ÇİP ÜZERİNDE SİSTEM MİMARİLİ FPGA KULLANARAK GERÇEK ZAMANLI GÖRÜNTÜ İŞLEME ALGORİTMALARININ GERÇEKLEŞTİRİLMESİ

Recep ÖZALP

Yüksek Lisans Tezi
Mekatronik Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Ayşegül UÇAR
AĞUSTOS-2018

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇİP ÜZERİNDE SİSTEM MİMARİLİ FPGA KULLANARAK GERÇEK ZAMANLI GÖRÜNTÜ İŞLEME ALGORİTMALARININ GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Recep ÖZALP

(161134102)

Tezin Enstitüye Verildiği Tarih : 17 Temmuz 2018
Tezin Savunulduğu Tarih : 02 Ağustos 2018

Tez Danışmanı: Doç. Dr. Ayşegül UÇAR (F.Ü)

Diğer Jüri Üyeleri: Dr. Öğr. Üyesi Özal YILDIRIM (M. Ü.)

Doç. Dr. Mehmet KARAKÖSE (F.Ü)

AĞUSTOS–2018

ÖNSÖZ

Hem lisans hem de yüksek lisans eğitimim boyunca beni sürekli destekleyen ve fikirlerinden istifade ettiğim, beraber çalışma imkânı bulduğum değerli hocam Sayın Doç. Dr. Ayşegül UÇAR'a bu çalışmamda da benim için çok değerli vaktini ayırdığı için teşekkürlerimi sunarım.

Bu çalışma, Fırat Üniversitesi Bilimsel Araştırma Projeleri (FÜBAP) komisyonu tarafından MF.17.33 proje numarasıyla desteklenmiştir. FÜBAP komisyonuna teşekkür ederim.

Son olarak da, hayatım boyunca her daim yanımda olan, sevgi ve desteklerini hiçbir zaman esirgemeyen aileme teşekkürü borç bilir, saygılarımı sunarım.

Recep ÖZALP
ELAZIĞ-2018

İÇİNDEKİLER

Sayfa No

ÖZET

Bu tezde, gri seviye görüntü dönüşümü ve kenar bulma algoritmaları, Zynq-7000 mimarisine sahip FPGA üzerinde gerçek zamanlı olarak uygulanmıştır.

Oluşturulan deney düzeneğinde, gömülü sistem uygulaması için Picozed gömülü görüntü işleme kartı kullanılmıştır. Kullanılan kart, üzerinde Xilinx Zynq-7000 mimarisine sahip XC7Z030 FPGA barındırmaktadır. Zynq-7000 mimarisi ARM işlemci ve FPGA bloklarına sahip olduğu için, yazılımsal ve donanımsal olarak programlanabilir yapıdadır. Bu iki yapı arasındaki iletişim Gelişmiş Genişletilebilir Arayüz (AXI-Advanced eXtensible Interface) üzerinden sağlanmaktadır. Ayrıca, ARM işlemci bir işletim sistemi kullanmaya da imkân tanımaktadır. Xilinx tarafından sağlanan Linux işletim sistemi, FPGA'da kullanılan donanımlar ve kartın barındırdığı donanımlar için sürücü desteği sağlamaktadır.

Bu tezde, kartın programlanmasında Yazılım Tanımlı Çip Üzerinde Sistem (SDSoC-Software Defined System on Chip) programlama ortamı seçilmiş ve AVNET tarafından sağlanan Picozed uyumlu SDSoC görüntü işleme platformu kullanılmıştır. Programlamada, C/C++ dilleri kullanılarak kartta kurulan Linux işletim sisteminde çalışan proje oluşturulmuştur. Görüntü işleme uygulamaları için aynı ortam donanımsal programlama yapmayı da mümkün kılmıştır. Donanımsal olarak gerçekleştirilmek istenen hesaplamalar, donanım hızlandırmalı hesaplama yapısı sayesinde doğrudan aynı ortamda tasarlanmıştır. Seçilen görüntü işleme uygulamaları; 1920X1080 piksel çözünürlükte, 60 çerçeve/saniye hızında gerçek zamanlı olarak gerçekleştirilmiştir. Uygulanan kenar bulma algoritmaları, Sobel, Prewitt ve Roberts kenar tespit yöntemleridir. Gömülü sistemin HDMI girişinden alınan görüntü verileri işlenerek HDMI protokolünde çıkışa aktarılmıştır.

Tezde her bir uygulama, hem yazılımsal hem de donanımsal olarak gerçekleştirilmiş ve performansları karşılaştırılmıştır.

Anahtar Kelimeler: FPGA, Zynq-7000, Picozed gömülü görme kartı, SDSoC, Kenar algılama

SUMMARY

IMPLEMENTATION OF REAL TIME IMAGE PROCESSING ALGORITHMS BY USING SYSTEM ON CHIP FPGA ARCHITECTURE

In this thesis, gray level image transformation and edge detection algorithms are implemented in real time on Zynq-7000 architecture FPGA.

In the experimental setup created, Picozed embedded vision card was used for embedded system application. The card used contains the XC7Z030 FPGA with Xilinx Zynq-7000 architecture. Zynq-7000 architecture is software and hardware programmable because it has ARM processor and FPGA blocks. Communication between these two structures is provided via the Advanced Extensible Interface (AXI). The ARM processor also allows the use of an operating system. The Linux operating system provided by Xilinx provides driver support for the hardware used in the FPGA and the hardware the card contains.

In this thesis, the Software Defined System on Chip (SDSoC) programming environment was chosen for the card programming and the Picozed compatible SDSoC image processing platform provided by AVNET was used. In the program, a project that runs on the Linux operating system installed on the card was created using C/C ++ languages. It is also possible to do the same media hardware programming for image processing applications. The hardware calculations are designed directly in the same environment thanks to the hardware accelerated calculation structure. Selected image processing applications; It was realized in 1920X1080 pixel resolution, 60 frames/second in real time.Applied edge detection algorithms are Sobel, Prewitt and Roberts edge detection methods. The image data received from the HDMI input of the embedded system is processed and transferred to the output in the HDMI protocol.

In this thesis, each application was realized both in software and hardware and their performances were compared.

Keyword: FPGA, Zynq-7000, Picozed embedded vision card, SDSoC, Edge detection

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

KISALTMALAR LİSTESİ

FPGA : Alanda Programlanabilir Kapı Dizileri-Field Programmable Gate Array
CLBs : Yapılandırılabilir Lojik Bloklar-Configurable Logic Blocks
VLSI : Çok Geniş Ölçekli Tümleşim-Very Large Scale Integration
LUT : Başvuru Tablosu -Look-Up Table
ASIC : Uygulamaya Özgü Tümleşik Devre-Aplication Specific Integrated Circuit
VHSIC : Çok Hızlı Tümleşik Devreler-Very High Speed Integrated Circuit
VHDL : VHSIC Donanım Tanımlama Dili-VHSIC Hardware Description Language
SRAM : Statik Rastgele Erişimli Bellek-Static Random Acces Memory
SoC : Çip Üzerinde Sistem-System on Chip
DWT : Ayrık Dalgacık Dönüşümü-Discrete Wavelet Transform
IDWT : Ters Ayrık Dalgacık Dönüşümü-Inverse Discrete Wavelet Transform
DSP : Dijital Sinyal İşlemcisi-Digital Signal Processing
AXI : Gelişmiş Genişletilebilir Arayüz-Advanced eXtensible Interface
AMBA : ARM Gelişmiş Mikrodenetleyici Veriyolu Mimarisi-Advanced Microcontroller Bus Architecture
IP : Fikri Mülkiyet-Intelectual Property
ADAS : Gelişmiş Sürücü Destek Sistemleri-Advanced Driver Assistance Systems
GPS : Küresel Konumlandırma Sistemi-Global Positioning System
PS : İşlemci-Processing System
PL : Programlanabilir Lojik-Programmable Logic
IDE : Entegre Tasarım Ortamı-Integrated Development Kit
XSDK : Xilinx Yazılım Geliştirme Ortamı-Xilinx Software Development Kit
HLS : Yüksek Seviyeli Sentez -High Level Synthesis
SDSoC : Yazılım Tanımlı SoC-Software Defined SoC
DMA : Doğrudan Bellek Erişimi-Direct Memory Acces
POSIX : İşletim Sistemi Arabirimi-Portable Operating System Interface for Unix
GUI : Grafiksel Kullanıcı Arayüzü-Graphical User Interface
RTOS : Gerçek Zamanlı İşletim Sistemi-Real Time OS
SCI : Sistem Çağrı Arayüzü-System Call Interface
BSP : Bord Destek Paketi-Board Support Package
FSBL : Birinci Aşamalı Önyükleyici-First Stage Bootloader
RAM : Rasgele Erişimli Bellek -Random Access Memory-
SSBL : İkinci Aşamalı Önyükleyici-Second Stage Bootloader
SOM : Birim Üzerinde Sistem-System On Module
OCM : Çip Üzerinde Bellek- on Chip Memory

1. GİRİŞ

Son teknolojik gelişmeler hayatı hızlı bir hale getirmiştir. Hayatın hızlanmasıyla daha hızlı cihazlara gereksinim eskisinden çok daha fazla artmaktadır. Bu alanda birçok çalışma yapılmasına ve gelişmeler olmasına rağmen hayat, sürekli daha hızlısına muhtaç bir hale evirilmektedir. Bu ihtiyaç hayatın her alanında etkisini göstermektedir. Ekonomiden sağlığa, ulaşımdan güvenliğe ve haberleşmeye kadar çoğu alan çok daha hızlı sistemlere ihtiyaç duymaktadır.

Günümüzde artık bütün sistemler dijital olma yolunda ilerlemektedir. Artık çoğu işler dijital cihazlar sayesinde yapılmaktadır. Dünyadaki hızlanmadan kasıt, aslında dijital cihazların hızlanmasıdır. Dijital sistemler ne kadar hızlıysa hayat da o kadar hızlı olacaktır. Bu sebeple hızlanma gereksinimine çare olarak dijital sistemleri hızlandırmaya yönelik sayısız çalışma yapılmaktadır.

Klasik dijital işlemciler, seri işlem yapma özelliğine sahip olduğundan işlem süresi kullanılan işlemcinin hızına bağlı olarak değişir. Ayrıca dijital işlemcilerde programlama işlemi önceden oluşturulmuş sabit mimari üzerinde gerçekleşir. Bu mimari üzerinde gerçekleştirilen işlemlerin yoğunluğuna göre işlem süresi uzar ya da kısalır. Kapsamlı işlemlerin olduğu uygulamalarda işlem süresi artmaktadır. Hız gerektiren sistemler için seri işlem özelliğine sahip işlemciler yeterli olmamakta bunun yerine paralel işlem yeteneğine sahip işlemciler kullanılmaktadır. Paralel hesaplama yeteneğine sahip cihazlardan biri de Alanda Programlanabilir Kapı Dizileri (FPGA-Field Programmable Gate Array)'lardır. FPGA'lar özelleştirilebilir mimari yapıya sahip olduklarından kullanıcı isteklerine cevap verecek niteliktedir. Ayrıca paralel programlamaya imkân tanıması, eşzamanlı iş yükü olan sistemlerde çok önemli bir çözüm olarak ön plana çıkmaktadır. Ancak esnek programlamayı sağlarken yapısında bir dezavantaj oluşur. Esnek programlamaya imkân tanıyan kapı blokları silikon yapının çok verimsiz kullanılmasına neden olur [1]. Bu dezavantajı ortadan kandırmak için Çip Üzerinde Sistem (SoC- System on Chip) mimarili FPGA'lar üretilmiştir. Bu FPGA'lar, programlanabilir FPGA bloklarının yanında bir de aynı çip üzerinde ARM-A9 mikro denetleyicisi barındırır. Bu sayede çok önemli hız ve paralellik gerektiren işlemler için FPGA blokları kullanılabilecek, görece daha az hız gerektiren sistemlerde gereksiz FPGA blokları kullanımından kaçınmak sağlanacaktır. Ayrıca bu yapı programlamada da bir esneklik sağlar. Böylece, çalışmalar için çok önemli bir konumda bulunmayan yan uygulamalar için

görece zor bir programlama yapısına sahip FPGA programlamaya harcanacak zaman da azaltılmış olur. Günümüzde artık birçok gömülü sistem tasarımında bu şekilde heterojen bir yapıya sahip tasarımlar kullanılmaktadır [2].

Sayısal hesaplama hızlarının artmasıyla beraber, yüksek işlem kapasitesine ihtiyaç duyan görüntü işleme uygulamaları da yaygınlaşmaktadır. Tıp, güvenlik, askeri, endüstri, bilimsel araştırmalar, otomotiv, havacılık gibi geniş uygulama alanına sahiptir [3]. Görüntü işleme uygulamalarının 2013 yılında yapılan bir araştırmada yaklaşık olarak 25 milyar dolar pazar payına sahip olduğu tespit edilmiştir [4]. Günümüzde bu rakamın teknolojik gelişmelerle beraber en az 10 katına çıktığı tahmin edilebilir. Görüntü işleme uygulamalarında birçok piksele sahip görüntülerin kısa zamanda işlenmesi gerekir. FPGA'ların görüntü işleme uygulamalarında kullanılma sebebi; enerji verimliliği, paralel işlem kapasitesi yetenekleridir. FPGA'lar birçok gerçek zamanlı uygulamalar için tercih edilmiştir [5].

Neshatpour, Arezou ve diğerleri [6]; Xilinx Zynq-7000 FPGA üzerinde büyük biyomedikal verilerin donanım hızlandırmalı hesaplama kullanılarak işlenmesi çalışması yapmışlardır.

Onat [7], Sobel, Pprewit ve Laplace filtreleri kullanarak gerçek zamanlı video sinyallerinin işlenmesini gerçekleştirmiştir. Bu çalışmayı Onat, Xilinx Zynq-7000 FPGA'sı üzerinde donanımsal olarak gerçekleştirmiştir.

Han ve Oruklu [8], gerçek zamanlı trafik işareti algılama işleminde Zynq-7000 kullanmıştır.

Abdelgawad, Safar ve Wahba [9]; Zynq-7000 üzerinde gerçekleştirdiği çalışmalarında; yüksek seviyeli sentez ortamında Canny kenar algılama algoritması gerçekleştirmiştir.

Dobai ve Sekanina [10], görüntü filtreleme işlemlerini Zynq platformu üzerinde gerçekleştirmişlerdir.

Russel ve Fischaber [11], Zynq platformu üzerinde OpenCV temelli yol çizgilerini algılama çalışması yapmıştır.

Ayadi, Loukil ve diğerleri [12], Zynq platformu üzerinde Linux işletim sistemi kullanarak Yüksek Verimli Video Kodlaması uygulamasını gerçek zamanlı olarak gerçekleştirmiştir.

Yu, Yang ve diğerleri [13], gerçek zamanlı yaya izleme için ölçeklenebilir donanım/yazılım platformu çalışmasını Zynq-7000 platformunda gerçekleştirmiştir.

Khongprasongsiri, Kumhom ve diğerleri [14], gerçek zamanlı şerit tanıma sistemini yüksek seviyeli sentez kullanarak Zynq-7000 üzerinde 130 çerçeve/saniye hızında ve 480x270 piksel çözünürlükte gerçekleştirmiştir.

Srijongkon, Duangsoithong ve diğerleri [15], araç sayma sistemini; adaptif arka plan yöntemini kullanarak SDSoC ortamında programlayarak Zynq üzerinde gerçekleştirmişlerdir.

Yao, Liu ve diğerleri [16], lazer profil ölçümü çalışmasını Zynq platformunda gömülü olarak gerçekleştirmiştir.

FPGA'lar daha birçok görüntü işleme uygulamasında kullanım alanı bulmuştur [17-22].

Tezin amacı; günümüzde geniş uygulama alanına sahip bazı temel görüntü işleme uygulamalarını, çip üzerinde sistem mimarili FPGA kullanarak, yüksek çözünürlükte ve gerçek zamanlı olarak gerçekleştirmektir.

Bu tez, dört ana bölümden oluşmaktadır. Birinci bölüm, FPGA'lar ve Zynq mimarisine sahip FPGA'lar ile ilgilidir. Bu bölümde, FPGA'ların temel yapısından Zynq mimarisine evirilmesi, bu mimarinin özellikleri, faydaları ve kullanım alanlarına değinilmiştir.

Tezin ikinci bölümünde, Zynq mimarisinin desteklediği işletim sistemi yapıları ve özel olarak tezde de uygulama kısmında kullanıldığı için Linux işletim sisteminden bahsedilmiş ve temel yapısı ve çalışma mantığı üzerinde durulmuştur.

Üçüncü bölümde, tezin ana konusunu oluşturan görüntü işleme algoritmaları üzerinde durulmuş, çalışma yapısı ve türleri hakkında bilgi verilmiştir.

Son bölüm ise önceki bölümlerde anlatılan konuların Zynq platformu üzerinde gerçek zamanlı uygulamaları yapılmıştır. Bu uygulamalar; alınan görüntüyü doğrudan iletme, gri seviye renk dönüşümü ve kenar bulma çalışmalarını kapsamaktadır.

2. ÇİP ÜZERİNDE SİSTEM MİMARİLİ FPGA'LAR

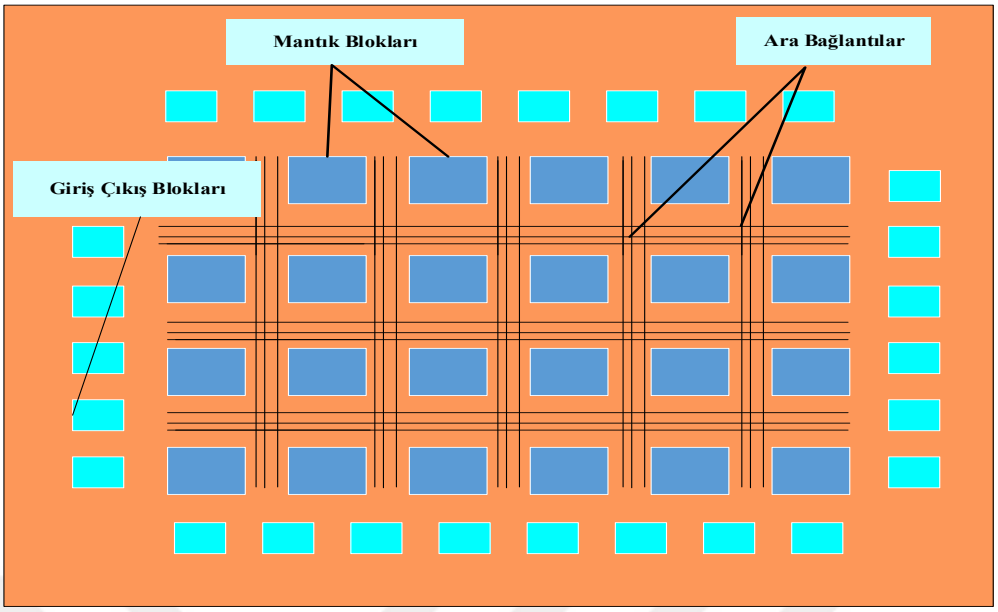
2.1. FPGA

1985 yılında aynı zamanda Xilinx'in kurucusu Ross Freeman tarafından icat edilen FPGA'lar, mimari yapısı itibariyle lojik bloklar ve ara bağlantılarından oluşan ve bu bağlantıların tamamı kullanıcı tarafından programlanabilen entegre devreleridir.

Tasarımcı, FPGA'da tasarlayacağı devreyi donanım tanımlama dilini VHSIC Donanım Tanımlama Dili (VHDL- VHSIC Hardware Description Language) ya da Verilog kullanarak tasarlar. Yazılan kod bilgisayar destekli tasarım yazılımı yardımıyla, sentezlenir. Başka bir yazılım aracı kullanılarak daha önce sentezlenmiş koda karşılık gelen uygun bağlantılar FPGA içinde gerçekleştirilir. Gerekli veri akışı oluşturulduktan sonra yazılım, kodları FPGA içerisine aktararak FPGA'yı programlar [23].

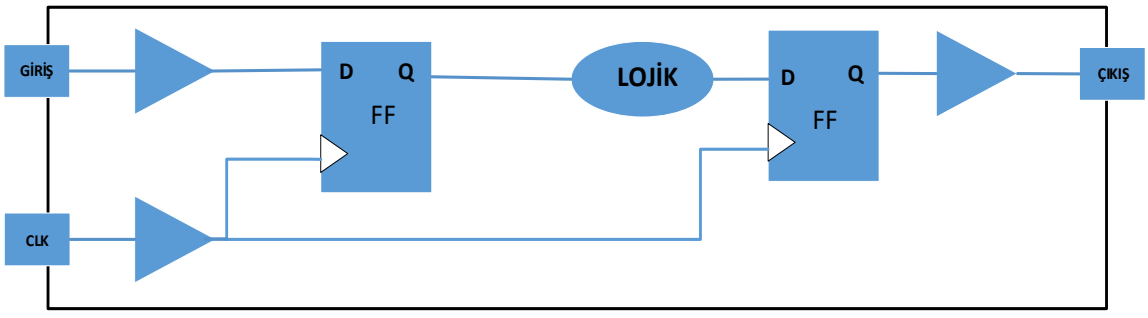
2.1.1. FPGA'nın Yapısal Özellikleri

Sayısal elektronikte zaman içerisinde yaşanan gelişim çok büyük ölçekli sistemlerin tasarımını da beraberinde getirmiştir. Yarıiletken ve üretim teknolojilerinde gelinen nokta sayesinde bugün Çok Geniş Ölçekli Tümleşim (VLSI-Very Large Scale Integration) çipleri içerisinde milyonlarca transistor bulunmaktadır. Malzeme ve üretim teknolojilerinin daha iyi noktalara tasınmış olması bunun son ürünlere de yansıtılması gerekliliğini beraberinde getirmektedir. Nitekim gelişen teknoloji yanında tüketici istekleri de artan oranda büyümektedir. Bilimdeki gelişmeleri mühendislik anlayışı içerisinde en kısa sürede ve verimlilikte tüketiciye ulaştırabilmek amacıyla yeni araç ve gereçler mühendislerin hizmetine sunulmaktadır. FPGA sayısal tasarım mühendislerinin, bilimin yarattığı imkânları teknolojiye yansıtabilmesi için geliştirilmiş bir çiptir [24]. FPGA'lar üç temel yapıdan oluşmaktadır. Bunlar: programlanabilir mantık blokları (CLBs- Configurable Logic Blocks), giriş çıkış blokları (I/O-Input/Output) ve ara bağlantı bloklarıdır. Şekil 2.1'de FPGA ların temel yapısı gösterilmiştir.



Şekil 2.1. FPGA'nın iç yapısı

FPGA, içerisinde bulunan tipik bir mantık devresi flip-flop dizileri arasına sıkıştırılmış tümleşik mantık bloklarından oluşmaktadır. Tümleşik bloklar, çıkışı yalnızca girişine bağlı olan alt devreler olarak bilinmektedir. İki girişli AND, OR gibi tüm iyi bilinen basit mantıksal fonksiyonlar bu gruba dahildir. Herhangi bir karmaşıklık seviyesinde olan bir fonksiyon bu basit bloklar kullanılarak gerçeklenebilir. Yol seçiciler (multiplexer), kodlayıcılar (encoder), kod çözücüler (decoder) gibi yapılar tümleşik bloklara örnekler olarak verilebilmektedirler. Bu devrenin girdisi birçok bitten oluşabilir. Şekil 2.2'de bu flip flop devresinin tasarımı gösterilmektedir.



Şekil 2.2. FPGA mantıksal tasarım devresi

Bugün gelinen nokta itibariyle FPGA üretimi alanında, çok fazla sayıda olmamakla birlikte çeşitli üretici firmalar mevcuttur. Bu firmaların geliştirdiği FPGA'lar farklı mimarilere ve üretim teknolojilerine dayanmaktadır. Farklı mimarilerin ve teknolojilerin birbirlerine göre üstünlükleri, güvenlik, güç tüketimi ve maliyet gibi ölçütlere göre değerlendirilir [25]. FPGA'lar, üretici firmalara göre değişen farklı mimarilere sahip olmasının yanında temel olarak tasarım yapılandırmasının büyük kısmını Statik Rastgele Erişimli Bellek (SRAM-Static Ramdom Access Memory) hücreleri ile sağlar. Bu nedenle binlerce kez programlanabilir bir yapıya sahiptirler. FPGA'lar programlanırken hedeflenen mantıksal yapıyı gerçekleştirmek için farklı yapıları kullanabilirler. MUX ve LUT tabanlı hücre mimarileri en çok kullanılan hücre mimarileridir [26]. Başvuru Tablosu (LUT-LookUp Table) tabanlı hücre mimarisine sahip bir FPGA'da, mantık hücreleri giriş sinyallerinin alacağı her değer için sonucun ne olacağının tutulduğu bir başvuru tablosu içermektedir. MUX tabanlı hücre mimarisinde ise bir mantık hücresi sadece çoğullayıcılardan oluşur ve istenen fonksiyon bu yapılarla gerçeklenir.

FPGA'lar, sabit fonksiyonlu Uygulamaya Özgü Tümleşik Devre (ASIC-Application-Specific Integrated Circuit) teknolojisine göre birçok getirisi vardır. ASIC'ler, genel olarak istenilen bir devreyi elde etmek için aylarca sürebilen üretim aşamalarına ve yüksek miktarda maliyetlere sahiptir. Buna karşın FPGA'lar, bir saniyeden daha kısa bir sürede yapılandırılabilirlik ve tasarımda herhangi bir hata oluşması durumunda sürekli olarak yeniden programlanabilir bir yapı sunmaktadır. Ayrıca FPGA teknolojisine, birkaç dolardan başlayıp birkaç bin dolara kadar olan düşük maliyetlerle her yerde sahip olma imkânı bulunmaktadır [27]. FPGA teknolojisi iletişim, mobil telefonlar gibi pek çok yeni uygulamalarda en uygun mikro elektronik teknolojisi olmaya başlamıştır. Bunun nedenleri genel olarak, bu cihazların oldukça yüksek kapasiteli ve düşük maliyetli olmaları ve elektronik tasarım araçları kullanılarak kısa tasarım süreçleri ile hızlı bir şekilde piyasaya sunulmaları olarak gösterilebilir. VLSI, ASIC cihazlarının geliştirilmesindeki yüksek maliyet ve üretim sonrasında tasarım değişikliklerinin uygulanışının yetersiz kalmasından dolayı FPGA bu cihazlara göre büyük fayda sağlamaktadır [28]. FPGA'ların sahip olduğu paralel, hızlı işlem yapabilme yeteneği ve sahada birçok kez yeniden programlanabilir olması gibi özellikleri bu cihazların endüstride ve akademik araştırmalarda geniş bir şekilde kullanılmasına olanak tanımıştır. FPGA'lar, güç kalitesi ile ilgili çalışmalarda [29,30], gerçek zamanlı görüntü işlemede [31], yapay sinir ağlarının donanımsal olarak gerçekleştirilmesinde [32, 33], Ayrık Dalgacık Dönüşümü (DWT- Discrete Wavelet

Transform) ve Ters Ayrık Dalgacık Dönüşümü (IDWT-Inverse Discrete Wavelet Transform) gibi sinyal işleme algoritmalarının uygulanmasında etkili bir şekilde kullanım alanı bulmuştur [34, 35].

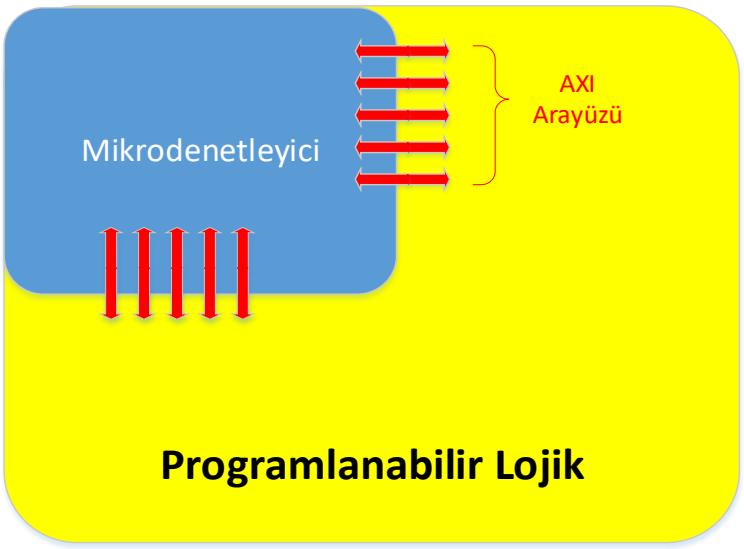
2.1.2. FPGA ile Yapılan Çalışmalar

[36]'da yapılan çalışmada FPGA'da tasarlanan algoritma sayesinde robot kontrolü gerçekleştirilmiştir. FPGA'nın hızlı paralel işlem yeteneğinden faydalanılarak gerçekleştirilen 3 boyutlu bir görüntüleme sistemi başka bir çalışmadır [37]. [38]'de Online güç kalitesi izleme sisteminin tasarımında LabVIEW yazılımı ve CompactRIO donanımı kullanılarak FPGA'da gerçekleştirilmiştir. [39]'de yine LabVIEW yazılımı ve CompactRIO donanımı kullanılarak bir asenkron motor için gerçek zamanlı durum izleme uygulaması gerçekleştirilmiştir. FPGA'da yapılan elektrik enerji sistemlerinde güç kalitesi analizi de yapılan çalışmalardan biridir [40]. [41]'te Tıp alanında yapılan bir çalışmada kalp atışlarının analizi ile hastalık teşhis çalışmaları yapılmıştır. Bu gibi çok sayıda çalışma FPGA'ların üstün özelliklerinden faydalanılarak yapılmıştır [42-45].

2.2. Çip Üzerinde Sistem Mimarili FPGA'lar

SoC, tek bir çip içerisinde birden fazla işe uygun mimarilerin uygulanmasıyla oluşturulmuş yapıya verilen addır. SoC terimi genellikle ASIC uygulamalarında kullanılmaktadır. ADC-DAC çeviricilerin tek bir çip üzerinde tasarlanmasıyla oluşturulmuş devreler buna örnek olarak verilebilir.

FPGA'larda SoC terimi, tek bir çip üzerinde hem bir işlemci mimarisini hem de programlanabilir lojik blokları barındırması anlamında kullanılmaktadır. Şekil 2.3'te bu mimarinin temsili yapısı görülmektedir.

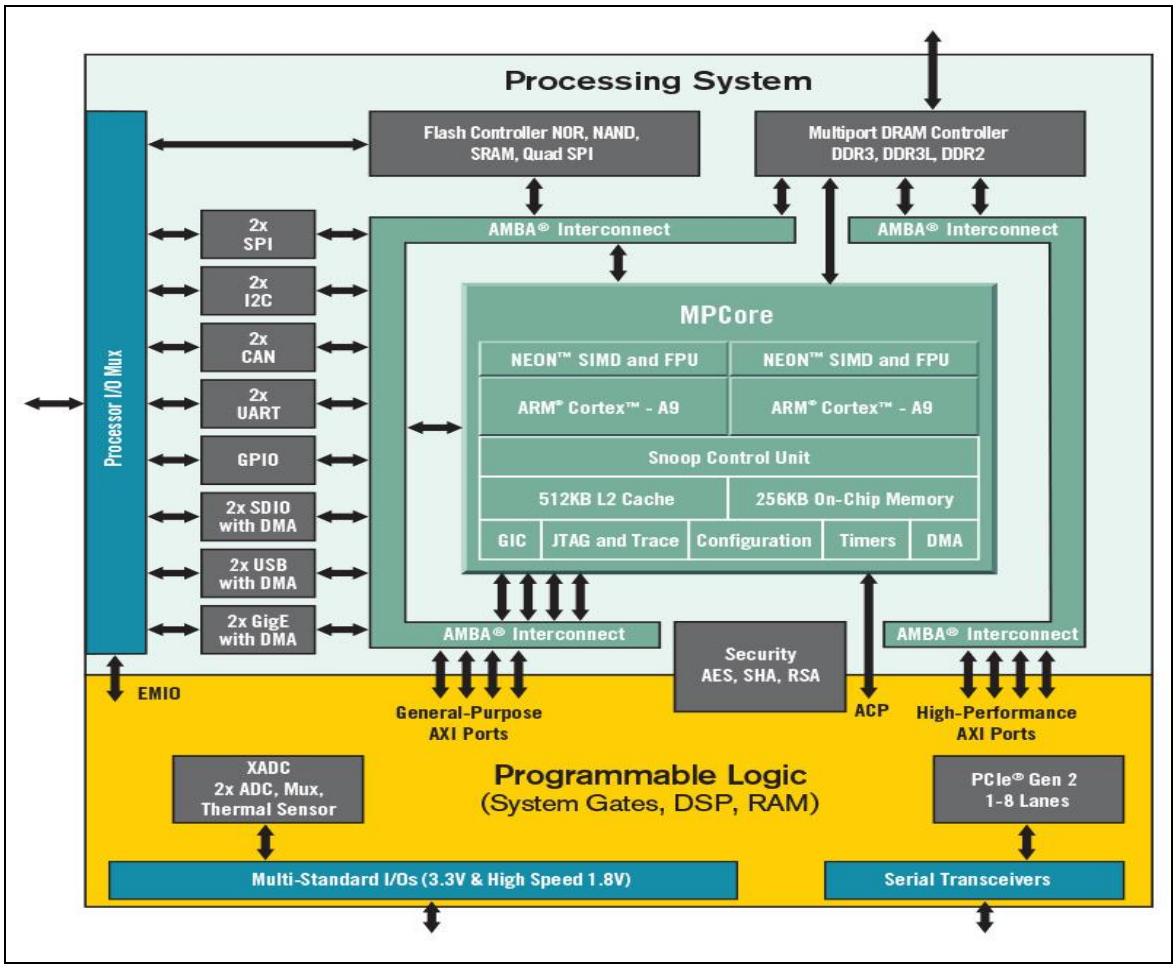


Şekil 2.3. SoC mimarisi [46]

2.2.1. Zynq-7000 SoC

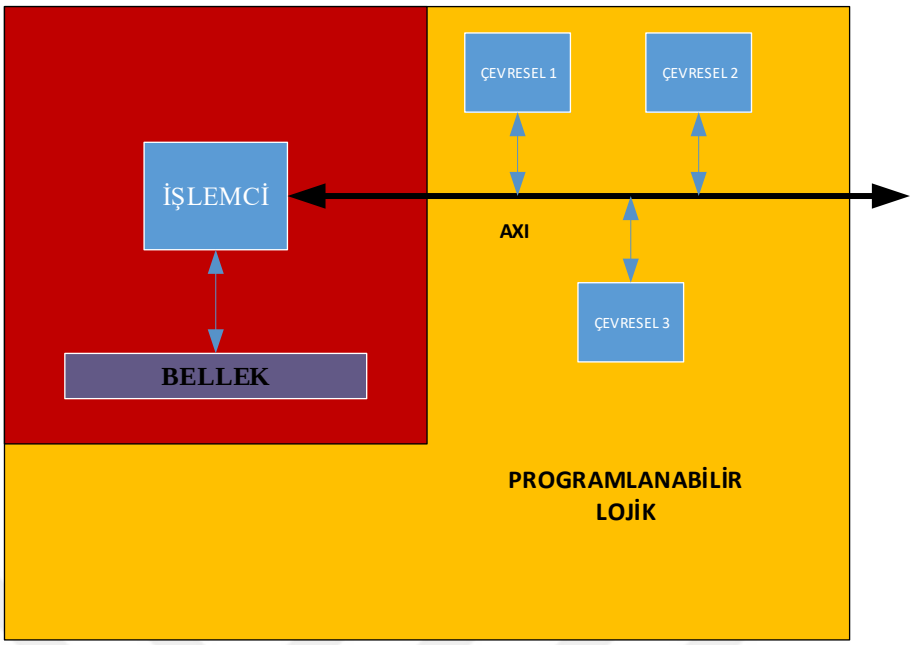
Zynq, Xilinx tarafından geliştirilen FPGA blokları ve ARM Cortex A-9 işlemcisinin aynı programlanabilir çip üzerinde bulunduğu cihazlardır. Bu cihazlara Zynq (çinko) ismi, tıpkı çinko gibi birçok alanda kullanılmaya uygun olduğu için verilmiştir. Esnek olması ve çok çeşitli platformlar için uygun olması ile Zynq cihazları, birçok metalle alaşım yapmaya uygun çinko metaline benzetilmektedir [47].

Zynq'in en önemli ve belirgin özelliği, çift çekirdek ARM Cortex A-9 işlemcisinin, FPGA mantıksal işlem bokları ile birleştirmiş olmasıdır. Zynq, Linux gibi bir işletim sistemini çalıştırmaya uygun ARM A-9 işlemciyi ve Xilinx-7 serisi FPGA bloklarını barındırmaktadır [48]. Ayrıca Xilinx-7 serisi mimari, Dijital Sinyal İşlemcisi (DSP- Digital Signal Processing) bloklarına da sahiptir. Bu mimari, işlemci ve programlanabilir bloklar arasında yüksek bant genişliği ve düşük gecikme süresi sağlayan endüstri standardı, AXI arayüzü ile tamamlanmaktadır [49]. Bu şekilde iki farklı platformun birleştirilmesi, her iki platformun farklı uygulamalar için en ideal şekilde kullanılmasını mümkün kılmaktadır. Şekil 2.4'te bu mimarinin yapısal gösterimi yer almaktadır.



Şekil 2.4. Zynq-7000 mimarisi [50]

Zynq platformunda işlemci tarafından kullanılabilen bazı çevresel donanımlar doğrudan FPGA bloklarına bağlıdır. Bu yapı nedeniyle Zynq çipinde bulunan ARM işlemci bu çevresellere doğrudan erişemez. Çevresellere erişim ancak programlanabilir bloklarla tasarlanacak donanımlarda AXI arayüzü kullanılarak sağlanabilir. Hâlihazırda Xilinx tarafından sağlanan hazır IP bloklar bu iletişim tipini destekler yapıda tasarlanmıştır. Kullanıcı tarafından tasarlanacak IP bloklara da bu arayüz eklenerek çevresellere ARM işlemci erişimi sağlanabilir. Bu yapı Şekil 2.5'te gösterilmiştir.



Şekil 2.5. Zynq-7000 temel yapısı [46]

2.3. Gelişmiş Genişletilebilir Arayüz

Gelişmiş genişletilebilir arayüz, 1996 yılında piyasaya sürülen mikrodenetleyici veri yollarının bir ailesi olan ARM Gelişmiş Mikrodenetleyici Veriyolu Mimarisi (AMBA-Advanced Microcontroller Bus Architecture)'nin bir parçasıdır. ARM AMBA protokolleri, SoC içindeki fonksiyonel blokların bağlantısı ve yönetimi için açık standart, çip arası bağlantı ara yüzüdür. Çok sayıda kontrolör ve çevre birimi ile çoklu işlemci tasarımlarının geliştirilmesini kolaylaştırır [51].

AXI'nin ilk sürümü 2003'te piyasaya sürülen AMBA 3.0'dır. 2010'da piyasaya sunulan AMBA 4.0 AXI4; AXI'nin ikinci büyük sürümünü içermektedir. AXI4, AXI4-Lite, AXI4-Stream olmak üzere üç tip AXI4 arabirimi vardır.

AXI4, bellek eşlemeli(memory mapped) ara yüzler içindir ve tek bir adreslemeyle 256 bit veri aktarım döngüsüne kadar yüksek veri aktarımına izin verir.

AXI4-Lite; basit, tek bir veri alışverişli bellek eşlemeli bir arabirimdir hem tasarımda hem de kullanımda çalışmak için basit bir ara yüzdür.

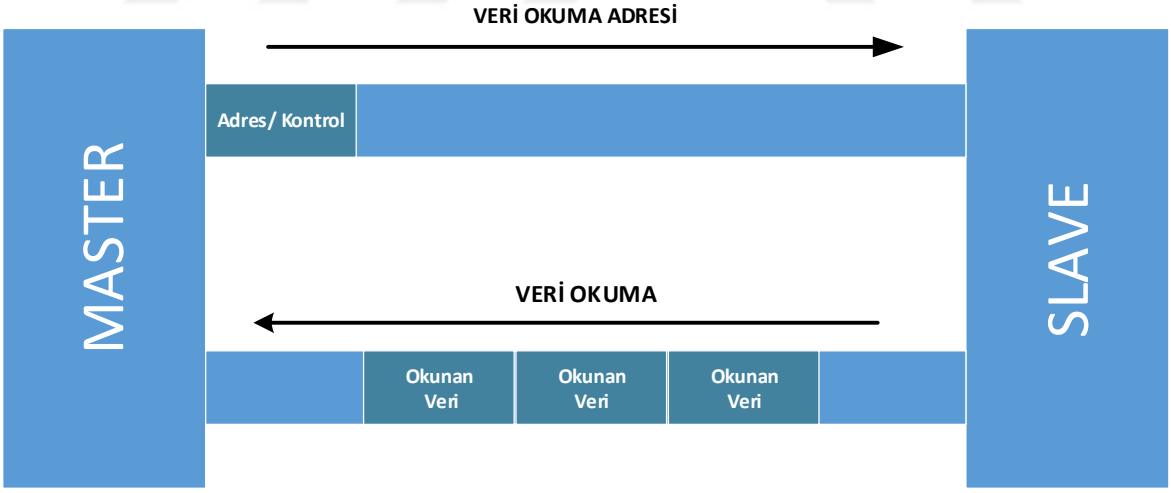
AXI4-Stream, bir adres aşaması gereksinimini tamamen ortadan kaldırır ve sınırsız veri çoğaltma boyutuna izin verir. AXI4-Stream arayüzleri ve transferleri adres fazlarına sahip değildir ve bu nedenle hafıza haritası olarak kabul edilmez.

Xilinx, AXI arayüzünde standartlaşarak, geliştiricilerin sadece IP (Intelectual Property) için tek bir protokol öğrenmelerini yeterli kılmıştır. Birçok IP sağlayıcıları AXI protokolünü desteklemektedir.

AXI, birbiriyle bilgi alışverişi yapan IP çekirdeklerini temsil eden, tek bir AXI master ve AXI slave arasındaki bir arabirimi tanımlar. Birden fazla bellek eşlemeli AXI master ve slave, AXI altyapısına sahip IP blokları kullanılarak birbirine bağlanabilir. AXI4-Lite arayüzleri beş farklı kanaldan oluşur:

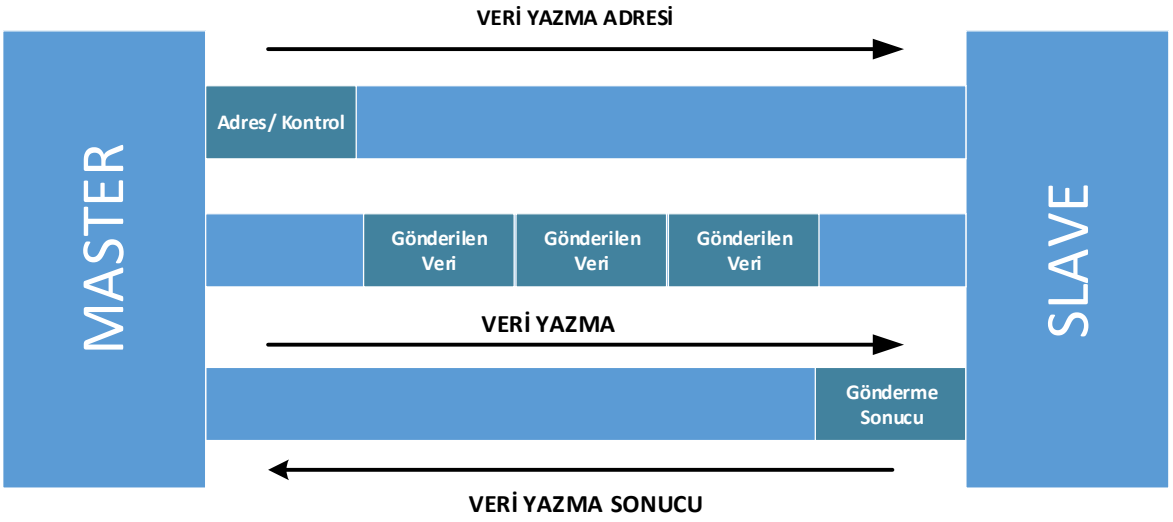
- Okuma Adresi Kanalı,
- Yazma Adresi Kanalı,
- Okuma Veri Kanalı,
- Yazma Veri Kanalı,
- Yazma Yanıtı Kanalı,

Şekil 2.6'da bir AXI4 okuma işleminin okuma adresini ve okuma veri kanallarını nasıl kullandığını göstermektedir.



Şekil 2.6. AXI4 okuma protokolü [49]

Şekil 2.7'de ise yazma işleminin nasıl olduğu gösterilmiştir.



Şekil 2.7. AXI4 yazma protokolü [49]

2.3.1. AXI4-Stream Protokolü

Akış verilerinin aktarımı için tek bir kanalı tanımlar. AXI4-Stream kanalı, AXI4'ün yazma veri kanalını modellemektedir. AXI4'ün aksine, AXI4-Stream (akış) arayüzleri sınırsız miktarda veri patlamasını destekler.

Bellek eşlemeli protokollerde (AXI3, AXI4 ve AXI4-Lite), tüm işlemler bir sistem bellek alanı ve verilerin bir hedef adresin içine aktarılması kavramını içerir. Bellek eşlemeli sistemler genellikle, sistemi tanımlamak için daha homojen bir yol sağlar. Bunun nedeni IP'nin tanımlı bir bellek haritasının etrafında çalışmasıdır.

AXI4-Stream protokolü, tipik olarak bir adres kavramının mevcut olmadığı veya gerekli olmadığı bir veri merkezli uygulamalar için kullanılır. Her bir AXI4-Stream, veri akışı ile tek yönlü bir kanaldan sağlanır. Bu düşük işlem seviyesinde (bellek eşlemeli protokol türlerine kıyasla), IP arasında etkili bir veri taşıma mekanizması tanımlanmıştır. Ancak, IP arasında birleştirici bir adres bağlamı yoktur. AXI4-Stream IP, veri akışı uygulamalarındaki performans için daha iyi optimize edilebilir, ancak belirli bir uygulama alanı etrafında daha uzmanlaşmış olma eğilimindedir.

2.3.2. Doğrudan Bellek Erişimi

Çoğunlukla bir Doğrudan Bellek Erişimi (DMA- Direct Memory Access) motoru, verileri belleğe ve dışına taşımak için kullanılabilir.

AXI DMA motoru, sistem belleği ve AXI4-Stream tipi hedef çevre birimleri arasında yüksek performanslı doğrudan bellek erişimi sağlar. AXI DMA, CPU'nun aktarım kontrolünü ve donanım otomasyonu yürütmesini devre dışı bırakmasına izin verir.

2.3.3. AXI VDMA

AXI Video DMA (AXI VDMA) motoru, sistem belleği ve AXI4-Stream tipi hedef çevre birimleri arasında yüksek performanslı doğrudan bellek erişimi sağlar. AXI VDMA ayrıca, Merkezi İşlem Birimi (CPU)'nun aktarım kontrolünü ve donanım otomasyonuna aktarımını durdurmasına izin veren Scatter Gather (SG) özelliğini de desteklemektedir. AXI VDMA ve SG motorları, AXI4-Stream ve AXI4 bellek eşlemeli veri yolları arasındaki temel köprüleme elemanı olan AXI DataMover yardımcı çekirdeği etrafında inşa edilmiştir. AXI VDMA aşağıdakileri özelliklere sahiptir:

- En fazla 32 çerçeve arabelleği için dairesel çerçeve tamponu erişimi ve tam video karelerinin bölümlerinin aktarımını sağlar.
- Bir kare üzerinde bekleme, aynı video karesi verilerinin tekrar tekrar aktarılmasına izin verir.
- Bağımsız kare eş zamanlaması ve bağımsız bir AXI saati, her bir kanalın farklı bir kare hızında ve farklı bir piksel hızında çalışmasını sağlar. İki bağımsız çalışan AXI VDMA kanalı arasında eş zamanlaması sağlamak için isteğe bağlı bir Gen-Lock eş zamanlaması özelliği vardır. Gen-Lock, AXI VDMA slave'lerini otomatik olarak bir veya daha fazla AXI VDMA master'ına eş zamanlama yöntemini sağlar, böylece slave master ile aynı video karesinde çalışmak zorunda kalmaz.

2.3.4. Zynq Platformunun Getirileri

Bir uygulama geliştirirken çok çeşitli platformlardan faydalanmak mümkündür. Genel amaçlı işlemciler, DSP'ler ya da FPGA'lar kullanılarak çok çeşitli uygulamalar geliştirilebilir.

Genel amaçlı bir işlemci veya DSP kullanıldığında hedeflenen uygulama, işlemcinin izin verdiği sınırlar çerçevesinde ve yazılımsal olarak gerçekleşecektir. Ayrıca işlem akışı seri bir şekilde işleyecek, kapsamlı ve yoğun uygulamalarda işlem süresi önemli ölçüde artacaktır.

Aynı işlemler bir Zynq platformu kullanılarak yapıldığında ise karmaşık işlem gerektiren uygulama alt hesaplama rutinleri için paralel hesaplama yeteneğine sahip FPGA blokları ve DSP blokları kullanılabilecek, nispeten hız gerektirmeyen işlemler ARM Cortex A-9 işlemcisi üzerinden gerçekleştirilecektir. FPGA, belirli bir görev için tam olarak optimize edilebilmekte ve hedeflenen işlem için en uygun mimari yapı oluşturulmaktadır.

Programlanabilir mantık doğası gereği, paralel olarak uygulamaların uygulanabilmesi için ideal bir ortam sunar. Örneğin matematiksel işlemlerin aynı anda çok sayıda örnek veya piksel üzerinde gerçekleştirildiği sinyal ve görüntü işleme uygulamaları gibi. Bu uygulamalarda işlemleri, seri olarak birim işlem zamanında bir işlem yapılarak uzun işlem süresinde hesaplamak yerine aynı anda farklı pikselleri işleyerek, işleme zamanının düşürülmesi sağlanır.

2.3.5. Zynq Platformu Uygulama Alanları

Zynq platformu, yapısı ve getirilerinden dolayı birçok uygulama alanına sahiptir. Bunlar aşağıdaki gibi verilebilir:

- Otomotiv
- İletişim
- Savunma ve havacılık
- Robotik ve kontrol
- Görüntü işleme
- Tıp
- Yüksek performanslı hesaplama

2.3.5.1. Otomotiv

Günümüzde otomobiller motor yönetiminden, pencere, ayna ve aydınlatma gibi fonksiyonların kontrolüne, seyrüsefer ve bilgi-eğlence sistemlerine kadar önemli miktarda elektronik içermektedir. Gelişmiş Sürücü Destek Sistemleri (ADAS-Advanced Driver Assistance Systems), özellikle sürücü güvenliği ve rahatlığı için araçlarda sunulan sistemlerin toplanmasıyla ilgilidir. Bunlar; şerit ayrılma uyarı sistemleri, yol işareti tanıma,

park yardımı, gösterge görüntüleri ve sürücü farkındalığını izleyen sistemlerdir. Zynq cihazları bu otomotiv sistemlerini gerçekleştirmek için kullanılabilir [52], [53]. Zynq'in işleme kabiliyetleri, bu tür sistemleri özellikle çok uygun hale getirmektedir. Bunun ötesinde bütün işlemlerin tek bir cihazdan kontrol edilmesi fiziksel yer kazancı da sağlayacaktır.

2.3.5.2. İletişim

FPGA'lar, hem kablosuz iletişimde hem de kablolu, paket tabanlı sistemlerde gerekli olan hesaplama yoğun işlemlerin üstlenilmesi için kullanılabilmektedir. Bu alan çeşitlidir ve karasal ve uydu iletimi, mobil destek altyapısı, kablolu ağ teçhizatı, radar, sonar, Küresel Konumlandırma Sistemi (GPS-Global Positioning System) ve diğer birçok iletişim sistemleri için alıcı-vericilere sahiptir.

Kablosuz iletişimde, kablosuz sistem ve standartların sayısı artmaya devam etmektedir. Esnek radyo kavramı, radyo spektrumunu daha iyi kullanma ve radyo donanımını, çalışmasını dinamik olarak değiştirebilen tek bir cihazda birleştirmeyi sağlar. Zynq, ideal bir esnek radyo platformu sağlar. Kablolu iletişimde, yazılım kontrolü altında işlevselliği yükseltme yeteneğine sahip "yumuşak tanımlanmış ağlar" kullanılarak benzer bir esneklik düzeyi aranır [54].

2.3.5.3. Savunma ve Havacılık

Savunma sistemleri, çok çeşitli iletişim, görüntü işleme, havacılık, seyrüsefer ve ulaşım sistemleri ile ilgili teknolojiyi içerir. Savunma elektroniği genellikle genişletilmiş sıcaklık aralıklarına ve güvenlik özelliklerine sahip sivil uygulamalara kıyasla daha yüksek bir sağlamlık seviyesi gerektirir [55].

İlgi çeken bir alan olan savunma alanı, askeri personel ve donanımların; uçaklar, uydular, sinyal istihbarat cihazları ve diğer savunma sistemleri ile birbirine bağlı olduğu "ağa bağlı savaş alanı" kavramıdır [56]. Ağa bağlı savaş alanının amacı bilgiyi toplamak, filtrelemek ve paylaşmak ve böylece askeri operasyonların etkinliğini optimize etmektir.

Sivil havacılık uygulamaları arasında ise seyrüsefer ve araç üstü uçuş sistemleri, uydu ve yer haberleşmeleri ve radar sistemleri yer almaktadır.

Zynq platformu savunma ve havacılık alanının ihtiyaç duyduğu sağlamlık ve çok çeşitli gereksinimler için kullanılabilecek esneklik ve hıza sahiptir.

2.3.5.4. Robotik, Kontrol ve Endüstri

Üretim ve hizmetlerden, yüksek enerji fiziği deneylerine kadar uzanan endüstriyel ve bilimsel süreçler, hassas kontrol ve enstrümantasyon gerektirir.

Zynq cihazları bu tür işlemler için çok uygun bir platformdur. Programlanabilir bloklardan yararlanarak hızlı, gerçek zamanlı işleme ve çoklu sensör girişlerini ve aktuatör çıkışlarını eşzamanlı olarak kontrol edebilirler. Zynq, bir işletim sistemini de desteklediği için robotik ve kontrol alanlarında esnek şekilde kullanım alanı bulmaktadır.

Motor kontrol algoritmaları, endüstrinin bazı dallarında özellikle önemlidir. Örneğin, ABD imalat sanayine yönelik bir anket, sanayide tüketilen elektrik enerjisinin yaklaşık %50'sinin "makine sürücüleri", yani motorlar, pompalar ve fanlar ile ilişkilendirilebileceğini göstermektedir [57]. Zynq, İşlemci (PS-Processing System) ve Programlanabilir Lojik (PL-Programmable Logic) arasındaki yüksek bant genişliği bağlantısından dolayı motor kontrolüne çok uygundur ve sıkı bir geri bildirim döngüsüne olanak tanımaktadır [58].

2.3.5.5. Görüntü İşleme

Görüntü ve video işleme, tüketici ve profesyonel kullanım için kameralar, video sıkıştırma ve depolama sistemleri, yayın donanımları, görüntüleme teknolojisi, endüstriyel süreç izleme, güvenlik ve gözetleme gibi çeşitli uygulamaları içerir.

Görüntü ve video işleme alanının, çeşitli tüketici ve ticari ürünlerdeki özelliklerin yanı sıra tıp, endüstri, savunma ve güvenlik alanlarında da kullanım alanı bulmaktadır.

Görüntü işleme, tek veya “hareketsiz” resimlerle ilgilenirken, video işleme, genellikle belirli bir kare hızında, geçici bir dizi resim (“kareler”) anlamına gelir. Bilgisayar görmesi (veya makine görmesi) sistemleri, görüntülerden veya video verilerinden anlam çıkarılabilir. Uygun olduğu takdirde, kararlar bu bilgilere dayanılarak alınabilir.

Zynq'in işleme yetenekleri, büyük miktarlarda piksel verilerini hem hesaplanabilir olarak işlenmesini gerektiren, hem de görüntülerden bilgi elde etmek için PL ve PS için uygun olan yazılım algoritmalarını işleme özelliklerine sahiptir [56].

2.3.5.6. Tıp

Tıbbi teşhisteki önemli bir konu, vücudun içini "görme" işlemidir. Bu işlem; Bilgisayarlı Tomografi (CT-Computed Tomography) tarayıcıları, ultrason ve Manyetik Rezonans Görüntüleyicileri (MRI) gibi tıbbi görüntüleme ekipmanı gerektirir. Bu donanımlardan elde edilen görüntülerin geliştirilmesi ve gösterilmesi, genellikle büyük veri setlerinde gerçekleştirilecek karmaşık görüntü işleme algoritmaları gerektirir. Diğer görüntü işleme uygulamalarında olduğu gibi, Zynq tarafından sağlanan PL ve PS özelliklerinin birleşimi hem yüksek hızlı paralel işlemeyi hem de yazılım tabanlı algoritmaları destekler.

Tıpta diğer uygulamalar, robot destekli cerrahide cihazların kontrolünü ve endoskopik ve hasta izleme donanımlarını ve ev sağlık teknolojileri gibi gerçek zamanlı cerrahi görüntülemeyi içermektedir [59].

2.3.5.7. Yüksek Performanslı Hesaplama

Yüksek performanslı hesaplama, büyük veri kümelerinin hızlı bir şekilde işlenmesini gerektiren uygulamaları kapsar. Bunlar, genellikle özel donanım işlemleriyle hızlandırılabilir işlemlerdir [60]. Finansal modelleme [61], petrol ve gaz arama analizi, bilimsel deneylerden veri işleme, radyo astronomi [62] ve yakalanan radar sinyallerinin analizi gibi çeşitli uygulamaları içerir. Ancak bunlarla sınırlı değildir, ayrıca veri merkezleri ve bulut bilişim tesisleri için de geliştirilen altyapıyı da kapsamaktadır.

2.4. Zynq-7000 Programlama Araçları

Diğer programlanabilir cihazlarda olduğu gibi Zynq platformu da uygulama geliştirme aşamasında birçok programlama aracı ve programlama dilini desteklemektedir.

Zynq programlarken diğer FPGA'lardan farklı olarak sadece donanımsal programlama yapılmadığı için programlama mantığı biraz farklıdır. Zynq cihazlarında

bulunan FPGA blokları donanımsal programlama gerektirirken, ARM işlemci programlanırken yazılımsal programlama kullanılır. Ayrıca platformda bulunan DSP hesaplama blokları da programlanarak kullanılabilmektedir. ARM-A9 işlemci sayesinde istenildiği takdirde Linux işletim sistemi ile de kullanılabilmektedir. Bu denli esnek bir platform programlanırken birçok programlama aracı ve programlama dilleri kullanılabilmektedir. Zynq programlarken kullanılabilecek programlama araçları şunlardır:

- Vivado Design Suite,
- Vivado-HLS,
- Xilinx SDK,
- Xilinx SDSoC-SDx,
- Matlab.

2.4.1. Vivado Design Suite IDE

Xilinx tarafından geliştirilen, Xilinx FPGA'lar için donanımsal gömülü sistem geliştirme ortamıdır. Zynq-7000 mimarisinin barındırdığı FPGA bloklarının programlaması, bu mimarinin barındırdığı ARM işlemcinin genel ayarları ve FPGA blokları ile işlemci bağlantısı bu ortamda yapılır. Vivado Design Suite, verimliliği artırmak için tasarlanmıştır. Xilinx FPGA'larının artan özellikleri nedeniyle geliştirilen Vivado, eski Xilinx ISE Design Suite'in yerine daha kapsamlı bir kullanıcı arayüzü sunmaktadır. Bu kapsamlı yapı sayesinde çok boyutlu tasarım zorluklarının aşılması amaçlanmıştır. Tüm Vivado Design Suite araçları, yerel bir takım komut dili (Tcl) arayüzü ile yazılmıştır. Vivado Design Suite için grafik kullanıcı arabirimi (GUI) olan Vivado Entegre Tasarım Ortamında (IDE-Integrated Development Environment) bulunan tüm komut ve seçeneklere Tcl ile erişilebilir. Ortam; şematik programlama, Verilog programlama ve VHDL programlama dillerini desteklemektedir. Şematik programlama ile özellikle görüntü işleme uygulamaları olmak üzere birçok hazır FPGA bloklarının kullanılmasına olanak tanımaktadır. Yapılan çalışmaların simülasyonunun da yapılabildiği bir simülatöre sahiptir.

2.4.2. Xilinx SDK

Xilinx Yazılım Geliştirme Ortamı (XSDK- Xilinx Software Development Kit), Xilinx mikroişlemcilerinde gömülü uygulamalar oluşturmak için Entegre Tasarım Ortamıdır. Bu mikroişlemciler, Zynq- UltraScale + MPSoC, Zynq-7000 SoCs ve MicroBlaze işlemcilerdir. SDK, homojen ve heterojen çok işlemcili tasarım, hata ayıklama ve performans analizi sağlayan uygulama ortamıdır. Eclipse tabablı olarak geliştirilmiştir. Vivado gömülü donanım tasarım ortamına doğrudan arabirim oluşturan ve donanım / yazılım birlikte hata ayıklama yetenekleri dahil olmak üzere tam yazılım tasarımı ve hata ayıklama akışları, editör, derleyiciler, derleme araçları, flash bellek yönetimi ve JTAG hata ayıklama tümleştirmesi, Xilinx Yazılım Komut Satırı Aracı (XSCT- Xilinx Software Command-line Tool) barındırmaktadır.

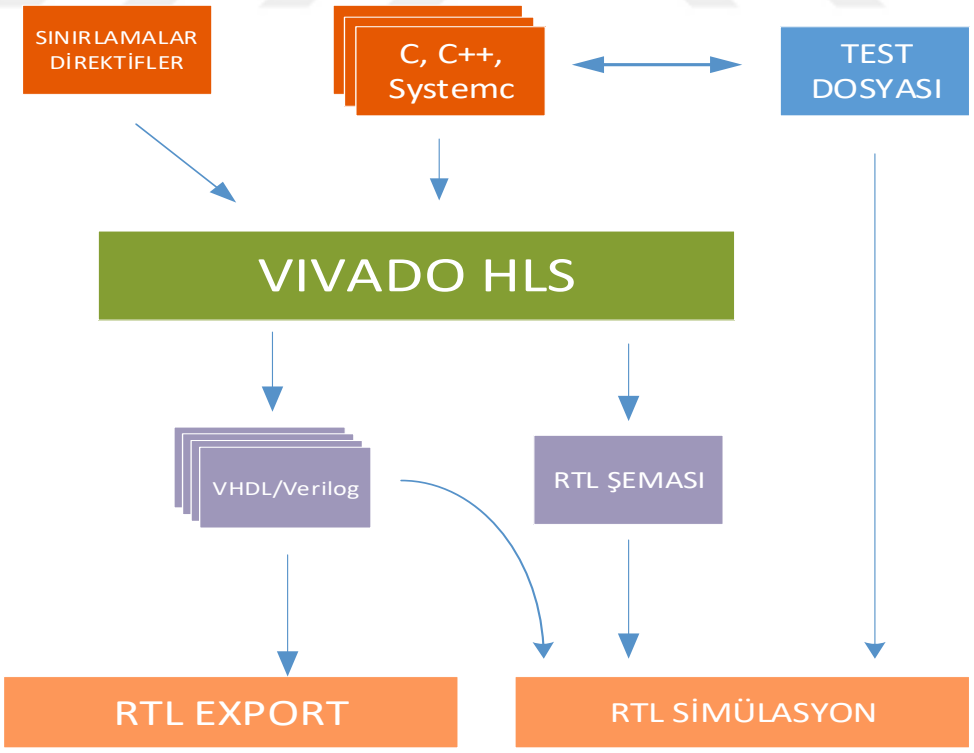
Xilinx System Debugger, Zynq UltraScale + MPSoC, Zynq-7000 SoC ve MicroBlaze çekirdeklerini destekleyen entegre bir hata ayıklayıcıdır. Hem komut satırından hem de XSCT'yi kullanarak ve SDK GUI içinden hata ayıklama kullanılabilir. Kesme noktaları veya izleme noktaları ayarlama, program yürütme boyunca adım atma, program değişkenlerini ve yığını görüntüleme ve bellek içeriğini görüntüleme gibi tüm yaygın hata ayıklama özelliklerini destekler. Aynı anda, aynı hata ayıklama ortamından farklı işlemcilerde (çok işlemcili bir sistemde) çalışan programların hatalarını ayıklayabilir. Örneğin, bir Zynq veya Zynq UltraScale+ tabanlı tasarımında, Sistem Hata Ayıklayıcı tek bir JTAG kablosuyla aynı hata ayıklama oturumunda hem ARM CPU'larını hem de birden çok MicroBlaze yazılımını görüntüleyebilir.

XSDK, proje oluşturmada üç işletim sistemi türünü desteklemektedir. Bunlar, bağımsız işletim sistemi, gerçek zamanlı işletim sistemi ve Linux işletim sistemidir.

XSDK; desteklenen tüm Xilinx donanım IP'leri, POSIX uyumlu çekirdek kütüphanesi, ağ oluşturma ve dosya işleme kütüphanesi için kullanıcı tarafından özelleştirilebilir sürücüleri içerir. Bu kütüphaneler ve sürücüler, özellik gereksinimlerine, bellek gereksinimlerine ve donanım özelliklerine göre özel tasarım için ölçeklendirilebilir yapıdadır.

2.4.3. Vivado Yüksek Seviyeli Sentez

Günümüzde haberleşme, medikal, savunma ve tüketici uygulamalarında kullanılan gelişmiş algoritmalar her zamankinden daha karmaşıktır. Bu karmaşıklıktan dolayı VHDL dili kullanarak donanım oluşturma işlemi hem zaman almakta hem de yazılan kodun kontrol edilmesini zorlaştırmaktadır. Vivado-HLS(Yüksek Seviyeli Sentez-High Level Syntesis) yüksek seviyeli diller kullanarak işlem donanımları (IP Core- İntellectuel Property) üretmeye yarayan programlama aracıdır. Yüksek seviyeli dil kullanıldığı için programlama süresi kısalmaktadır. Vivado-HLS'nin kullandığı yüksek seviyeli diller C, C ++ ve SistemC dilleridir. Bu dillerle yazılan kodlar daha sonra VHDL veya verilog dilinde yazılmış kodlara dönüştürülür. Algoritmik tanımlama, veri tipi belirtimi (tamsayı, sabit nokta veya kayan nokta) ve arabirimler (FIFO, AXI4, AXI4-Lite, AXI4-Stream), video ve DSP uygulamaları için geniş kütüphane desteği sağlama gibi geniş özelliklere sahiptir. C/C++ simülasyon ortamı desteği sunmaktadır. Ayrıca VHDL ve Verilog dillerine dönüşüm yapıldıktan sonra bu dillerde lojik simülasyon ortamına sahiptir. Şekil 2.8'de Vivado HLS aracı IP çekirdek oluşturma aşamaları gösterilmiştir.



Şekil 2.8. Vivado HLS aracı IP çekirdek oluşturma aşamaları [63]

2.4.4. Xilinx SDSoC-SDx

Yazılım Tanımlı SoC (SDSoC-Software Defined SoC) ortamı, Zynq-7000 Tüm Programlanabilir SoC'ler ve Zynq UltraScale+ ve MPSoC'leri kullanarak heterojen gömülü sistemlerin uygulanması için bir Eclipse tabanlı tümleşik geliştirme ortamıdır. SDSoC sistem derleyicisi, C / C ++ ile yazılmış uygulama kodunu derleyerek, uygulamaya özel bir yazılım ve donanım kodu üretir. Bir SDSoC platformu, işletim sistemi, harici bellek arabirimleri, özel giriş /çıkış birimleri, önyükleme yükleyicileri, platform çevre birimleri için sürücüler de dahil olmak üzere temel donanım ve yazılım mimarisi ve uygulama bağlamını tanımlar. SDSoC ortamında oluşturulan her proje belirli bir platformu hedefler. Bu sayede, farklı platformlar için yüksek düzeyde uygulamaya özel sistemler oluşturabilir ve yeniden kullanılabilir.

SDSoC sistem derleyicileri, yazılım ve donanım fonksiyonları arasındaki veri akışını belirlemek için programı analiz eder ve programı gerçekleştirmek için uygulamaya özel bir sistem oluşturur. Yüksek performans elde etmek için, her donanım işlevi bağımsız bir iş parçacığı olarak çalışır. Sistem derleyicileri, donanım ve yazılım iş parçacıkları arasında senkronizasyon sağlayan donanım ve yazılım bileşenleri üretir. Uygulama kodu, birçok donanım işlevini, belirli bir donanım işlevinin birden çok örneğini içerebilir. SDSoC IDE, hızlı bir performans tahmini yeteneği sağlar. SDSoC sistem derleyicileri, bir platformu hedefler ve sentezlenebilir C / C ++ fonksiyonlarını programlanabilir mantık içine derlemek için Vivado High-Level Synthesis (HLS) aracını kullanır. Daha sonra, Vivado Design Suite araçlarını çalıştırarak DMA'lar, ara bağlantılar, donanım arabellekleri ve diğer IP'ler dahil olmak üzere eksiksiz bir donanım sistemi oluştururlar. Tüm donanım işlev çağrılarının orijinal davranışlarını koruduğundan emin olmak için SDSoC sistem derleyicileri sisteme özgü yazılım taslakları ve yapılandırma verileri üretir. Program, oluşturulan IP bloklarını kullanmak için gerekli sürücü çağrıları içerir. Uygulama ve oluşturulan yazılım standart bir GNU toolchain kullanılarak derlenir ve bağlanır. Tek kaynaktan, eksiksiz uygulamalar üretilmesine, program düzeyinde yeniden düzenleme yaparak tasarım ve mimari değişiklikleri üzerinde yineleme yapılmasına olanak tanır ve hedef platformda uygulamalar gerçekleştirmek için gereken süreyi önemli ölçüde azaltır.

2.4.5. Matlab

Matlab, teknik hesaplamalar ve matematiksel problemlerin çözümü ve analizi için tasarlanmış bir yazılım geliştirme aracıdır. Matlab, Xilinx FPGA'larını programlamak için kullanılabilecek araçları barındırmaktadır. Bu araçlar; genellikle DSP tasarımı için kullanılan Sistem Üreticisi ve Matlab diliyle yazılan uygulamaları VHDL veya Verilog diline çeviren HDL Kodlayıcıdır.

Sistem Üreticisi, FPGA sistem tasarımı için Mathworks Simulink tasarım platformunu kullanan bir tasarım aracıdır. Donanım tasarımlarının oluşturulması için üst düzey, model tabanlı bir ortam sağlar. System Generator tasarımını Vivado IP Kataloguna dahil edilebilecek bir IP birimine paketlemesini sağlar. Sistem Üreticisi tasarımı, daha sonra, IP Kataloğu'ndan başka bir birim gibi kullanılabilir ve bir Vivado kullanıcı tasarımına eklenebilir [64]. Sistem Üreticisi, tasarımların hızlı ve kolay bir şekilde oluşturulması için blokların birbirine bağlanabileceği IP tasarımı için sezgisel bir ortam sunar. Basit matematiksel operatörlerden karmaşık DSP işlemlerine kadar çok sayıda blok mevcuttur. İşlevsellik her zaman HDL kodundaki en okunabilir formda uygulanmaz ve bu, bazı tasarımların takip edilmesini zorlaştırabilir. Sistem Üreteci tasarımında seçilen seçeneklere bağlı olarak, IP uygulaması HDL'de elle kodlanmış gibi etkili olmayabilir.

HDL Kodlayıcı, Matlab fonksiyonlarından ve Simulink modellerinden sentezlenebilir HDL kodunun (hem VHDL hem de Verilog) üretilmesini sağlayan bir araçtır. Kullanıcının HDL tasarımının düşük seviyenin incelikleri olmadan algoritmaların ve modellerin geliştirilmesine konsantre olmasını sağlar. HDL Kodlayıcı iş akışı, HDL kodunun doğrulanması için araçlar sağlar ve üretilen HDL kodunun orijinal Matlab / Simulink modeli ile birlikte test edilmesini sağlar. Bu sayede, oluşturulan HDL kodunda bulunabilecek herhangi bir hata kolayca tespit edilebilir. HDL kod en iyilenmesi, kritik yolların vurgulanmasına, HDL mimarisinin kontrol edilmesine ve donanım kaynağı kullanım tahmini oluşturulmasına olanak tanıyan, uygulama üzerinde büyük miktarda kontrol sağlamak için hedef FPGA aygıtını belirleme seçeneğini sunar. HDL Kodlayıcı tarafından oluşturulduktan sonra, HDL kodu, bir IP çekirdeği oluşturmak için kullanılabilir. HDL Kodlayıcı, mevcut Matlab / Simulink modellerinden hızlı bir şekilde IP oluşturulmasına ve minimum tasarım değişiklikleri yapılmasına izin verirken, bazı dezavantajları vardır. Şu anda tüm Matlab fonksiyonları ve Simulink blokları HDL üretimini desteklememektedir. Bu, bazı işlevlerin desteklenen işlevler veya bloklarla

yeniden işlenmesi gerekebileceği anlamına gelir. Dikkat edilmesi gereken bir diğer nokta da, donanım uygulaması ve en iyilenmesi açısından büyük miktarda özelleştirme yapılmasına rağmen, üretilen HDL kodunun elle kodlanmış HDL gibi etkili olamayacağıdır.

3. ZYNQ ÜZERİNDE İŞLETİM SİSTEMİ

3.1. İşletim Sistemi

İşletim Sistemleri, uygulama yazılımları ile bilgisayar donanımı arasındaki bağlantıyı sağlayan özel yazılımlardır. Bilgisayarda programların çalıştırılmasını ve istenilen temel işlevlerin yerine getirilmesini işletim sistemi sağlar. Dosya ve dizin oluşturma, silme, kopyalama, taşıma işlemleri, disk biçimlendirilmesi (format), tarih, saat, donanım ayarları, vb. işlemler işletim sistemi tarafından gerçekleştirilir.

3.2. Zynq Üzerinde İşletim Sistemi

Gömülü bir sistem geliştirirken bir tasarımcının karşılaştığı temel sorulardan biri, gömülü bir işletim sistemi dahil edip etmemektir. Zynq platformunda bulunan çift çekirdekli işlemciyi en iyi şekilde kullanabilmek için mevcut seçeneklerden biri de işletim sistemi kullanmaktır.

Gömülü bir işletim sistemi tüm gömülü sistem uygulamaları için gerekli olmasa da, kullanımlarının birçok avantajı vardır. Bunlardan ilki pazara arz edilme zamanın azaltılmasıdır. Gömülü bir işletim sisteminin geliştirme süresini azaltabileceği bir dizi kilit alan vardır. İşletim sistemi satıcıları çok çeşitli mimariler ve platformlar için destek sağlar. Geliştirilmiş olan yazılım belirli bir aygıttan ziyade bir işletim sisteminde hedeflendiğinden, yeni bir mimariye veya aygıta geçme süreci sorun oluşturmamalıdır. Örneğin, gömülü Linux ve geleneksel Masaüstü sürümü arasında büyük ölçüde benzerlik vardır. Bir tasarımcı, Linux'un çeşitli masaüstü sürümleri için uygulamalar geliştirmeye aşinaysa, gömülü Linux için geliştirilme işlemi kolay olacaktır ve öğrenmesi nispeten kolay olacaktır. Bu, bir tasarımcının yeni bir geliştirme ortamına aşina olması için gereken süreyi önemli ölçüde azaltacaktır. Geliştirilmiş taşınabilirlik, İşletim Sistemi Arabirimi (POSIX- Portable Operating System Interface for Unix) gibi standartlaştırılmış bir sistem arabirimine sahip bir işletim sisteminin kullanılmasıyla sağlanabilir. Genel olarak, POSIX standardına sahip işletim sistemlerinin kullanımı, işletim sistemi seviyesinde taşınabilirlik konusunda daha fazla esneklik sağlar.

İkinci avantaj ise; bakım ve geliştirme maliyetlerinin azaltılmasıdır. Gömülü bir işletim sistemi kullanarak, geliştirilmesi ve test edilmesi gereken özel kod miktarı azaltılmaktadır.

3.2.1. İşletim Sistemi Türleri

Gömülü sistemde kullanılacak işletim sisteminin türünü belirlerken bir dizi olasılık vardır. Bu tür örnekler, basit bir bağımsız işletim sistemi (Standalone Operating System), bir Gerçek Zamanlı İşletim Sistemi (RTOS) veya gömülü Linux'un sayısız varyasyonu gibi özelleştirilmiş yerleşik bir işletim sistemini içerir.

3.2.1.1. Bağımsız İşletim Sistemi

Bağımsız bir işletim sistemi (Standalone OS), sistemin işlemciye özgü işlevlere erişmek için kullanabileceği çok düşük düzeyde yazılım birimleri sağlamayı amaçlayan basit bir işletim sistemidir. Özellikle Zynq platformu ile ilgili olarak Xilinx, önbelleklerin yapılandırılması, kesintilerin, istisnaların ve diğer donanımla ilgili işlevlerin ayarlanması gibi işlevler sağlayan bağımsız bir işletim sistemi platformu sağlar. Bağımsız platform, doğrudan İşletim Sistemi katmanının altında bulunur ve bir uygulama doğrudan işlemci özelliklerine erişmek istediğinde kullanılır [65]. Bağımsız bir işletim sistemi, kod çalıştırma üzerinde hızlı kontrol sağlar, ancak işlevsellik açısından oldukça sınırlıdır. Sadece yazılım işlevlerinin basit ve tekrarlayıcı olduğu uygulamalar için kullanılması daha uygundur. Bağımsız bir işletim sistemi tarafından yürütülen görevlerin sayısı, daha fazla görev eklemek, bağımsız olarak gereken görev yönetimini hızla artırabileceğinden, nispeten küçük olmalıdır.

3.2.1.2. Gerçek Zamanlı İşletim Sistemleri

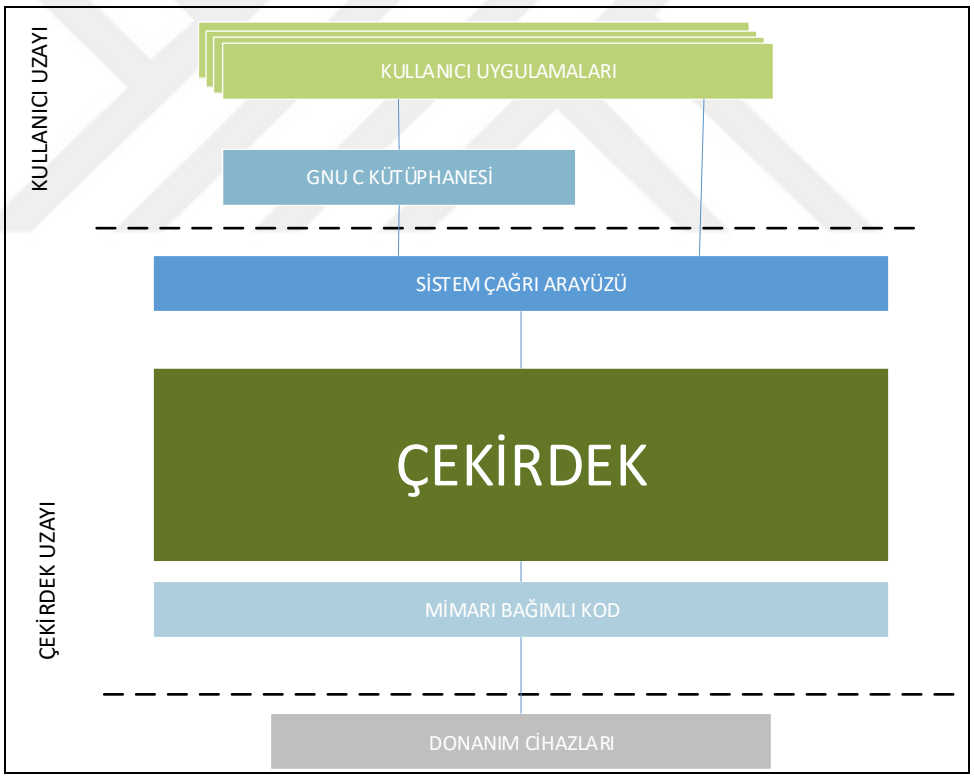
Bir Gerçek Zamanlı İşletim Sistemi (RTOS)'un belirleyici özelliği programlayıcı tarafından garanti edilen çalışma zamanıdır. Bir RTOS'un amacı, yüksek bir iş hacmine ulaşmak değildir, bunun yerine, belirli bir görev için hem hızlı hem de öngörülebilir bir şekilde yanıt vermektir. Birçok gömülü sistemin işlevi, yazılımın olaylara kısa, tanımlanmış bir yanıt süresi içinde yanıt vermesini gerektirir. Çoğu modern RTOS sistemi, gerçek zamanlı çekirdeği tamamlayan bir dizi yüksek seviyeli işlev içerir. Bu işlevler bir Grafiksel Kullanıcı Arayüzü (GUI-Graphical User Interface), iletişim protokolü yığınları ve belirli bir derecede çevresel aygıt yönetimini içerebilir. Gömülü bir sistemde, RTOS cihazı kontrol etmekten ve gerekli tepki seviyesini sağlamaktan sorumludur. Yazılım

görevleri, her bir göreve ayrılmış CPU zamanını programlayan RTOS tarafından kontrol edilir [65].

3.2.1.3. Linux

Linux, Linus Torvalds tarafından 1991 yılında geliştirilmiş işletim sistemi çekirdeğidir. Torvalds, Richard Stallman tarafından sağlanan GNU araçlarını kullanarak, Linux çekirdeğini oluşturmuştur. Bugün, Linux her zamankinden daha popüler olmuştur ve cep telefonları ve uydu navigasyon sistemlerinden süper bilgisayar ve sunuculara kadar geniş bir cihaz yelpazesinde bulunmaktadır.

Şekil 3.1'de GNU/Linux sisteminin genelleştirilmiş üst düzey mimarisi gösterilmiştir.



Şekil 3.1. GNU/Linux mimarisi [66]

Çekirdek, çok daha fazla bileşen içeren çok daha karmaşık bir kavramdır. GNU C kütüphanesinin yanı sıra uygulamalar ve sistem programları çekirdeğin üstünde çalışır. Sistem programları, sistemin çalışmasını sağlayan çeşitli işletim sistemi hizmetlerinin

uygulanmasında gereklidir. Fiziksel donanım, çekirdeğin altında bulunur ve kullanıcı alanı, çekirdek tarafından soyutlanır. Bu fiziksel donamımlar, herhangi bir sistem depolamasını, ağ kartlarını veya bir geliştirme kartındaki GPIO (General Purpose Input Output- Genel Amaçlı Giriş Çıkış)'yu içerir.

Sistem Çağrı Arayüzü (SCI- System Call Interface), kullanıcı alanından sistemin çekirdeğine kadar işlev çağrılarını kolaylaştırır. Linux çekirdeği işletim sisteminin kalbini oluşturur ve kullanıcı alanının donanım ile etkileşime girebileceği bir dizi araç sağlar. Linux çekirdeği, bellek ve süreç yönetimi, sanal dosya sistemi ve aygıt sürücüleri gibi katmanlı alt sistemlere ayrılabilir. Çekirdek kodu, Linux tarafından desteklenen tüm işlemci mimarileri için ortak olduğundan, mimariden bağımsız olarak kabul edilir. Bunun altında mimariye bağlı kod; yani, işlemci ve platforma özgü olan ve genellikle Bord Destek Paketi (BSP-Board Support Package) olarak adlandırılan koddur.

Mimariye Bağlı Kod, Linux çekirdek yığınının en alttaki katmanıdır. Linux çekirdeği, çok çeşitli farklı donanım platformlarını desteklemektedir ve bu platformların mimariye bağımlı kodla uyum içinde çalışması için özel kodlar gereklidir. Bu, BSP olarak bilinir ve mimari aileleri ve işlemcileri için kaynak dosyalar, ortak önyükleme destek dosyaları, DMA donanım arabirimleri, kesinti işleme ve belirli bir işlemci ailesi ile ilgili diğer öğeleri içerir [67].

Herhangi bir işletim sistemindeki gibi bir Linux aygıt sürücüsü, bir sistemdeki donanım aygıtları ile işletim sisteminde çalışan programlar arasında soyutlama sağlayan bir yapı olarak düşünülebilir. Aygıt sürücülerini kullanarak, belirli bir aygıttan bağımsız tüm programlarda standart bir çağrı kümesi uygulanabilir. Bu çağrılar, gerekli işlemleri gerçekleştiren sürücüye özgü işlevlere karşılık gelir. Bu sayede çekirdeğin dışındaki sürücülerin oluşturulmasına ve bunları gerektiği gibi eklenmesine olanak tanınmaktadır [68].

3.2.1.3.1. Linux Önyükleme

Bir Linux sistemi açıldığında veya sıfırlandığında meydana gelen ilk şey, işlemcinin önceden belirlenmiş bir konumda kod yürütmesidir. Bir masaüstü bilgisayar için, bu konum, Temel Giriş / Çıkış Sistemi (BIOS- Basic Input Output System) olarak bilinir ve ana kart üzerindeki flash belleğin bir bölümünde saklanır. Günümüz bilgisayarları, önyükleme aygıtları alanında çok yönlülük sunarken, BIOS'un yapacağı ilk şey, hangi

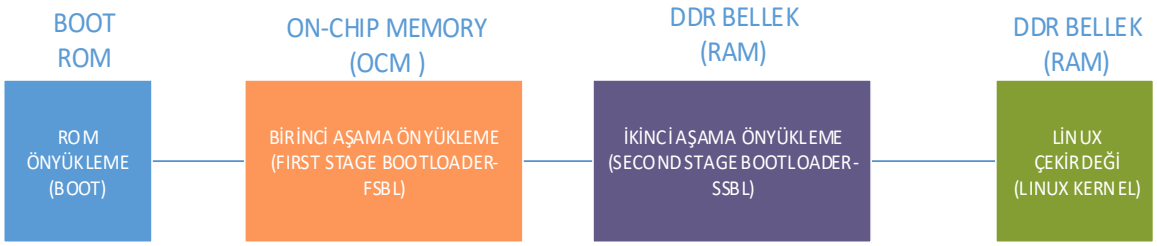
aygıtların önyüklenebilir seçenekler sunacağını belirlemektir [61]. Önyükleme (booting) aygıtı belirlendikten sonra, FSBL (First Stage Bootloader) Rasgele Erişimli Bellek (RAM-Random Access Memory)'e yüklenir ve işlemci tarafından yürütülür. FSBL çok küçük bir kod bölümüdür (512 bayttan az) ve tek amacı İkinci Aşamalı Önyükleyiciyi (SSBL-Second Stage Bootloader) RAM'e yüklemektir. SSBL, bir önyükleme menüsünün sunulduğu önyükleme işleminde bir aşamadır. Bu menü, kullanılabilir olası önyükleme seçeneklerinin bir listesini sunar. Listeden bir önyükleme seçeneği seçildiğinde, önyükleyici ilgili işletim sistemini yükler, daha sonra çekirdeği başlatmadan önce kendini sıkıştırır ve belleğe yükler. Çekirdek başlatılmadan önce önyükleyici tarafından yürütülen diğer işlevler vardır. Gerekli çekirdek birimleri başarıyla başlatıldığında, önyükleme işleminin son aşaması, ilk kullanıcı alanı işlevini başlatmaktır. Bu işlev, sistemin tüm üst düzey bölümlerini başlatır. Bu, Linux önyükleme işleminin üst düzey genel görünümünü sonuçlandırır.

3.2.1.3.2. Zynq Önyükleme

Gömülü Linux önyükleme işleminin bir masaüstü dağıtımından farklı olduğu bir dizi önemli alan vardır. Zynq aygıtları, açılışta başlatılan önyükleme ROM'undan başlayarak, bir dizi aşamada önyükleme gerçekleşir. Cihazın önyükleme kipi değeri önyükleme kipini belirler [69]. Önyükleme modu desteklenen arabirimler, JTAG, NAND Flash, NOR Flash, QSPI Flash veya SD karttır [70]. Önyükleme kipi belirlendikten sonra, önyükleme ROM'u önyükleme üstbilgisini okuyacaktır ve yapılandırma parametreleri verildiğinde, görüntüyü doğrulayacak ve FSBL görüntüsünü belirtilen arabirimden OCM(on Chip Memory)'ye yükleyecektir. Görüntü Çip Üzerinde Bellek (OCM-on Chip Memory)'ye aktarıldıktan sonra, CPU'nun kontrolü FSBL'e teslim edilir.

Tipik olarak, FSBL, CPU'yu kullanarak PS ve PL'yi yapılandırmak [70] için talimatlar içerir.

Şekil 3.2'de önyükleme aşamaları görünmektedir.



Şekil 3.2. Zynq Linux önyükleme aşamaları [71]

Linux'u bir Zynq-7000 AP cihazında önyüklemek için, önyükleme ortamınızda dört dosya bulunması gerekir. Bu dosyalar şunlardır:

1. BOOT.BIN
2. zImage
3. devicetree.dtb
4. ramdisk8M.image.gz

Bu dosyaların ilki BOOT.BIN, Zynq önyükleme görüntüsü dosyasıdır. Bu dosya 2 zorunlu dosya, FSBL, SSBL dosyaları ve bağlanabilir biçim (.elf) dosyaları, isteğe bağlı bir bit akışı (.bit) dosyası içerir. Adından da anlaşılacağı gibi, FSBL ve SSBL dosyaları, cihaza Linux yüklemek için kullanılan önyükleyicinin son aşamalarını içerir. Bit akışı, Zynq-7000 AP cihazının programlanabilir mantığını yapılandırmak için kullanılan dosyadır. ZImage dosyası, sıkıştırılmış Linux çekirdeğini içerir. SSBL tarafından belleğe yüklendikten sonra kendini açacaktır.

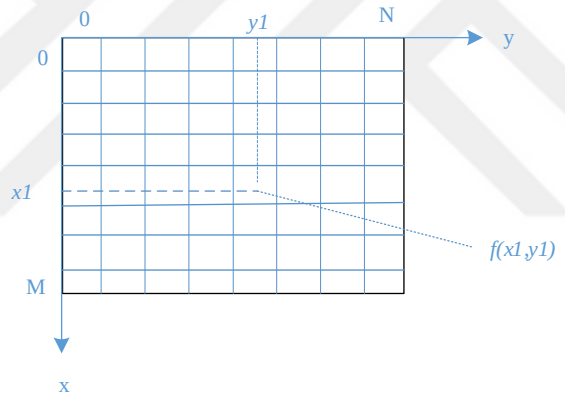
Linux'un önyükleneceği donanım bilgisi, aygıt ağacı (.dtb) dosyasında bulunur. Aygıt ağacı, aygıt ağacı kaynağı (.dts) dosyası olarak adlandırılan, insan tarafından okunabilir metin dosyasında tanımlanır ve daha sonra U-Boot tarafından anlaşılabilen aygıt ağacını oluşturmak için derleyici tarafından ikili formda derlenir.

Ramdisk8M.image.gz dosyasının son hali, belleğe yüklendiğinde RAM'in bir disk sürücüsü gibi kullanılmış olmasına izin veren bir RAM disk görüntüsüdür. Bu, Linux sisteminin kök dizini yüklemek için bir dosya sistemi olarak kullanabileceği RAM'de geçici bir disk sürücüsü oluşturur.

4. GÖRÜNTÜ İŞLEMEDE TEMEL KAVRAMLAR

4.1. Sayısal Görüntü

Sayısal görüntü, özel elektronik cihazlar kullanılarak görüntüden yansıyan ışığı mercek ya da objektiften yararlanarak bir düzlemde toplanan ışık enerjisinin, ışığa duyarlı devre elemanları sayesinde elektriksel sinyallere dönüştürülmesiyle oluşturulmuş sayısal verilerdir. M satır, N sütun sayısı olmak üzere iki boyutlu görüntünün örneklenmesi sonucu oluşan sayısal veri MxN boyutlu matrise benzetilebilir. Örneklenmiş görüntüdeki en küçük birim piksel olarak adlandırılır. Şekil 4.1'de MxN boyutlu görüntüyü tanımlayan matris formu gösterilmiştir. Görüntü *f* olarak gösterilirse, yatayda y1 noktası, düşeyde ise x1 noktası hizasında bulunan pikselinin değeri *f(x1,y1)* şeklinde gösterilmektedir.



Şekil 4.1. Görüntü verilerinin gösterimi

4.2. Renk Uzayları

Renkli görüntü verilerini tanımlamak için birçok renk uzayı vardır. Bu renk uzayları bazı renkleri temel renk kabul eder ve bu renklerin yoğunluğuna göre sayısal görüntü verisini oluşturur.

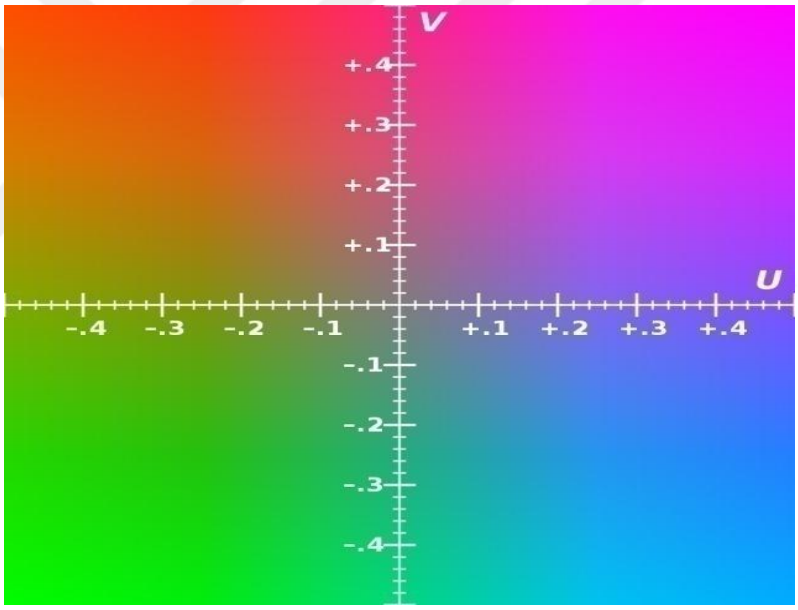
RGB Renk Uzayı: Bu renk uzayı kırmızı (**R**ed), yeşil (**G**reen) ve mavi (**B**lue) renkleri temel alınarak oluşturulmuştur. Bu renk uzayının kullanıldığı iki boyutlu görüntülerde her piksel; kırmızı, mavi ve yeşil renk yoğunluk değerine sahiptir. RGB renk uzayına sahip görüntüler tanımlanırken üç boyutlu matrisler kullanılır burada birinci ve

ikinci boyut resimde piksellerin konumunu üçüncü boyut ise ilgili pikselin renk yoğunluğunu belirtir.

HSV Renk Uzayı: Renk özü (**H**ue), doygunluk (**S**aturation) ve parlaklık (**V**alue) değerleri temel alınmıştır.

CMYK Renk Uzayı: Cam göbeği **(C**yan), Eflatun (**M**agenta), sarı (**Y**ellow), siyah (**K**ey) renklerinin kısaltmasıdır. CMYK renk uzayı, dijital renk tanımlamaları için belirtilen bu dört renk karıştırılarak yapılmaktadır.

YUV(YCbCr) Renk Uzayı: Y parlaklık (Luminance), **U** mavi renk yoğunluğu (Chrominance1), **V** kırmızı renk yoğunluğunu (Chrominance2) temsil eder. YUV biçimi renk yoğunlukları ve bunlara karşılık gelen değerler Şekil 4.2'de gösterilmiştir. YUV biçiminde Y parlaklık değeri RGB görüntüde gri görüntüye denk gelmektedir.



Şekil 4.2. YUV biçimi renk yoğunlukları ve değerleri

Şekil. 4.2'de görüldüğü gibi U kanalı mavi renk tonlarını, V kanalı ise kırmızı renk tonlarını göstermektedir. U ve V renk kanalları -0.5 ile +0.5 arasında değer almaktadır. Bu değerler 8 bitlik görüntü verisine dönüştürüldüğünde -0.5 değeri 0 değerine, 0.5 değeri ise 255 değerine karşılık gelmektedir.

YUV422 biçimi: YUV422 biçiminde MxN boyutlu görüntüde her pikseli tanımlamak için gerekli olan Y,U ve V bileşenlerinden Y, MxN defa örneklenirken U ve

V'nin her biri Mx(N/2) defa örneklenir. Örneğin resim 640×480 boyutunda ise 307200 bayt (640*480) Y bileşeni (siyah-beyaz kısım) yine 307200 bayt da renk bilgisidir (UV).

4.3. Gri Seviye Görüntü

Gri seviye görüntü, renkli görüntüden farklı olarak sadece beyaz ve siyah tonlarını içerir. İki boyutlu tek bir görüntü uzayıyla temsil edilirler. 8 bitlik renk derinliğine sahip bir görüntü için bir gri görüntü pikselinde 0 değeri siyah 255 değeri beyaz olmak üzere, sayı büyüdükçe siyahtan beyaza giden grinin farklı tonlarında değerler alırlar.

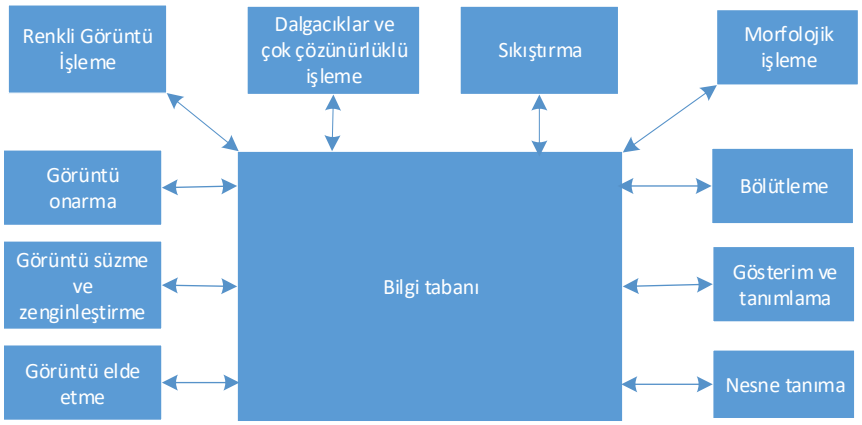
4.4. Sayısal Görüntü İşleme

Görüntü işleme, bir görüntü üzerinde bir takım işlemler yaparak görüntüde bir takım iyileştirmeler ve düzeltmeler yapma ya da ondan bir takım anlamlı veriler çıkararak görüntüdeki nesneleri tanıma çalışmalarını içermektedir. Bir görüntü iki boyutlu *f(x,y)* fonksiyon olarak tanımlanırsa, görüntü işleme bu *f(x,y)* fonksiyonunu farklı matematiksel işlemlere tabi tutarak bu fonksiyon üzerinde değişiklik yapmak veya bu fonksiyondan anlamlı veriler çıkarmak olarak tanımlanabilir.

Bilgisayar teknolojilerindeki gelişmelere paralel olarak sayısal veri işleme uygulamaları giderek artmaktadır. Bir görüntü karesi birçok pikselden oluşur. Her piksel RGB renk uzayı dikkate alındığında her renk için 8 bitlik renk derinliğine karşılık 24 bit (3 bayt) veriden oluşur. Görüntünün boyutuna bağlı olarak piksel sayısı artacak bu da görüntü verisi boyutunu arttıracaktır. Bu ölçüde büyük miktardaki veriyi işlemek için hem büyük bellek kapasitesine hem de hızlı hesaplama yeteneğine sahip sistemler gerekmektedir. Bilgisayarların boyutlarının küçülmesi, bellek kapasitelerinin ve veri işleme hızlarının artışı görüntü işleme teknolojilerindeki gelişmeyi hızlandırmıştır. Tüm bu gelişmeler sayısal görüntü işleme teknolojisinde kullanılan yazılımların da gelişmesini de sağlamıştır.

4.5. Sayısal Görüntü İşlemede Temel Adımlar

Sayısal görüntü işleme bazı temel adımlara sahiptir. Şekil 4.3'de bu adımlar gösterilmektedir.



Şekil 4.3. Sayısal görüntü işleme aşamaları

1. Görüntü elde etme: Görüntülerin çoğu, bir aydınlatma kaynağı ve bu kaynaktan gelen enerjinin görüntülen nesnelerce yansıtılması ve bu yansıyan enerjinin bir takım cihazlarla algılanmasıyla elde edilir.

2. Görüntü zenginleştirme: İşleme sonucunda elde edilen görüntü orijinal görüntüden görsel olarak daha iyi olmaktadır. Zenginleştirme yöntemleri çok çeşitlidir. Ancak burada önemli olan, zenginleştirilen görüntünün uygulamaya yönelik olmasıdır. Örneğin, uydu görüntüleri için uygulanan zenginleştirme işlemi, X ışınlı görüntüler için uygun olmayabilir.

3. Görüntü onarma: Görüntü onarma; orijinal görüntüyü, görüntü bozucu etkisine uygun olarak geliştirilen bir süzgeçten geçirerek bozulmayı ortadan kaldırmaktır.

4. Renkli görüntü işleme: Renkli işleme, nesnelerdeki renk bilgisini kullanmayı amaçlayan uygulamalarda kullanılmaktadır. Renk bilgisiyle nesne tanıma ve nesneyi görüntüden çıkarma sağlanabilmektedir.

5. Dalgacıklar: Farklı çözünürlük mertebelerindeki görüntüleri göstermenin temelidir.

6. Sıkıştırma: Görüntüyü saklamak için gerek duyulan depolamayı ve iletmek için istenen bant genişliğini azaltmayı hedefleyen teknikleri içerir.

7. Morfolojik işlemler: Görüntüdeki sınırlar, iskeletler ve dışbükey zarf gibi bölgesel şekilleri elde etmek için kullanılan yöntemlerdir.

8. Bölütleme yöntemleri: Bir görüntüyü oluşturan bölge veya nesnelerin alt bölümlere ayrılarak daha alt seviyeden sorunların çözümüne yardımcı olacak işlemlerdir.

9. Gösterim ve tanımlama: Bölütlenmiş görüntüdeki her bir bölümü tanımlama yöntemlerini oluşturur.

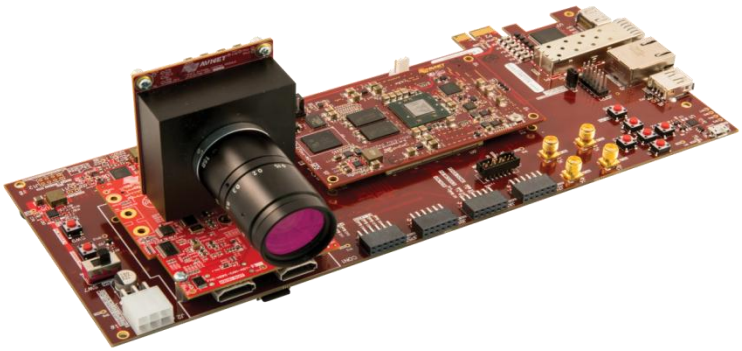
10. Nesne tanıma: Yukarıda anlatılan görüntü işleme yöntemleri sonucunda hedef nesne için görüntüden elde edilen belirleyici verilerden bir karar mekanizması yardımı ile hedef nesneye benzer nesneleri tanıma işlemlerini kapsamaktadır.

5. DENEYSEL ÇALIŞMA

5.1. Kullanılan Donanımlar

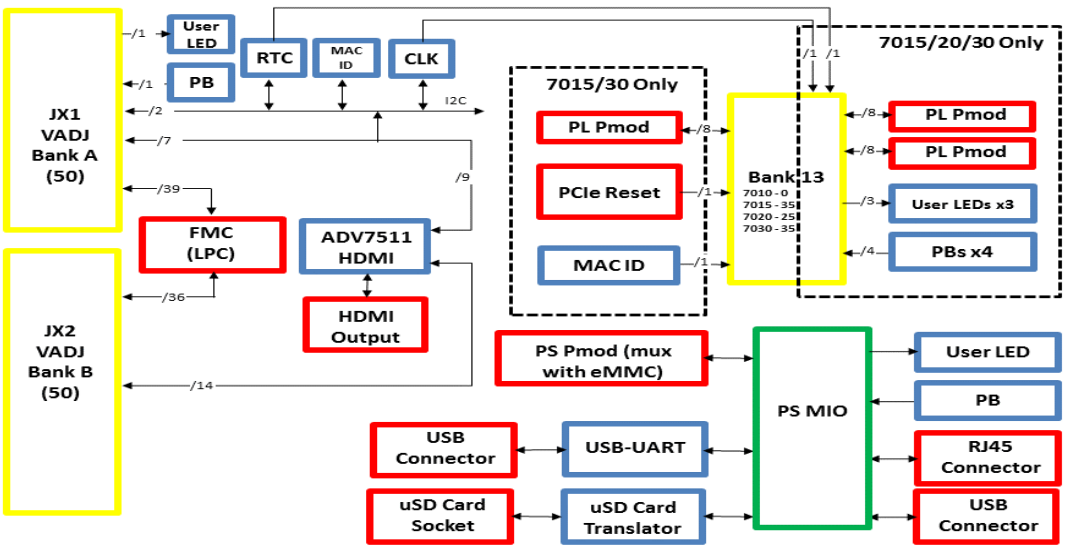
5.1.1. Picozed Gömülü Görüntü İşleme Kartı

Bu tezde, Şekil 5.1'deki gömülü sistemli görüntü işleme uygulamaları için geliştirilmiş Picozed FPGA kartı kullanılmıştır. Bu kart, Picozed FMC anakart ve Picozed 7Z030 Birim Üzerinde Sistem (SOM-System on Module) olarak iki ayrı birimden oluşan programlama kartı; görüntü işleme uygulamaları için kullanılabilecek kamera konektörü ve HDMI giriş/çıkış birimleri bulunan FMC birimi ve Python-1300 kamera içermektedir. Picozed 7Z030 SOM üzerinde, Zynq-7000 mimarili XC7Z030-1SBG485C serili SoC bulunmaktadır. Çok fazla kullanılabilir çevresele sahip bir programlama kartının ayrıntıları Ek 1'de verilmiştir.



Şekil 5.1. Picozed gömülü görüntü işleme kartı

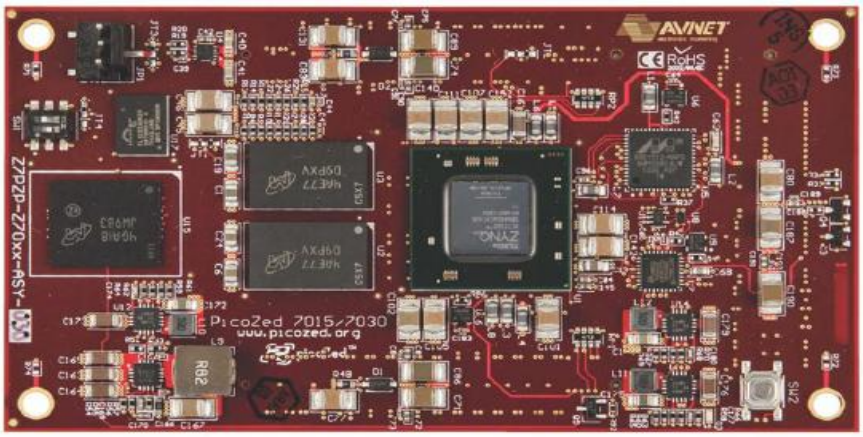
Bu geliştirme kartına ait çevreseller ve donanımları gösteren şema Şekil 5.2'de gösterilmiştir.



Şekil 5.2. Picozed gömülü görme kartı donanım şeması [72]

5.1.1.1. Xilinx XC7Z030-1SBG485C AP SOC Modül

Xilinx XC7Z030-1SBG485C AP SOC birimi, Xilinx XC7Z030 SoC FPGA içerir. Birim uyumlu geliştirme kartlarında kullanılabilecek şekilde tasarlanmıştır. Bu birim Şekil 5.3'te gösterilmektedir.



Şekil 5.3. Xilinx XC7Z030-1SBG485C AP SOC modül

Yapılandırma kip seçimi için değiştirilebilir anahtar bulunmaktadır. Bu birim üzerinde bulunan XC7Z030 serisi FPGA'ların işlemci ve lojik blok özellikleri Ek 2'de verilmiştir.

5.1.1.2. Yapılandırma Kipleri

Zynq-7000 AP SoC aygıtları hem güvenli olmayan hem de güvenli önyüklemeyi destekleyen çok aşamalı bir önyükleme işlemini kullanır. Picozed sıfırlandığında, cihaz kipi pinleri, kullanılacak birincil önyükleme aygıtını belirlemek için okunur. Bu aygıtlar; NOR, NAND, Q-SPI, SD Kart veya JTAG'dır. PicoZed 7015/7030, bu önyükleme aygıtlarının 3'üne izin verir. Bunlardan QSPI varsayılan değerdir. SD kart ve JTAG önyükleme aygıtlarına, pin ayarları değiştirilerek kolayca erişilebilir.

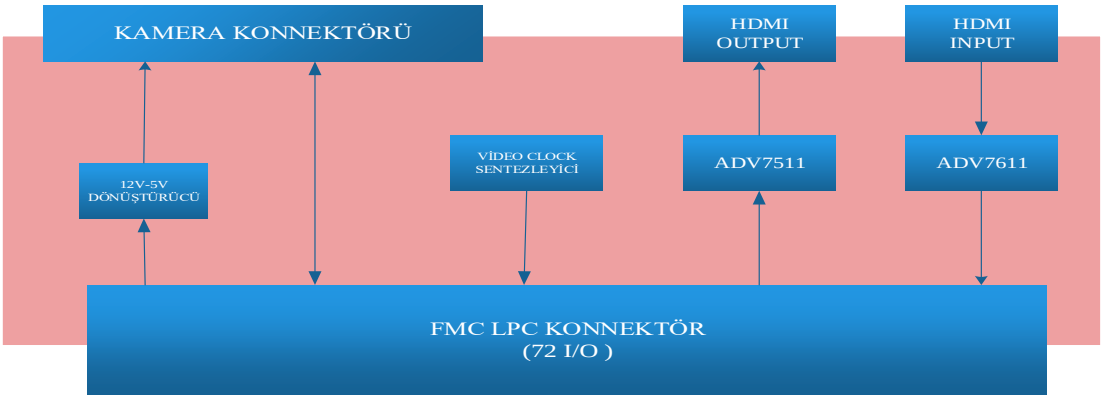
5.1.1.3. Saat Kaynağı

PicoZed 7030 SOM, özel bir 33.3333 MHz saat kaynağını Zynq-7000 AP SoC PS'ye bağlar. PS altyapısı, PL sistemi için dört adete kadar saat üretebilir. PL için üretilebilecek saat, en çok 200 MHz'dir.

5.1.1.4. HDMI Giriş/Çıkış FMC Birimi

FMC-HDMI-CAM, video işleme için tasarlanan arabirimleri içeren FMC birimidir. Herhangi bir kontrol birimine sahip değildir. Tek başına kullanılamaz ve üzerindeki çevresellerin kullanımı için bu birimin takılacağı harici bir karta ihtiyaç duyar. FMC birimi, isteğe bağlı kamera birimlerinin doldurulmasını sağlayan bir kamera ara yüzüne de sahiptir.

Şekil 5.4'de, FMC-HDMI-CAM FMC birimi mimarisini göstermektedir.



Şekil 5.4. HDMI ve kamera birimi [73]

5.1.2. Yüksek Çözünürlüklü Monitör

Bu çalışmada; 1920x1080 çözünürlüklü görüntü desteği sağlayan, HDMI girişe sahip, 21.5” ekran boyutunda monitör kullanılmıştır. Bu sayede işlenmiş görüntü ekranda gösterilmiştir. Monitöre ilişkin görsel Şekil 5.5’te gösterilmiştir.

Şekil 5.5 Yüksek çözünürlüklü monitör

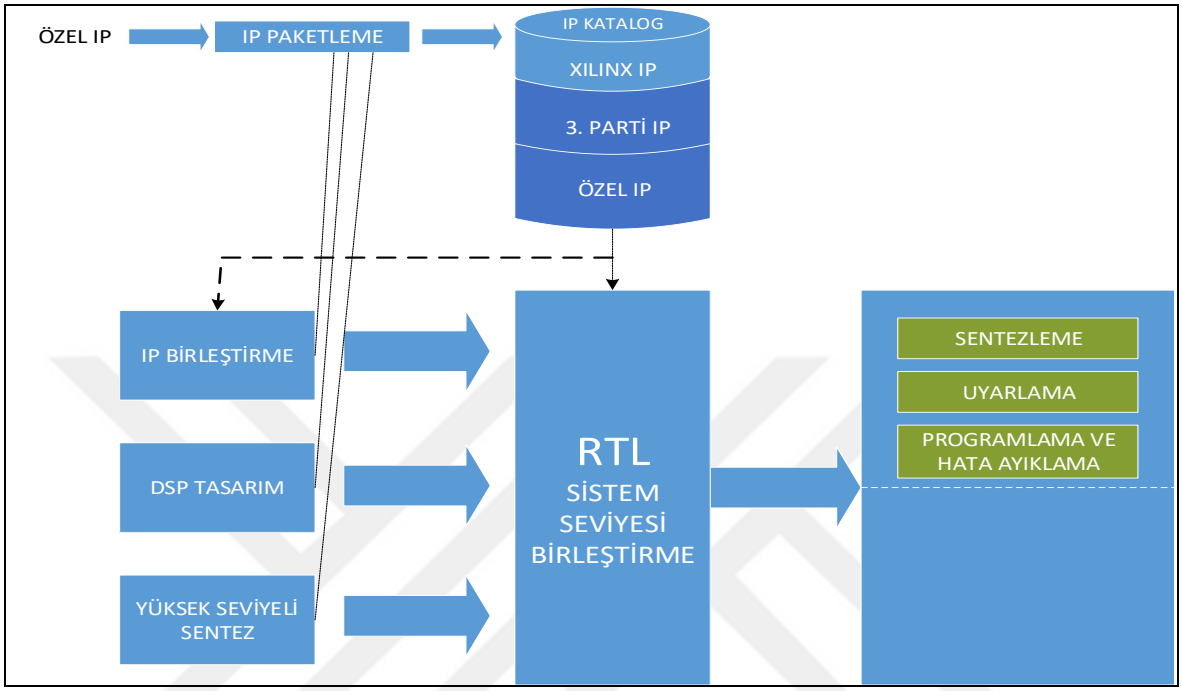
5.2. Kullanılan Yazılımlar

5.2.1. Vivado Design Suite

Xilinx tarafından geliştirilen Vivado Design Suite, Zynq platformu için donanımsal programlamanın yapıldığı programlama ortamıdır. Vivado programlama için çoklu dil desteği sağlamaktadır. Bunlar, VHDL, Verilog ve şematik programlama ortamıdır. İstenildiği takdirde VHDL ve Verilog ile yapılan projeler şematik eşdeğerine dönüştürülerek şematik programlama ortamında kullanılabilir.

Xilinx Vivado, çok sayıda hazır IP çekirdek desteği sunmaktadır. Bu IP çekirdeklerin büyük kısmı Zynq platformunun sahip olduğu AXI ara yüzünü desteklemektedir. Bunlar kullanılarak daha kısa sürede tasarımlar geliştirilmesi mümkün olmaktadır. Ayrıca Vivado, üçüncü parti IP çekirdeklerin ve IP çekirdek tasarım araçları kullanılarak tasarlanan IP’lerin de tasarımlarda kullanılmasını mümkün kılmaktadır. Çoğu donanım üreticisi donanımlarının FPGA’lar ile kullanımını kolaylaştırmak için Vivado uyumlu IP desteği de

sunmaktadır. Bu sayede gerekli donanımların projelere eklenmesi çok kolay olmakta ve programcılara tasarımlarını çok daha hızlı yapma imkânı tanımaktadır. Şekil 5.6'da IP çekirdek kullanım aşamaları gösterilmiştir.



Şekil 5.6. IP çekirdek kullanımı [46]

Vivado programlama aracı PL programlama yapmasının yanında, Zynq platformunun barındırdığı PS'in yapılandırmasında da kullanılmaktadır. Şekil 5.7'de Vivado Design Suite'te iş akış şeması görülmektedir.

Şekil 5.7. Vivado Design Suite iş akış şeması

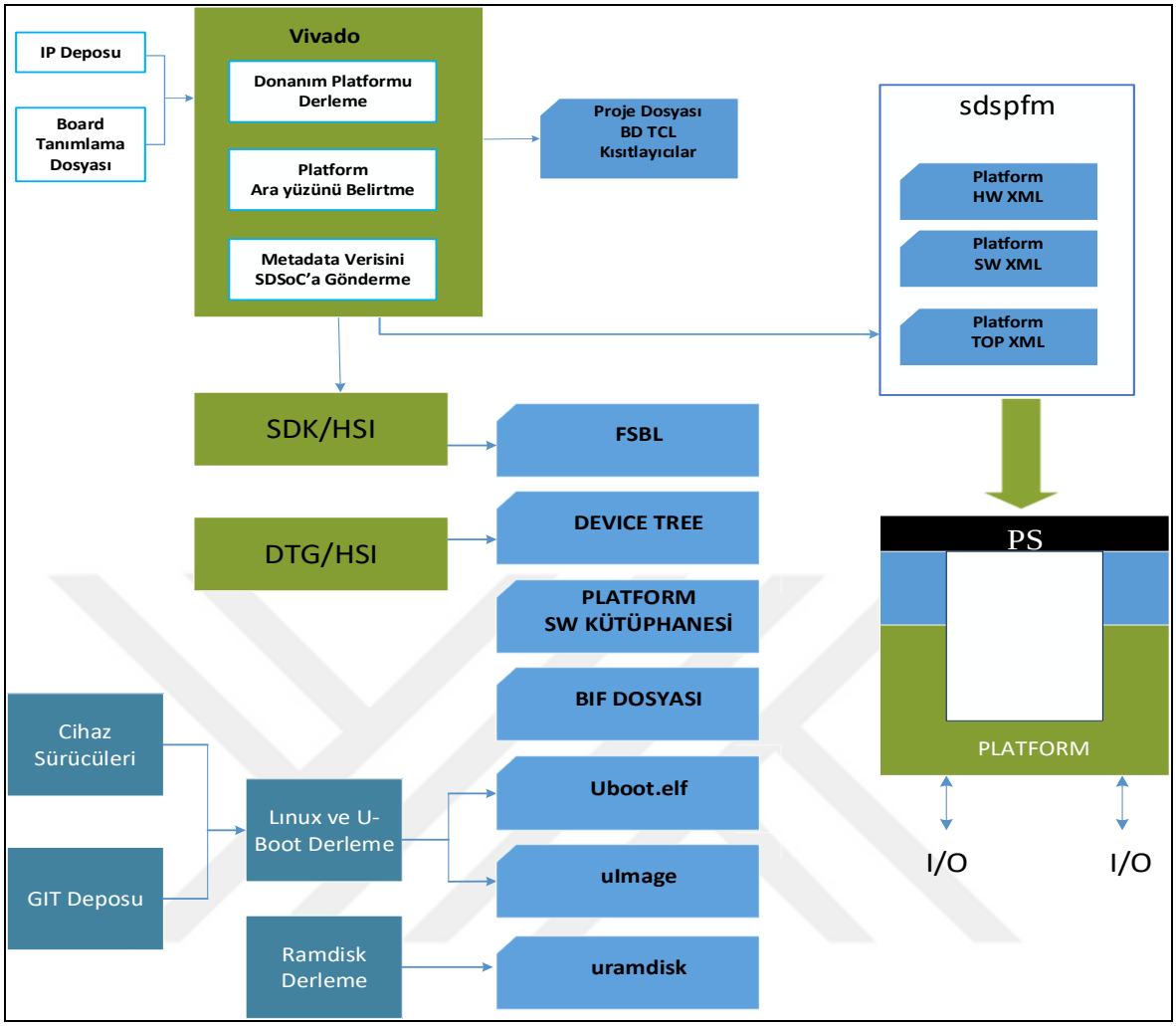
Bu çalışmada Zynq platformu PL tasarımında Vivado programlama aracı kullanılmıştır. Tasarımlarda şematik programlama kullanılmış, tasarımlarda Xilinx tarafından sağlanan IP desteğinden faydalanılmıştır. Ayrıca AVNET firması tarafından tasarlanan programlama kartı kullanıldığı için bu firma tarafından sağlanan IP çekirdeklerden de faydalanılmıştır.

5.2.2. Xilinx SDSoC

Xilinx SDSoC, Xilinx tarafından Zynq platformu cihazlarının hem PS ve hem PL tarafını programlamak için geliştirilmiş Eclipse tabanlı gömülü sistem geliştirme ortamıdır. Ortam, Zynq cihazları için Vivado Design Suite'te oluşturulmuş donanımsal platformlar kullanılarak PS programlama yapmaya ve bu platformlara AXI ara yüzü bağlantılı donanımsal hesaplama blokları tasarlayarak eklemeye imkân tanıyan ve donanım hızlandırmalı hesaplama yeteneği kazandırmayı amaçlayan ortamdır. Programlama dili olarak C/C++ dillerini desteklemektedir. Yeni tasarlanacak donanım hızlandırmalı hesaplama blokları için Vivado HLS aracını kullanılmaktadır. Vivado HLS ortamımda kullanılan C /C ++ dilleri de donanımsal fonksiyonlar tasarlanırken kullanılabilmektedir. Bu sayede tek bir ortam kullanılarak hem PL hem de PS programlamak mümkün olmaktadır. Ancak SDSoC ile donanımsal programlama yaparken tasarlanan bütün donanımların AXI arayüzü ile uyumlu olması dikkat edilmesi gereken bir noktadır.

Xilinx SDSoC üç tip işletim sistemini destekler. Bunlar; Standalone OS, RTOS ve Linux işletim sistemidir.

Bu çalışmada SDSoC yazılımı kullanılmıştır. AVNET firmasının Picozed kartları için oluşturduğu SDSoC Linux platformu kullanılarak uygulamalar geliştirilmiştir. Bir SDSoC platformu Şekil 5.8'deki bölümleri içermektedir.



Şekil 5.8. SDSoC platformu bölümleri [74]

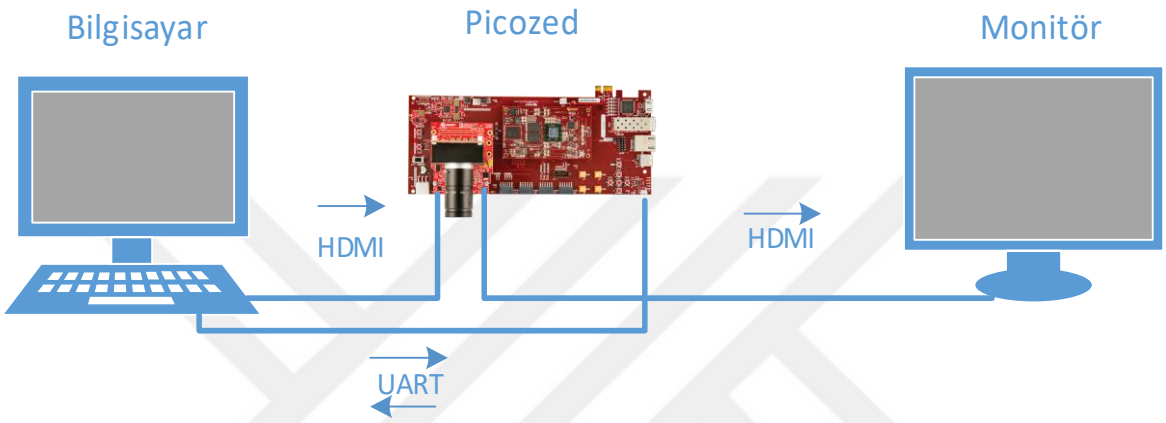
5.3. Sistem ve Donanım Tasarımı

Bu çalışmada, Zynq platformu üzerinde çalışan gömülü bir sistem tasarlanmıştır. Bu gömülü sistem, PL yapılandırmasından ve PS üzerinde çalışan Linux işletim sisteminden oluşmaktadır. Sistem genel olarak HDMI girişinden aldığı görüntü verilerini işlemekte ve yine HDMI çıkışından dışarı aktarmaktadır. Görüntü, 1920x1080 piksel çözünürlüğünde ve 60 çerçeve/saniye (fps-frame per second)'den oluşmaktadır. Ayrıca Linux işletim sistemi bir kullanıcı grafik arayüzüne sahip değildir ve konsol üzerinden UART arabirimi vasıtasıyla kontrol edilmektedir.

Geliştirilen sistemin donanım kısmında, gömülü sistemin çalışacağı platform olan Picozed Gömülü Görme Kartı; bu platformun programlanacağı ve kontrol edilip

işlenmemiş görüntü verilerinin aktarılacağı bilgisayar; işlenmiş görüntü verilerinin görüntüleneceği ve gömülü sistem için çıkış birimi olacak monitörden oluşmaktadır.

Donanımlar arası veri bağlantısı; bilgisayarla gömülü sistem arası UART arabirimi, bilgisayardan gömülü sisteme HDMI standardında veri haberleşmesi, gömülü sistemden monitöre HDMI standardında veri haberleşmesi şeklindedir. Sisteme ait donanımsal yapı Şekil 5.9'da görülmektedir.



Şekil 5.9. Tasarlanan sistemin temsili gösterimi

Gömülü sistem SD kart üzerinden yapılandırılmaktadır. Sistem için geliştirilen uygulama SD kart üzerinden PL'i yapılandırmakta, PS üzerinde çalışan Linux işletim sistemi ise aynı SD kart üzerinden yüklenmektedir.

Yazılım kısmı ise üç bölümden oluşmaktadır. Bunlar; SDSoC platformu için PL donanımı geliştirme, SDSoC platformunda yazılım geliştirme, SDSoC platformunda AXI standardı uyumlu donanım geliştirme çalışmalarıdır. AVNET tarafından sağlanan SDSoC platformunda video işleme uygulamaları için hali hazırda kullanılabilecek PL donanımı olduğu için uygulama geliştirme aşamasında bu kısım üzerinde durulmamıştır. Ancak geliştirilmiş donanımın yazılımsal gereksinimlerini bilmek için bölümüne tam hakimiyet gerekmektedir. Bu amaçla bu hazır donanım ayrıntılı bir şekilde incelenmiş ve gerekli olan yazılımsal çalışmalar tespit edilmiştir. Bunun dışında PL donanımına müdahale edilmemiştir. PS'i programlamak için yazılımsal programlama kullanılmış, PL'de donanım hızlandırmalı hesaplamalar yapmak için Vivado HLS aracının desteklediği yapıda AXI standardı ile uyumlu PL donanım geliştirme çalışmaları yapılmıştır.

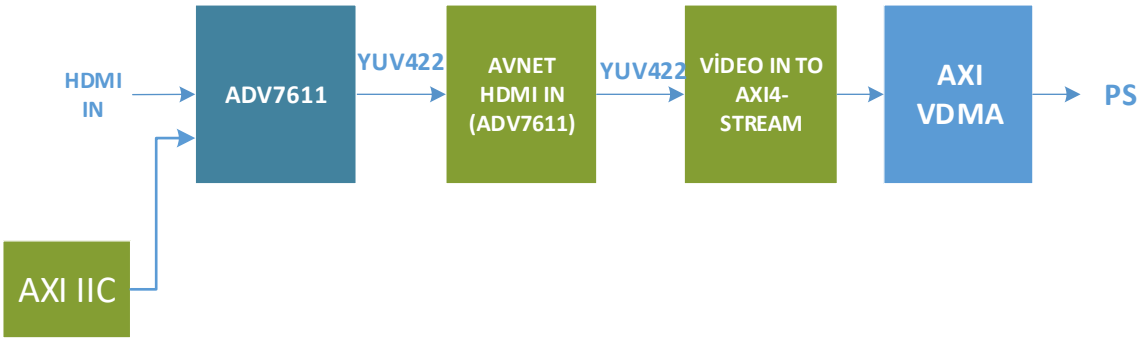
5.4. SDSoC Yazılımsal Programlama

Yazılımsal programlamada gömülü sistem geliştirme ortamı olarak Eclipse tabanlı SDSoC IDE kullanılmıştır. SDSoC platformunda gömülü sistem için Linux işletim sistemi seçilmiştir. Kullanılan Linux işletim sistemi, Xilinx tarafından önceden oluşturulmuş bir Linux işletim sistemidir. Bu Linux işletim sistemi, PL'de kullanılan çoğu IP çekirdek için sürücüye sahiptir. Ayrıca DRM Framework sayesinde grafik işlemleri için donanımlara kolayca erişim sağlanmaktadır. SDSoC platformu bu işletim sistemi üzerinde çalıştırılabilen gömülü sistem projeleri geliştirmek üzere tasarlanmıştır. Proje derlendiğinde Linux işletim sistemi dosyaları ve çalıştırılabilir dosya üretilmektedir.

Bir SDSoC projesi, bir donanım platformuna bağlı olarak geliştirildiği için yazılımsal PS programlama yaparken hangi donanımların üzerinde bu işlemin gerçekleştirildiği çok önemlidir. Çünkü PL'de tasarlanan AXI arabirimine sahip donanımların PS tarafından kontrol edilmeleri gerekir.

Görüntü verileri HDMI girişinden okunmaktadır. Ancak cihazda doğrudan kullanılabilecek HDMI girişi mevcut değildir. Bu nedenle HDMI giriş ve çıkış modülüne sahip FMC birim kullanılmıştır. FMC de bulunan HDMI girişi doğrudan PL giriş çıkış bloklarına bağlı değildir. HDMI girişi, ADV7611 kontrolcüsüne bağlanmıştır. ADV7611 HDMI girişinden HDMI protokolüyle aldığı veriyi, 24 bit YUV444, 24 bit RGB ya da 16 bit YUV422 biçiminde çıkışa aktarmaktadır. Kullanılan cihaz, sadece 16 bit YUV422 biçimi kullanılabilir şekilde tasarlanmıştır.

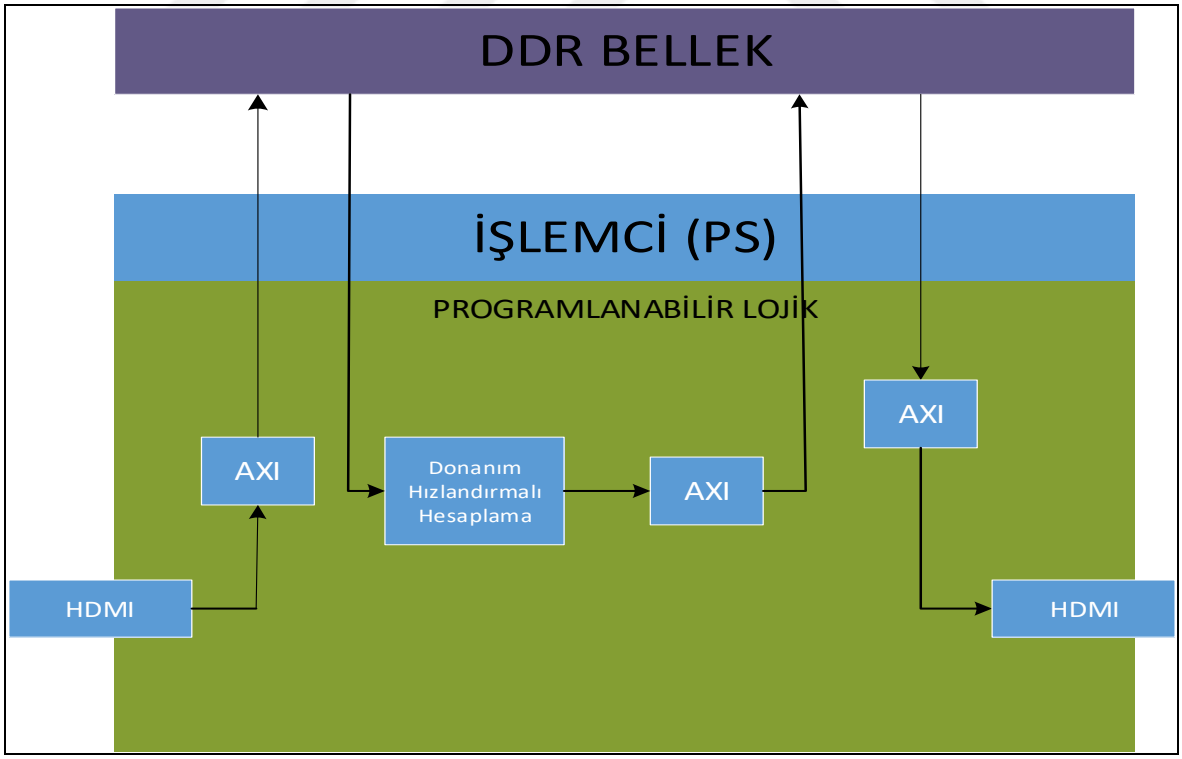
FMC biriminde, ADV7611 çıkışı ve kontrol girişleri, PL giriş /çıkış bloklarına bağlantılı şekilde tasarlanmıştır. ADV7611 kontrolcüsü, I2C haberleşmesi üzerinden kontrol edilmektedir. PL tasarımında kullanılan AXI IIC IP çekirdeği, I2C parametrelerinin PS tarafından verilmesini sağlamaktadır. ADV7611'den YUV422 biçimindeki veriyi alabilmek için, AVNET tarafından sağlanan AVNET HDMI IN IP çekirdeği kullanılmaktadır. Bu IP çekirdek çıkışından alınan veri SDSoC ortamında kullanılmak üzere Video AXI4-Strem arayüzü ve AXI Video DMA sayesinde PS belleğine (DDR RAM) taşınmaktadır. HDMI girişinden verinin alınmasından PS'ye taşınmasına kadar işlem aşamaları Şekil 5.10'da gösterilmiştir.



Şekil 5.10. Vivado Design Suite donanımsal yapı veri taşınım aşaması

AXI4-Stream ile PS'e taşınan veriler, SDSoC donanım hızlandırmalı hesaplama kullanılarak işlenmektedir. İşlenen veriler daha sonra AXI4-Stream ile PS belleğine oradan da AXI4 ile PL'e taşınmaktadır. PL'de video verilerinin çıkış birimine YUV422 biçiminde yönlendirilmesini sağlamak için Xylon tarafından geliştirilmiş Multilayer Video Controller IP çekirdeği kullanılmaktadır.

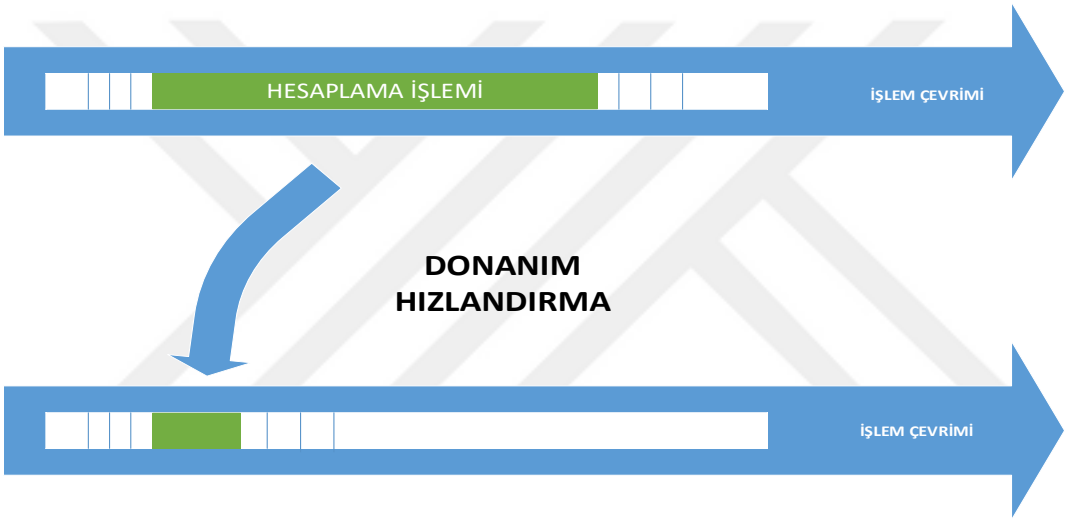
Şekil 5.11, bir SDSoC projesinin genel yapısını göstermektedir.



Şekil 5.11. SDSoC projesinin genel yapısı [75]

5.4.1. Donanım Hızlandırmalı Hesaplama

Donanım hızlandırmalı hesaplama, SDSoC ortamında PS programlama yaparken kısa işlem zamanı gerektiren ya da PS mimarisi kullanılarak zaman alacak işlemleri PL'in paralel işlem yeteneklerini kullanarak daha kısa sürede gerçekleştirmek için kullanılan yapıdır. Donanım hızlandırmalı hesaplamada oluşturulan hesaplama donanımı PS'e AXI arayüzü üzerinden bağlantı halindendir. Kontrol ve parametre değerlerini PS'den almaktadır. İş akışı yazılım mimarisi boyunca ilerlerken, iş akışının bir parçası PL üzerinde oluşturulan donanımda gerçekleşir. Bu iş akışı yapısı Şekil 5.12'de gösterilmiştir.



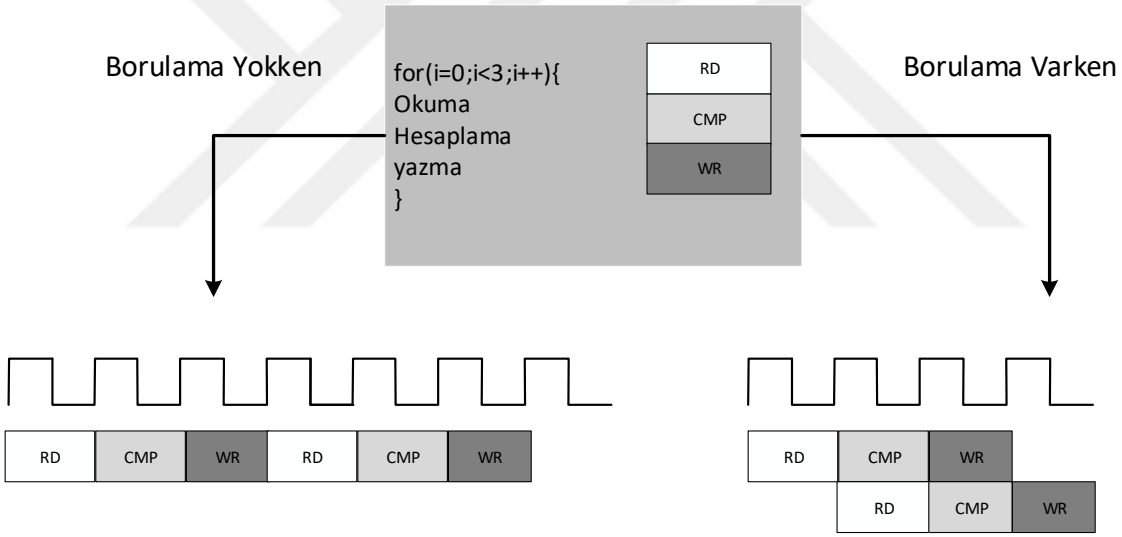
Şekil 5.12. SDSoC donanım hızlandırmalı hesaplama iş akışı [46]

Donanım hızlandırmalı hesaplama işleminde tasarlanacak donanım için farklı bir programlama ortamı kullanılmaz ve bu işlem de SDSoC programlama ortamında C/C++ dilleri kullanılarak gerçekleştirilir. Kodların derleme aşamasında program doğrudan Vivao HLS ortamına bağlanır. Kodlar Vivado HLS ortamı tarafından derleneceği için tamamen bu ortamın gereklerini sağlamalıdır. Her ne kadar C/C++ ile yazılıyor olsa da bu kodların PL' de sentezlenebilir kodlara dönüştürülmesi için bazı kısıtlamalar mevcuttur. Vivado HLS ortamında bütün matematiksel fonksiyonlar, operatörler ve değişkenler doğrudan sentezlenebilir değildir. Çoğu fonksiyon ve değişken için kütüphaneler kullanılmaktadır.

Vivado HLS ortamında donanımları hızlandırmak için bazı direktifler kullanılmaktadır. Bu direktifler sayesinde PL'in sahip olduğu bazı yetenekler derleyiciye belirtilebilmektedir.

5.4.1.1. Döngü Boru Hattı ve Döngü Açma

Her iki döngü boru hattı ve döngü açması, döngü özyinelemeleri arasındaki paralellikten yararlanarak donanım işlevinin performansını iyileştirir [76]. Döngü borulama C/C ++ gibi sıralı dillerde, bir döngüdeki işlemler sırayla yürütülür ve döngünün sonraki yinelemesi, yalnızca geçerli döngü yinelemesindeki son işlem tamamlandığında başlayabilir. Döngü boru hattı, bir döngüdeki işlemlerin aşağıdaki Şekil 5.13'te gösterildiği gibi eşzamanlı olarak uygulanmasına izin verir.



Şekil 5.13. Döngü borulama [75]

Yukarıdaki şekilde gösterildiği gibi, boru hattı olmadan, iki okuma işlemi arasında üç saat döngüsü vardır ve tüm döngünün tamamlanması için altı saat döngüsü gerektirir. Bununla birlikte, boru hattı ile iki okuma işlemi arasında sadece bir saat döngüsü vardır ve bu, tüm çevrimin tamamlanması için dört saat döngüsü gerektirir; yani, döngünün bir sonraki yinelemesi, mevcut yinelemenin bitmesinden önce başlayabilir. Döngü borulama için ***#pragma HLS PIPELINE*** direktifi kullanılmaktadır.

Döngü açması döngü yinelemeleri arasında paralellikten yararlanma başka bir tekniktir. Döngü gövdesinin birden çok kopyasını oluşturur ve döngü yineleme sayacını buna göre ayarlar. (5.1)'deki temsili kod parçacığı, normal döngüyü gösterir:

```
int sum = 0;
 for(int i = 0; i < 10; i++) {
sum += a[i];
}
```
(5.1)

Döngü bir basamak açıldığında ise (5.2)'deki kodlar elde edilir.

```
int sum = 0;
for (int i = 0; i < 10; i+=2) {
sum += a[i];
 sum += a[i+1];
}
```
(5.2)

Döngü açması, her döngü yinelemesinde daha fazla işlem oluşturur. Böylece Vivado HLS bu işlemler arasında daha fazla paralellikten yararlanabilir. Daha fazla paralellik, daha fazla verim ve daha yüksek sistem performansı anlamına gelir. Döngü açması için ***#pragma HLS unroll factor=N*** direktifi kullanılır. Burada N değeri döngünün ne kadar sayıda açılacağını belirtir. Yukarıdaki döngü açma örneğinin eşdeğeri (5.3)'te gösterilmiştir.

```
int sum = 0;
for(int i = 0; i < 10; i++) {
 #pragma HLS unroll factor=2
 sum += a[i];
}
```
(5.3)

5.4.1.2. Döngü Borulama ve Açma ile Elde Edilen Paralelleştirmeyi Sınırlayan Faktörler

Her iki döngü boru hattı ve döngü açma, döngü yinelemeleri arasındaki paralellikten yararlanır. Bununla birlikte, döngü yinelemeleri arasındaki paralellik, iki ana faktörle sınırlıdır: bir tanesi döngü yinelemeleri arasındaki veri bağımlılıkları, diğeri ise kullanılabilir donanım kaynaklarının sayısıdır.

Bir özyinelemede, bir işlemden üretilen veri, sonraki bir işlemde kullanılıyorsa bağımlı veri olarak adlandırılır. Bu tür işlemlerde veri bağımlılığı olduğu için paralel çalışmak imkânsızdır. Bu nedenle döngü borulama ve açma işlemleri yapılamaz.

Veri borulama ve açma işlemleri donanımsal olarak gerçekleştirildiği için eğer hedef platformda yeterli donanım mevcut değilse bu işlemler gerçekleştirilemez.

5.5. Yapılan Uygulamalar

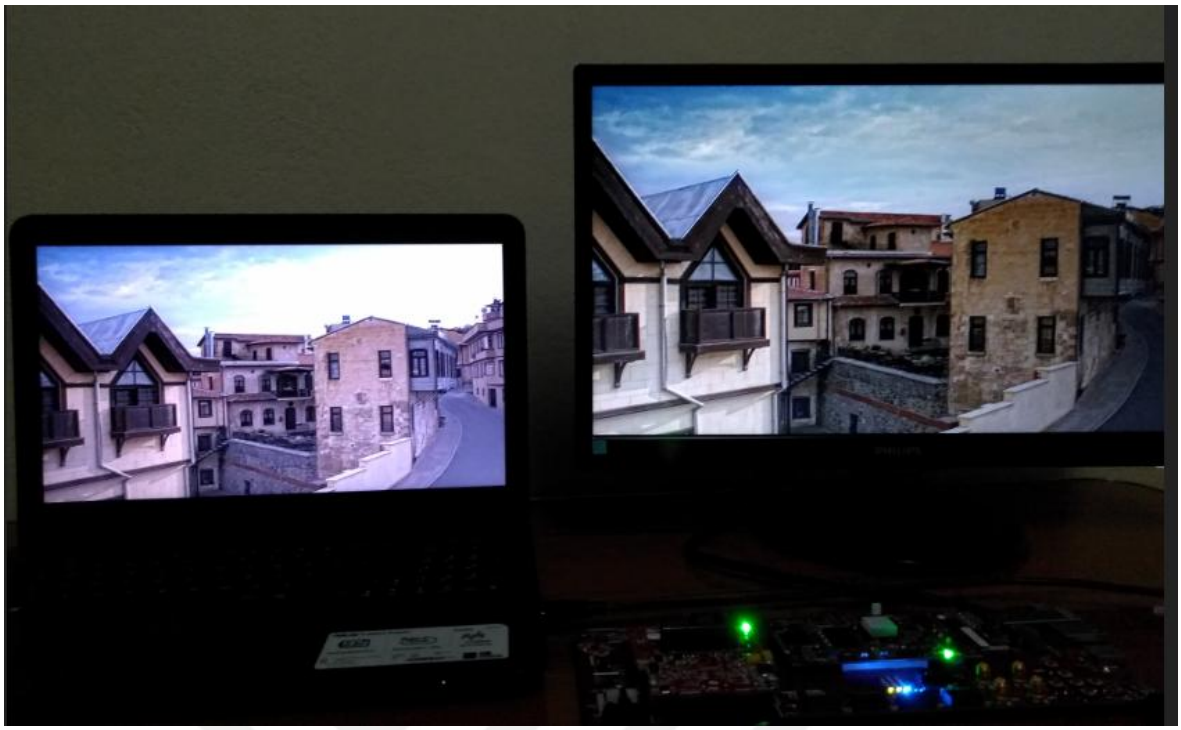
5.5.1. Alınan Görüntüyü Ekrana Yansıtma

Yapılan çalışmada ilk olarak HDMI girişiyle alınan görüntü ekrana yansıtılmıştır. HDMI girişine, bilgisayar HDMI çıkışı bağlanmış ve bilgisayardan gelen 1080P 60 çerçeve/saniye (fps) görüntü işlenmek üzere gömülü sisteme iletilmiştir.

Gömülü sistemde tasarlanan yazılımda, ilk olarak donanım hızlandırma olmadan ARM işlemci yardımıyla görüntü ekrana aktarılmış ve giriş görüntüsünün yalnızca 7 çerçevelik kısmı ekrana yansıtılabilmiştir. Bu işlem esnasında bir çerçevelik tampon bellek (buffer) kullanılmıştır.

Bu uygulamanın ikinci aşamasında, donanım hızlandırmalı yapı kullanılarak veriler alınıp ekrana yansıtılmıştır. Bu işlem esnasında herhangi bir çerçeve kaybı olmadan işlem tamamlanmıştır. Sadece bir çerçevelik tampon bellek kullanıldığı için giriş görüntüsü ile çıkış görüntüsü arasında bir çerçevelik zaman gecikmesi vardır.

Şekil 5.14'te bu uygulamaya ilişkin ekran görüntüsü görülmektedir.



Şekil 5.14. Alınan görüntünün ekrana yansıtılması uygulama görseli

Tablo 5.1'de donanım hızlandırmalı hesaplamaya ilişkin kullanılan FPGA kaynakları gösterilmiştir.

Tablo 5.1. Alına görüntünün ekrana yansıtılmasında kullanılan FPGA kaynakları

Kaynaklar	**Kullanılan**	**Toplam**	**% Kullanım**
DSP	0	400	0
Blok RAM	11	256	4.15
LUT	4551	218600	2.08
FF (Flip Flop)	5965	157200	3.79

Tablo 5.2'de ARM işlemci üzerinde yapılan yazılımsal programlama ve donanım hızlandırmalı hesaplama işlem süreleri gösterilmiştir.

Tablo 5.2. Uygulama işlem süreleri karşılaştırması

İşlem Platformları	**İşlenen Görüntü Karesi (çerçeve/saniye)**
Donanım hızlandırmalı hesaplama	60
ARM işlemci	7

5.5.2. Gri Seviye Renk Dönüştürme

Bu aşamada, renkli görüntü gri seviye görüntü biçimine çevrilmiştir. Giriş YUV422 biçiminde olan görüntü de üç tanımlayıcı renk yoğunluk değeri bulunmaktadır. Bunlar; parlaklık (Y), mavi renk (U) ve kırmızı renk (V)'dir. Bu biçimde bir renk iletilirken UY ve VY şeklinde 16 bit veri boyutunda iletilir. Her 16 bitlik verinin en ağırlıksız 8 biti parlaklık yani gri seviye değerini verdiği için gri seviye biçimine rahatça geçilebilmektedir. U ve V renk yoğunluk değerleri kendi renk aralığı içinde sıfır yapılarak gri seviye görüntü elde edilmektedir. img_gri, gri seviye görüntüyü; img_renkli, renkli görüntüyü göstermek üzere, gri seviye dönüşümü (5.4)'te gösterildiği şekilde yapılmıştır.

$$img\_gri = (img\_renkli \;\&\; (0xFF)) \mid (0x80 << 8) \tag{5.4}$$

Bu uygulamada 16 bitlik veri ayrıştırılmış ve gri seviye renk değeri elde edilmiştir. Alınan görüntünün gri seviye dönüşümü yapılarak çıkışa aktarılmıştır.

Uygulama ilk olarak yazılımsal olarak ARM işlemcide gerçekleştirilmiş ve giriş görüntüsünün 7 çerçevelik kısmı gri seviye biçiminde çıkışa aktarılmıştır. Bir çerçevelik tampon bellek kullanılmıştır.

İkinci kısımda aynı çalışma donanım hızlandırmalı hesaplama kullanılarak yapılmış ve herhangi bir veri kaybı olmadan çıkış görüntüsü elde edilmiştir. Şekil 4.15'de ekran görüntüsü gösterilmiştir.

Şekil 5.15. Gri seviye renk dönüşümü uygulama görseli

Tablo 5.3'te donanım hızlandırmalı hesaplamaya ilişkin kullanılan FPGA kaynakları gösterilmiştir.

Tablo 5.3. Gri seviye renk dönüşümünde kullanılan FPGA kaynakları

FPGA Kaynakları	**Kullanılan**	**Toplam**	**% Kullanım**
DSP	0	400	0
Blok RAM	9	256	3.4
LUT	4543	218600	2.08
FF (Flip Flop)	5925	157200	3.77

Tablo 5.4'de ARM işlemci üzerinde yapılan yazılımsal programlama ve donanım hızlandırmalı hesaplama işlem süreleri gösterilmiştir.

Tablo 5.4. Uygulama işlem süreleri karşılaştırması

İşlem Platformları	**İşlenen Görüntü Karesi (çerçeve/saniye)**
Donanım hızlandırmalı hesaplama	60
ARM işlemci	7

5.5.3. Kenar Bulma Çalışmaları

Ayrıt saptama, görüntü renk yoğunluk değerlerinde hızlı değişim olan görüntüleri bölütlemek için kullanılan bir yaklaşımdır. Ayrıtları bulma amacıyla renk yoğunluk değerlerindeki değişimleri saptama, birinci veya ikinci türevi kullanarak gerçekleştirilebilir.

5.5.3.1. Görüntü Gradyanı

Bir f görüntüsünün *(x,y)* konumundaki en büyük değişimini vermektedir. Gradyan ∇f ile, yatay gradyan g_x, ve dikey gradyan g_y olarak gösterilirse, gradyan (5.5)'de gösterildiği şekilde oluşur;

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{5.5}$$

Bir yöndeki gradyan büyüklüğünün değişim oranı değeri olan ve M(x,y) olarak gösterilen ∇f vektörünün büyüklüğü (5.6)'da verilmiştir.

$$M(x,y) = mag(\nabla f) = \sqrt{{g_x}^2 + {g_y}^2} \tag{5.6}$$

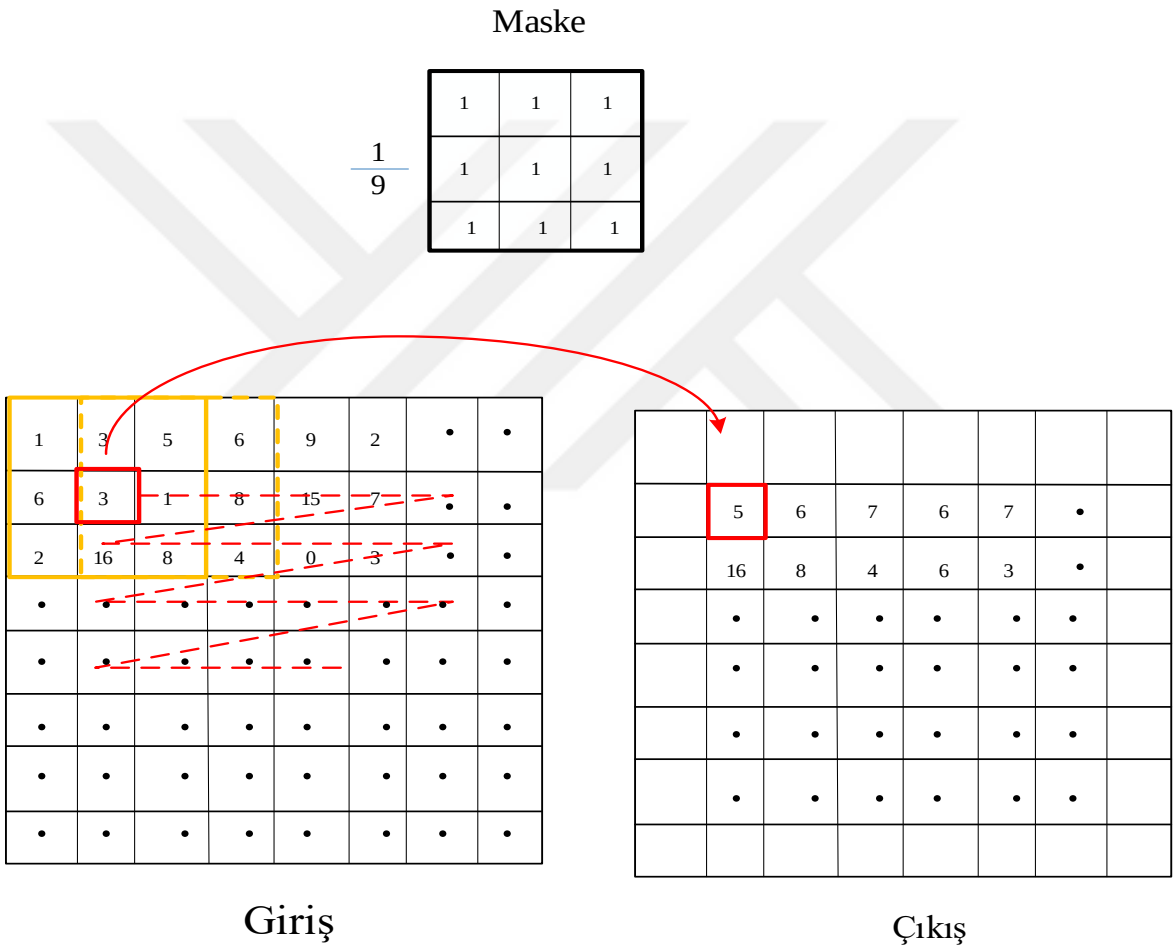
5.5.3.2. Görüntüde Konvolüsyon

Konvolüsyon işlemi; bir çekirdek (maske) şablonun, resim üzerindeki piksellerle 'kaydırma ve çarpma' işlemi olarak tanımlanabilir. Bu işlemde çekirdek şablon resim üzerinde kaydırılır ve değeri resim üzerindeki uygun piksellerle çarpılır. Konvolüsyon

işlemi; O çıkış görüntüsü, I giriş görüntüsü, f ise çekirdek matrisi olmak üzere matematiksel olarak (5.7)'deki gibi tanımlanabilir.

$$O(x,y) = \iint_{-\infty}^{\infty} f(x,y)I(x-\alpha, y-\beta)\, d\alpha d\beta \qquad (5.7)$$

Burada α ve β kaydırma parametrelerini göstermektedir. Görüntüde konvolüsyon işlemi ise Şekil 5.16'da gösterilmiştir.



Şekil 5.16. Görüntüde konvolüsyon işlemi

5.5.3.3. Gradyan İşleçleri

Bir görüntünün gradyanı, görüntüdeki her bir piksel konumunda $\partial f/\partial x$ ve $\partial f/\partial y$ hesaplanarak bulunur. Ancak görüntü işlemede sayısal ayrık görüntülerle uğraşıldığı için

sayısal türev yaklaşımı kullanılır. Bu yaklaşıma göre (5.5) tekrar düzenlenirse (5.8) ve (5.9) elde edilir.

$$g(x) = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \tag{5.8}$$

$$g(y) = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y) \tag{5.9}$$

İki boyutlu görüntülerde yatay ve dikey ayrıntıları ortaya çıkartmak için dikey ve yatay noktada sayısal türev alan iki boyutlu maskeye ihtiyaç vardır. 2x2 boyutlu maskeler kavramsal olarak basittirler fakat maskelerin simetrik olma koşulu sağlanmadığından hesaplamada kullanışlı değildirler. Bu nedenle hesaplamalarda 3x3 boyutlu maskeler kullanılır. Şekil 5.17'e gösterilen üç boyutlu temsili çerçevede basit sayısal türev.(5.10) ve (5.11)'deki gibi hesaplanır.

I_1	I_2	I_3
I_4	I_5	I_6
I_7	I_8	I_9

Şekil 5.17. 3x3 görüntü penceresi

$$g(x) = \frac{\partial f}{\partial x} = (I_7 + I_8 + I_9) - (I_1 + I_2 + I_3) \tag{5.10}$$

$$g(y) = \frac{\partial f}{\partial y} = (I_3 + I_6 + I_9) - (I_1 + I_4 + I_7) \tag{5.11}$$

Bu hesaplamayı gerçekleştirmek için maske matrisi bütün görüntü matrisi boyunca konvolüsyon işlemine tabi tutulur.

Bu türevsel hesaplamayı yapan Prewitt maske ise Şekil 5.18'de gösterilmektedir.

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

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

Şekil 5.18. Prewitt işleci

(5.12) ve (5.13)'teki türev hesaplama yöntemiyle elde edilen Şekil 4.18'deki Prewitt maskesinde merkez konumunda iki kullanılmasının gürültüyü yumuşattığı görülmüştür. Sobel işleci ise daha iyi gürültü bastırma özelliği nedeniyle tercih edilmektedir.

$$g(x) = \frac{\partial f}{\partial x} = (I_7 + 2I_8 + I_9) - (I_1 + 2I_2 + I_3\ ) \quad (5.12)$$

$$g(y) = \frac{\partial f}{\partial y} = (I_3 + 2I_6 + I_9) - (I_1 + 2I_4 + I_7\ )$$

Şekilde 5.19'da bir başka kenar tespit etme işleci Sobel işlecinin maskeleri gösterilmektedir.

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

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

Şekil 5.19. Sobel işleci

Diyagonal (çapraz) ayrıntılar için ise Robert maskesi kullanılmaktadır. Bu maske Şekilde 5.20'de gösterilmiştir.

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

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

Şekil 5.20. Roberts işleci

Gradyan işleminden sonra (5.6)'da gösterildiği gibi gradyan genliğinin hesaplanması gerekmektedir. Kare ve karekök alma işlemleri hesapsal yükü arttıracağı için uygulamalarda tercih edilmez. Sıklıkla gradyan genliğinin mutlak değeri yaklaşımı kullanılmaktadır. Bu hesaplama (5.14)'te gösterilmiştir.

$$M(x,y) \approx |g_x| + |g_y| \tag{5.14}$$

5.5.3.4. Kenar Bulma Uygulaması

Kenar bulma uygulaması çalışmasında Sobel, Prewitt, Roberts kenar bulma algoritmaları kullanılmıştır. Bu algoritmalar ilk olarak yazılımsal hesaplama kullanılarak yapılmış, ikinci aşamada ise aynı hesaplama işleminde donanım hızlandırmalı hesaplama yeteneği kullanılarak FPGA üzerinde gerçekleştirilmiştir.

Bu uygulamalarda her bir kenar bulma yönteminde sadece filtre matrisleri değiştiği için farklı yöntemler kullanılmasına rağmen aynı hesaplama hızları elde edilmiştir.

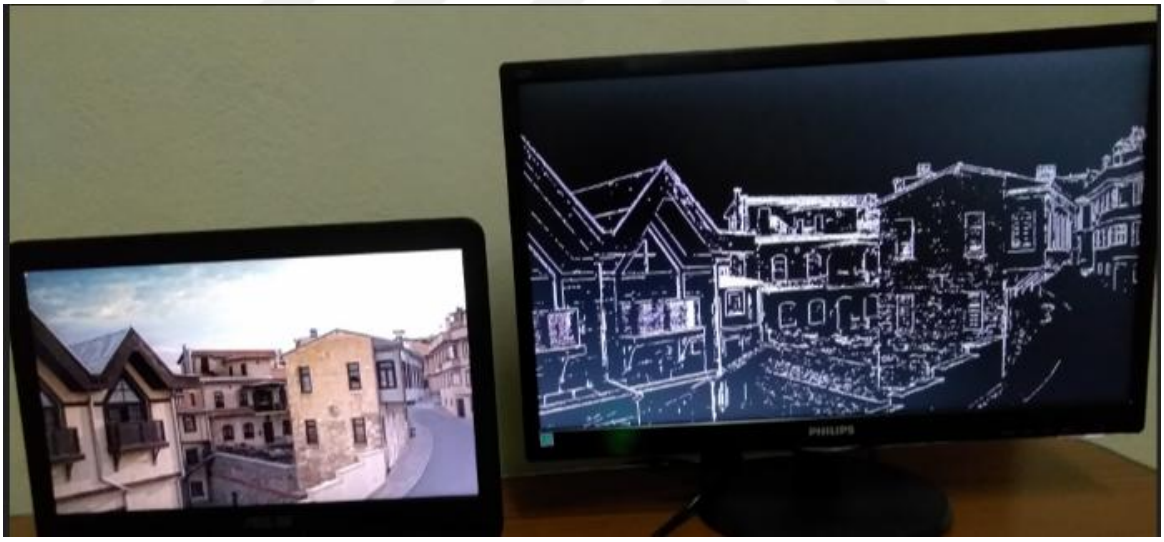
Hesaplama işleminde 3x1920 boyutlu tampon bellek kullanılmış ve bu tampon bellekten konvolüsyon işlemi için 3x3'lük matrisler alınmıştır. Bu matrisler maske matrisi ile konvole edilmiştir.

Hesaplama işlemi sonucunda, yazılımsal programlama aşamasında, giriş görüntüsünden sadece 3 çerçevelik kısım işlenerek çıkışa aktarılabilmiştir. Donanım hızlandırmalı hesaplama kullanılarak yapılan uygulamada ise herhangi bir görüntü kaybı yaşanmadan bütün çerçeveler işlenebilmiştir.

Şekil 5.21'de Prewitt filtresi uygulanmış örnek görüntü görülmektedir.

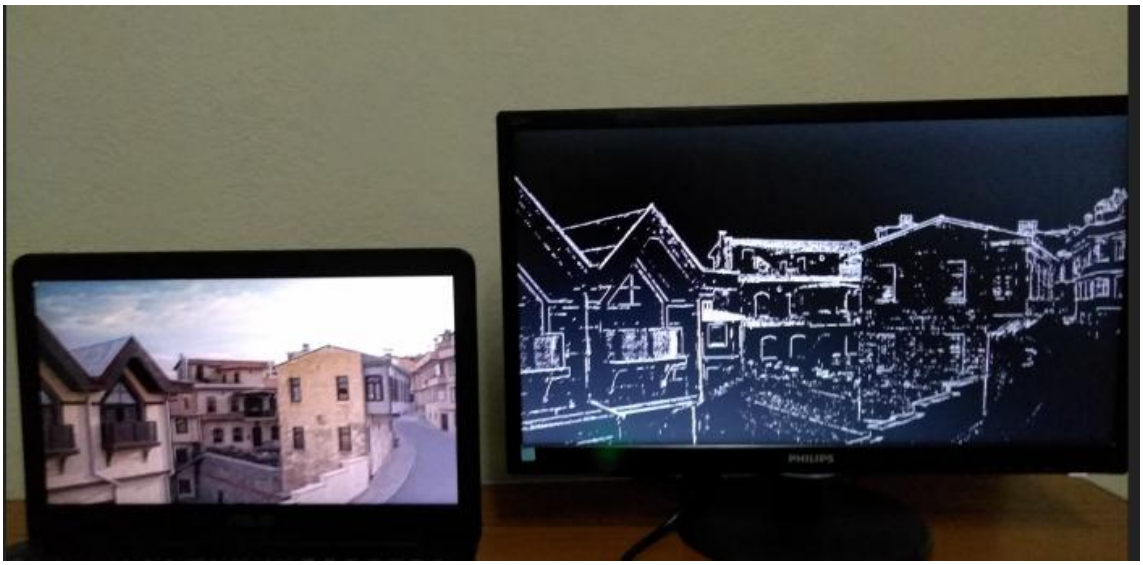
Şekil 5.21. Prewitt filtresi uygulanmış örnek görüntü

Şekilde 5.22'de Roberts filtresi uygulanmış görüntü gösterilmiştir.



Şekil 5.22. Roberts filtresi uygulanmış örnek görüntü

Şekil 5.23'de Sobel filtresi uygulanmış örnek görüntü gösterilmiştir.



Şekil 5.23. Sobel filtresi uygulanmış örnek görüntü

Tablo 5.5'de bu uygulamada kullanılan FPGA kaynakları gösterilmiştir. Tablo 5.6'da ise aynı işlemin yazılımsal ve donanımsal olarak gerçekleştirilmesinin karşılaştırılması gösterilmiştir.

Tablo 5.5. Kenar bulma uygulamasında kullanılan FPGA kaynakları

Kaynaklar	**Kullanılan**	**Toplam**	**% Kullanım**
DSP	0	400	0
Blok RAM	9	256	3.4
LUT	4861	218600	2.22
FF (Flip Flop)	6175	157200	3.93

Tablo 5.6. Uygulama işlem süreleri karşılaştırması

İşlem Platformları	**İşlenen Görüntü Karesi (çerçeve/saniye)**
Donanım hızlandırmalı hesaplama	60
ARM işlemci	3

6. SONUÇLAR

Bu çalışmada, görüntü işleme algoritmalarından gri seviye görüntü işlemleri ve kenar tespit algoritmaları FPGA tabanlı, gerçek zamanlı olarak gerçekleştirilmiştir. Çalışmada AVNET tarafından geliştirilen Xilinx Zynq-7000 mimarili XC7Z030 SoC FPGA içeren Picozed gömülü kartı kullanılmıştır.

Çalışmada melez bir yapıya sahip Zynq FPGA'ların getirilerinden faydalanmak ve görüntü işleme algoritmalarını gerçek zamanlı gerçekleştirirken donanım hızlandırmalı hesaplama yapısını kullanmak amaçlanmıştır.

Çalışmada kullanılan gömülü platform hem ARM işlemci hem de FPGA bloklarına sahip olduğu için giriş çıkış çevresellerini kontrol etme gibi bazı işlemler yazılımsal olarak, bazı önemli hesaplamalar ise FPGA hesaplama blokları kullanılarak hesaplanmıştır. AXI arayüzü kullanılarak işlemci ve FPGA blokları arasında iletişim sağlanmaktadır. Platform, Linux işletim sistemi desteği sağladığı için yazılımsal uygulamalarda ve FPGA hesaplama bloklarına AXI arayüzü sayesinde erişerek gerekli komutları vermede Linux çekirdeği kullanma imkânı sağlanmıştır.

Uygulamaların geliştirilmesinde SDSoC IDE kullanılmış ve hedef işletim sistemi Linux olarak seçilmiştir. SDSoC ortamı üzerinde uygulama geliştirilebilir platform desteği sağlamaktadır. Bu çalışmada AVNET tarafından sağlanan görüntü işleme uygulamaları için geliştirilmiş platform kullanılmıştır. SDSoC ortamında Zynq cihazında, Linux işletim sistemi üzerinde çalıştırılabilir proje geliştirilmiştir. Geliştirme aşamasında ortam yazılımsal programlamanın yanında Vivado HLS ortamında donanımsal hesaplama yapısına uygun fonksiyonlar için FPGA blokları yapılandırma kodları üretebilmektedir. Bu sayede uygulama içindeki görüntü işleme algoritmaları için donanımsal yapı kullanılmıştır.

Çalışmada, kartın FMC HDMI giriş/çıkış modülü HDMI girişinden alınan görüntü verileri işlenerek yine aynı modülün HDMI çıkışından monitöre aktarılmaktadır. Görüntü kaynağı olarak bilgisayar kullanılmıştır. Her bir uygulama, karşılaştırmalı olarak önce yazılımsal olarak ARM işlemci üzerinde, daha sonra sadece zaman alıcı hesaplama işlemleri için donanım hızlandırmalı yapı kullanılarak gerçekleştirilmiştir.

İlk uygulamada, bilgisayardan alınan 1920x1080 piksel çözünürlükteki görüntü verileri FPGA ile alınıp ekrana yansıtılmıştır. İkinci uygulamada, alınan görüntü verileri gri seviye görüntü biçimine çevrilip çıkışa aktarılmıştır. Üçüncü uygulama olarak gri

seviye biçimine çevrilen görüntüye kenar bulma algoritmaları uygulanmıştır. Kenar bulma algoritmalarından Roberts, Prewitt ve Sobel filtreleri gerçekleştirilmiştir.

Uygulamalarda; video kare hızı, bilgisayar HDMI çıkışının sınırı olan 60 çerçeve/saniye kare hızında ve 1920x1080 piksel çözünürlüğünde sınırlanmıştır. Bu hızda ve çözünürlükte görüntü girişine sahip yazılımsal uygulamalarda en fazla 7 karenin işlenerek çıkışa aktarıldığı gözlemlenirken, aynı uygulamanın hesaplama kısmı donanım hızlandırmalı hesaplama yapısı kullanılarak yapıldığında en fazla 2 görüntü karesi kaybı yaşanmıştır. Bu kayıp sadece algoritma ilk çalıştığı anda olmuştur. Sürekli zamanda herhangi bir görüntü karesi kaybı yaşanmadığı görülmüştür. Sistemin mevcut uygulamalarla gerçek zamanlı çalıştığı görülmektedir.

Zynq kullanarak gömülü sistem tasarımı, başlangıç öğrenme aşamasında konusunda melez yapıyı öğrenme ve programlama konusu oldukça fazla zaman almaktadır. Platform çok geniş bir yapıda olduğu için öğrenirken çok fazla dokümandan destek almak gerekmektedir. Kapsamlı çalışmalarda öğrenme aşamasında çok fazla doküman desteği bazen işleri karmaşık hale getirmektedir. Zamanla, çok ve kapsamlı doküman desteği bu sakıncayı faydaya dönüştürmektedir. Programlama aşamalarında kullanılan yapı, oldukça verimli olduğu için öğrendikten sonra ilerlemek daha kolay olmaktadır. Bu yapı, kapsamlı uygulamalar için de daha verimli bir yapıya sahiptir. Ayrıca, Linux çekirdeği desteği sağlanması çoğu çevresel donanıma erişme ve bazı temel uygulamalar için kütüphane desteği bu yapıyı oldukça cazip kılmaktadır.

Bu çalışma sonunda, Zynq platformunu programlama ve bazı görüntü işleme algoritmalarını gerçek zamanlı gerçekleştirme amaçları amacına ulaşmıştır. Linux işletim sistemini öğrenme, Linux çekirdeğini tanıma ve Linux işletim sistemi üzerinde çalışacak uygulamalar geliştirme çalışmaları da bu tezden elde edilen kazanımlar olmuştur.

Elde edilen kazanımlar sayesinde ileri bir çalışma olarak aynı platform üzerinde yapay öğrenme uygulamaları da yapılabilecektir.

KAYNAKLAR

[1] **Eberli, F.,** 2013. Next generation FPGAs and SOCs-How embedded system can profit, Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Portland, OR, USA, 610-613.

[2] **Ning M., Shaojun, W., Yeyong, P., Yu, P.,** 2014. Implementation of LS-SVM with HLS on Zynq, Proceeding of the IEEE International Conference on Field-Programmable Technology (FPT), Shanghai, China, 346-349.

[3] **Coşkun, M., Uçar, A., Yıldırım, Ö., Demir, Y.,** 2017. Face recognition based on convolutional neural network, International Conference on Modern Electrical and Energy Systems (MEES), Kremenchuk, Ukraine, 376-379.

[4] https://www.embedded-vision.com/industry-analysis/market-analysis/2014/03/26/image-recognition-market-worth-2565-billion-2018, August 2014.

[5] **Kestur, S., Davis, J. D., Williams, O.,** 2010. BLAS comparison on FPGA, CPU and GPU, Proceedings of the IEEE Annual Symposium on VLSI, IEEE Computer Society, Washington, DC, USA, 288-293.

[6] **Neshatpour, K., Koohi, A., Farahmand, F.,** 2016. Big biomedical image processing hardware acceleration: A case study for K-means and image filtering, Proceedings of International Symposium on Circuits and Systems (ISCAS), Montreal, QC, Canada, 1134-1137.

[7] **Onat, E.,** 2017. FPGA implementation of real time video signal processing using Sobel, Robert, Prewitt and Laplacian filters, 25th Signal Processing and Communications Applications Conference (SIU), Antalya, Türkiye.

[8] **Han, Y., Oruklu, E,** 2014. E., Real-time traffic sign recognition based on Zynq FPGA and ARM SoCs, Proceedings of the IEEE International Conference on Electro/Information Technology, Milwaukee, WI, USA, 373-376.

[9] **Abdelgawad, H. M., Safar, M., Wahba, A. M.,** 2015. High Level Synthesis of canny edge detection, Engineering and Technology International Journal of Computer and Information Engineering, World Academy of Science, 9(1), 148-152.

[10] **Dobai, R., Sekanina, L.,** 2013. Image filter evolution on the xilinx Zynq platform, Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Torino, Italy, 164-171.

[11] **Russell, M., Fischaber, S.,** 2013. OpenCV based road sign recognition on Zynq, Proceeding of the IEEE International Conference Industrial Informatics (INDIN), Bochum, Germany, 596-601.

[12] **Ayadi, L. A., Loukil, H., Ayed, M. A. B., Masmoudi, N.,** 2018. Efficient implementation of HEVC decoder on Zynq SoC platform, Proceedings of IEEE International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sousse, Tunisia.

[13] **Yu, Z., Yang, S., Sillitoe, I., Buckley, K.,** 2017. Towards a scalable hardware/software co-design platform for real-time pedestrian tracking based on a ZYNQ-7000 device, Proceedings of IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Bangalore, India, 127-132.

[14] **Khongprasongsiri, C., Kumhom, P., Suwansantisuk, W., Chotikawanid, T., Chumpol, S., Ikura, M.,** 2018. A hardware implementation for real-time lane detection using high-level synthesis, International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, Thailand.

[15] **Srijongkon, K., Duangsoithong, R., Jindapetch, N., Ikura, M., Chumpol, S.,** 2017. SDSoC based development of vehicle counting system using adaptive background method, Proceedings of IEEE Regional Symposium on Micro and Nanoelectronics (RSM), Batu Ferringhi, Malaysia, 235-238.

[16] **Yao, A., Liu, B., Liu, G., Yu, Z.,** 2017. Embedded laser profile measurement based on Zynq, First International Conference on Electronics Instrumentation & Information Systems (EIIS), Harbin, China.

[17] **Monson J., Wirthlin M., Hutchings, B. L.,** 2013. Optimization techniques for a high level synthesis implementation of the Sobel fitler, International Conference on Reconfigurable Computing and FPGAs (ReConFig), Cancun, Mexico, 1-6.

[18] **Hong. N. T. K, Cecile, B, Tuan. V. P.,** 2014. Power evaluation of Sobel filter on Xilinx platform, IEEE Faible Tension Faible Consommation, Monaco, 1-5.

[19] **Swapnil G., Kavitkar, Prashant L. Paikrao,** 2014. FPGA based image feature extraction using xilinx system generator, (IJCSIT) International Journal of Computer Science and Information Technologies, 3743-3747.

[20] **Monson, J., Wirthlin, M., Hutchings, B.L.,** 2013. Implementing high-performance, low-power FPGA-based optical flow accelerators in C, Application-Specific Systems, Proceedings of the Architectures and Processors (ASAP), IEEE 24th International Conference, Washington, DC, USA, 363-369.

[21] **Gentsos C., Louisa C., Sotiropoulou N., Spiridon N.,** 2010. Real-time canny edge detection parallel implementation for FPGAs, Athens, Greece, 17th IEEE International Conference on Electronics, Circuits and Systems, 499-502.

[22] **Chaithra. M. N., Reddy, K. M.,** 2013. Implementation of canny edge detection algorithm on FPGA and displaying image through VGA interface, International Journal of Engineering and Advanced Technology (IJEAT), 2(6), ISSN: 2249 8958.

[23] **Long P.H.W., Tsoi K.H.,** 2005. Field programmable gate array technology for robotics applications, Robio 05, 70-76.

[24] **Salcic, Z.,** 1997. A microcontroller/FPGA-based prototyping system for embedded applications, Microprocessors and Microsystems, 21(4) , 249-256.

[25] **Maxfield, C.,** 2004, The design warrior's guide to FPGAs: devices, tools and flows, 978-0-7506-7604-5, Elsevier, Amsterdam.

[26] **Lindh, L., Starner, J., Adomat, J.,** 1996. Experiences with VHDL and FPGAs, Journal of Systems Architecture, 42(2), 97-104.

[27] **Kuon, I., Tessier, R., Rose, J.,** 2007. FPGA architecture: survey and challenges, Foundations and Trends in Electronic Design Automation, 2(2), 135-253.

[28] **Seals, R., Whapshott, G**., 1997. Programmable logic: PLDs and FPGAs, UK: Macmillan.

[29] **Ferrigno, L., Landi, C., Laracca, M.,** 2008. FPGA-based measurement instrument for power quality monitoring according to IEC standards, Proceedings of the Instrumentation and Measurement Technology Conference (IMTC), IEEE, 906-911.

[30] **Choong, F., Reaz, M. B. I.,** 2005. Implementation of power quality disturbance classifier in FPGA employing wavelet transform, ANN and Fuzzy Logic, SETIT, Tunus.

[31] **Torres-Huitzil, C., Arias-Estrada, M.,** 2004. Real-time image processing with a compact FPGA-based systolic architecture, Real-Time Imaging, 10(3), 177-187.

[32] **Omondi, A. R., Rajapakse, J. C.,** 2006. FPGA implementations of neural networks, Springer, ISBN: 0387284850.

[33] **Yılmaz, N.,** 2008. Alan programlamalı kapı dizileri (FPGA) üzerinde bir YSA'nın tasarlanması ve donanım olarak gerçekleştirilmesi, Yüksek Lisans Tezi, Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Konya.

[34] **Cavuslu, M.A, Karakaya, F.,** 2010. Hardware implementation of discrete wavelet transform and inverse discrete wavelet transform on FPGA, Proceeding of the Signal Processing and Communications Applications Conference (SIU), Diyarbakir, Turkey, 141-144.

[35] **Mahmoud, M.I.,** 2007. Comparison between haar and daubechies wavelet transformions on FPGA technology, World Academy of Science, Engineering and Technology , 26, 68-72.

[36] **Kung Y.S., Shu G.S.,** 2005. Development of a FPGA based motion control IC for robot arm, Proceeding of the IEEE International Conference on Industrial Technology, Hong Kong, China, 1397-1402.

[37] **Paralı L., Taskın S., Pinar A. M.. Reynolds R., Smith P., Bell L., Keller H.,** 2001. The design of Mars lander cameras for Mars pathfinder, Mars Surveyor 98 and Mars Surveyor 01, IEEE Transactions on Instrumentation and Measurement, 50(1), 63-71.

[38] **Kadir M. Z. A Ab Othman S., Tuan S., Lombigit L.,** 2008. On-Line power quality monitoring system using CompactRIO device: a review on system development, European Journal of Scientific Research ISSN 1450-216X, 21(2), 288-295.

[39] **Bakhri S. Ertugrul N. Soong W.L. Al-Sarawi S.,** 2007. Investigation and development of a real-time on-site condition monitoring system for induction motors**,** Proceeding of the Australasian Universities Power Engineering Conference, Perth, Australia, 1-6.

[40] **Bilik P., Koval L., Hajduk J.,** 2008. CompactRIO embedded system in power quality analysis, Proceedings of the International Multiconference on Computer Science and Information Technology, Wisia, Poland, 577-580.

[41] **Al-Naami B., Chebil J., Trabsheh B., Mgdob H.,** 2006. Developing custom signal processing algorithm with LabView FPGA and CompactRIO to detect the aortic stenosis disease, Computers in Cardiology, 33, 193-196.

[42] **Mousouliotis, P. G., Panayiotou, K. L., Tsardoulias, E. G., Petrou, L. P., Symeonidis, A. L.**, 2018. Expanding a robot's life: Low power object recognition via FPGA-based DCNN deployment, Proceedings of Modern Circuits and Systems Technologies (MOCAST), 1-4.

[43] **Hashemian R., Riddley J.,** 2007. FPGA e-Lab, a technique to remote access a laboratory to design and test, Proceedings of the IEEE International Conference on Microelectronic Systems Education, San Diego, CA, USA.

[44] **Katsantonis, K., Kachris, C., Soudris, D.,** 2018. Efficient hardware acceleration of recommendation engines: a use case on collaborative filtering, Proceedings of Reconfigurable Computing. Architectures, Tools and Applications: 14th International Symposium, ARC 2018, Santorini, Greece, 10824(67).

[45] **Tamilarasi, S., Sundararajan, J.,** 2018. FPGA based seizure detection and control for brain computer interface, Cluster Computing, 1-8.

[46] **Crockett, L. H., Elliot, R. A., Enderwitz, M. A., Stewart, R. W.,** 2014. The Zynq Book, Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, Scotland, UK, 1st Edition.

[47] www.xilinx.com/products/silicon-devices/soc/zynq-7000/zynq-101.html

[48] **Santarini, M.,** Third Quarter 2010. Xilinx redefines state of the art with new 7 series FPGAs, Xcell Journal, 6-11.

[49] Xilinx Inc., November 2012. AXI reference guide, UG761, v14.3, https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug761_axi_reference_guide.pdf.

[50] Xilinx, Inc, https://www.xilinx.com/content/dam/xilinx/imgs/block-diagrams/zynq-mp-core-dual.png

[51] Xilinx, Inc., AXI reference guide, UG761, v13.1, March 2011, https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf.

[52] **Fons, F., Fons, M.,** Third Quarter 2010. Making biometrics the killer app for FPGA dynamic partial reconfiguration, Xcell Journal, 24-31.

[53] **Velez, G.,** March 2014. A reconfigurable embedded vision system for advanced driver assistance, Journal of Real Time Image Processing, Volume 10(4), 725-739.

[54] **Santarini, M.,** Second Quarter 2014. Xilinx's new sdnet environment enables 'softly' defined networks, Xcell Journal, 8-13.

[55] Xilinx Inc., Defense Grade Zynq-7000Q AP SoCs, www.xilinx.com/products/silicon-devices/soc/zynq-7000q.html

[56] **Santarini, M.,** Fourth Quarter 2009. Xilinx FPGAs to power next-generation networked battlefield, Xcell Journal, 8-14.

[57] U. S. Energy Information Administration, Electricity use by machine drives varies significantly by manufacturing industry, 2013.

[58] **Lin, Y.,** October 2012. Using xilinx devices to solve challenges in industrial applications, Xilinx White Paper, WP410, v2.0.

[59] **Khan, K.,** April 2012. FPGAs help drive innovation in complex medical systems, Medical Electronics Design.

[60] **Sundararajan, P,** September 2010. High performance computing using FPGAs, Xilinx White Paper, WP375, v1.0.

[61] **Weston, S.,** First Quarter 2011. FPGAs speed the computation of complex credit derivatives, Xcell Journal, 18-25.

[62] **Hampson, G.,** Second Quarter 2011. Xilinx FPGAs beam up next-gen radio astronomy, Xcell Journal, 30.

[63] Xilinx, Inc, May 2014. Vivado Design Suite User Guide: High-Level Synthesis, UG902, v2014.1, http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_2/ug902-vivado-high-level-synthesis.pdf

[64] Xilinx Inc, April 2014. Vivado design suite user guide: model based DSP design using system generator, UG897, v2014.1, www.xilinx.com/support/documentation/sw_manuals/xilinx2014_2/ug897-vivado-sysgen-user.pdf

[65] ARM, Real-Time Operating Systems (RTOS), community.arm.com/docs/DOC-2764.

[66] IBM Developer Works, October 2007. Anatomy of the Linux File System, http://www.ibm.com/developerworks/linux/library/l-linux-filesystem/

[67] **Holt, A., Huang, C. Y.,** 2018. Overview of GNU/Linux, Embedded Operating Systems, Proceedings of Springer, Cham, 11-40

[68] **Corbet, J., Hartman, G. K., Rubini, A.,** 2005. An introduction to device drivers in Linux device drivers, 3rd Edition, O'Reilly, 1-14.

[69] Xilinx, Inc, February 2014. Zynq-7000 All programmable soc technical reference manual, UG585, v1.7, www.xilinx.com/support/documentation /user_guides /ug821-zynq-7000-swdev.pdf

[70] Digilent Inc, January 2013. Embedded Linux development guide, www.digilentinc.com/Data/Products/EMBEDDED-LINUX/Digilent_Embedded_Linux_Guide.pdf

[71] Xilinx, Inc, February 2014. Zynq-7000 All Programmable SoC Technical Reference Manual, UG585, v1.7, http://www.xilinx.com/support/documentation/user_guides/ug821-zynq-7000-swdev.pdf

[72] Avnet, Inc, 2017. PicoZed™ FMC Carrier Card V2 Hardware User Guide, v1.1, http://zedboard.org/sites/default/files/documentations/5285-UG-PZCC-FMC-V2-V1_1.pdf

[73] Avnet, Inc, 2012. HDMI Input/Output FMC Module with Camera Interface Hardware Guide, https://forums.xilinx.com/xlnx/attachments/xlnx/NewUser/42212/1/AES-FMC-HDMI-CAM-G-FMC_HDMI_CAM_HG_v0_1.pdf

[74] Xilinx, Inc, July 2016. SDSoC Environment User Guide Platforms and Libraries, UG146, v2016.2, https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_2/ug1146-sdsoc-platforms-and-libraries.pdf

[75] Xilinx, Inc, July 2016. SDSoC Environment User Guide, UG1027, v2016.2, https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_2/ug1027-sdsoc-user-guide.pdf

[76] **Jiang, W., Sha, E. H. M., Zhuge, Q., Yang, L., Chen, X., Hu, J.,** 2018. Heterogeneous FPGA-based cost-optimal design for timing-constrained CNNs, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

EKLER

EK-1. Picozed Gömülü Görüntü İşleme Kartı

- Xilinx XC7Z030-1SBG485C AP SOC birimi
- Çoklu yapılandırma kipi
  - QSPI flash,
  - eMMC,
  - JTAG,
  - Mikro SD kart ile yapılandırılabilir
- 1 GB DDR3 (x32) bellek
- 128 Mb QSPI flash
- 4 GB eMMC bellek
- 10/100/1000 ethernet ara yüzü
- USB 2.0 OTG PHY
- Yerleşik osilatör - 33.333 MHZ
- FMC LPC konektör (72 diferansiyel, 3 tek uçlu)
- JTAG arabirim konektörü
- Tek JTAG konektörü, otomatik anahtarlama devresi aracılığıyla Zynq ve FMC kartına erişim sağlar.
- Mikro SD kart yuvası
- HDMI 1080p çıkışı
- PCIe x 1 Gen2 arabirimi
- SMA konektör
- SMA-TX P / N
- SMA-RX P / N:
- SFP ara yüzü
- I2C SWE programlanabilir saat sentezleyicisi ile I2C yapılandırmasına sahip EEPROM
- USB UART - Mikro USB konektörü ve alıcı-verici
- MAC ID I2C EEPROM
- MAC ID UNI / O tek telli EEPROM (sadece SFP + MAC ID - 7015/30 SOM).

- I2C gerçek zaman saati (RTC)
- RTC için CR1025 pil tutucu
- 1 PS kullanıcı basma düğmesi
- 5 PL Kullanıcı butonları
- Yapılandırma butonları (CARRIER_RST_N, PG_MODULE_N)
- PL kırmızı kullanıcı LED'leri
- 2 PS kırmızı kullanıcı LED'i
- Durum LED'leri (VIN_HDR, FPGA DONE & PG_MODULE)

EK-2. Xilinx XC7Z030-1SBG485C AP SOC Modül

- İşlemci (PS-Processing System)
  - Çift çekirdek ARM Cortex-A9 Temelli Uygulama İşlemci Birimi(APU-Application Processor Unit)
  - 1 GHz' e kadar CPU frekansı
  - ARMv7-A mimari
  - Thumb-2 komut seti
  - NEON medya işleme motoru
  - Kayan nokta birimi
  - Zamanlayıcı (timer) ve kesmeler (interrupt)
  - ROM
  - 256 KB RAM (OCM)
  - Çoklu Dinamik Hafıza Yönetimi
  - 16-bit veya 32-bit DDR3, DDR3L, DDR2, veya LPDDR2 bellek arayüzleri
  - 1GB bellek
  - Statik bellek arayüzü
  - 1-bit SPI, 2-bit SPI, 4-bit SPI (quad-SPI), ya da quad-SPI (8-bit)
  - 8-kanal DMA kontrolörü
  - 2x 10/100/1000 Ethernet MAC IEEE 802.3 ve IEEE 1588
  - 2x USB 2.0 OTG
  - 8-bit ULPI PHY
  - 2xCAN 2.0
  - 2xSD/SDIO 2.0/MMC3.31

- 2x UARTs (en çok 1 Mb/s)
- 2x I2C
- ARM AMBA® AXI ara yüzü

- Programlanabilir Lojik
  - 125K lojik hücre
  - 78600 LUT
  - 157200 flip-flops
  - 265x36 Kb(9.3 Mb) blok RAM
  - 400 DSP Blok
  - 1.2V -3.3V I/O
  - Gen2 PCI express
  - 2x 12-bit analog-dijital dönüştürücü

ÖZGEÇMİŞ

Recep ÖZALP, 1991 yılında Gaziantep'te doğdu. Orta öğrenimini İ.M.K.B. Anadolu Lisesi'nde tamamladı. 2012 yılında girdiği Fırat Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği Bölümü'nden 2016 yılında mezun oldu. 2016 yılında girdiği Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Mekatronik Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine başladı.