RADAR SİNYAL İŞLEME ALGORİTMALARININ FPGA VE GPU ÜZERİNDE UYGULANMASININ BAŞARIM ANALİZİ

MUHAMMET ÖZGÜR

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ARALIK 2014

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Osman EROĞUL

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Muhammet ÖZGÜR tarafından hazırlanan RADAR SİNYAL İŞLEME ALGORİTMALARININ FPGA VE GPU ÜZERİNDE UYGULANMASININ BAŞARIM ANALİZİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Oğuz ERGİN

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Yrd. Doç. Dr. Murat ÖZBAYOĞLU _____

Üye : Doç. Dr. Oğuz ERGİN _____

Üye : Doç. Dr. Ali BOZBEY _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

.....

Muhammet ÖZGÜR

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Doç. Dr. Oğuz ERGİN**
Tez Türü ve Tarihi : **Yüksek Lisans – Aralık 2014**

Muhammet ÖZGÜR

RADAR SİNYAL İŞLEME ALGORİTMALARININ FPGA VE GPU ÜZERİNDE UYGULANMASININ BAŞARIM ANALİZİ

ÖZET

Gerçek zamanlı ve paralel işlem gerektiren bir çok sivil ve askeri uygulama FPGA üzerinde gerçekleştirilmektedir. FPGA, radar uygulamalarında gerçek zamanlı verinin yüksek hızlarda alınıp işlenmesi için kullanılmaktadır. Ayrıca FPGA donanım üzerinde tekrar tekrar programlanabilmesi geliştirme esnasında büyük bir avantajdır.

GPU'lar, grafik işlemleri yapmanın yanı sıra veri paralelliğinin yoğun olduğu işlemlerde kullanılmaya başlanmıştır. Bir çok uygulama grafik işlemciler sayesinde hızlandırılmış ve paralel işlemlerin yoğun olduğu uygulamalarda genel amaçlı işlemcilere karşı üstünlük sağlamışlardır.

Bu tez kapsamında, radar sinyal işleme algoritmaları, sayısal aşağı indirgeme, darbe sıkıştırma, doppler işleme ve sayısal hüzmeleme FPGA ve GPU üzerinde uygulanıp karşılaştırılmıştır. Bu algoritmalar FPGA için Xilinx System Generator kullanılarak, GPU için OpenCL ile tasarlanmıştır. Tasarlanmış olan benzetim modelinde sinyal varış yönü, sinyalin menzili ve doppler frekansı kestirilmektedir.

Sonuçlar gerçek zamanlı veri işleme konusunda FPGA'nın GPU'lara göre avantajlı olduğunu göstermektedir. Fakat varış yönü tahmini için kullanılan tasarımda anten sayısının ve bakılacak açı sayısının artması, tasarımda çarpıcı sayısının artmasına dolayısıyla tasarımın bir FPGA üzerine sığmamasına neden olmaktadır.

Anahtar Kelimeler: FPGA, GPGPU, radar sinyal işleme, sayısal hüzmeleme

University : **TOBB Economics and Technology University**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Associate Professor Dr. Oğuz ERGİN**
Degree Awarded and Date : **M.Sc. – December 2014**

Muhammet ÖZGÜR

PERFORMANCE ANALYSIS OF IMPLEMENTATION OF RADAR SIGNAL PROCESSING ALGORITHMS ON FPGA AND GPU

ABSTRACT

Many civil and military application which requires real time and parallel processing is implemented on FPGA. FPGAs are used to process and transfer high speed real time data in radar applications. Besides, it is a big advantage to program FPGA again and again while it is on hardware.

Users started to use GPUs in data parallel applications besides doing graphical processing. Many application are accelerated by running on GPUs and GPUs give better performance results than CPUs in applications that requires parallel processing.

In this thesis, radar signal processing algorithms; digital down conversion, pulse compression, doppler processing and digital beam forming are implemented on both FPGA and GPU. And the results are compared. These algorithms are implemented on Xilinx System Generator for FPGA and OpenCL for GPU. In the designed simulation model, the direction of arrival, range and the doppler frequency of the simulated target can be detected.

The results show that while processing real time data, FPGA has advantage over GPU. On the other hand, in the design used for detecting the direction of arrival, increasing number of antenna and look angle cause to increase the number of multiplication unit. Thus the total design cannot be fit on a single FPGA.

Keywords: FPGA, GPU, radar signal processing, digital beamforming

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Doç. Dr. Oğuz ERGİN'e yine kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine, dostluk ve yardımseverlikleriyle gönlümde ayrı bir yeri olan Kasırga laboratuvarında çalışan yüksek lisans ve lisans öğrencilerine, tez konusu ile ilgili çalışmalarımda direk katkısı olan Hasan HASSAN'a, RST Uzaktan Algılama A.Ş'deki iş arkadaşlarıma, her zaman yanımda olan aileme, her konuda beni her zaman destekleyen ve yanımda olan canım eşim Elif'e teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

ÖZET.....iv

ABSTRACT.....v

TEŞEKKÜR.....vi

İÇİNDEKİLER.....vii

ÇİZELGELERİN LİSTESİ.....viii

ŞEKİLLERİN LİSTESİ.....ix

KISALTMALAR.....xi

1. GİRİŞ.....1

2. TEMEL BİLGİLER.....3

2.1. FPGA.....3

2.1.1. Xilinx System Generator.....7

2.1. GPU.....8

3. RADAR SİNYAL İŞLEME ALGORİTMALARININ UYGULANMASI.....11

3.1. Radar Hakkında Genel Bilgiler.....11

3.2. Sayısal Taban Banda İndirgeme (Digital Down Conversion).....12

3.3. Darbe Sıkıştırma (Pulse Compression).....20

3.4. Doppler İşleme (Doppler Processing).....21

3.5. Sayısal Hüzmeleme (Digital Beam Forming).....27

4. FPGA VE GPU SONUÇLARININ KARŞILAŞTIRILMASI.....33

5. SONUÇLAR.....41

KAYNAKLAR.....42

ÖZGEÇMİŞ.....43

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 2.1. Çeşitli Xilinx FPGA ailelerinin kaynak miktarları.....4

Çizelge 2.2. Xilinx 7 serisi FPGA'lerde DSP slice sayısı.....5

Çizelge 4.1. 10 Menzil hücresi hesaplaması için FPGA kaynak kullanımı 34

Çizelge 4.2. 20 Menzil hücresi hesaplaması için FPGA kaynak kullanımı 34

Çizelge 4.3. 30 Menzil hücresi hesaplaması için FPGA kaynak kullanımı 34

Çizelge 4.4. GPU'da doğrultu vektörü ile çarpma işlemi süreleri.....38

Çizelge 4.5. GPU'da doppler işleme algoritması hesaplama süreleri.....38

Çizelge 4.6. GPU'da harcanan toplam süreler.....39

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 2.1. FPGA mantık birimleri ve programlanabilir bağlantılar.....3

Şekil 2.2. Genel FPGA mimarisi.....5

Şekil 2.3. Simulink model kütüphanesi.....7

Şekil 2.4. CUDA kod örneği 8

Şekil 2.5. OpenCL bellek hiyerarşisi.....9

Şekil 2.6. OpenCL kod örneği.....10

Şekil 3.1. RADAR çalışmasının gösterimi.....11

Şekil 3.2. Gerçek sinyal simulink modeli.....13

Şekil 3.3. Zaman ekseninde gerçek sinyal görüntüsü.....13

Şekil 3.4. Frekans ekseninde gerçek sinyal görüntüsü.....14

Şekil 3.5. Karmaşık sinyalin simulink modeli.....14

Şekil 3.6. Zaman ekseninde karmaşık sinyal görüntüsü.....15

Şekil 3.7. Frekans ekseninde karmaşık sinyal görüntüsü.....15

Şekil 3.8. Sayısal taban banda indirgeme tasarımı.....16

Şekil 3.9. Sayısal taban banda indirgeme çarpını ve filtre tasarımı 16

Şekil 3.10. Ara frekanstaki sinyalin frekans eksenindeki görüntüsü.....17

Şekil 3.11. Karmaşık sinyal ile çarpım sonucu.....18

Şekil 3.12. FDATool.....18

Şekil 3.13. Taban banttaki sinyalin frekans eksenindeki görüntüsü 19

Şekil 3.14. Ara frekanstaki ve taban banttaki sinyallerin zaman eksenindeki görüntüsü.....19

Şekil 3.15. Taban banda indirme ve darbe sıkıştırma algoritmalarının tasarımı.....20

Şekil 3.16. Darbe sıkıştırma sonucu oluşan sinyal.....21

Şekil 3.17. Hızlı zaman, yavaş zaman matrisi.....22

Şekil 3.18. Doppler işleme tasarımı.....23

Şekil 3.19. Doppler Frekansı eklenmiş hedefin uyumlu filtre çıktısı.....24

Şekil 3.20. Hedef doppler frekansı.....25

Şekil 3.21. Gürültü eklenmiş sinyalin DDC algoritmasındaki çıktısı.....25

Şekil 3.22. Gürültü eklenmiş sinyalin uyumlu filtre çıktısı.....26

Şekil 3.23. Gürültü altında hedef doppler frekansı.....27

Şekil 3.26. Antenlerin ve hedefin bulunduğu uzay.....29

Şekil 3.27. Sayısal hüzmeleme tasarımı.....30

Şekil 3.28. Doğrultu vektörüyle çarpılmadan önce darbe sıkıştırma sonucu.....31

Şekil 3.29. Doğrultu vektörüyle çarpıldıktan sonra darbe sıkıştırma sonucu.....31

Şekil 4.1. Kullanılan Mantık Birimi Sayısı.....35

Şekil 4.2. Kullanılan DSP Slice Sayısı.....36

Şekil 4.3. OpenCL doğrultu vektörü ile çarpma kernel kodu.....37

Şekil 4.4. GPU'da İşlem Süreleri.....40

KISALTMALAR

Kısaltmalar	Açıklama
RADAR	Radio detection and ranging
FPGA	Field programmable gate array
GPU	Graphics processing unit
OpenCL	Open computing language
ASIC	Application specific integrated circuit
FFT	Fast Fourier Transform
Verilog HDL	Verilog hardware description language
VHDL	Very high speed integrated circuits hardware description language
I	Inphase
Q	Quadrature
DDS	Direct digital synthesizer
DSP	Digital signal processing

1. GİRİŞ

Radar temel olarak uzaktan nesne algılama sistemidir. Nesneleri algılamada elektromanyetik dalgalar ve bu dalgaların nesneler üzerinden yansıma özelliği kullanılmaktadır. Radar sistemleri değişik kullanım amaçları için tasarlanmaktadır. Örneğin bir meteoroloji radarı meteorolojik ürünlerden (yağmur, kar, dolu gibi) geri yansıyan elektromanyetik dalgaları algılayarak bu nesneleri bir birinden ayırt edebilmektedir. Başka amaçla tasarlanmış olan savunma radarları ise bulutların, yağmurun arkasındaki uçağı algılayabilmektedirler. Radarların bu özelliklerine göre RF donanımları, yaydıkları elektromanyetik dalgaların frekansı, sinyal işleme gereksinimleri farklılık göstermektedir.

Radar verileri paralel işlemeye uygun veri bağımlılığı olmayan verilerdir. Aynı zamanda gerçek zamanlı olarak işlenmeleri gerekmektedir. Bu amaçla temel radar sinyal işleme algoritmaları incelenmiştir. Paralel işlem yapma özellikleriyle öne çıkan iki donanım, FPGA ve GPU, karşılaştırılmıştır.

Xilinx system generator, Matlab simulink model tabanlı tasarım aracında bir kütüphane olarak bulunmaktadır. FPGA üzerinde çalışacak olan radar sinyal işleme algoritmaları bu kütüphane kullanılarak FPGA'e gömülebilir bir şekilde tasarlanmıştır. GPU üzerinde çalıştırılacak algoritmalar OpenCL ile yazılmıştır.

Radar sinyal işleme algoritmaları üzerine bir çok araştırma vardır. Son yıllarda araştırmalar varış yönü tahmini ve hüzmeleme algoritması üzerine yoğunlaşmıştır. Hüzmeleme yöntemi ile varış yönü tahmini üzerine çeşitli algoritmalar [11]'da uygulanmış ve sonuçları elde edilmiştir. [12]'de darbe-doppler radarının yüksek işlem gücü gerektirdiğinden bahsedilmiş ve bunun için sayısal sinyal işlemciler ile çok çekirdekli paralel işlem birimi kurulmuştur. Deneysel sonuçların paralellik verimliliğinin sıralı işlem yapan işlemcilere göre %90 civarında olduğunu gösterdiği söylenmektedir. [13]'de sayısal hüzmeleme yöntemi FPGA üzerinde FPGA'in yeniden konfigüre edilebilirlik özelliği kullanılarak uygulanmıştır. Bu makalede yapılan uygulamada hedef takibi yapılmıştır. [14]'de uydu uygulamaları için analog ve sayısal hüzmeleme yöntemleri karşılaştırılmıştır. Karşılaştırma 1x2'lik bir prototip donanım üzerinde yapılmıştır.

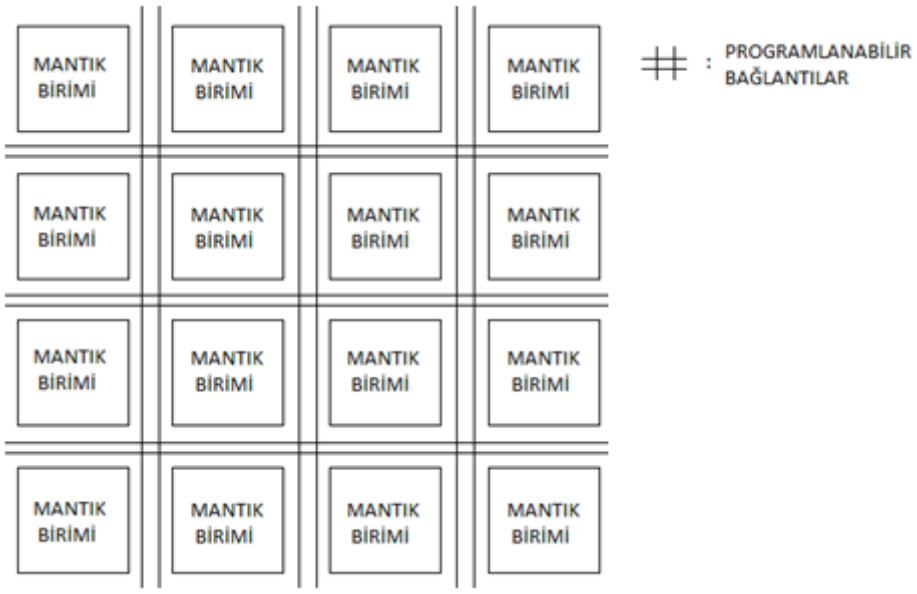
Tezin 2. bölümünde FPGA, Xilinx system generator ve GPU hakkında temel bilgilerden bahsedilmiştir. 3. bölümde radar hakkında genel bilgilerden bahsedildikten sonra uygulanmış olan radar sinyal işleme algoritmalarından bahsedilmiştir. Bu algoritmalar; sayısal taban banda indirgeme, darbe sıkıştırma, doppler işleme ve sayısal hüzmelemedir. 4. bölümde FPGA ve GPU üzerinde uygulanan algoritmaların karşılaştırılması yapılmıştır. 5. bölümde geçmiş yapılmış olan çalışmalardan bahsedilmiştir. Tez 6. bölüm olan sonuçlar kısmı ile sonlandırılmıştır.

2. TEMEL BİLGİLER

Bu bölümde tez kapsamında karşılaştırması yapılacak olan FPGA ve GPU hakkında genel bilgiler verilmiştir.

2.1 FPGA

Alanda Programlanabilir Kapı Dizileri (FPGA), programlanabilir mantık birimleri ve bu birimlerin bağlantılarını içeren sayısal tümleşik devrelerdir. Sivil ve askeri olmak üzere bir çok kullanım alanı vardır. Gerçek zamanlı paralel işlem ihtiyacı olan uygulamalarda uygulamaya özel tümleşik devreler (ASIC) veya FPGA kullanılmaktadır. FPGA elektronik bir devre içerisinde ASIC ile aynı özellikleri göstermektedir. Fakat FPGA programlandıktan sonra başka bir uygulama için tekrar programlanarak içeriği değiştirilebilmektedir. FPGA üzerinde çalışacak bir uygulama tasarlamak için donanım tanımlama dilleri olan Verilog HDL veya VHDL kullanılmaktadır.



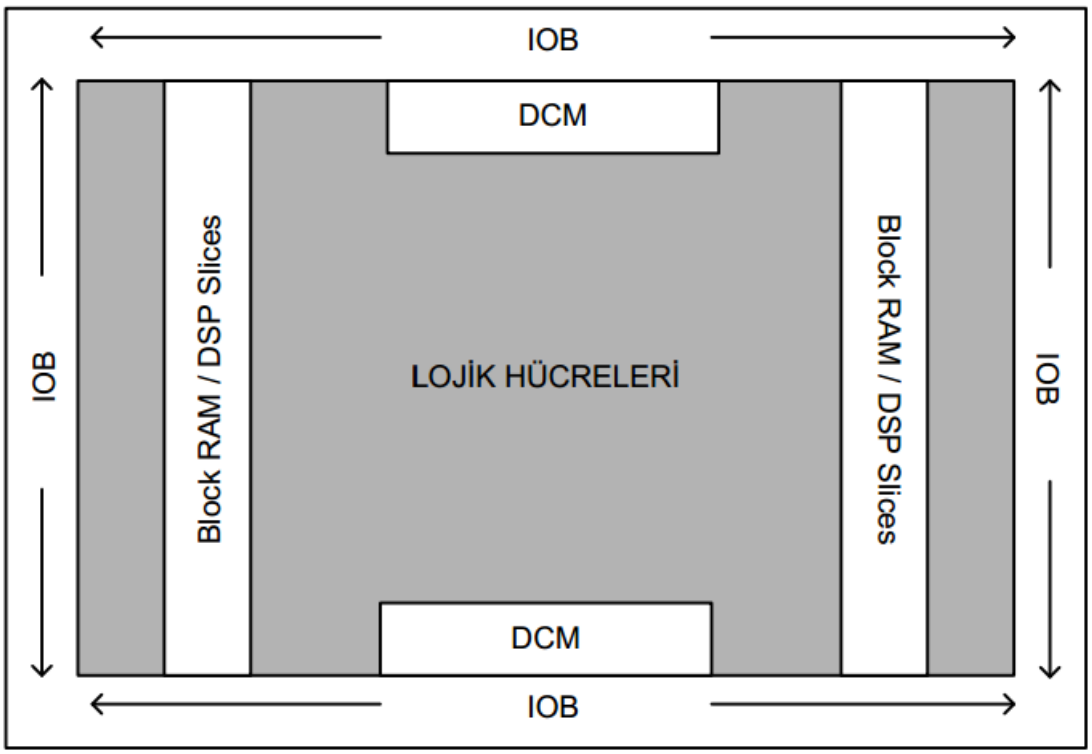
Şekil 2.1. FPGA mantık birimleri ve programlanabilir bağlantılar

FPGA içerisindeki kaynakların miktarı ve dağılımı değişiklik göstermektedir. Xilinx firmasının bazı FPGA'lerinin kaynak miktarı Çizelge 2.1'de gösterilmiştir.

Çizelge 2.1. Çeşitli Xilinx FPGA ailelerinin kaynak miktarları

FPGA Ailesi / Kaynaklar	**Logic Slice**	**Block RAM (kb)**	**DSP Slice**	**User I/O**
Spartan-3A XC3S200A	4032	288	16	248
Spartan-3A XC3S1400A	25344	576	32	502
Virtex-5 XC5VLX50	9600	1152	32	400
Virtex-5 XC5VLX330	103680	10368	192	1200
Virtex-7 XC7V585T	182100	28620	1260	850
Virtex-7 XC7V2000T	305400	46512	2160	1200

FPGA'de programlanabilir mantık birimleri temel işlemleri yapacak donanımı içermektedir. Mantık birimi içerisinde temel olarak LUT, yazmaç ve çoklayıcılar bulunmaktadır. Ayrıca mantık birimleri etrafında özel işler yapan bellek birimi (RAM), sayısal saat yönetici (DCM), çarpma gibi işlemler yapan DSP slice birimleri bulunmaktadır. Tüm birimlerin etrafında ise giriş çıkış blokları bulunmaktadır. Genel olarak FPGA mimarisi Şekil 2.2'de görülmektedir.



Şekil 2.2. Genel FPGA mimarisi

Radar sinyal işleme algoritmalarında kullanılacak çarpıcı miktarı kaynak kullanımında belirleyici faktör olmaktadır. Çarpım işlemleri FPGA içerisinde bu işlem için özelleşmiş DSP slice birimleri ile yapılmaktadır. Çizelge **2.2**'de Xilinx 7 serisi FPGA'lerde DSP slice miktarları gösterilmiştir[10].

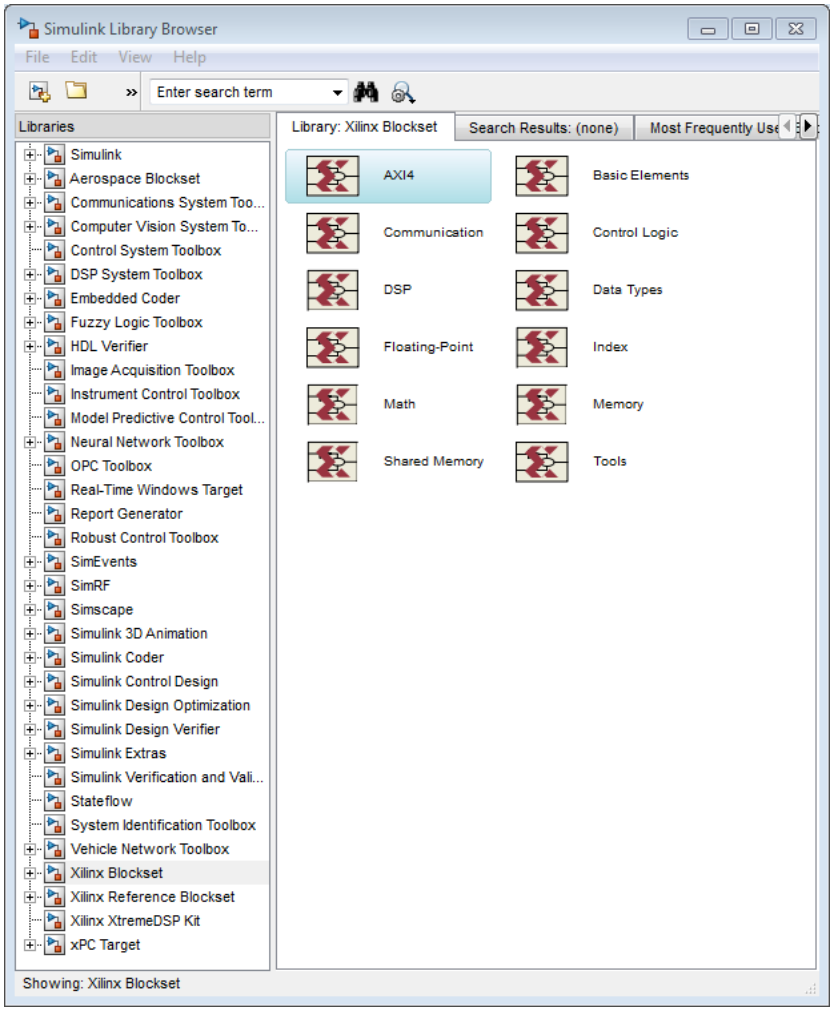
Çizelge 2.2. Xilinx 7 serisi FPGA'lerde DSP slice sayısı

7 serisi FPGA	**Toplam DSP Slice Sayısı**
7A15T	45
7A35T	90
7A50T	120
7A75T	180
7A100T	240

7A200T	740
7K70T	240
7K160T	600
7K325T	840
7K355T	1440
7K410T	1540
7K420T	1680
7K480T	1920
7V585T	1260
7V2000T	2160
7VX330T	1120
7VX415T	2160
7VX485T	2800
7VX550T	2880
7VX690T	3600
7VX980T	3600
7VX1140T	3360
7VH580T	1680
7VH870T	2520

2.1.1 Xilinx System Generator

Simulink, MATLAB'ın model tabanlı tasarım geliştirme ortamıdır. Model tabanlı tasarım, görsel avantajları sayesinde tasarım zamanını iyileştirmektedir. Simulink ortamında MATLAB tarafından sağlanan temel bir model kütüphane ile geliştirme yapılabilmektedir.



Şekil 2.3. Simulink model kütüphanesi

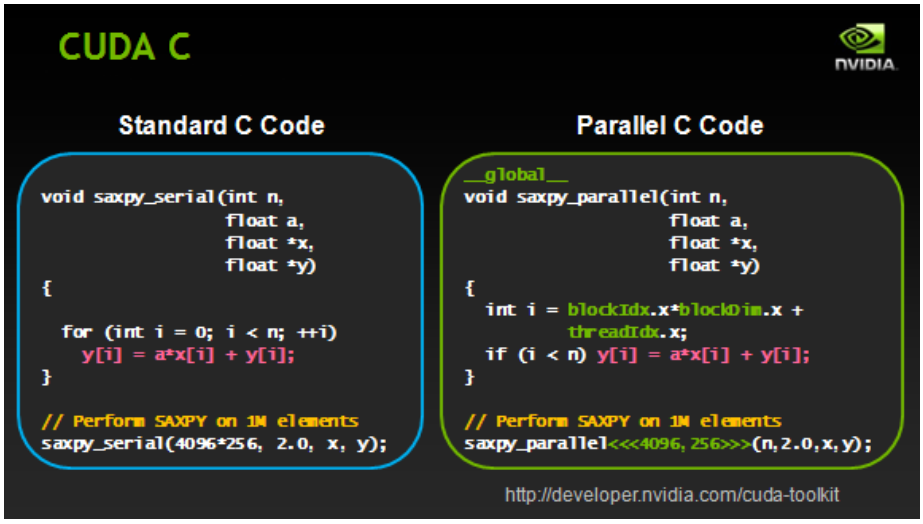
Xilinx System Generator Şekil 2.3'de görüldüğü üzere bir Simulink kütüphanesidir. Xilinx firmasının sağladığı bu eklenti sayesinde FPGA üzerinde çalışabilen modeller ile tasarım yapılabilmektedir. Normalde FPGA üzerinde çalıştırılacak olan tasarım donanım tanımlama dilleri olan VHDL veya Verilog HDL ile tasarlanmaktadır. Donanım tanımlama dilleri ile sayısal sinyal işleme algoritma tasarımının yapılması test edilmesi ve doğrulanması oldukça uzun süreler almaktadır. FPGA üzerinde çalıştırılacak olan sayısal sinyal işleme algoritmalarının model tabanlı bir şekilde geliştirilmesi harcanan süreyi kısaltmaktadır.

2.2 GPU

GPU teknolojisindeki ilerlemelerle birlikte, günümüzde kullanılan modern GPU'lar programlanabilir ara yüzler sunar hale gelmişlerdir. Bu programlanabilir ara yüzler sayesinde GPU'nun işlem gücü ve paralel işleyebilme yeteneği sadece grafik işlemlerinde değil aynı zamanda genel amaçlı hesaplamalarda da kullanılabilir hale gelmiştir[2].

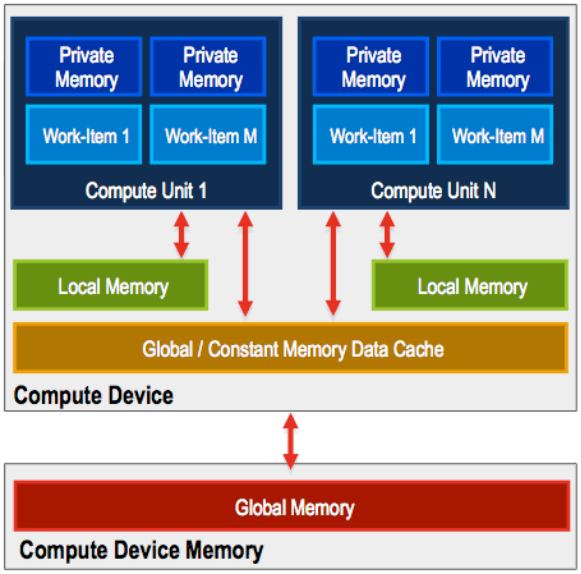
GPU'yu genel amaçlı hesaplamalarda kullanan uygulamalar GPU'ların grafik aygıtlarına özel olan köşe kenar dönüşümü, dokulandırma, renklendirme, gölgelendirme vb. özelliklerinden ziyade SIMD şeklinde çalışan işlem hattı mimarisinden yararlanırlar. Bu uygulamalar genel olarak; işaret işleme, ses işleme, görüntü işleme, şifreleme, yapay sinir ağları, bioinformatik, paralelleştirilebilen bilimsel hesaplamalar, istatistiksel hesaplamalar gibi yüklü miktarda verinin küçük parçaları üzerinde bağımsız ve paralel olarak işlem yapılmasına uygun olan uygulama alanlarında başarılıdırlar[4].

GPU'ları genel amaçlı programlamak için en çok kullanılan diller OpenCL ve CUDA'dır. CUDA, NVIDIA tarafından geliştirilmiş olup temel olarak C programlama dili üzerine bir eklenti olarak sunulmuştur. CUDA sadece NVIDIA GPU kartları üzerinde programlama yapılabilmesine olanak sağlamaktadır. Şekil 2.4'de örnek bir CUDA C kodu görülmektedir[3].



Şekil 2.4. CUDA kod örneği

Khronos grup, paralel hesaplama, grafik, bilgisayarla görme vb. algoritmaların çeşitli platformlar ve cihazlar üzerinde çalışması için açık standartlar oluşturan, kar amacı gütmeyen teknoloji şirketleri birliğidir. OpenCL bu grubun çıkardığı bir standarttır. OpenCL, ARM, Apple, NVIDIA gibi büyük şirketler tarafından desteklenmektedir. OpenCL ile hazırlanan kodlar ev sahibi (host) ve araç (device) olmak üzere iki farklı kısımdan oluşmaktadır. Ev sahibi kod kısmı araç üzerinde çalışacak kernelin hazırlanmasını ve düzenlenmesini içermektedir. OpenCL yazılım hiyerarşisinde GPU iş elemanlarından oluşmaktadır. Kernel kodu kendisine atanan GPU üzerinde yer alan iş elemanları tarafından çalıştırılır ve her bir iş elemanına kernel host tarafından atanır. GPU içerisindeki bütün iş elemanları aslında aynı kod bloğunu çalıştırır fakat farklı verileri üzerinde işlem yaparlar. İş elemanları bir araya gelerek iş gruplarını meydana getirirler. Sadece aynı iş grubunda yer alan iş parçacıkları (thread) aynı bellek alanını (yerel bellek) paylaşarak birbirleriyle eşzaman (synchronous) bir şekilde çalışabilirler. Farklı iş gruplarının eşzaman olarak çalışabilmesi için global belleği kullanması gerekir, bu da çok büyük bir yavaşlamaya sebep olacağı için tercih edilmez. Dolayısıyla farklı iş grubunda yer alan iş parçacıkları farklı bellek alanlarını kullandıkları için senkronize olarak çalışamazlar. İş gruplarının büyüklükleri kullanıcı tarafından uygulamaya özel olarak başarımı arttıracak şekilde ayarlanabilir[6]. OpenCL bellek hiyerarşisi Şekil 2.5'de gösterilmiştir[7].



Şekil 2.5. OpenCL bellek hiyerarşisi

Geleneksel olarak bir dizinin elemanlarının çarpılması işlemiyle, aynı işlemin paralel olarak OpenCL'de yapıldığı kod parçaları Şekil 2.6'te gösterilmiştir.

```
// Geleneksel Çarpma
void vectorMult(
const float* x,
const float* y,
float* sonuc,
const unsigned int sayi ) {
    for(int i = 0; i < sayi; i++)
        sonuc[i] = x[i] * y[i];
}

// OpenCL ile paralel çarpma
kernel void vectorMult(
global const float* x,
global const float* y,
global float* sonuc ) {
    int id = get_global_id(0);
    sonuc[id] = x[id] * y[id];
}
```

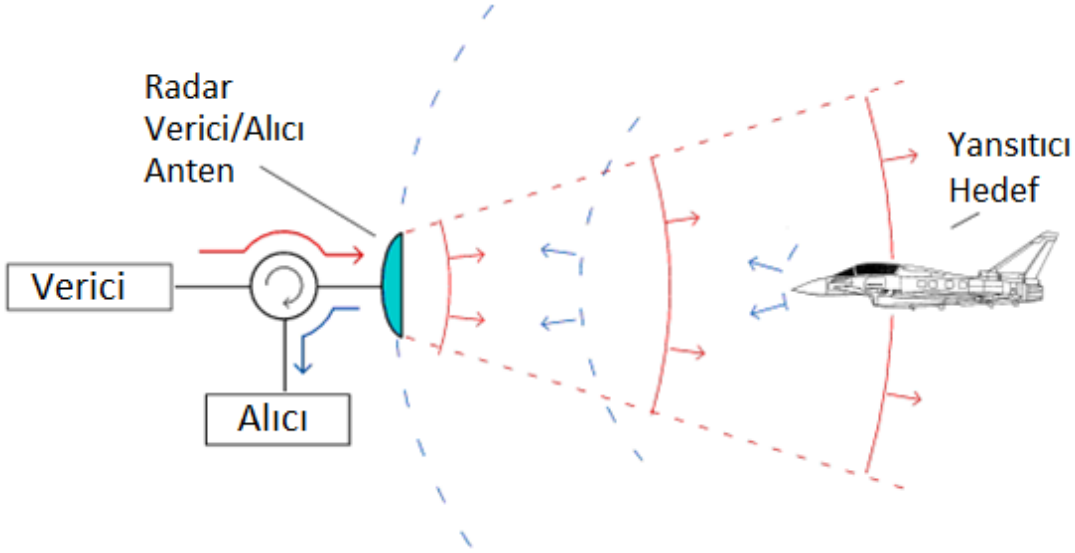
Şekil 2.6. OpenCL kod örneği

3. RADAR SİNYAL İŞLEME ALGORİTMALARININ UYGULANMASI

3.1 Radar Hakkında Genel Bilgiler

Radar kelimesi radyo ile tespit etme ve menzil bulma kelimelerinin İngilizcelerinin kısaltmasıdır. İsminden de anlaşılacağı üzere radar, elektromanyetik radyo dalgalarının hedef üzerinden yansımasını tespit etmektedir. Bu şekilde uzaktan algılama yapılmaktadır. Radar'ların günümüzde bir çok uygulama alanı vardır. Hava savunma sistemleri, füzesavar sistemleri, meteorolojik olayların tespit edilmesi kullanım alanlarına verilebilecek örneklerden bazılarıdır.

Bu tez kapsamında benzetimi yapılacak RADAR sistemi sayısal hüzmeleme yöntemi uygulanan darbe-doppler radarıdır. Bu radar yüksek frekansta darbeler gönderip her hangi bir yansıma olup olmadığını tespit eder. Tespit edilen yansımalardan hedefin menzili, hızı, yönü çıkarılabilmektedir.



Şekil 3.1. RADAR çalışmasının gösterimi[1]

Şekil 3.1'de Radar'ın gönderdiği darbeler kırmızı olarak gösterilmiştir. Hedeften dönen sinyaller mavi ile gösterilmiştir. Mavi olarak gösterilen sinyaller RADAR tarafından algılanıp hedef tespit edilmektedir.

Sayısal hüzmeleme, uzaktan algılama ve kablosuz iletişim gibi uygulamalarda kullanılan bir radar sinyal işleme algoritmasıdır. Sayısal hüzmeleme yöntemi akıllı

anten sistemlerinde kullanılmaktadır. Bu yöntem sayesinde antenler mekanik olarak döndürülmeden istenilen yöne bakılabilmektedir. Bir çok antenden oluşan akıllı anten sistemlerinde sinyal varış yönü tahmini hüzmeleme algoritması kullanılarak yapılabilmektedir.

Radar tarafından yüksek frekansta sinyal gönderilmesi ve alınması antenler ve RF devreler ile yapılmaktadır. Radar tarafından yüksek frekansta alınan sinyal RF devreler aracılığı ile ara frekansa (IF) düşürülmektedir. Ara frekanstaki sinyal analog sayısal dönüştürücü (ADC) devreler ile örneklenmektedir. Bu aşamadan sonra radar sinyal işleme algoritmaları sayısal olarak yapılmaktadır.

3.2 Sayısal Taban Banda İndirgeme (Digital Down Conversion)

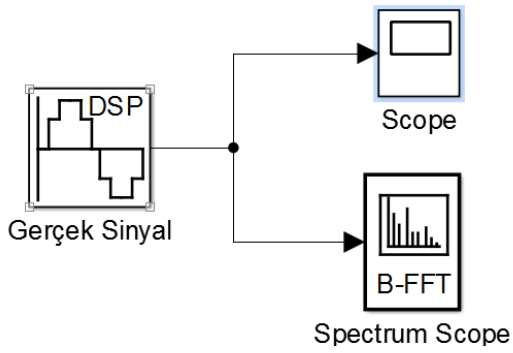
Ara frekanstaki sinyal bant genişliği radar tarafından gönderilen darbe sinyalinde bir modülasyon yapılmamış ise darbe genişliğinin frekans değerine eşittir. Darbe sinyaline modülasyon yapılmış ise bant genişliği yapılan frekans modülasyonun frekansına eşittir. Yapılmış olan tasarımda 50 MHz frekansa sahip olan ara frekanstaki sinyal analog sayısal dönüştürücüler ile 200 MHz 'de örneklenmektedir.

Ara frekanstaki sinyal içerisinde barındırdığı bilgiye göre hala yüksek frekansta bulunmaktadır. Bu nedenle ara frekanstaki sinyal sayısal taban banda indirgeme yöntemi ile taban banda yani 0 frekans etrafına indirilmektedir. Bu yöntem sonucunda I (Inphase) ve Q (Quadrature) sinyalleri elde edilir. Bu sinyaller karmaşık sinyaller olarak ifade edilerek (I - gerçek, Q - sanal kısım) sinyal işleme algoritmalarında karmaşık sinyaller olarak kullanılırlar. I ve Q sinyallerinden hedeften dönen sinyalin fazı ve genliği elde edilebilmektedir.

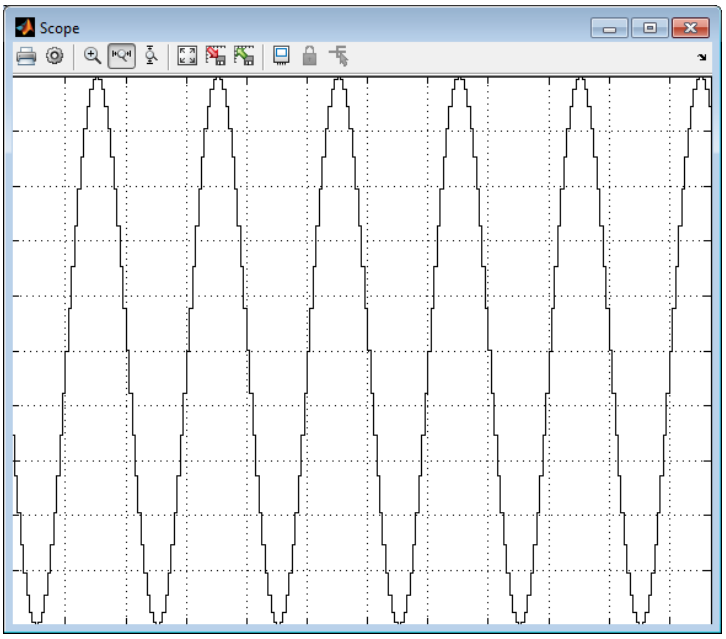
Ara frekanstaki sinyalin taban banda indirilmesi ile ilgili olarak basit bir formül kullanılmaktadır.

$$\cos a \times \cos b = \frac{1}{2}\cos(a+b) + \frac{1}{2}\cos(a-b) \tag{4.1}$$

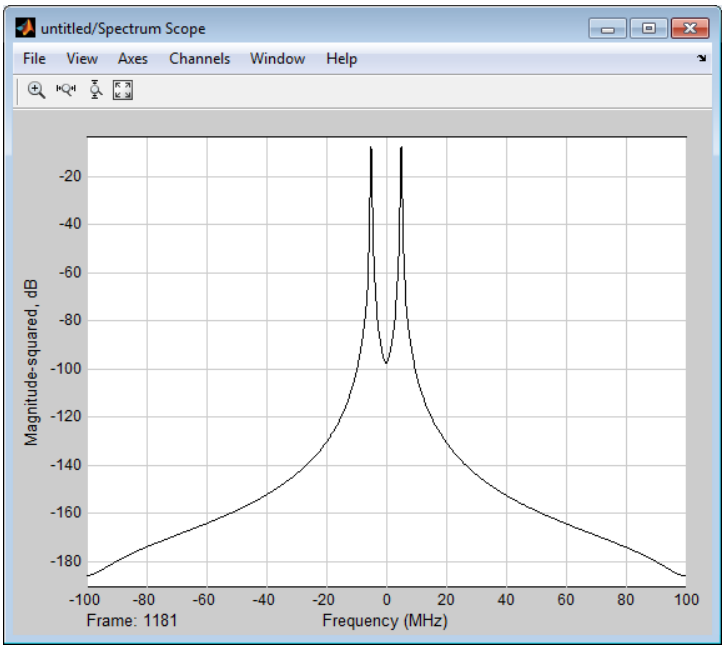
Gerçek sinyaller frekans ekseninde hem pozitif hem negatif frekansı aynı anda temsil ederler. Simulink ile oluşturulan Şekil 3.2'deki model gerçek sinyallerin görüntüsünü zaman ve frekans ekseninde elde etmek amacıyla tasarlanmıştır.



Şekil 3.2. Gerçek sinyal simulink modeli



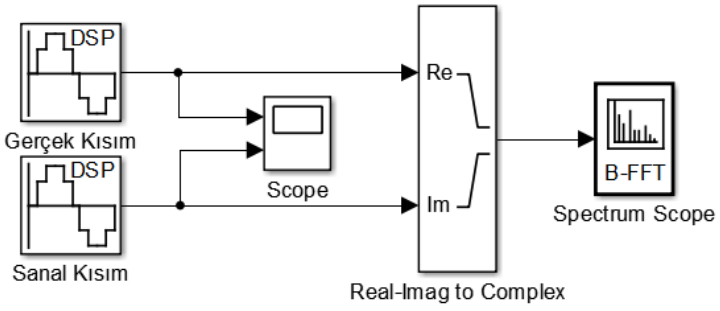
Şekil 3.3. Zaman ekseninde gerçek sinyal görüntüsü



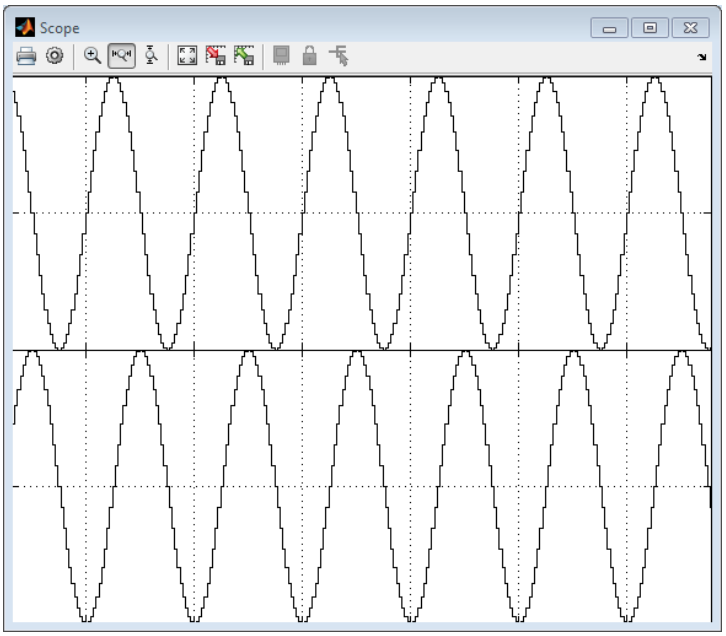
Şekil 3.4. Frekans ekseninde gerçek sinyal görüntüsü

Modelde 200 MHz örnekleme frekansı ile 5 MHz frekansa sahip bir sinyal üretilmiştir. Şekil 3.3'te üretilmiş olan gerçek sinyalin zaman eksenindeki görüntüsü görülmektedir. Şekil 3.4'te görüleceği üzere frekans eksenindeki görüntüsü +5MHz ve -5MHz'de görülmektedir.

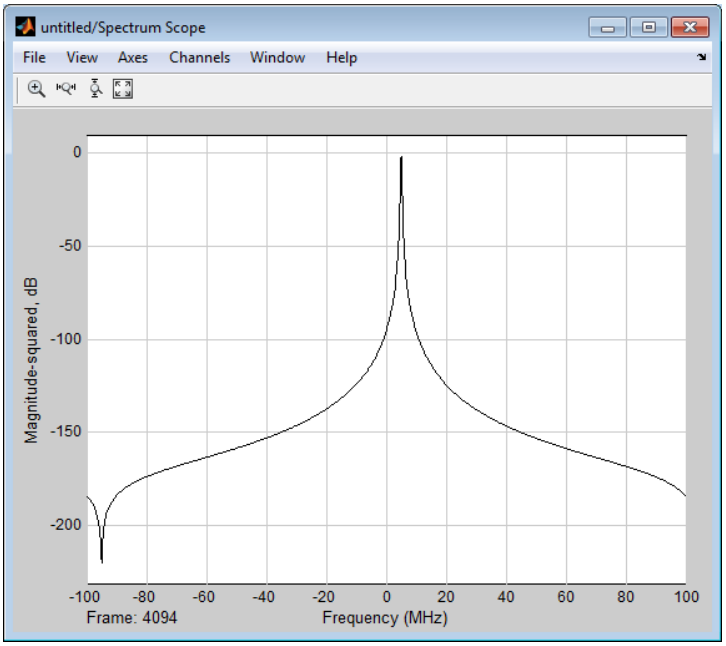
Karmaşık sinyaller aralarında 90° faz farkı olan iki sinyalden oluşmaktadır. Bu sinyallerin frekans uzayındaki karşılıkları sadece pozitif veya sadece negatif frekansı temsil eder. Simulink ile oluşturulan Şekil 3.5'teki model, karmaşık sinyallerin görüntüsünü zaman ve frekans ekseninde elde etmek amacıyla tasarlanmıştır.



Şekil 3.5. Karmaşık sinyalin simulink modeli



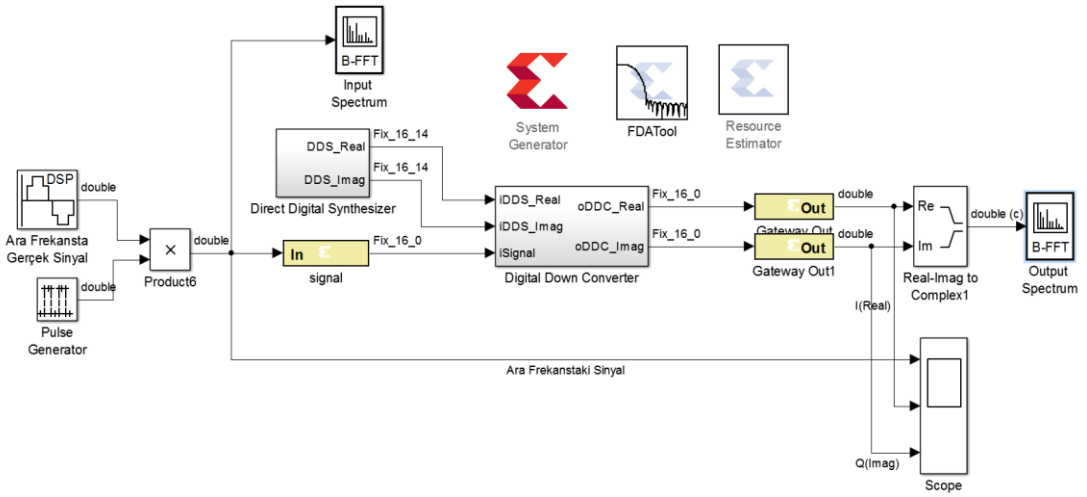
Şekil 3.6. Zaman ekseninde karmaşık sinyal görüntüsü



Şekil 3.7. Frekans ekseninde karmaşık sinyal görüntüsü

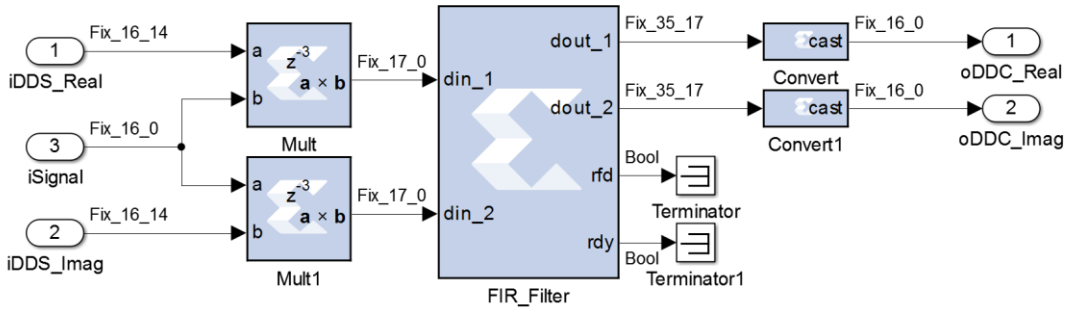
Modelde 200 MHz örnekleme frekansı ile 5 MHz frekansa sahip bir karmaşık sinyal üretilmiştir. Şekil 3.6'da üstteki sinyal karmaşık sinyalin gerçek kısmı, alttaki sinyal ise sanal kısmıdır. Şekil 3.7'te görüleceği üzere frekans eksenindeki görüntüsü +5MHz'de görülmektedir.

Ara frekansta gelen gerçek sinyal, bu frekanstaki karmaşık sinyal ile çarpılarak taban banda indirilmektedir. Şekil 3.8'de görülen tasarım FPGA üzerine gömülebilir olması açısından Xilinx System Generator blokları ile tasarlanmıştır.



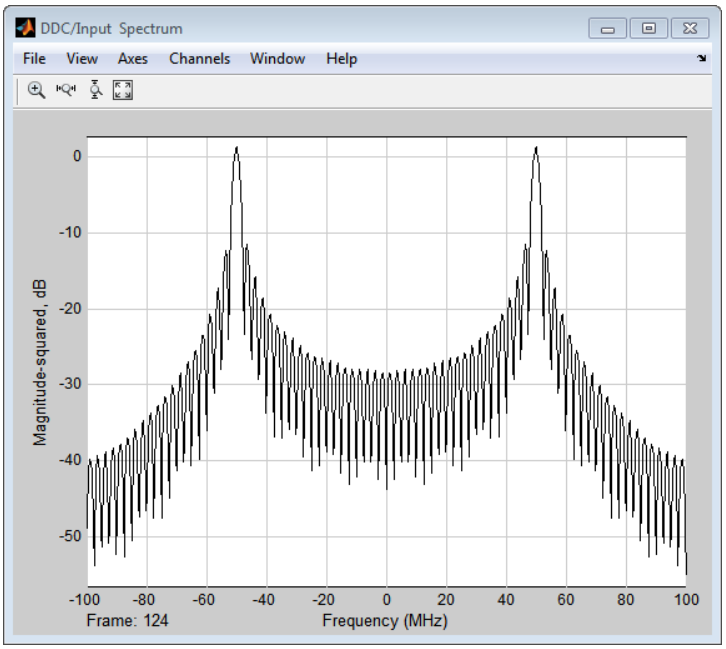
Şekil 3.8. Sayısal taban banda indirgeme tasarımı

Ara frekanstaki sinyali taban banda indirmek üzere aynı frekansta karmaşık sinyal Xilinx DDS Compiler ile üretilmiştir ve bu sinyaller birbirleriyle çarpılmışlardır.



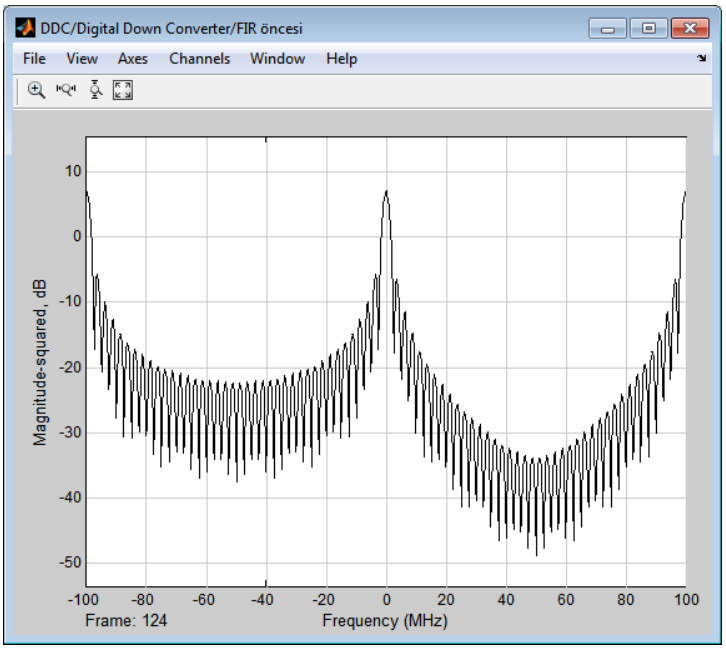
Şekil 3.9. Sayısal taban banda indirgeme çarpını ve filtre tasarımı

Ara frekansta giriş sinyalinin (iSignal) frekansı Şekil 3.10'da görüldüğü üzere 50 MHz'dir.



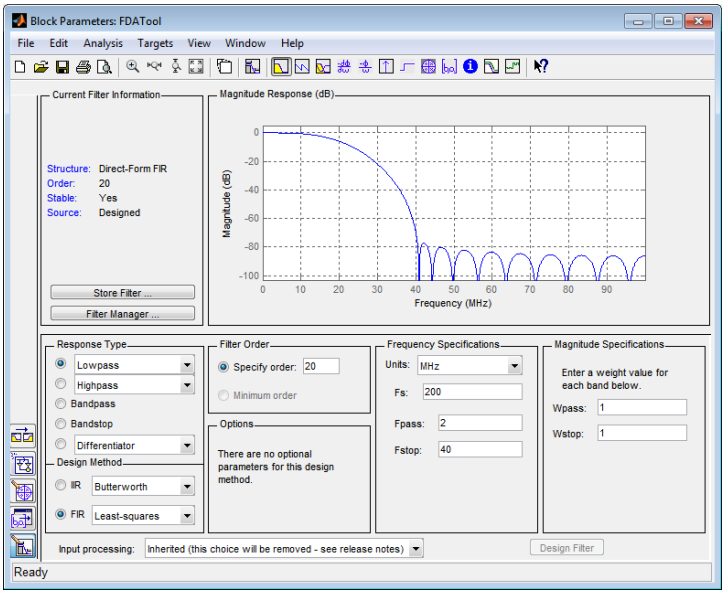
Şekil 3.10. Ara frekanstaki sinyalin frekans eksenindeki görüntüsü

Şekil 3.10'da frekansı gösterilen sinyal -50MHz Xilinx DDS Compiler tarafından üretilen karmaşık sinyal ile çarpılmıştır. Çarpım sonucunda Şekil 3.11'de görülen 0 Hz ve 100 MHz 'de sinyaller oluşmuştur.



Şekil 3.11. Karmaşık sinyal ile çarpım sonucu

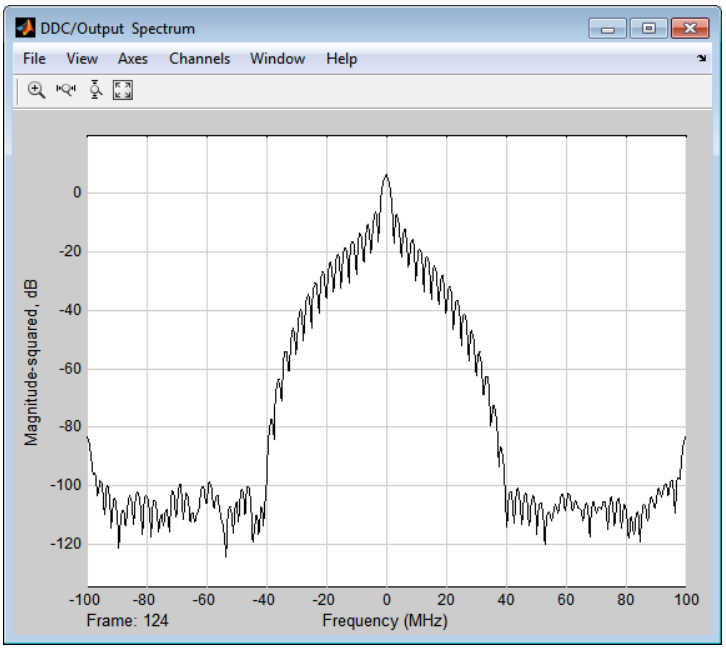
Yüksek frekanslı sinyalin filtrelenmesi için FDATool ile 20 taplık alçak geçirgen filtre tasarımı yapılmıştır. FDATool ara yüzü Şekil 3.12'de görülmektedir.



Şekil 3.12. FDATool

FDATool tarafından üretilen filtre katsayıları ve Xilinx FIR Compiler kullanılarak yüksek frekanslı sinyal filtrelenmiştir. Böylece ara frekanstaki sinyalin taban banda

indirilme işlemi tamamlanmıştır. Taban banttaki sinyalin frekans eksenindeki görüntüsü Şekil 3.13'te görülmektedir.



Şekil 3.13. Taban banttaki sinyalin frekans eksenindeki görüntüsü

Şekil 3.14. Ara frekanstaki ve taban banttaki sinyallerin zaman eksenindeki görüntüsü

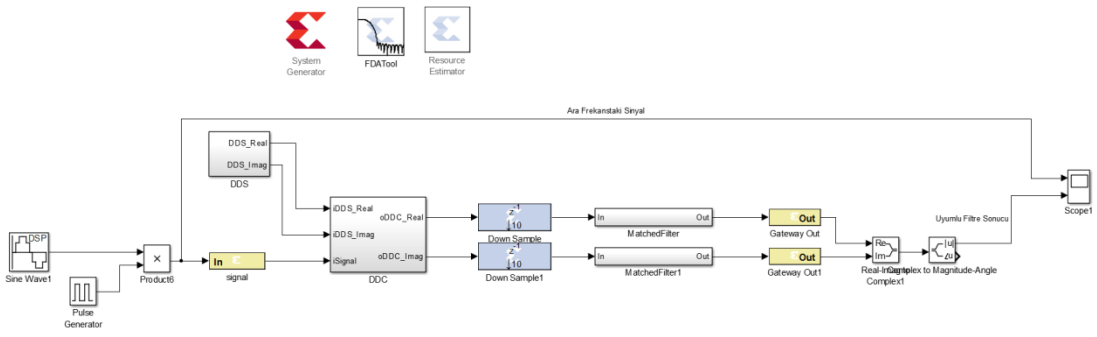
Taban banda indirilmiş sinyalin zaman eksenindeki görüntüsü Şekil 3.14'te görülmektedir. Taban banttaki sinyal üzerinde örnek seyreltme işlemi uygulanarak bu aşamadan sonraki algoritmaların daha düşük frekansta çalışmaları sağlanır.

3.3 Darbe Sıkıştırma (Pulse Compression)

Darbe sıkıştırma işlemi darbe üzerindeki gücün bir noktaya toplanmasını sağlamaktadır. Bu amaçla uyumlu filtreleme (matched filter) işlemi gerçekleştirilmiştir. Bu işlem radarın hedeften dönen sinyalleri dinlemesi esnasında hedef tespitini sinyal gürültü oranını arttırarak yapmasını sağlamaktadır. Bu işlem esnasında Radar tarafından gönderilen sinyal ile hedeften dönen sinyaller çapraz korelasyon işleminden geçirilir.

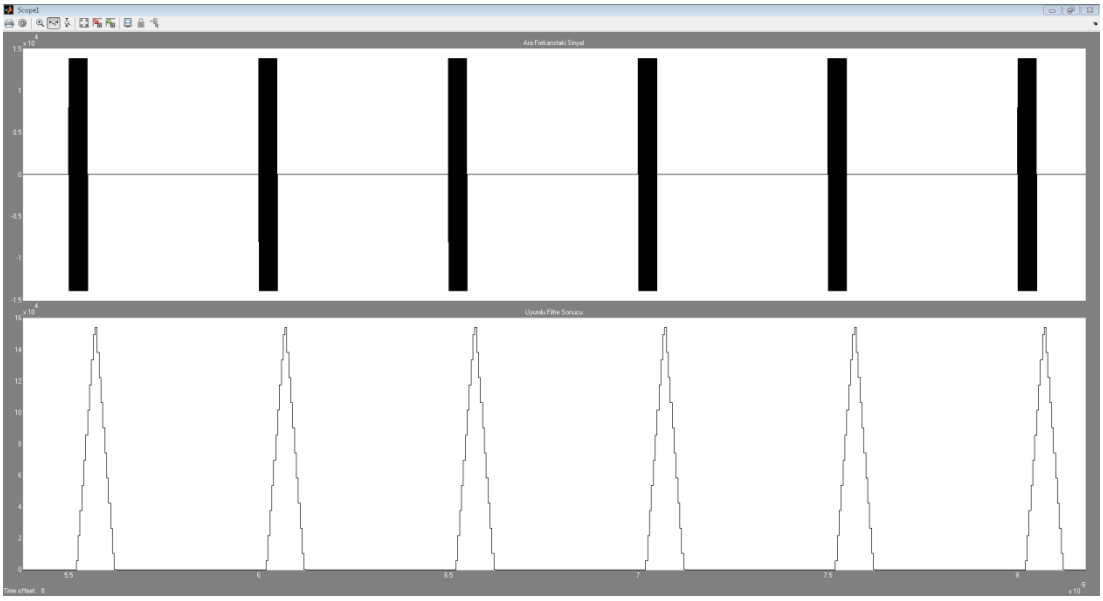
FIR filtre ile çalışma teorisi aynıdır. Fakat buradaki fark radar tarafından gönderilen darbe sinyalinin karmaşık eşleniği FIR filtreye katsayı olarak girilir. Böylelikle radar tarafından gönderilen sinyalin fazı hedeften dönen sinyalden çıkarılmış olur ve sadece hedefin eklemiş olduğu faz sinyal üzerinde kalır. Radarın RF darbe gönderme için kullandığı bazı cihazlarda gönderilen darbelerin fazı kontrol edilebildiği gibi eski teknolojiye sahip magnetron[] gibi bazı cihazlarda gönderilen darbenin fazı kontrol edilememektedir. Magnetron kullanıldığında radar tarafından gönderilen her bir darbe kaydedilerek uyumlu filtre katsayıları her bir darbede değiştirilmektedir.

Sayısal Taban Banda İndirgeme (Digital Down Conversion)bölümünde anlatılmış olan tasarımın I ve Q çıkışı örnek seyreltme işleminden sonra darbe sıkıştırma birimi tarafından işlenir.



Şekil 3.15. Taban banda indirme ve darbe sıkıştırma algoritmalarının tasarımı

Darbe sıkıştırma işlemi sonucunda Şekil 3.16'da alt bölümde görülen sinyal elde edilmektedir. şeklin üst bölümünde görülen sinyal ara frekanstaki sinyaldir. Ara frekanstaki sinyal sayısal aşağı indirgeme ve darbe sıkıştırma algoritmalarından geçirilerek gücünün bir noktaya toplanması sağlanmıştır.



Şekil 3.16. Darbe sıkıştırma sonucu oluşan sinyal

3.4 Doppler İşleme (Doppler Processing)

Radar darbe sinyalleri gönderip hedeften dönen sinyalleri algılamaktadır. Radarın menzil çözünürlüğü gönderdiği darbelerin bant genişliğine bağlıdır.[] Eğer darbeler içerisinde modüle edilmiş bir sinyal yok ise sinyal bant genişliği denklem 4.1'de gösterilmiştir.

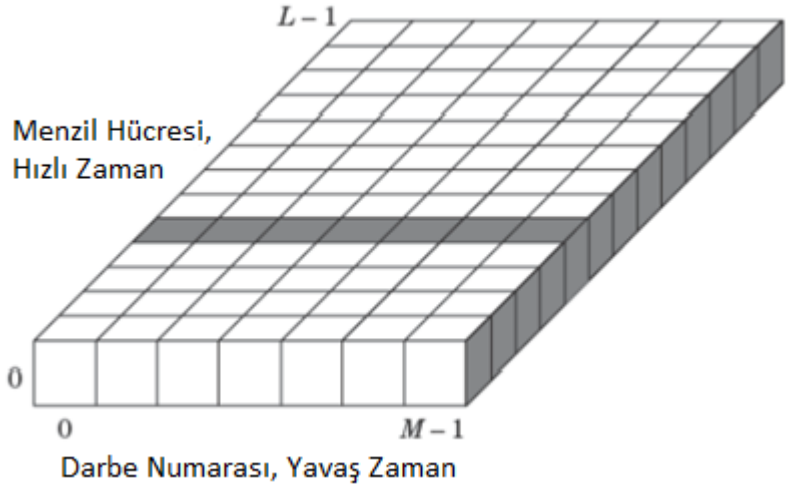
$$Sinyal\ Bant\ Genişliği(BW) = \frac{1}{\text{Darbe Genişliği(PW)}} \tag{4.1}$$

Radar menzil çözünürlüğü denklem 4.2'de gösterilmiştir.

$$\Delta R = \frac{\text{c}}{\text{2xBW}} \tag{4.2}$$

ΔR menzil çözünürlüğünü, c ışık hızını ve BW bant genişliğini ifade etmektedir. Radar darbeyi gönderdikten sonra dönen sinyal hedefin bulunduğu uzaklığın 2 katı mesafe kat ettiğinden formülde ışık hızı 2'ye bölünmüştür.

Radarın belirsiz olmayan menzil mesafesi darbelerin tekrar etme sıklığıyla değişmektedir. Sinyal işleme algoritmaları hedefleri tespit etmek için birden çok darbe sonrası dönen verileri birlikte işlemektedir. Darbe gönderildikten sonra bir sonraki darbeye kadar geçen süre "hızlı zaman (fast time)", her darbe gönderildiğinde aynı menzil hücresinden dönen verilerin incelendiği zaman da "yavaş zaman (slow time)" olarak radar literatüründe isimlendirilmiştir.



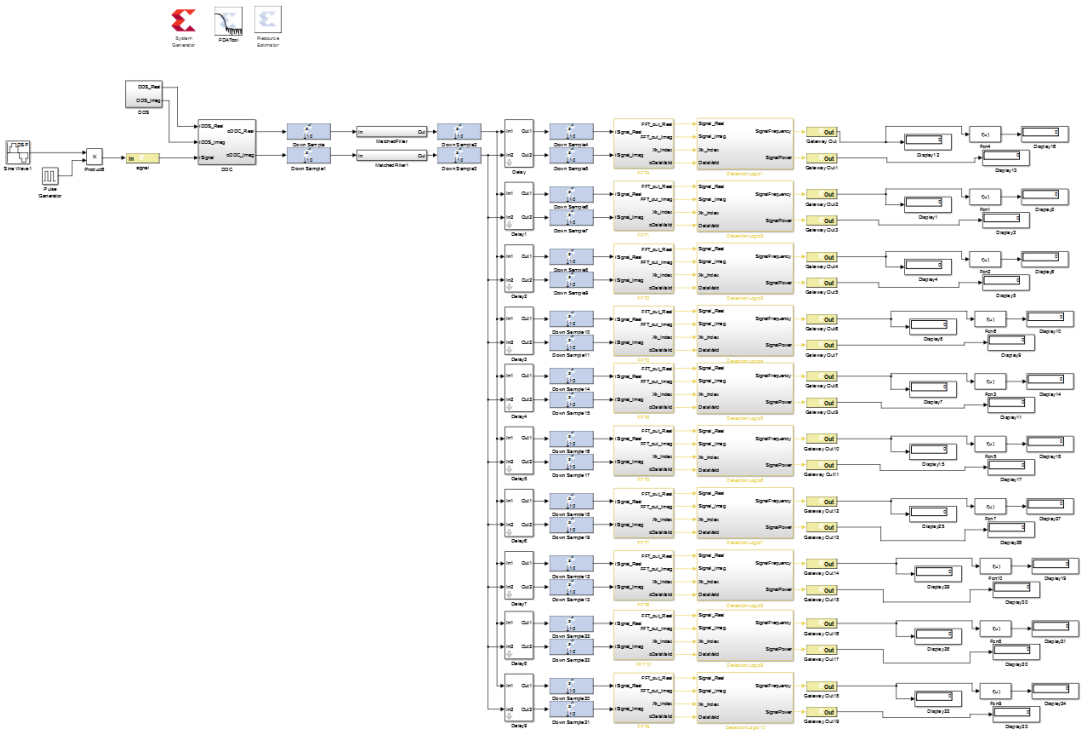
Şekil 3.17. Hızlı zaman, yavaş zaman matrisi

Şekil 3.17'de L adet sayıda menzil hücresi bulunmaktadır. menzil hücrelerinin bulunduğu eksen hazlı zaman eksenidir. M adet darbe gönderilmiş ve bu darbeler sonucu aynı menzilden dönen sinyal yavaş zaman ekseninde sıralanmışlardır. Yavaş zaman ekseninde FFT alınarak hedefin hangi menzilde ve hangi doppler frekansına sahip olduğu bulunur.

Belirsiz olmayan doppler frekansı darbe tekrar etme frekansı ile doğru orantılıdır. Doppler frekans tespitinde yavaş zaman verileri kullanıldığından, hedef üzerindeki doppler frekansının darbe tekrar etme frekansıyla örneklendiği söylenebilir.

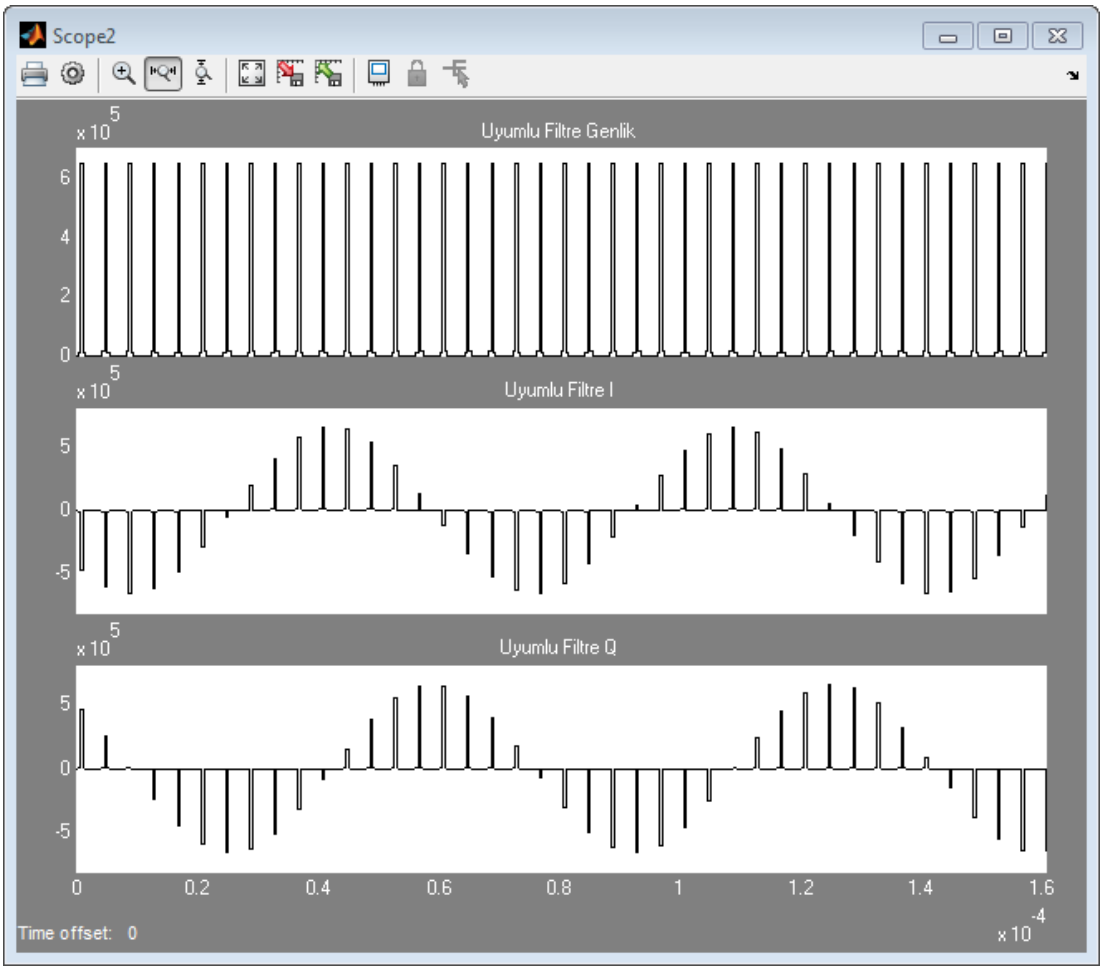
Tespit edilecek menzil hücresi sayısı ile tespit edilecek doppler frekansı aynı anda iyileştirilememektedir. İki değişken arasında ödünleşme yapmak gerekmektedir.

Darbe sıkıştırma algoritmasından sonra doppler işleme algoritması tasarıma eklenmiştir.



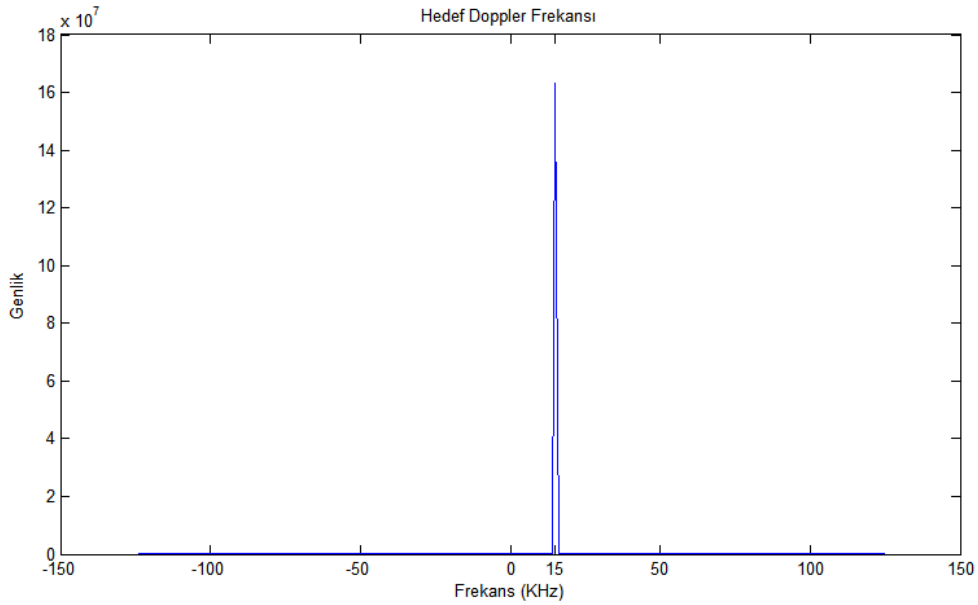
Şekil 3.18. Doppler işleme tasarımı

Şekil 3.18'de sadece 10 menzil hücresi için 10 farklı Xilinx FFT bloğu kullanılmıştır. FFT bloklarından sonra basit bir eşik değer ile tespit devresi kullanılmıştır. Tasarımda üzerine 15 KHz doppler frekansı eklenmiş hedefin benzetimi yapılmıştır. Uyumlu filtre çıkışında Şekil 3.19'de görüldüğü üzere I ve Q sinyalleri üzerinde doppler frekansı görülmektedir. Şekilde görülen her darbenin aynı hedeften döndüğü düşünülmüştür.



Şekil 3.19. Doppler Frekansı eklenmiş hedefin uyumlu filtre çıktısı

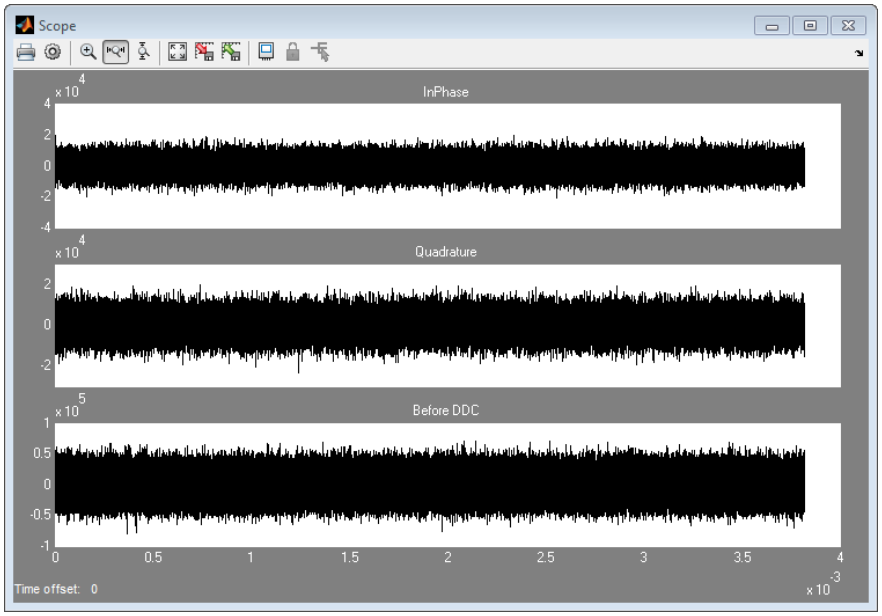
Uyumlu filtre çıkışı yavaş zaman ekseninde FFT algoritmasına sokulup frekansı tespit edildiğinde hedefin hızı Şekil 3.20'deki gibi bulunmuş olur.



Şekil 3.20. Hedef doppler frekansı

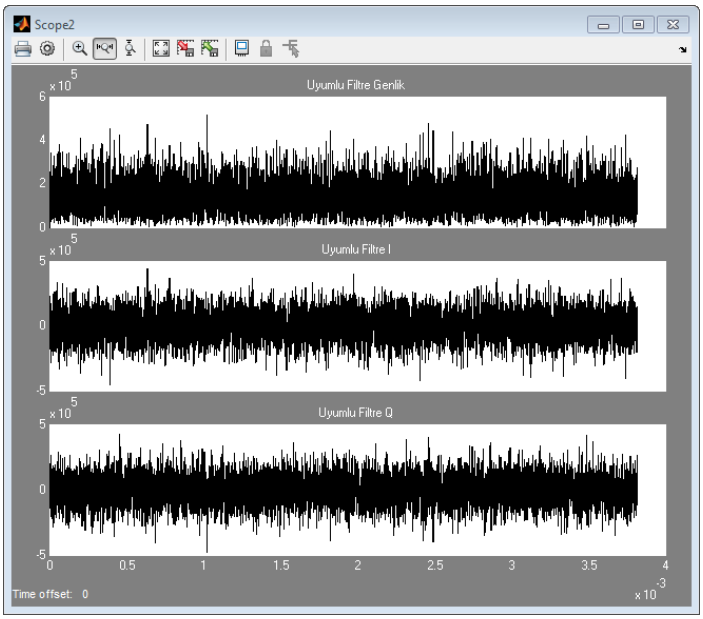
Sistemin gürültü altındaki başarımı ölçüldüğünde hedefi ve hedefin doppler frekansını başarılı bir şekilde bulduğu gözlemlenmiştir.

Sinyal gücünün 4 katı güce sahip gürültü giriş sinyaline eklenmiştir. Gürültülü sinyalin sayısal taban banda indirgeme algoritması çıkışı elde edilen sinyaller yukarıdan aşağıya sırasıyla I, Q ve giriş sinyalleri Şekil 3.21'de görülmektedir.



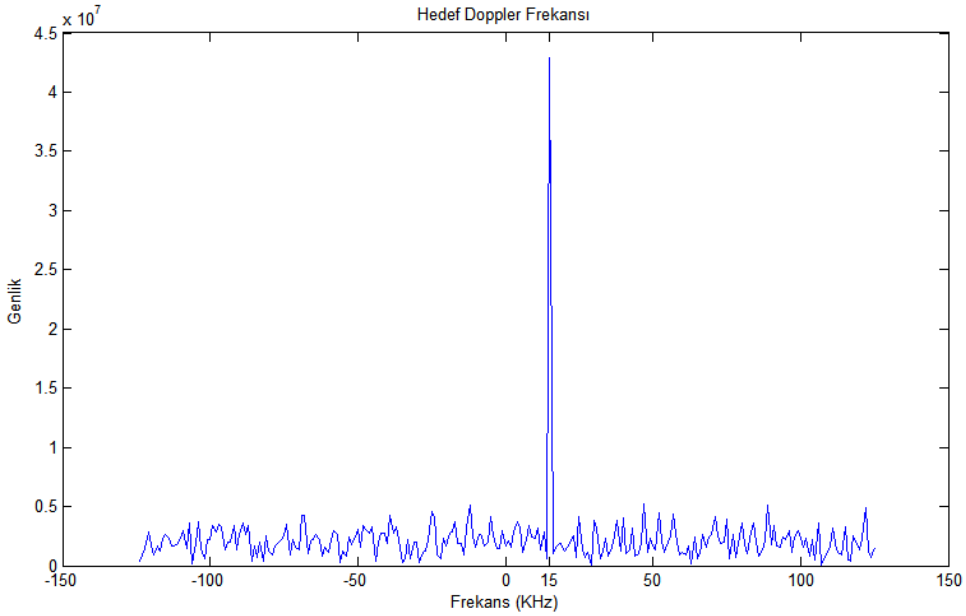
Şekil 3.21. Gürültü eklenmiş sinyalin DDC algoritmasındaki çıktısı

Uyumlu filtre çıkışı Şekil 3.22'de görülmektedir.



Şekil 3.22. Gürültü eklenmiş sinyalin uyumlu filtre çıktısı

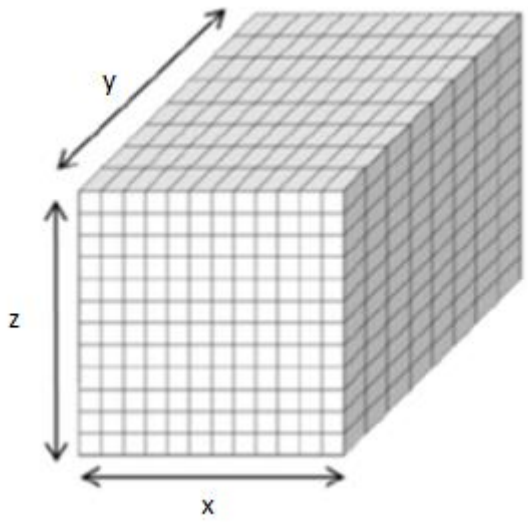
Doppler işleme algoritması uygulandığında hedef ve üzerindeki doppler frekansı gürültü altında da Şekil 3.23'te görüldüğü üzere tespit edilmiştir.



Şekil 3.23. Gürültü altında hedef doppler frekansı

3.5 Sayısal Hüzmeleme (Digital Beam Forming)

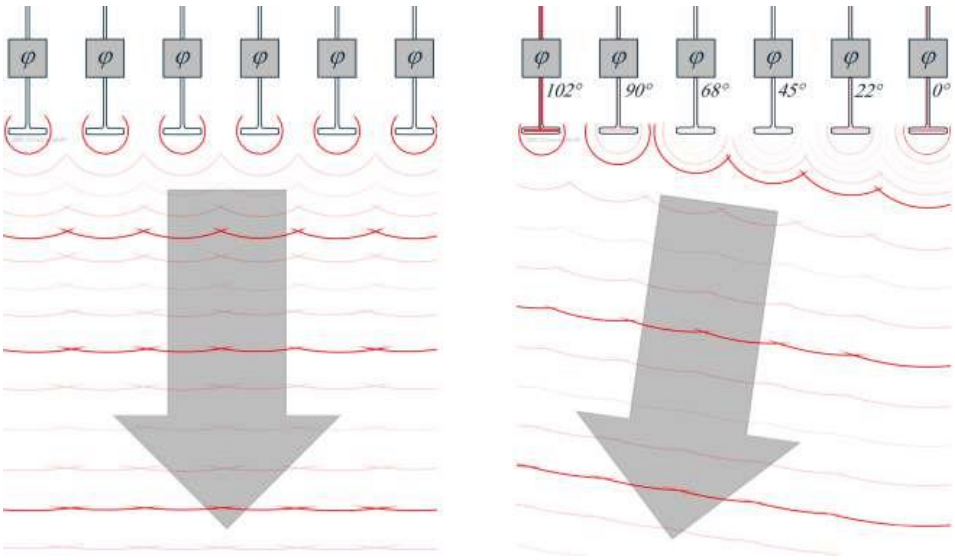
Sayısal hüzmeleme yöntemi akıllı anten sistemlerinde kullanılmaktadır. Bu yöntem sayesinde antenler mekanik olarak döndürülmeden istenilen yöne bakılabilmektedir. Bir çok antenden oluşan akıllı anten sistemlerinde sinyal varış yönü tahmini hüzmeleme algoritması kullanılarak yapılabilmektedir.



Şekil 3.24. Radar sinyal işleme veri küpü

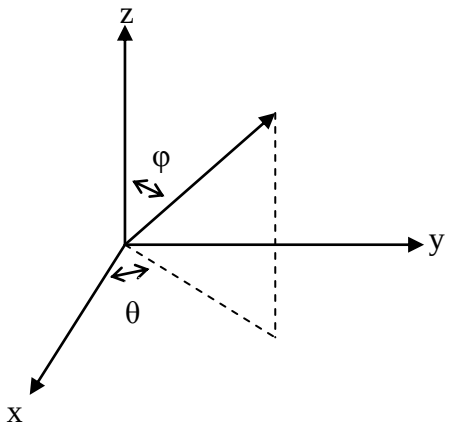
Şekil 3.24'de radar sinyal işleme algoritmalarında kullanılan veriler bir veri küpü şeklinde gösterilmiştir. 'y' hızlı zamanı göstermektedir. Gönderilen her bir darbe sonrası sırasıyla menzil hücrelerinden dönen veri bu eksende yazılmaktadır. 'x' yavaş zaman verisini göstermektedir. Bu eksen darbe sayısını tutmaktadır. Doppler işleme algoritmasında bu eksende FFT alınarak her bir menzil hücresinde eğer varsa doppler frekansı elde edilmektedir. Bu eksenler hali hazırda bir önceki bölüm olan doppler işleme bölümünde anlatılmıştı. Hüzmeleme algoritmasında hızlı zaman ve yavaş zaman katmanı her bir anten için oluşturulmaktadır. Böylece işlenecek veri miktarı katlanarak büyümektedir.

Hüzmeleme algoritmasının uygulanabilmesi için değişik sıralamaya sahip bir çok anten kullanılabilmektedir. Bu tez kapsamında sinyal işleme algoritmaları üzerinde durulduğundan anten dizilerinin dağılımının etkisi incelenmeyecektir. Doğrusal dizilmiş antenlerden gelen sinyaller incelenecektir.



Şekil 3.25. Doğrusal dizili antenlerin sinyal göndermesi

Antenlere giden sinyale bakılmak istenen yöne göre gerekli olan faz farkı verildiğinde anten mekanik olarak hareket ettirilmeden Şekil 3.25'te görüldüğü üzere istenilen yöne bakılabilmektedir[8]. Bu durum antenler alıcı durumundayken de geçerlidir.



Şekil 3.26. Antenlerin ve hedefin bulunduğu uzay

Şekil 3.26'da antenlerin ve hedefin bulunduğu uzay kartezyen koordinat sistemiyle ifade edilmiştir. Antenin bakılması istenen açı için gerekli faz bilgileri çıkarılıp darbe sıkıştırma algoritmasından sonra her bir anten ve her bir açı için hesaplanan faz bilgilerini içeren karmaşık sayı ile I ve Q sinyalleri çarpılmaktadır. Her bir anten için hesaplanan faz bilgilerini içeren vektöre doğrultu vektörü (steering vector) denilmektedir. Denklem 3.2'de bakılan açı 'k', anten sayısı 'm', anten konumları 'x', doğrultu vektörü e(k) ile gösterilmektedir[9].

$$e(\vec{k}) = \begin{bmatrix} e^{-j\vec{k}\vec{x_0}} \\ e^{-j\vec{k}\vec{x_1}} \\ \vdots \\ e^{-j\vec{k}\vec{x_{m-1}}} \end{bmatrix} \tag{3.2}$$

Bakılan açıyı gösteren k vektörü denklem 3.3'te gösterilmiştir. λ sinyalin dalga boyunu ifade etmektedir. Dizilen antenler arasındaki uzaklık λ cinsinden ayarlandığı için çarpım sonucu bu değerin bir önemi kalmamaktadır.

$$\vec{k} = -\frac{2\pi}{\lambda}\begin{bmatrix} \sin\varphi\ \cos\theta \\ \sin\varphi\ \cos\varphi \\ \cos\varphi \end{bmatrix} \tag{3.3}$$

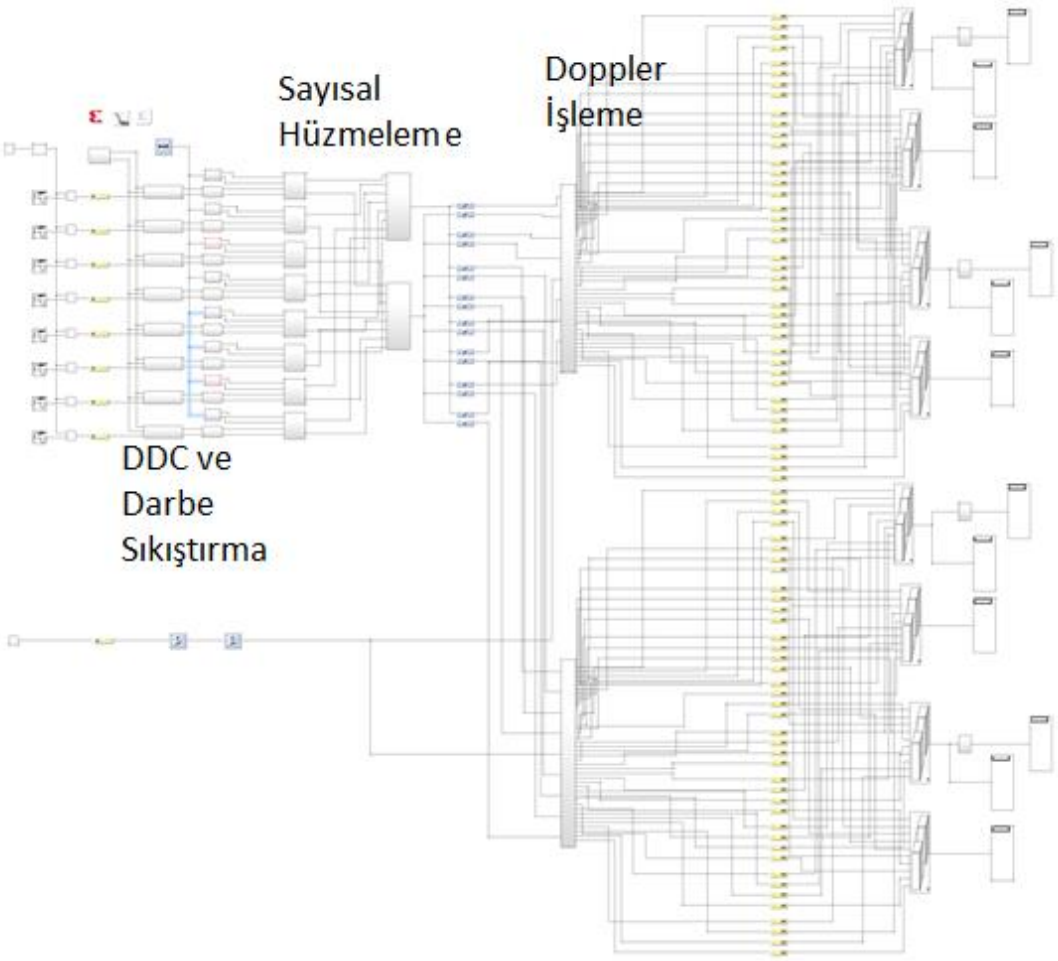
Antenlerin konumlarını gösteren x vektörü denklem 3.4'te görüldüğü üzere x, y, z bileşenlerinden oluşmaktadır.

$$\vec{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.4}$$

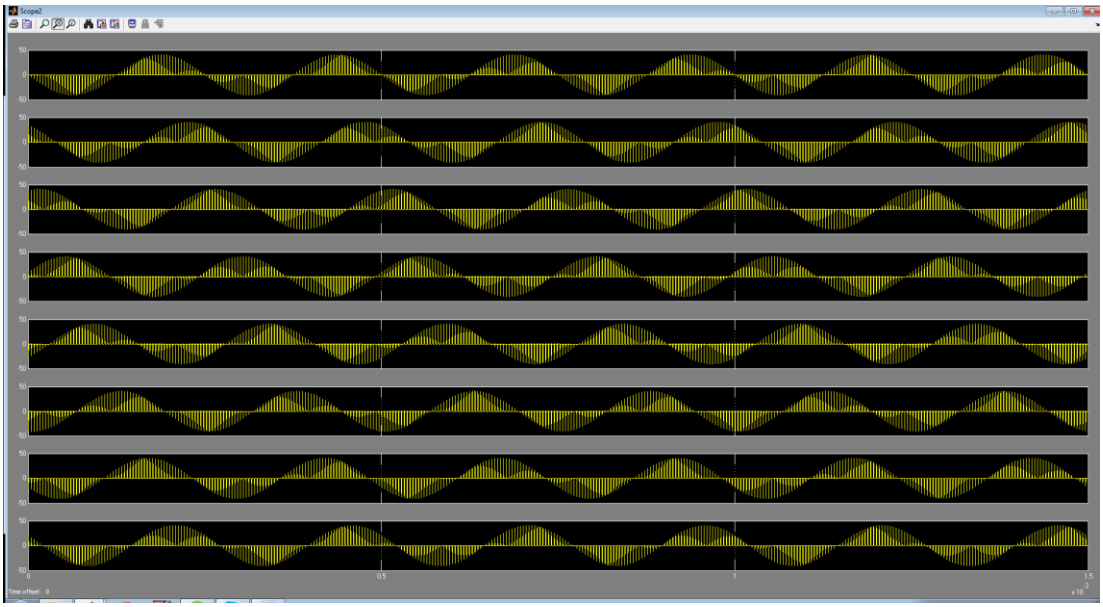
Böylece yukarıdaki formüller birleştirildiğinde bakılmak istenen açı için her bir antenden elde edilen sinyallerin çarpılması gereken karmaşık sayılar denklem 3.5 ile bulunmaktadır.

$$e(\vec{k}) = \begin{bmatrix} e^{-j\frac{2\pi}{\lambda}[\sin\varphi\ \cos\theta\ x_1 + \sin\varphi\ \cos\varphi\ y_1 + \cos\varphi\ z_1} \\ e^{-j\frac{2\pi}{\lambda}[\sin\varphi\ \cos\theta\ x_2 + \sin\varphi\ \cos\varphi\ y_2 + \cos\varphi\ z_2} \\ \vdots \\ e^{-j\frac{2\pi}{\lambda}[\sin\varphi\ \cos\theta\ x_{m-1} + \sin\varphi\ \cos\varphi\ y_{m-1} + \cos\varphi\ z_{m-1}} \end{bmatrix} \quad (3.5)$$

Xilinx system generator ile tasarlanan sayısal hüzmeleme modeli Şekil 3.27'de görülmektedir. Bu tasarımda 8 adet anten ile 8 adet açıya bakılmıştır. Her bir anten için sayısal taban banda indirgeme, darbe sıkıştırma, bakılmak istenen açı miktarında doğrultu vektörüyle çarpma ve doppler işleme algoritmaları uygulanmıştır. Her bir açı ve menzil hücresi için FFT alınıp basit bir eşik değer ile hedef tespit devresinden geçirilerek hedefin menzili, hızı ve açısı bulunmuştur.

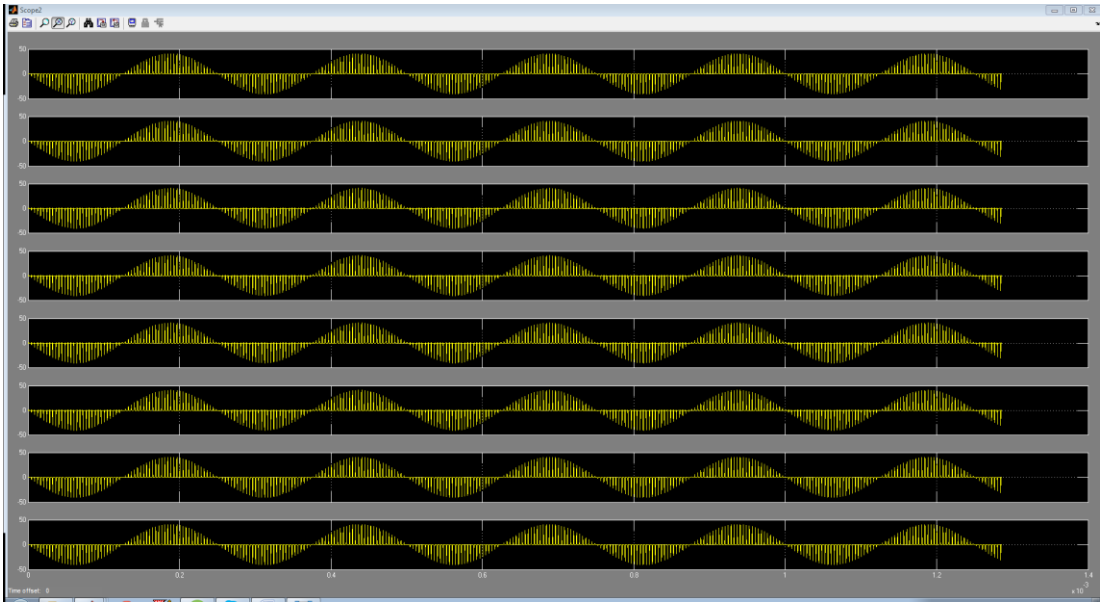


Şekil 3.27. Sayısal hüzmeleme tasarımı



Şekil 3.28. Doğrultu vektörüyle çarpılmadan önce darbe sıkıştırma sonucu

Belirli bir açıdan ve belirli bir menzilden, belirli bir doppler frekansına sahip sinyalin benzetimi simulink bloklarıyla yapılıp antenlere verilmiştir. Antenlerden alınan veriler sayısal taban banda indirgeme ve darbe sıkıştırma işlemlerinden sonra sinyalin görüntüsü Şekil 3.28'de gösterilmiştir. Benzetimi yapılan sinyalin açısına göre hesaplanan doğruldu vektörü ile bu açılar çarpıldığında Şekil 3.29'daki sinyaller elde edilmektedir.



Şekil 3.29. Doğrultu vektörüyle çarpıldıktan sonra darbe sıkıştırma sonucu

Doğrultu vektörüyle çarpma işleminden sonra tüm antenlerin verileri toplanarak elde edilen sinyal üzerinde doppler işleme algoritması uygulanmaktadır. Antenlerden elde edilen veriler eğer sinyalin geldiği açı için hesaplanan doğrultu vektörü ile çarpılmazsa, anten verilerinin toplanması sonucu elde edilen sinyalin gücü düşmektedir.

4. FPGA VE GPU SONUÇLARININ KARŞILAŞTIRILMASI

FPGA için yapılan tasarım Xilinx system generator ile tasarlanmıştır. FPGA üzerinde tasarımda anten sayısı, bakılacak açı sayısı, menzil hücresi sayısı gibi değişkenlerin artmasıyla FPGA kaynaklarının kullanım miktarı incelenmiştir. Ayrıca GPU'nun radar sinyal işleme algoritmalarının gerçek zamanlı işlem yapma ihtiyacını karşılayıp karşılamadığı incelenmiş olup, bu açıdan FPGA ile GPU arasında başarım karşılaştırması yapılmıştır.

Radar sinyal işleme algoritmalarında sinyalin en yüksek frekansı ADC'den örneklendiği andadır. Daha sonra sayısal taban banda indirgeme algoritması sonucu taban banttaki sinyal örnek seyreltme işleminden geçirilir ve frekansı düşürülür. Frekansı düşürülmüş olan sinyal darbe sıkıştırma algoritmasına sokulmaktadır. Bu algoritma çıktısı tekrar örnek seyreltme işlemine sokulur. Örnek seyreltme işlemleri sonucunda her bir menzil hücresi başına tek bir örnek kalır. Bu aşamadaki sinyalin frekansı darbe genişliğiyle ters orantılıdır. Geri kalan radar sinyal işleme algoritmaları bir kaç MHz olan bu düşük frekansta çalışır. FPGA kısmında tasarımı sınırlayan FPGA kaynaklarıdır. GPU'da uygulanacak radar sinyal işleme algoritmaları, darbe sıkıştırma algoritması çıktısından sonraki kısım olarak belirlenmiştir. Doğrultu vektörüyle 8 antenden gelen verinin çarpılıp toplanması ve sonrasında bakılacak her bir açı için doppler işleme algoritmasının uygulanması GPU'da yapılmıştır.

FPGA üzerinde çalışacak tasarım doğrultu vektörü ile bakılacak açı sayısına ve hesaplama yapılacak menzil hücresi miktarına göre ayrı ayrı tasarlanmıştır. Tasarımlarda bakılacak açı miktarı ve menzil hücresi sayısının çarpımı kadar FFT alınmıştır. Tasarımlarda kaynak kullanımı açısından sınırlayan faktör DSP slice sayısı yani kullanılan çarpıcı sayısı olmuştur.

Çizelge 4.1, Çizelge 4.2 ve Çizelge 4.3'te sırasıyla 10, 20 ve 30 menzil hücresi için değişen sayıda bakılan açı sayısına göre yapılan tasarımlarda kullanılan FPGA kaynak miktarları verilmiştir.

Çizelge 4.1. 10 Menzil hücresi hesaplaması için FPGA kaynak kullanımı

Kullanılan Kaynak Miktarları				
FPGA Kaynakları	Bakılan Açı Sayısı			
	2	4	8	16
Mantık Birimi	18872	34612	66092	129052
DSP - Slice	656	1096	1976	3736

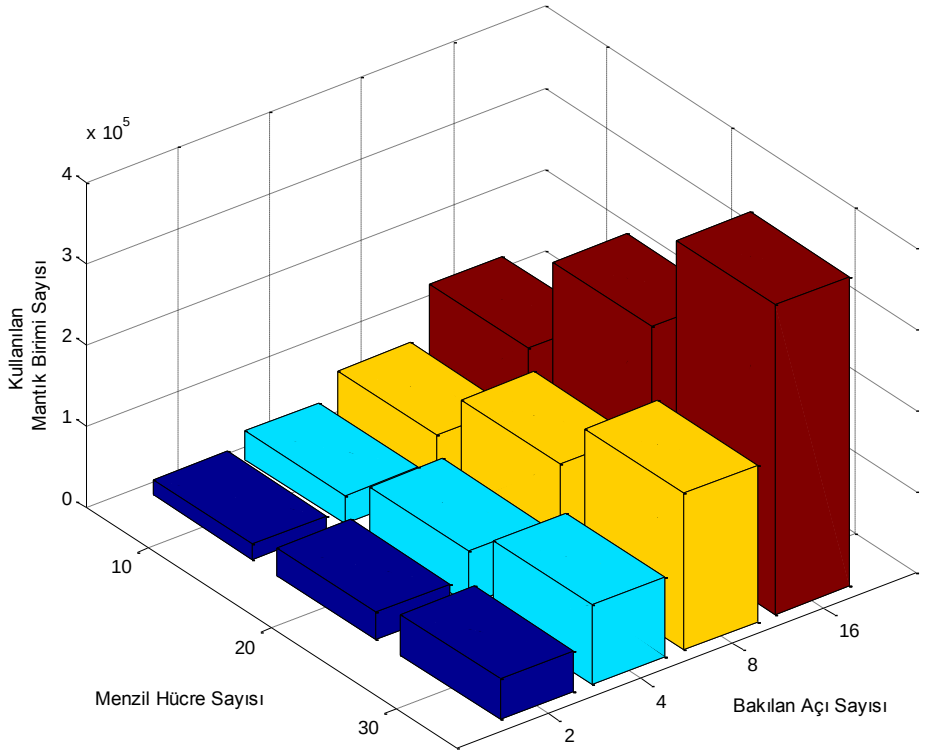
Çizelge 4.2. 20 Menzil hücresi hesaplaması için FPGA kaynak kullanımı

Kullanılan Kaynak Miktarları				
FPGA Kaynakları	Bakılan Açı Sayısı			
	2	4	8	16
Mantık Birimi	33849	65348	128974	254972
DSP - Slice	1096	1976	3736	7256

Çizelge 4.3. 30 Menzil hücresi hesaplaması için FPGA kaynak kullanımı

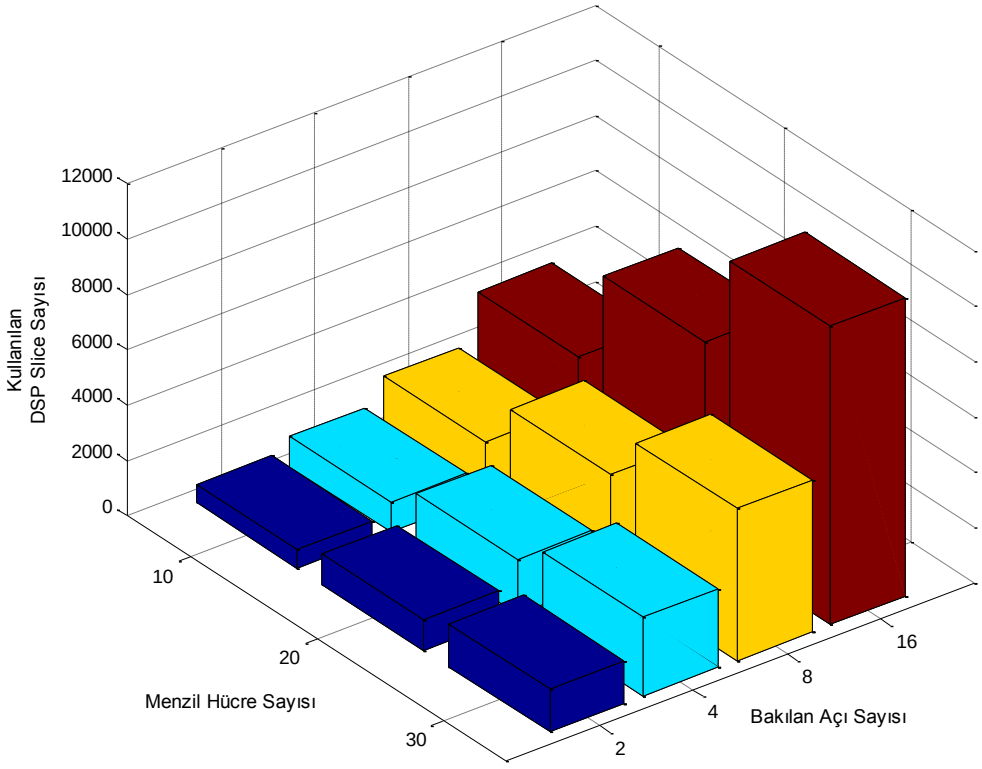
Kullanılan Kaynak Miktarları				
FPGA Kaynakları	Bakılan Açı Sayısı			
	2	4	8	16
Mantık Birimi	50352	97572	192012	380892
DSP - Slice	1536	2856	5496	10776

Şekil 4.1'deki grafik tasarımda değişen menzil hücre sayısı ve bakılan açı sayısına göre mantık birimi kullanımının değişimini göstermektedir.



Şekil 4.1. Kullanılan Mantık Birimi Sayısı

Şekil 4.2'deki grafik tasarımda değişen menzil hücre sayısı ve bakılan açı sayısına göre DSP slice kullanımının değişimini göstermektedir.



Şekil 4.2. Kullanılan DSP Slice Sayısı

Bir FPGA içerisinde yaklaşık olarak en çok 3600 DSP slice bulunmaktadır. Kaynak kullanım sonuçlarından anlaşılacağı üzere tasarım büyüdükçe DSP slice miktarı kısıtlayıcı faktör olmuştur ve tasarım tek bir FPGA'e sığamaz hale gelmeye başlamıştır.

GPU'da alınacak başarım sonuçları için öncelikle doğrultu vektörü ile çarpma işlemi yapılmıştır. OpenCL ile yapılan işlemin kernel kodu Şekil 4.3'de gösterilmiştir.

```
__kernel void mult(
__global float2* anten,
__constant float2* sv,
__global float2* result,
__local float2* lmem,
const int num_anten,
const int num_angles)
{
	int gid = get_global_id(0);
	int size_anten = get_global_size(0)/num_anten;
	int lid = get_local_id(0);
	int lsize = get_local_size(0);
	int gr_id = get_group_id(0);

	lmem[lid] = anten[(lid/32)*size_anten + gr_id*32 + lid%32];
	barrier(CLK_LOCAL_MEM_FENCE);

	float2 res = {0, 0};

	float2 a, c;
	float2 mul1;

	for(int i = 0; i < num_anten; i++){
		a = lmem[i*32 + lid%32];
		c = sv[(lid/32)*num_angles + i];

		mul1 = a*c;
		res.x += mul1.x - mul1.y;
		res.y += (a.x + a.y)*(c.x + c.y) - mul1.x - mul1.y;
	}
	result[size_anten*(lid/32) + gr_id*32 + lid%32] = res;
}
```

Şekil 4.3. OpenCL doğrultu vektörü ile çarpma kernel kodu

Şekil 4.3'de kernel kodu verilen işlemin hesaplama süreleri Çizelge 4.4'te verilmiştir.

Çizelge 4.4. GPU'da doğrultu vektörü ile çarpma işlemi süreleri

Hesaplama Süreleri (mikro saniye)				
Hesaplama Yapılan Menzil Hücresi Sayısı	Bakılan Açı Sayısı			
	2	4	8	16
10	22,208	26,752	37,984	63,616
20	22,244	26,816	38,368	64,128
30	22,312	26,837	38,4	64,936

GPU'da doppler işleme algoritmalarının uygulanması için clFFT isminde hazır bir OpenCL FFT kütüphanesi kullanılmıştır. Bu işlemin hesaplama süreleri Çizelge 4.5'te görülmektedir.

Çizelge 4.5. GPU'da doppler işleme algoritması hesaplama süreleri

Hesaplama Süreleri (mikro saniye)				
Hesaplama Yapılan Menzil Hücresi Sayısı	Bakılan Açı Sayısı			
	2	4	8	16
10	20,864	21,344	24,448	26,56
20	21,376	22,976	26,72	38,88
30	25,088	27,68	33,632	56,832

Yapılan FFT işlemleri çarpma işlemlerinden hızlı çıktığı görülmektedir. Bunun sebebi hazır çok iyi en iyilemesi yapılmış bir FFT kütüphanesi kullanılmış olmasıdır.

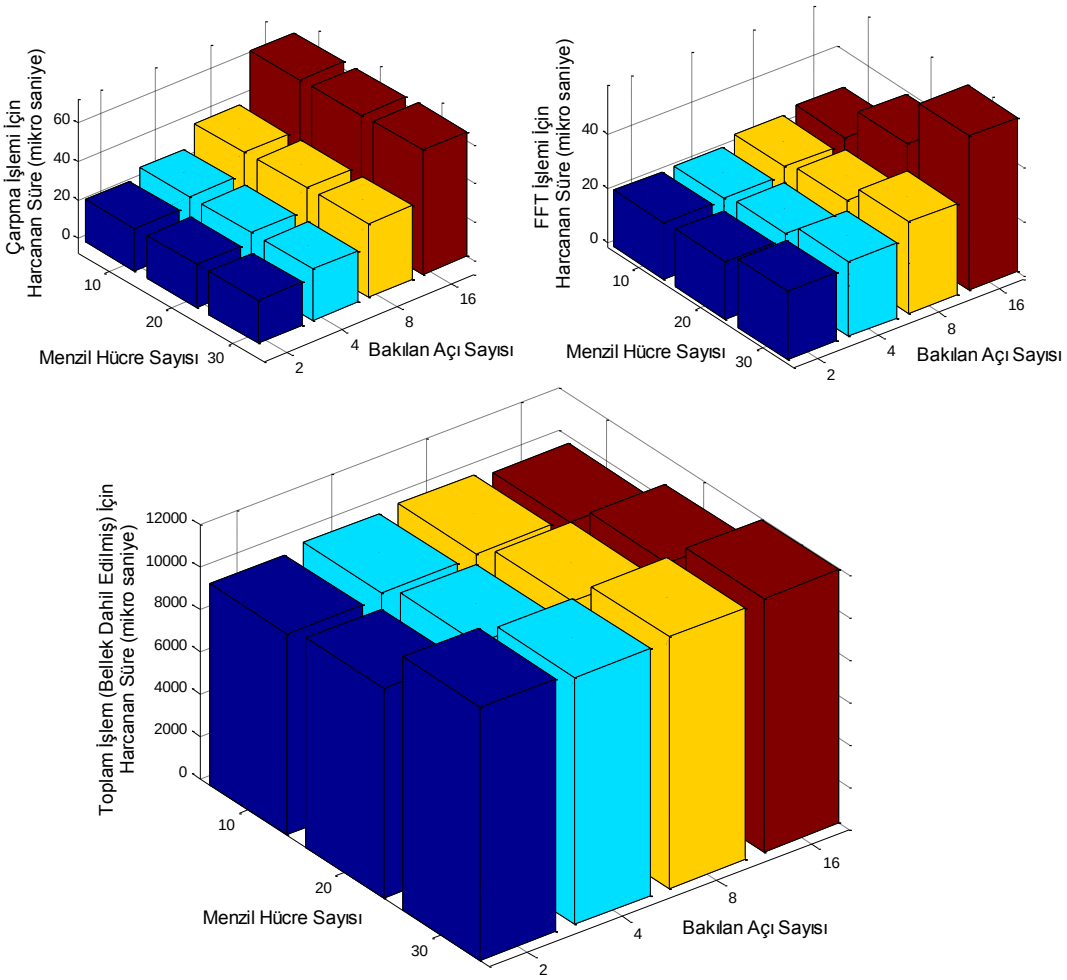
Yukarıda sonuçları verilen GPU işlemleri kernel işlemlerinin süreleridir. CPU'dan GPU belleğine veri aktarımı verilmiş olan süreler içerisinde yoktur. Veri aktarım

süreleri de dahil edildiğinde Çizelge 4.6'da görüldüğü üzere süreler çok hızlı bir şekilde büyümüştür.

Çizelge 4.6. GPU'da harcanan toplam süreler

Toplam Süre (mikrosaniye)				
Hesaplama Yapılan Menzil Hücresi Sayısı	Bakılan Açı Sayısı			
	2	4	8	16
10	9510	9755	9884	9654
20	9907	10324	10597	10706
30	11970	11668	11892	11993

Şekil 4.4'te üst kısımda bulunan iki grafikte GPU'da yapılan çarpma ve FTT işlemlerinin süreleri gösterilmiştir. Alttaki grafikte ise bellek erişim sürelerinin de dahil edildiği toplam işlem süresi gösterilmiştir. Grafikten de görüleceği üzere GPU'da yapılan tek başına çarpma ve FFT işlemleri bellek erişimine göre çok kısa sürmektedir.



Şekil 4.4. GPU'da İşlem Süreleri

FPGA'de yapılmış olan benzetimlerde sinyalin gerçek zamanlı işlenmesi için gereken süre 655.360 mikro saniyedir. Şekil 4.4'te görüleceği üzere bellek işlemleri dahil edilmediği taktirde GPU üzerinde çalışan kernel kodu bu hızı fazlasıyla sağlamaktadır. Fakat CPU belleğinden GPU belleğine veri transferi bir dar boğaz yaratmakta ve bu sürenin sağlanmasını engellemektedir.

5. SONUÇLAR

Radar sistemleri günden güne yaygınlaşmaktadır. Radar sinyal işleme algoritmaları yüksek paralel işlem gücü gerektirmektedir. Paralel işlem gücü konusunda önderlik eden iki donanım, FPGA ve GPU, radar sinyal işleme algoritmaları uygulanarak karşılaştırılmıştır.

Sonuçlar incelendiğinde FPGA gerçek zamanlı olarak verileri işleyebilmektedir fakat tasarım büyüdükçe FPGA'de çarpıcı (DSP slice) kaynak ihtiyacı da büyük oranda artmaktadır. GPU üzerinde çalışan kernel kodu gerçek zamanlı olarak veriyi işleyebilme yeteneğine sahiptir. Fakat verilerin CPU belleğinden GPU belleğine aktarılması ve geri alınması süre bakımından bir dar boğaz oluşturmaktadır.

KAYNAKLAR

[1] Richards, M. A., Scheer, J. A., Holm, W. A., Principles of Modern Radar, Vol. I: Basic Principles, *SciTech Publishing*, 2010

[2] Yağlıkçı, A. Giray, Yüksek Lisans Tezi, 2014, FPGA tabanlı sayısal sinyal işleme algoritmalarına özelleştirilmiş yardımcı işlemci tasarımı

[3] http://www.nvidia.com.tr/object/cuda-parallel-computing-tr.html

[4] Yıldız, Ertan, Yüksek Lisans Tezi, 2011, NVIDIA cuda ile yüksek performanslı görüntü işleme

[5] http://www.ustudy.in/node/3199

[6] Aykenar, M. Burak, Yüksek Lisans Tezi, 2013, Mobil platformlarda fır filtre tasarımı için FPGA ve GPU uygulamalarının enerji ve başarım analizi

[7] Khronos OpenCL Working Group, "The OpenCL specification version 1.2," 2010, http://khronos.org/opencl.

[8] http://www.radartutorial.eu/06.antennas/Digital%20Beamforming.en.html

[9] Johnson, D. H., Dudgeon, D. H., Array Signal Processing: Concept and Techniques, *Prentice Hall*, Englewood Cliffs, NJ, 1993

[10] http://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf

[11] Krishnaveni, V., Kesavamurthy, T. and Aparna, B., Beamforming for Direction-of-Arrival (DOA) Estimation-A Survey, International Journal of Computer Applications 61(11):4-11, 2013.

[12] Klilou A., EURASIP Journal on Advances in Signal Processing 2014, 2014

[13] Seugin, E., Tessier, R., Knapp, E., Jackson, R. W., A Dynamically-Reconfigurable Phased Array Radar Processing System, International Conference onField Programmable Logic and Applications, 2011

[14] Aliakbarian, H., Volski, V., Westhuizen E. van der, Wolhuter, R., and Vandenbosch, G. A. E., Analogue versus Digital for Baseband Beam Steerable Array used for LEO Satellite Applications, Proceedings of the Fourth European Conference on Antennas and Propagation (EuCAP), 2010

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ÖZGÜR, Muhammet

Uyruğu : T.C.

Doğum tarihi ve yeri : 20.05.1988 Eskizara-Bulgaristan

Medeni hali : Evli

Telefon : 0 (536) 247 56 25

Faks : 0 (312) 292 42 90

e-mail : mozgur@etu.edu.tr

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
Y. Lisans	TOBB ETÜ	2015
Lisans	ODTÜ/Elektrik-Elektronik Müh.	2011

İş Deneyimi

Yıl	**Yer**	**Görev**
2013-...	RST Uzaktan Algılama A.Ş.	Sayısal Tasarım Mühendisi
2011-2013	TOBB Ekonomi ve Teknoloji Üniversitesi	Tam Burslu Y. Lisans

Yabancı Dil

İngilizce

Rusça (Başlangıç)

Yayınlar

Aykenar, M.B.; **Özgür, M.**; Bayraktar, V.E.; Ergin, O., "Tag simplification: Achieving power efficiency through reducing the complexity of the wakeup logic," Energy Aware Computing (ICEAC), 2011 International Conference on , Nov. 2011.

Aykenar, M.B.; **Özgür, M.**; Şimşek, O.S.; Ergin, O., " Adapting the Columns of Storage Components for Lower Static Energy Dissipation," VLSI and System-on-Chip (VLSI-SoC), 2013 IEEE/IFIP 21th International Conference on, Oct. 2013.