T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

VİDEO İŞLEME İÇİN FPGA TABANLI BİR DONANIM PLATFORMU

VECDİ EMRE LEVENT

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

DANIŞMAN
DOÇ. DR. FETHULLAH KARABİBER

İSTANBUL, 2015

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

VİDEO İŞLEME İÇİN FPGA TABANLI BİR DONANIM PLATFORMU

VECDİ EMRE LEVENT tarafından hazırlanan tez çalışması 17.06.2015 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Doç. Dr. Fethullah KARABİBER

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Doç. Dr. Fethullah KARABİBER

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Mehmet Fatih AMASYALI

Yıldız Teknik Üniversitesi _____

Doç. Dr. Hasan Fatih UĞURDAĞ

Özyeğin Üniversitesi _____

ÖNSÖZ

Hiçbir zaman desteklerini esirgemeyen, her zaman anlayışı, gerek akademik gerekse hayat tecrübeleri ile yanımda olan, değerli Danışman Hocam Sayın Doç. Dr. Fethullah Karabiber hocama teşekkürü bir borç bilirim.

Tezimin konusu olan FPGA tasarımına lisans eğitimi esnasında başlamama olanak sağlayan, gerek akademik gerekse hayat tecrübesini her zaman paylaşan Sayın Ferhat Canbay hocama teşekkür ederim.

Computational Intelligence Bioinformatics (CIB) çalışma grubundaki her çalışma arkadaşıma ayrıca teşekkürlerimi sunarım.

Son olarak her zaman yanımda olan aileme teşekkür ederim.

Haziran, 2015

Vecdi Emre LEVENT

İÇİNDEKİLER

Sayfa

BÖLÜM 4

BÖLÜM 5

BÖLÜM 6

KISALTMA LİSTESİ

ARP	Adres Çözümleme Protokolü
ASIC	Uygulamaya Özel Tümleşik Devre
BUFG	Saat çevrim Tamponlayıcısı
BUFPLL	PLL Tamponlayıcısı
CLK	Saat
DCM	Dijital Saat Yöneticisi
DSP	Dijital Sinyal İşleyici
EDID	Genişletilmiş Görüntü Kimlik Verisi
FPGA	Alanda Programlanabilir Kapı Dizileri
GPU	Grafik İşleme Birimi
HDL	Donanım Tanımlama Dili
HDMI	Yüksek Çözünürlüklü Çokluortam Arayüzü
HDP	Takılma Tespit Edici
LUT	Başvuru Çizelgesi
JPEG	Birleşik Fotoğraf Uzmanları Grubu
MAC	Ortam Erişim Yönetimi
PHY	Fiziksel Katman
PLL	Faz Kilitli Döngüler
RAM	Rasgele Erişim Bellek
REG	Saklayıcı
RGB	Kırmızı, Yeşil, Mavi
RST	Reset
RST_N	Negatif Reset
RTL	Saklayıcı Transfer Dili
TMDS	Aktarım Minimize Edilmiş Diferansiyel Sinyaller
USB	Evrensel Seri Veriyolu
VHDL	Yüksek Hızlı Tümleşik Devreler İçin Donanım Tanımlama Dili

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

VİDEO İŞLEME İÇİN FPGA TABANLI BİR DONANIM PLATFORMU

Vecdi Emre LEVENT

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Doç. Dr. Fethullah KARABİBER

Günümüzde yüksek işlem gücü gerektiren uygulamalarda, donanımsal olarak sabit mimariye sahip olan mikrokontrolörler, bilgisayarlar tek başlarına ihtiyaç duyulan performansı verememektedir. Sabit donanım yapısına sahip olan işlemcileri kullanmak yerine uygulamaya özgü geliştirilen çip tasarımları gerçeklemesi performans bakımından ihtiyaca cevap verebilmektedir. Çip tasarımlarında standart mimariden farklı olarak merkez işlem birimi bulunmaz. Bunun yerine çipin içerisindeki farklı bölgelerde tasarımcı tarafından belirtilmiş olan donanımlar çalışır. Dolayısıyla FPGA içerisinde paralel çalışan donanımların hesaplama gücü karşısında sabit mimarinin rekabet etme şansı olmamaktadır.

Özellikle görüntü işleme algoritmalarında, çözünürlük ve saniye başına işlenmesi gereken kare sayısı ile doğru orantılı olarak yüksek işlem gücüne ihtiyaç duyulmaktadır. Bunun gibi durumlarda sabit donanıma sahip olan mimariler ancak gerçek zamanlı olmayan, yani daha önceden kayıtlı olan bir video veya görüntünün işlenmesini gerçekleştirebilmektedir. Ancak gerçek zamanlı hesaplama ihtiyacı olduğunda, sabit mimariler ardışık gelen karelerin üzerinde koşması gereken algoritmayı gerekli zaman aralığında çalıştıramayacak ve sistem gerçek zamanlı özelliğini yitirecektir. FPGA üzerinde koşacak olan algoritmanın en fazla kazanç sağlayabilmesi, algoritmadaki akışta ne kadar paralel işlem yapmaya müsait olduğu ile doğru orantılıdır.

Bu çalışmada bir görüntü işleme sistemi iskeleti sunulmaktadır. Görüntü işleme uygulamalarında yaygın olarak kullanılan ortanca, gauss, sobel gibi filtrelerin donanım uygulamaları gerçeklemesi yapılmıştır. Bu sistem hem FPGA hem de işlemci ile birlikte çalışan heterojen bir yapıdadır. Sistem, işlemci ile işlenecek olan görüntünün ve hangi algoritma koşacağının bilgilerini FPGA'e aktarır, FPGA üzerinde işlenmiş olan görüntü, işlemciye geri aktarılmaktadır. Tasarımın bu yapıda olması sistemin yeniden kullanılabilirliğini arttırmakta ve başka sistemler için altyapı özelliği taşımaktadır. Tezde kullanılan UDP, ARP, DHCP protokolleri, HDMI yakalama ve aktarım yöntemi, EDID veri akışı gibi yapıların detayları paylaşılmıştır. Tasarımda verilen modüllerin çalışma prensipleri hakkında detaylıca bilgi verilmiştir. Bu bilgiler referans alındığında, verilen modüllerin diğer sistemlere entegrasyon hızı artacaktır.

Bu çalışma üzerinde Xilinx Spartan6 serisi LX-45 modeli FPGA bulunan Digilent firmasının Atlys FPGA geliştirme kartı kullanılarak bir görüntü işleme sistemi geliştirilmiştir. Geliştirme kartının üzerinde dahili olarak bulunan HDMI, Ethernet ve DRAM arayüzleri kullanılmıştır. Sistemin akışında öncelikle HDMI portu üzerinden alınan yüksek çözünürlüklü görüntü, görüntü işleme modülüne aktarımı gerçekleştirilmektedir. Görüntü işleme modülünden çıkan veriler, görüntü sıkıştırma modülüne aktarılmaktadırlar. Görüntü sıkıştırma, görüntünün işlemciye aktarılması esnasında daha az bant genişliği kullanması için önemlidir. Bu işlemden sonra sıkıştırılmış görüntünün DRAM üzerinde tamponlanma işlemi gerçekleşmektedir. Son aşamada ise görüntü Ethernet modülü üzerinden işlemciye aktarım sağlanmaktadır. İşlemci tarafında ise geliştirilen .Net uygulaması ile görüntünün alınması, ekrana gösterilmesi ve FPGA üzerinde koşacak algoritmanın seçilebildiği bir arayüz kullanıcıya sunulmuştur.

Geliştirilen sistem üzerinde farklı filtreler uygulanarak performansı ölçülmüştür. Bu ölçümler Matlab, OpenCV ve GPU üzerinde gerçeklenen uygulamaların performansları geliştirilen sistemin performansı ile karşılaştırılarak sistemin başarımı gösterilmiştir.

Anahtar Kelimeler: Görüntü İşleme, Çip Tasarımı, Gerçek Zamanlı Sistemler, Heterojen Sistemler, FPGA

ABSTRACT

FPGA BASED A HARDWARE PLATFORM FOR VIDEO PROCESSING

Vecdi Emre LEVENT

Department of Computer Engineering

MSc. Thesis

Adviser: Assoc. Dr. Fethullah KARABİBER

Microcontrollers and computers, which have a fixed architecture, cannot provide the performance needed with their own for the applications that require high computing power. Instead of using the structure having fixed hardware processors, realization of application specific chip design can meet to needs in terms of performance. Unlike the standard architecture of the central processing unit does not exist on the chip design. Instead of this, hardware runs on different locations inside of chip which is specified by designer. Since FPGA has capability of parallel processing, it has big advantage over the fixed architecture in terms of computation power.

Especially on image processing algorithms, high processing power is required with directly in proportion of resolution and the number of frames to be processed per second. On these situations, fixed hardware architectures can process a previously stored video or image which is only non-real-time. When a video or image needs to processed on real-time, the system with fixed architecture cannot process it in necessary time scale of sequentially receiving frames and will lose real-time operational function. Gaining maximum calculating power is directly related to availability of parallelism in algorithms.

This study presents an image processing framework system. Widely used image processing filters such as median, gauss, sobel filters are implemented in a hardware platform. This system is a heterogeneous structure which can employ FPGA and processor. The system transfers the image to be processed and information including which algorithm will run. The processed image on FPGA is transferred back to the processor. This structural design increases reusability of the system and infrastructure functionality for other systems. In this study, some information about UDP, ARP, DHCP protocols, HDMI capturing and transmitting methods and EDID data flow are given. The designed modules are explained in great detail with the working principles. If designer takes this information as a reference, speed of development and integration will increase.

In this study, an image processing platform is developed using Digilent companies Atlys FPGA development board which has a Xilinx Spartan 6 series LX-45 model FPGA. HDMI, Ethernet and DRAM interfaces. These interfaces are built-in on the development board. Firstly in the system flow, high resolution images received via the HDMI port and transfers to image processing module. The processed image transferred to image compression module. Image compression is important for using less bandwidth during to transmission of image to the processor. After this operation, compressed image is buffered on DRAM. In the final stage, image is transferred back to processor through Ethernet module. On the processor side, received images displays on the screen and the some information about application, which run on FPGA, is provided with a user interface in a developed .Net application.

The performance was measured by applying commonly used filters on the developed system. The experimental results show that the developed hardware system has high performance computing than Matlab, OpenCV and GPU .

Keywords: Image Processing, Chip Design, Real Time Systems, Heterogeneous Systems, FPGA

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Günümüzde donanım ve bilgisayarın bir arada çalıştığı sistemler yaygınlaşmaktadır. Özellikle görüntü işleme alanında yüksek işlem gücü gerektiren algoritmalar sıkça kullanılmaktadır. Bu ihtiyaçlara FPGA adındaki içerisinde programlanabilir mantık kapıları bulunduran bütünleşik devreler ile bilgisayar bağlantılı sistemler cevap verebilmektedir. Yani hem işlemci hem de FPGA bir arada kullanılabilmektedir.

Yapılan araştırmalarda birçok görüntü işleme uygulamasının donanım uyarlaması, görüntü işleme sistemleri, heterojen sistemlerin incelemesi gerçekleştirildiği görülmektedir. İlk olarak incelenen donanım iskeleti yapısı, kendisine verilen bir algoritmayı, aynı sonuçları veren parametrize ve optimize edilmiş olan donanımı veren bir yapıdır[1][2][3][4]. Bu yapının FPGA üzerinde görüntü işleme üzerine kullanılabilmektedir. Verilen bir görüntü işleme algoritmasının daha yüksek frekans ve alan kullanımının azaltılmasında etkilidir.

Matlab Simulink, Labview gibi uygulamalar ile FPGA donanım sentezi yapılabilmektedir. Bu uygulamalarda yüksek seviye dillerde yazıldıkları için birçok işlem donanım sentezleyiciye bırakılmaktadır[5][6]. Bu yüzden gerek alan gerek hız yönünden başarılı sonuçlar ortaya koyamamaktadır.

Bir başka tasarım yaklaşımı ise yüksek seviyeli dillerden donanım sentezi gerçekleştirmektir. Yüksek seviyeli diller C, C++ gibi diller ile belli kısıtlar çerçevesinde geliştirilen uygulamaların otomatik olarak algoritmayı analiz ederek Verilog, VHDL gibi donanım tanımlama dillerine dönüştürülmesi sağlanabilmektedir. Bu yaklaşım yüksek

seviyelin sahip olduğu, yeniden kullanılabilirlik, soyutlama, hızlı geliştirme avantajlarını donanım tasarımı için uygun hale getirmektedir. Handel C bu yapıda çalışan yüksek seviyeli bir dildir[7][8]. Ancak bu tür yapılarda RTL kod oluşturma işlemi yüksek seviyeli bir dilden yapıldığı için alan ve hız kayıpları olacaktır. Bu noktada ya yüksek seviyeli dillerin getirdiği avantajlar ya da el ile tasarlandığından ötürü daha yavaş geliştirme hızında ancak daha hızlı ve az alan kullanan tasarımlar tercih edilmelidir.

FPGA üzerinde hızlı tasarım geliştirilmesine olanak sağlayan bir başka yazılım ise Toronto Üniversitesinde geliştirilen LegUp ismindeki yazılımdır[9][10]. Bu yazılım açık kaynak kodlu olup, C dilinden Verilog diline dönüştürülmesini sağlamaktadır.

West of England Üniversitesinde geliştirilen FPGA tabanlı video aktarım yöntemi sistemi bulunmaktadır[11]. Bu sistem HDMI üzerinden almış olduğu görüntüyü USB portu üzerinden hedef cihaza aktarmaktadır. Geliştirdikleri tasarımda FPGA, bilgisayara bağlı bir web kamera olarak gözükmekte, HDMI'dan alınan görüntüler ekrana gösterilmektedir.

FPGA içerisindeki zamanlamayı minimize etmek için çok kanallı iskelet sistemi sunulmuştur[12]. Orijinal ASIC ile karşılaştırılabilecek kadar iyileştirme yapılmaktadır. Çalışma zamanında FPGA'in programlanması, uygulama bakımından çeşitliliği arttırmaktadır. Uygulama renk uzay dönüşümü motoru(CSC) ile paralelliğin arttırılması ile performansın ciddi bir şekilde artacağını göstermiştir.

Gudis ve arkadaşları[13] tarafından yapılan çalışmada, gerçek zamanlı gömülü uygulamalar için düşük güç tüketimli ve yüksek performans sağlayan, stereo algoritmalara uygun FPGA gerçeklemesi sunulmuştur. Yüksek çözünürlük ve saniye başına kare sayısı uygulamanın ihtiyaçlarına göre düzenlenebilmektedir. Yapılan çalışmada 640x480 çözünürlüğünde 30 FPS ile stereo görüntü işleme gerçekleştirilmiştir. Aynı tasarımın GPU üzerinde koşturulduğunda, GPU'a göre 10 kat daha az güç tüketimi, daha az gecikme ile çalıştığı gösterilmiştir.

Bu tez kapsamında son olarak California Üniversitesinde geliştirilen RIFFA adındaki araç incelenmiştir[11][12]. Bu araç üzerinde PCI Express bağlantısı olan FPGA'ler ile kullanılabilmektedir. Yönetici işlemciden veriyi FPGA'e PCI Express kanalı ile göndermektedir. Ayrıca yine kendi geliştirmiş oldukları işlemci tarafında koşan işletim

sistemi için sürücüler sunmaktadırlar. Bu sürücüler Windows, Linux için, Xilinx ve Altera firmalarının ürettiği PCI Express çekirdekler ile uyumluluk ve C/C++, Phyton, Matlab ve Java için kütüphaneler sunmaktadırlar. Genel bir iskelet sistemi sunmaktadırlar. Kullanıcı, koşturmak istediği tasarımını RIFFA modulu ile gerekli bağlantılarını kurması halinde FPGA üzerinden çıktı verileri almaktadır. Genel bir sistem olmasından ötürü kullanılan alan el ile tasarlanan uygulamaya özgü modüllerden daha fazla olmaktadır.

1.2 Tezin Amacı

Görüntü işleme algoritmaları genellikle yüksek işlem hesaplama gücü gerektirmektedirler. Özellikle görüntü çözünürlüğü, saniye başına kare sayısı değerleri yükseldikçe seri çalışan işlemci gibi mimariler ile bu hesaplama gücü ihtiyacını karşılanamamaktadır. Günümüz modern ekran kartlarında bulunan Grafik İşleme Birimi (GPU) paralel hesaplama yeteneğine sahiptir. Ancak sahip olduğu donanım yapısı her amaç ile kullanılabileceği düşünülerek tasarlanmış ve kullanıcı tarafından değiştirilemeyen sabit bir yapıdadır. Her ne kadar paralel işlem yaparak işlemciye göre çok daha yüksek işlem gücü sunsa da yukarıda belirtilen sebeplerden ötürü, çok daha fazla işlem gücü gerektiren uygulamalarda yetersiz kalmaktadır. Bu durumda FPGA çözüm olmaktadır. Bu yapı içerisinde birçok programlanabilir mantık kapıları ve bunlar arasında bağlantıları barındırır. Bu bağlantıların ne şekilde olacağı tasarımcı tarafından belirlenmektedir. Bu özelliği ile sabit yapıda olan işlemci ve GPU gibi mimarilerin önüne geçmektedir. Çünkü gerçekleştirilmek istenen uygulamaya özel tasarım yapılabilmektedir.

Bu çalışmada bilgisayar ve donanımın bir arada çalışacağı bir görüntü işleme iskelet sistemi sunulmaktadır. Çeşitli görüntü işleme algoritmaları FPGA üzerinde hesaplama yapabilmesi için tasarlanmış, bilgisayar üzerinden istenen görüntü FPGA'e gönderilerek işlenmiş olan görüntünün alınması gerçekleştirilmiştir. Bu yapının önemi bilgisayar ile FPGA'in bir arada kullanılabilmesidir. FPGA tasarımından bağımsız olarak sadece gerekli haberleşme protokollerine uyarak, kullanıcı FPGA üzerindeki görüntü işleme algoritmalarının çıktılarını alabilmektedir. Bu sistemlerin başka sistemlere entegrasyonu ve ilgili modülün çalışma prensibinin anlaşılması için gerekli tüm bilgiler alt başlıklarda bulunmaktadır.

Bu kapsamda kullanılan geliştirme kartının üzerindeki HDMI, DRAM ve Ethernet arayüzleri kullanılmıştır. Kullanıcının bilgisayar yazılımı aracılığı ile Ethernet portu üzerinden seçmiş olduğu görüntü algoritmasının ne olduğunu belirleyebilmektedir. Seçim işleminden sonra HDMI portu üzerinden alınan yüksek çözünürlüklü görüntü, öncelikle seçilmiş olan görüntü işleme modülü tarafından işlenmeye başlanır. Bu modülden çıkan görüntüler bilgisayara aktarım esnasında daha az bant genişliği kullanılması amacıyla JPEG sıkıştırma modülüne gönderilmektedirler. Sıkıştırılmış olan görüntü verileri bilgisayara gönderilmeden önce DRAM üzerinde tamponlanmaktadırlar. Bunun gerekliliği, her an hedef cihaza aktarımın mümkün olmamasıdır. Özellikle bilgisayarlarda işletim sistemi koşmasından dolayı, zamanlama kesin olamamaktadır. Hedef cihaz ile Ethernet portu üzerinden anlaşma yapılarak veri kullanıcının hedef cihazına aktarılmaktadır. Ethernet portu üzerinden gelen veriler bilgisayar ekranına gerçekleştirilen .NET uygulaması aracılığı ile verilmektedir.

Tez çalışmasının Bölüm 2'de FPGA'in yapısı, hangi diler aracılığı ile nasıl programlanabileceği, donanım sentezinin ne anlama geldiği, geliştirilen bir tasarımın doğrulamasının nasıl yapılabileceği ve bu çalışmada kullanılan FPGA geliştirme kartından bahsedilmiştir. Bölüm 3'te HDMI portu üzerinden görüntünün nasıl yakalandığı, aktarımının nasıl yapıldığı, cihazlar arasında görüntü aktarımından önce çalışan veri değişimleri, sıkıştırma işlemini gerçekleştiren modülün içyapısı ve hangi adımlardan oluştuğu gösterilmiştir. Bölüm 4'te görüntü işleme modüllerinin donanımsal olarak nasıl tasarlandıkları gösterilmiştir. Bölüm 5'de ise görüntü aktarım işleminde ihtiyaç duyulan protokollerden ve sistem üzerinde kullanımları gösterilmiştir. Son olarak Bölüm 6 olan sonuç bölümünde, sistemin gecikmesi, FPGA üzerinde kullandığı kaynaklar verilmiştir.

1.3 Hipotez

Tez kapsamında geliştirilen sistemin yapısı detaylı bir biçimde verilmiştir. Bu sistemin üzerine ekleme çıkartma yapılması oldukça kolaydır. Dolayısıyla geliştirilen sistem çok farklı görüntü işleme algoritmaları ile kullanmak mümkündür. Bu da sistemin tekrar kullanılabilirlik özelliğinden kaynaklanmaktadır. FPGA tasarımlarında, diğer platformlara göre tasarlanma zamanı çok daha yüksektir. Sunulan sistemin sahip olduğu donanımların oluşturulması oldukça önemli bir zaman almaktadır. Tasarım zamanı sebebi ile pek çok

kullanıcı yapacakları işlemleri mecburen daha düşük performanslı platformlarda yapmak durumunda kalmaktadırlar. Bu sistem geliştiricilerin kullanımına açık olup, baştan tasarlama işlemindeki zaman maliyetini indirgemektedir.

Tez çalışmasında sıradaki cümle hipotez olarak temel alınmıştır. Görüntü işleme algoritmaları için işlemci, GPU gibi sabit donanıma sahip mimarilerin hız getirilerinin yeterli olmadığı durumlarda uygulamaya özgü tasarım yapılabilen paralel çalışma yeteneği olan FPGA ile tasarlanmış olan donanım bilgisayar iskelet sisteminin kullanılması, uygulama geliştirme hızının artmasına, FPGA tasarımı hakkında bilgisi olmayan bir tasarımcının sistemin hız avantajından faydalanabilmesine, tekrar kullanılabilirliğin artmasına katkı sağlar.

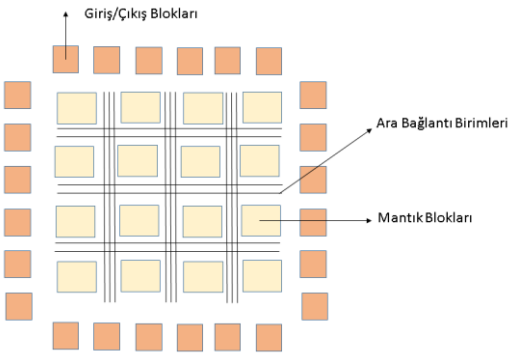
BÖLÜM 2

SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ

Bu bölümde Sahada Programlanabilir Kapı Dizilerinin(FPGA) ne oldukları, neden ve hangi durumlarda tercih edildiği, diğer hesaplama donanımlarından farklarının neler oldukları, donanım tanımlama dilleri, sentezleme araçları ve kullanılan geliştirme kartı hakkında bilgi verilecektir.

2.1 FGPA Yapısı

FPGA(Field Programmable Gate Array) yani Alanda Programlanabilir Kapı Dizileri içerisinde programlanabilir mantık bloklarını içeren tümleşik bir devredir. İşlemcinin çalıştığı seri hesaplama mantığı ile çalışmaz. Paralel hesaplamaya uygun olan her türlü alanda kullanılabilir. Çünkü esas hesaplama gücü paralel yapılması gereken işlemlerden gelmektedir. Şekil 2.1'de verilen FPGA iç yapısında göründüğü üzere, mantık kapıları arasında birçok bağlantı vardır. Birbirleri ile bağlantılarının nasıl olacağı tasarımcı tarafından belirlenmektedir.

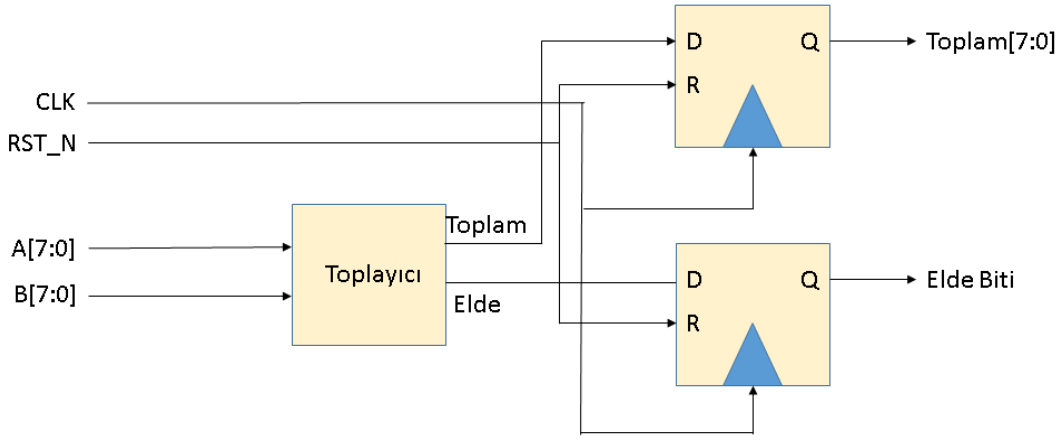


Şekil 2.1 FPGA İç Yapısı

2.2 Donanım Tanımlama Dilleri

Çeşitli elektronik devrelerin, bağlantılarının nasıl olacağının tanımlamak için kullanılan dillerdir. Ortaya çıkartılmak istenen devre, çeşitli donanım tanımlama dilleri aracılığı(HDL) ile sentezleyici araçları tarafından oluşturulur. Bu oluşturulan devrelerin kontrolü ise simülasyon araçları tarafından gerçekleştirilmektedir. Gerçeklemek istenen donanımın doğrudan devre elemanlarını bir araya getirmek yerine, donanım tanımlama dilleri ile yapmak tercih edilmektedir. Çünkü devrelerin boyutu büyüdükçe doğrudan devre elemanları ile tasarım yapmak karmaşıklaşmaktadır. Donanım tanımlama dilleri daha üst seviye üzerinden tasarım yapmaya imkân vermektedir. Bu durum, bir bilgisayar uygulaması geliştirilirken nasıl makine kodu veya C dili seçilebileceği gibidir. Makine kodu seçildiği zaman her şeye hakim olunmakta ancak geliştirme zamanı artmaktadır. Aynı şekilde c dili ile geliştirilme yapınca, bazı şeyler derleyicinin inisiyatifine bırakılmakta ama geliştirme zamanı düşmektedir. HDL ile geliştirilme yapılması geliştirme süresinin azaltılmasına, dikkatli yazılmadığı zaman hız kaybı ve alan artmasına sebebiyet vermektedir. Burada bilgisayar örneğinde olduğu gibi bazı şeyler sentezleme araçlarının inisiyatifindedir.

Yaygın olarak kullanılan iki adet HDL bulunmaktadır. Bunlar Verilog ve VHDL dilleridir. Her ikisini de kullanarak aynı devre ortaya çıkartılabilir. Ancak birbirlerine göre bazı avantajları ve dezavantajları bulunmaktadır. Verilog dilinde bulunan veri tipleri VHDL diline göre daha basittir. Verilog dilinde kullanıcının kullanabildiği iki tür REG ve WIRE adında veri tipler bulunur. REG saklayıcı yani herhangi bir sinyale senkron bir biçimde atama yapılan anlamında, WIRE ise üzerine sürekli atama yapılan kablo anlamına gelmektedir. Bu sadelik özelliği sebebi ile Verilog tercih edilebilir. Ancak VHDL dilinde kullanıcının kendi oluşturabildiği özel veri tipleri bulunmaktadır. Ayrıca Verilog dilinde bulunmayan kütüphane yapısı, VHDL dilinde bulunmaktadır. Bu diller geliştirilecek olan kendisini tanımlamak için ya da simülasyon için kullanılmaktadır. Kendisini tanımlamak için geliştirildiğinde kodun sentezlenebilir olması gerekmektedir. Bu koda RTL(Saklayıcı Transfer Dili) adı verilmektedir. Şekil 2.2'de örnek toplayıcı devresi ve Şekil 2.3'te bu devrenin Verilog ve VHDL dilleri ile gösterimi verilmiştir.



Şekil 2.2 Örnek Toplama Devresi

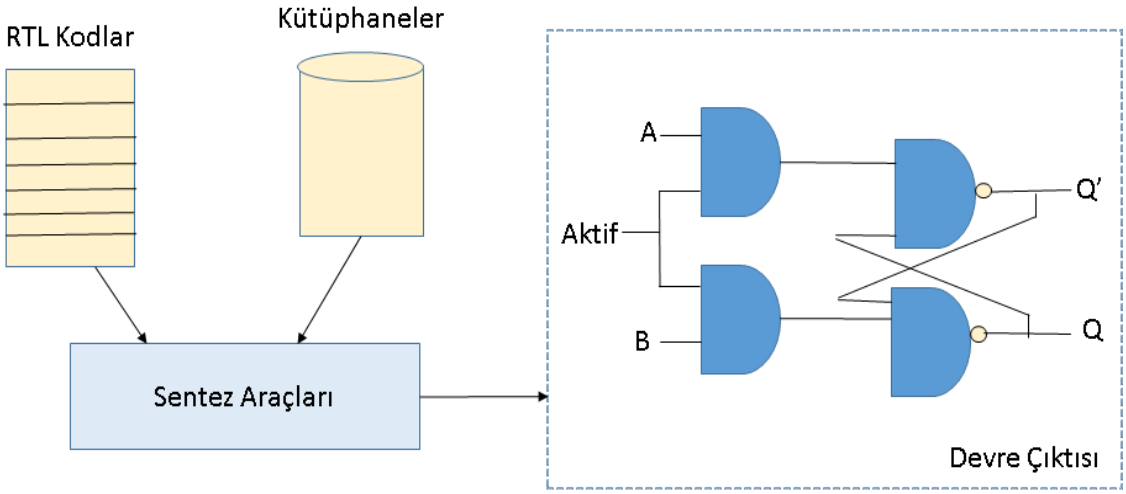
```
module adder(
                input[7:0] A,
                input[7:0] B,
                input clk,
                input rst_n,
                output reg[7:0] sum,
                output reg carry);

                always@(posedge clk) begin
                                if(!rst_n)
                                  {carry, sum} <= 0;
                                else
                                  {carry,sum} <= A + B;
                end

endmodule
```

```
LIBRARY ieee:
USE ieee std_logic_1164 all;
USE ieee.std_logic_arith.all;
USE ieee.STD_LOGIC_UNSIGNED.a1l:

ENTITY adder IS
PORT(
                A : IN std_logic_vector ( 7 DOWNTO 0);
                B : IN std_logic_vector ( 7 DOWNTO 0);
                clk : IN std_logic;
                cst_n : IN std_logic;
                carry : OUT std_logic;
                cum : OUT std_logic_vector ( 7 DOWNTO 0)
);
END adder :

ARCHITECTURE rtl OF adder IS
                signal sum_int : std_logic_vector (B downto U):
                signal AB : std_logic_vector (B downto U):
                signal BB : std_logic_vector (B downto U):

BEGIN

A8 <= "0" & A;
B8 <= "0" & B;
sum_int <= A8 + BB;

adder: process (clk)
begin
                if rst_n = '0' then
                                carry <= '0';
                                sum <= "UUUUUUUU";
                elsif clk'event and clk = '1' then
                                carry <= sum_int(B);
                                sum <= sum_int(7 downto 0):
                end if:
end process adder;
END ARCHITECTURE rtl;
```

Şekil 2.3 Toplayıcı Devresi için Verilog(Sağdaki) ve VHDL(Soldaki) Örnek Kodları[16]

2.3 Donanım Sentezi

Devre tasarımları önceden el ile şematik tasarımlar ile gerçekleştirilmekteydi. Devreyi optimize etmek için Karnaugh haritaları kullanılıp şematik tasarım çizilirdi. Bu yaklaşım devrelerin birkaç yüz elemandan oluştuklarında başarılı sonuçlar vermekteydi ancak devrenin karmaşıklığı arttıkça bu yöntem el ile yapılamaz bir hal almaktaydı. Günümüzde devre tanımları donanım tanımlama dilleri aracılığı ile yapılmaktadır. Yapılan tasarım sonunda ortaya çıkan kodun, devre dönüşümünün gerçekleştirilmesi gerekmektedir. Bu dönüşüm sentez araçları kullanılarak gerçekleşmektedir[17][18]. Sentez araçları çeşitli algoritmalar kullanarak devrenin performansını en üste çekmeyi hedefler. Çeşitli sentez araçları tasarımcılara performans hedefi için hız, alan, güç gibi seçenekler sunmaktadırlar. Bu araçların ortaya çıkarttığı ve, veya, veya değil, toplayıcı, seçici gibi mantık elamanlardan oluşan devre çıkartmaktadırlar.

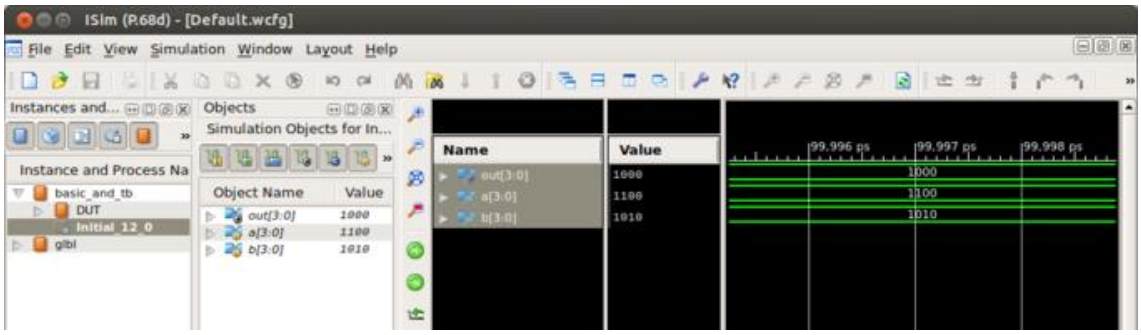


Şekil 2.4 RTL'den Devre Sentezlenmesi

Yüksek seviyede bir dilden çevirmenin en büyük avantajı insanın yapacağı hataları azaltmak ve tasarımın yüksek seviye olarak soyutlarlar. Bu da tasarımın yeniden kullanılabilirliğini arttırıp, tasarım için harcanan zamanı azaltırlar.

2.4 Simülasyon

Geliştirilen donanımın doğrulanması çok önemlidir. Simülasyon, donanımda ortaya çıkabilecek birçok tasarım hatalarını FPGA üzerinde çalıştırılmadan anlaşılmasını sağlamaktadır. FPGA üzerinde hata ayıklama yapmak çok zordur. Çünkü bilgisayar ortamındaki gibi bir konsol ekranı yoktur, ayrıca bilgisayar uygulamalarının derlenmesi ile kodun sentezlenme süresi karşılaştırıldığında oldukça uzundur. Bu sebeplerden ötürü öncelikte çeşitli simülasyon yazılımları üzerinde doğrulama yapılır. Bu çalışmada Xilinx firmasının sunduğu ISE programının içerisinde bulunan ISIm adındaki simülasyon aracı kullanılmıştır.

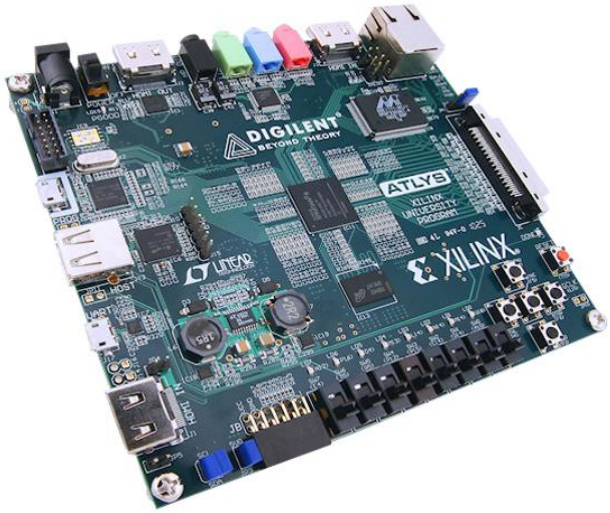


Şekil 2.5 ISIm Test Bankı Arayüzü[19]

Bu uygulamada tasarımdaki seçilen sinyallerin her bir saat çevriminde üzerine atanan değerler görüntülenebilmektedir. İstenenden farklı bir atamanın yapıldığının görüldüğü durumda tasarıma tekrar gidilerek gerekli düzenlemeler yapılır, ardından tekrar simüle edilir. Bu yaklaşım az sayıda sinyal ve birkaç saat çevrimi izlenecekse başarılı olmaktadır. Ancak yüzlerce sinyalin olduğu, milyonlarca saat çevriminin izlenmesi gereken bir durumda gözle takip edilemez olacaktır. Bu durumda da test bankı kodları denen Verilog veya VHDL dilinde yazılmış kodlara ihtiyaç vardır. Bu kodlar tasarımın nasıl davranacağını bilen, tasarıma uygun sinyalleri besleyerek, tasarımdan çıkan sinyalleri olması gereken sinyaller ile karşılaştırırlar. Karşılaştırma istenen saat çevrimi sayısına kadar yapılır, hangi saat çevriminde hatalı sonucun, olması gereken sonucun gösterilir. Tasarımcı ilgili saat çevrimindeki sinyalleri kontrol ederek tasarımda değişikliklerini tamamlar.

2.5 FPGA Geliştirme Kartı

FPGA piyasasının büyük bir çoğunluğunu oluşturan iki adet FPGA üreticisi şirket bulunmaktadır. Bu şirketler Xilinx ve Alteradır. Tez kapsamında Xilinx firmasının bir FPGA çipi kullanılmıştır. Xilinx firması farklı hız ve boyutlarda FPGA'ler üretmektedir. Çalışmada Digilent firmasının sunduğu Atlys adındaki geliştirme kartı üzerinde çalışılmıştır. Bu kartın üzerinde birçok arayüz bulunmaktadır. Spartan 6 LX45 modelinde bir FPGA bulunmaktadır. Bu FPGA Xilinx firmasının ürettiği orta düzeyde sayılabilecek türden bir FPGA'dir. Üzerinde bulunan arayüzlerden ise HDMI giriş/çıkış, Ethernet ve Dram kontrolörü kullanılmıştır.

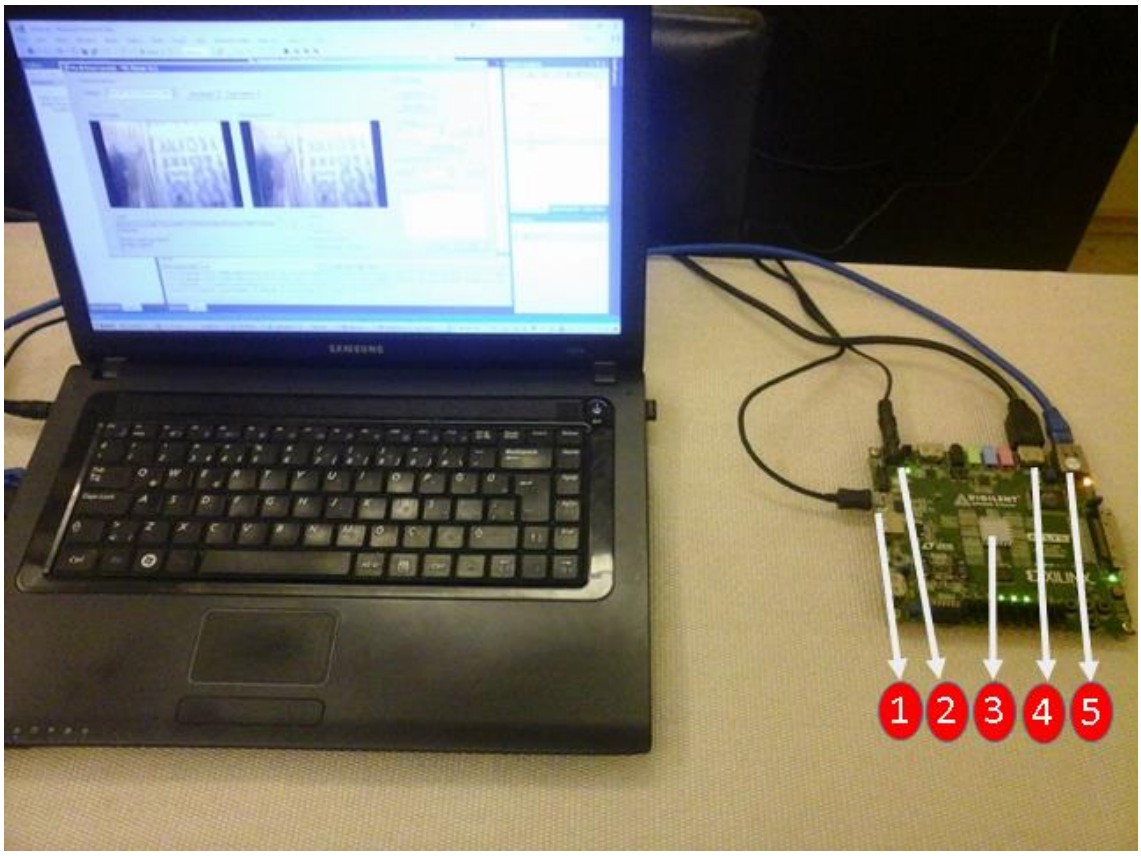


Şekil 2.6 Digilent Atlys Geliştirme Kartı

Spartan-6 LX45 yüksek performanslı mantık için optimize edilmiştir. Sunduğu özellikler aşağıda verilmiştir[20].

- 6,822 adet herbiri4 adet 6-giriş LUT ve 8 adet flip-flop içeren dilimler
- 2.1Mbit hızında blok RAM
- 4 adet Saat çevrimi düzenleyicisi (8 DCM ve 4 PLL)
- 6 adet faz kilitli döngüler
- 58 DSP dilimleri
- 500MHz+ saat hızları
- Bir adet10/100/1000 Ethernet PHY içinRJ-45 konektör ve RS-232 seri bağlantısı

- İki adet HDMI video giriş portu ve iki adet HDMI çıkış portu
- Programlama ve veri transferi için kartın üzerinde dahili olarak iki adet kartın üzerinde USB2



Şekil 2.7 Sistemin Bilgisayar Bağlantısı

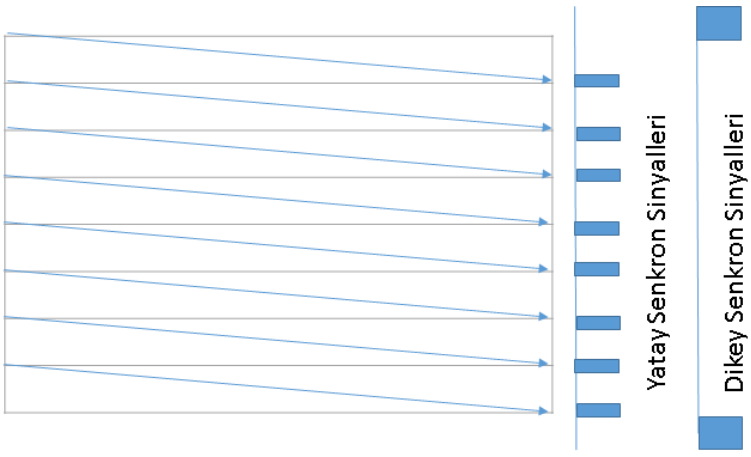
Şekil 2.7 de sistemin bilgisayara olan bağlantısı verilmiştir. Şekilde 1 numaralı bağlantı FPGA programlaması için kullanılan soketi, 2 numaralı bağlantıda FPGA güç girişi, 3 numarada ise FPGA, 4 numarada HDMI soketi ve 5 numarada Ethernet bağlantı soketi bulunmaktadır. HDMI portu görüntünün aktarımı için, Ethernet portu görüntünün alımı ve çeşitli parametrelerin gönderilmesi için, son olarak USB portuna takılan seri port kablosu ile sentezlenmiş olan tasarımın FPGA'e aktarımının yapılmasını sağlamaktadır.

BÖLÜM 3

GÖRÜNTÜ YAKALAMA VE SIKIŞTIRMA

Günümüzde birçok video sinyali kodlaması analogdan dijital alana geçmektedir. Dijital sinyallerin, analog sinyallere göre en büyük avantajı aktarım esnasında değişime uğramamalarıdır. Analog sinyallerin gürültü ve kayıplar olmaksızın aktarım yapılması mümkün olmamaktaydı. Dolayısıyla az kayıplı ya da kayıpsız aktarmak için dijital aktarım yöntemleri tercih edilmektedir.

Dijital video verileri, format fark etmeksizin ayrık resim görüntülerine bölünmektedir. Bu bölünen her bir resme "frame" yani kare adı verilmektedir. Bir karede tarama sırası, piksellerin geliş sırası, soldan sağa doğru olmaktadır. Her satırda, satırın dolduğunun anlaşılması için dikey senkron sinyali aktif olur. Bu sinyal geldikten sonra bir sonraki satırın başından devam edilir. Karedeki her satır dolduğunda ise yatay senkron sinyali gelerek bir sonraki karenin işlemine başlanacağı anlaşılır.

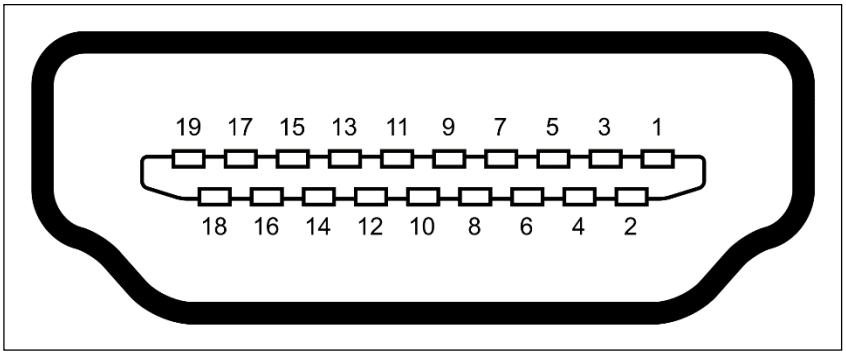


Şekil 3.1 Görüntü İletimi ve Yatay/Dikey Senkron Sinyalleri

Şekil 3.1'de piksellerin geliş sırası, yatay/dikey sinyalleri gösterilmiştir. Bir saniye içerisinde aktarımı yapılan kare sayısı FPS(Frame per Second) olarak tanımlanmaktadır. Bu gelen yatay senkron sinyalleri sayılarak elde edilmektedir.

3.1 Alıcı Modülü

Çalışmada HDMI(High-Definition Multimedia Interface) ara yüzünden alınan görüntüler geliştirme kartına beslenmiştir. HDMI çok geniş alanlarda kullanılan, ses ve görüntüyü dijital olarak sıkıştırmadan tek kablo üzerinden gönderen gelişmiş bir arabirimdir. Birçok öncü elektronik firması, 2002 yılında bir araya gelerek ortaya çıkartmıştır. HDMI DVI arayüzündeki sistemi kullanmaktadır. Bunlara ek olarak HDMI ses taşıma özelliğine sahiptir. HDMI gelişmekte olan bir arayüzdür. Günümüzde 2.0 versiyonu bulunmaktadır. Çalışmamızda ise 1.3 versiyonu kullanılmıştır. 1.3 versiyonu maksimum 10.2 Gbps'a kadar desteklemektedir.



Şekil 3.2 HDMI 1.3 Pin Sıralaması

HDMI ile görüntü aktarımı için aktarılacak olan sinyalin kodlanması gerekmektedir. Kodlama TMDS(Transition-minimized differential signaling) tekniği ile gerçekleştirilmektedir. Son olarak video akışının başlayabilmesi için EDID(Extended Display Identification Data) veri değişiminin yapılması gerekmektedir.

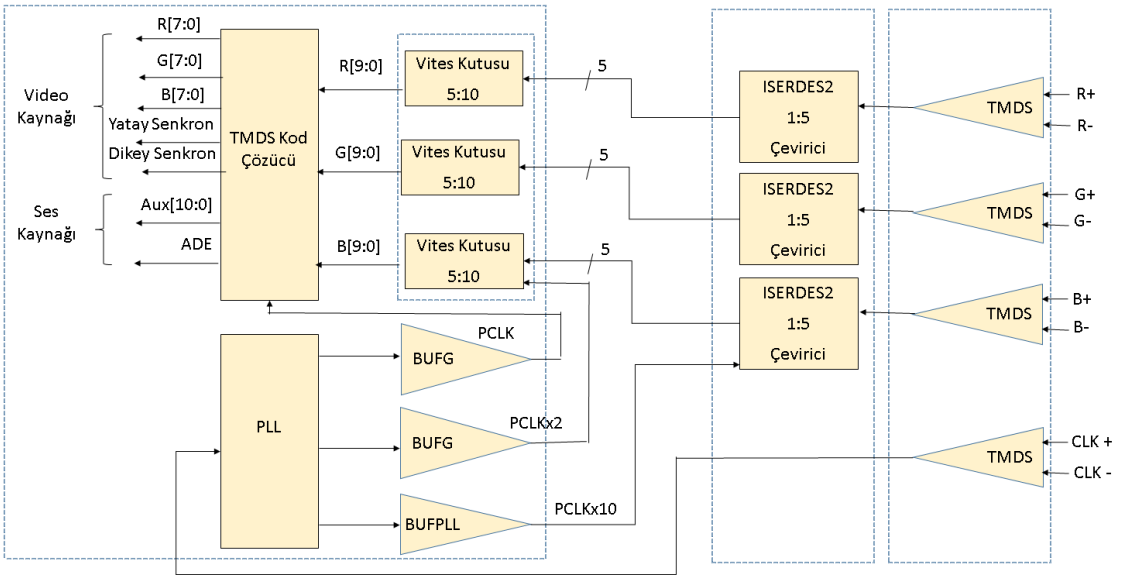
Çizelge 3.1 HDMI 1.3 Pin Açıklamaları[20][21]

Pinler	Açıklama
Pin1	TMDS Veri2+
Pin2	TMDS Veri2 Toprak
Pin3	TMDS Veri2-
Pin4	TMDS Veri1+
Pin5	TMDS Veri1 Toprak
Pin6	TMDS Veri1-
Pin7	TMDS Veri0+
Pin8	TMDS Veri0 Toprak
Pin9	TMDS Veri0-
Pin10	TMDS Saat+
Pin11	TMDS Saat Toprak
Pin12	TMDS Saat-
Pin13	CEC(Consumer Electronics Control)
Pin14	Yedek Pin
Pin15	EDID için Seri Saat
Pin16	EDID için Seri Veri Yolu
Pin18	EDID Toprak
Pin19	Sökme/Takma Algılama

TMDS alıcı, verici modülleri ve EDID veri aktarımları hakkında bilgiler bir sonraki alt başlıklarda verilecektir. Çizelge 3.1'de HDMI soketi üzerinde bulunan pinlerin açıklamaları verilmiştir.

3.2 Alıcı Modülü

Bu modül TMDS kodlanmış halde gelen kırmızı, mavi, yeşil ve referans saat çevrim sinyallerini alarak 24 bitlik RGB, yatay, dikey senkron sinyalleri haline getirmektedir. Alınan RGB renk sinyalleri TMDS kod çözücü modülüne aktarılmaktadır[21][22].

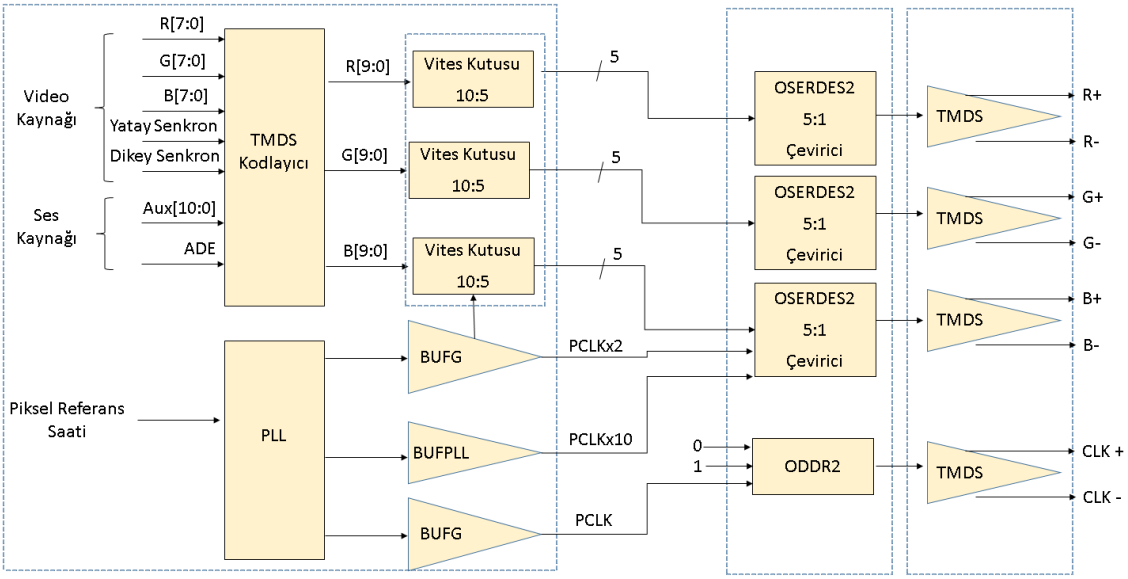


Şekil 3.3 TMDS Alıcı Modülü

TMDS kod çözücü modül 10 bitlik sinyaller almaktadır. Seri halde gelen sinyalleri 10'ar bitlik hale getirilmesi yani paralel bir yapıya dönüştürülmesine ihtiyaç duyulmaktadır. Bunun için iki ayrı modül kullanılmıştır. ISERDES2 modülü aldığı sinyalleri 5 bitlik hale getirmektedir. 5 bitlik sinyaller ise 10 bitlik hale Vites kutusu modülü sayesinde dönüşmektedir. Vites kutusu modülü ve ISERDS2 modülü TMDS kod çözücü modülüne göre daha yüksek frekansta çalışmak zorundadır. Çünkü TMDS kod çözücü her saat çevriminde 30 bitlik veri alırken vites çözücü modüllerine toplamda 15 bitlik veri girmektedir. Dolayısıyla Vites çözücü modülünün 2 kat hızlı çalışması gerekmektedir. Aynı ihtiyaçtan ötürü ISERDES2 modülü de saat çevrim sinyali hızlandırılmıştır. Bu işlemler PLL modülü tarafından yapılmaktadır.

3.3 Verici Modülü

Bu modül kaynak video sinyallerini TMDS sinyalleri haline dönüştürerek, hedef HDMI cihaza aktarmaktadır. Kırmızı, mavi ve yeşil renk bilgilerini taşıyan 24 bitlik sinyal, aktarılan karenin yatay ve dikey senkron sinyalleri ve referans saati sinyalleri modüle verilmektedir[18][19].

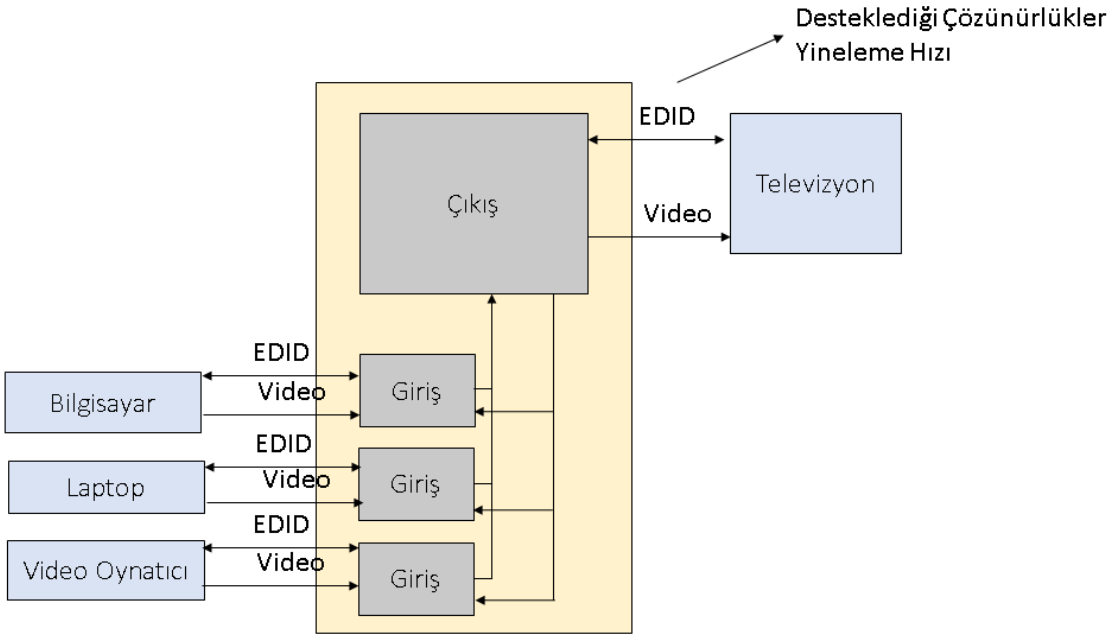


Şekil 3.4 Görüntü İletimi ve Yatay/Dikey Senkron Sinyalleri

Alınan video kaynağı sinyalleri TMDS Kodlayıcı modülüne verilerek R,G,B 10'ar bitlik kodlanmış sinyalleri vermektedir. HDMI portunda veriler seri aktarım yapılmaktadır. Dolayısıyla 10 bitlik alınmış sinyallerin birer bite dönüştürülmesi yani 10:1 serileştirilmesi gerekmektedir. Bu çevirim iki aşamada yapılmıştır. 10:5 dönüşümü vites kutusu tarafından, dönüşen 5 bit ise OSERDES2 tarafından 1 bit e serileştirilme işlemini gerçekleştirmiştir. Her saat çevriminde pikseller geldiği için bu çevrimler hızlı olmak zorundadır. Bunun için PLL modülü kullanarak mevcut saat hızından daha hızlı saat çevrimleri oluşturulup, çevrim modüllerine verilmiştir. Son olarak HDMI portu üzerinde saat çevrimine ihtiyaç duyulmaktadır. Bu sinyal piksel referans saatinin tamponlanmış hali olarak aktarımı yapılmaktadır.

3.4 EDID Transferi

EDID, Genişletilmiş Görüntü Kimlik Verisi video cihazları arasında aktarımı yapılan bir veri değişim standardıdır. Videonun aktarılacağı cihazın desteklediği çözünürlük, yineleme hızı gibi parametrelerin video kaynakları ile paylaşılmasını sağlar.

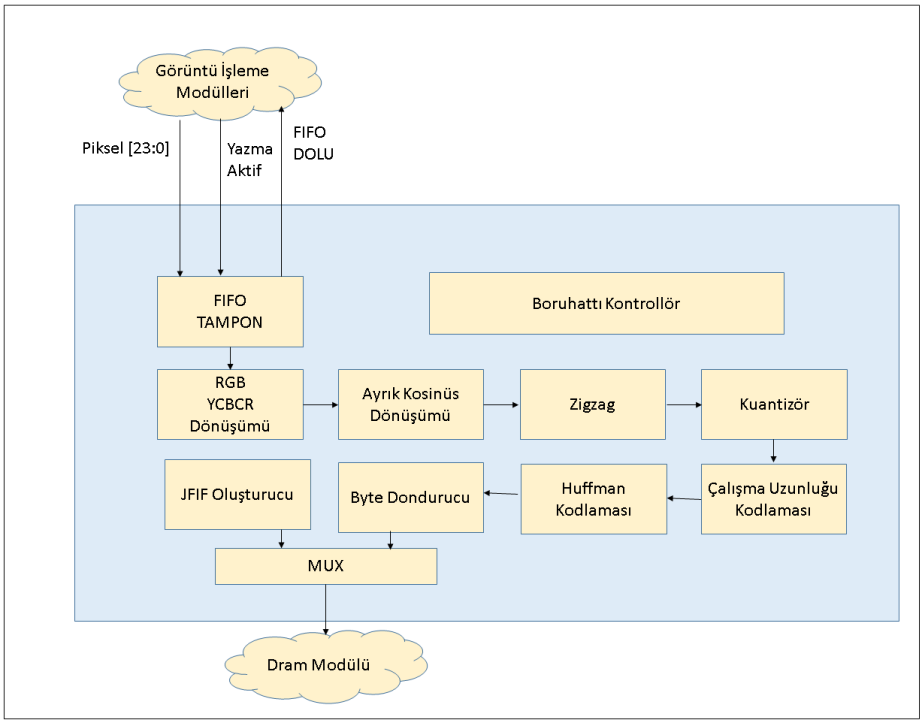


Şekil 3.5 EDID Aktarım Mimarisi[22]

EDID ayrıca üretici firmanın ismini, seri numarasını, ürün türünü, filtre türünü, parlaklık, yatay ve dikey çözünürlük gibi bilgileri de aktarmaktadır. Bu aktarım sadece bir defa video aktarımı başlamadan önce yapılır. Kaynak ve hedef arasındaki kablo çıkartılıp tekrar takıldığında HPD sayesinde tekrar aktarımı sağlanır. Eğer bu aktarım sağlıklı bir biçimde gerçekleştirilemezse video aktarımı başlamayacaktır. Bazı eski cihazlarda EDID veri değişimi gerçekleştiremeyebilirler. Bu durumda EDID emülatör cihazları hedef kaynak arasında gerekli değişimleri gerçekleştirir.

3.5 Görüntü Sıkıştırma

Veri aktarımının bant genişliği sebebi ile limitlendiği durumlarda hedef cihaza saniye başına aktarılacak paketlerin toplam boyutlarının azaltılması gerekmektedir. Görüntü sıkıştırma, aktarım esnasında kullanılacak bant genişliğini azaltmaktadır. Dolayısıyla sisteme eklenecek daha karmaşık görüntü işleme algoritmaları, daha yüksek çözünürlük ve saniye başına kare sayısının arttırılabilmesi için önemlidir. Burada, Opencores.org sitesine Michal Krepa tarafından geliştirilmiş olan JPEG görüntü sıkıştırma modülü kullanılmıştır[26].



Şekil 3.6 JPEG Dönüşüm Aşamaları

Şekil 3.6'da JPEG çevirici modülün içyapısındaki modüller ve sistemin diğer modülleri ile bağlantısı verilmiştir. JPEG çevrim modülü her saat çevriminde piksel alabilmekte ve boru hattı mekanizmasına uygun olarak çalışmaktadır. Modülün çıkışında JPEG formatına dönüştürülmüş byte dizileri çıkmaktadır. Genel olarak uygulanan adımlar aşağıda sıralanmıştır.

1- Piksellerin ara işlemlerinin tutulacağı tampon ihtiyacı bulunmaktadır. İlk aşamada görüntü işleme modülünden alınan pikselleri FIFO tamponda tutulmaktadır.

2- RGB renk uzayından YCBCR renk uzayına dönüşümü yapılır. Bu renk uzayı resim video gösterimlerinde sıkça kullanılmaktadır. Y parlaklığı, CB ve CR değerleri ise renk bilgilerini tutmaktadır.

3- Ayrık kosinüs dönüşümü, resimde kosinüs katsayılarının hesaplanması için kullanılmaktadır. Bu işlem resimde yüksek ve düşük frekanslı noktaları vermektedir. Tespit edilen yüksek frekanslı noktaların çoğu atılarak daha az bilgi ile ifade edilebilecek bir veri oluşur. Bu işlem sonucunda yüksek frekanslı kenar gibi öğeler belirginliğini yitirip bulanık bir hal alacaktır. Ayrıca görüntü işleme bölümünde kenar bulucu filtreler ile resimdeki yüksek frekanslı elemanlar anlatılmaktadır.

4- Zigzag işlemi piksellerin sol üstünden başlayıp, zig zaglar çizerek seri halde veri elde edilme yöntemidir. Buradaki amaç art arda gelen verilerin birbirlerine olabildiğince benzer getirilmeye çalışılmasıdır. Bu da sıkıştırma başarısının artması anlamına gelmektedir.

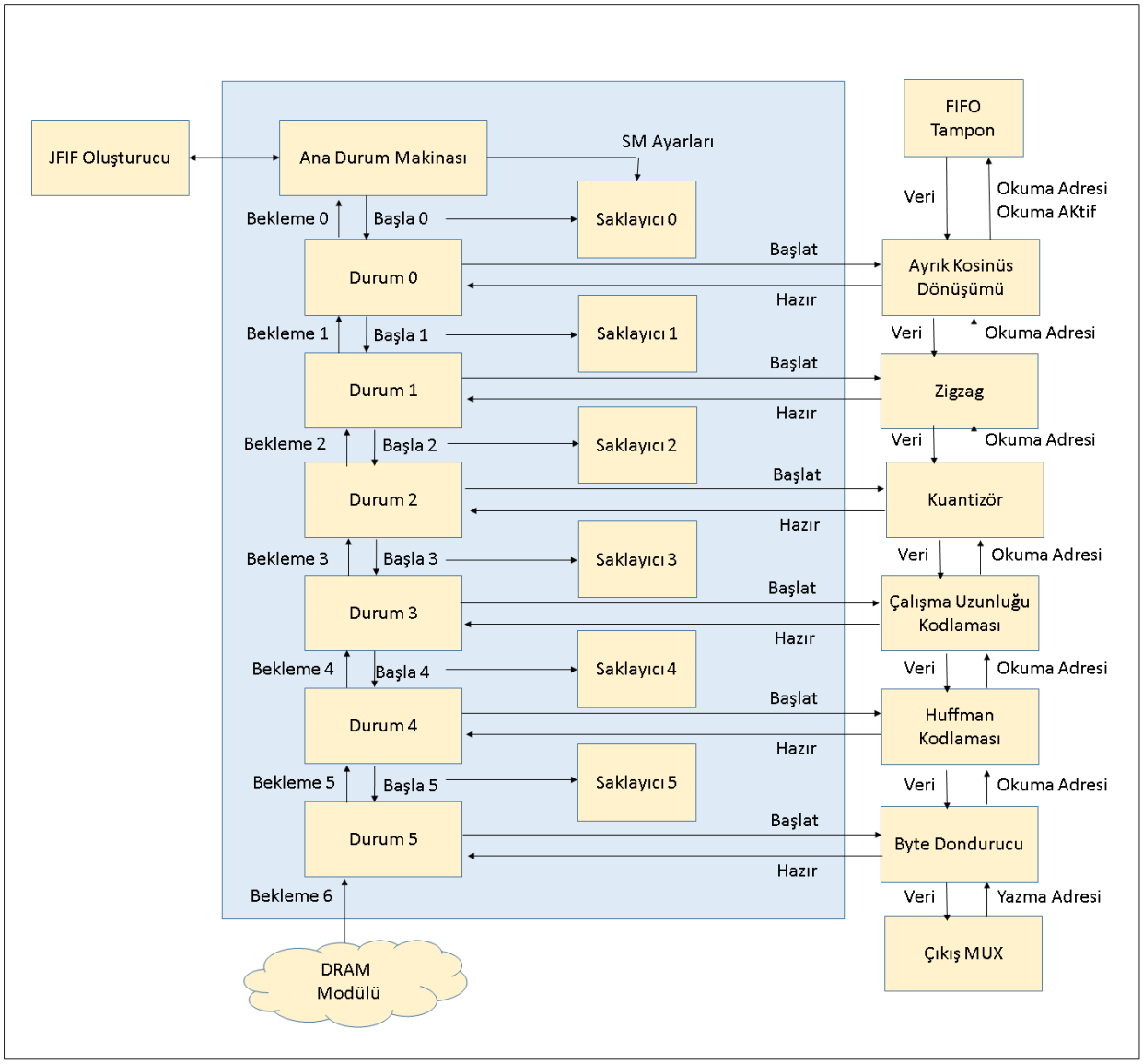
5- Kuantizör modülü ard arda gelen piksellerin aralarındaki farkın az olduğu durumlarda bu piksellere aynı değerin verilmesini sağlar. Dolayısıyla benzer pikselleri gruplamaktadır. İfade edilmesi gereken piksel değerini azaltarak sıkıştırmaya katkı sağlamaktadır.

6- Bu aşamada birbirini takip eden piksel dizisinde tekrarların gösterimlerinin azaltılmasında kullanılmaktadır. Örneğin art arda gelen 1000 adet 200 değeri varsa, bunu {1000,200} şeklinde ifade eder.

7- Huffman kodlaması, çalışılan piksel gruplarında çok tekrar eden piksellerin gösterim değerinin daha az bitle, az tekrar edilenlerin ise daha çok bit ile ifade etmesini sağlamaktadır.

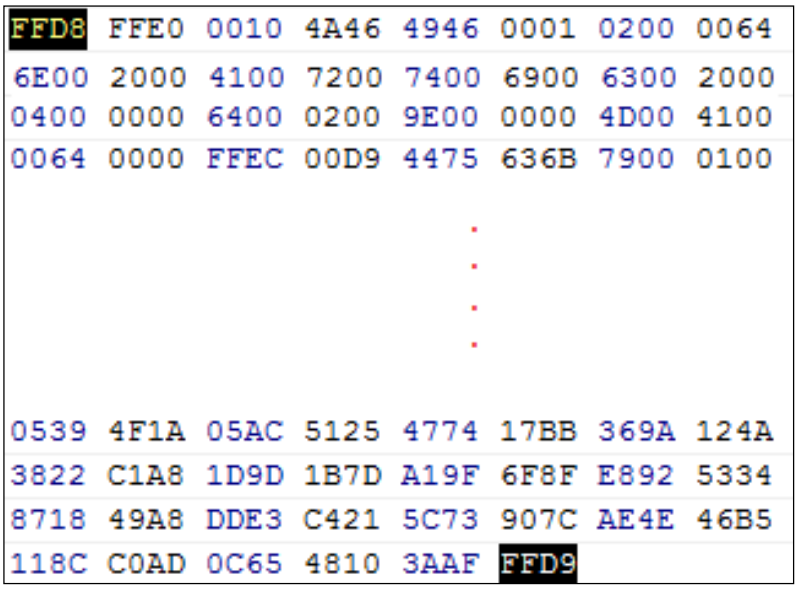
8- Son aşamada JPIF modülü, JPEG dosyası için gerekli başlık ve bitiş bytelarını vermektedir. Byte dönüştürücü modül ise aldığı sıkıştırılmış diziyi JPEG byteları haline getirmektedir.

İçyapıda adımların uygulanma akışı, sonlu durum makinası tarafından yönetilmektedir. Şekil 3.7'te verilmiş olan yapı durumların sıralamasını, geçişlerine çıkış modülü bağlantısını göstermektedir.



Şekil 3.7 JPEG Durum Makinası Diyagramı

JPEG görüntüleri başlangıç başlığı FFD8, bitiş ise FFD9 byte'ları olmaktadır. Bu özellik kullanılarak gelen veri akışının içerisinden FFD8 ve FFD9 byteları arasında kalan veriler alınarak ekrana gösterilmektedir.



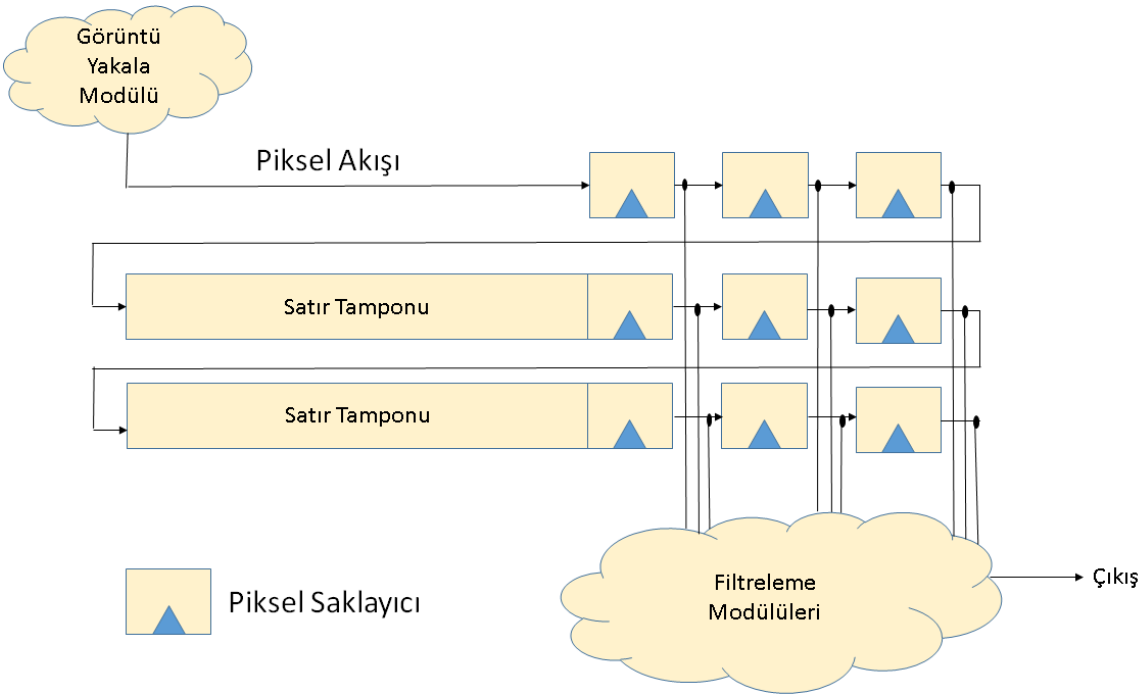
Şekil 3.8 Örnek JPEG Byteları

FPGA tarafından gönderilmekte olan görüntüler ard arda olduğu için her FFD9 byte ile biten resimden sonra FFD8 ile yeni bir görüntü gelmektedir.

BÖLÜM 4

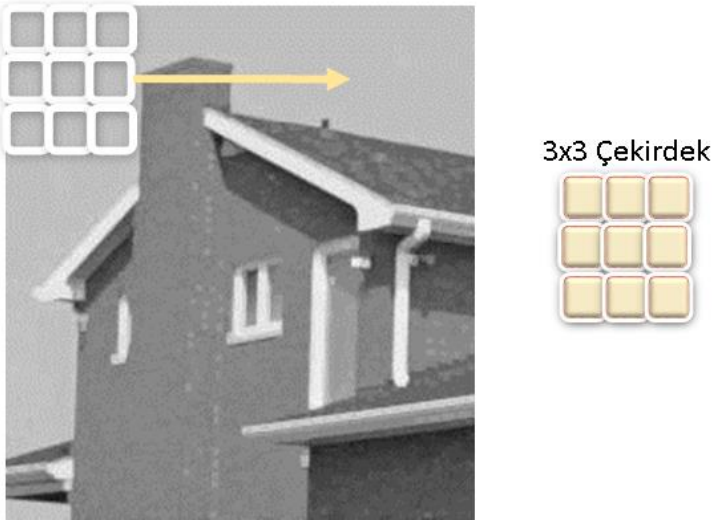
GÖRÜNTÜ İŞLEME

Yüksek çözünürlüklü görüntünün yakalanmasının ardından, kullanıcı tarafından seçilen görüntü işleme çekirdeği koşmaktadır. Yaygın olarak kullanılan filtrelerin tasarımı gerçekleştirilmiştir. Bunlar ortanca, ortalama, gauss, keskinleştirici, kenar bulucu, kabartıcı, gradyan, sobel, aşındırıcı ve genişletici filtre olmak üzere 10 adet filtrenin FPGA üzerinde gerçeklemesi yapılmıştır. Filtrenin uygulanacağı piksel gruplarının seçimi Şekil 4.1'de gösterilmiştir.



Şekil 4.1 Kayan Pencereler ile 3x3 Filtreleme

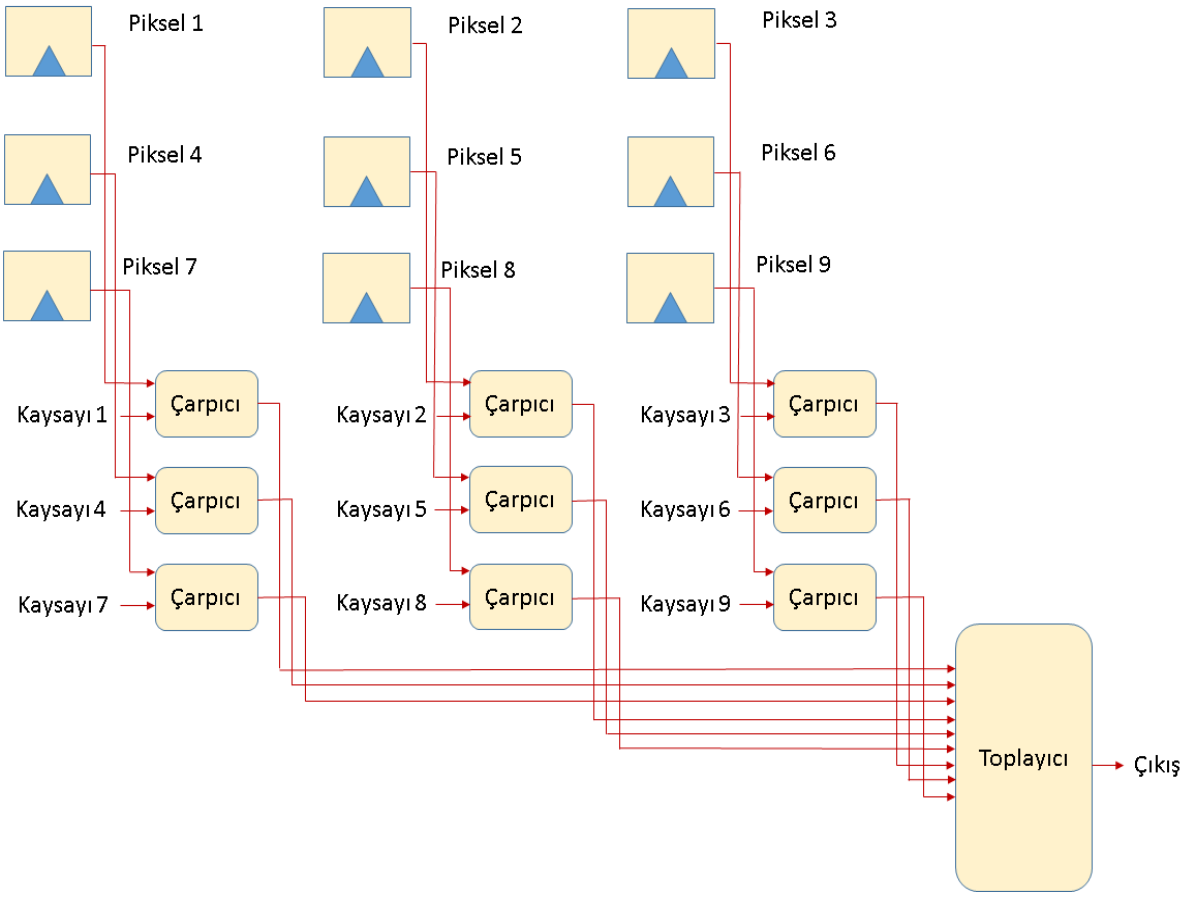
Bölüm 3'te anlatılan görüntü yakalama modülü, her bir pikseli bir saat çevriminde vermektedir. Sistem 3x3 filtre yapması için tasarlandığından, gelen pikseller 2 adet satır tamponunda tutulmuş, 3. satır için sadece 3 adet saklayıcıya ihtiyaç duyulmuştur. Dolayısı ile pikseller filtreden geçip sonuçları alınırken, hesaplama işi tamamlanmış olan piksellerin yerlerine görüntü yakalama modülünden yeni pikseller aktarılmaktadır. Görüntü üzerinde filtrenin hareketi Şekil 4.2'te verilmiştir[27].



Şekil 4.2 Filtrenin Görüntü Üzerindeki Hareketi

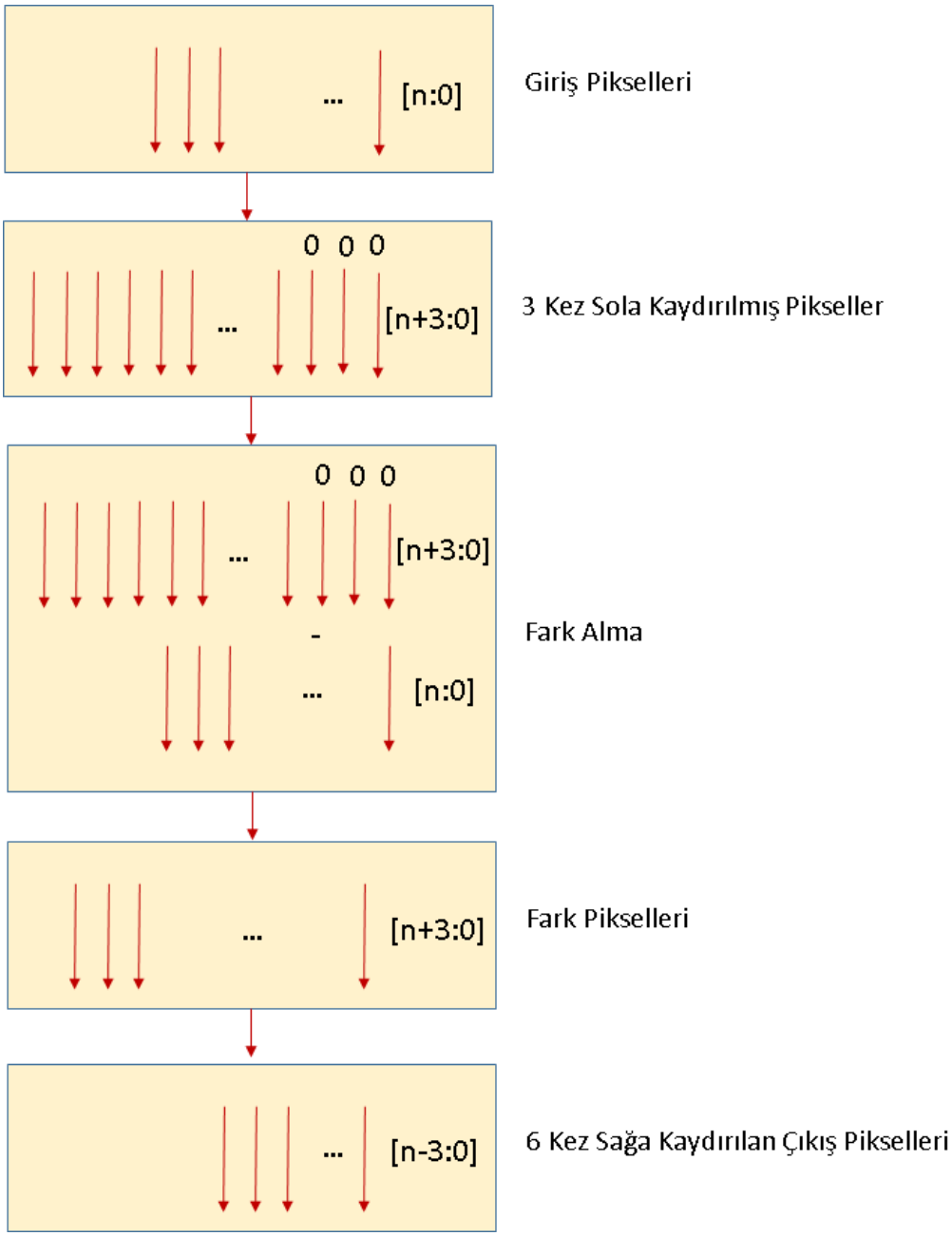
Bazı filtrelemelerde piksellerin sabit katsayılar ile çarpılıp, toplanması gerekmektedir. 3x3 filtre yapılırken her bir pikselin çarpıcısı, toplamda 9 adet çarpıcı bulunmaktadır. Pikseller ile katsayılar paralel olarak çarpım sonuçları toplama modülüne gelmektedir.

Modülün mimarisi Şekil 4.3'te verilmiştir.



Şekil 4.3 Filtre Katsayı Piksel Hesaplama Mimarisi

Filtrelemelerde bölme işlemine ihtiyacı duyulmaktadır. FPGA üzerinde bölüm işlemi yapmak alan ve gecikme açısından maliyetli bir işlemdir. Bölüm işlemi Xilinx firmasının bölme modülü kullanıldığında 12 saat çevrimi sürmektedir. Sistemin gerçek zamanlı çalışması için bölme modülünün boru hattı (pipeline) yapısına uygun olarak tasarlanması gerekmektedir. Bu yöntem hem FPGA üzerinde kaynak kullanımının çok olmasına hem de gecikmenin artmasına neden olacaktır. Bu sebeple 9'a bölmek için Xilinx firmasının modülünü kullanmak yerine (2^3*(Toplam Piksel) - 1)/64 ≈9,1428 işlemi, 9'a yakın bir sayıya bölüm gerçekleştirilmiştir[28].



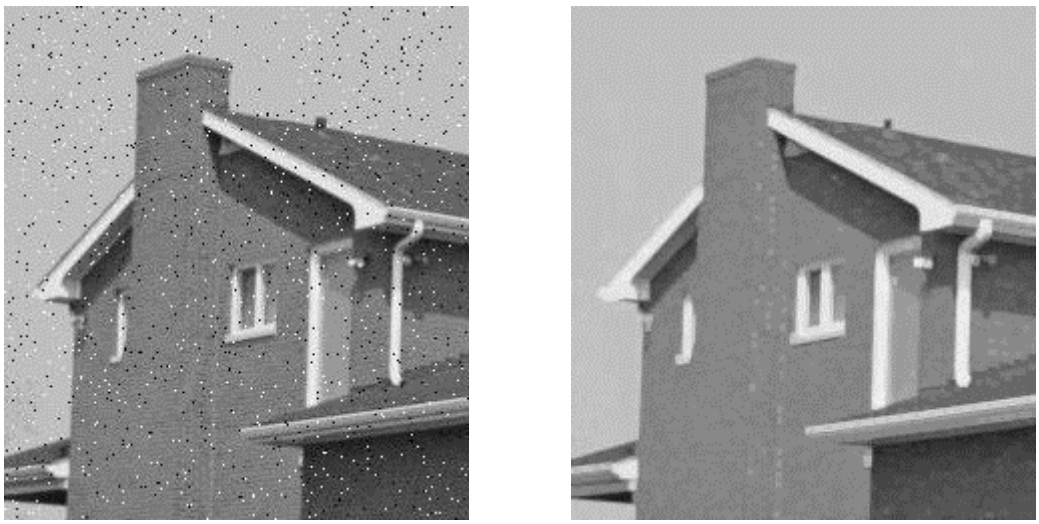
Şekil 4.4 Düşük Maliyetli Bölme İşlemi

Bölme işlemi mimarisi Şekil 4.4 de verilmiştir. Bölüm maliyeti, kaydırma işleminin maliyeti olmadığından sadece bir çıkartma devresine düşürülmüştür.

İlerleyen alt bölümlerde tasarlanmış olan filtreleme modülleri hakkında bilgi verilecektir.

4.1 Ortanca Filtre

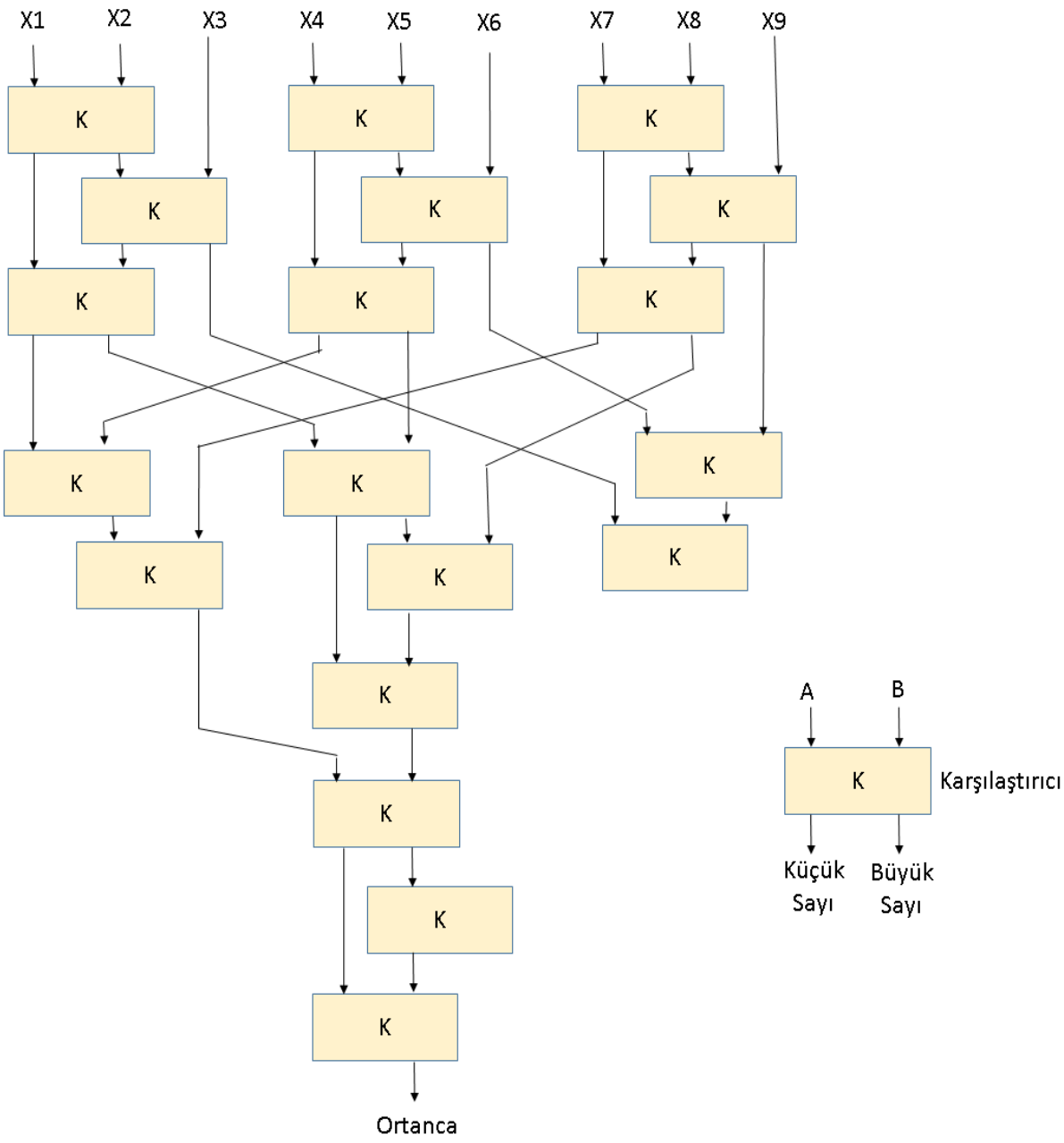
İstatistik ve görüntü işleme konularına sıkça rastlanılan bir filtre türüdür. Tuz-biber gürültüsünü kaldırmada kullanılmaktadır. Bu gürültü türünde, görüntüde ani değişim gösteren bozulmuş pikseller bulunmaktadır. Lineer olmayan bir filtredir, çünkü çevre pikselleri alınarak sıralama işlemi yapılmaktadır. Algoritmada ilk olarak pencere genişliği seçilmektedir. Pencere genişliği, filtrenin her bir piksel için işlem yapacağı komşu piksellerin sayısını belirlemektedir. Pencere genişliğinin belirlenmesi resimdeki gürültü miktarı ile orantılı olmalıdır. Çok gürültülü bir resimde pencere genişliği büyük tutulmalıdır. Pencere seçiminden sonra, seçilen genişlikteki komşu pikseller değerlerine göre sıralanır. Sıralama işlemi için birçok farklı algoritma bulunmaktadır. Sıralanmış olan piksellerin, ortanca olanı seçilerek mevcut piksel ile değiştirilir.



(a) Giriş Tuz Biber Gürültülü Resim (b) Filtre Çıktısı

Şekil 4.5 Tuz-Biber Gürültülü Görüntü ve Medyan Filtre Çıktısı

Çip üzerinde paralel çalışan, medyan filtre mimarisi uygulanmıştır[29]. Pencere genişliği 3x3 olacak şekilde, 9 pikselin ortancası hesaplanmıştır. Karşılaştırma işlemi için 19 adet karşılaştırıcı kullanılmıştır. Şekil 4.5'te verilen giriş resmine göre filtre çıktısı verilmiştir.



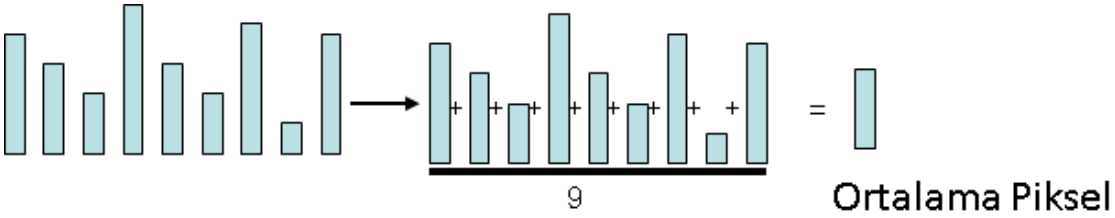
Şekil 4.6 Optimize Edilmiş 19 Karşılaştırıcılı, 3x3 Medyan Mimarisi[30]

Şekil 4.6'da verilen mimaride karşılaştırıcılar kutu içerisinde K ile gösterilmiştir. İki girişi ve iki çıkışı bulunmaktadır. Girişlerinden büyük olanını sol taraftan, küçük olanını ise sağ taraftan vermektedir. Yaklaşık olarak 12.5ns gecikme ile işlemler yapılabilmektedir. Kullanılan saat frekansı 100mhz olduğu için 2 saat çevriminde işlemler gerçekleştirilmiştir. Karşılaştırma işlemleri FPGA üzerindeki hazır karşılaştırıcı modüller kullanılarak gerçekleştirilmiştir.

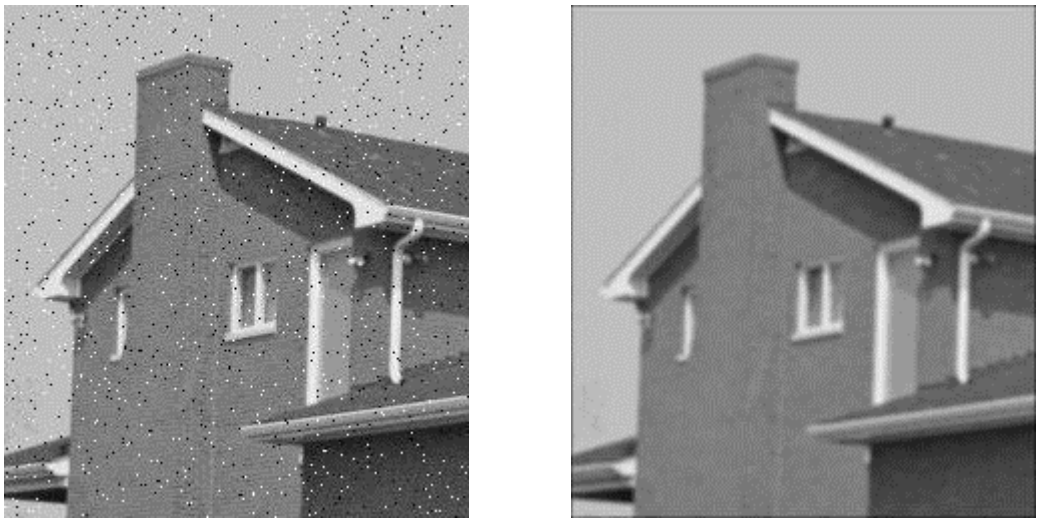
4.2 Ortalama Filtre

Resimdeki keskin geçişleri azaltarak, yumuşatmak için kullanılır. Lineer bir algoritmadır. Seçilen pencere boyutuna göre komşu piksellerin toplamını, seçilen piksel sayısına bölerek ortalama piksel elde edilir. Şekil 4.7'de 3x3 filtrede uygulaması gösterilmiştir.

Bölüm işlemi maliyetinden kurtulmak için Görüntü İşleme giriş bölümünde anlatılan bölme mimarisi kullanılmıştır. Şekil 4.8'de verilen giriş resmine göre filtre çıktısı verilmiştir.



Şekil 4.7 3x3 Ortalama Piksel Hesaplanması



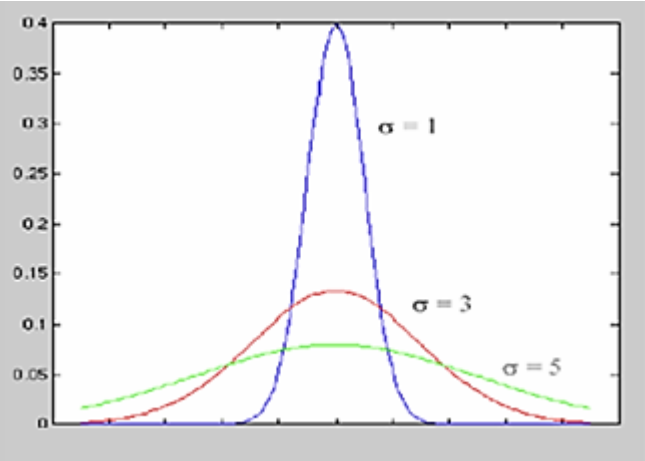
(a) Giriş Tuz Biber Gürültülü Resim (b) Filtre Çıktısı

Şekil 4.8 Tuz-Biber Gürültülü Görüntü ve Ortalama Filtre Çıktısı

4.3 Gauss Filtre

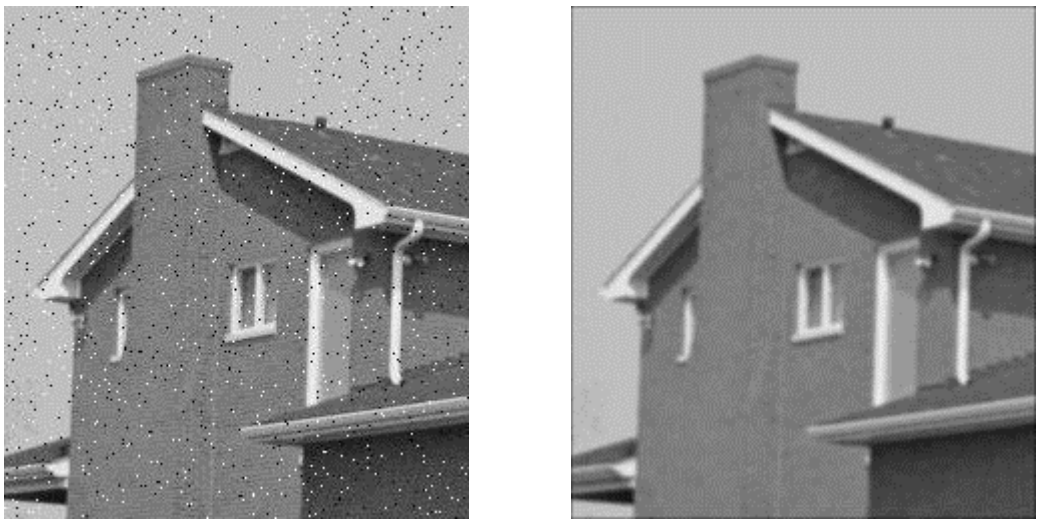
Gauss filtresi, resmi bulanıklaştırmaya, gürültü ve detayları azaltmak için kullanılan bir filtredir. Görüntü işleme algoritmalarında ön işleme aşamaları esnasında sıkça kullanılmaktadır. Alçak sinyalleri geçiren bir filtreleme yöntemidir. Dolayısı ile kenar gibi yüksek frekanslı bileşenler görüntüden kaybolur. Ortalama filtresinin yaptığı gibi görüntüdeki geçişleri, kenarları yumuşatmaktadır. Ortalama filtresine göre daha yumuşak bir sonuç çıkartmaktadır. Ortalama filtresinde oluşan Bokeh etkisini azaltmaktadır. Eşitlik 4.1'de verilmiş olan formül ile hesaplanmaktadır. Şekil 4.9'da ise farklı Sigma Gauss dağılımları verilmiştir.

$$g(x,y) = \frac{1}{(2\pi\sigma^2)} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{4.1}$$



Şekil 4.9 İki Boyutlu Gauss Sigma Dağılımları

Verilen formülde x yatay düzlemdeki orijinden olan uzaklığı, y dikey düzlemdeki orijinden olan uzaklığı, sigma değeri ise gauss dağılımının standart sapmasını belirtmektedir. Sigma parametresi resmin ne oranda yumuşatılacağını belirlemektedir. Şekilde farklı sigmaların dağılma olan etkisi verilmiştir[31].



(a) Giriş Tuz Biber Gürültülü Resim (b) Filtre Çıktısı

Şekil 4.10 Tuz-Biber Gürültülü Görüntü ve Gauss Filtre Çıktısı

Çip üzerinde 3x3 pencere genişliğinde filtre uygulanmıştır. Şekil 4.10'da verilen giriş resmine göre filtre çıktısı verilmiştir. Çekirdek değerleri olarak Çizelge 4.1'deki değerler kullanılmıştır.

Çizelge 4.1 Gauss Çekirdeği Değerleri

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Çekirdekte bulunan bölme işlemleri 2 sayısının katlarıdır. Dolayısı ile bölme işlemi yapmak yerine kaydırma işlemi yapmak yeterli olmaktadır. 2'nin gerekli kuvveti kadar sağa kaydırılarak bölüm gerçekleştirilmiştir.

4.4 Keskinleştirici Filtre

Keskinleştirici filtre, resmin üzerine yüksek geçirgen filtre uygulanıp, orijinal resmin üzerine eklenmesi ile elde edilir. Yüksek geçirgen filtre sonucundan kenarlar bulunmaktadır. Kenarlar resmin üzerine eklenmesi görüntüde keskinliğe sebep olmaktadır[32].

(a) Filtresiz Resim (b) Filtre Çıktısı

Şekil 4.11 Filtresiz Resim ve Keskinleştirici Filtre Çıktısı

Çip üzerinde 3x3 pencere genişliğinde filtre uygulanmıştır. Şekil 4.11'da verilen giriş resmine göre filtre çıktısı verilmiştir. Çekirdek değerleri olarak Çizelge 4.2'deki değerler kullanılmıştır.

Çizelge 4.2 Keskinleştirici Çekirdeği Değerleri

0	-1	0
-1	5	-1
0	-1	0

Çekirdekte bulunan bölme işlemleri 2 sayısının katlarıdır. Dolayısı ile bölme işlemi yapmak yerine kaydırma işlemi yapmak yeterli olmaktadır. 2'nin gerekli kuvveti kadar sağa kaydırılarak bölüm gerçekleştirilmiştir.

4.5 Kenar Bulucu Filtre

Kenar bulucu filtre, ışığın ya da rengin değişimlerinin olduğu yerleri ortaya çıkartmaktadır. Yüksek frekanslı elemanları geçiren bir filtre olduğu için kenar gibi yüksek frekanslı elemanlar resim üzerinde kalmaktadır.

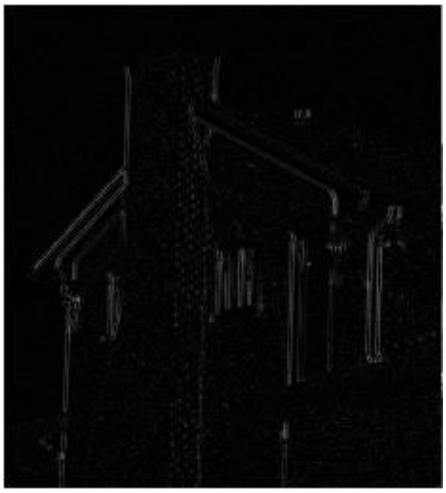
(a) Filtresiz Resim (b) Filtre Çıktısı

Şekil 4.12 Yatay Kenar Bulma Filtre Çıktısı

Çip üzerinde 3x3 pencere genişliğinde filtre uygulanmıştır. Şekil 4.12'da verilen giriş resmine göre filtre çıktısı verilmiştir. Yatay kenar bulma çekirdek değerleri olarak Çizelge 4.3'teki değerler kullanılmıştır.

Çizelge 4.3 Yatay Kenar Bulucu Çekirdeği Değerleri

0	-1	0
0	2	0
0	-1	0



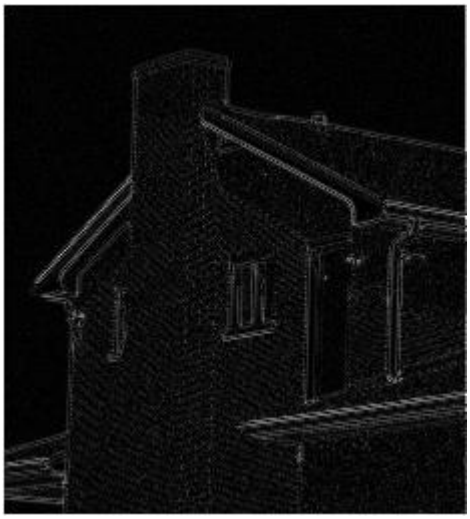
(a) Filtresiz Resim (b) Filtre Çıktısı

Şekil 4.13 Dikey Kenar Bulma Filtre Çıktısı

Şekil 4.13'da verilen giriş resmine göre filtre çıktısı verilmiştir. Dikey filtre için çekirdek değerleri olarak Çizelge 4.4'teki değerler kullanılmıştır.

Çizelge 4.4 Dikey Kenar Bulucu Çekirdeği Değerleri

0	0	0
-1	2	-1
0	0	0



(a) Filtresiz Resim (b) Filtre Çıktısı

Şekil 4.14 Yatay ve Dikey Kenar Bulma Filtre Çıktısı

Şekil 4.14’de verilen giriş resmine göre filtre çıktısı verilmiştir. Yatay ve Dikey filtre için çekirdek değerleri olarak Çizelge 4.5’teki değerler kullanılmıştır.

Çizelge 4.5 Yatay-Dikey Kenar Bulucu Çekirdeği Değerleri

0	1	0
1	-4	1
0	1	0

Bölüm işlemi maliyetinden kurtulmak için Görüntü İşleme giriş bölümünde anlatılan bölme mimarisi kullanılmıştır.

4.6 Kabartıcı Filtre

Kabartıcı filtre, resimdeki kenarları kabartmak ya da gölgelemek için kullanılmaktadır. Bu filtreleme piksel gruplarındaki değişimin yönünü vermektedir[33].

(a) Filtresiz Resim (b) Filtre Çıktısı

Şekil 4.15 Kabartıcı Filtre Çıktısı

Çip üzerinde 3x3 pencere genişliğinde filtre uygulanmıştır. Şekil 4.15'de verilen giriş resmine göre filtre çıktısı verilmiştir. Çekirdek değerleri olarak Çizelge 4.6'daki değerler kullanılmıştır.

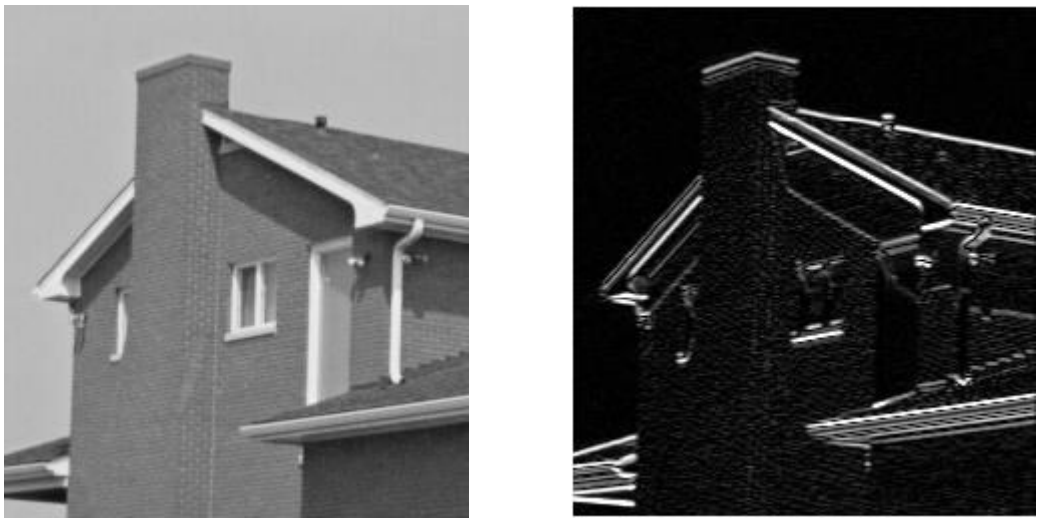
Çizelge 4.6 Kabartıcı Filtre Çekirdeği Değerleri

-2	-1	0
-1	1	1
0	1	2

Çekirdekte bulunan bölme işlemleri 2 sayısının katlarıdır. Dolayısı ile bölme işlemi yapmak yerine kaydırma işlemi yapmak yeterli olmaktadır. 2'nin gerekli kuvveti kadar sağa kaydırılarak bölüm gerçekleştirilmiştir.

4.7 Gradyan Filtre

Gradyan filtre, verilen doğrultuda resmin üzerinde renk/ışık değişimlerini tespit ederek çıktı verir. En sık kullanım alanı kenar bulma algoritmalarındadır. Güçlü özellik çıkarımı ve doku eşleme işlemlerinde önişlem olarak kullanılabilirler. Yatay, dikey olmak üzere iki çekirdeği bulunmaktadır.



(a) Filtresiz Resim (b) Filtre Çıktısı

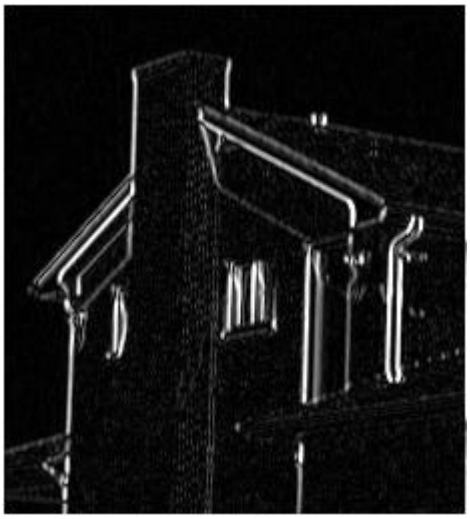
Şekil 4.16 Yatay Gradyan Filtre Çıktısı

Çip üzerinde 3x3 pencere genişliğinde filtre uygulanmıştır. Şekil 4.16'da verilen giriş resmine göre filtre çıktısı verilmiştir. Yatay filtre için çekirdek değerleri olarak Çizelge 4.7'deki değerler kullanılmıştır.

Çizelge 4.7 Yatay Gradyan Filtre Çekirdeği Değerleri

-1	0	1
-1	0	1
-1	0	1

(a) Filtresiz Resim



(b) Filtre Çıktısı

Şekil 4.17 Dikey Gradyan Filtre Çıktısı

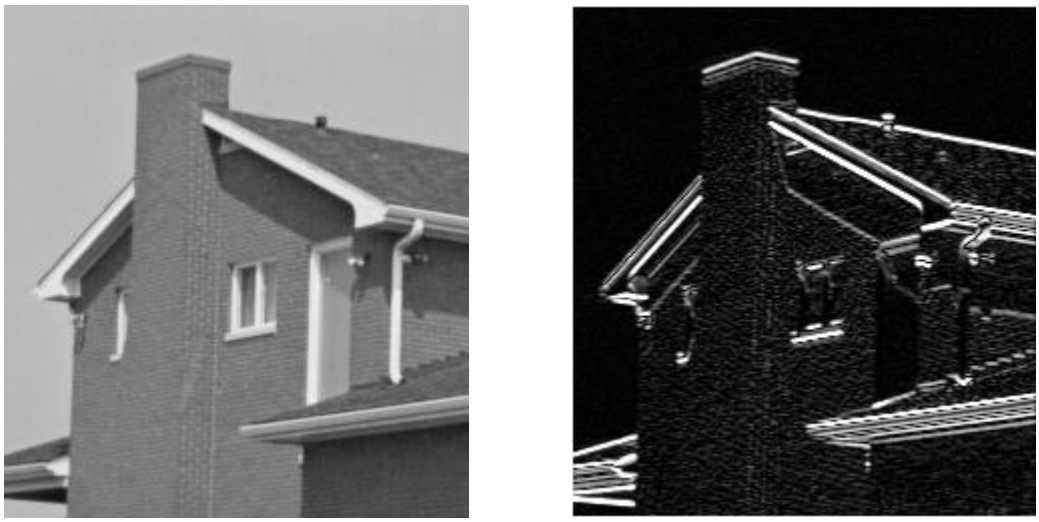
Şekil 4.17'da verilen giriş resmine göre filtre çıktısı verilmiştir. Şekil 4.17'de verilen giriş resmine göre filtre çıktısı verilmiştir. Dikey filtre için çekirdek değerleri olarak Çizelge 4.8'deki değerler kullanılmıştır.

Çizelge 4.8 Dikey Gradyan Filtre Çekirdeği Değerleri

-1	-1	-1
0	0	0
1	1	1

4.8 Sobel Filtre

Kenar bulmak için kullanılan yöntemlerden birisidir. Kenarları ve geçişleri belirgin hale getirmektedir. Görüntüdeki yoğunluk değişiminin olduğu yerleri bularak kenarı tespit etmektedir. Bu yöntemde iki adet çekirdek bulunmaktadır. Bunlar yatayda bulunan kenarları ve düşeyde olan kenarları ortaya çıkartmaktadır[34].



(a) Filtresiz Resim (b) Filtre Çıktısı

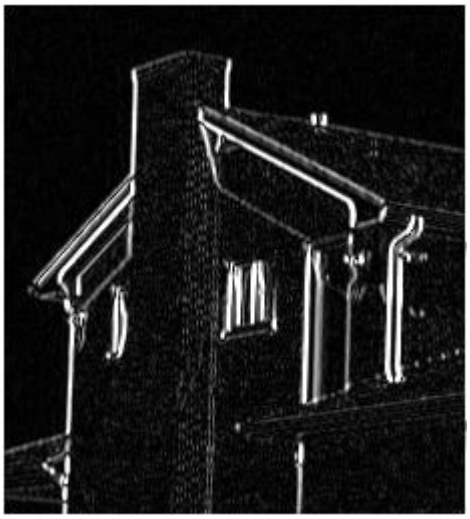
Şekil 4.18 Yatay Sobel Filtre Çıktısı

Çip üzerinde 3x3 pencere genişliğinde filtre uygulanmıştır. Şekil 4.18'de verilen giriş resmine göre filtre çıktısı verilmiştir. Yatay filtre için çekirdek değerleri olarak Çizelge 4.9'daki değerler kullanılmıştır.

Çizelge 4.9 Yatay Sobel Filtre Çekirdeği Değerleri

1	0	-1
2	0	-2
1	0	-1

(a) Filtresiz Resim



(b) Filtre Çıktısı

Şekil 4.19 Dikey Sobel Filtre Çıktısı

Şekil 4.19'da verilen giriş resmine göre filtre çıktısı verilmiştir. Dikey filtre için çekirdek değerleri olarak Çizelge 4.10'daki değerler kullanılmıştır.

Çizelge 4.10 Dikey Sobel Filtre Çekirdeği Değerleri

1	2	1
0	0	0
-1	-2	-1

Bölüm işlemi maliyetinden kurtulmak için Görüntü İşleme giriş bölümünde anlatılan bölme mimarisi kullanılmıştır.

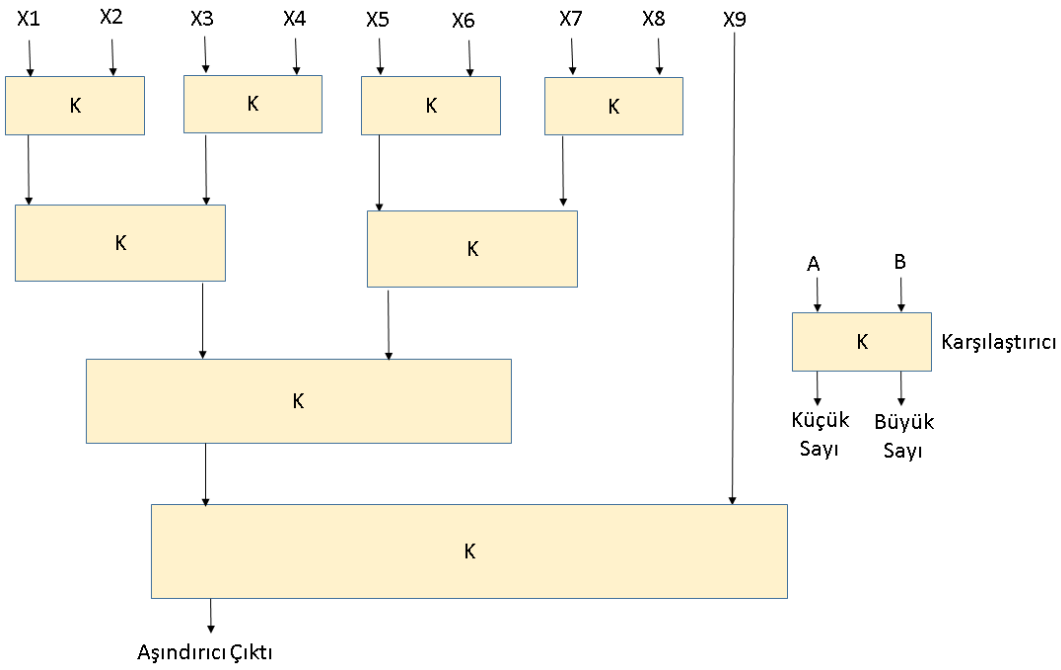
4.9 Aşındırıcı Filtre

Aşındırıcı filtre, seçilen piksel grubunu aşındırmaktadır. 3x3 filtreleme yapılırken, her piksel için komşusu olan 8 pikselden en küçük olanının değeri verilmektedir[35].

(a) Filtresiz Resim (b) Filtre Çıktısı

Şekil 4.20 Aşındırıcı Filtre Çıktısı

Şekil 4.20'de verilen giriş resmine göre filtre çıktısı verilmiştir. Filtreleme için 8 adet karşılaştırıcı kullanılmıştır. Filtre mimarisi Şekil 4.21'de verilmektedir.



Şekil 4.21 Aşındırıcı Filtre Mimarisi

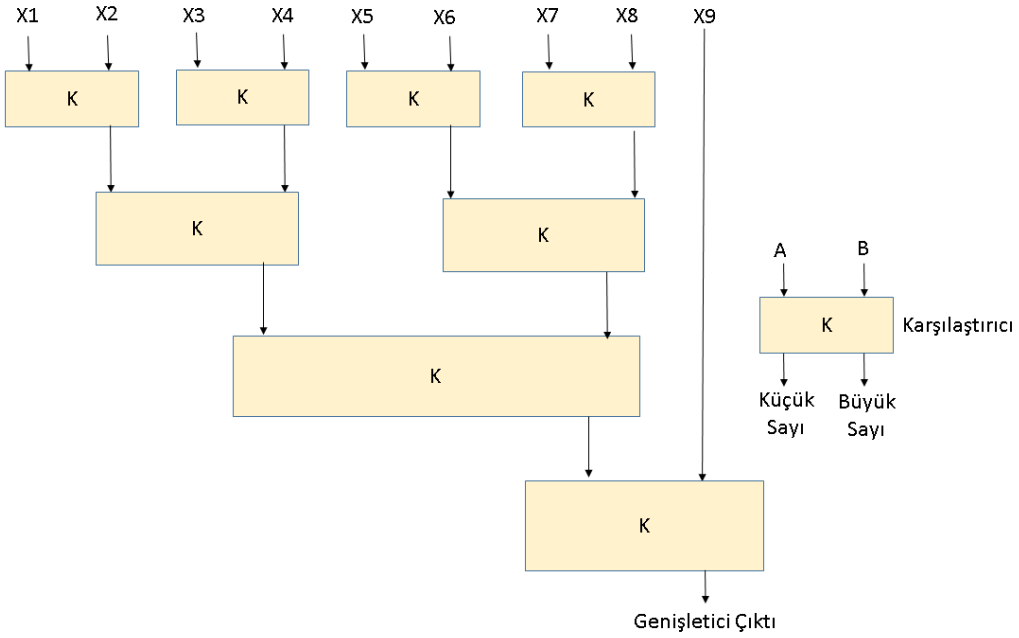
4.10 Genişletici Filtre

Aşındırıcı filtre, seçilen piksel grubunu genişletmektedir. 3x3 filtreleme yapılırken, her piksel için komşusu olan 8 pikselden en büyük olanının değeri verilmektedir[32].

(a) Filtresiz Resim (b) Filtre Çıktısı

Şekil 4.22 Genişletici Filtre Çıktısı

Şekil 4.22'de verilen giriş resmine göre filtre çıktısı verilmiştir. Filtreleme için 8 adet karşılaştırıcı kullanılmıştır. Filtre mimarisi Şekil 4.23'te verilmektedir.



Şekil 4.23 Genişletici Filtre Mimarisi

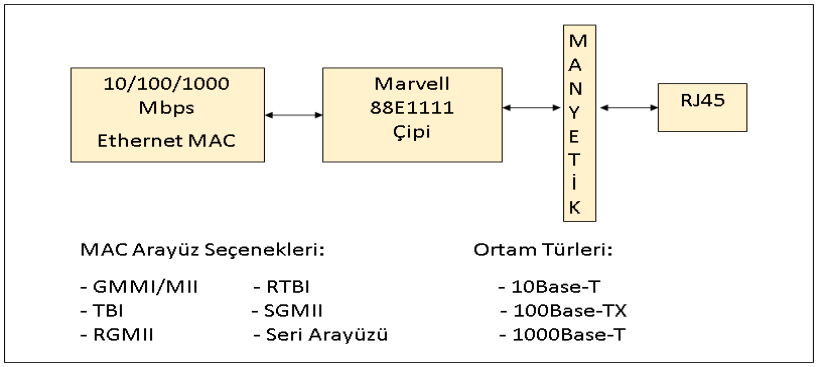
BÖLÜM 5

HABERLEŞME SİSTEMİ

Bölüm 2'de anlatılan sistemin işlemiş olduğu görüntünün, Bölüm 3'te gösterildiği üzere sıkıştırılması sonrasında hedef cihaza aktarımı bu bölümde anlatılacak olan modüller ile gerçekleştirilmiştir. Görüntünün aktarımı, koşturulan algoritmanın doğruluğunun ölçümünü yapmak için gerekli olmaktadır. Ayrıca geliştirilmiş olan bu sistemin kullanıcının kendi sistemine aktarabilmesi için görüntü aktarım işlemi kullanıcıya sunulmuştur. Bu kapsamda kullanılan Spartan6 FPGA geliştirme platformunun üzerinde bulunan PHY çipi ile Ethernet haberleşme modülü geliştirilmiştir. Haberleşme için UDP, DHCP ve ARP protokollerinin gerçeklemesi yapılmıştır.

5.1 PHY Çipi

PHY çipi OSI modelindeki ilk katman olan fiziksel katmanın görevlerini yapmaktadır. Geliştirme kartında dahili bulunması sebebi ile donanımsal olarak fiziksel katmanın tasarlanması gereksinimi ortadan kaldırmaktadır.



Şekil 5.1 88E1111 Çipinin Uygulamalardaki Kullanımı[36]

Şekil 5.1'de gösterilen MAC Arayüz Seçenekleri bölümünde, çipin kullanılabilecek olan arayüz çeşitleri verilmiştir. Aktarımda yüksek hız gereksinimi olabileceği için GMII kullanılmıştır.

Çizelge 5.1 88E1111 Çipi Bağlantı Pinleri ve Açıklamaları

Pin Adı	Açıklama
MDI	Ortam Bağımlı Arayüz. Ethernet kablolarının crossover kabloya gerek olmadan haberleşmeyi sağlar.
GTX_Clk	GMII Aktarım Saati. Gigabit bağlantı olması sebebi ile 125mhz ile çalışmaktadır. Aktarım için kullanılan saattir. TX_En, TX_Err ve TXD[7:0] sinyalleri ile senkrondur.
TX_En	Aktarım Aktif Sinyali. Bu sinyal aktarımın olduğu zamanlarda aktif olmaktadır.
TX_Err	Aktarım Hatası. Kabloda aktarım hatası olduğu zaman aktif olmaktadır.
TXD	Aktarımı yapılan verileri taşır. Aynı anda tek yönlü olarak 8 bit taşıyabilmektedir.
RX_Clk	GMII Alım Saati. Gigabit bağlantı olması sebebi ile 125mhz ile çalışmaktadır. Alım için kullanılan saattir. RX_En, RX_Err ve RXD[7:0] sinyalleri ile senkrondur.
RX_En	Aktarım Aktif Sinyali. Bu sinyal aktarımın olduğu zamanlarda aktif olmaktadır.
RX_Err	Alım Hatası. Kabloda alım hatası olduğu zaman aktif olmaktadır.
RXD	Aktarımı yapılan verileri taşır. Aynı anda tek yönlü olarak 8 bit taşıyabilmektedir.
CRS	Taşıyıcı Algılama. Alma ortamının boş olmadığı durumlarda aktiftir.
COL	Çakışma. Hem aktarım hemde alım ortamı aktif olduğunda aktif olur.

Çizelgede verilmiş olan tüm pinler sürülerek gigabit Ethernet bağlantısı gerçekleştirilmiştir. Bu aşamada aktarım için iki yol izlenebilmektedir. Birincisi aktarımın gereksinimlerine göre özel bir protokol oluşturularak haberleşmenin sağlanması olabilir. Diğer bir yol ise standart protokollerin kullanılması olacaktır. Bunun ile ilgili bilgiler 5.2 bölümde aktarılacaktır.

5.2 Aktarım Protokolleri ve Tasarımları

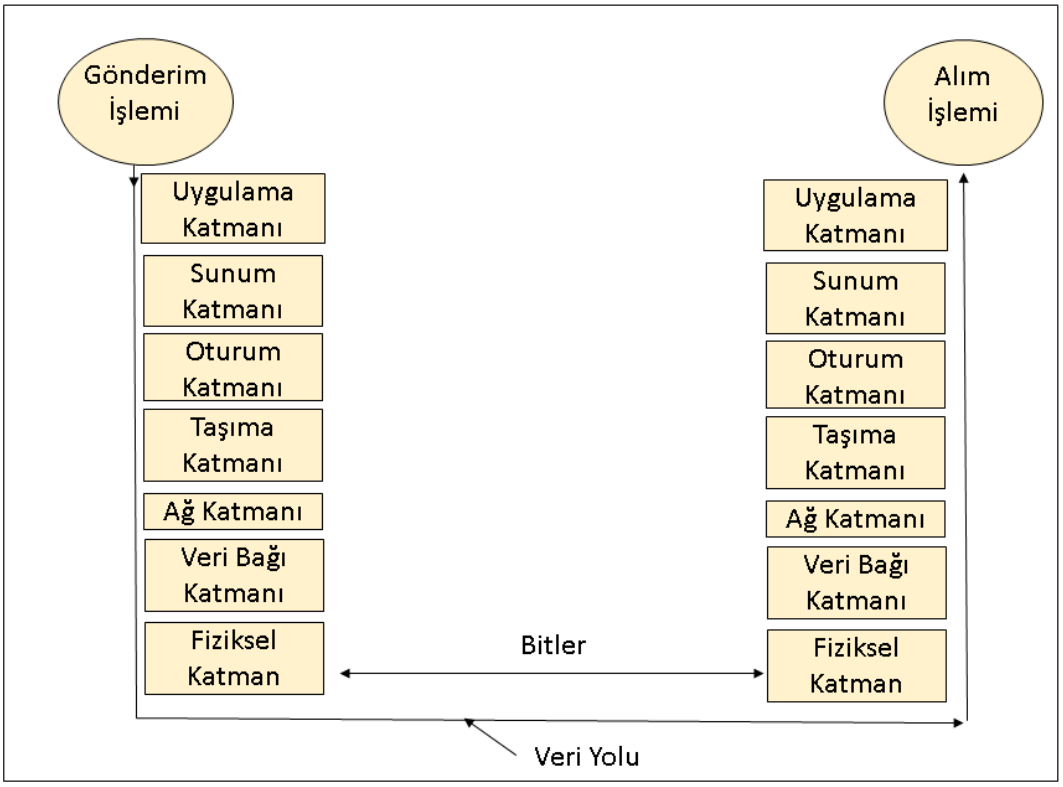
Geliştirilen sistemin genel amaçlı, kolay entegre edilebilir olması istendiği için OSI standartlarında haberleşmenin daha uygun olacağı görülmektedir. Özel protokollerin kullanılması durumunda hedef cihazda bunun açılabilmesi için ayrıca çözümler gerektirmektedir. Bu sebeplerden ötürü OSI standartlarında aktarım sağlanmıştır. Alt bölümlerde OSI modeli, kullanılan protokoller ve bu protokollerin çip üzerinde tasarımları hakkında bilgi verilecektir.

5.2.1 OSI Modeli

Uluslar arası Standartlaştırma Örgütü(ISO) tarafından geliştirilmiş olan OSI(Open System Interconnection) modeli, cihazların kendi aralarında haberleşmelerinin standartlaştırılmasını sağlamaktadır. Kullanılan donanım ya da ağ özelliklerinin haberleşme için önemi olmamaktadır. Model 7 katmandan oluşmaktadır. Bunlar sırasıyla fiziksel, veri bağı, ağ, taşıma, oturum, sunum ve uygulama katmanlarıdır. Her katman kendi ile ilgili olan görevi yaptıktan sonra verinin üst ya da alt katmana iletimini gerçekleştirirler. Katmanlara ayrılması her katmanın kendi içerisinde ayrı ayrı geliştirilebilmesi, düzenlenebilmesi, kısaca modülerlik kazandırmaktadır. [37][38] Katmanların kısaca açıklamaları aşağıda verilmiştir.

- **Fiziksel Katman**: Ağ üzerinde verinin fiziksel karakteristiğini tanımlamaktadır. Aktarımı yapılacak olan verinin, ortama uygun olarak dönüştürümünün nasıl yapılacağını belirtir. Bu ortam elektrik, ışık, radyo vb… sinyalleri içerebilir. Veriler 0 ve 1'ler şeklinde sinyaller olarak aktarılmaktadırlar. Gönderici ortama sinyalleri aktarırken, alıcı cihazın fiziksel katmanında ortamdan sinyalleri alıp 0 ve 1'ler dizisine çevirmektedir.

- **Veri Bağı Katmanı**: Fiziksel ortama bağlanmak için gerekli olan işlemler yapılmaktadır. Ağdaki cihazların belirlenmesi, o esnada ortamın kullananın tespiti, kaynak ve hedef cihazlarının aynı frekansta çalışması ve fiziksel katmandan gelen verinin hata kontrolünü gerçekleştirmektedir. Fiziksel katman ve ağ katmanı arasında format dönüşümünü sağlamaktadır. Bu katmanda işlemler önemli bir ölçüde ağ kartının içerisinde gerçekleşmektedir.
- **Ağ Katmanı**: Ağa gönderilecek olan paketin, yönlendiriciler tarafından aktarılacağı adresin eklendiği takmandır. Bu katmanda veriler paketler halinde taşınmaktadırlar. Bir üst katman olan taşıma katmanının gönderdiği istekleri yanıtlar ve alt katman olan veri bağı katmanına iletmektedir. Paketin ağdaki gideceği en uygun yolu bulmaktadır. IP ve Arp protokolü bu katmanda çalışmaktadır.
- **Taşıma Katmanı:** Üst katmanlardan aktarılan verinin bölüm işlemini gerçekleştirir. Bölümler ağın paket boyutunda olmaktadır. Taşıma katmanının üstündeki katmanlar, donanım ile ilgilenmeksizin veri ile ilgilenirler. Aynı durum alt katmanlarda, veri ile ilgilenmeksizin donanım ile ilgilenirler. TCP, UDP vb… protokoller bu katmanda bulunmaktadırlar.
- **Oturum Katmanı:** Ağdaki cihazlar arasındaki iletişimin başlaması, kullanılması ve bitirilmesi işlemlerini gerçekleştirirler. Cihazın iki ve üzeri cihaz ile iletişim halinde olduğunda, veri aktarımının doğru cihaza sağlanmasını sağlar. Netbios, Sockets gibi protokoller bu katmandadır.
- **Sunum Katmanı:** Hedef cihaza gönderilecek olan paketin, hedef cihaz için anlaşılabilir olması için gerekli dönüşümü gerçekleştirir. Verinin formatı, filtrelenmesi, sıkıştırılması vb… işlemler bu katmanda tamamlandıktan sonra oturum katmanına gönderilir. Farklı uygulamalar birbirlerinin verilerini bu şekilde kullanabilir hale gelmektedirler.
- **Uygulama Katmanı:** Katmanların en üstünde bulunmaktadır. Kullanıcıya en yakın olan katmandır. Cihazlardaki uygulamaların ağa erişimlerindeki arayüz görevini sağlar. HTTP, SMTP, POP, TTP vb… protokoller bu katmanda bulunmaktadırlar.



Şekil 5.2 OSI Katmanları

5.2.2 Haberleşme Protokolleri

Geliştirilen sistemde gerçek zamanlı görüntü aktarımı yapılmak istendiğinden dolayı yüksek hıza ihtiyaç duyulmaktadır. Aktarım için UDP veya TCP protokolü kullanılabilir. Ancak TCP protokolünde bağlantı kurulması, aktarım kontrolü ve kapanış için geçecek olan süre ek maliyet getirecektir. Dolayısıyla UDP protokolünün getirmiş olduğu hız avantajı sebebi ile UDP protokolü aktarım için kullanılmıştır. Tasarımda ihtiyaç duyulan bir başka nokta ise UDP ile haberleşmeye başlamadan önce veri alınacak olan cihaz ARP sorgusu göndermektedir. Geliştirilen sistem otomatik olarak ARP yanıtı vermektedir. Bir diğer protokol ise DHCP protokolüdür. Sistemin ağa bağlandığında otomatik olarak IP alması, kendini tanıtması için gerekli bir protokoldür. Dolayısı ile UDP, ARP ve DHCP protokolleri tasarım için gerekli olmaktadır. Aşağıdaki alt başlıklarda bu protokoller hakkında bilgi verilecektir.

5.2.2.1 **UDP Protokolü**

UDP hızlı aktarım yapması sebebi ile özellikle gerçek zamanlı görüntü/ses aktarımlarında tercih edilen bir protokoldür. Hızlı aktarmasının nedeni bağlantı kurma, kontrol ve kapatma gibi özellikleri barındırmaz. Ancak bu özelliği paketin hedefe varmasını garanti etmez. Bunu aşmak için ayrıca bir tasarım yapılmalıdır. Bu yaklaşım gecikmeyi arttıracaktır. Paket yapısı Çizelge 5.2'de verildiği gibidir[35].

Çizelge 5.2 UDP Paket Yapısı

<table>
<tr><td colspan="6">Hedef Mac Adresi 0-3 Byte</td></tr>
<tr><td colspan="3">Hedef Mac Adresi 4-5 Byte</td><td colspan="3">Kaynak Mac Adresi 0-1 Byte</td></tr>
<tr><td colspan="3">Kaynak Mac Adresi 2-5 Byte</td><td colspan="3">Veri Uzunluğu</td></tr>
<tr><td>Versiyon</td><td>Başlık Uzunluğu</td><td>Servis Türü</td><td colspan="3">Toplam Uzunluk</td></tr>
<tr><td colspan="3">Kimlik</td><td>Bayraklar</td><td colspan="2">Parça Numarası</td></tr>
<tr><td colspan="2">Yaşam Süresi</td><td>Protokol</td><td colspan="3">Başlık Sağlaması</td></tr>
<tr><td colspan="6">Kaynak IP Adresi</td></tr>
<tr><td colspan="6">Hedef IP Adresi</td></tr>
<tr><td colspan="4">IP Ayarları(Varsa)</td><td colspan="2">Doldurma</td></tr>
<tr><td colspan="3">UDP Kaynak Port</td><td colspan="3">UDP Hedef Port</td></tr>
<tr><td colspan="3">UDP Mesaj Uzunluğu</td><td colspan="3">UDP Sağlaması</td></tr>
<tr><td colspan="6">Veri</td></tr>
<tr><td colspan="6">…</td></tr>
<tr><td colspan="6">Sağlama</td></tr>
</table>

Bir UDP paketinde birçok anlam taşıyan başlıklar vardır. Verinin iletimi başlıklardaki bilgilere göre yönlendiriciler tarafından iletimi sağlanmaktadır. Tam bir paketin içereceği başlıkların anlamları aşağıda açıklanmıştır.

- **Hedef Mac Adres**i: Hedef cihazın ağ arayüzüne atanmış olan 48 bitlik benzersiz tanımlayıcı anahtardır.
- **Kaynak Mac Adresi**: Gönderen cihazın ağ arayüzüne atanmış olan 48 bitlik benzersiz tanımlayıcı anahtardır.
- **Versiyon:** Gönderilen paketin IPv4 ya da Ipv6 olduğunu belirtir.
- **Başlık Uzunluğu**: Paket başlığının uzunluğunu belirtmektedir. En az 20 byte olabilir.
- **Servis Türü:** Bağlı olunan ağın isteyebileceği hizmetleri belirtmektedir. Öncelik, gecikme, veri akışı, güvenlik, verim gibi öncelik belirtmek mümkündür.
- **Toplam Uzunluk:** Paketin byte türünden büyüklüğünü bildirmektedir.
- **Kimlik:** Kaynak tarafından verilmiş numaradır. Paketlerin birleştirilmesi esnasında bu numaralar kullanılmaktadır.
- **Bayraklar:** Paketin parçalanmaya izninin olup olmadığı bilgisi, parçalanmış olup olmadığı bilgilerini içermektedir.
- **Parça Numarası:** Eğer Veri parçalanmış ise, parçalar içerisindeki yerini belirtmektedir.
- **Yaşam Süresi:** Ağ cihazlarından en fazla geçebileceği sayıyı içermektedir. Bir paket en fazla, yaşam süresi sayısı kadar cihazdan geçebilir. Sonrasında hedefe ulaşmamış ise paket düşürülür. Bir paket en fazla 255 ağ cihazını geçecek şekilde ayarlanabilir.
- **Protokol:** Ulaşım katmanından hangi protokolün kullanılacağı bilgisini içermektedir.
- **Başlık Sağlaması:** Başlıkta hata varsa bu sağlama sayesinde tespit edilir. Hatalı paketler yok edilir.
- **Kaynak IP Adresi:** 32 bitlik kaynak IP adresini içermektedir.
- **Hedef IP Adresi:** 32 bitlik hedef IP adresini içermektedir.

- **Kaynak Port Numarası**: Hedeften bir cevap alınması gerektiren durumlarda, hedefin geri gönderirken kullanması gereken port numarasını içermektedir. Alanın sıfır olması durumunda kaynağın port numarası bulunmadığı anlamına gelmektedir.
- **Hedef Port Numarası**: İletilecek olan paketin hedef cihazdaki varacağı port'un numarasını içermektedir. Hedef port numarasının bilinmediği durumlarda sıfır ile doldurulmaktadır.
- **UDP Mesaj Uzunluğu**: UDP paketinin içeresindeki kullanıcının oluşturduğu verinin uzunluğu bulunmaktadır. Bir UDP paketi teorik olarak en fazla 65.507 byte'dan oluşabilir.
- **Sağlama:** Veri ve başlıklarda hata kontrolü için kullanılmaktadır. Eğer gönderici tarafından hesaplanmamışsa tüm bitler sıfırlar ile doldurulur. Ipv6'da doldurulması zorunludur.

5.2.2.2 **ARP Protokolü**

Ağlarda veri alış verişi için donanımsal adresin bilinmesi gerekmektedir. Çoğu cihaz daha önce veri göndermemiş olduğu bir yere veri göndermeden önce göndereceği yere o adreste birisinin olup olmadığı ve varsa fiziksel adresini isteyen bir ARP sorgusu gönderir. Bu işlem hedef cihazın IP sinin bilindiği ancak fiziksel adresinin bilinmediği durumlarda yapılmaktadır. Cihazlar gönderdikleri istek doğrultusunda yanıt aldıklarında kendi ARP tablolarına cihazın Mac adresi ve IP adresini yerleştirirler. Daha sonraki göndereceksleri paketlerde bu tabloya bakıp, mevcut kayıt varsa tekrar sorgu göndermezler. Cihazlar bu protokolün yanıtını alamadıkları takdirde veri akışına başlamazlar. Dolayısıyla geliştirilen sistem kendisine bir ARP sorgusu geldiğinde yanıtlamak durumundadır. Bu yüzden geliştirilen sisteme ARP sorgusunu yanıtlayacak tasarım sisteme dahil edilmiştir. Aşağıdaki çizelgede ARP paketinin içerdiği başlıklar ve açıklamaları yer almaktadır.[34]

Çizelge 5.3 ARP Paket Yapısı

<table>
<tr><td colspan="2">Donanım Türü</td><td>Protokol Türü</td></tr>
<tr><td>Donanım Adres Uzunluğu</td><td>Protokol Adres Uzunluğu</td><td>Operasyon Kodu</td></tr>
<tr><td colspan="3">Kaynak Donanım Adresi 0-3 Byte</td></tr>
<tr><td colspan="2">Kaynak Donanım Adresi 4-5 Byte</td><td>Kaynak Protokol Adresi 0-1 Byte</td></tr>
<tr><td colspan="2">Kaynak Protokol Adresi 2-3 Byte</td><td>Hedef Donanım Adresi 0-1 Byte</td></tr>
<tr><td colspan="3">Hedef Donanım Adresi 2-5 Byte</td></tr>
<tr><td colspan="3">Hedef Protokol Adresi 0-3 Byte</td></tr>
</table>

- **Donanım Türü**: Veri hattı katmanının ağ protokol tipini belirttiği alandır.
- **Protokol Türü**: Protokollerin bu alanda kullanabilecekleri numaralardır. Örneğin IPv4 için 0x0800 değerini almaktadır.
- **Donanım Adres Uzunluğu**: Donanım adresinin uzunluğunu byte türünden içermektedir. Ethernet adres uzunluğu 6 bytedır.
- **Protokol Adres Uzunluğu**: Üst katman protokolündeki adresin byte türünden uzunluğunu içerir. IPv4 adres 4 byte, IPv6 adresi ise 16 bytedır.
- **Operasyon Kodu**: Giden ARP paketinin istek ya da yanıt olduğunun belirtildiği operasyon kodunu içerir.
- **Kaynak Donanım Adresi**: Kaynak donanımın adresini içermektedir.
- **Kaynak Protokol Adresi**: Kaynak protokol adresini göstermektedir.
- **Hedef Donanım Adresi:** Hedef donanım adresini içermektedir. İstek içeren bir ARP paketinde bu alan FF:FF:FF:FF:FF:FF ile doludur.
- **Hedef Protokol Adresi**: Hedef protokol adresini içermektedir.

5.2.2.3 **DHCP Protokolü**

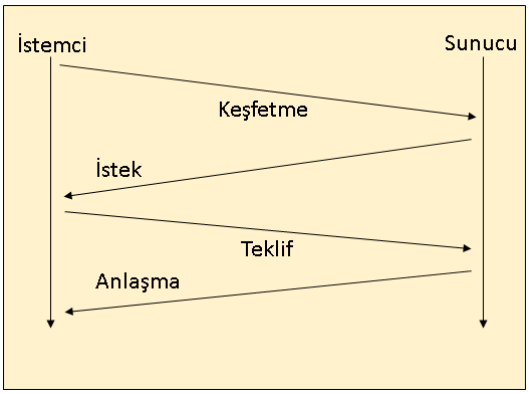
DHCP protokolü, ağ yönlendirici cihazların ağa bağlanan cihazlar ile IP ataması yaparken anlaşma kurmasını sağlamaktadır. Ağa bağlı olan her makineye otomatik olarak IP dağıtmak amacı ile geliştirilmiştir. Otomatik IP alabilmek için bu isteğin yanıtlanması zorunludur. 4 temel aşamadan oluşmaktadır. Sunucu arama, teklif, istek ve anlaşma olmak üzere 4 temel aşamadır. Geliştirilen sistem yönlendirici cihazlar ile kullanılabilir olması için DHCP protokolü sisteme dahil edilmiştir. Böylelikle sistemin yönlendirici üzerinden uzaktan erişimine imkan tanınmıştır. DHCP protokolünde Çizelge 5.4'te verildiği gibi birçok başlığı bulunmaktadır[34].

Çizelge 5.4 DHCP Paket Yapısı

Operasyon Kodu	Donanım Türü	Donanım Adres Uzunluğu	Atlama
Aktarım Kimliği			
Saniyeler		Bayraklar	
İstemci İp Adresi			
Kendi İp Adresi			
Sunucu İp Adresi			
Geçit İp Adresi			
İstemci Donanım Adresi			
Sunucu Adı			
Önyükleme Dosya Adı			
Ayarlar			

- **Operasyon Kodu:** DHCP mesajının türünü belirtmektedir. 1 ise istek, 2 ise yanıt anlamına gelmektedir.
- **Donanım Türü:** Ağda kullanılan donanım türünü belirtmektedir. Ethernet için 1 kodu bulunmaktadır.

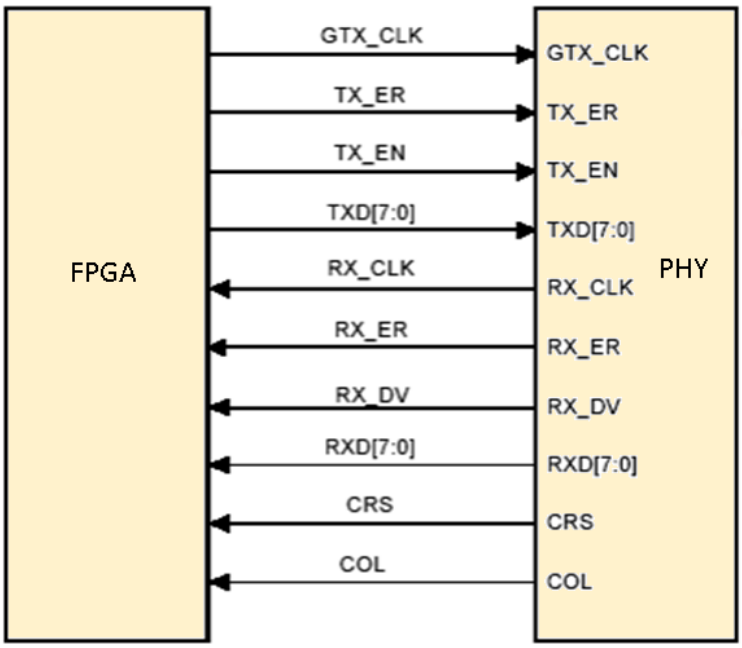
- **Donanım Adres Uzunluğu:** Donanım adresinin uzunluğunun belirtildiği alandır. Ethernet için 6 kullanılmaktadır.
- **Atlama:** Mesajın aktarılırken aktarılmış olan aktarıcı cihaz sayısı.
- **Aktarım Kimliği:** Sunuculardan gelen isteklerin istemciler tarafından karşılaştırılması için kullanılmaktadır.
- **Saniyeler:** İstemcinin DHCP protokolünü işlemeye başladığından sonra geçen süre.
- **Bayraklar:** Bir istemcinin IP adresinin bilinmediği durumlarda broadcast yapılarak istemcinin paketi alması sağlanır. Bu bit broadcast ya da olmadığını belirtmektedir.
- **İstemci IP Adresi:** İstemcinin IP adresini içermektedir. Bu adres istemcinin IP adresini doğruladığı zaman geçerli olmaktadır.
- **Kendi I Adresi:** Sunucunun doğrulamış olduğu istemci IP adresini içermektedir.
- **Sunucu IP Adresi:** Ayarlama sürecinde istemcinin kullanacağı IP adresini taşımaktadır.
- **Geçit IP Adresi:** Ağdaki yönlendirme görevini yapan cihazın IP adresini içermektedir.
- **İstemci Donanım Adresi:** İstemcinin donanım adresini içermektedir. Tanımlama ve iletişimde kullanılmaktadır.
- **Sunucu Adı:** Sunucunun ismini içermektedir.
- **Önyükleme Dosya Adı:** İsteğe bağlı olarak kullanılmaktadır. DHCPDISCOVER mesajı içerisinde geçmektedir.
- **Ayarlar:** DHCP işlemleri için bazı basit ayarları içermektedir.



Şekil 5.3 DHCP Protokol Akışı

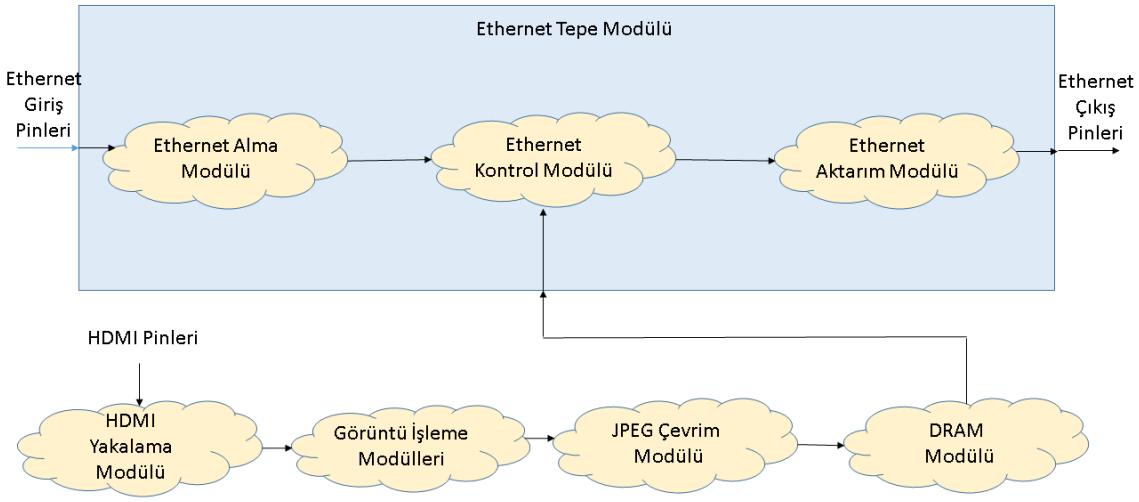
5.2.3 Haberleşme Tasarımı

Sistemin haberleşme bölümü tasarımı bölüm 5.1'de anlatılan Marvell 88E1111 çipi kullanılarak Ethernet modülü tasarlanmıştır. Gigabit Ethernet haberleşme yapılacağı için 125Mhz frekansta çalıştırılması gerekmektedir. Aşağıdaki şekilde FPGA ve kullanılan Marvell 88E1111 çipinin arasındaki bağlantı gösterilmiştir.



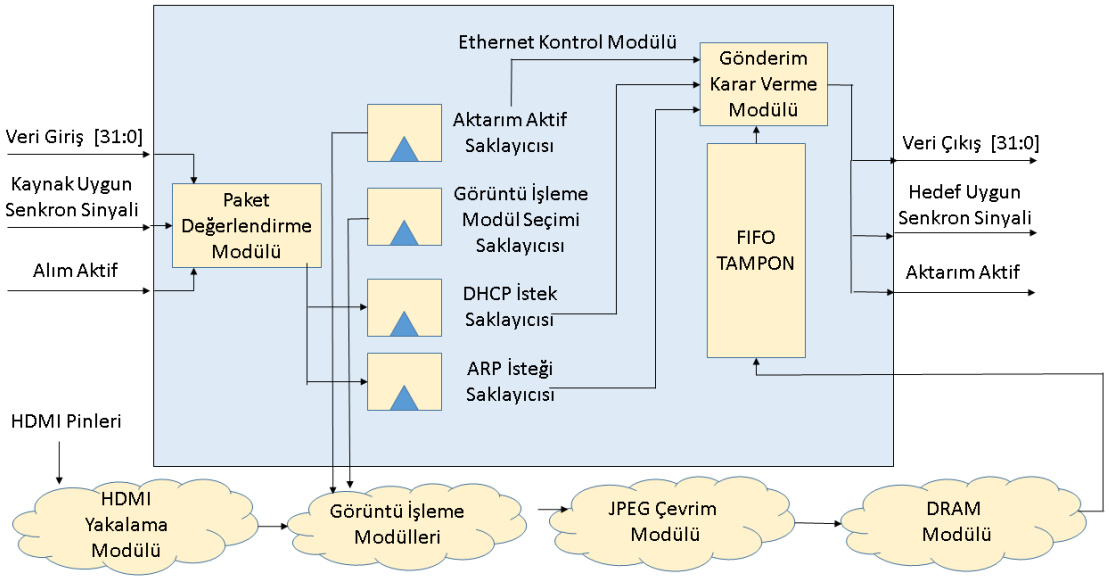
Şekil 5.4 PHY FPGA Bağlantı Şeması[39]

İşlenmiş görüntünün hedef cihaza aktarımını sağlamak için paket gönderme modülü, hedef cihazdan gelecek komutların alınması için paket alma modülü gerçekleştirilmiştir. Paket gönderme modülünün altında çalışan ARP ve DHCP modülleri gerektiği zaman aktif olmaktadırlar.



Şekil 5.5 Ethernet Tasarım Bağlantısı

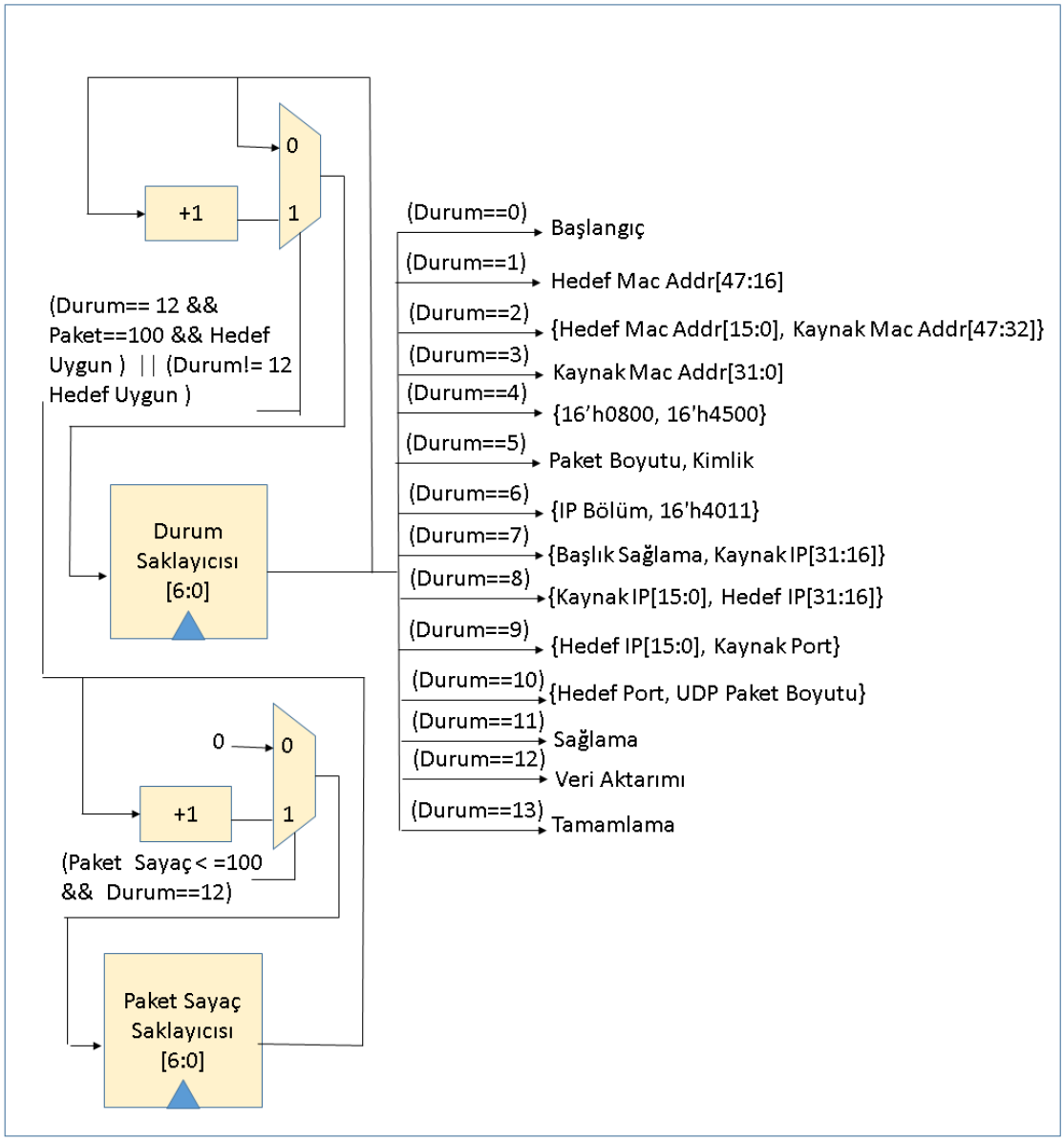
Bu modüller için sonlu durum makinesi modeli kullanılmıştır. Yapılacak işler durumlara bölünerek gerçekleştirilmiştir.



Şekil 5.6 Ethernet Tepe Modül Hiyerarşisi

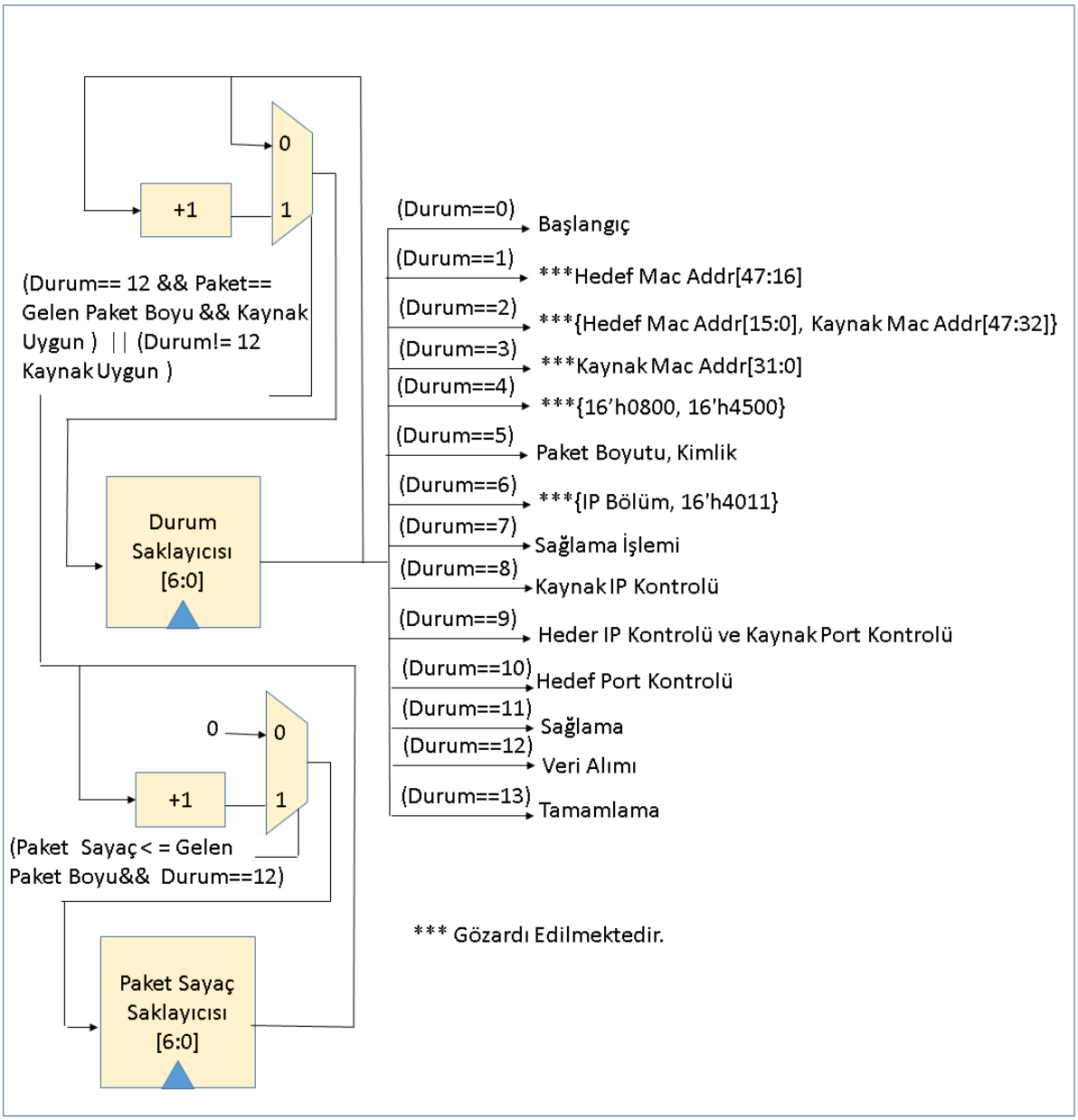
Şekil 5.6'te verilen Ethernet'ten gelen paketin değerlendirilip, gelen isteğe göre sistemin cevap verdiği hiyerarşi verilmiştir. Paket gönderme modülü 16 durumdan oluşmaktadır. Yapılan deneysel çalışmalarda gönderilen bir UDP paketinin 512 byte'dan büyük olduğu durumlarda paketin kaybolduğu ya da bozulduğu görülmüştür. Dolayısıyla gönderimi

yapılacak olan paketlerin 512 byte'i aşmamasına dikkat edilmiştir. UDP paketinde 8, Ipv4 başlığı 20, toplam başlık byte toplamı 28 byte olmaktadır. Geriye 484 byte'lık veri gönderebilecek alan kalmaktadır. Paket gönderim modülü mimarisi Şekil 5.7'de verilmiştir.



Şekil 5.7 Paket Gönderim Modülü

Dinamik olarak ayarlanabilen 32 bitlik veri setlerinin gönderilmesi için sistemde 100 değeri bulunmaktadır. Dolayısıyla 400 byte veri + 28 byte başlıklar ile 428 byte'lık paketler halinde hedef cihaza aktarımı yapılmaktadır. ARP ve DHCP protokollerini içeren modüller paket alım modülünden gelen verilere göre başlatılmaktadır. Üst bölümlerde anlatılan protokollerin başlıklarına göre Şekil 5.8'e benzer bir aktarım yapılmaktadır.

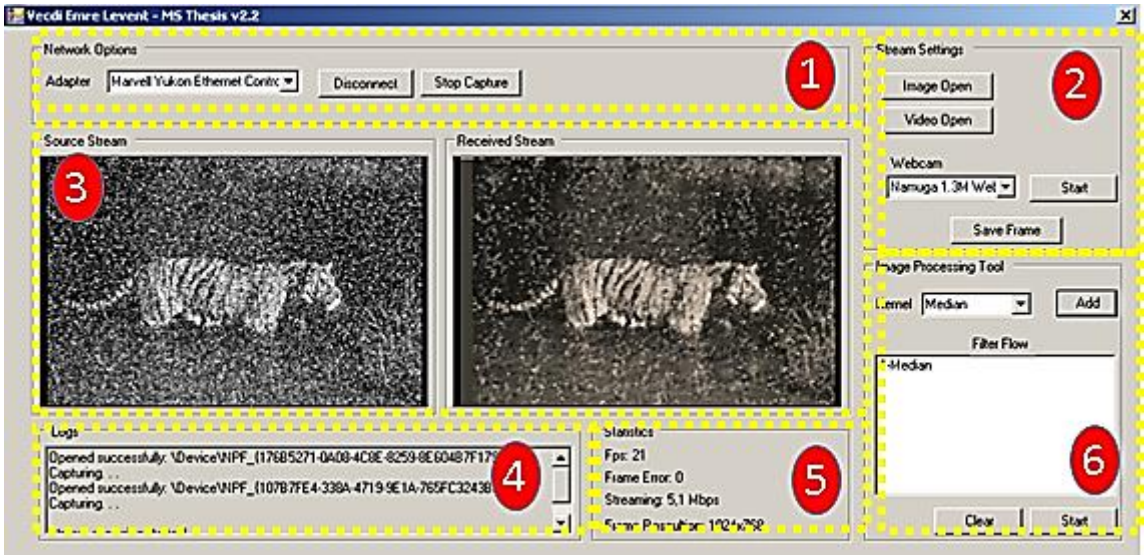


Şekil 5.8 Paket Alım Modülü

Paket alım işlemi 16 durumdan oluşmaktadır. Ağa bağlı olan FPGA'e birçok ilgisiz paket gidebilmektedir. Bunları filtrelemek amacı ile birkaç yaklaşımda bulunulmuştur. Bunlardan ilki gelen paketin UDP, ARP ya da DHCP paketi olup olmadıklarının kontrolünü yapmaktır. Bu paketlerin haricindekiler düşürülmektedir. Eğer paket bir ARP yada DHCP paketi ise, gönderici modüle gerekli olan cevap gönderilir ve bu paketler yanıtlanır. Paket UDP paketi ise göndericinin IP adresi kontrol edilir. Paket istenen adresten geldiğinde işleme konulmaktadır. Görüntü işleme bölümünde anlatılan 6 adet filtre türü, sistem ayarlarının tamamı paket alma bölümünden gelen içeriğe göre ayarlanmaktadır.

5.3 Yazılımı Tasarımı

Geliştirilen sistem hedef cihaza UDP paketleri göndermektedir. Hedef cihazda gönderilen verilerin alınabilmesi için yazılım gerekmektedir. Bu sebepten ötürü .Net ortamında soket programlama yazılmıştır.



Şekil 5.9 Uygulama Giriş Arayüzü

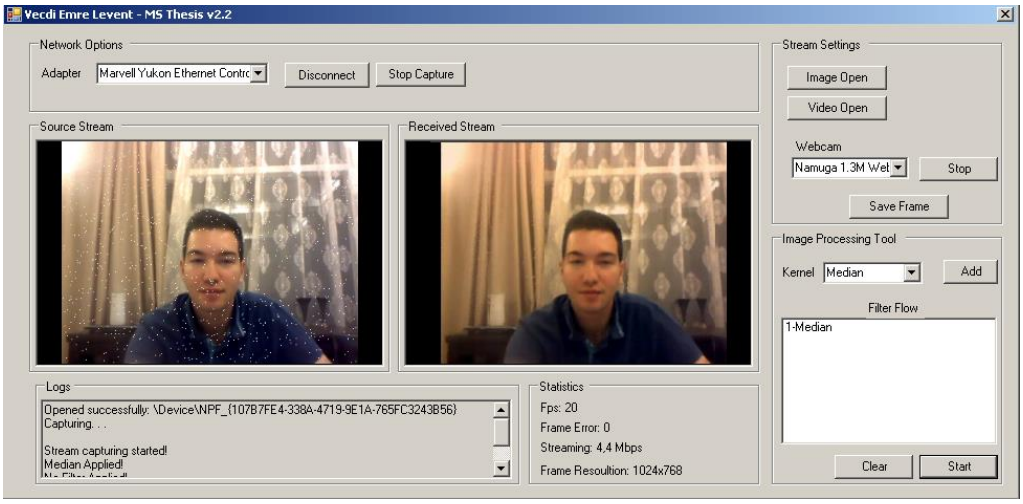
Geliştirilen uygulamada sistemin tüm parametreleri dinamik olarak ayarlanabilmektedir.

Hangi algoritmanın FPGA üzerinde koşacağının kararı Şekil 5.9'da verilen araç ile ayarlanmakta, değişiklikler uygulanmasıyla birlikte FPGA'a Ethernet portu üzerinden ilgili veri aktarılmaktadır. Yazılım 6 farklı bölümden oluşmaktadır. Bu bölümlerin görevleri aşağıda açıklanmıştır.

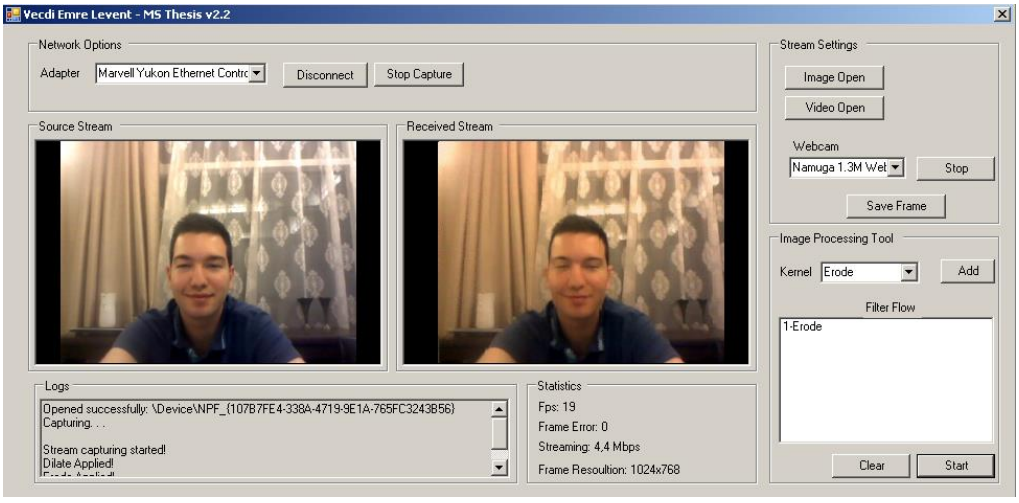
- 1. Bölüm: Ağ ayarları bulunmaktadır. Bu bölümde hangi ağ arayüzünden haberleşmenin yapılacağı seçilebilmekte ve haberleşme başlatma ve durdurma özellikleri bulunmaktadır.
- 2. Bölüm: FPGA'e beslenecek olan medya kaynağı seçilmektedir. Resim, video ve bilgisayara bağlı olan web kamerası seçilebilir.
- 3. Bölüm: Bilgisayardan gönderilen görüntü ve FPGA'den alınan çıktı görüntü gösterilmektedir.
- 4. Bölüm: Sistemin çalışırken ürettiği kayıtlar bulunmaktadır. Ethernet ile ilgili bağlanma durumları bu bölümden takip edilebilmektedir.

- 5. Bölüm: Saniyede gelen kare sayısı, hatalı gelen kare sayısı, akış hızı ve çözünürlük bilgileri bulunmaktadır.
- 6. Bölüm: FPGA üzerinde işlem yapacak olan filtrenin seçimi yapılmaktadır.

Şekil 5.10'da tuz biber gürültülü webcam görüntüsüne ortanca filtre, Şekil 5.11'de ise erozyon filtresi uygulanmaktadır.



Şekil 5.10 Uygulama Web Kamerası Medyan Filtre Çıktısı



Şekil 5.11 Uygulama Web Kamerası Aşındırma Filtre Çıktısı

Sistem'e beslenecek olan görüntü 3 farklı şekilde olabilmektedir. Bilgisayarda kayıtlı olan bir videonun ya da resmin gönderimi ve bilgisayara bağlı olan kameralardan seçilenin vereceği canlı görüntü FPGA'a gönderilebilmektedir. Bu aktarımlar HDMI üzerinden

olmaktadır. Dolayısıyla Ethernet üzerinde bant işgal etmemektedir. Ayrıca geri alınan resmin kaydedilebilmesi desteklenmektedir.

BÖLÜM 6

SONUÇ VE ÖNERİLER

Sistemin video kaynağından işlenecek kareyi almayı tamamlayıp kullanıcının ekranına ilgili kareyi göstermesi yaklaşık 0,3 saniye gecikme ile gerçekleştirilmiştir. Bu süre görüntünün alınması, işlenmesi, sıkıştırılıp Bilgisayar'a aktarılması arasında geçen süreyi ifade etmektedir. Sistem tamamen boru hattı yapısına uygun gerçekleştirilmesinden ötürü her zaman çıktı verebilecek durumdadır. Bu özelliği ile gerçek zamanlı çalışabilmektedir.

Bu sistemde kullanılan, görüntü işleme algoritmalarının ön işlem olarak ihtiyaç duyduğu filtreleme algoritmaları bilgisayara kıyasla daha yavaş koşmaktadır. Bunun sebebi aynı anda sadece bir adet karenin işlenmesidir. Tasarım birden çok karenin beslenebileceği şekilde genişletildiğinde, paralel çalışabilirliğin getirdiği avantaj ile üstel olarak performansı artacaktır.

Ayrıca paralel çalışmaya uygun karmaşık görüntü işleme algoritmalarının eklenmesi, sistemin paralellikten doğan hesaplama gücünü çoğaltacaktır. Sistemin şu andaki durumu hem işlemci tarafında hesaplamaların yapılmayışı, yani işlemcinin yorulmaması ve hem de geliştirilmeye açık, tekrar kullanılabilir bir yapıda olması en önemli avantajlarıdır. Yani tasarımcılar tarafından kolaylıkla yeni algoritmalar eklenebilmektedir.

6.1 Tasarım Sentez Sonuçları

Tasarımın sentez sonucunda kullandığı kaynaklar çizelgelerde verilmiştir. Bu kaynaklar Spartan 6 LX45 FPGA üzerinde sentezlendiğinde sentez aracının verdiği sonuçlardır.

Çizelge 6.1 Kullanılan Kaynaklar

Toplam Saklayıcı Sayıları	7,927
Toplam Arama Tabloları	9,567
Kullanılan LUT Flip-Flop Sayıları	6,958
Giriş Çıkış Bankları	43
Kullanılan BUFG/BUFGMUX Sayıları	11

Tasarımcı hangi FPGA üzerine sentezleme yapacak ise, ilgili kullanacağı FPGA'in kaynaklarına bakmalıdır. Bu kaynaklar FPGA üzerinde ne kadar filtre yapabileceğini anlamada yardımcı olmaktadır. Tam sonuç almak Xilinx ISE yazılımında hedef FPGA seçilerek sentezleme yapılması gerekmektedir. Tabloda verilmiş olan sentez sonuç değerlerinden daha az alana sahip olan bir FPGA kullanılmak isteniyorsa, tasarımda optimizasyonlara veya bazı modüllerin çıkartılması gerekli olacaktır.

6.2 FPGA, Matlab, OpenCV ve GPU Karşılaştırmaları

Intel P8700 işlemcili, 4GB RAM ve Windows 7 işletim sistemine sahip bir bilgisayarda çalışılmıştır. FPGA, Matlab ve OpenCV çıktı süreleri çizelgelerde verilmiştir. OpenCV versiyon 2.4, Matlab 2013a, GPU ise ATI RADEON HD 4660 ekran kartı ile OpenCL 1.2 kullanılarak sonuçlar elde edilmiştir. İşlemci üzerinde sonuçlar resmin okunması, işlendikten sonra tekrar sabit diske kaydedilme işlemlerinden bağımsız olarak, sadece görüntüye ilgili görüntü işleme algoritmasının uygulandığı bölümlerde geçen süre hesaba katılmıştır. FPGA üzerinde ise görüntünün aktarımı tamamlanmadan görüntü işleme algoritması koşmaya başlamaktadır. Bölüm 4'te bahsedilen satır tamponlanma yapısında 2 satır dolduktan sonra işlenmeye başladığı için sadece gecikme söz konusudur.

Çalışmada 1024x768 çözünürlüklü görüntüler ile çalışıldığı için 1024*2+3 adet pikselin gelme gecikmesi ve üzerinde ilgili çekirdeğin çıktı verme süresi bulunmaktadır. FPGA'in ilk çıktısını vermeye başlaması ile yeni pikseller sisteme verildiği müddetçe sürekli yeni çıktı vermektedir. Dolayısıyla sistem boru hattı yapısı ile tasarlandığı için çıktı süresi 1 saat çevrimi ve ilgili algoritmanın gecikmesi olarak verilebilir. Saat çevrimi FPGA'in üzerinde bulunan elemanın periyodu ile ilgilidir. Çalışmada kullanılan FPGA'in üzerinde 50 MHz, 10ns periyotu bir saat çevrimi bulunmaktadır. Kullanılan geliştirme kartı için sonuçlar verilmiştir. FPGA üzerinde alınan sonuçlar ilgili görüntü işleme filtresinin çıktı verme süreleridir. Çizelge 6.2'de işlemci üzerinde Matlab ve OpenCV ile, GPU üzerinde ise OpenCL dili ile filtrelerin performans sonuçları verilmiştir. Bu sonuçlara göre bilgisayarda koşturulan kodlar GPU platformuna göre çok daha yavaş işlenmektedir. GPU'ların paralel işlem yapma yeteneğine sahip oldukları için filtrelerdeki çarpma işlemleri paralel gerçekleştirilmektedir. Bilgisayar üzerinde ve GPU üzerindeki hesaplamalar süreleri her çağrıldıklarında değişmektedir. Bunun sebebi işletim sistemi ve GPU üzerindeki işlem karar verici mekanizmanın, arka planda gerçekleştirdiği işlemlerin ne kadar süreceğinin kesin olarak bilinememesinden kaynaklanmaktadır. Hesaplanma sürelerinin birbirine çok yakın olmasından ötürü standart sapma değeri'de 0'a yakın olmaktadır.

Çizelge 6.2 Hesaplama Süreleri Karşılaştırmaları

Süreler	Matlab	OpenCV	GPU
Ortanca Filtre	15.04ms	4.66 ms	0.23 ms
Gauss Filtre	68.60ms	5.43 ms	1,05 ms
Aşındırıcı Filtre	62.64ms	6.06 ms	2,75 ms
Genişletici Filtre	62.64ms	6.14 ms	2,75 ms
Diğer Çekirdek Tabanlı	62.77ms	11.95 ms	1,33 ms

Çizelge 6.3'te FPGA filtre çıktı verme süreleri verilmiştir. Filtrelerin çıktı verme süresi her gelen resim için sabittir. Dolayısıyla FPGA üzerindeki hesaplamaların standart sapma değeri 0 olmaktadır.

Çizelge 6.3 FPGA Filtre Çıktı Verme Süreleri

Süreler	FPGA
Ortanca Filtre	1024*2 + 6 Saat Çevrimi = 0.02054
Gauss Filtre	1024*2 + 4 Saat Çevrimi = 0.02052
Aşındırıcı Filtre	1024*2 + 5 Saat Çevrimi = 0.02057
Genişletici Filtre	1024*2 + 5 Saat Çevrimi = 0.02057
Diğer Çekirdek Tabanlı Filtreler	1024*2 + 4 Saat Çevrimi = 0.0256

Bu hesaplama süreleri görüntüye 100 defa ilgili filtrenin uygulanıp ortalamasının alınması ile elde edilmiştir.

Sonuçlarda FPGA tablosunda çıktı verme süreleri verilmiştir. Diğer platformların FPGA'in çıktı verme süresine göre karşılaştırmaları bulunmaktadır. Gerçeklenen çekirdek tabanlı filtreler diğer filtreler adı altında paylaşılmıştır. Bu filtrelerin grup olarak verilmesinin sebebi belli katsayılar ile çarpılıp toplandıkları için ortalama hesaplanma süreleri bir birinie çok yakın olmaktadır.

6.3 Öneriler ve Gelecek Çalışmalar

Çalışma tasarımın içine daha karmaşık görüntü işleme algoritmalarının eklenmesi ile geliştirilebilir. Görüntünün sıkıştırılma işlemi JPEG dönüşümü ile gerçekleştirilmektedir. Bu yaklaşım yerine MPEG gibi video sıkıştırma standartları uygulanarak yapıldığı takdirde performansın artacağı öngörülmektedir. Ethernet modülü gigabit bağlantıyı desteklemektedir. Ancak daha fazla bant genişliği ihtiyacı olan uygulamalarda PCI Express Arayüzünün tasarlanması ihtiyacı giderebilir. Son olarak tasarıma görüntünün beslenmesi HDMI portu üzerinden gerçekleşmektedir. Bu durum karelerin ardışık olarak verilmesi anlamına gelmektedir. Birbiri arasında bağımlılığın bulunmadığı karelerin işlenmesi paralel gerçekleştirilebilir. Bu durumda görüntü beslemesinin birden çok kareyi aynı anda yapabileceği hale çevrilmesi gerçekleştirilebilir.

KAYNAKLAR

[1] K Benkrid, , D Crookes ve A Benkrid, (2002). "Towards a general framework for FPGA based image processing using hardware skeletons", Parallel Computing, 28: 1141–1154.

[2] Benkrid K., Crookes D., Smith, J. ve Benkrid A., (2001). "High Level Programming for FPGA Based Image and Video Processing Using Hardware Skeletons", FCCM '01. The 9th Annual IEEE Symposium, 29 Mart-2 Nisan 2001, Kaliforniya, 219-226.

[3] Hanen C., Jean P. D., Romuald A. ve Roland C. , (2013). "Parallel embedded processor architecture for FPGA-based image processing using parallel software skeletons", EURASIP Journal on Advances in Signal Processing, 2013-153.

[4] Mittal S.,Gupta S. ve DasguptaS., (2008). "FPGA:An Efficient And Promising Platform For Real-Time Image Processing Applications", Proceedings of the National Conferenceon Research and Development in Hardware&Systems, 20-21 Haziran 2008, Kolkata.

[5] J.A. Kalomirosa ve J. Lygourasb, (2008). "Design and evaluation of a hardware/software FPGA-based system for fast image processing" Microprocessors and Microsystems, 32: 95-106

[6] M. Haldar, A. Nayak, A. Choudhary ve P. Banerjee,(2001). "Automated synthesis of pipelined designs on FPGAs for signal and image processing applications described in MATLAB" ASP-DAC '01 Proceedings of the 2001 Asia and South Pacific Design Automation Conference, 30 Ocak-2 Şubat 2001, Yokohama,645-648.

[7] S.M Loo, B.E Wells, N. Freije ve J. Kulick, (2002). "Handel-C for rapid prototyping of VLSI coprocessors for real time systems", System Theory, 2002. Proceedings of the Thirty-Fourth Southeastern Symposium on, 6-10.

[8] P. Martin, (2001). "A Hardware Implementation of a Genetic Programming System Using FPGAs and Handel-C", Genetic Programming and Evolvable Machines, 2:317-343.

[9] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammona, J. H. Anderson, S. Brown ve T. Czajkowski, (2011). “LegUp: high-level synthesis for FPGA-based processor/accelerator systems”, FPGA '11 Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays, 27 Şubat-1Mart 2011, New York, 33-36.

[10] B. Fort, A. Canis, J. Choi, N. Calagar, R. Lian, S. Hadjis, Y.T. Chen, M. Hall, B. Syrowik, T. Czajkowski, S.D. Brown ve J.H. Anderson, (2014). "Automating the Design of Processor/Accelerator Embedded Systems with LegUp High-Level Synthesis", Int'l Conference on Embedded and Ubiquitous Computing (EUC), 26-28 Ağustos 2014, Milano, 120-129.

[11] https://github.com/timvideos/HDMI2USB/wiki, 12 Ocak 2015.

[12] A. Mykyta, D. Patru, E. Saber, G. Roylance ve B.Larson, (2012). “Reconfigurable framework for high-bandwidth stream-oriented data processing”, SOC Conference (SOCC) IEEE International, 12-14 Kasım 2012, Niagara Falls, 177-183.

[13] E. Gudis, G. Wal, S. Kuthirummal ve S. Chai, (2012). “Multi-Resolution Real-Time Dense Stereo Vision Processing in FPGA”, Field-Programmable Custom Computing Machines (FCCM), IEEE 20th Annual International Symposium, 29 Nisan-1 Mart, Toronto, 29-32.

[14] M. Jacobsen ve R.Kastner, (2013). “RIFFA 2.0: A reusable integration framework for FPGA accelerators.” Field Programmable Logic and Applications (FPL), 2013 23rd International Conference, 2-4 Eylül 2013, Porto, 1-8.

[15] M. Jacobsen, Y.Freund ve R. Kastner, (2012). “RIFFA: A reusable integration framework for FPGA accelerators.”, Field-Programmable Custom Computing Machines (FCCM), 2012 20th IEEE Annual International Symposium, 29 Nisan-1 Mayıs 2014, Toronto, 216-219.

[16] http://www.fpgasimulation.com/systemverilog_primer, 22 Aralık 2014.

[17] http://www.asic-world.com/verilog/synthesis1.html, 25 Aralık 2014.

[18] J. Cong ve K. Minkovich, (2007). “Optimality Study of Logic Synthesis for LUT-Based FPGAs”, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, 26: 230-239.

[19] http://svenand.blogdrive.com/archive/105.html, 30 Aralık 2014.

[20] http://www.digilentinc.com/atlys/, 15 Aralık 2014.

[21] B. Feng, XAPP495, (2010). Implementing a TMDS Video Interface in the Spartan-6 FPGA, 13 Aralık 2010.

[22] http://www.vesa.org, 15 Aralık 2014.

[23] http://www.ce.org, 10 Aralık 2014.

[24] http://www.hdmi.org, 10 Aralık 2014.

[25] http://en.wikipedia.org/wiki/Extended_Display_Identification_Data, 10 Aralık 2014.

[26] M. Krepa, (2013). “EV_JPEG_ENC IP Core”, opencores.org, 10 Ocak 2015.

[27] http://blog.teledynedalsa.com/2012/05/image-filtering-in-fpgas, 5 Aralık 2014.

[28] http://www.hackersdelight.org/divcMore.pdf, 5 Aralık 2014.

[29] H. Yueli ve H. Ji , (2009). “Research on Image Median Filtering Algorithm and Its FPGA Implementation”, Intelligent Systems, 2009. GCIS '09. WRI Global Congress, 19-21 Mayıs 2009, Xiamen, 226-230.

[30] S.S.Tavse, P.M.Jadhav ve M.R.Ingle, (2012). “Optimized Median Filter Implementation on FPGA Including Soft Processor”, International Journal of Emerging Technology and Advanced Engineering, 2:236-239.

[31] J. Diaz, E. Ros, F. Pelayo ve E.M. Ortigosa, (2006). “FPGA-based real-time optical-flow system”, Circuits and Systems for Video Technology IEEE Transactions on, Şubat 2006, 274-279.

[32] http://homepages.inf.ed.ac.uk/rbf/HIPR2/unsharp.htm, 23 Kasım 2014.

[33] S. Harding ve W. Banzhaf, (2009). “Genetic programming on GPUs for image processing”, International Journal of High Performance Systems Architecture, 1: 231-240.

[34] V. Sanduja ve R. Patial , (2012). “Sobel Edge Detection using Parallel Architecture based on FPGA”, International Journal of Applied Information Systems, 3: 20-24.

[35] X. Zhai, F. Bensaali ve S. Ramalingam, (2011). “Real-time license plate localisation on FPGA”, Computer Vision and Pattern Recognition Workshops (CVPRW), 20-25 Haziran 2011, Kolorado, 14-19.

[36] http://www.marvell.com/transceivers/assets/Marvell-Alaska-Ultra-88E1111-GbE.pdf, 20 Ocak 2015.

[37] N. Briscoe, (2000). Understanding the OSI 7-layer model. PC Network Advisor, 120: 13-14.

[38] bidb.itu.edu.tr/seyirdefteri, 4 Temmuz 2014.

[39] http://opencores.org/project,ethernet_tri_mode, 14 Temmuz 2014.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı :Vecdi Emre LEVENT

Doğum Tarihi ve Yeri :06.10.1991 Konak

Yabancı Dili :İngilizce

E-posta :emre@levent.tc

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Mühendisliği	İstanbul Arel Üniversitesi	2013
Lise	Fen Bilimleri	Özel Silifke Lisesi	2009

İŞ TECRÜBESİ

Yıl	Firma/Kurum	Görevi
2012	Aselsan	Stajyer
2011	IBM	Stajyer
2011	Türk Hava Yolları	Stajyer

YAYINLARI

Bildiri

1. **Levent V.E., Karabiber F., "FPGA Tabanlı Gerçek Zamanlı bir Görüntü İşleme Sisteminin Gerçeklenmesi", ASYU 2014 Akıllı Sistemler ve Uygulamaları, Katip Çelebi Üniversitesi, İzmir (9-10 Ekim) 2014**

2. **Elbir A., Levent V.E., Karabiber F., "A Software Tool for Evaluating the Effect of Proximity Measures on Clustering Methods" IRED CEET, Kuala Lumpur, Malaysia (2-3 Ağustos) 2014**

3. **Levent V. E., Diri B., "Türkçe Dokümanlarda Yapay SinirAğları ile Yazar Tanıma", Akademik Bilişim 2014, Mersin Üniversitesi, Mersin, (5-7 Şubat), 2014**

Proje

1. **TÜBİTAK 1001 Projesi, M-RIVA: Video İşleme Algoritmalarının FPGA'ler üzerinde Gerçek-Zamanlı olarak Gerçeklenmesi için Metodoloji Geliştirilmesi, Bursiyer**