KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

TEMEL GÖRÜNTÜ İŞLEME ALGORİTMALARININ GERÇEK ZAMANLI OLARAK FPGA İLE GERÇEKLENMESİ

MEHMET ALİ ALTUNCU

KOCAELİ 2015

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

TEMEL GÖRÜNTÜ İŞLEME ALGORİTMALARININ GERÇEK ZAMANLI OLARAK FPGA İLE GERÇEKLENMESİ

MEHMET ALİ ALTUNCU

Prof.Dr. Yaşar BECERİKLİ
Danışman, Kocaeli Üniv. .....

Yrd. Doç. Dr. Suhap ŞAHİN
Jüri Üyesi, Kocaeli Üniv. .....

Prof.Dr. A. Coşkun SÖNMEZ
Jüri Üyesi, İTÜ .....

Tezin Savunulduğu Tarih: 09.01.2015

ÖNSÖZ VE TEŞEKKÜR

Bu çalışmada temel görüntü işleme algoritmalarını FPGA üzerinde gerçek zamanlı olarak gerçekleyen bir sistem tasarlanması amaçlanmıştır. Bu çalışmada, daha önce yapılan birçok uygulamadan farklı olarak işlemci ve FPGA beraber kullanılmıştır. Böylece geleneksel işlemcilerin kolaylıkla yapabileceği sıralı işlemler için FPGA kaynaklarının tüketilmemesi amaçlanmıştır.

Bu çalışmanın ortaya çıkmasında fikirleri ve tecrübesi ile yol gösteren, birikimlerini benimle paylaşan, gösterdiği sabır, anlayış, verdiği sürekli destek ve yakınlık için, danışmanım saygıdeğer hocam Prof. Dr. Yaşar BECERİKLİ'ye,

Aynı şekilde, birikimlerini benimle paylaşan, gösterdiği sabır, anlayış, verdiği sürekli destek ve yakınlık için, saygıdeğer hocam Yrd. Doç. Dr. Suhap ŞAHİN'e,

Bu çalışma süresince emek harcayan, yardımlarını esirgemeyen değerli arkadaşım, Taner GÜVEN'e,

Bu çalışma süresince desteklerini esirgemeyen ve sağladıkları huzurlu çalışma ortamından ötürü Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümündeki çalışma arkadaşlarıma,

Öğrenim hayatım boyunca gösterdikleri maddi ve manevi destekler ile bugünlere ulaşmamı sağlayan aileme teşekkür ediyorum.

Ocak – 2015 Mehmet Ali ALTUNCU

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER DİZİNİ VE KISALTMALAR

* : Konvolüsyon İşlemi
<< : Bit Düzeyinde Sola Kaydırma
>> : Bit Düzeyinde Sağa Kaydırma

Kısaltmalar

AMBA : Advanced Microcontroller Bus Architecture (Gelişmiş Mikro Denetleyici Veri Yolu Mimarisi)
ASIC : Application-Specific Integrated Circuit (Uygulamaya Özgü Entegre Devre)
AXI : Advanced eXtensible Interface (Gelişmiş Genişletilebilir Arayüz)
DDR : Double Data Rate (Çift Veri Hızı)
DMA : Direct Memory Access (Doğrudan Bellek Erişimi)
DSP : Digital Signal Processing (Dijital Sinyal İşleme)
FIFO : First In First Out (İlk Giren İlk Çıkar)
FPGA : Field Programmable Gate Array (Alanda Programlanabilir Kapı Dizileri)
FPS : Frame Per Second (Saniyedeki Çerçeve Sayısı)
GGB : Geçici Görüntü Belleği
GGBD : Geçici Görüntü Belleği Denetleyici
GİYİ : Görüntü İşleme Yardımcı İşlemcisi
HDMI : High Definition Multimedia Interface (Yüksek Çözünürlüklü Multimedya Arayüzü)
HDL : Hardware Description Language (Donanım Tanımlama Dili)
HDTV : High Definition Television (Yüksek Çözünürlüklü Televizyon)
LCD : Liquid Crystal Display (Sıvı Kristal Ekran)
LUT : Look Up Table (Bakma Tablosu)
MHZ : Megahertz
RGB : Red Green Blue (Kırmızı Yeşil Mavi)
SRAM : Static Random Access Memory (Sabit Rasgele Erişimli Bellek)
USB : Universal Serial Bus (Evrensel Seri Veriyolu)
VGA : Video Graphics Array (Video Grafikleri Dizilimi)

TEMEL GÖRÜNTÜ İŞLEME ALGORİTMALARININ GERÇEK ZAMANLI OLARAK FPGA İLE GERÇEKLENMESİ

ÖZET

Günümüzde görüntü işleme uygulamaları, teknolojinin gelişimine paralel olarak endüstriyel otomasyon, güvenlik, sağlık ve trafik denetim gibi alanlarda sıklıkla tercih edilmektedir. Bu alanlarda yapılan uygulamalardaki en büyük zorluk, uygulamanın gerçek zamanlı bir sistemde hızlı bir şekilde çalışmasını sağlamaktır. Genel amaçlı bilgisayarlardaki işlemci, bellek ve çevresel cihazların kaynak bakımından yetersiz olmasından dolayı bilgisayarlar üzerinde kullanılan yazılımlar ile gerçek zamanlı çalışabilecek bir hıza ulaşmak her zaman mümkün olmamaktadır. Bu yüzden FPGA'ler bu tarz uygulamalarda sıklıkla kullanılmaktadır. Bu çalışmada temel görüntü işleme algoritmalarından bazıları gerçek zamanlı olarak FPGA tabanlı geliştirme kartında gerçeklenmiştir. Çalışmada; Xilinx firmasının ürettiği ZedBoard Zynq-7000 geliştirme kartı kullanılmıştır. USB kamera sayesinde alınan işlenmemiş görüntülerde görüntüyü griye dönüştürme ve konvolüsyon işlemleri sonucu görüntüde kenar belirleme, bulanıklık giderme, keskinleştirme gibi uygulamalar gerçekleştirilip, sonuç geliştirme kartının HDMI çıkışına bağlanan monitör sayesinde izlenmiştir. Bu işlemler için Verilog donanım tanımlama dili kullanılmıştır.

Anahtar Kelimeler: FPGA, Gerçek Zamanlı Sistemler, Görüntü İşleme, Konvolüsyon, Verilog.

IMPLEMENTATION OF BASIC IMAGE PROCESSING ALGORITHMS ON FPGA IN REAL TIME

ABSTRACT

Nowadays, the importance of image processing is rapidly increasing in such fields as industrial automation, security, health, and traffic control in parallel with the developments in technology. The most challenging difficulty in applications used in these fields is to make system run real-time. It is not always possible to make the system run real-time by the use of a software used on a general purpose computer since the resources of memory, CPU and peripheral devices in computers are limited. Thus, FPGAs are commonly used in such applications. In this study, some of the basic real time images processing algorithms are implemented in a FPGA-based development kit. The ZedBoard Znyq-7000 development kit produced by Xilinx Company was used in the study. As a result of grayscale conversion and convolution operations, such applications as edge detection, sharpening and blurring were realized on the images taken by USB camera. The processed images were viewed by the use of a monitor connected to HDMI output of the development kit. Verilog HDL was used for these operations.

Keywords: FPGA, Real-Time Systems, Image Processing, Convolution, Verilog.

GİRİŞ

Günümüzde görüntü işleme uygulamaları, teknolojinin gelişimine paralel olarak endüstriyel otomasyon, güvenlik, sağlık ve trafik denetim gibi alanlarda hızlı bir artış göstermektedir. Bu alanlarda yapılan uygulamalardaki en büyük zorluk, sistemin gerçek zamanlı bir sistem olarak çalışmasını sağlamaktır (Rodriguez ve diğ., 2002). Sistemin gerçek zamanlı olarak çalışmasını, bilgisayarlardaki bellek ve çevresel cihaz kaynaklarının kısıtlı olmasından dolayı, genel amaçlı bir bilgisayar üzerinde uygulanan bir yazılım tarafından karşılamak her zaman mümkün değildir. Çünkü birçok görüntü işleme uygulamalarında her bir piksel üzerinde onlarca işlem yapılmaktadır. Bu işlemlerin işlemciler tarafından sıralı bir şekilde yapılması hem kaynak tüketimi hem de performans açısından olumsuz sonuçlar oluşturmaktadır (Kiran ve diğ., 2008). Fakat FPGA (Field Programmable Gate Arrays)'lerin işlemcilerden farklı olarak paralel işlem yapabilme yeteneği vardır. Bu sayede FPGA'lerde işlemler parçalara bölünüp aynı anda birden fazla işlem yapılabilir. Bu yüzden yüksek hıza ihtiyaç duyulan görüntü işleme uygulamalarında FGPA'ler sıklıkla tercih edilir (Çelik, 2013).

Mesela Nelson (2000) yüksek lisans tezinde sayı sıralı filtreleme, morfolojik işlemler ve konvolüsyon algoritmalarını FPGA donanımı üzerinde gerçeklemiş ve sonuçları MATLAB programında yazdığı program sonuçlarıyla karşılaştırmıştır.

Caner (2006) ise yüksek lisans tezinde akan bir video üzerinde yapay sinir ağları kullanarak FPGA donanımı üzerinde araç plaka tanıma sistemi gerçeklemiştir ve bu sistemi maliyet ve kendi başına çalışabilirlik açısından önemli olan sistemler için uygulanabilir hale getirmiştir.

Özcan (2009) lisans projesinde görüntü sensöründen alınan görüntü üzerinde gerçek zamanlı yumuşatma, keskinleştirme, kenar belirleme gibi filtreleme işlemlerini gerçeklemiştir ve sistemle bilgisayar arasındaki haberleşmeyi, RS232 protokolünü kullanan seri bir haberleşme birimiyle sağlamıştır.

Gacar (2009) yüksek lisans tezinde kameralardan alınan görüntülerin VGA (Video Graphics Array) girişine sahip bilgisayar monitöründe gösterilmesini sağlayan arabirim tasarlamıştır ve böylece plazma ve LCD (Liquid Crystal Display) televizyonlar yerine daha düşük maliyetli cihazların kullanılmasına çözüm sağlamıştır.

Çayır (2010) yüksek lisans tezinde FPGA platformundan yararlanarak yüksek çözünürlüklü görüntüler üzerinde görüntü temizleme uygulamalarında sıkça kullanılan gerçek zamanlı medyan filtre algoritmasını gerçeklemiştir.

Irmak (2010) ise yüksek lisans tezinde gerçek zamanlı trafik işareti tanıyan bir sistemi FPGA üzerinde gerçeklemiş ve şekilsel yönden farklı 32 adet trafik işaretini, yüksek bir doğruluk oranıyla tanıyabilen bir sistem tasarlamıştır.

Özsaraç (2011) yüksek lisans tezinde FPGA'de gerçek zamanlı video stabilizasyonu yapan bir sistem tasarlamış ve sonuçlarını MATLAB'de yazılan program ile kıyasladığında oldukça başarılı sonuçlar elde etmiştir.

Özçelik (2012) yüksek lisans tezinde temel görüntü işleme algoritmalarının bazılarını Altera DE2-70 ve DE7 geliştirme kartları üzerinde gerçek zamanlı bir sistemde uygulamıştır. Analog video girişinden alınan görüntü bilgisini işleyerek, sonucu VGA çıkışından ekrana yansıtmıştır.

Kızılkaya (2012) ise yüksek lisans tezinde Verilog donanım tanımlama dilini kullanarak termal görüntüde nesne ve hareket tespiti, kenar çıkarımı, gürültü azaltma, histogram görüntüleme ve doğrusal olmayan karşıtlık ayarı ve hareketli nesne tespiti gibi uygulamaları Altera DE2-70 kartında çalıştırmıştır.

Aktumak (2013) yüksek lisans tezinde kızılötesi kameralardan alınan görüntülerin çözünürlüğünün arttırılması için görüntü iyileştirme algoritmalarını gerçek zamanlı olarak FPGA üzerinde uygulamış ve sonuçları MATLAB programındaki uygulama sonuçlarıyla karşılaştırmıştır.

Oflamaz (2013) yüksek lisans tezinde gürültü ve bulanıklık içeren düşük çözünürlüklü bir videonun, FPGA içerisinde gerçek zamanlı olarak çözünürlüğünü

arttırmış ve sonuç olarak daha net bir video olarak yansıtılan bir sistem gerçekleştirmiştir.

Çelik (2013) yüksek lisans tezinde fraktal şeklin elde edilmesi, elmas-kare algoritması kullanılarak farklı yükseklik seviyelerine sahip şekillerden oluşan görüntülerin elde edilmesi ve görüntüdeki gürültülerin azaltılmasında kullanılan filtreleme işlemlerini FPGA üzerinde uygulamıştır.

Yalçın (2013) yüksek lisans tezinde gerçek zamanlı olarak trafik işareti bulma ve tanımlama için sistemi öncelikle MATLAB programında tasarlayıp optimize etmiş, sonrasında tasarımı FPGA üzerine aktarmış ve sonuç olarak % 90'a yakın doğruluk oranı elde etmiştir.

Kiran ve diğ. (2008) yaptıkları çalışmada görüntü işleme algoritmalarını FPGA, MATLAB ve C++ ortamında gerçeklemiş ve sonuçları geçen işlem süreleri açısından karşılaştırdığında FPGA'in diğerlerine göre daha hızlı olduğu sonucuna varmıştır.

Annovi ve diğ. (2009) iki boyutlu görüntülerde eşik değerin üzerindeki pikseller için kümeleme algoritmalarını FPGA'de uygulamış ve sistemi gerçek zamanlı sistemlerde kullanılabilecek şekilde tasarlamıştır.

Ramirez ve diğ. (2010) lineer ve morfolojik filtreleme işlemleri için FPGA'de gerçek zamanlı bir mimari tasarlamış, sonuç olarak FPGA'deki mevcut kaynakların küçük bir kısmını kullanmıştır.

Wu ve diğ. (2010) çalışmalarında FPGA'de video görüntü kenar belirleme sistemi tasarlamış ve yazılımsal bir sistem ile tamamen donanım kaynakları kullanılarak oluşturulan bir sistem arasındaki farklılıkları göstermeyi amaçlamışlardır.

Mayya ve diğ. (2010) sabit bir kamera kullanarak gerçek zamanlı uygulamalar için insan hareketi algılama sistemi tasarlamış ve sonuçları işlemci kullanarak oluşturduğu sistemle karşılaştırmıştır.

Lin ve diğ. (2010) önerdikleri çalışma ile diğer kübik konvolüsyon interpolasyon yöntemlerine oranla daha düşük kaynak kullanımı ve daha yüksek performans elde etmişlerdir.

Guo ve diğ. (2010) paralel işleme türlerinden biri olan Sobel kenar belirleme algoritmasını FPGA'de uygulamış ve sonuçları farklı eşik seviyelerine göre karşılaştırmıştır.

Barranco ve diğ. (2010) 64x64 boyutundaki resimlerde konvolüsyon işlemini FPGA'de gerçeklemiş ve farklı boyutlardaki çekirdek matrislerine göre FPGA'de kullanılan kaynak kullanımı oranlarını karşılaştırmıştır.

Gentsos ve diğ. (2010) gerçek zamanlı Canny kenar belirleme algoritmasını FPGA üzerinde paralel ve saniyede 240 kare işleyebilen bir mimari tasarlamıştır ve FPGA'deki toplam alan sadece %2 oranında kullanılmıştır.

Carlo ve diğ. (2011) uzay uygulamaları için 2 boyutlu konvolüsyon işlemini FPGA'de gerçeklemiş ve kullandıkları çekirdek matrisi boyutuna göre FPGA'de tüketilen kaynak kullanımlarını karşılaştırmışlardır.

Harinarayan ve diğ. (2011) hava görüntüleme sistemlerinde kullanılabilecek Sobel ve Prewitt kenar belirleme algoritmalarını FPGA'de gerçeklemiş ve otomatik navigasyon cihazları için uygulanabilir bir sistem tasarlamıştır.

Russo ve diğ. (2012) çalışmalarında konvolüsyon işlemini CUDA, MATLAB, FPGA ve C platformlarında uygulamış ve sonuçları farklı çözünürlükteki görüntülerdeki performanslara göre karşılaştırmıştır.

Torres ve diğ. (2013) iki boyutlu sonlu dürtü yanıtı ve medyan filtre algoritmalarını ardışık düzen (pipeline) tekniği kullanarak FPGA'de sentezlemiş ve yüksek çözünürlüklü, gerçek zamanlı video sistemleri için kullanılabilir bir mimari tasarlamışlardır.

AlAli ve diğ. (2013) çalışmalarında görüntü iyileştirme ve filtreleme algoritmalarını FPGA donanımında gerçeklemiş, ayrıca FPGA'de güç tüketimini azaltmak için bazı optimizasyon algoritmalarını da tasarladığı sistemde uygulamıştır.

Hanumantharaju ve diğ. (2013) görüntü iyileştirme, yumuşatma, kenar belirleme ve filtreleme işlemlerinde kullanılan iki boyutlu Gauss fonksiyonunu, gerçek zamanlı

olarak saniyede 30 kare işleyebilen ve FPGA'deki kaynakların küçük bir kısmını tüketen bir tasarım geliştirmiştir.

Wang ve diğ. (2014) kızılötesi resimler için, küçük nesneleri algılama sistemini FPGA ve DSP (Digital Signal Processing) sistemlerini beraber kullanarak gerçeklemiş ve gerçek zamanlı uygulamalarda kullanılabilecek kadar hızlı çalışabilen bir mimari geliştirmiştir.

Chaple ve Daruwala (2014) görüntü kenar belirlemede kullanılan Sobel Algoritmasını Spartan-3, Spartan-6 ve Virtex-5 geliştirme kitlerinde gerçeklemiş ve bu geliştirme kitlerinde tüketilen kaynak kullanım oranlarını karşılaştırmıştır.

Bağbaba ve diğ. (2014) çalışmalarında, medyan filtre algoritmasını mikroişlemci ve FPGA'yi beraber kullanarak gerçeklemiş ve farklı seviyelerdeki tuz ve biber gürültüsü uygulanmış resimlerde sonuçları karşılaştırmıştır.

Bu tez çalışmasında ise temel görüntü işleme algoritmalarından bazıları gerçek zamanlı olarak FPGA tabanlı geliştirme kartında gerçeklenmiştir. Çalışmada; Xilinx firmasının ürettiği ZedBoard Zynq™-7000 geliştirme kartı kullanılmıştır. USB (Universal Serial Bus) kamera sayesinde alınan işlenmemiş görüntü bir takım işlemlerden geçirilerek, sonuç geliştirme kartının HDMI çıkışına bağlanan monitör sayesinde izlenmiştir.

Tezin birinci bölümünde, FPGA hakkında genel bilgilere yer verilip, FPGA'lerin içyapısından, özelliklerinden ve avantajlarından bahsedilmiştir. İkinci bölümde, görüntü işleme yöntemleri ve kullanım alanları hakkında genel bilgiler anlatılmştır. Üçüncü bölümde, tasarlanan sistemde kullanılan donanım ve yazılımlar hakkında bilgiler verilmiştir. Dördüncü bölümde, gerçeklenen görüntü işleme algoritmalarının aşamaları detaylı anlatılmıştır. Beşinci bölümde yapılan çalışma sonunda elde edilen bulgular ele alınmıştır. Sonuç bölümünde ise; çalışmalardan elde edilen bulguların değerlendirmesi ve ileride yapılabilecek çalışmalar hakkında bilgiler verilmiştir.

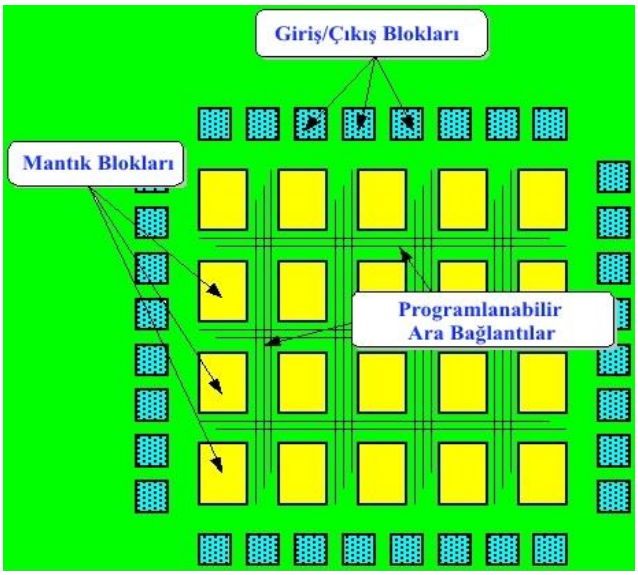
1. FPGA HAKKINDA GENEL BİLGİLER

1.1. FPGA'in Tanımı

FPGA'ler kullanım amacına göre iç yapısı değiştirilebilen entegre devrelerdir. FPGA'ler ihtiyaç duyulan mantıksal fonksiyonların kullanıcılar tarafından oluşturulabilmesi amacıyla alanda programlanabilecek şekilde üretilmiştir. Bu yüzden FPGA'in tanımında kullanılan 'alanda' kelimesi, bu devrelerin alanda üretimden sonra kullanıcılar tarafından programlandığını belirtmek için kullanılır (Çelik, 2013).

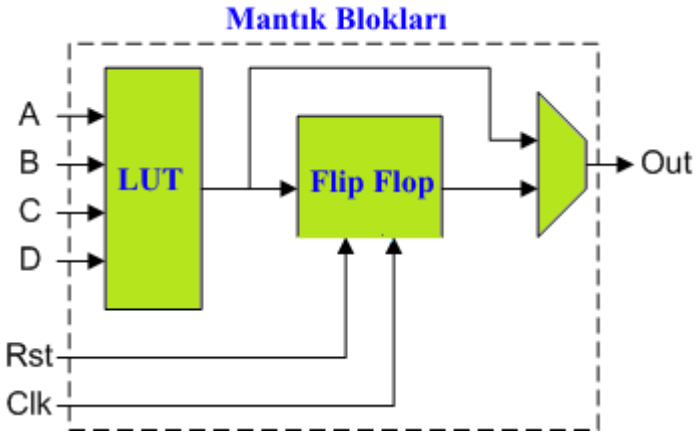
1.2. FPGA'in İç Yapısı

FPGA'ler programlanabilir mantık blokları (CLB), giriş çıkış blokları (I/O) ve ara bağlantı blokları olmak üzere üç temel bloktan (Şekil 1.1) oluşmaktadır.



Şekil 1.1. FPGA' in iç yapısı

Programlanabilir mantık blokları, FPGA'in temel mantık birimidir ve mantıksal fonksiyonlarının gerçeklendiği yapılardır. Bu bloklar (Şekil 1.2), genellikle 1 adet bakma tablosu (LUT), 1 adet flip flop ve 1 Adet çoklayıcıdan (MUX) oluşmaktadır (Caner, 2006).



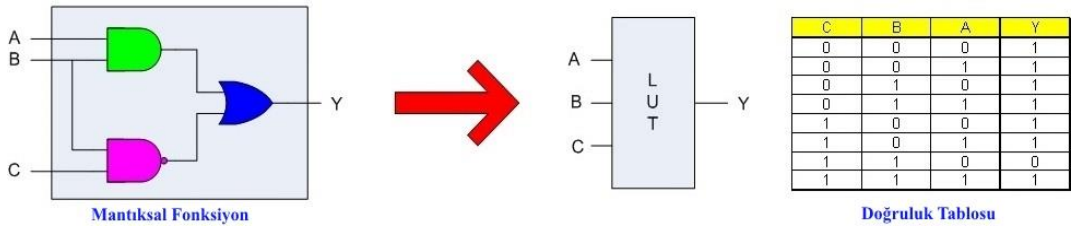
Şekil 1.2. Mantık blokları iç yapısı (URL-1)

Giriş-çıkış blokları dahili FPGA birimleri ile dış ortam arasında haberleşmeyi sağlayan birimlerdir. Bu bloklar kullanıcın belirlediği tasarıma göre giriş, çıkış ya da hem giriş hem de çıkış olarak programlanabilirler. Programlanabilir ara bağlantılar ise, mantık blokları ile giriş-çıkış blokları arasındaki bağlantıyı sağlarlar. Bu bağlantılar, yollandırma kanalları ve programlanabilir anahtarlardan oluşurlar (Şahin, 2004).

1.3. FPGA'in Çalışma Metodu

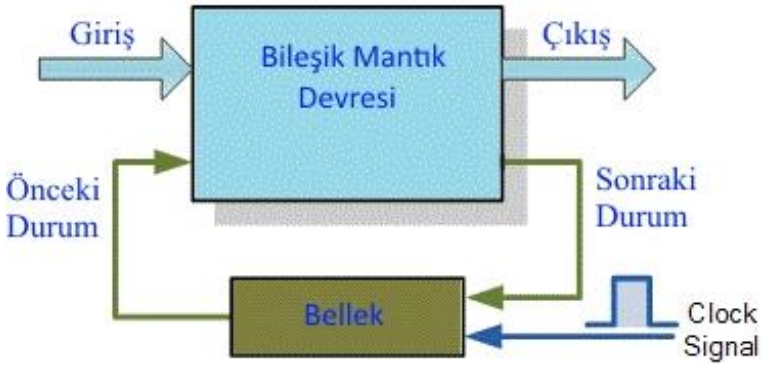
FPGA'de yapılan işlemler sayısal devrelerin gerçeklenmesi şeklindedir. Sayısal tasarımda geçerli olan kurallar FPGA tasarımlarında da geçerlidir. Sayısal tasarımlarda kullanılan devreler bileşik ve sıralı devreler olmak üzere ikiye ayrılır.

Bileşik devreler mantıksal kapılar, giriş ve çıkış değişkenlerinden oluşmaktadır. Bileşik devrelerde bütün girişler dış kaynaklardan beslenir ve çıkışlardan girişlere herhangi bir geri besleme olmaz. Yani çıkış değişkenleri giriş değişkenlerine bağlı olarak değişmektedir. SRAM (Static Random Access Memory) tabanlı FPGA'ler içerisinde bu devreler LUT (Look Up Table) kullanılarak oluşturulur (Sarıtaş ve Karataş, 2013). Şekil 1.3'te basit bir bileşik devrenin LUT içerisinde nasıl gerçeklendiği gösterilmiştir.



Şekil 1.3. Bileşik mantık devresi

Sıralı devreler ise bileşik devrelere geri besleme ve zamanlama gibi olguların eklenmesiyle oluşan devrelerdir (Ekiz, 2003). Bu devrelerde çıkış değerleri, bileşik devrelerin aksine sadece girişler tarafından tanımlanmazlar. Bu devrelerde bir geri besleme vardır. Yani girişlerin önceki değeri de çıkışları etkilemektedir. Bu yüzden sıralı devrelerde ek olarak bir belleğe ihtiyaç vardır (Şekil 1.4). FPGA'de bu bellek ihtiyacını karşılamak için flip floplar kullanılmaktadır.

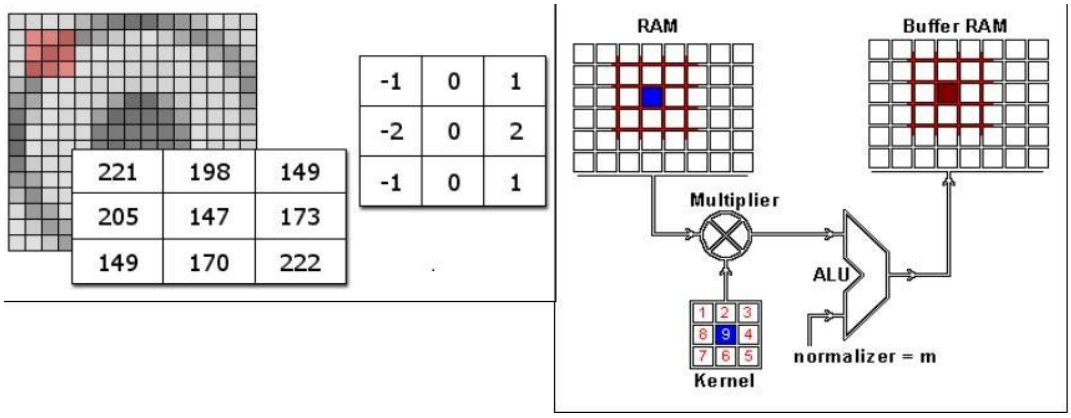


Şekil 1.4. Sıralı mantık devresi

1.4. FPGA'in Avantajları

FPGA'leri ASIC (Application Specific Integrated Circuit) gibi diğer entegre devrelerden ayıran önemli bir özelliği, donanım yapısının kullanım amacına göre defalarca değiştirilip çalıştırılabilmesidir. Örneğin; ASIC gibi devrelerde üretilen bir cihaz, hatalı üretilmişse bu cihazın donanım yapısı değiştirilip, tekrar çalıştırılması mümkün değildir. FPGA'de üretilen bir cihaz ise hatalı üretilse dahi, hata düzeltilerek donanım yapısı değiştirilir ve tekrar çalıştırılabilir. Bu durum kullanıcıya programlama açısından büyük bir esneklik kazandırır. FPGA'in bir başka önemli özelliği geleneksel işlemcilerden farklı olarak işlemleri paralel olarak yapabilmesidir. Paralel işlem yapabilme aynı anda birden fazla işlemin yapılabilmesidir. Örnek olarak görüntü işleme uygulamalarında yaygın olarak kullanılan konvolüsyon

işlemini verebiliriz. Konvolüsyon işleminde görüntü matris değerlerinin, 3x3 boyutundaki konvolüsyon çekirdeği ile işleme tabi tutulmaktadır. Bu şekilde yapılmak istenildiğinde her bir konvolüsyon işlemi için görüntü matrisindeki 9 değerin (resim karesi) alınması gerekmektedir. Geleneksel işlemciler kullanıldığında ilk önce görüntünün ilk resim karesi ve bu resim karesindeki 9 değer sırayla alınır, konvolüsyon işlemine tabi tutulur ve işlem sonlandıktan sonra ikinci resim karesi alınmaya başlanır (Çelik, 2013). Sonuç olarak işlemcilerde bir işlem sonlandıktan sonra bir başka işlem yapılır. Yani bir işlem sonlanmadan diğer işlemler yapılamamaktadır. FPGA'de ise bu işlemler daha farklı yapılabilir. FPGA'in işlemleri paralel olarak yapabilmesi sayesinde, aynı anda birden fazla resim karesi alınabilir ve konvolüsyon işlemi uygulanabilir. Böylece bu tarz uygulamaların FPGA'de yapılması, geleneksel işlemcilerde yapılana göre hız açısından oldukça avantajlı bir durum oluşturmaktadır.



Şekil 1.5. FPGA'de işlemlerin paralel olarak yapılması

1.5. FPGA Kullanım Alanları

Yeniden programlanabilir olmasından dolayı FPGA'ler birçok alanda uyumlu bir şekilde kullanılmaktadır. Havacılık ve savunmada, şerit ve yol tanıma gibi sürücü destek sistemleri için otomotiv sektöründe, tüketici elektroniğinde, yüksek performanslı bilgisayarlarda ve veri depolamada, endüstriyel otomasyonda, tıbbi uygulamalarda, güvenlik sektöründe, video ve görüntü işlemede, kablolu ve kablosuz iletişim gibi birçok alanda FPGA'ler kullanılmaktadır (URL-2). Ayrıca ASIC devrelerinin seri üretime geçişinden önce FPGA ile gömülü yazılım test edilerek sistemin doğrulanmasında da kullanılmaktadır.

2. SAYISAL GÖRÜNTÜ İŞLEME

2.1. Görüntü İşleme Tanımı

Görüntü işleme, görüntü üzerindeki yararlı bilgileri ayıklamak amacıyla, görüntüyü sayısal forma dönüştürmek ve görüntü üzerinde bazı işlemleri gerçekleştirmek için kullanılan bir yöntemdir. Başka bir deyişle görüntü işleme, girişi resim ya da video karesi, çıkışı ise görüntü veya görüntü ile ilgili özellikleri olan bir sinyal dağıtma türü olarak da tanımlanabilir (Arvind ve diğ., 2014).

x, y düzlemsel koordinatlar ve (x, y) koordinat çiftinin genliği bu noktadaki gri seviyesi olmak üzere, bir görüntü iki boyutlu f(x, y) fonksiyonu olarak tanımlanabilir. f'in x, y ve genlik değerleri sonlu ve ayrık miktarlarda olduğunda bu görüntülere sayısal görüntü denir. Sayısal görüntüler, her biri belirli konum ve değere sahip sonlu sayıda elemandan oluşmaktadır. Bu elemanlara piksel denilmektedir (Gonzalez ve Woods, 2008).

Sayısal görüntü işleme yöntemleri, bilgisayarlar kullanılarak sayısal görüntülerin düzenlenmesi için kullanılmaktadır. Bu yöntemler genel olarak, ön işleme, geliştirme ve bilgi çıkarımı olmak üzere üç aşamadan oluşmaktadır (Bandyopadhyay ve Ghosh, 2013). Ön işleme, istenmeyen bozulmaları azaltmak ve sonraki işlemlerde görüntüdeki yararlı ve önemli bilgileri geliştirmek için, düşük seviyede görüntüler üzerinde kullanılan yöntemdir (Miljkovic, 2006). İyileştirme, görüntü içerisindeki bilgilerin yorumlanabilirliğini ya da görüntünün görsel algılığını arttırmak için kullanılan yöntemdir. Bu yöntem sırasında görüntüdeki bir ya da birkaç özellik, istenilen amaca uygun olacak şekilde değiştirilmektedir (Maini ve Aggarwal, 2010). Bilgi çıkarımı ise, görüntüleri sınıflandırmak amacıyla görüntüye ait bilgi ve özellikleri çıkaran yöntemlerdir. Bu yöntemler endüstriyel otomasyon, biyomedikal, güvenlik, biyometrik kimlik doğrulama gibi alanlarda önemli bir değere sahiptir (Choras, 2007).

2.2. Görüntü İşleme Kullanım Alanları

Görüntü işleme uygulamaları hemen hemen her alanda uygulanmakta ve bu uygulamaların kullanım alanları gittikçe artmaktadır. Görüntü işleme kullanım alanlarından bazıları şu şekildedir:

- ✓ Kriminoloji / Adli Tıp
- ✓ Tıbbi Görüntüleme
- ✓ Uzaktan Algılayıcı Sistemler
- ✓ Askeri Sanayi
- ✓ Nakliyat
- ✓ Kalite Kontrol
- ✓ Sayısal Kamera Görüntüleri
- ✓ Morfoloji
- ✓ Bilgisayarla Görme

2.3. Görüntü İşleme Algoritmaları

2.3.1. Renkli görüntülerin griye dönüştürülmesi

Renkli ya da başka bir deyişle RGB (Red, Green, Blue) resimleri griye dönüştürme yöntemi, görüntü işleme uygulamalarında sıklıkla kullanılan bir yöntemdir. Bu yöntemin sıklıkla kullanılmasının nedeni renkli görüntüler yerine, gri tonlamalı görüntüler üzerinde çalışılarak, görüntü üzerinde yapılacak hesaplama gereksinimlerinin azaltılmasıdır. Böylece uygulamalarda, gereksiz işlem daha iyi performans elde edilir. Renkli bir resmin griye dönüştürülmesi için birden çok yöntem kullanılmaktadır. Bu yöntemlerden bazıları, Denklem (2.1), (2.2) ve (2.3)'de verilen;

$$Y = \frac{1}{3}(R + G + B) \tag{2.1}$$

$$Y = 0,2989.R + 0,5870.G + 0,1140.B \tag{2.2}$$

$$Y = 0,2126.R + 0,7152.G + 0,0722.B \tag{2.3}$$

formüllerle hesaplanır (Kanan ve Cottrell, 2012). Burada Y; gri seviye piksel renk bilgisi, R; görüntünün kırmızı renk değeri, G; görüntünün yeşil renk değeri, B ise; görüntünün mavi renk değeridir.

Denklem (2.1), renkli görüntünün kırmızı, yeşil ve mavi bileşenlerinin aritmetik ortalaması alınarak hesaplanmaktadır. Denklem (2.2), RGB kanallarının ağırlıklı bir kombinasyonunu kullanarak, insan gözü parlaklık algısına olan uygunluğuna göre tasarlanmıştır (Pratt, 2007). Ayrıca bu yöntem bilgisayarla görme alanında sıkça kullanılır ve MATLAB programında "rgb2gray" fonksiyonu da bu formülle oluşturulmuştur (Şekil 2.1). Denklem (2.3) ise, Denklem (2.2)'deki katsayılara yakın değerlerle hesaplanır. Bu yöntem yüksek çözünürlüklü televizyonlarda (HDTV) kullanılmaktadır (Jack, 2001).

a) b)

Şekil 2.1. MATLAB programında renkli bir resmin griye dönüştürülmesi a) renkli görüntü, b) gri seviyeli görüntü

2.3.2. İki boyutlu lineer konvolüsyon

Konvolüsyon, görüntü işlemede sıkça kullanılan bir başka önemli algoritmalarından biridir. Konvolüsyon işlemi görüntü matris değerleriyle çekirdek matris değerlerinin çarpımlarının toplamını gerçekleştiren bir işlemdir. Konvolüsyon işlemi, görüntüye düzlemsel alçak ve yüksek geçiren filtre uygulamak için kullanılır (Jayaraman ve diğ., 2009). Matematiksel olarak iki boyutlu konvolüsyon işlemi, Denklem (2.4)'de gösterilen;

$$y(m,n) = x(m,n) * h(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x[k,l].h[m-k,n-l] \quad (2.4)$$

formülle temsil edilmektedir. Burada x giriş görüntüsünü, y çıkış görüntüsünü, h ise çekirdek matrisini temsil etmektedir.

Konvolüsyon işlemine ait özellikler şu şekilde sıralanabilir (Burger ve Burge, 2009):

Yer değiştirme: $X * H = H * X$ (2.5)

Giriş görüntüsü ve çekirdek matrisi yer değiştirilirse bile sonuç değişmemektedir.

Lineerlik: $(s.X) * H = X * (s.H) = s.(X * H)$ (2.6)

Giriş görüntüsünün skaler bir sabit ile çarpılıp, konvolüsyon işlemine tabi tutulmasıyla giriş görüntüsünün, skaler bir sabit ile çarpılmış çekirdek matrisi ile konvolüsyon işlemine tabi tutulması aynı sonucu vermektedir.

$$(X1 + X2) * H = (X1 * H) + (X2 * H) \quad (2.7)$$

Denklem (2.7)'de gösterildiği gibi piksel piksel eklenen iki giriş resmin birlikte konvolüsyon işlemine tabi tutulmasıyla, iki resmin ayrı ayrı konvolüsyon işlemine tabi tutulup toplanması aynı sonucu vermektedir.

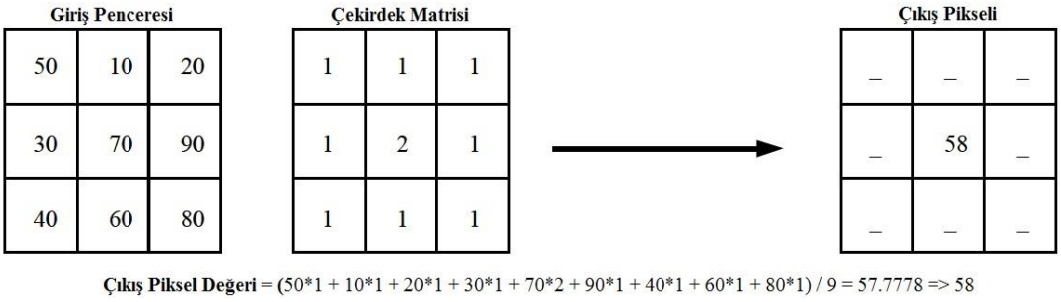
$$(s + X) * H \neq s + (X * H) \quad (2.8)$$

Fakat Denklem (2.8)'de gösterildiği gibi, giriş görüntüsünün skaler bir sabit ile toplanıp, konvolüsyon işlemine tabi tutulmasıyla skaler bir sabit ile konvolüsyon işleminin sonucunun toplanması birbirine eşit değildir.

Birleşme: $A * (X * H) = (A * X) * H$ (2.9)

Denklem (2.9)'da görüldüğü gibi konvolüsyon işleminin birleşme özelliği vardır.

Konvolüsyon işlemi uygulanan bir görüntüde, giriş piksel pencere boyutu ile çekirdek matrisi daima aynı boyuttadır. Çıkış pikseli ise, hesaplama sonunda en yakın tam sayıya yuvarlanmaktadır. Konvolüsyon işleminde bir pikselin hesaplanmasına ait bir örnek Şekil (2.2)'de gösterilmiştir.



Şekil 2.2. Konvolüsyon işleminde bir pikselin hesaplanması

Konvolüsyon işleminde kullanılan farklı çekirdek matrislerine göre görüntüde kenar belirleme, keskinleştirme, bulanıklık giderme gibi işlemler yapılır (Bağbaba ve diğ., 2014). Çekirdek matrisinin boyutları kullanılan uygulamaya göre farklılık gösterse de yaygın olarak 3x3' lük matrisler kullanılmaktadır. Şekil (2.4) de farklı çekirdek matrisi kullanılarak, konvolüsyon işlemine tabi tutulan bir görüntüye ait sonuçlar gösterilmiştir.

c) d)

e) f)

Şekil 2.3. Bir görüntüye konvolüsyon işleminin uygulanması a) orijinal görüntü, b) bulanıklık giderme, c) keskinleştirme, d) yumuşatma

3. KULLANILAN DONANIM VE YAZILIMLAR

3.1. ZedBoard Zynq-7000 FPGA Geliştirme Kartı

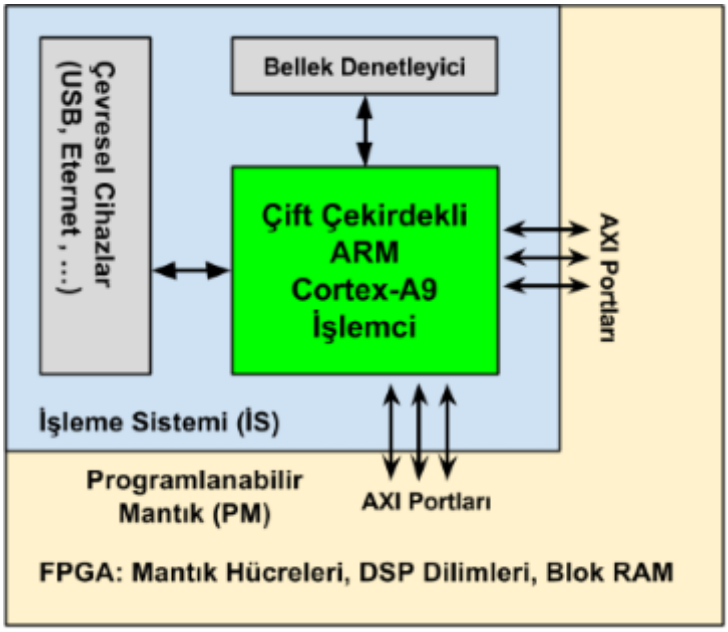
Tez çalışmasında gerçekleştirilen uygulamalarda Xilinx firmasının ürettiği ZedBoard Zynq-7000 FPGA geliştirme kartı (URL-3) kullanılmıştır. Zynq-7000 standart FPGA'lerden farklı olarak, aynı çip üzerinde Artix-7 FPGA ve ARM Cortex A9 işlemciyi beraber barındırmaktadır. Bu yapıya sahip bir FPGA kartında, uygulamadaki hesaplamanın yoğun olduğu kısımlar FPGA üzerinde, hesaplama içermeyen kontrol kısımları ise işlemci tarafında yazılım kullanılarak yapılabilmektedir. Zynq-7000 e ait donanım görünümü Şekil 3.1'de ve Zynq-7000 geliştirme kartının özellikleri Tablo 3.1'de gösterilmiştir.

Şekil 3.1. Zedboard Zynq-7000 FPGA geliştirme kartı

Tablo 3.1. Zynq-7000 geliştirme kartının özellikleri

İşlemci	Çift çekirdekli ARM Cortex A9
Hafıza	512 MB DDR3, 4 GB SD Kart
Giriş / Çıkış Arabirimleri	Programlama ve seri haberleşme için USB-UART, 10/100/1G Ethernet, USB OTG 2.0, 12-Bit VGA Çıkışı, HDMI Çıkışı, Ses Giriş Çıkışı, Kulaklık, Mikrofon
Ekranlar	128x32 OLED
Anahtarlar ve LED'ler	Programlanabilir mantık ve işleme sistemleri için erişilebilir sıfırlama düğmeleri ve LED'ler
Saat Çevrim Frekansı	Programlanabilir mantık için 100 Mhz Osilatör, İşleme sistemi için 33.333 Mhz Osilatör

Zynq, işleme sistemi ve programlanabilir mantık olmak üzere iki kısımdan oluşmaktadır. İşleme sistemi (İS), ARM Cortex A9 işlemci, kayan nokta birimi (floating point unit), DDR denetleyici, gigabit ethernet denetleyici, USB (Universal Serial Bus) denetleyici gibi yapıları barındırdığı için geleneksel bir işlemci gibi çalışmaktadır. Programlanabilir mantık (PM) kısmı ise standart FPGA'de bulunan tüm yapıları barındırmaktadır. İşleme sistemi ve programlanabilir mantık kısımları arasındaki haberleşme yüksek performanslı veri yolları ile sağlanmaktadır. Zynq-7000'nin içyapısı Şekil 3.2'de gösterilmiştir.



Şekil 3.2. Zynq-7000 iç yapısı

3.1. Usb Kamera

Tez çalışmasında gerçekleştirilen uygulamalarda görüntüler A4 Tech PK-635K marka usb kamera aracılığıyla alınmıştır. Kullanılan kamera (Şekil 3.3) 40 fps hızıyla maksimum 640x480 çözünürlükte görüntü yakalayabilmekte ve USB 2.0 portundan kolayca kurulumu yapılabilmektedir.

Şekil 3.3. A4 tech usb kamera

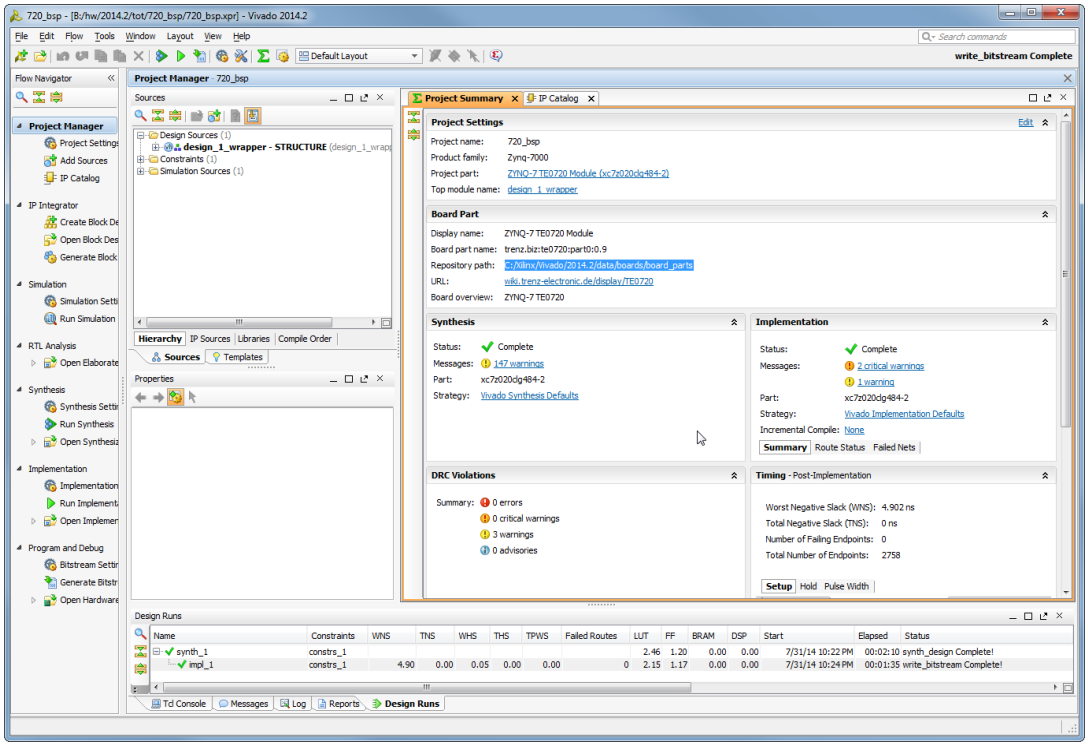
3.2. Kullanılan Diğer Donanımlar

Çalışmada ayrıca çeşitli algoritmalardan geçirilen görüntülerin ekranda gösterilebilmesi için monitör ve HDMI kablo ve usb çoklayıcı kullanılmıştır. HDMI kablonun bir ucu FPGA geliştirme kartının çıkış portuna, diğer ucu monitörün giriş portuna bağlanarak usb kameradan alınan orijinal görüntü ve algoritmaların gerçeklenmesiyle oluşan görüntüler gerçek zamanlı olarak ekranda gösterilmesi sağlanmıştır.

3.3. Kullanılan Yazılımlar

Çalışmada yazılım aracılığıyla USB kameradan görüntü alınması için C programlama dili, görüntü işleme algoritmalarını donanımsal olarak gerçeklemek içinse Xilinx firmasının hazırlamış olduğu Vivado arayüzü üzerinde, Verilog donanım tanımlama dili kullanılmıştır.

Vivado arayüzü entegre yazılım geliştirme platformudur. Mantık kapıları tasarımı devre tasarımı ve simülasyonu yazılan HDL (Hardware Description Language) kodunun derlenmesi işlemleri Vivado arayüzü aracılığıyla yapılmaktadır. Vivado kullanılarak, şematik ve HDL dilleri ile tasarım oluşturulur ve oluşturulan tasarım sentezlenir ve sentezlenen tasarımın FPGA'e gömülmesi gerçekleştirilmektedir. Ayrıca tasarımın doğruluğu Vivado arayüzünde simülasyon yapılarak kontrol edilebilmektedir (Çelik, 2013). Şekil 3.4'de Vivado arayüzünün anasayfa ekranı gösterilmektedir.



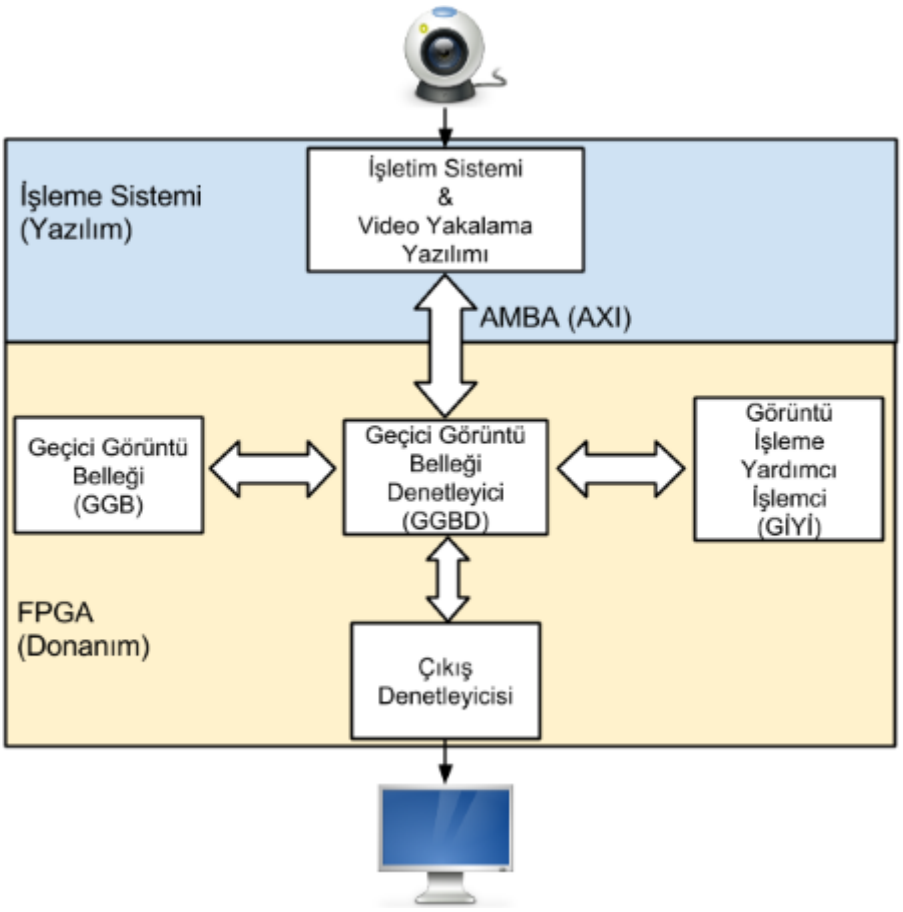
Şekil 3.4. Vivado ara yüzü anasayfa ekranı

4. SİSTEM VE DONANIM TASARIMI

4.1. Sistem Tasarımı

4.1.1. Genel sistem

Geliştirilen sistem, yazılım ve donanım olmak üzere iki ana kısımdan oluşmaktadır. Yazılım, görüntü kaynağından görüntü yakalama ve donanım kısmına aktarma işlemini gerçekleştirmektedir. Yazılımda, Linux üzerinde çalışabilen herhangi bir görüntü kaynağı (USB kamera, ağdan gelen bir video akışı, video dosyası vb.) kullanılabilmektedir. Donanım kısmında ise görüntü işleme algoritmaları gerçeklenmiştir. Yazılım ve donanım kısımlarının birlikte kullanılmasıyla oluşan genel sisteminin şeması Şekil 4.1'de gösterilmiştir.



Şekil 4.1. Genel sistem şeması

Sistemde ilk olarak yazılım ve işletim sistemi aracılığıyla görüntü kaynağından görüntü çerçevesi alınmaktadır. Ardından yazılım görüntü çerçevesi üzerinde işlenecek olan kısmını FPGA üzerindeki geçici belleğe aktarmaktadır. Donanım tarafında ise aktarım işlemi Geçici Görüntü Belleği Denetleyici (GGBD) modülü ile kontrol edilmektedir. Görüntü GGBD üzerinden Geçici Görüntü Belleği (GGB) modülüne aktarılır. GGB modülünde görüntü parçasının işlenmemiş ve işlenmiş halleri saklanabilmektedir.

Görüntü işlemle algoritmalarını Görüntü işleme yardımcı işlemcisi (GİYİ) isimli modül gerçekleştirmektedir. Görüntü işleme işlemleri, GİYİ modülünün pixelleri GGB modülünden okuması ve piksellerin işlenmiş halini geri yazması şeklinde gerçekleşir. Bu modüller Bölüm 4.2' de ayrıntılı olarak açıklanmıştır. Donanım tarafında görüntü parçasının üzerinde yapılan işlemler tamamlandıktan sonra yazılıma durum bilgisi aktarılır. Bu aşamadan sonra işlenmiş olan görüntü gerekirse yazılım kısmından okunabilir. Geliştirilen sistemde FPGA kartının HDMI çıkışı kullanılarak görüntünün işlenmiş ve işlenmemiş hali monitörden gösterilmektedir.

4.1.2. Donanım/Yazılım ara yüzü

Çalışmada işleme sistemi ile programlanabilir mantık arasındaki bağlantıda gelişmiş mikro denetleyici veri yolu mimarisi (AMBA) ara yüz standartlarından, gelişmiş genişletilebilir ara yüzü (AXI) kullanılmıştır. AXI (Advanced eXtensible Interface), donanım modülleri arasında veri aktarımını sağlayan standartlaştırılmış protokoller bütünüdür. AXI, bellek eşlemeli (memory mapped) veya akış (stream) olmak üzere iki farklı şekilde çalışabilmektedir. Bu çalışmada bellek eşlemeli AXI kullanılmıştır. Bellek eşlemeli AXI, modül içerisindeki bellek ve yazmaçların işleme sistemi tarafındaki bellek adresleri ile ilişkilendirilerek çalışmaktadır. İşleme sistemi tarafından programlanabilir mantık kısmına erişim, eşlenen adrese yapılan okuma veya yazma işlemi şeklindedir.

Çalışmada, işleme sistemi ile programlanabilir mantık arasındaki iletişimde görüntü aktarımı ve kontrol olmak üzere iki ayrı AXI bağlantısı kullanılmıştır. Görüntü aktarımı için kullanılan AXI bağlantısında GGB modülüne erişilebilmektedir. Kontrol bağlantısından işlemin ilerleme durumunu belirten yazmaca erişilmektedir.

Görüntü aktarımı için kullanılan AXI bağlantısı 32 bit modda çalıştırılmıştır. Kameradan 24 bit RGB formatında görüntü alınmış fakat FPGA'e 32 bit olarak aktarılmıştır. Piksellerin 32 bit olarak aktarılmasının sebebi her saat çevriminde (clock cycle) bir piksel aktarılmasını sağlamaktır. Böylece donanım kısmındaki adres dönüşümleri ve bellekteki okuma-yazma işlemleri daha kolay ve az kaynak tüketerek gerçekleştirilmiştir. 24 bitten 32 bite dönüşüm en yüksek anlamlı sekiz bitin sıfırlanması şeklinde yapılmıştır. Bu sayede yazılım kısmında pikseller üzerinde ek işlem yapılmasına gerek kalmamıştır.

4.1.3. Yazılım

USB kamera, network video akışı, video dosyası gibi kaynaklardan görüntü yakalanması, FPGA'de yazılıma göre çok karmaşık ve zor bir işlemdir. Ayrıca görüntü yakalama için görüntü kaynağına özel donanım tasarımı yapılması gerekmektedir. Geliştirdiğimiz görüntü işleme uygulamasındaki amacımız görüntü yakalamadan çok, görüntü üzerinde gerçek zamanlı işlem yapabilmektir. Bu yüzden görüntü yakalama için yazılım kullanımı bize oldukça kolaylık sağlamıştır.

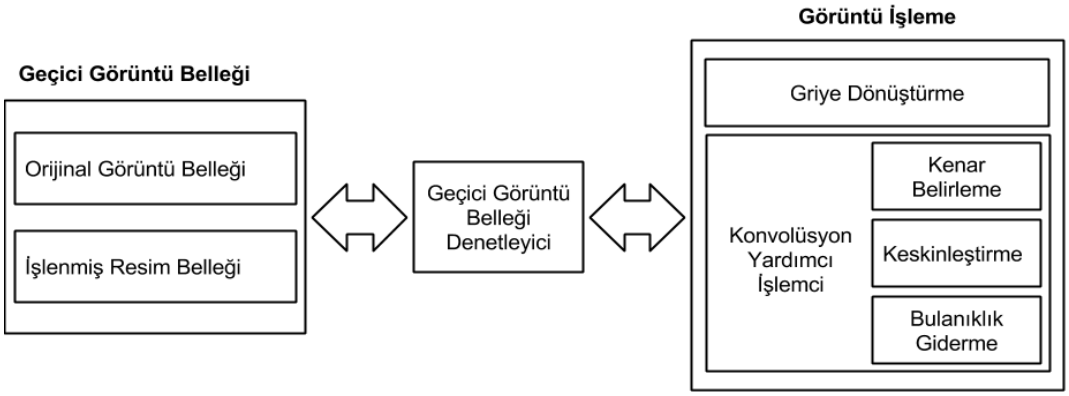
Kullandığımız geliştirme kartındaki ARM işlemci işletim sistemi çalıştırabilmektedir. Linux işletim sistemi ve kullanıcı uzayı (userspace) kütüphaneleri donanım için standart bir ara yüz sağlamaktadır. Bu sayede farklı görüntü kaynaklarına kolay bir şekilde yazılım üzerinden erişebilmektedir. Bu yüzden sistemin görüntü yakalama kısmında Linux işletim sistemi üzerinde geliştirilen bir yazılım kullanılmıştır.

Geliştirdiğimiz yazılım kullanılan videoya ait görüntü çerçevelerinin birim zamandaki geçiş hızına (frame rate) göre belli aralıklarla kameradan görüntü yakalamaktadır. Görüntünün işlenecek olan parçasının her pikseli donanım kısmına aktarıldıktan sonra yazılımda yoğun bekleme (busy waiting) yapılarak işlemin tamamlanması beklenmektedir. İşlem tamamlandığında donanım kısmı işlem tamamlanma durumunu bir bayrak (flag) ile yazılıma bildirmektedir. Ardından yazılım, eğer varsa görüntü çerçevesi içindeki sonraki parçaya geçer. Görüntü çerçevesindeki tüm pikseller işlendikten sonra bir sonraki görüntü çerçevesi kameradan alınıp aynı işlemler tekrarlanmaktadır. Test sisteminde her görüntü çerçevesi tek parça olarak kullanılmıştır.

Bu çalışma “Video For Linux 2” (Cox, 2000) kütüphanesi kullanılarak C programlama dilinde geliştirilen bir yazılım aracığıyla USB kameradan görüntü alınarak gerçekleştirilmiştir. Çalışmadaki donanım tasarımında herhangi bir değişiklik yapılmadan, sadece yazılım üzerinde değişiklik yapılarak farklı görüntü kaynakları da kullanılabilmektedir. Örneğin görüntü kaynağı olarak ağdan gelen bir video akışı veya hafıza kartı üzerindeki bir video dosyası kullanılabilir.

4.2. Donanım Tasarımı

Görüntü işleme yöntemleri GİYİ modülü tarafından gerçekleştirilmektedir. GGB modülü orijinal resmi ve işlenmiş resmi saklayan iki bellek biriminden oluşmaktadır. Görüntü işleme sırasında orijinal resmin ihtiyaç duyulan pikselleri GGB modülünden okunup, işlendikten sonra bu modülün ilgili alanına yazılmaktadır. Bu işlemler sırasında GGBD arada köprü görevi görmektedir. Şekil 4.2’de görüntü işlemede kullanılan modüller ve içeriklerinin sembolik diyagramı gösterilmiştir.



Şekil 4.2. Donanımda kullanılan modüller ve içerikleri

4.2.1. Geçici görüntü belleği

Bu modülün içerisinde piksel sayısı derinliğinde 24 bit genişliğinde iki bellek sistemi kullanılmıştır. Belleklerden birinde görüntünün orijinal hali, diğerinde ise işlenmiş hali saklanır. Her iki belleğinde adres aralığı 0 ile piksel sayısı (satır x sütun) aralığındadır. Bellekler byte (8 bit) şeklinde adreslenmek yerine 24 bit şeklinde adreslenmiştir. Kullanılan görüntü formatı 24 bit RGB olduğu için bir bellek hücresi bir pikseli alacak şekilde tasarlanmıştır.

Bellekler “eş zamanlı okuma yapabilen çift portlu bellek” biçiminde oluşturulmuştur. Bu sistemde ram de iki port bulunmaktadır. Portlardan biri okuma veya yazma, diğeri ise sadece okuma yapabilmektedir. İki portlu RAM kullanımı, görüntü işleme işlemleri ve görüntünün ekranda gösterilmesinin eş zamanlı olarak yapılabilmesini sağlamıştır.

4.2.2. Renkli resimleri griye dönüştürme tasarımı

Yapılan uygulamada renkli bir resmi griye dönüştürmede FPGA’de bulunan kaynak kullanımını azaltmak için Denklem (2.2)’de gösterilen formüldeki değerlere yakın katsayılar ile çarpma ve bölme işlemleri yapılmıştır. Çarpma ve bölme işlemleri bit kaydırma ve toplama işlemi olarak gerçekleştirilmiştir. Griye dönüştürme yöntemi için uyguladığımız formül Denklem (4.1) ve formülün elde ediliş biçimi Denklem, (4.2), (4.3), (4.4) ve (4.5)’de gösterilen,

$$Y = 0,296875.R + 0,59375.G + 0,109375.B \tag{4.1}$$

$$\begin{aligned} r &= 0,296875.R \\ &= (0,25 + 0,03125 + 0,015625).R \\ &= R >> 2 + R >> 5 + R >> 6 \end{aligned} \tag{4.2}$$

$$\begin{aligned} g &= 0,59375.G \\ &= (0,5 + 0,0625 + 0,03125).G \\ &= G >> 1 + G >> 4 + G >> 5 \end{aligned} \tag{4.3}$$

$$\begin{aligned} b &= 0,109375.B \\ &= (0,0625 + 0,03125 + 0,015625).B \\ &= B >> 4 + B >> 5 + B >> 6 \end{aligned} \tag{4.4}$$

$$Y = r + g + b \tag{4.5}$$

şeklinde ifade edilmiştir. Burada “ >> ” simgesi sağa kaydırma işlemini temsil etmektedir.

4.2.3. Konvolüsyon işlemi tasarımı

Konvolüsyon yardımcı işlemcisi (co-processor), 3x3 boyutundaki pencere üzerinde işlem yapıp, merkezdeki pikselin değerini hesaplamaktadır. Hesaplama, pencere içindeki 9 pikselin konvolüsyon çekirdeğindeki değerlerle çarpılması ve bu çarpımların sonuçlarının toplanmasıyla yapılmaktadır. Bu toplam 255'den büyük ise 255, 0'dan küçükse 0 olarak alınmaktadır.

Uygulamada kullanılan konvolüsyon çekirdekleri 2'nin üsleri veya yakın katsayılar içerdiği için çarpma işlemleri kaydırma ve toplama işlemi şeklinde yapılmıştır. Denklem (4.6) ve (4.7)'de bir pikselin değerinin hesaplanması örnek üzerinde gösterilmiştir.

$$H = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{4.6}$$

$$\begin{aligned} y_{11} = {} & (x_{00} + (x_{01} << 1) + x_{02} + \\ & (x_{10} << 1) + (x_{11} << 2) + (x_{12} << 1) + \\ & x_{20} + (x_{21} << 1) + x_{22}) >> 4 \end{aligned} \tag{4.7}$$

Denklem (4.6) ve (4.7)'de gösterilen örnekte, 2 ile çarpma işlemi 1 bit sola kaydırma, 4 ile çarpma işlemi 2 bit sola kaydırma şeklinde gerçekleştirilmiştir. Toplamların sonucunu 16'ya bölmek için ise 4 bit sağa kaydırma işlemi kullanılmıştır.

Görüntünün piksellerini tarama işlemi (Bosi ve Bois, 1999) önerdiği yöntem gibi her saat çevriminde konvolüsyon ara belleğine yeni bir piksel eklenip, en eski pikselin çıkarılması şeklinde yapılmaktadır. 3x3'lük konvolüsyon penceresinde satırlar arasındaki pikselleri saklamak için 2 tane fifo (first in, firs out) arabelleği kullanılmıştır. Şekil 4.3'de konvolüsyon işlemini yapan modülün blok diyagramı gösterilmiştir.

Şekil 4.3. Konvolüsyon işlemini yapan modülün çalışma sistemi

Bir pikselin hesaplanmasında fazla sayıda sıralı işlem olduğu için mantık tasarımında uzun bir veri yolu gereklidir. Geliştirme aşamasında tüm işlemi bir saat çevriminde yapan tasarım test edilmiştir. Bu tasarımda gerçek zamanlı çalışma için yeterli frekansa ulaşılamadığı görülmüştür. Tasarımın daha yüksek frekanslarda çalışabilmesi için işlem, birden fazla parçaya bölünerek veri yolu kısaltılmıştır. Böylece sistem daha yüksek frekanslarda çalıştırılabilmiştir. Bölünen işlemler ardışık düzen (pipeline) yöntemiyle yapıldığı için görüntün tamamı yine piksel sayısı kadar saat çevriminde hesaplanmıştır. Denklem (4.7)'de gösterilen işlemin, konvolüsyon işleminde bir piksel hesabının ardışık düzen yöntemiyle nasıl yapıldığı Denklem (4.8), (4.9) ve (4.10)'da gösterilmiştir.

$$y_{11a} <= x_{00} + (x_{01} << 1) + x_{02} + (x_{10} << 1) \tag{4.8}$$

$$y_{11b} <= (x_{11} << 2) + (x_{12} << 1) + x_{20} + (x_{21} << 1) + x_{22} \tag{4.9}$$

$$y_{11} <= (y_{11a} + y_{11b}) >> 4 \tag{4.10}$$

5. BULGULAR

Şekil 5.1'de test sisteminin çalışma sırasında çekilmiş fotoğrafı verilmiştir. Test sisteminde ZedBoard FPGA, HDMI monitör, standart USB kamera ve USB çoklayıcı kullanılmıştır.

Şekil 5.1. Test sistemi ekran görüntüsü

Yapılan çalışmada 4 farklı konvolüsyon çekirdeği FPGA üzerinde gerçeklenmiştir. Kullanılan konvolüsyon çekirdekleri Denklem (5.1), (5.2), (5.3) ve (5.4)'de gösterilmiştir. Görüntüde kenar belirleme yöntemi için Denklem (5.1) ve Denklem (5.2), keskinleştirme yöntemi için Denklem (5.2) ve bulanıklık giderme yöntemi için ise Denklem (5.3)'deki konvolüsyon çekirdek matrisleri kullanılmıştır.

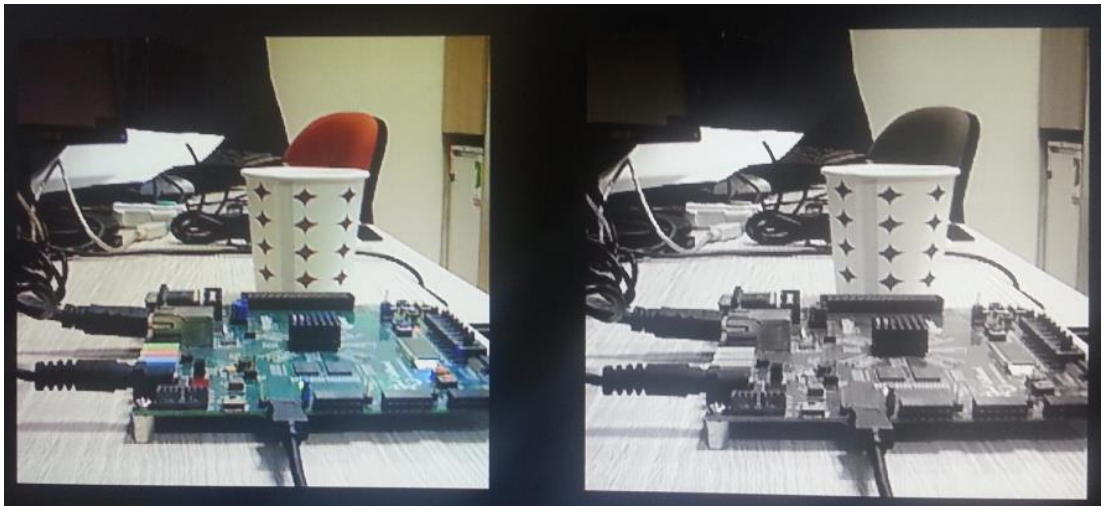
$$H = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (5.1)$$

$$H = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad (5.2)$$

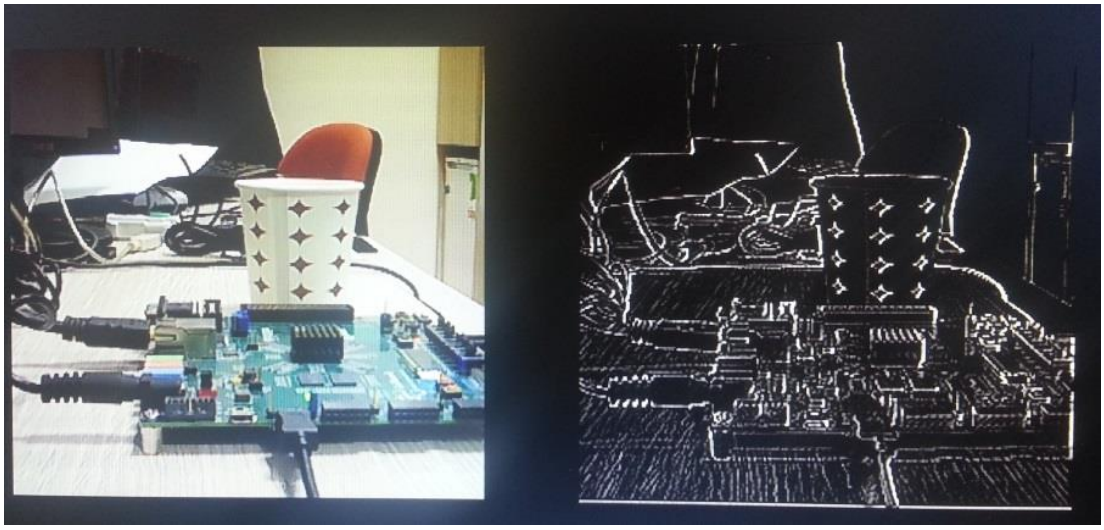
$$H = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad (5.3)$$

$$H = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \quad (5.4)$$

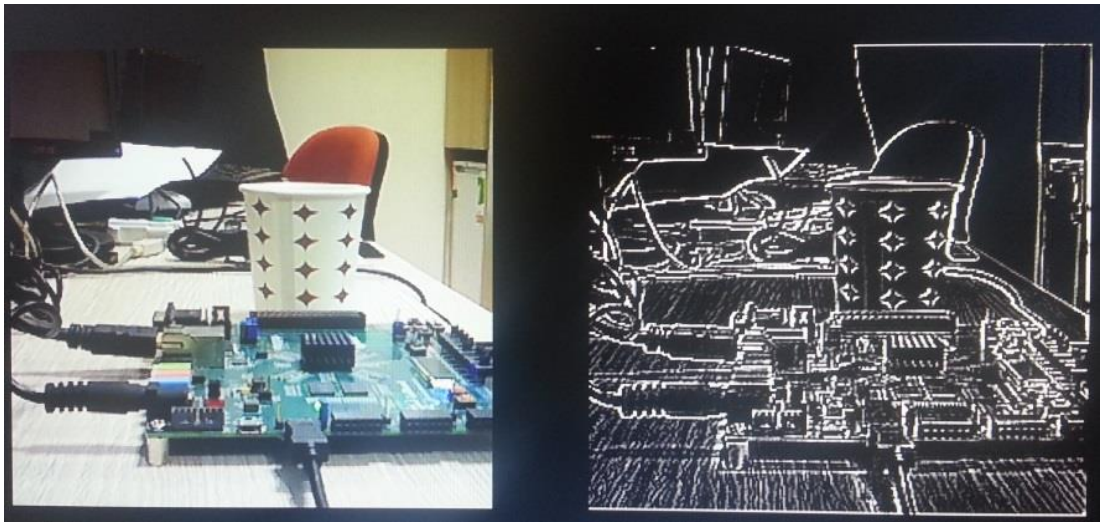
Kullanılan konvolüsyon çekirdek matris seçimi FPGA üzerindeki anahtarlar kullanılarak çalışma anında değiştirilebilmektedir. Şekil 5.2, Şekil 5.3, Şekil 5.4, Şekil 5.5 ve Şekil 5.6 'da uygulanan herbir yöntem için ekran çıktıları gösterilmiştir.



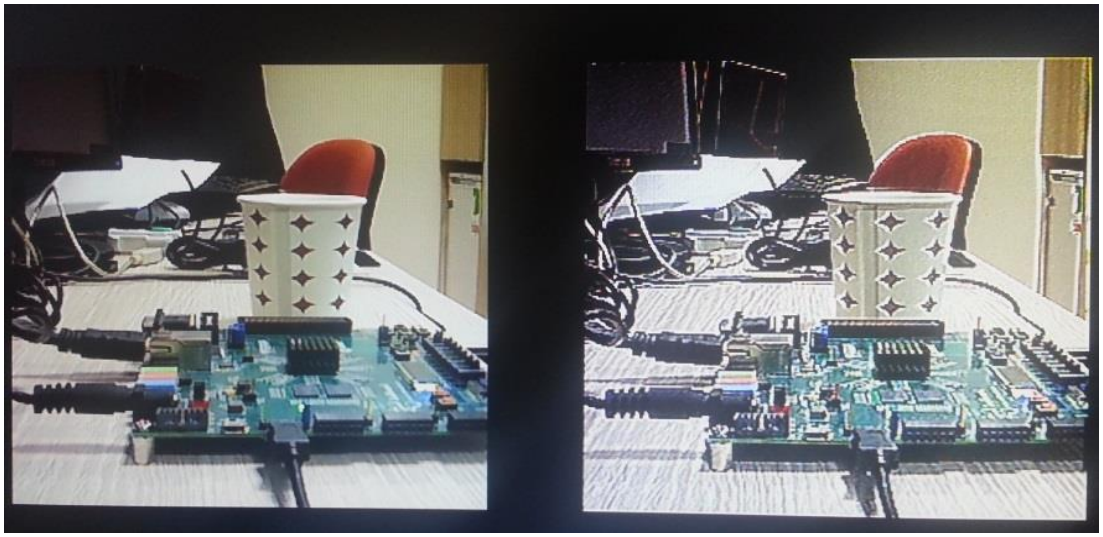
Şekil 5.2. Griye dönüştürme



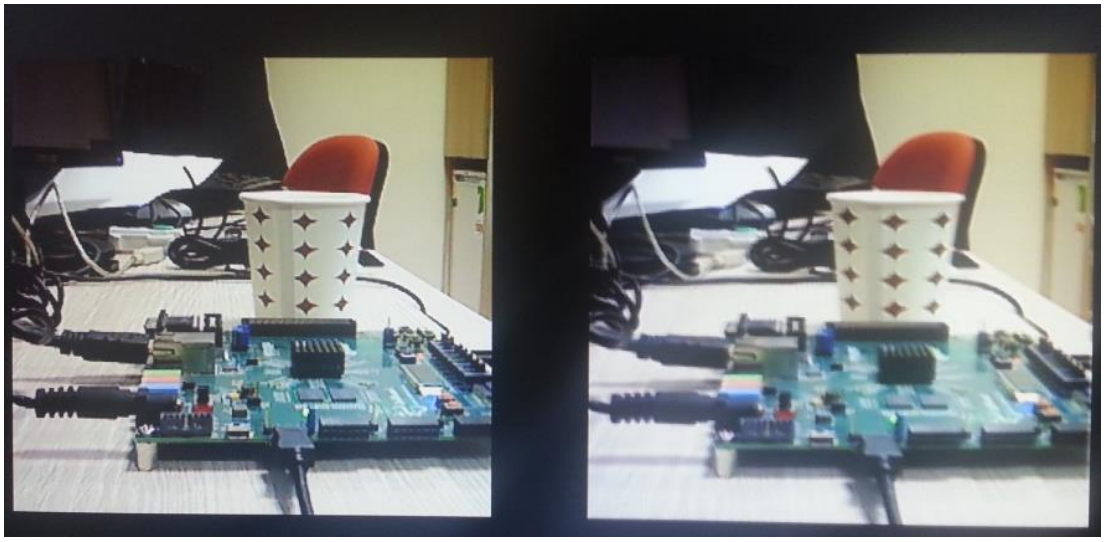
Şekil 5.3. Denklem (5.1) kullanılarak elde edilen kenar belirleme



Şekil 5.4. Denklem (5.2) kullanılarak elde edilen kenar belirleme



Şekil 5.5. Keskinleştirme



Şekil 5.6. Bulanıklık giderme

Tasarım Vivado 2014.2 ara yüzü kullanılarak Xilinx ZedBoard Zynq-7020 FPGA geliştirme kartı kullanılarak gerçeklenmiştir. FPGA kartında kullanılan kaynak kullanım oranları Tablo 5.1'de gösterilmiştir. Tablo 5.1'de görüldüğü gibi FPGA'de bulunan kaynakların küçük bir kısmı kullanılmıştır. Tablodaki sonuçlardan, sonraki çalışmalarda yüksek çözünürlüklü görüntüler üzerinde de görüntü işleme algoritmalarının başarıyla gerçeklenebileceği çıkarımına varılmıştır.

Tablo 5.1. FPGA kaynak kullanım miktarları

Kaynak	Kullanılan	Mevcut	Kullanılan %
Flip Flop	3652	106400	3,43
Bakma Tabloları	2648	53200	4,98
Hafızalı Bakma Tabloları	208	17400	1,20
Giriş / Çıkış Birimleri	46	200	23,00
Öbek Ram	97.5	140	69,64
Tümden Saat İşareti	5	32	15,62
Sayısal Saat Yöneticisi	1	4	25,00

Geliştirilen sistemde 256x256 boyutundaki görüntüler üzerinde işlem yapıldı ve konvolüsyon yardımcı işlemci frekansı 150 Megahertz (Mhz) olarak kullanıldı. Bir görüntü çerçevesi için harcanan süre yaklaşık olarak 0.44 milisaniyedir. Konvolüsyon yardımcı işlemcisi 2000 fps hızın üstüne çıkabilecek bir tasarıma sahiptir. Görüntü çerçevesinin kullandığımız AXI bağlantısının FPGA'e aktarımı ise yaklaşık 25 milisaniye sürmektedir. Bu yüzden gerçek zamanlı sistemde yaklaşık olarak 40 fps hıza ulaşılabilmiştir.

6. SONUÇ VE ÖNERİLER

Bu çalışmada temel görüntü işleme algoritmalarından renkli görüntüleri griye dönüştürme ve konvolüsyon işlemi algoritmaları gerçek zamanlı olarak FPGA tabanlı geliştirme kartında gerçeklenmiştir. Çalışmada; Xilinx firmasının ürettiği ZedBoard Zynq-7000 geliştirme kartı kullanılmıştır. USB kamera sayesinde alınan işlenmemiş görüntülerde görüntüyü griye dönüştürme ve konvolüsyon işlemleri sonucu görüntüde kenar belirleme, bulanıklık giderme, keskinleştirme gibi uygulamalar gerçekleştirilip, sonuç geliştirme kartının HDMI çıkışına bağlanan monitör sayesinde izlenmiştir. Bu işlemler için, FPGA'de bulunan kaynakların küçük bir kısmı kullanılmıştır.

Çalışma sonucunda 256x256 resimde yaklaşık 40 fps hızla çalışan gerçek zamanlı bir sistem elde edilmiştir. Konvolüsyon yardımcı işlemci tasarımı büyük resimlerle çalışmak için yeterli olmasına rağmen, çalışma düşük boyutlu resimlerle sınırlı kalmıştır. Bunun sebebi gerçekleştirdiğimiz FPGA'e görüntü aktarımı tasarımının yeterli olmamasıdır. İleriye yönelik çalışmalarda doğrudan bellek erişimi (DMA) kullanılarak, FPGA'e görüntü aktarımının hızlandırılması ve bu sayede yüksek çözünürlüklü görüntüler üzerinde gerçek zamanlı olarak çalışılması amaçlanmaktadır.

KAYNAKLAR

Aktukmak M., Kızılötesi Kameralar İçin Gerçek Zamanlı Sayısal Video Süper Çözünürlük FPGA Uygulaması, Yüksek Lisans Tezi, Orta Doğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2013, 338325.

AlAli M. I., Mhaidat K. M., Aljarrah I. A., Implementing Image Processing Algorithms in FPGA Hardware, *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies*, Amman, Jordan, 3-5 December 2013.

Annovi A., Berretta M., Crescioli F., Dell'Orso M., Giannetti P., Laurelli P., Maccarrone G., Sansoni M., Sartori L., Volpi G., A Fast FPGA-Based Clustering Algorithm for Real Time Image Processing, *Nuclear Science Symposium Conference Record*, Orlando, USA, 24 October 2009.

Arvind R., Gowtham C. M., Karthi M., Prabhu T. N., Intensification of Unreliable Radiant Images by Using İnstinctive Sustained Algorithms, *International Journal of Computer Science and Information Technologies*, 2014, **5**, 1414-1417.

Bagbaba A. C., Ors B., Erozan A. T., Görüntü İyileştirme İşlemcisi ve Uygulamaları, *22nd Signal Processing and Communications Applications Conference*, Trabzon, Türkiye, 23-25 Nisan 2014.

Bandyopadhyay S. K., Ghosh S., Image Extraction Using Image Mining Technique, *IOSR Journal of Engineering*, 2013, **3**, 36-42.

Barranco L., Rodriguez F., Jimenez A., On the AER Convolution Processors for FPGA, *2010 IEEE International Symposium on Circuits and Systems*, Paris, France, 30 May-2 June 2010.

Bosi G., Bois Y. S., Reconfigurable Pipelined 2D Convolvers for Fast Digital Signal Processing, *IEEE VLSI Systems Transactions*, 1999, **3**, 299-308.

Burger W., Burge M. J., *Principles of Digital Image Processing*, 1st ed., Springer, London, 110-113, 2009.

Carlo S. D., Gambardella G., Indaco M., Rolfo D., Tiotto G., Prinetto P., An Area-Efficient 2-D Convolution Implementation on FPGA for Space Applications, *2011 IEEE 6th International Design and Test Workshop*, Beirut, Lebanon, 11-14 December 2011.

Chaple G., Daruwala R. D., Design of Sobel Operator Based Image Edge Detection Algorithm on FPGA, *International Conference on Communications and Signal Processing*, Melmaruvathur, India, 3-5 April 2014.

Choras R. S., Image Feature Extraction Techniques and Their Applications for CBIR Andbiometrics Systems, *International Journal of Biology and Biomedical Engineering*, 2007, **1**, 6-16.

Cox A., *Video4Linux Programming*, 1st ed., Alan Cox, Boston, 19-25, 2000.

Çayır T., FPGA Üzerinde Görüntü İyileştirme Uygulaması, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2010, 259804.

Caner H., FPGA Donanımı Üzerinde Araç Plakası Tanıma Sistemi, Yüksek Lisans Tezi, Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2006, 200853.

Çelik A. R., Görüntü İşleme Algoritmalarının FPGA Donanımı Üzerinde Gerçeklenmesi, Yüksek Lisans Tezi, Kahramanmaraş Sütçü İmam Üniversitesi, Fen Bilimleri Enstitüsü, Kahramanmaraş, 2013, 338755.

Ekiz H., *Mantık Devreleri*, 3. Baskı, Değişim Yayınları, 33-38, 2003.

Gacar A., FPGA Tabanlı Görüntü İşleme Arabirimi, Yüksek Lisans Tezi, Ege Üniversitesi, Fen Bilimleri Enstitüsü, İzmir, 2009, 256805.

Gentsos C., Sotiropoulou C. L.,Vassiliadis N., Real-Time Canny Edge Detection Parallel Implementation for Fpgas, *17th IEEE International Conference on Electronics, Circuits, and Systems*, Athens, Greece, 12-15 December 2010.

Gonzalez R. C., Woods E. C., *Digital Image Processing*, 3rd ed., Prentice Hall, 23-25, 2008.

Guo Z., Xu W., Chai Z., Image Edge Detection Based on FPGA, *2010 Ninth International Symposium on Distributed Computing and Applications to Business Engineering and Science*, Hong Kong, 10-12 August 2010.

Hanumantharaju M. C., RaviShankar M., Rameshbabu D. R., Design and FPGA Implementation of an 2D Gaussian Surround Function With Reduced on-Chip Memory Utilization, *International Conference on Advances in Computing, Communications and Informatics*, Mysore, India, 22-25 August 2013.

Harinarayan R., Pannerselvam R., Ali M. M., Tripathi D. K., Feature Extraction of Digital Aerial Images by FPGA Based Implementation of Edge Detection Algorithms, *IEEE International Conference on Emerging Trends in Electrical and Computer Technology*, Tamil Nadu, India, 23-24 March 2011.

Irmak H., FPGA Üzerinde Gerçek Zamanlı Trafik İşareti Tanıma Sistemi, Yüksek Lisans Tezi, Orta Doğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2010, 269407.

Jack K., *Video Demystified*, 3rd ed., LLH Technology Publishing, USA, 17-21, 2001.

Jayaraman S., Esakkirajan, S., Veerakumar T., *Digital Image Processing*, 1st ed., Tata McGraw Hill Education Private Limited, New Delhi, 85-92, 2009.

Lin C., Sheu M., Liaw C., Chiang H., Fast First-Order Polynomials Convolution Interpolation for Real-Time Digital Image Reconstruction, *IEEE Transactions on Circuits and Systems for Video Technology*, 2010, **20**, 1260-1264.

Kanan C., Cottrell G. W., Color-To-Grayscale: Does The Method Matter in Image Recognition?, *PLoS ONE*, 2012, **7**, 842-846.

Kızılkaya R., Görüntü İşleme Algoritmalarının Sahada Programlanabilir Kapı Dizileri Çalışma Kartında Gerçeklenmesi, Dokuz Eylül Üniversitesi, Fen Bilimleri Enstitüsü, İzmir, 2011, 328340.

Kiran M., War K. M., Kuan L. M., Meng L. K., Kin L. W., Implementing Image Processing Algorithms Using Hardware in the Loop Approach for Xilinx FPGA, *Proceedings of the International Confereence on Electronic Design*, Penang, Malaysia, 1-3 December 2008.

Maini R., Aggarwal H., A Compherensive Review Of Image Enhancement Techniques, *Journal Of Computing*, 2010, **2**, 8-13.

Mayya M., Zarka N., Alkadi M. S., Embedded System for Real-Time Human Motion Detection, *Image Processing Theory Tools and Applications International Conference*, Paris, France, 7-10 July 2010.

Miljkovic O., Image Pre-processing Tool, *Kragujevac J. Math*, 2009, **32**, 97-107.

Nelson A. E., Implementation of Image Processing Algorithms on FPGA Hardware, Master of Science Thesis, Faculty of the Graduate School of Vanderbilt University, Nashville, USA, 2000.

Oflamaz C. U., FPGA Tabanlı Video Görüntü Çerçevelerini İyileştirme Cihazı Geliştirilmesi, Yüksek Lisans Tezi, Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2013, 346070.

Özcan A. R., Gerçek Zamanlı Lineer Görüntü İşleme Algoritmalarının FPGA İle Gerçeklenmesi, Lisans Tezi, Yıldız Teknik Üniversitesi, Elektrik-Elektronik Mühendisliği, İstanbul, 2009.

Özçelik F., Görüntü İşleme Algoritmalarının FPGA Üzerinde Gerçeklenmesi, Yüksek Lisans Tezi, Gazi Üniversitesi, Bilişim Enstitüsü, Ankara, 2012, 316619.

Özsaraç İ., Gerçek Zamanlı Sayısal Video Sabitlemenin FPGA Uygulaması, Yüksek Lisans Tezi, Orta Doğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2011, 286130.

Pratt W. K., *Digital Image Processing*, 4th Edition, John Wiley & Sons, New York, 455-465, 2007.

Ramirez J. M., et al., An FPGA-Based Architecture for Linear and Morphological Image Filtering, *20th International Conference on Electronics, Communications and Computer*, Cholula, Mexico, 22-24 February 2010.

Rodriguez M. A., Perez J. M., Gomez J. A., An FPGA-Based Implementation for Median Filter Meeting Real-Time Requirements of Automated Visual Inspection Systems, *10th Mediterranean Conference on Control and Automation*, Lisbon, Portugal, 9-12 July 2002.

Russo L. M., Pedrino E. C., Kato E., Roda V. O., Image Convolution Processing: A GPU Versus FPGA Comparison, *VIII Southern Conference on Programmable Logic*, Bento Goncalves, Brazil, 20-23 March 2012.

Şahin S., FPGA İle Yapay Sinir Ağının Donanımsal Gerçeklenmesi, Yüksek Lisans Tezi, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Kocaeli, 2004, 155100.

Sarıtaş E., Karataş S., *Her Yönüyle FPGA ve VHDL*, 1.Baskı, Palme Yayınları, Ankara, 12-17, 2013.

Szeliski R., *Computer Vision: Algorithms and Applications*, 1st ed., Springer, USA, 112-113, 2013.

Torres A. F., Jojoa J. D., Medina J., Image Convolution Processing: Design of 2-D Filters for Video Processing Using FPGAs, *XVIII Symposium of Image, Signal Processing and Artifical Vision*, Bogota, Colombia, 11-13 September 2013.

URL-1: Postgraduate courses for the electronics industry, [Online], http://www.ami.ac.uk/courses/ami4460_fpga/u02/, (Ziyaret Tarihi: 2 Aralık 2014).

URL-2: Xilinx [online], http://www.xilinx.com/applications.html, (Ziyaret Tarihi : 20 Kasım 2014).

URL-3: Xilinx [online], http://www.xilinx.com/support/university/boards-portfolio/xup-boards/XUPZedBoard.html, (Ziyaret Tarihi : 13 Aralık 2014).

Wang Z., Song H., Xiao H., He W., A Real-Time Small Moving Object Detection System Based on Infrared Image, *International Conference on Mechatronics and Automation*, Tianjin, China, 3-6 August 2014.

Wu J., Sun J., Liu W., Design And Implementation of Video Image Edge Detection System Based on FPGA, *International Conference on Image and Signal Processsing*, Yantai, China, 16-18 October 2010.

Yalçın H., FPGA Üzerinde Gerçek Zamanlı Trafik İşareti ve Tanıma, Yüksek Lisans Tezi, Orta Doğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2013, 340963.

KİŞİSEL YAYIN VE ESERLER

Altuncu M. A., Güven T., Becerikli Y., Şahin S., Real-Time System Implementation for Image Processing with Hardware/Software Co-Design on the Xilinx Zynq Platform, *2nd International Conference On Electrical And Electronics Engineering (ICEEE)*, Ankara, Türkiye, 28-29 April 2015.

Kır B., **Altuncu M. A.**, Şahin S., FPGA Based İmplementation of CORDIC Using Different Number Format, *International Conference On Technological Advances In Electrical Electronics And Computer Engineering (TAEECE)*, Konya, Türkiye, 9-11 May 2013.

ÖZGEÇMİŞ

27.04.1987 tarihinde Siirt'te doğan Mehmet Ali ALTUNCU, 2005 yılında Siirt Lisesinden mezun oldu ve 2006 yılında Sakarya Üniversitesi Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümünde lisans öğrenimine başladı. Dört yıllık lisans eğitimin ardından 2010 yılında mezun oldu. Aynı yıl Gümüşhane Üniversitesi Yazılım Mühendisliği Bölümü'ne araştırma görevlisi olarak atandı ve 8 ay burada çalıştıktan sonra 2011 yılında Siirt Üniversitesi Bilgisayar Mühendisliği Bölümü'ne ÖYP (Öğretim Üyesi Yetiştirme Programı) kapsamında araştırma görevlisi olarak atandı. 2011-2012 eğitim öğretim yılı, bahar yarıyılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalında yüksek lisans eğitimine başladı ve bu tarihten itibaren 35. Madde kapsamında Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak çalışmaktadır.