ait eksik ve kısa devre olmuş su yollarını ve eksik delik gibi PCB hatalarını tespit etmek için bilgisayarlı görü sistemi önermişlerdir. Ayrıca PCB'yi tanıdıktan sonra orijinal PCB ile karşılaştırarak PCB arıza türleri de kategorize etmektedir. Sistemde bir görüntü karşılaştırma tekniği kullanarak, test edilen kart görüntüsünden referans kartı görüntüsünü çıkararak olası hataları bulabilen çıkarma algoritması tanımlamışlardır.

Ortega [42], üzerinde komponent bulunmayan PCB hatalarını tespit etmek için uzaysal filtreleme ve dalgacık tabanlı otomatik optik inceleme sistemi sunmaktadır. Referans görüntü dalgacık dönüşümü ve test görüntüsü dalgacık dönüşümünün yaklaşık değerlerinin çıkarılması ve ardından bir medyan filtre ile gürültünün kaldırılması esasına dayanan ve ters dalgacık dönüşümünü hesaplayarak hatalı görüntüyü elde eden bir sistem tasarlamıştır.

Liu ve Qu [3], PCB hata tespiti için görüntü yumuşatma, kontrast iyileştirme, keskinleştirme ve diğer ön işleme işlemlerini yaparak görüntü kalitesini iyileştirmek için gri tonlama dönüşüm algoritması kullanmışlardır. Hata tespiti için görüntü sapması algılama algoritmasını sunmuşlardır.

Chavan, Dokhe ve diğerleri [43], üzerinde komponent bulunan PCB'lerde kısa devre, eksik komponent gibi hataları PCB görüntüsünü bölümlere ayırarak tespit etmişlerdir. Canny kenar tespit algoritması, kontur analizi gibi çeşitli algoritmalar; gauss ve sobel filtreleri kullanmışlardır. XOR işlemi uygulayarak hata tespitini gerçekleştirmişlerdir.

Filtrelemelerle ilgili yapılan literatür araştırmaları ve karşılaştırmaları aşağıda yer almaktadır.

Sushma ve Kumar [44], FPGA tabanlı gerçek zamanlı medyan filtreleme tasarımı yapmışlardır. Çalışmalarında RAM'i görüntü çerçevesini kaydetme amacıyla kullanmışlardır. Görüntü çözünürlüğü, 1920x1080 olarak arttırıldığında tasarımları çalışamaz. Çünkü, tüm görüntü çerçevesini RAM'e kaydetme söz konusudur ve RAM, bu çözünürlükte yetersiz kalacaktır. Tezde sunulan mevcut tasarımda, görüntü çözünürlüğü 1920x1080p olsa bile tasarımın çalışabileceği MMCM IP bloğunda frekans değişikliği yapılarak gözlemlenmiştir. Bu çözünürlüğün kullanılamamasının sebebi, ZedBoard FPGA kaynak kullanımının yetersiz

kalmasıdır. Sushma ve Kumar'a ait çalışmada satır arabelleği olmadığı için birden fazla filtreleme yöntemi kullanılamaz. Tezdeki tasarımda, filtreleme aşamasında sadece satır arabellekleri ile 2 satır kaydedilerek filtreleme işlemi kaskadlanarak yapılmaktadır. Ayrıca, Sushma ve Kumar'a ait çalışmada ilerideki çalışmalar için RGB görüntü üzerinde filtreleme yapımını öneri olarak sunmuşlardır. Tezdeki tasarım, RGB görüntü üzerinde medyan ve gauss filtreleme yapmayı sağlamaktadır.

Rodriguez ve diğerleri [11], hem FPGA üzerinde hem de yazılımsal olarak gerçek zamanlı medyan filtresi tasarlamışlardır. FPGA ile donanım uygulaması tarafından elde edilen görüntü sonucu, yazılım sürümü ile elde edilen sonuç ile aynıdır. Fakat donanımın daha düşük yürütme süresine sahip olması çalışmalarında avantaj sağlamaktadır. Mevcut tez çalışması ile karşılaştırıldığında, gürültü gidermede yalnızca medyan filtre kullanılmışlardır ve RGB görüntü yerine gri tonlu görüntü üzerinde işlem yapılmıştır.

Literatürdeki medyan filtre çalışmalarından bazılarında, gri tonlu görüntüler üzerinde medyan filtre tasarımı [24, 45] yapılmıştır. Bisht ve Vijay [24], tasarımında minimum gecikmeyi 2.41 ns olarak gerçekleştirmişlerdir ve çalışmalarının gerçek kamera arayüzüne uyarlanabileceğini sonuçlarda belirtmişlerdir. Mevcut tez çalışmasında, minimum gecikmenin 0.012 ns olarak tespit edilmesi bulgular kısmında gösterilmiştir.

Literatürdeki diğer medyan filtre çalışmalarında [46, 47, 48]; FPGA tabanlı, yalnızca simülasyon olarak medyan filtre tasarımı yapılmıştır.

Wang ve Zhu [49], FPGA tabanlı gerçek zamanlı gauss filtreleme ile ilgili gri tonlamalı görüntü üzerinde, işlemci kullanarak tasarımı gerçekleştirmişlerdir. Mevcut tez çalışmasında, işlemci kullanılmadan RGB görüntü üzerinde filtrelemeler gerçekleştirilmiştir.

6. SONUÇ VE ÖNERİLER

Baskı devre kartlarında bulunan eksik komponent hatalarının gerçek zamanlı olarak tespit edilerek kullanıcıya monitörde gösterimini sağlayan, FPGA tabanlı ve VHDL ile kodlanarak geliştirilmiş donanımsal sistem tasarımı gerçekleştirilmiştir. Eksik komponent hatasının tespitinden önce, görüntüde bulunan gürültü bileşenleri sebebiyle görüntünün ön işlemden geçmesi gerekmektedir. Görüntü iyileştirmede kullanılan, gürültülerin giderilmesini sağlayan medyan ve gauss filtreleme uygulamaları, önce simülasyonda test edilmiş ardından bu filtrelemeler kaskadlanarak gerçek zamanlı olarak uygulanmıştır.

Bilgisayar ortamında Vivado programı kullanılarak görüntü alımı, medyan filtreleme, gauss filtreleme ve hata tespiti için yazılan VHDL kodları, bilgisayarın USB bağlantısı ile ZedBoard FPGA içerisine gömülür. OV7670 görüntü sensöründen 30 fps hızında alınan gerçek zamanlı RGB444 formatında 320x240 boyutuna sahip görüntü, VGA aracılığı ile 640x480 boyutundaki 60 Hz ile çalışan monitöre yansıtılır.

Eksik komponent hata tespiti için PCB üzerinde bulunan komponentlere ait lehimli yerler, monitör üzerinde kırmızı renkte gösterilmiştir. Bunun için RGB444 formatından HSV değerleri elde edilmiştir. Elde edilen HSV değerlerinden lehim rengi aralığı belirlenmiştir. Bu lehim rengi aralığında kalan pikseller için RGB'nin kırmızı kanalı atanarak, diğer kısımlar için RGB olarak alınan piksellerin direkt olarak monitörde gösterilmesi sağlanmıştır.

ZedBoard kartı üzerinde bulunan anahtarların değiştirilmesi ile kullanıcının isteğine bağlı olarak medyan ve gauss filtreleme seçimi, filtreleme sayısı ve hata tespiti yapılan görüntünün monitörde gösterimi sağlanır.

Tez çalışması, hazır şekilde sunulan araçlar kullanılmadan saf donanım olarak tamamen VHDL ile kodlanarak Vivado programında oluşturulmuştur. Sistemin FPGA tabanlı, donanımsal olması sayesinde paralel bir şekilde tüm pikseller bir saat sinyali ile aynı anda işlenmektedir. Böylece VHDL ile kodlanarak saf donanım oluşturulması; yazılım veya donanım / yazılım ortak tasarımının uygulamasından daha iyi işlem hızına sahip performans sunmaktadır.

Tasarımda, ZedBoard FPGA geliştirme kartının üzerinde bulunan Zynq-7000 işlemcisi kullanılmamıştır. Bu işlemcinin kullanılmaması sayesinde yazılan VHDL kodları, marka ve model fark etmeden tüm FPGA geliştirme kartlarında kullanılabilir.

Donanımsal sistem, görüntünün bir çerçevesini saklayan çerçeve arabelleği olmadan çalışabilmektedir. Çerçeve arabelleklerinin kullanılmasının tek nedeni, 30 fps hızında görüntü alımı yapan kameranın ve 60 Hz destekleyen VGA'nın, farklı hızlarda çalışmasıdır. Bu sebeple donanımsal sistemin hız açısından düzeltilmesinde çift çerçeve arabelleği kullanılmıştır. Ayrıca mevcut tasarımda yer alan iki adet çerçeve arabelleği sayesinde sistem, farklı hıza sahip VGA ve kamera için çalışabilmektedir.

Tez çalışmasında, PCB üzerinde bulunan komponentlerin renklendirilerek ekranda gösterilmesi sağlanabilir. Fakat, RGB444 formatında monitörde gösterilen görüntüde kart renginin belirli kısımlarının koyu yeşil olması sebebiyle siyah komponentler ile kart rengi arasında renk ayrımı yapılamamaktadır. 24 bit çözünürlüğü destekleyen bir kameradan RGB888 formatında görüntü alınıp işlenirse, bu formatın 16 milyon renk içermesi sayesinde koyu yeşil ile siyah renk ayrımı yapılabilmektedir. RGB444 formatına geçildiğinde bu renk sayısı 4096'ya düşmektedir. Format farkından oluşan renk kaybının gösterimi için yüksek çözünürlüklü kamera ile görüntü alınarak gerçek zamanlı olmayan biçimde MATLAB ortamında deneme yapılmıştır. Bu görüntüde lehimli alanların kırmızı renk ile gösterilmesinde daha iyi sonuç alınmıştır. Tez çalışmasında, RGB444 formatı kullanılmasının nedeni, donanımsal olarak kaynakların yetersiz olmasıdır. Örneğin, OV7670 görüntü sensörünün sağladığı renk çözünürlüğü, maksimum 8 bit ve ZedBoard VGA'nın renk çözünürlüğü, 12 bittir. RGB888 formatını destekleyen yüksek çözünürlüklü kamera kullanılırsa mevcut komponentler PCB üzerinde renklendirilerek gösterilir ve eksik komponent çıkarımı yapılabilir.

İleride gerçekleştirilecek çalışmalarda çeşitli lehim hatalarının tespit edilmesi [50], derin öğrenme algoritmaları kullanılarak [51] tüm komponent hatalarının gerçek zamanlı olarak RGB formatında tespit edilmesi sağlanabilir. Önerilen çalışmaların uygulanabilmesi için görüntüdeki gürültülerin kaldırılarak ön işlemden geçirilmesi, PCB üzerindeki komponentlerin (IC, SMD direnci, transistör, diyot vb.) yerinin belirlenmesi [52] gerekmektedir. Mevcut tez çalışması, görüntüyü filtreleyerek ön işlemden geçirdiği için baskı devre kartı görüntüleri dışında tüm görüntü işleme uygulamalarında kullanılabilir. Nesne tanıma işlemlerinin yer aldığı yüksek seviyeli görüntü işleme içeren çalışmalara uyarlanabilir.

KAYNAKLAR

[1] Vinita, Kaushik, S., 2016, PCB Fault Detection by Image Subtraction Method, International Journal of Scientific Engineering and Applied Science (IJSEAS), 2(1), 147-152.

[2] Liu, Z., Qu, B., 2021, Machine vision based online detection of PCB defect, ScienceDirect, Vol. 82.

[3] Kasturkar, A., Lokhande, S. D., 2016, PCB Fault Detection by Image Processing Tools : A Review, International Journal of Innovative Research in Science, Vol. 5, Issue 6.

[4] Hashim, H.S., Abdullah, S.N.H.S., Prabuwono, 2010, A.S.: Automated visual inspection for metal parts based on morphology and fuzzy rules. In: Proc. ICCAIE, pp. 527–531.

[5] Akbar, H., Prabuwono, A.S., 2008, Webcam based system for press part industrial inspection., *IJCSNS*, 8(10), 170–177.

[6] Aghamohammadi, A., Prabuwono, A.S., Sahran, S., Mogharrebi, M., 2011, Solar cell panel crack detection using particle swarm optimization algorithm., In: Proc. ICPAIR, pp. 160–164.

[7] Anoop, K. P, Sarath, N.S, Sasi Kumar, V. V., 2015, A Review of PCB Defect Detection Using Image Processing, *International Journal of Engineering and Innovative Technology (IJEIT)*, Vol. 4, Issue 11.

[8] Ortega, J., Mejía, L.F., Santoyo, A., 2007, PCB Inspection System using Image Processing and Wavelet Transform, *6th Mexican international conference on Advances in artificial intelligence.*

[9] Padnos, G., 2011, Improved efficiency using root cause failure analysis., *Global SMT and Packaging*, 11(6), 14.

[10] Çavuşlu, M. A., Kösten, M. M., 2017, VHDL İle Sayısal Tasarım ve FPGA Uygulamaları, 2. Baskı, İnkılap Kitabevi Yayın San. Tic. A.Ş., İstanbul, ISBN: 978-605-9118-46-0.

[11] Rodríguez, M. A. V., Pérez, J. M. S., Pulido, J. A. G., 2002, An FPGA-based Implementation For Median Filter Meeting The Real-Time Requirements Of Automated Visual Inspection Systems, 10th Mediterranean Conference on Control and Automation - MED2002, Lisbon, Portugal, July 9-12.

[12] Maxfield, C., 2011, FPGAs: Instant Access, Elsevier, 1-12.

[13] Moore, A., Wilson, R., 2017, FPGAs For Dummies, 2nd Intel Special Edition, John Wiley & Sons, Inc., Hoboken, New Jersey, ISBN: 978-1-119-39047-3.

[14] Sarıtaş, E., Karataş, S., 2013, "Her Yönüyle FPGA ve VHDL", Palme Yayıncılık, Ankara, ISBN: 978-605-355-129-4.

[15] Jahanirad, H., Karam, H., 2017, BIST-based Testing and Diagnosis of LUTs in SRAM-based FPGAs, Emerging Science Journal, 1(4), 216-225.

[16] Cofer, R. C., Harding, B. F., 2006, Rapid System Prototyping with FPGAs: Accelerating the Design Process, Elsevier.

[17] Xu, M., 2020, Image processing system based on FPGA and convolutional neural network, Microprocessors and Microsystems, Elsevier, 103379.

[18] Dejazzer, Digital Camera: OV7670 CMOS camera + DE2-115 FPGA board + LandTiger 2.0 board, http://www.dejazzer.com/eigenpi/digital_camera/digital_camera.html , [Ziyaret tarihi: 24 Ağustos 2021].

[19] Gonzalez, R. C., Woods, R. E., 2002, Digital Image Processing, Prentice Hall, ISBN: 0-201-180758.

[20] RapidTables, RGB to HSV color conversion, https://www.rapidtables.com/convert/color/rgb-to-hsv.html, [Ziyaret tarihi: 24 Ağustos 2021].

[21] Scan Tips, 2021, "Understanding File Types, Bit Depth, & Memory Cost of Images", https://www.scantips.com/basics1d.html , [Ziyaret tarihi: 26 Ağustos 2021].

[22] Zhang, Y., Zhou, Z., Zheng, L., Zhu, Z., Xiao, P., 2019, Visual Recognition and Localization of Printed Circuit Board, Journal of Physics: Conference Series, 2019

International Conference on Advanced Information Systems and Engineering, 23–25 Aug., Cairo, Egypt, Vol. 1454.

[23] Appiah, O., Asante, M., Hayfron-Acquah, J. B., 2020, Improved approximated median filter algorithm for real-time computer vision applications, Journal of King Saud University-Computer and Information Sciences.

[24] Bisht, R., Vijay, R., 2017, Hardware implementation of real time window based switching median filter. *International Journal Of Engineering Sciences & Research Technology*, 6(7), 732-740.

[25] Fiveko, 2017, "Median Filter – edge preserving filter tutorial", https://fiveko.com/median-filter/ , [Ziyaret tarihi: 27 Ağustos 2021].

[26] İbrahim Çayıroğlu, Dersler, Görüntü İşleme, "5.Hafta: Bulanıklaştırma (yumuşatma) algoritmaları", http://www.ibrahimcayiroglu.com/ , [Ziyaret tarihi: 2 Ekim 2022].

[27] Shipitko, O. S., Grigoryev, A. S., 2018, Gaussian filtering for FPGA based image processing with High-Level Synthesis tools. In Proceedings of the IV International Conference on Information Technology and Nanotechnology, Sarma, Russia (pp. 24-27).

[28] Goel, N., Kaur, S., Kumar, Y., 2022, Machine learning-based remote monitoring and predictive analytics system for crop and livestock, In AI, Edge and IoT-based Smart Agriculture, Academic Press, Elsevier, pp. 395-407,

[29] TinyVGA, VGA Signal 640 x 480 @ 60 Hz Industry standard timing, http://tinyvga.com/vga-timing/640x480@60Hz, [Ziyaret tarihi: 28 Ekim 2021].

[30] NANDLAND, Go Board – VGA Introduction, https://www.nandland.com/goboard/vga-introduction-test-patterns.html , [Ziyaret tarihi: 31 Ekim 2021].

[31] MIT, OV7670 Datasheet, http://web.mit.edu/6.111/www/f2016/tools/OV7670_2006.pdf , [Ziyaret tarihi: 31 Ekim 2021].

[32] Maraşlı, F., Öztürk, S., 2018, Görüntü İyileştirme ve Görüntü Onarma Teknikleri ile Yapılmış Uygulamalar, II. Uluslararası Bilimsel ve Mesleki Çalışmalar Kongresi – BILMES 2018, October.

[33] Digilent, ZedBoard Zynq-7000 Development Board Reference Manual, https://digilent.com/reference/programmable-logic/zedboard/reference-manual, [Ziyaret tarihi: 10 Eylül 2021].

[34] FPGAKey, ZedBoard System Architecture, https://www.fpgakey.com/tutorial/section365 , [Ziyaret tarihi: 8 Eylül 2021].

[35] Xilinx, 2021, "SoCs with Hardware and Software Programmability: Zynq-7000 SoC Product Advantages", https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html, [Ziyaret tarihi: 30 Ekim 2021].

[36] Design & Reuse, 2012, https://www.design-reuse.com/news/31035/xilinx-vivado-design-suite-webpack-edition.html, [Ziyaret tarihi: 30 Ekim 2021].

[37] Embedded Thoughts, 2016, Driving a VGA Monitor Using an FPGA, https://embeddedthoughts.com/2016/07/29/driving-a-vga-monitor-using-an-fpga/, [Ziyaret tarihi: 1 Kasım 2021].

[38] Nayak, P. R., Anitha, K., Parameshachari, B. D., Banu, R., Rashmi, P., 2017, PCB Fault Detection Using Image Processing, IOP Conf. Series (ICMAEM), 225.

[39] Sunil, S. S., Ambadas, W. S., Gopinath, S. S., 2019, Chinchole, M.G., PCB Fault Detection using image processing in Matlab, IOSR Journal of Engineering (IOSR JEN), 61-65.

[40] Shinde, N. K., Morade, S, 2015, PCB Inspection System using Image Processing, International Journal of Science, Engineering and Technology Research (IJSETR), 4(4), 1009-1012.

[41] Vinita, Kaushik, S., 2016, PCB Fault Detection by Image Subtraction Method, International Journal of Scientific Engineering and Applied Science (IJSEAS), 2(1), 147-152.

[42] Ortega, J., Mejía, L.F., Santoyo, A., 2007, PCB Inspection System using Image Processing and Wavelet Transform, *6th Mexican international conference on Advances in artificial intelligence.*

[43] Chavan, R. R., Chavan, S. A., Dokhe, G. D., Wagh, M. B., Vaidya, A. S., 2016, Quality Control of PCB using Image Processing, International Journal of Computer Applications (0975 – 8887), Vol. 141, No 5.

[44] Sushma, M., Kumar, S.V.S., 2019, Execution of Median Filter Built on FPGA for Trimming Noise Meeting the Real Time Requirements, *International Journal of Recent Technology and Engineering (IJRTE)* , ISSN: 2277-3878, Vol. 8, Issue 4.

[45] Wang, B., Xiang, Q, 2020, Fast median filter image processing algorithm and its FPGA implementation. *Front Signal Process*, *4*(4), 88-94.

[46] Bevara, V., Sanki, P. K., 2020, A new fast and efficient 2-D median filter architecture. *Sādhanā*, *45*(1), 1-5.

[47] Garg, L., Garg, A., Kumar, A., 2015, FPGA based design of median filter to reduce noise in imaging, *International Journal of Electrical and Electronics Engineers (IJEEE)*, Vol. 07, Issue 01.

[48] Gang, L., Yitao, L., Feng, W., Zhenzhen, Y., 2012, The Implementation and Analysis of Fast Median Filter Based on FPGA., In *Advances in Electric and Electronics*, pp. 447-451., Springer, Berlin, Heidelberg.

[49] Wang, C., Zhu, S., 2015, A design of fpga-based system for image processing, *Journal of simulation*, *3*(1), 7.

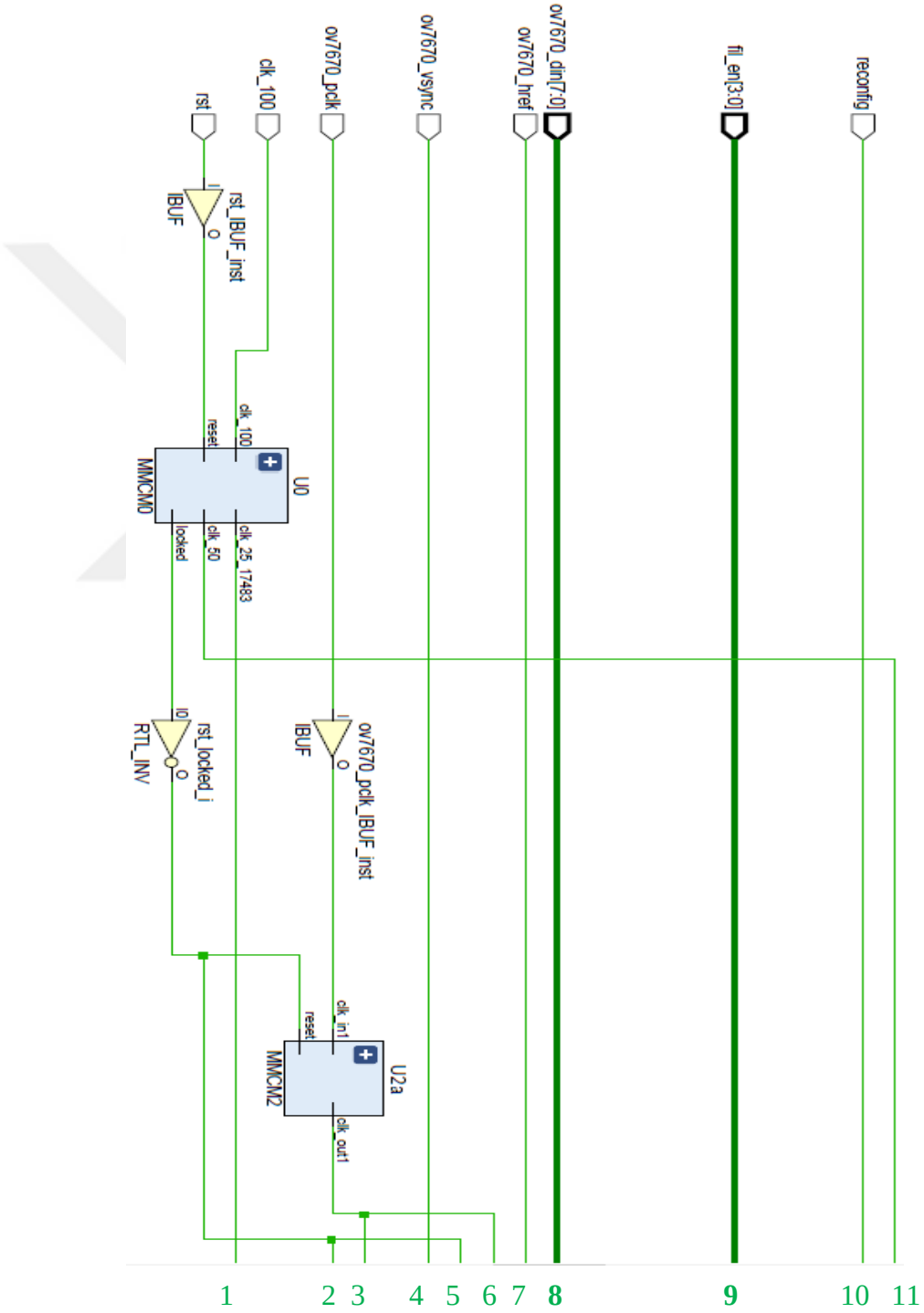
[50] Pennachia, D.L., Sanchez-Martinez, M.A., 2007, A FPGA Implementation of Solder Paste Deposit on Printed Circuit Boards Error Detector Based in a Bright and Contrast Algorithm, Journal of Applied Research and Technology, 5(2), 89-102.

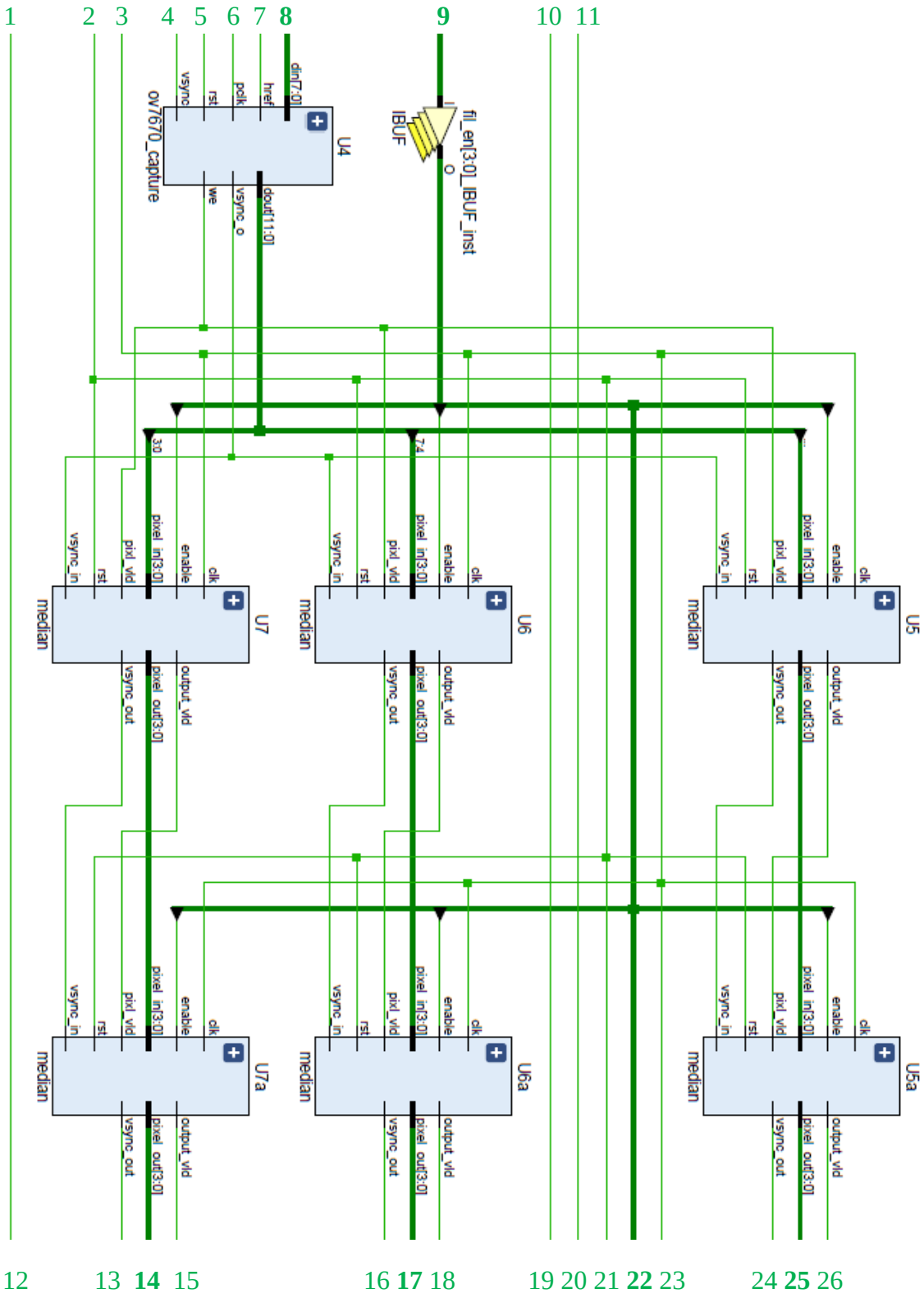
[51] Adibhatla, V. A., Chih, H. C., Hsu, C. C., Cheng J., Maysam, F. A., Shieh, J. S., 2020, Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural Networks, MDPI electronics, 9(9), 1547.

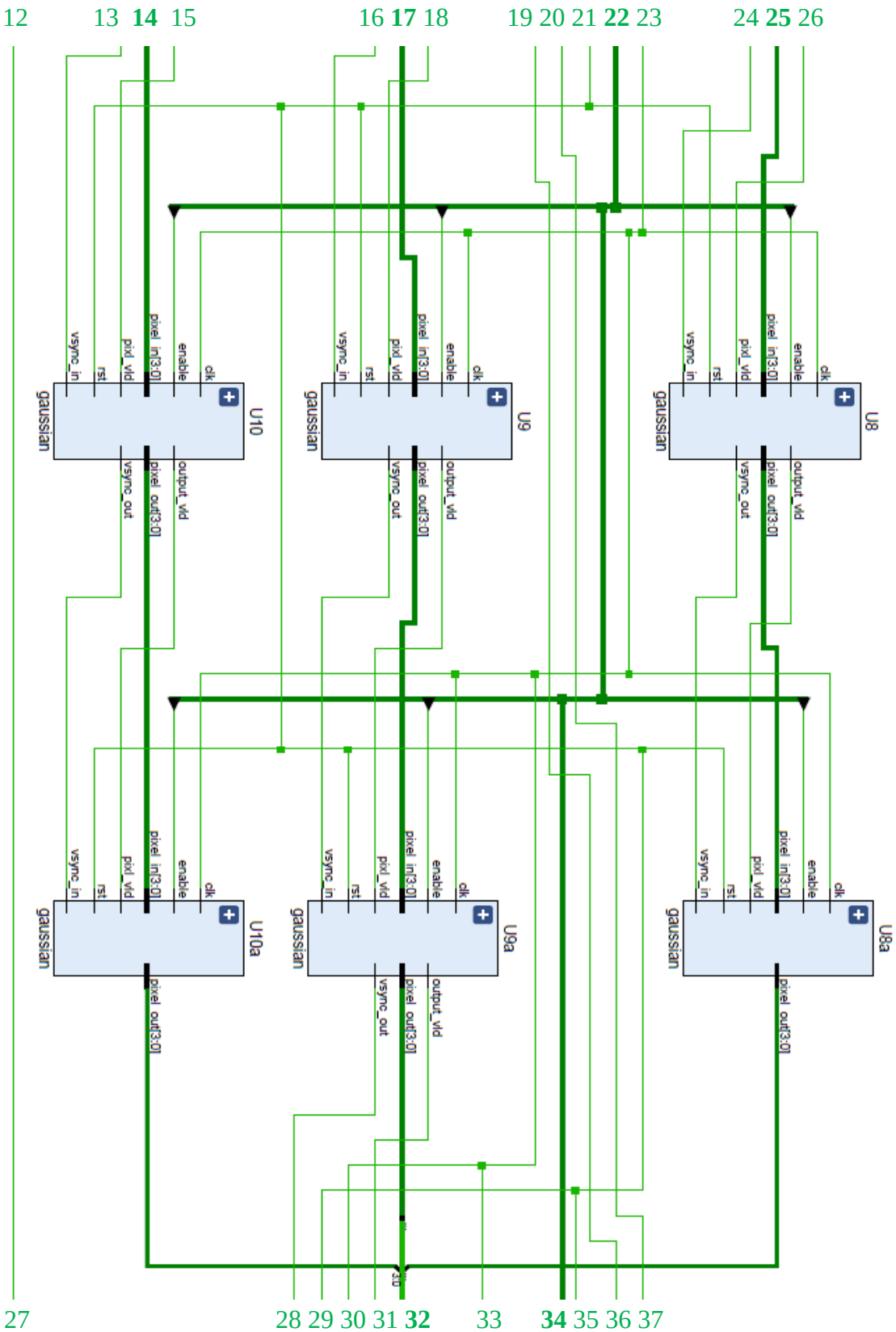
[52] Zeng, Z., Ma, L., Zheng, Z., 2010, Extracting PCB Components Based on Color Distribution of Highlight Areas, ComSIS, Vol. 7, No. 1.

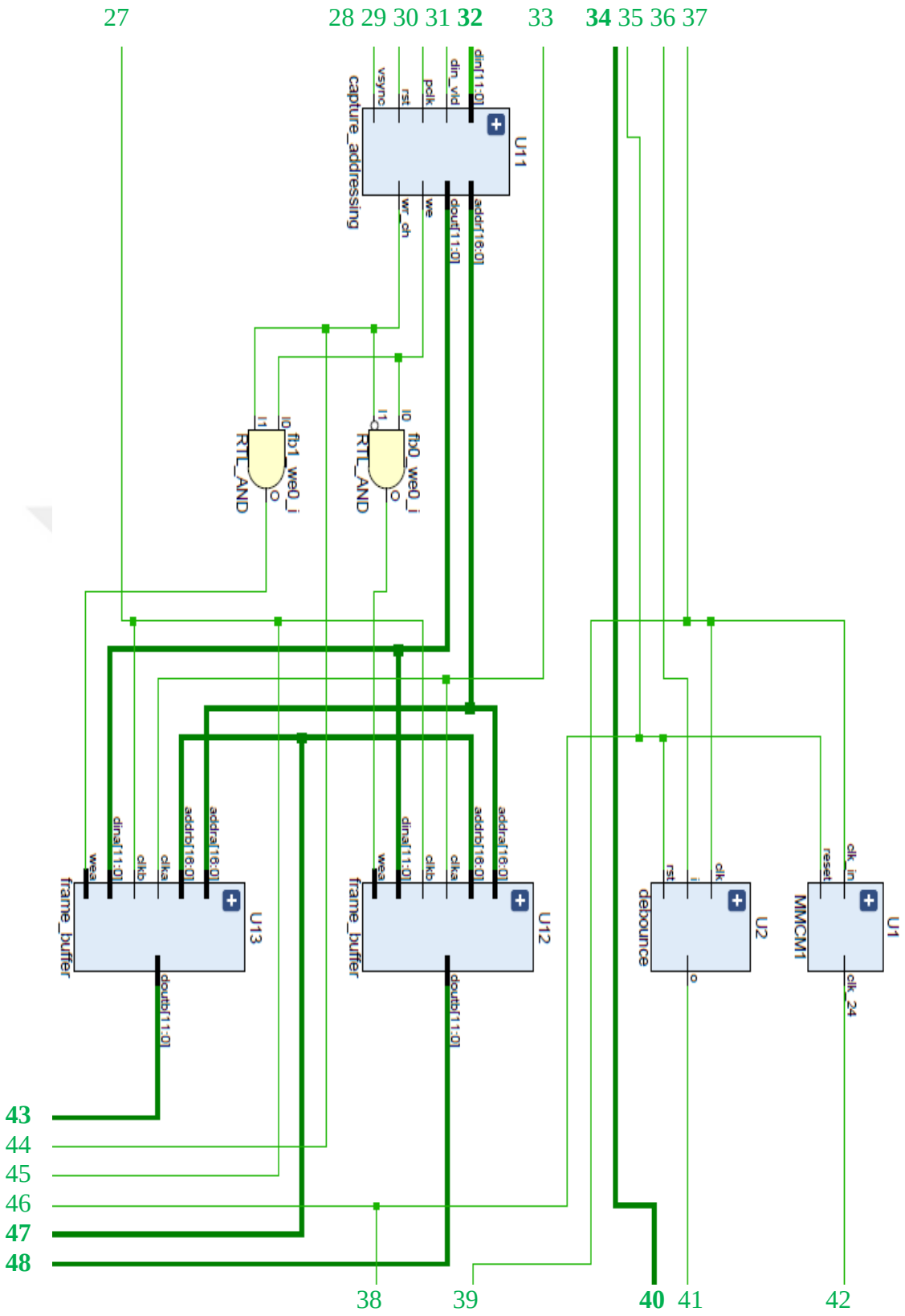
EKLER

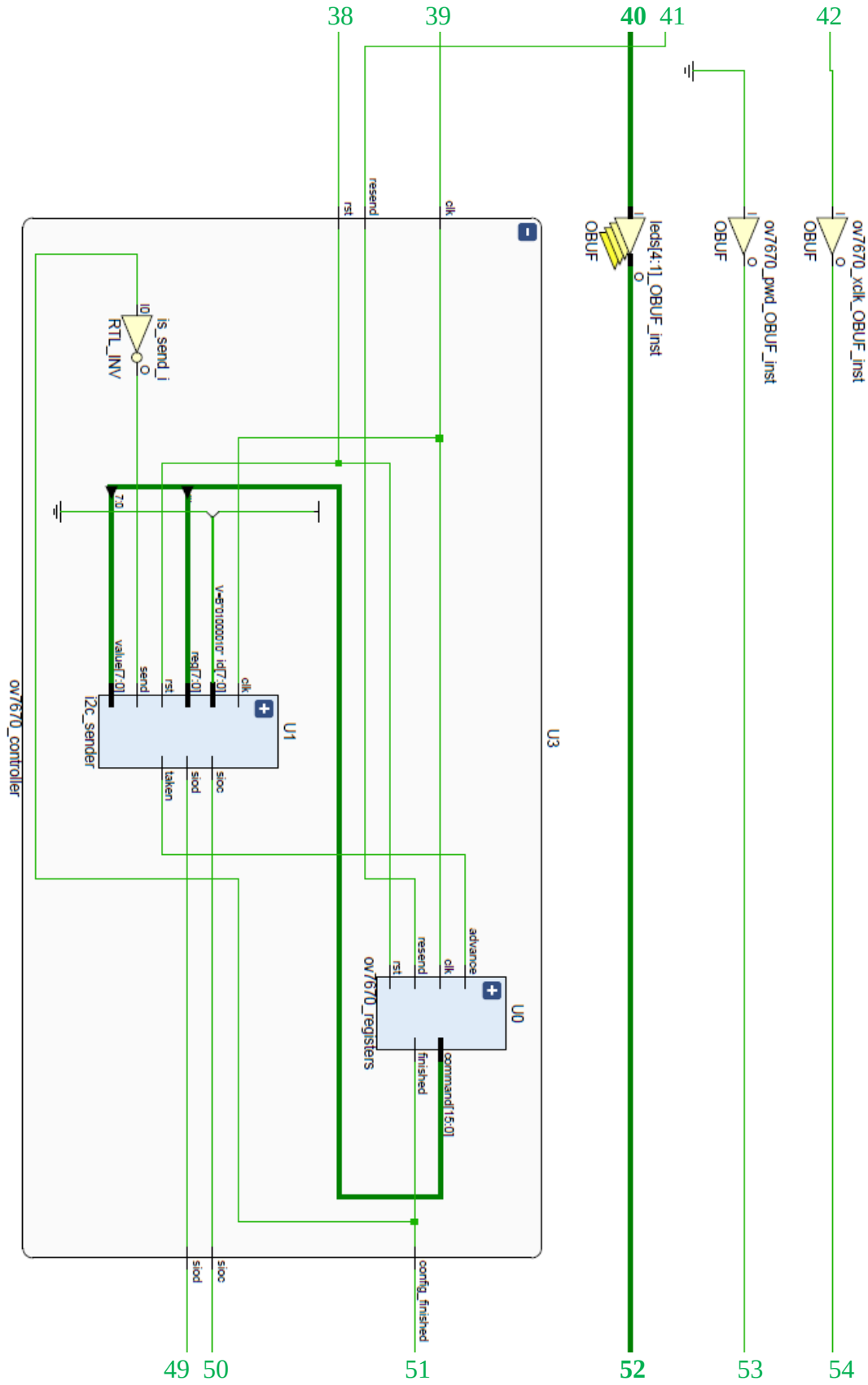
EK 1. VHDL kodlarının RTL şematik gösterimi.

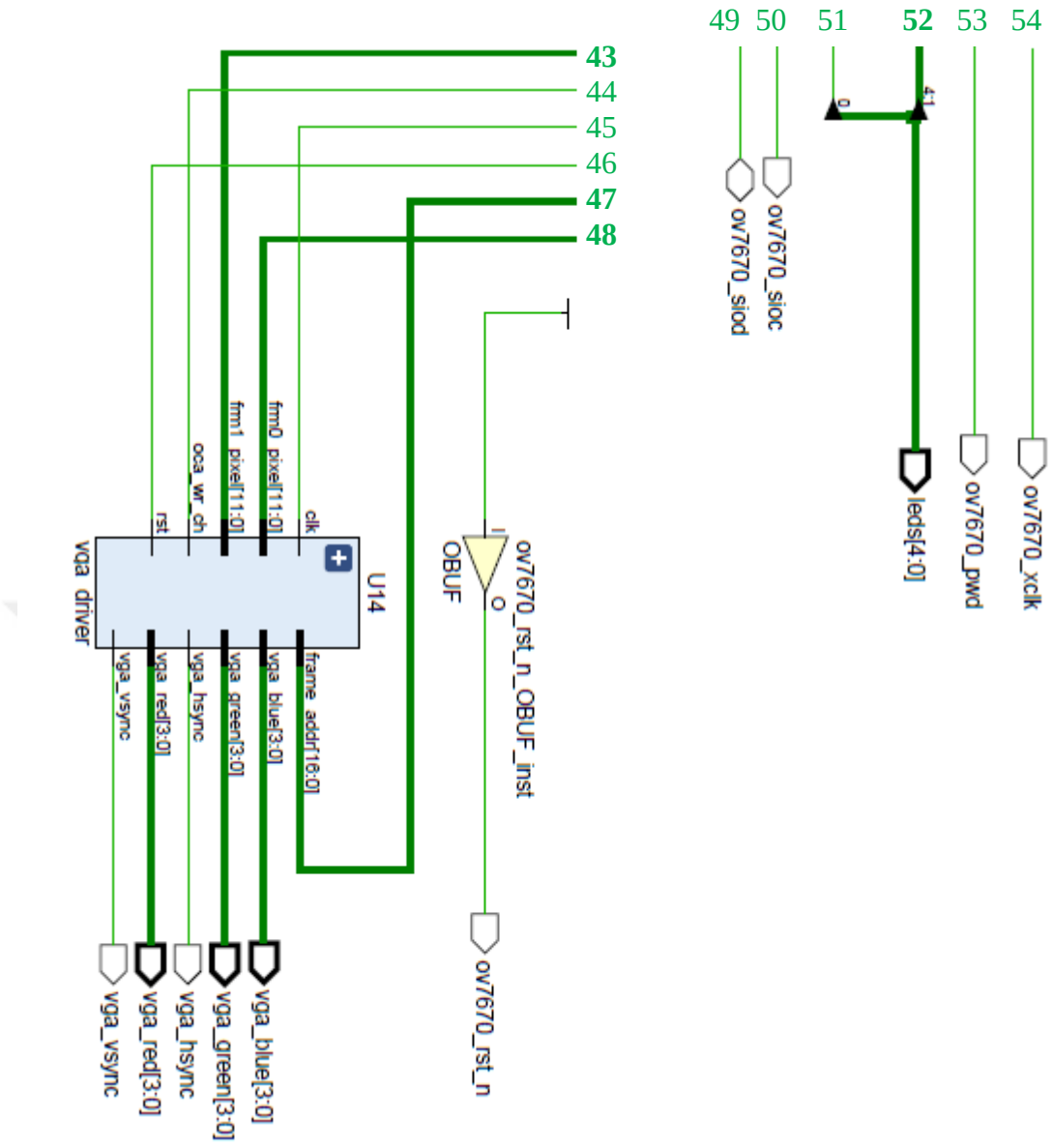



1 2 3 4 5 6 7 8 9 10 11
ov7670_capture
U4
vsync
rst
pclk
href
din[7:0]
we
vsync_o
dout[11:0]
IBUF
fil_en[3:0]_IBUF_inst
I
O
median
U7
U6
U5
U7a
U6a
U5a
vsync_in
pixl_vid
enable
clk
pixel_in[3:0]
vsync_out
output_vid
pixel_out[3:0]
3:0
7:4
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

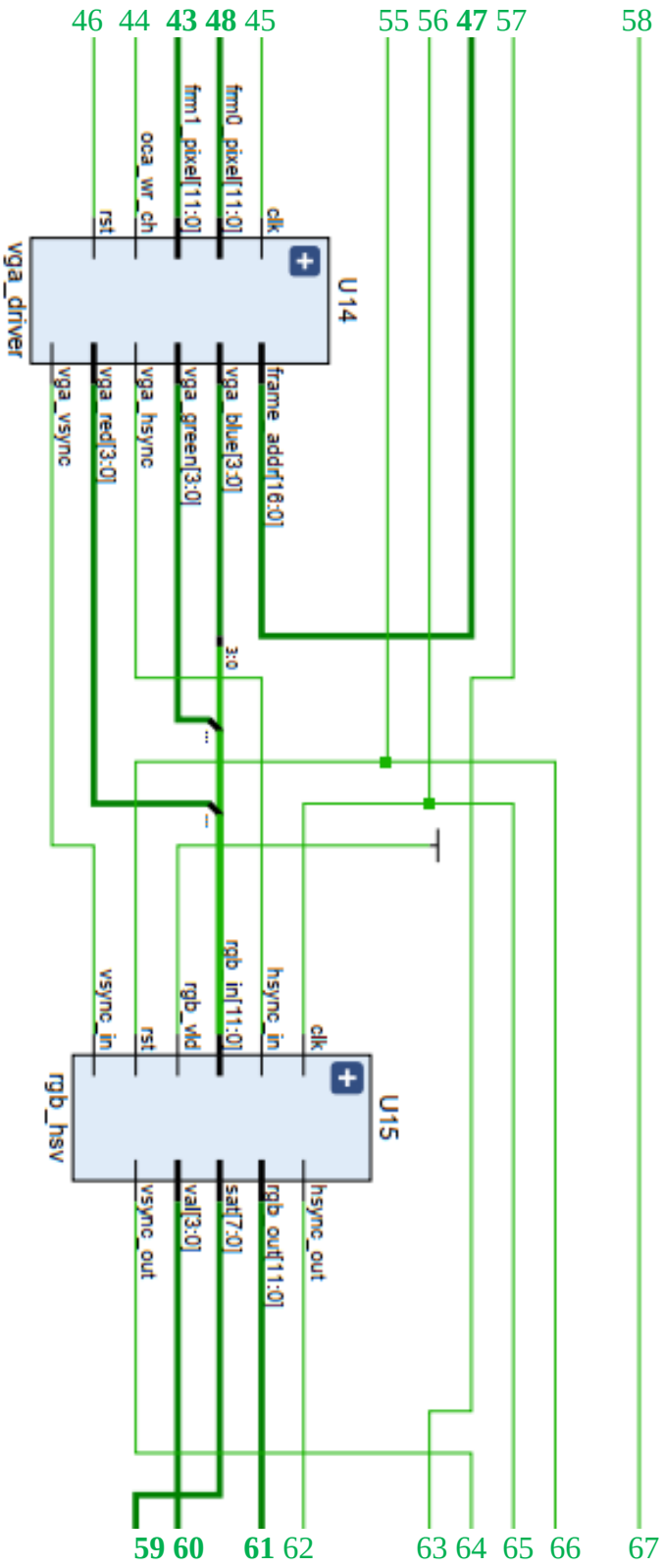

12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
gaussian
U10
U9
U8
U10a
U9a
U8a
vsync_in
rst
pixl_vld
pixel_in[3:0]
enable
clk
vsync_out
output_vld
pixel_out[3:0]
27 28 29 30 31 32 33 34 35 36 37

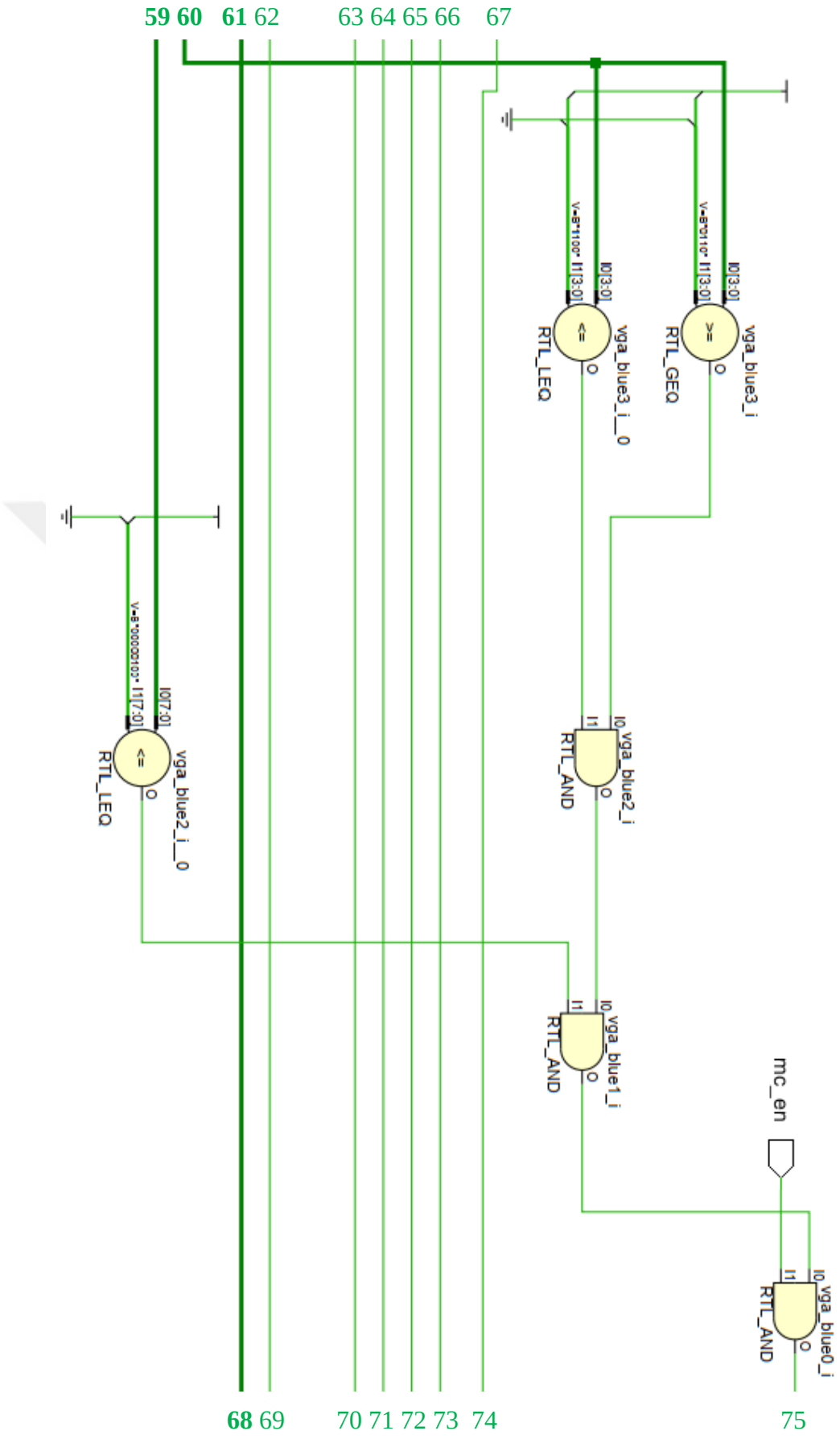

27
28 29 30 31 32
33
34 35 36 37
capture_addressing
U11
din[11:0]
din_vld
pclk
rst
vsync
wr_ch
we
douf[11:0]
addr[16:0]
RTL_AND
fb1_we0_i
fb0_we0_i
I0
I1
O
frame_buffer
U13
U12
wea
dina[11:0]
clkb
clka
addrb[16:0]
addra[16:0]
doutb[11:0]
debounce
U2
clk
o
MMCM1
U1
clk_in
reset
clk_24
43
44
45
46
47
48
38
39
40
41
42

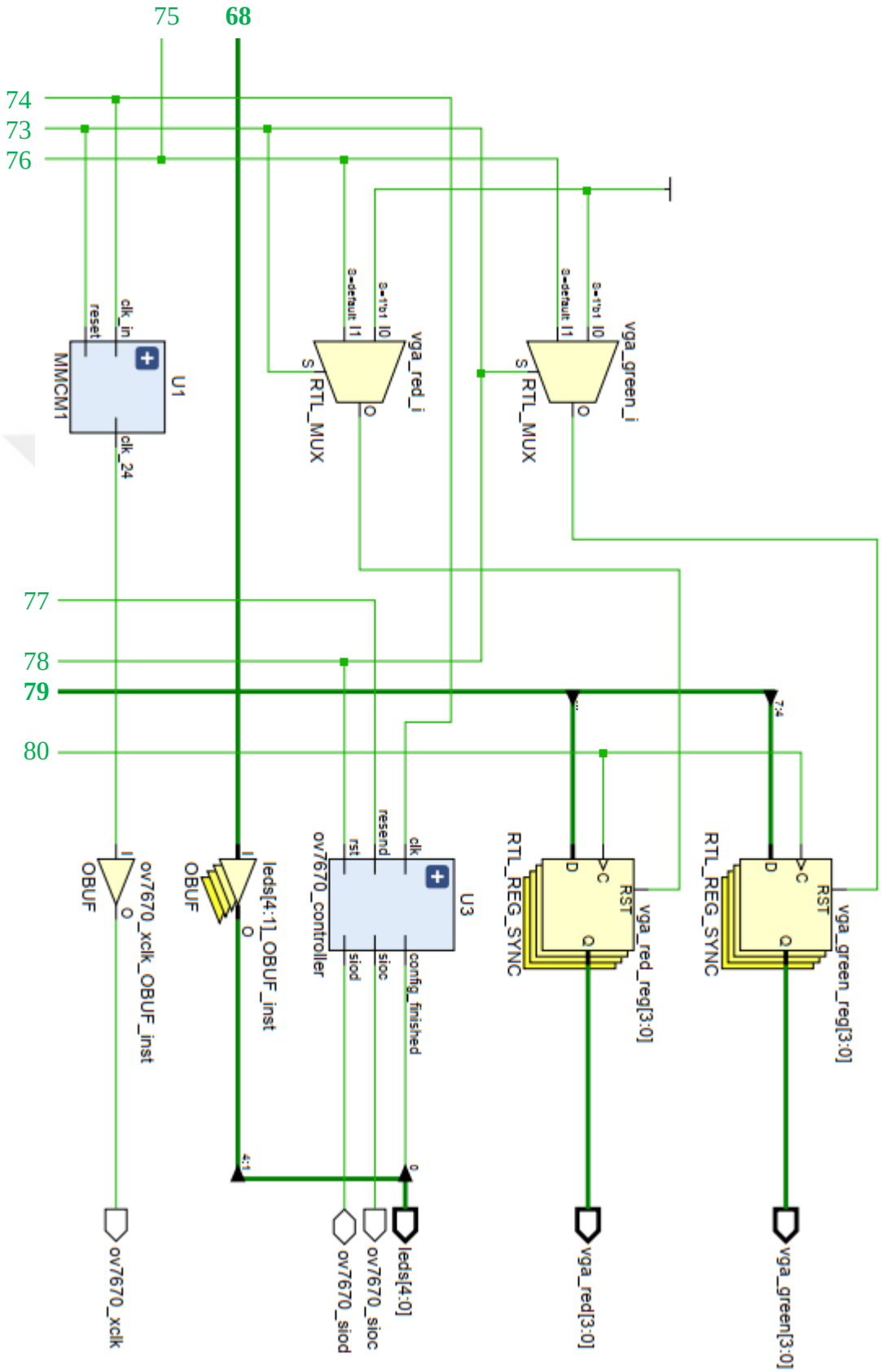

38
39
40
41
42
resend
clk
rst
leds[4:1]_OBUF_inst
OBUF
ov7670_pwd_OBUF_inst
OBUF
ov7670_xclk_OBUF_inst
OBUF
is_send_i
RTL_INV
I0
O
7:0
value[7:0]
send
rst
reg[7:0]
clk
i2c_sender
U1
taken
siod
sioc
ov7670_controller
U3
advance
resend
rst
clk
ov7670_registers
U0
finished
command[15:0]
siod
sioc
config_finished
49
50
51
52
53
54

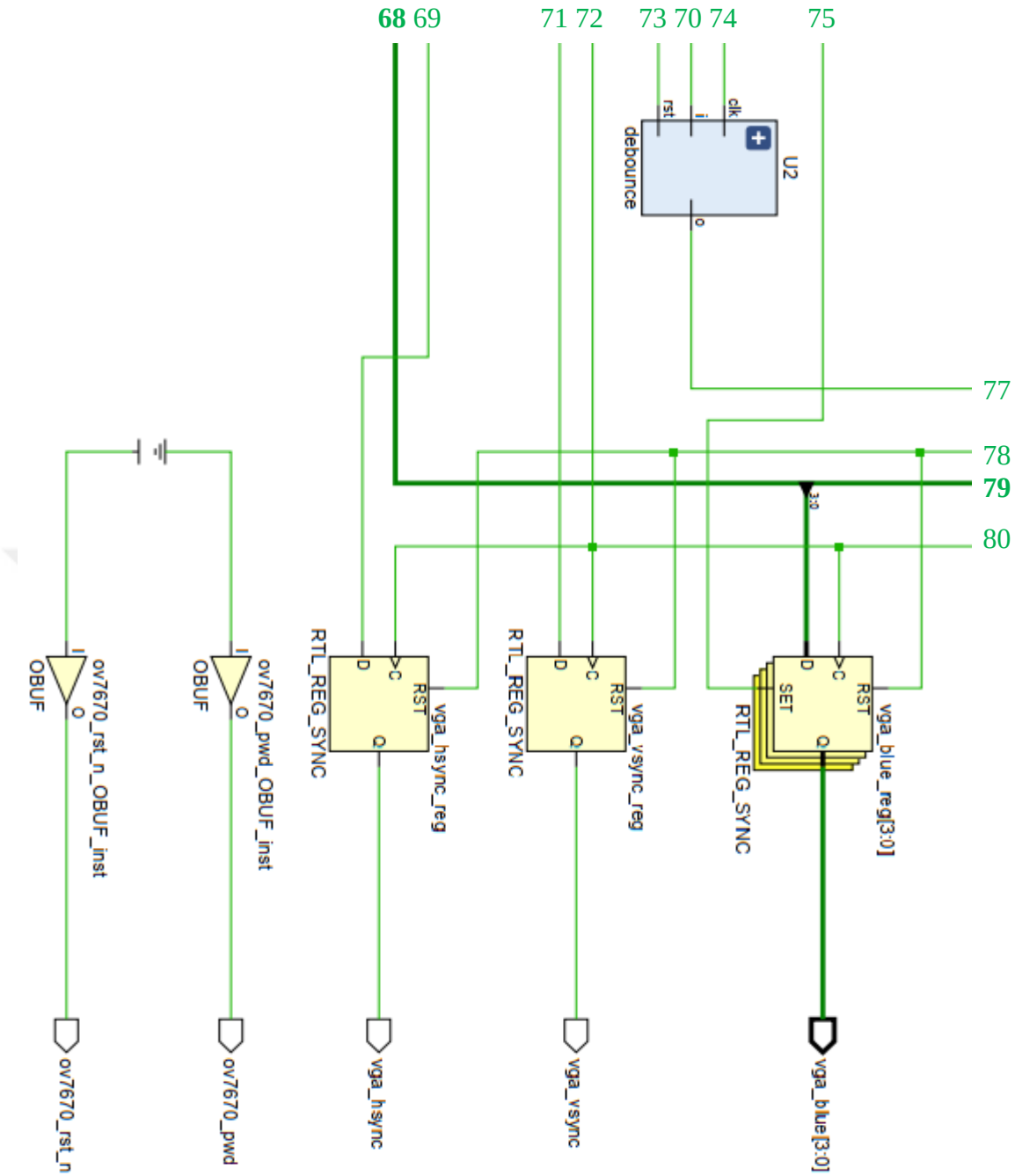

49
50
51
52
53
54
43
44
45
46
47
48
ov7670_siod
ov7670_sioc
leds[4:0]
ov7670_pwd
ov7670_xclk
4:1
0
vga driver
U14
rst
oca_wr_ch
frm1_pixel[11:0]
frm0_pixel[11:0]
clk
frame_addr[16:0]
vga_vsync
vga_red[3:0]
vga_hsync
vga_green[3:0]
vga_blue[3:0]
OBUF
I
O
ov7670_rst_n_OBUF_inst
ov7670_rst_n
vga_vsync
vga_red[3:0]
vga_hsync
vga_green[3:0]
vga_blue[3:0]

EK 2. HSV modülü eklenmiş VHDL kodlarının RTL şematik gösterimi.




59 60 61 62 63 64 65 66 67
I[3:0]
V="1100" I[3:0]
<=
RTL_LEQ
vga_blue3_i_0
I[3:0]
V="0110" I[3:0]
>=
RTL_GEQ
vga_blue3_i
I[7:0]
V="00000100" I[7:0]
<=
RTL_LEQ
vga_blue2_i_0
I1
I0
RTL_AND
vga_blue2_i
I1
I0
RTL_AND
vga_blue1_i
mc_en
I1
I0
RTL_AND
vga_blue0_i
68 69 70 71 72 73 74 75


75
68
74
73
76
77
78
79
80
reset
clk_in
MMCM1
U1
clk_24
S
RTL_MUX
vga_red_i
vga_green_i
O
OBUF
ov7670_xclk_OBUF_inst
ov7670_xclk
leds[4:1]_OBUF_inst
ov7670_controller
resend
rst
clk
siod
sioc
config_finished
U3
ov7670_siod
ov7670_sioc
leds[4:0]
RTL_REG_SYNC
D
C
RST
Q
vga_red_reg[3:0]
vga_green_reg[3:0]
vga_red[3:0]
vga_green[3:0]


68 69
71 72
73 70 74
75
77
78
79
80
rst
i
clk
debounce
U2
o
OBUF
ov7670_rst_n_OBUF_inst
ov7670_rst_n
OBUF
ov7670_pwd_OBUF_inst
ov7670_pwd
RTL_REG_SYNC
D
C
RST
Q
vga_hsync_reg
vga_hsync
RTL_REG_SYNC
vga_vsync_reg
vga_vsync
RTL_REG_SYNC
SET
vga_blue_reg[3:0]
vga_blue[3:0]

İNTİHAL RAPORU İLK SAYFASI

Merve Aydın

ORİJİNALLİK RAPORU

%3	%2	%0	%2
BENZERLİK ENDEKSİ	İNTERNET KAYNAKLARI	YAYINLAR	ÖĞRENCİ ÖDEVLERİ

BİRİNCİL KAYNAKLAR

1	Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) Öğrenci Ödevi	%1
2	Submitted to Istanbul University Öğrenci Ödevi	<%1
3	acikbilim.yok.gov.tr İnternet Kaynağı	<%1
4	nek.istanbul.edu.tr:4444 İnternet Kaynağı	<%1
5	edoc.pub İnternet Kaynağı	<%1
6	hdl.handle.net İnternet Kaynağı	<%1
7	earsiv.etu.edu.tr İnternet Kaynağı	<%1
8	elektronikhobi.net İnternet Kaynağı	<%1
9	inis.jinr.ru İnternet Kaynağı	

10	**Submitted to University of the West Indies** Öğrenci Ödevi	<%1
11	**www.datasheet.hk** İnternet Kaynağı	<%1
12	**fulla.fnal.gov** İnternet Kaynağı	<%1
13	**wiki.sj.ifsc.edu.br** İnternet Kaynağı	<%1
14	**openaccess.firat.edu.tr** İnternet Kaynağı	<%1
15	**www.toddtimber.se** İnternet Kaynağı	<%1
16	**Mauricio Rêgo Mota da Rocha. "Geração de testes a partir de máquinas de estados finitos estendidas extraídas de diagramas de sequência UML", Universidade de Sao Paulo, Agencia USP de Gestao da Informacao Academica (AGUIA), 2020** Yayın	<%1