MOBİL PLATFORMLARDA FIR FİLTRE TASARIMI İÇİN FPGA VE GPU UYGULAMALARININ ENERJİ VE BAŞARIM ANALİZİ

MEHMET BURAK AYKENAR

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

AĞUSTOS 2013

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Necip CAMUŞCU

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Mehmet Burak AYKENAR tarafından hazırlanan MOBİL PLATFORMLARDA FIR FİLTRE TASARIMI İÇİN FPGA VE GPU UYGULAMALARININ ENERJİ VE BAŞARIM ANALİZİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Oğuz ERGİN

Tez Danışmanı

Tez Jüri Üyeleri

Başkan :Doç. Dr. Oğuz ERGİN _____

Üye : Yrd. Doç. Dr. Murat ÖZBAYOĞLU _____

Üye : Dr. Fatih SAY _____

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.

.....

Mehmet Burak AYKENAR

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 – Ağustos 2013**

Mehmet Burak AYKENAR

MOBİL PLATFORMLARDA FIR FİLTRE TASARIMI İÇİN FPGA VE GPU UYGULAMALARININ ENERJİ VE BAŞARIM ANALİZİ

ÖZET

Doğrusal faz cevabı ve istikrarlı olması itibariyle FIR filtreler sayısal sinyal işleme uygulamalarında çokça kullanılmaktadırlar. Gömülü sistemlerde kullanılan ve paralel uygulanma yapısındaki FIR filtreleri gerçeklemek için FPGA'lar en gözde çözüm olarak gözükmektedirler.

Son zamanlarda büyük ölçekte paralellik özelliği gösteren ve yüksek hesaplama gücü gerektiren problemlerin çözümü için GPGPU etkili bir yöntem olarak ortaya çıkmıştır. Fakat GPGPU'lar yüksek güç tüketimleriyle kötü bir üne sahiptirler. Mobil platformlar için düşük güç tüketimi her zaman gerekli olduğu için çok çekirdekli güce aç GPGPU'lar mobil uygulamalarda çok etkili bir çözüm olmamışlardır. Xilinx'in Virtex serisi gibi güçlü FPGA'lar da aşırı enerji harcamaları nedeniyle mobil platformlarda kullanılması uygun gözükmemektedir.

Bu çalışmada, değişik tap büyüklüklerine sahip FIR filtreler Spartan 3A DSP FPGA ve ARM Cortex-A9 ve orta sınıf Vivante GPU'dan oluşan bir mobil heterojen sistem üzerinde gerçeklenmiştir. Gerçeklenen FIR filtreler enerji harcamaları ve başarım yönlerinden karşılaştırılmıştır. Spartan FPGA enerji harcaması Xilinx'in Xpower Estimator yazılım aracı kullanılarak hesaplanırken heterojen sistemin enerji harcaması sisteme giren akımın ölçülmesiyle hesaplanmıştır. FIR filtre gerçeklenirken FPGA için değişik mimariler ve GPGPU için de birçok kod eniyilemesi kullanılmıştır.

Sonuçlar Spartan 3A DSP FPGA'nın başarım ve enerji harcama yönünden GPGPU'ya göre daha üstün olduğunu göstermektedir. GPGPU, FPGA'lara göre henüz daha yeni bir saha olduğundan dolayı ilerleyen yıllarda yeni tekniklerle birlikte GPGPU tasarımlarının FPGA'lara yetişip geçmesi beklenebilir.

Anahtar Kelimeler: FPGA, GPGPU, FIR Filtre, mobil platformlar, düşük-güçlü sistemler,

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. – August 2013**

Mehmet Burak AYKENAR

FIR FILTER ENERGY AND PERFORMANCE COMPARISON OF GPU AND FPGA FOR MOBILE PLATFORMS

ABSTRACT

FIR filtering is one of the most common needs of digital signal processing applications demanding linear phase response and stability. Due to the inherent parallel property of FIR filter structure, FPGAs are common solutions to implement FIR filters for embedded systems.

Recently, GPGPU emerged as an effective technique for solving computation-intensive problems having massive level of parallelism. However, they are infamous with their huge power consumption. Since low power is always a requirement for mobile platforms, many-core power hungry GPGPUs seem not to be a very effective solution. Powerful FPGAs like Virtex series of Xilinx are also consume substantial amount of energy and not suitable to be utilized in mobile platforms.

In this work, FIR filters of different tap sizes on Spartan 3A DSP FPGA and a mobile heterogeneous system which includes an ARM Cortex-A9 accompanying a light-weight Vivante GPU. The implemented FIR filters are compared with respect to energy consumption and throughput values. Spartan FPGA energy consumption is calculated via Xilinx's Xpower Estimator software tool, while the heterogeneous system energy is calculated by measuring the current flowing into the heterogeneous system. Different architectures for FPGA implementation and variety of code optimizations are done for GPGPU implementation.

The results show that Spartan 3A DSP FPGA outperforms the heterogeneous platform with respect to both energy efficiency and performance aspects. GPGPU is relatively a new concept with respect to FPGAs; thus, new GPGPU designs may challenge FPGAs with new emerging techniques.

Keywords: FPGA, GPGPU, FIR Filter, mobile platforms, low-power systems

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, ROKETSAN'daki amirlerim ve iş arkadaşlarıma, her zaman yanımda olan ve özellikle iyi eğitim almam için her türlü zahmete katlanan anneciğim, babacığım ve kardeşime, tez çalışmam ve diğer konularda sabrı ve özverisiyle beni hep destekleyen biricik eşime teşekkürü bir borç bilirim.

Hamd âlemlerin Rabbi olan Allah'adır (Kuran-ı Kerim - 1/2)

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

KISALTMALAR

Kısaltmalar	Açıklama
FIR	Finite impulse response
FPGA	Field programmable gate array
GPU	Graphics processing unit
DSP	Digital signal processing/processor
OpenCL	Open computing language
IIR	Infinite impulse response
SOS	Second order sections
FFT	Fast Fourier transform
DA	Distributed arithmetic
LUT	Look-up table
VHDL	Very high speed integrated circuits hardware description language
RAM	Random access memory
DCM	Digial clock manager
API	Application programming interface

1. GİRİŞ

Sinyaller farklı frekans bileşenlerinden oluşmaktadırlar ve bu frekans bileşenlerinden bazıları istenmeyen, ya da gürültü olarak adlandırılırlar. Sinyalin bilgi taşıyan anlamlı kısmının istenmeyen kısımlardan ayrışması için sayısal filtreleme tekniği kullanılan yaygın bir metottur. Daha hızlı işlemciler ve daha çok sığa kapasiteli bellek birimleri geliştikçe sayısal filtreler analog filtrelerin yerlerini almaya başladılar. Analog filtreler direnç, kapasitör ve indüktör gibi pasif devre elemanlarıyla gerçeklenebildiği gibi transistör ve operasyonel yükselteç gibi aktif devre elemanlarıyla da gerçeklenebilirler [1]. Analog filtre kullanmanın dezavantajı sıcaklık ve zamanla filtrenin karakteristiğinin ve performansının değişmesidir. Sayısal filtrelerse kararlıdırlar ve her türlü değişime karşı daha dayanıklıdırlar. Ayrıca yüksek sığalı bellekler ve hızlı işlemcilerle beraber analog filtrelerle gerçeklenmesi mümkün olmayan çok yüksek dereceli filtrelerin tasarımları da kolaylaşmış oldu.

FIR filtreler doğrusal faz cevabı ve her zaman kararlı olma özellikleri nedeniyle sinyal işleme uygulamalarında en çok kullanılan sayısal filtrelerdir. Daha keskin filtre karakteristiğine ulaşmak için özyinelemesiz yapıları yüzünden daha fazla taba sahip FIR filtreler kullanılmak zorunda kalınmaktadır. FIR filtreler temel olarak üç adet bileşenden oluşmaktadırlar: Çarpıcı, gecikme elemanı ve toplayıcı. Bir FIR filtre uygulamasındaki en zaman alan ve karmaşık olan operasyon şüphesiz ki çarpma işlemidir. Bir FIR filtrenin tap sayısı, ya da seviye sayısı, kaç adet çarpıcı kullanılması gerektiğini ifade eder. Daha fazla çarpıcı kullanma koşuluyla daha yüksek seviyeli FIR filtrelerle çok keskin cevaplar elde edilebilir. FIR filtre uygulamasındaki çarpma işlemleri birbirlerinden bağımsız oldukları için eğer yeterli sayıda çarpıcı birim varsa çarpma işlemleri paralel olarak gerçekleştirilebilir. Bu yüzden bir FIR filtre uygulamasında başarımı arttırmak için paralel çalışabilen çarpma birimlerine sahip donanımlar tercih edilmelidirler. FPGA'lar ve GPU'lar bu tarz sistemlere örnek olarak verilebilir.

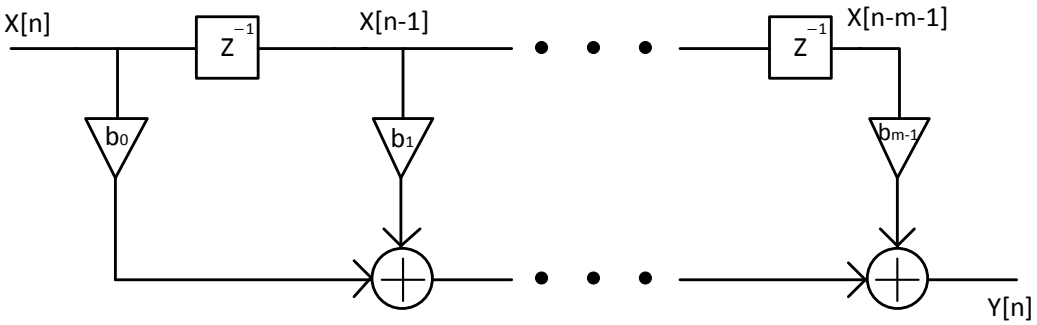
Sinyal operasyonlarını barındıran bütün sistemlerde FIR filtreler gereklidirler. Bu sistemlere örnek olarak cep telefonları, tabletler, navigasyon cihazları vs. verilebilir. Batarya ömrünü uzun tutmak için bu tarz mobil platformlarda düşük güç tüketimi çok büyük önem arz etmektedirler. Bu yüzden basitçe çok güçlü bir FPGA ya da GPU alıp bu tarz mobil platformlarda FIR filtre gerçeklemek kolay ve uygulanabilecek bir yöntem değildir.

Bu tezde, biri Spartan 3A DSP tabanlı biri de ARM Cortex-A9 ve Vivante GPU'dan oluşan ve OpenCL ara yüzünü kullanan heterojen sistemde düşük güç tüketimli iki adet farklı FIR filtre mimarisi tasarlanmıştır. Bu iki sistemde çalışan FIR filtrelerin

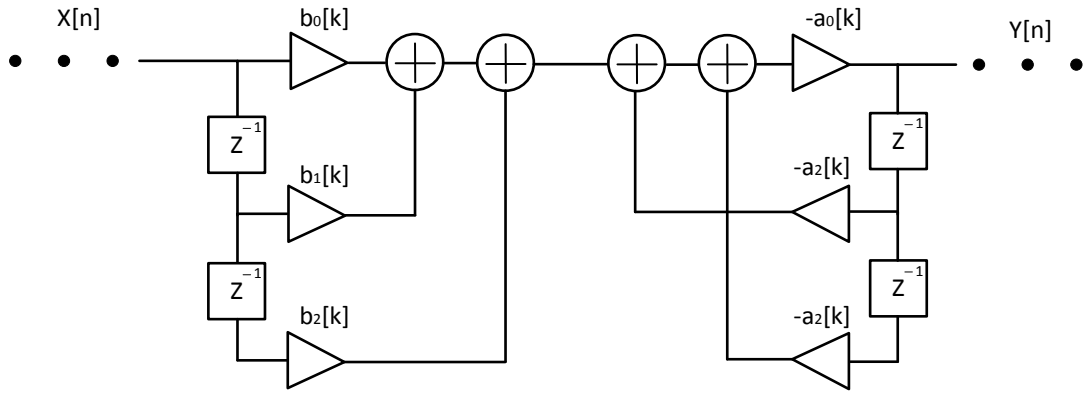
başarım ve enerji harcamaları karşılaştırılmıştır. Spartan 3A DSP FPGA üzerinde gerçeklenen FIR filtre heterojen sisteme göre daha iyi enerji sarfiyatı ve başarıma erişmiştir. Tezin ilk kısmında FIR filtre teorisi anlatılmış, ardından FIR filtrenin FPGA ve GPU üzerinde gerçeklenme sürecinden bahsedilmiştir. Karşılaştırma bölümünde gerçeklenmiş FIR filtrelerin başarım ve enerji harcamaları karşılaştırmalarının sonuçları verilip daha sonra geçmişte yapılan akademik çalışmalar verilmiştir. Sonuçlar kısmında da tez çalışmasının sonuçları özetlenmiş ve gelecekte yapılabilecek çalışmalardan bahsedilmiştir.

2. FIR FİLTRE TEORİSİ

İmpals cevaplarına göre sayısal filtreler iki grupta toplanmaktadırlar. Sonlu impals cevaba (Finite Impulse Response) sahip filtrelerde filtrenin girişine bir impals sinyal verildiğinde filtrenin çıkışında sinyal sonlu cevaba sahip olur, yani belirli bir süre sonrasında çıkış sıfıra gelir. Sonsuz impals cevaba sahip (Infinite Impulse Response) filtrelerde ise filtre girişine bir impals sinyal verildiğinde filtrenin çıkışında teorik olarak sonsuza kadar devam edecek bir çıktı oluşur. FIR ve IIR filtrelerin çıkış cevabındaki bu değişik davranış filtrelerin gerçeklenmesindeki yapı farklılıklarından meydana gelmektedir. FIR filtreler özyinelemesiz (non-recursive) özellik göstermektedirler, yani filtrenin çıkış sinyali sadece o andaki ve geçmişteki giriş sinyal örneklerinden hesaplanmaktadır. Tam tersine, IIR filtrelerde ise filtrenin çıkış sinyali giriş ve çıkış sinyal örneklerinden hesaplanmaktadır. FIR filtrelerin IIR filtrelere göre uygulamada en büyük avantajı; FIR filtrelerin doğrudan yapıda tam paralellik özelliği göstererek gerçeklenebilmesiyken IIR filtrelerin finite precision etkisinden dolayı ikinci-dereceden-kısımlar (second-order-sections/SOS) yapısında gerçeklenebilmesidir. IIR filtreler çarpanların nicemlenmesi (quantization) sonucunda eğer doğrudan yapıda gerçeklenirlerse kararlı durumdan kararsız duruma geçebilirler, bu da filtrenin yanlış çalışmasına sebep olur. SOS yapıda gerçeklenme IIR filtrelerde kararsız durum geçmeyi engelleyici bir faktördür. IIR filtreler SOS yapısında gerçeklendiklerinde tam paralel uygulamayı birbirine bağımlı ikinci dereceden kısımlar engellemektedir. Fakat FIR filtrelerde doğrudan-yapıda (direct-form) gerçeklenme sayesinde çarpma işlemleri paralel olarak gerçekleştirilebilmektedir. Şekil 2.1 ve Şekil 2.2'de sırasıyla doğrudan yapıda gerçeklenmiş bir FIR filtre ve SOS yapıda gerçeklenmiş bir IIR filtrenin bir k'nıncı kısmı verilmiştir.



Şekil 2.1. Doğrudan yapıda gerçeklenmiş FIR filtre



Şekil 2.2. SOS yapıda gerçeklenmiş IIR filtrenin k'nıncı kısmı

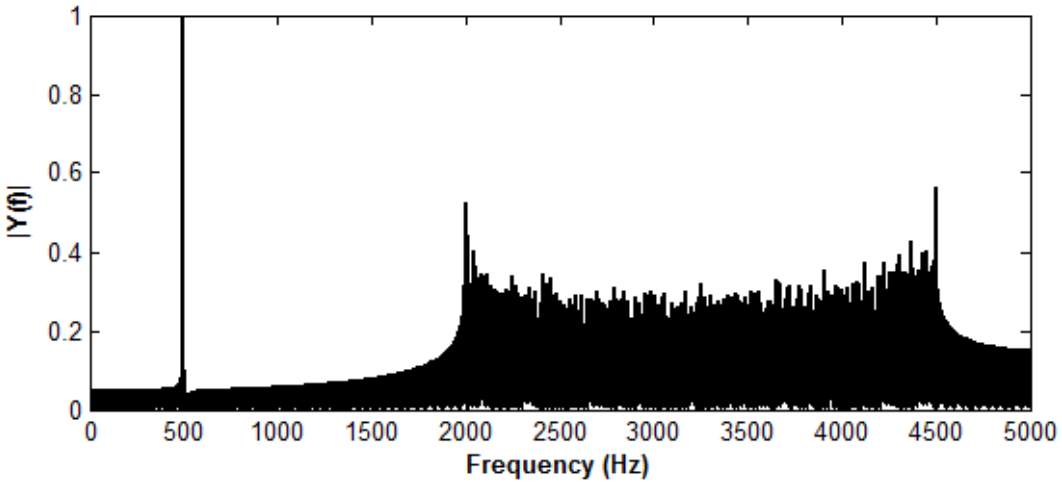
Şekil 2.1'de üçgen şeklinde gösterilen (b_0-b_{m-1}) değerleri FIR filtrenin çarpanları (coefficients) olarak tanımlanmıştır. Filtre çarpanları FIR filtre doğrusal fark denkleminden çıkarılmaktadır. M taplı bir FIR filtrenin doğrusal fark denklemi denklem 2.1'de verilmiştir. Filtre çarpanları, filtre giriş örnekleriyle (x[n]-x[n-m-1]) çarpılır daha sonra bu çarpım sonuçları Şekil 2.1'de çemberlerle ifade edilen toplayıcılarla toplanarak filtre sonucu hesaplanır. Kare şeklinde gösterilen ve içerisinde Z^{-1} ibaresi bulunan parçalar de ters-Z dönüşümünü ifade etmektedir. Zaman ekseninde ters-Z dönüşümü gecikme operasyonuna tekabül eder. Bu birimlerin sağ tarafındaki sinyal örneği, sol tarafında bulunan örnekten bir önceki örneği ifade etmektedir.

$$y(n)= b_0X(n) + b_1X(n-1)+\ldots+b_{m-1}X(n-m-1) \qquad (2.1)$$

Şekil 2.2'de gösterilen FIR filtrede, çıkış sinyal vektörü y(n), giriş sinyal örneklerinin [x(n), x(n-1), ... x(n-m-1)] filtre katsayılarıyla [b_0, b_1, … b_{m-n-1}] çarpılıp daha sonra bu çarpımların hep beraber toplanmasıyla elde edilmektedirler. FIR filtreleri doğrudan-yapıda gerçeklemenin en büyük avantajı birbirinden bağımsız çarpma işlemleridir. Eğer yeteri kadar çarpıcı birimi varsa bu çarpma işlemlerini paralel olarak gerçekleştirmek doğrudan-yapıda FIR filtre kullanılarak mümkün olabilmektedir.

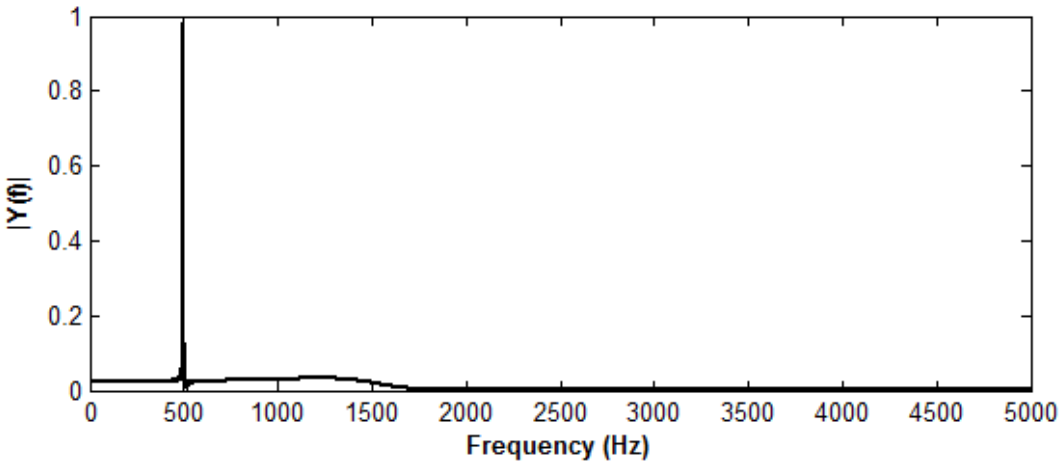
FIR filtre yöntemiyle çok değişik formlarda filtre çeşitleri tasarlanabildiği gibi bunlar arasında en popüler ve en çok kullanılanları alçak geçiren (lowpass), yüksek geçiren (highpass), band geçiren (bandpass) ve band durduran (bandpass) filtrelerdir. Bu tür filtreler arzu edilen frekans aralığını geçirir ya da durdurur. Bir sinyalin frekans

komponentlerinin genlik değerleri hızlı Fourier dönüşümü (Fast Fourier Transform/FFT) kullanılarak analiz edilebilir [2]. FFT teorisini açıklamak için 10 kHz'de örneklenmiş 500 Hz'lik bir sinüs sinyaline 2 – 4.5 kHz frekans aralığında gürültü eklenmiştir ve Şekil 2.3'te bu sinyalin FFT yardımıyla hesaplanan genlik spektrumu gösterilmiştir.



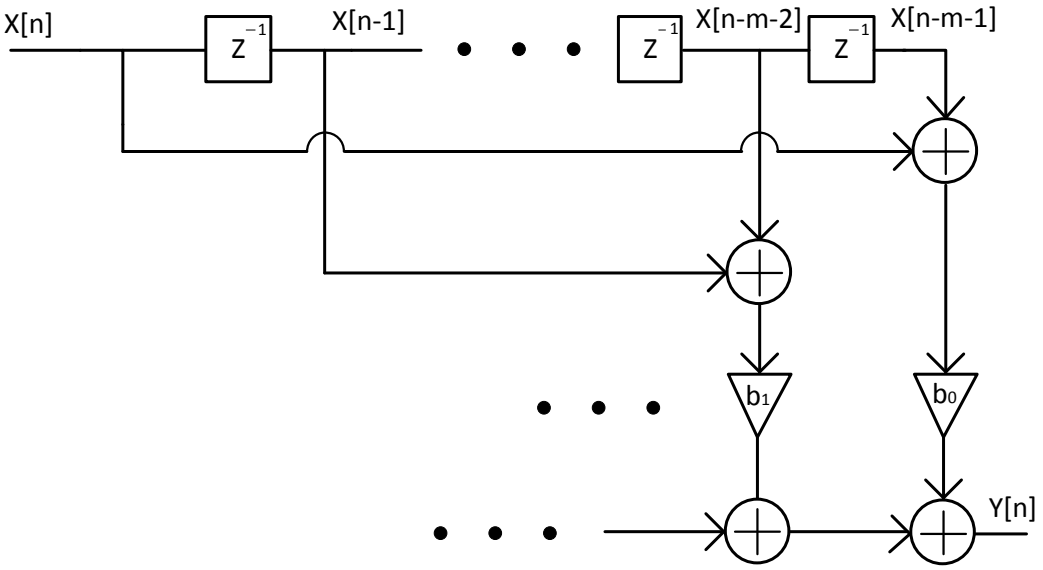
Şekil 2.3. Bir genliğinde 500 Hz'lik sinüs sinyali ve 2 – 4.5 kHz frekans aralığında çeşitli büyüklüklerde eklenmiş gürültünün genlik spektrumu

Şekil 2.3'te görüldüğü üzere sinyalin 500 Hz'de 1 büyüklüğünde bir sinüs bileşeni vardır. Ayrıca 2 – 4.5 kHz aralığında yaklaşık 0.3 – 0.6 büyüklüklerinde sinüs bileşenleri bulunmaktadır. Eğer sinyaldeki gürültü bileşenlerini temizlemek istersek 500 Hz'nin altını geçiren bir alçak geçiren FIR filtre kullanılabilir. Gürültülü sinyalin filtrelenmiş hali Şekil 2.4'te gözükmektedir. Filtrelenmiş sinyalde 500 Hz'lik bileşen etkilenmemiş, fakat 500 Hz'nin üzerindeki gürültü bileşenler elimine edilmiştir.



Şekil 2.4. Filtrelenmiş 500 Hz'de gürültülü sinüs sinyali

Birçok uygulamada FIR filtrelerde filtre katsayıları simetri özelliği göstermektedir [3]. Mesela m-taplı bir FIR filtre için eğer i = m/2 – 1 durumundayken b0 = b_{m-1}, b_1 = b_{m-2}, ... , b_i = b_{i+1} ise bu durumda FIR filtre simetrik doğrudan-yapıda şeklinde gerçeklenebilir. Simetrik doğrudan-yapıda gerçeklenen FIR filtrelere katlanmış (folded) yapıda da denilmektedir. Katlanmış FIR filtre yapısında iki adet giriş sinyali örneği aynı filtre katsayısıyla çarpılacağı için iki ayrı örnek için farklı birer çarpıcı kullanmaya gerek yoktur. Bu iki adet örnek aynı katsayıyla önce çarpılıp sonra toplanılması yerine önce iki örnek toplanıp daha sonra katsayıyla tek bir çarpma işlemi gerçekleştirilir. Bu sayede çarpma işlemi sayısı yarıya inmiş olur ve uygulamada büyük bir kolaylık sağlanmış olur. Şekil 2.5'te m-taplı bir FIR filtrenin katlanmış formda gerçeklenmiş hali verilmiştir. Çarpıcı birimi sayısı yarıya inmiştir.



Şekil 2.5. Katlanmış formda gerçeklenmiş m-taplı FIR filtre

Dağıtılmış aritmetik (distributed arithmetic/DA) tekniği FPGA'larda sinyal işlemede sıklıkla kullanılan bir metottur. DA tekniğinde çarpma işlemi, toplama ve kaydırma tekniğinin lookup table (LUT) ile kullanılmasıyla elimine edilir ve böylece FIR filtredeki çarpma işlemlerini paralel olarak uygulamak için özel çarpıcı birimlerine olan bağımlılıktan kurtulmuş olunur. DA tekniğinde en büyük dezavantajlardan birisi de gereken LUT sayısının filtre seviyesiyle eksponansiyel olarak artmasıdır. Bu yüzden yüksek dereceli FIR filtrelerde çokça FPGA lojik hücresi kullanmak gerekmektedir.

Denklem 2.2'de K-taplı bir FIR filtrenin doğrusal fark denklemi verilmiştir. Burada x ve y filtrenin giriş ve çıkış denklemlerini göstermektedir, a_k filtre katsayılarını ve K da filtre seviyesini ifade etmektedir.

$$y[n] = \sum_{k=0}^{K-1} a_k x[n-k] \tag{2.2}$$

DA tekniğinde, FIR filtrenin giriş vektörü olan x'in ikili komplement formunda L-bit olarak ifade edildiğini varsayalım. Katsayının sıfırdan küçük olduğu durumda en soldaki bit sayının işaretini belirtecektir. Bu durumda Denklem 2.3'te FIR filtrenin yeni oluşan denklemi görülebilir.

$$x[n-k] = -b_{k,0} + \sum_{l=1}^{L-1} b_{k,l} 2^{-l} \tag{2.3}$$

Denklem 2.3'te denklem 2.2'nin sonucunu kullanırsak denklem 2.4'ü elde ederiz.

$$y[n] = \sum_{k=0}^{K-1} a_k (-b_{k,0} + \sum_{l=1}^{L-1} b_{k,l} 2^{-l})$$

$$y[n] = -(\sum_{k=0}^{K-1} a_k b_{k,0}) + \sum_{l=1}^{L-1} (\sum_{k=0}^{K-1} (a_k b_{k,l})\, 2^{-l} \tag{2.4}$$

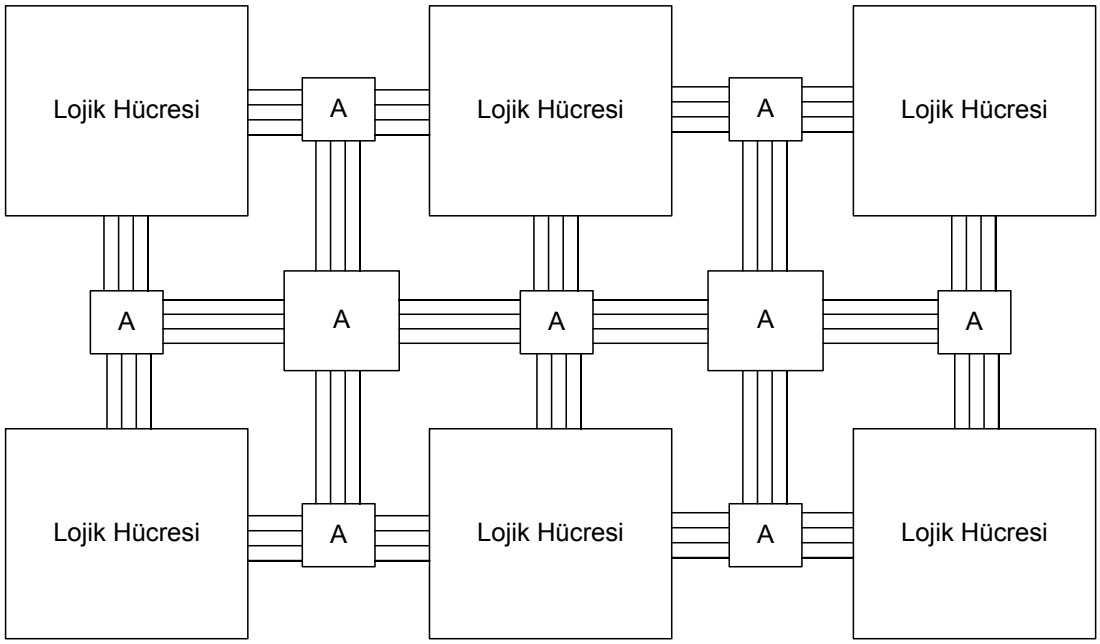
Denklem 2.4'te görüldüğü üzere parantez içerisindeki terimler 2^K kadar farklı değer alabilmektedirler ve bu değerler filtre katsayılarının toplamlarının alabileceği bütün kombinasyonları içerir. Bu değerler önceden hesaplanarak LUT'lara kaydedilir ve böylece çarpma-toplama algoritması yerine sadece LUT erişimi ve toplama işlemleri kullanılmış olur.

3. FPGA'DA FIR FİLTRE GERÇEKLENMESİ

FIR filtreler doğrudan yapıda tasarlandıklarında çarpma işlemleri birbirlerinden bağımsız olduklarından dolayı bu işlemleri yeterli sayıda çarpma birimi bulunduğu takdirde paralel olarak gerçekleştirmek mümkündür. FPGA'lar (Field Programmable Gate Array) programlanabilir mimarileri ve yüksek derecede paralel uygulamalar için uygun makro birimleri barındırmasından dolayı FIR filtreleri gerçeklemek için çok uygun sistemlerdir ve yaygın olarak bu amaçla kullanılmaktadırlar. Tezin bu bölümünde önce FPGA'lar ve uygulama alanları hakkında genel bir bilgi verilmiş, daha sonra da MATLAB DSP System Toolbox yardımıyla oluşturulan VHDL kodunun FPGA üzerinde gerçeklenmesiyle FIR filtre tasarlanması anlatılmıştır.

3.1 FPGA Hakkında Genel Bilgiler

FPGA'lar birbirlerine programlanabilir bağlantı birimleriyle bağlı matriks yapıda configure edilebilir lojik bloklardan (Configurable Logic Block/CLB) oluşan programlanabilir yarıiletken aletlerdir. FPGA'lar herhangi bir uygulama için kolayca programlanabilir, aynı FPGA içeriği değiştirilip tekrar programlanarak bir başka uygulama için de kullanılabilir. Bir FPGA'nın lojik hücrelerinden oluşan iç mimarisi teorik olarak Şekil 3.1'de verilmiştir.



Şekil 3.1. FPGA lojik hücreleri ve bağlantı yapısı

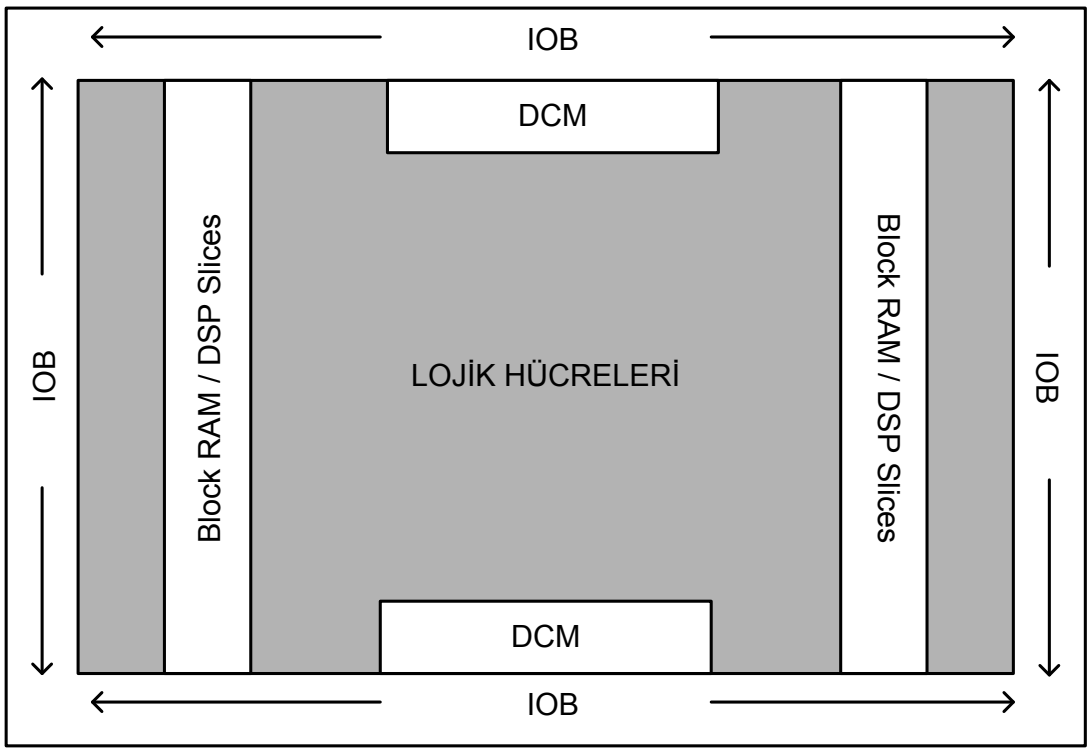
FPGA'lar esnek mimari yapıları ve programlanma özellikleri sayesinde kendilerine endüstride hızla yer bulmuşlardır ve günümüzde de otomotivden, telekomünikasyona, uzay uygulamalarından savunma sanayine ve özellikle yüksek performansla birlikte esneklik isteyen birçok uygulama alanında tercih edilmektedirler.

Xilinx ve Altera firmaları dünya üzerinde en çok müşteriye sahip iki firmadırlar. Bu firmalar birçok uygulamaya özel ve farklı ihtiyaçlara hitap eden değişik FPGA aileleri üretmektedirler. Xilinx firmasının piyasada sıklıkla kullanılan farklı FPGA ailelerinden FPGA örnekleri ve özellikleri çizelge 3.1'de verilmiştir.

Çizelge 3.1. Çeşitli Xilinx FPGA ürünlerinin kaynak durumları

	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

Lojik hücresi (Logic Slice) olarak isimlendirilen birimler FPGA'ların temel işlem ve depolama birimlerini içermektedirler. Bir lojik hücresinin içerisinde temel olarak 4 ya da daha fazla girişli lookup table (LUT), bir adet flip flop ve çeşitli çoklayıcılar (multiplexer) bulunmaktadır. Basitçe doğruluk tablosu oluşturulabilen sade devreler bir adet lojik hücresi kullanılarak gerçeklenebilir. Fakat karmaşık lojik yapısına ve çokça kaydedici yazmaca (register) ihtiyacı olan devreler ancak birden çok lojik hücresi kullanılarak gerçeklenebilirler. Sürekli gelişmekte olan FPGA teknolojisinde artık FPGA yongalarının içerisine özel fonksiyona sahip makro birimler yerleştirilmektedir. Bu makro birimler hard olarak çipe gömülü durumda olup sadece izin verilen ölçüde parametreleri programlanabilmektedirler. Bu makro bloklara örnek olarak DCM, RAM, DSP slice, çarpıcı birimleri gösterilebilir. Genel olarak bir FPGA'nın mimari yapısı şekil 3.2'de görülebilir.



Şekil 3.2. FPGA genel mimari yapısı

Bu tez çalışması için Xilinx'in Spartan ailesinden DSP işlemleri için özel olarak tasarlanmış Spartan 3A-DSP serisinden XC3SD3400A FPGA'sı seçilmiştir [4]. Bu FPGA 126 adet DSP makro bloğa sahiptir [5]. Her bir DSP blok içerisinde 18 bitlik çarpıcı birimi, çeşitli ön-toplayıcı birimleri ve 40-bitlik toplayıcı birimi ve sinyal seçimi için çoklayıcılar (multiplexer) bulunmaktadır. Doğrudan yapıda ve katlanmış yapıda gerçeklenecek olan FIR filtreler için DSP blokları kullanılmıştır. 126 adetlik DSP blok sayısıyla 126-taplık FIR filtrelerde çarpma işlemleri tamamen paralel olarak gerçekleştirilebilmektedir. Fakat 126-tap'tan daha fazla dereceli bir FIR filtre gerçeklenmek istenirse çarpma işlemlerinin hepsinin paralel olarak uygulanması imkansız olmaktadır ve kaçınılmaz olarak serileşme gerçekleşmektedir.

Bu tezde başarım ve enerji harcaması karşılaştırması için 16, 32, 64, 128, 256 ve 512 seviyeli FIR filtreler seçilmiştir. Doğrudan-yapıda mimarisini kullanan FIR filtre gerçeklenmelerinde çarpma işlemleri için seri aşama sayısı 128-tap FIR filtre için iki, 256-tap FIR filtre için üç, ve 512-tap FIR filtre gerçeklenmesi için beştir. 16, 32 ve 64-tap FIR filtrelerde çarpma işlemleri tamamen paralel olarak yapılabilmektedir. Katlanmış FIR filtre mimarisinde ise çarpma gereksinimi yarı sayıya düştüğü için 16, 32, 64 ve 128-tap FIR filtrelerin gerçeklenmesinde çarpma işlemleri tamamen paralel olarak uygulanabilmektedir. 256-tap FIR filtre için iki ve 512-tap FIR filtre

gerçeklenmelerinde çarpma işlemleri için üç adet seri aşama bulunmaktadır. Distributed arithmetic (DA) tekniğinde ise lojik hücreleri yeterli ise tamamen paralel çarpma işlemleri gerçekleştirilebilmektedir [6]. FIR filtre yapıları MATLAB yazılımın "DSP System Toolbox" aracı yardımıyla tasarlanmıştır. Bir sonraki bölümde bu yazılım hakkında genel bir bilgi verilmiş ve FPGA üzerinde gerçeklenecek FIR filtrelerin sentezlenebilir VHDL kodları bu yazılım yardımıyla oluşturulmuştur.

3.2 MATLAB DSP System Toolbox

Matlab™, Mathworks® firmasının geliştirdiği ve genel mühendislik uygulamaları için büyük kolaylık sağlayan bir yazılımdır. Matlab'ın en güçlü yanlarından birisi kolay yazılım ortamı ve değişik uygulamalara özel araç kutuları (toolbox) olmasıdır. Matlab'ın 2011 sürümünde 40'tan fazla uygulama için araç kutusu bulunmaktadır. Biyoenformatik, haberleşme sistemi, veri tabanı, ekonometri, sinyal işleme ve görüntü işleme bu araç kutularına örnek olarak verilebilir. Araç kutuları, farklı uygulamalar için yüksek seviyede fonksiyonlar sağlayarak analiz, tasarım ve kodlama aşamalarını çok hızlandırmaktadır. Araç kutuları uygulamalara özel Matlab API'leri olarak da düşünülebilir.

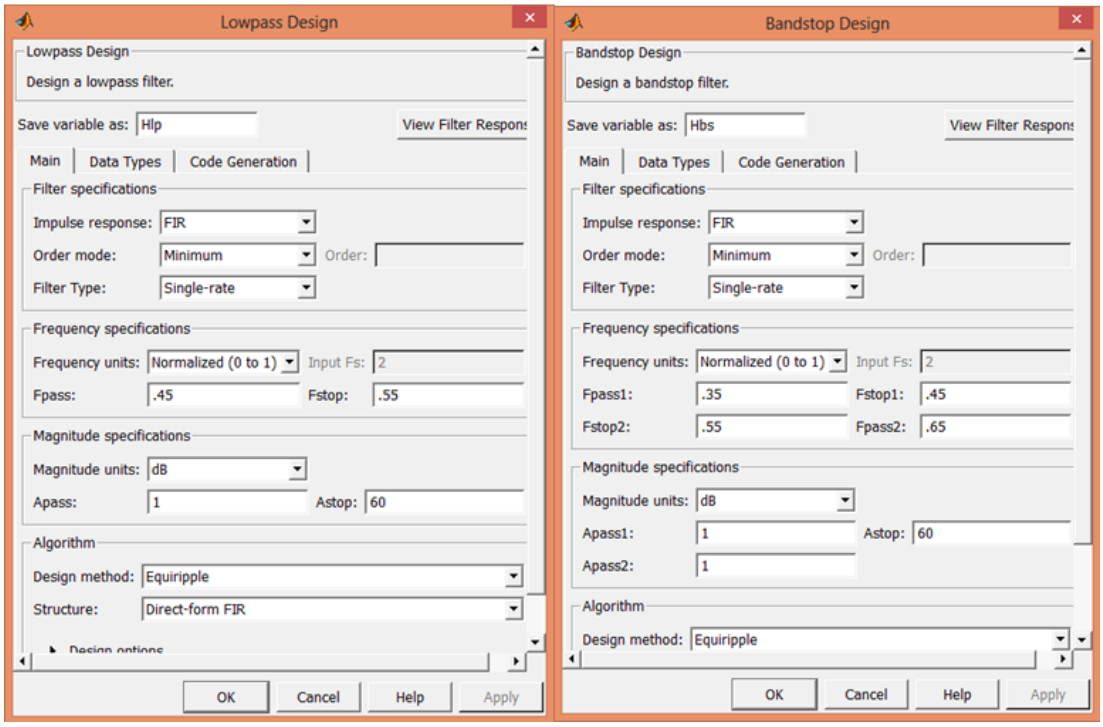
Sayısal bir filtrenin tasarımı birkaç aşamadan oluşmaktadır. Bunlar, sayısal filtrenin tipinin belirlenmesi, gereksinimlerin tanımlanması, tasarım algoritmasının seçimi ve filtre katsayılarının hesaplanması olarak özetlenebilir. Matlab'da sayısal filtre tasarlamak için birden çok metot bulunmaktadır. Bu metotlardan bazılarında sayısal filtre tasarımı, Matlab kodu yazılıp program çalıştırılarak oluşturulur. DSP System Toolbox, kod yazarak filtre tasarlamak isteyenler için bu aşamaları kolaylaştıracak fonksiyonlar içermektedir. Kod yazarak tasarlanan bir sayısal filtre, aslında görsel ara yüz komutları kullanılarak da tasarlanabilir.

Matlab'ın 2011 sürümünde sinyal işleme araç kutusunda iki adet farklı görsel ara yüz komutu bulunmaktadır. Bunlar *filterbuilder* ve *fdatool* komutlarıdır. İki ara yüz programı birbirinden farklı görünüme sahip olsalar da fonksiyonel olarak ikisi de aynı şekilde çalışmaktadır. Bu tez çalışmasında *filterbuilder* görsel ara yüz komutu kullanılarak FIR filtre tasarımını gerçekleştirilmiştir.

Matlab'ın komut ara yüzüne *filterbuilder* yazılıp enter'a basıldığında şekil 3.3'teki gibi *filterbuilder* görsel ara yüzü açılmaktadır. İlk olarak karşımıza çıkan pencere bizden filtre cevap türünü seçmemizi istemektedir.

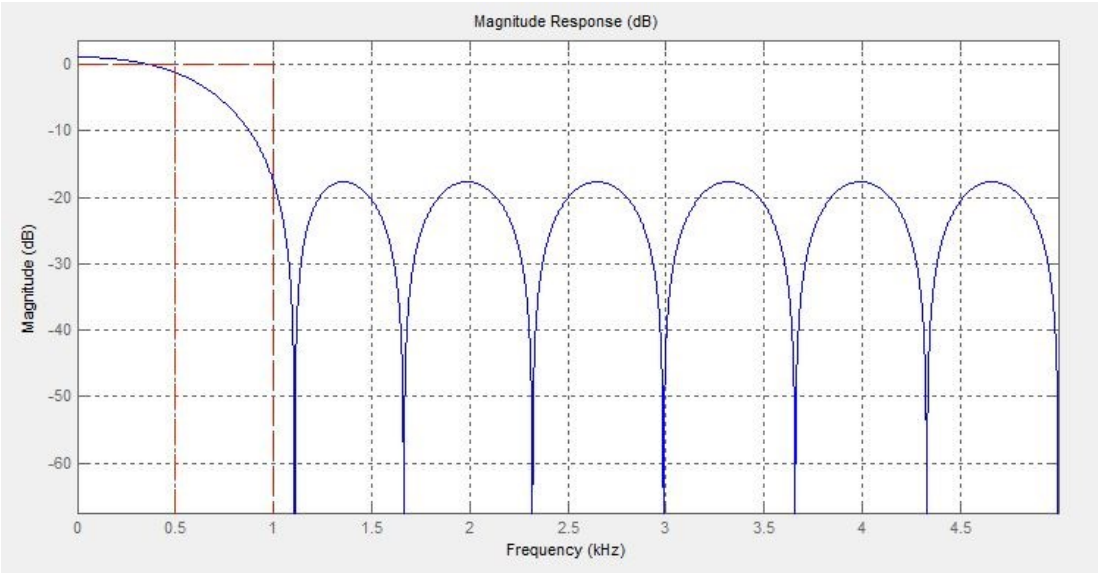
Filter cevap türü olarak sunulan seçenekler *Lowpass, Highpass, Bandpass, Bandstop, Differentiator, Hilbert Transformer, Arbitrary Response, Pulse Shaping, Nyquist,*

Halfband, Cascaded Integrator-Comb, CIC Compensator, Inverse-sinc Lowpass, Octave, Peak, Notch, Comb, Parametric Equalizer, Fractional Delay olarak tanımlanmıştır. Farklı filtre cevap türleri için farklı gereksinimler olduğu için, bir sonraki açılan pencere filtrenin cevap türünün seçimine göre farklılık göstermektedir. Örnek olarak şekil 3.3'de, filtre cevabı olarak *Lowpass* seçildiği takdirde frekans özellik kısmında *Fpass* (geçirme frekansı) ve *Fstop* (durdurma frekansı) seçilmekteyken filtre cevabı olarak *Bandstop* seçildiği takdirde frekans özellik kısmında *Fpass1*, *Fpass2*, *Fstop1*, ve *Fstop2* frekansları seçilmektedir.

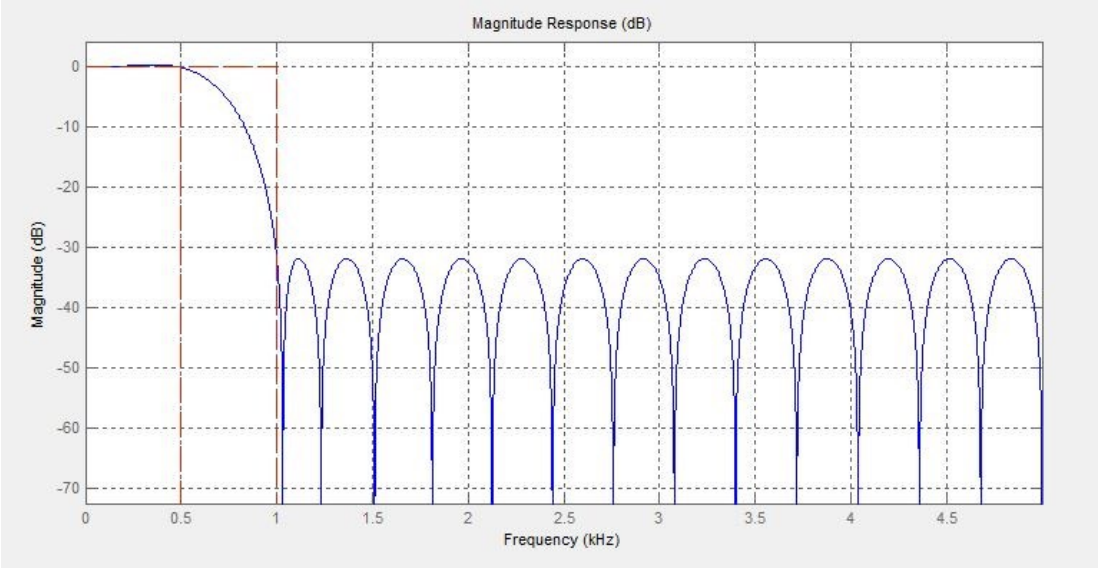


Şekil 3.3. MATLAB filtre tasarımı ana penceresi

Filtrenin gereksinimleri sağlayıp sağlayamadığı *View Filtre Response* düğmesine basılarak görüntülenir. Eğer filtre gereksinimleri girilen sayıda tapla sağlanamıyorsa daha büyük dereceli bir filtre tasarımı yapılıp filtre cevabı tekrar görüntülenebilir. Örnek olarak geçirme frekansı 500 Hz ve durdurma frekansı 1000 Hz olan ve durdurma genliğinin en az 20 dB olmasını istediğimiz bir FIR filtre tasarlayacak olalım. 16 tap kullanarak bu filtreyi gerçekleştirmek istediğimizde şekil 3.4'teki filtre cevabı oluşmaktadır. Bu durumda gereksinimlerin sağlanamadığı gözükmektedir. Tap sayısını 32 yapıp tekrar filtre cevabına baktığımızda ise şekil 3.5'te filtrenin gereksinimleri sağladığı gözlenmektedir.



Şekil 3.4. 16-tap FIR filtre cevabı



Şekil 3.5. 32-tap FIR filtre cevabı

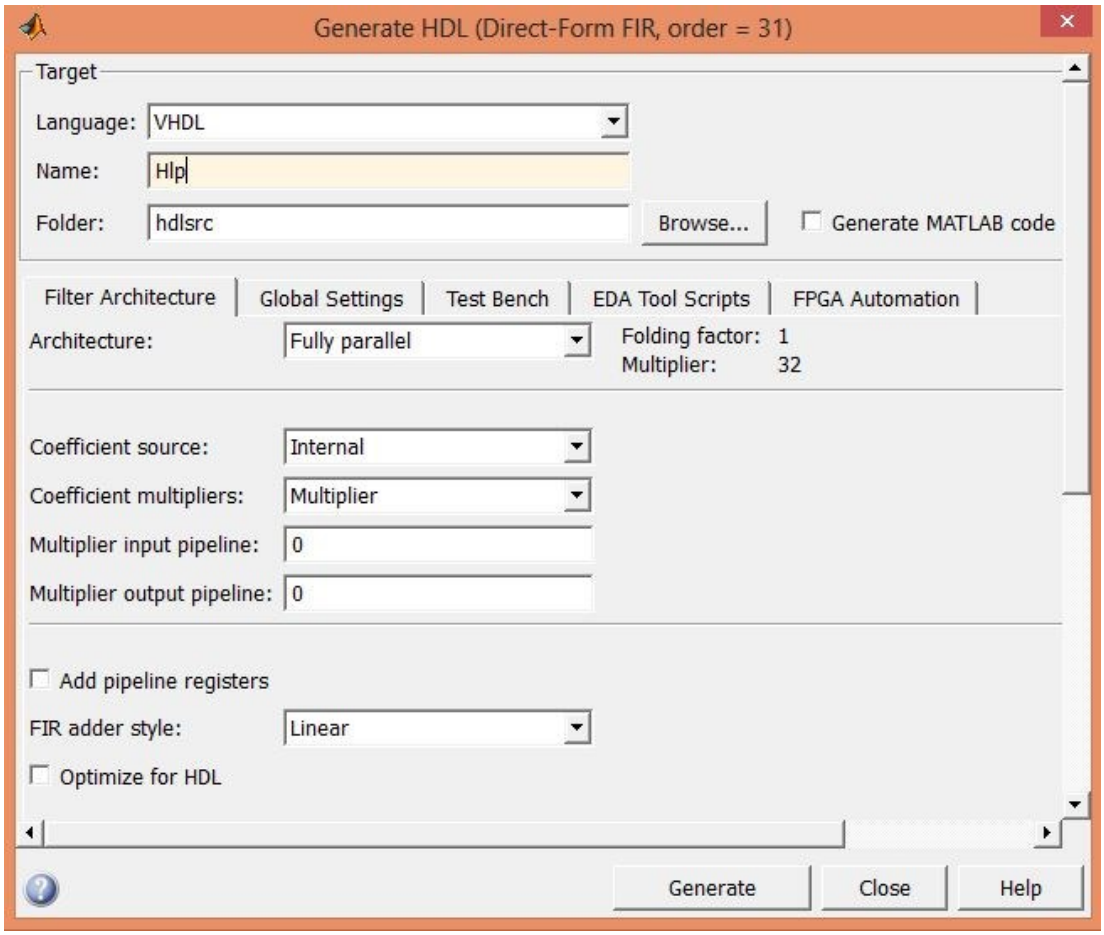
Matlab'da yapılan bu analiz aslında tam olarak gerçekçi sonuçları göstermemektedir. Çünkü Matlab filtre cevabı analizi sırasında filtre giriş çıkışlarını ve filtre katsayılarını *double* veri tipi olarak almaktadır. Ayrıca toplayıcı ve çarpıcı birimlerinin giriş çıkışlarını da 64-bitlik olarak algılamaktadır. Oysa bu varsayımlar bilgisayar ortamı için geçerlidirler. Gömülü bir sistemde 64-bitlik yazmaçlar ve

aritmetik birimler yerine 32-bitlik *float* veri tipleriyle çalışılabilecek ya da *fixed point* olarak isimlendirilen sınırlı bit sayısına sahip kaydedici ve aritmetik birimlerle çalışılmaktadır. Bu yüzden Matlab'da *double* olarak varsayılan filtre katsayıları, filtre giriş çıkış veri tipleri ve aritmetik birimlerin veri tiplerinin nicemlenmesi (quantization) gerekmektedir. Matlab yazılımı nicemleme etkisini de analiz safhasına ekleyip filtre cevabındaki değişimi nicemlemeye göre gösterebilme yeteneğine sahiptir.

Yapılan analiz sonrasında float veri tipi yerine fixed point veri tipi kullanılarak nicemleme yapıldığında filtre cevabında gözle görülür bir değişme görülmemiştir. Fixed point veri tipi kullanımı FPGA lojik hücresi ve DSP makro kaynaklarının tüketimi açısından çok önemli olduğu için *float* veri tipi yerine filtre cevabını etkilemeyecek biçimde ayarlanmış *fixed point* veri tipi kullanılmıştır. DSP makro bloklarındaki çarpıcı birimleri 18-bitlik olduğu için filtre katsayıları 18-bit genişliğinde nicemlenmiştir. Analog ve dijital sinyal dönüşümlerinde en çok kullanılan genişlik olması sebebiyle filtre giriş ve çıkış sinyalleri de 16-bit genişliğinde seçilmiştir.

3.3 VHDL ile Xilinx ISE Üzerinde Sistemin Doğrulanması

Matlab'da istenilen gereksinimleri sağlayan FIR filtre tasarlanıp kullanılacak donanım mimarisine göre nicemleme yapıldıktan sonra FPGA üzerine yüklenecek olan bit dosyasının oluşturulabilmesi için sentezlenebilir HDL (hardware description language) koduna ihtiyaç vardır. Bunun için yine Matlab DSP System Toolbox içerisinde HDL kodu oluşturan bölüm kullanılarak tasarlanan FIR filtrenin HDL kodu oluşturulur. "Generate HDL" düğmesine basıldığında karşımıza çeşitli ayarları yapabileceğimiz şekil 3.6'daki gibi bir pencere açılır.



Şekil 3.6. MATLAB HDL kodu oluşturma penceresi

Bu pencerede HDL kodu oluşturulacak FIR filtrede yapılacak ayarlar vardır. Dil seçeneğiyle Verilog ya da VHDL seçilebilir. Ayrıca mimarinin seri ya da paralel olması, boru hattı eklenmesi, giriş çıkış yazmaçları eklenmesi gibi seçenekler bulunmaktadır. DA mimarisi kullanılmak istenildiğinde de *Architecture* kısmında *Distributed Arithmetic* seçeneğinin seçili olması gerekmektedir.

Matlab'da oluşturulan HDL kodu daha sonra Xilinx'in ISE yazılımıyla gerekli FPGA ürünü seçilerek sentezlenir ve FPGA'nın pin atamalarıyla FIR filtre FPGA üzerinde kullanıma hazır hale gelir.

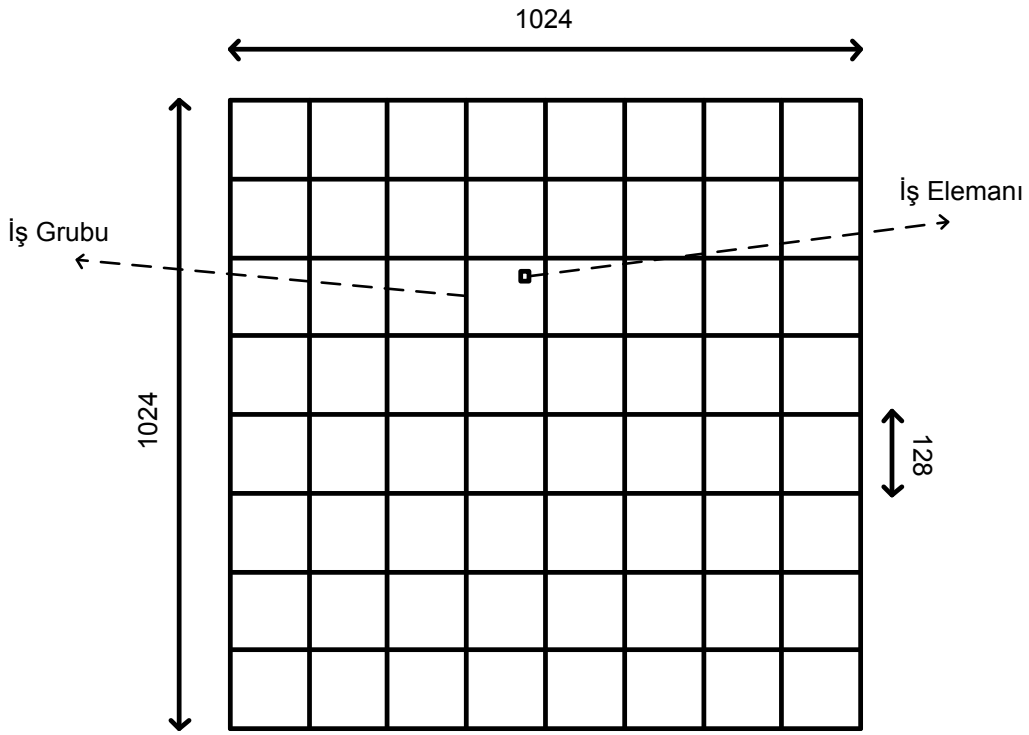
4. GPU'DA OPENCL İLE FIR FİLTRE GERÇEKLENMESİ

Bu bölümde ilk olarak OpenCL hakkında genel bilgiler verilmiş, daha sonra da FIR filtrenin gerçeklendiği heterojen sistemi içeren kart ve sistem tanıtılmıştır. OpenCL kullanılarak gerçeklenen FIR filtre detayları da son olarak anlatılmıştır.

4.1 OpenCL Hakkında Genel Bilgiler

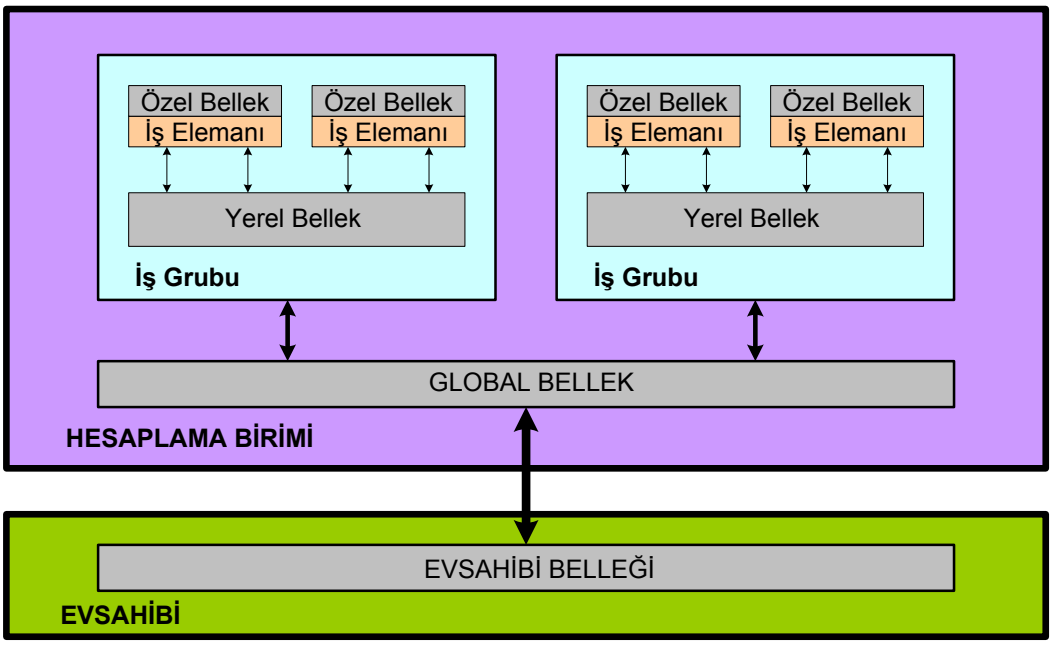
OpenCL (Open Computing Language), veri paralelliği (data-parallel) ve görev paralelliği (task-parallel) gösteren uygulamaların heterojen sistemlerde hızlandırılmasına yönelik hazırlanmış ve kullanılan bir programlama iskeletidir (framework) [7]. Bu amaçla hazırlanmış başka programlama dilleri de mevcuttur. Bunlar arasında en meşhuru NVIDIA tarafından geliştirilen CUDA C programlama iskeletidir. CUDA yalnızca NVIDIA ekran kartı bulunan sistemlerde çalışmaktadır fakat OpenCL platform-bağımsız olarak geliştirilmiştir. OpenCL, NVIDIA, ATI gibi ekran kartlarını desteklediği gibi çok çekirdekli mikroişlemciler ve hatta FPGA ve DSP sistemleri de OpenCL ile programlanabilmektedir.

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. OpenCL'de çalıştırılmak istenilen kernel global olarak nicemlenerek iki ya da üç boyutlu geometrik şekilde ifade edilir. Aynı zamanda iş gruplarının da büyüklükleri ayarlanır ve örnek olarak iki boyutlu bir kernel yapısında matris şeklinde bir yapı oluşur. Şekil 4.1'de bir kernel 1024x1024 boyutunda tanımlanmıştır ve iş gruplarının büyüklüğü de 128x128'dir.



Şekil 4.1. OpenCL'de iki boyutlu 1024x1024 büyüklüğünde ve 128x128'lik iş gruplarına sahip bir kernel yapısı

OpenCL bellek işlemlerini kolaylaştırmak için genel amaçlı çok mikroişlemcilerde olduğu gibi bellek hiyerarşisi geliştirmiştir. OpenCL iki ana kısımdan oluşmaktadır: Kernel atamalarının ve ayarlarının yapıldığı ev sahibi (host) ve kernelin çalıştırıldığı hesaplama birimi (compute device). Ev sahibi genelde CPU'dur ve ev sahibinin kullandığı bellek birimi ev sahibi belleği (host memory) olarak tanımlanır. Hesaplama birimi ev sahibiyle haberleşirken tek bir ara yüz belleğinden haberleşir. Bu ara yüz belleğine global bellek adı verilir. Global bellek bütün iş gruplarına hesaplama yapılacak veri aktarımı ve iş gruplarında hesaplanmış verilerin alımından sorumludur. İş grupları içerisinde iş elemanlarının paylaştığı bir bellek bulunur ve bu bellek yerel bellek (local memory) olarak adlandırılır. İş elemanları birbirleriyle yerel bellek aracılığıyla haberleşirler. OpenCL'de ayrıca her bir iş elemanının kendisine has özel bellekleri (private memory) vardır. OpenCL'de hiyerarşi çok katıdır; yani ev sahibinden bir veriyi özel belleğe getirmek için verinin sırasıyla global bellek ve yerel bellekten geçmesi gerekir. Fakat bazı durumlarda uygulamaya özel olarak global bellekten yazmaçlara erişilebilmektedir. Global bellek en fazla veri alanına sahiptir fakat en yavaş erişim de global belleğe erişimdir. Özel bellek ise en az veri alanına sahip olmakla birlikte en hızlı erişime de sahiptir. OpenCL bellek düzeni şekil 4.2'de verilmiştir.



Şekil 4.2. OpenCL bellek hiyerarşi yapısı

OpenCL'de paralelleştirme işlemini basit bir örnekle açıklamak gerekirse şekil 4.3'teki kod n elemanlı iki dizinin aynı indisteki elemanlarını çarparak farklı bir dizide kaydetmektedir. Geleneksel sıralı mikroişlemci yapısında aşağıda yazılan kod sırasıyla karşılıklı olarak iki diziden her bir elemanı çarpacaktır ve c dizisine kaydedecektir.

```
void gelenekselCarpma (int n, const float *a, const float *b, float *c) {
        int i;
        for (i = 0; i < n; i++)
                c[i] = a[i] * b[i];
}
```

Şekil 4.3. Geleneksel programlama dilinde dizi çarpma işlemi

Gelenksel çarpma işlemini OpenCL destekli bir hesaplama biriminde çalıştırmak için kodu biraz değiştirmek gereklidir. Kernelde çalışacak olan fonksiyonu belirtmek için

fonksiyon tanımının önüne *__kernel* ibaresi getirilir. Dizi değişkenlerinin atandığı belleği tanılmalak için değişkenlerin önüne *__global* ibaresi getirilir. Geleneksel C dilindeki *for* döngüsü kaldırılır. Geleneksel çarpma işleminin OpenCL kerneli olarak yazılmış hali şekil 4.4'te verilmiştir. Bu kernelde bulunan çarpma işlemi n adet iş elemanı üzerinde çalışmaktadır.

```
__kernel void paralelCarpma ( __global const float *a, __global const float *b,
        __global float *c )     {
        int id = get_global_id(0);
        c[id] = a[id] * b[id];
}
```

Şekil 4.4. OpenCL'de dizi çarpma işlemi kerneli

4.2 Frescale i.MX 6Quad Kartı Hakkında Bilgiler

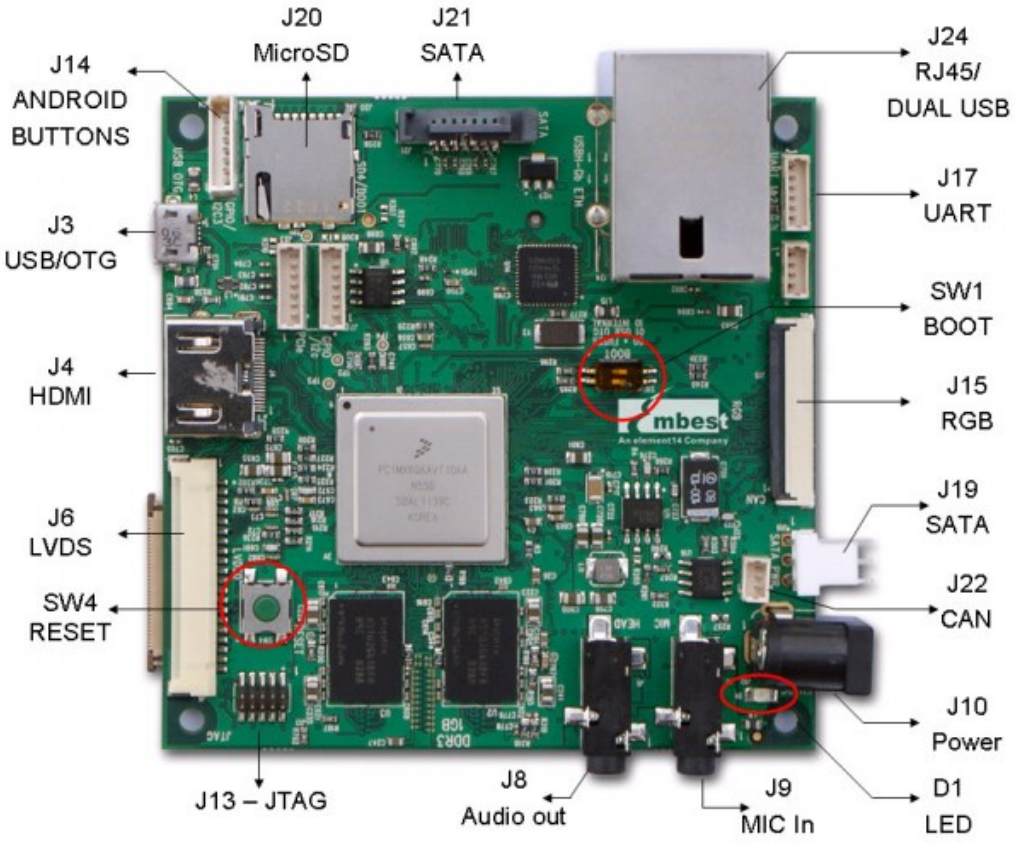
OpenCL'de yazılacak olan FIR filtre programının çalışacağı platform olarak Freescale® firmasının i.MX 6Quad isimli elektronik sistemi seçilmiştir [8]. Oldukça gelişmiş yapıda olan i.MX 6Quad sistemi özellikle netbook, PDA, SBC ve diğer mobil platformlarda kullanılmak üzere tasarlanmıştır. Üzerinde dış dünya bağlantısı olarak 10/100/Gb eternet portu, SATA-II, HDMI v1.4, LVDS, paralel RGB ara yüzü, dokunmatik ekran ara yüzü, analog audio giriş/çıkış, SD kart ara yüzü, USB, seri port, JTAG, kamera ara yüzü ve Android için tuş takımı girişleri bulunmaktadır.

i.MX 6Quad üzerinde işlemci birimi olarak ARM Cortex A9 bulunmaktadır. Bu işlemci dört çekirdeklidir ve 1.2 GHz saat sıklığında çalışmaktadır. OpenCL platformunun paralel işleri çalıştıracağı grafik işlemcisi ise i.MX 6Quad üzerinde gömülü olarak bulunan Vivante®'nin GC2000 işlemcisidir. GC2000 grafik işlemcisinin özellikleri çizelge 4.1'de verilmiştir. Sistem mobil bir platform olduğu için OpenCL'nin gömülü sistemler için düzenlediği *embedded profile* kullanılmıştır. OpenCL'nin gömülü profil versiyonunda 3D görüntü, 64-bit integer sayı ve atom fonksiyonu gibi özellikler performans ve enerji kazancı için kısıtlanmıştır.

Çizelge 4.1. Vivante GC2000 grafik işlemcisi özellikleri

VIVANTE GC2000 GRAFİK İŞLEMCİSİ	
Alet Tipi	GPU
OpenCL Versiyonu	OpenCL C 1.1
Profil	EMBEDDED_PROFILE
İşlemci Ünitesi	4
En Çok İş Elemanı Boyutu	3
Bir Boyuttaki En Çok İş Elemanı Sayısı	1024
Saat Sıklığı	500 MHz
Global Bellek Büyüklüğü	64 Mbyte
Global Bellek Önbelleği	YOK
Yerel Bellek Büyüklüğü	1 Kbyte
Yerel Bellek Tipi	Global
En Çok Sabit Buffer Büyüklüğü	4 Kbyte

i.MX 6Quad elektronik kartının üstten görüntüsü şekil 4.5‘te verilmiştir.



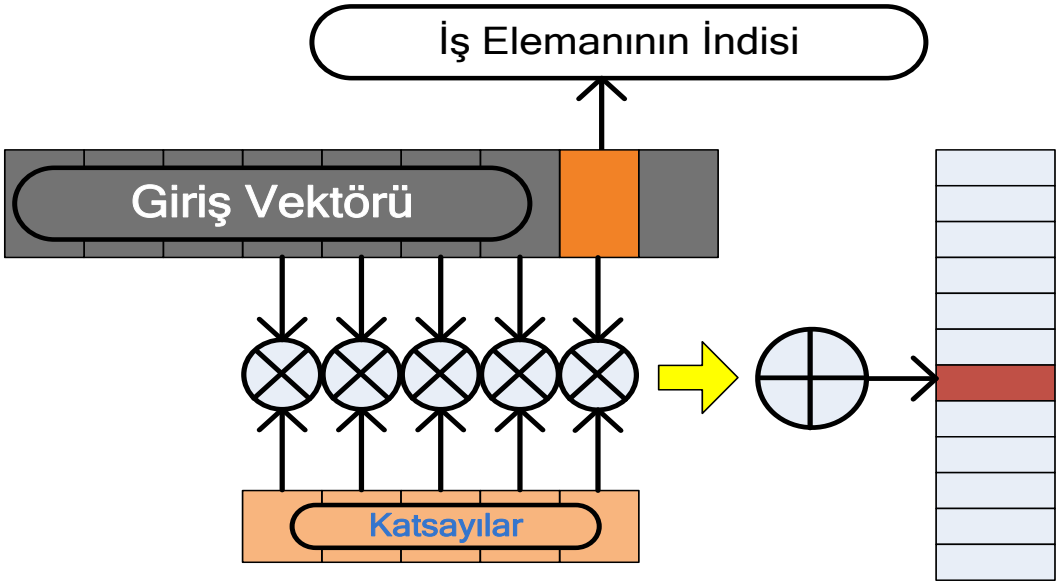
Şekil 4.5. i.MX 6Quad kartının görüntüsü

4.3 OpenCL Kullanarak FIR Filtre Tasarımı

FPGA tasarımında olduğu gibi GPU tasarımında da altı farklı tap büyüklüğünde FIR filtre seçilmiştir. Tap büyüklükleri 16, 32, 64, 128 ve 256'dır. FIR filtre katsayıları Matlab DSP System Toolbox kullanılarak hesaplanmıştır. Sistem giriş vektörü için de Matlab'da 500 Hz'lik sinüs sinyaline 2000-4500 Hz aralığında çeşitli büyüklüklerde rastgele gürültü eklenmiş ve 10 kHz hızında örneklenerek oluşturulmuştur. Her bir örnek 32-bit *float* veri tipindedir.

i.MX 6Quad elektronik bordu üzerinde OpenCL kullanılarak çalıştırılacak olan FIR filtre için üç farklı kernel tasarlanmıştır. İlk tasarlanan kernel kodu en yalın olanıdır ve herhangi bir eniyileme içermemektedir. İş elemanı boyutu diğer kernellerde olduğu gibi ikidir. Vivante'nin OpenCL sürücüsü her bir boyut için en fazla 64K iş elemanı desteklemekte olduğundan dolayı tek bir kernelde olabildiği kadar çok fazla çıktı alabilmek gerekmektedir. Basit kernel yapısında her bir iş elemanı, bir giriş vektöründeki örnekler buna tekabül eden filtre katsayısının çarpma ve toplama

hesaplamalarını yapıp tek bir çıkış elemanını hesaplamaktadır. Basit kernel yapısının çalışma prensibi şekil 4.6'da açıklanmıştır.



Şekil 4.6. Basit FIR filtre kernel çalışma mimarisi

İkinci kernel gerçeklenmesinde her bir çıkış örneği için yapılan çarpma işlemi simetrik FIR filtre metodolojisi kullanılarak yarıya indirilmiştir. Bu metotta iki farklı giriş örneği ayrı ayrı aynı katsayı değeriyle çarpılıp toplanması yerine giriş örnekleri toplanarak tek bir çarpma işlemi gerçekleştirilir. Şekil 4.7'de simetrik FIR filtre kerneli verilmiştir.

```
int gid = get_global_id(1)*get_global_size(0) + get_global_id(0);
int offset = gid  +  1;
gid += num_coeffs_half << 1;

//do the computation
for(int i = 0; i < num_coeffs_half; i++){
  if(gid >= 0){
    //sum symmetric elements and multiply once
    rslt += coeffs[i]*(input[gid--] + input [offset++]);
  }
 else { return; }
}
output[gid + num_coeffs_half] = rslt;
```

Şekil 4.7. Simetrik FIR filtre kernel kodu

Kernel için yapılan son eniyilemede kernel yapısı vektör tipi değişkenleriyle çalışacak şekilde düzenlenmiştir, böylece OpenCL derleyicisi vektör tipi buyruklar üretecek ve bu buyruklar da özel amaçlı donanım birimlerinde çalışabilecektir. Intel'in OpenCL derleyicisi otomatik olarak derleme sırasında kerneli vektörize etmektedir. Fakat Vivante OpenCL sürücüsü bunu otomatik olarak yapmamaktadır. Dışarıdan bu şekilde yapılacak bir vektörleştirme işlemi böylece i.MX 6Quad kartında çalışan FIR filtre için büyük avantaj kazandıracaktır. Vektör tipi kernelde *float4* veri yapısı kullanıldığı için her bir thread dört tane çıkış örneği hesaplayacaktır. Şekil 4.8'de vektör veri tipleri kullanılarak yazılmış FIR filtre kerneli verilmiştir. Artan bellek ve yazmaç kullanımına rağmen vektör kerneli performansta çok etkili sonuçlar vermiştir.

```
int gid = get_global_id(1)*get_global_size(0) + get_global_id(0);
int offset = gid;
if(gid < chunk_size){
//do the computation
  for(int i = taps_quarter; i >= 0; i--){
    if(offset >= 0){
        res_1 += coeffs_1[i]*input[offset];
        res_2 += coeffs_2[i]* input [offset];
        res_3 += coeffs_3[i]* input [offset];
        res_4 += coeffs_4[i]* input [offset--];
    }
    else{break;}
}
output[gid] = (float4)(dot(res_1, (float4)(1.0f)), dot(res_2,(float4)(1.0f)),
dot(res_3, (float4)(1.0f)), dot(res_4, (float4)(1.0f)));
```

Şekil 4.8. Vektör FIR filtre kernel kodu

İlk bakışta simetrik FIR filtre tasarımında vektör veri tipini kullanmak mantıklı ve etkili bir uygulama olarak gözükmektedir. Fakat uzak indisteki verilerin toplanması thread eşleşmesi sistemini bozmaktadır ve bellek erişimini düzensizleştirmektedir. Thread eşleşmesini düzeltmek için farklı pozisyonlardaki n adet elemanı vektör verisine yükleyen *vloadn* isimli kernel fonksiyonu bir çözüm olarak gözükse de vektör veri tipini direk kullanmak kadar performans sağlamamaktadır.

5. FPGA VE GPU SONUÇLARININ KARŞILAŞTIRILMASI

FPGA ve GPU'da alınan sonuçların karşılaştırılması hem başarım düzeyinde hem de enerji tüketimi düzeyinde yapılmıştır. Başarım karşılaştırması için saniyede hesaplanan çıktı örneği kullanılmıştır. GFLOPS ya da saniyede hesaplanan bit sayısı yani throughput gibi kavramlar FPGA ve GPU karşılaştırılmasında yanlış sonuçlar ve varsayımlara sebep olabilmektedir. GPU platformu OpenCL arayüzünü kullanmıştır ki her bir filtre katsayısı, giriş örnekleri ve de sonuçlar 32-bit float veri tipi olarak tanımlanabilmektedir. FPGA mimarisinde ise nicemlemenin sonuca etkisinin göz ardı edilebilecek seviyede olmasından dolayı fixed-point veri tipi kullanılmış, filtre katsayıları, giriş ve çıkış örnekleri farklı hassasiyette tanımlanarak avantaj sağlanmıştır. DSP makro bloklarda bulunan çarpıcı birimleri 18-bitlik çalışmaktadır, bu sebeple filtre katsayıları 18-bit çözünürlüğünde seçilmiştir. Filtre giriş ve çıkış sinyalleri piyasada en çok bulunan analog-dijital dönüştürücü çözünürlüğü olan 16-bitlik çözünürlükte seçilmiştir. GPU ve FPGA için nicemleme analiz edilmiş ve kullanılan veri tiplerinin ve çözünürlüklerin iki sistem için de yeterli olduğu görülmüştür. Bu durumda eğer saniyede hesaplanan bit sayısı (bps) gibi bir throughput değerinin karşılaştırma olarak alsaydık bir çıkış örneğinde GPU 32-bit, FPGA 16-bit üretecekti ve GPU FPGA'nın 2 katı hızında gözükecekti. Hâlbuki iki sistem de sadece bir örnek ürettiği için hızlarının eşit olması gereklidir. Bu yüzden saniyede hesaplanan örnek sayısı en mantıklı karşılaştırma parametresi olarak gözükmektedir.

FPGA'da çalışan FIR filtrenin başarım ölçümü için Xilinx'in ISIM isimli benzetim aracı kullanılmıştır. Bu benzetim aracı her bir saat vuruşunda FPGA'da tanımlanmış bütün sinyallerin anlık değerlerini gösterebilme yeteneğine sahip olduğu için filtre aşamaları rahatlıkla takip edilebilir ve filtrenin sonucu ne zaman hesapladığı da görülebilmektedir. Filtrenin saat sıklığı da yine Xilinx'in ISE yazılımıyla sentez raporunda görülmektedir. GPU'da gerçeklenen FIR filtre başarımını analiz etmek için sistemin saat değeri bir fonksiyon yardımıyla alınarak filtrenin çalışma süresi hesaplanabilmiştir.

FPGA platformunun güç tüketimini hesaplamak için Xilinx'in XPower Estimator isimli yazılımı kullanılmıştır [9]. Sentezlenmiş olan mantıksal kapılar, yazmaçlar, DSP bloklar ve saat sıklığı yazılımda tanımlanmış ve FIR filtrenin tüm birimleriyle birlikte sistem olarak güç tüketimi hesaplanmıştır. XPower Estimator isimli yazılım FPGA'nın durağan güç tüketimini (statik power dissipation) de tahmin edebilmektedir, böylece daha gerçekçi bir ölçüm yapılmış olmaktadır.

GPU'da çalışan FIR filtrenin güç tüketimi i.MX 6Quad sisteminin çektiği akım ölçülerek hesaplanmıştır. Sistemin çektiği maksimum akım gözlenmiştir ve güç

kaynağının voltaj değeriyle çarpılarak güç tüketimi hesaplanmıştır. FIR filtrenin enerji tüketimi ise çalışma zamanı ile güç tüketiminin çarpılmasıyla elde edilir.

FPGA ve GPU'nun enerji tüketimi çizelge 5.1'de verilmiştir.

Çizelge 5.1. FIR filtre uygulaması FPGA ve GPU için enerji harcamaları

FIR ORDER	Energy Harcaması (Joule)					
	FPGA			GPU		
	Direct Form	Symmetric	DA	Scalar	Symmetric	Vector
16	4,75	4,06	3,41	105,02	85,26	29,61
32	6,77	5,28	4,45	202,76	158,84	71,56
64	10,7	7,51	6,21	405,35	305,21	182,3
128	95,04	11,66	9,39	817,95	699,92	388,34
256	389,4	103,83	14,99	1636,25	1833,01	753,53
512	1607,57	443,43	N/A	3285,71	4019,44	1369,71

FPGA ve GPU'nun başarım karşılaştırması çizelge 5.2'de verilmiştir.

Çizelge 5.2. FIR filtre uygulaması FPGA ve GPU için throughput değerleri

FIR ORDER	Throughput (Saniyede Milyon Örnek)					
	FPGA			GPU		
	Direct Form	Symmetric	DA	Scalar	Symmetric	Vector
16	67,84	59,62	86,68	9,81	12,08	34,79
32	68,26	59,62	86,68	5,08	6,48	14,39
64	68,26	59,62	86,68	2,54	3,37	5,65
128	7,85	59,62	86,68	1,26	1,47	2,65
256	3,92	7,85	86,68	0,63	0,56	1,37
512	1,93	3,92	N/A	0,31	0,26	0,75

Distributed arithmetic (DA) metoduyla sentezlenen FIR filtrenin en yüksek throughput değerine sahip olduğu gözükmektedir. DA metodunda boru hattı tekniği kullanılmaktadır ve çarpma işlemleri tamamen paralel olarak gerçekleştirilmektedir. Filtrede en çok zaman alan işlem çarpma işlemi olduğu için sistemin saat sıklığı çarpma işlemine göre belirlenmektedir. Çarpma işlemleri paralel olarak gerçeklendiği için filtre seviyesi büyüse bile saat sıklığı değişmemekte, böylece throughput değeri de azalmamaktadır. DA tekniği doğrudan-form ve simetrik FIR filtreye göre daha fazla throughputa sahip olsa da 512 taplı filtre DA tekniği ile FPGA'nın mantık kapılarının yetersizliğinden dolayı sentezlenememiştir.

Doğrudan-form ve simetrik FIR filtre gerçeklenmesinin başarımı filtreyi sentezlemek için gereken çarpıcı sayısına bağımlıdır. Doğrudan-yapıda tasarımda 64-taplı FIR filtre tasarımına kadar bütün filtrelerde çarpma işlemleri paralel olarak yapılabilmektedir. Eğer çarpma işlemleri paralel olarak gerçekleştirilebilirse doğrudan-form simetrik FIR filtre tasarımına göre daha çok throughput sağlamaktadır. Bunun nedeni simetrik-form'da gereken fazladan toplama işlemidir. Fakat çarpma işlemleri için serileşme başladığı zaman, doğrudan-form için 128-tap, simetrik-form doğrudan-form'a göre daha başarılı hale gelmektedir. Enerji tüketimi serileşme yüzünden doğrudan-form'da daha yüksek çıkmaktadır, çünkü FIR filtre işleminin bitmesi için gereken süre daha fazladır. Çizelge 5.1 ve çizelge 5.2 incelendiğinde DA FIR filtre gerçeklenmesi en az enerji harcayan ve en çok throughputa sahip olan tasarım olarak gözükmektedir. Fakat, 512-tap FIR filtre için DA metodu FPGA'nın sahip olduğundan daha fazla kaynağa ihtiyaç duyduğu için filtre gerçeklenememiştir.

GPU tarafından bakıldığında ise vektor veri tipleri kullanılarak gerçeklenen FIR filtre bütün seviyedeki FIR filtreler için en yüksek throughput değerine sahiptir. GPU'nun FPGA'ya karşı en büyük avantajı daha yüksek seviyeli FIR filtrelerin de gerçeklenebilir olmasıdır. Fakat throughput değeri FIR filtre seviyesi yükseldikçe dramatik olarak düşmektedir ve enerji tüketimi de yükselmektedir. Çizelge 5.1 ve çizelge 5.2'ye baktığımızda FPGA'nın GPU'ya göre hem başarım olarak hem de enerji harcaması olarak daha iyi olduğunu görmekteyiz. GPU'daki en etkili yöntem olan vektör veri tipi kullanılmasına rağmen FPGA teknikleri bütün seviyedeki filtreler için GPU'dan daha başarılıdır. FPGA'ların enerji harcaması yönünden de GPU'lardan daha etkili bir araç olduğu gözükmektedir. Bu sonuçlara göre FIR filtre uygulaması için mobil platformlarda FPGA kullanımı GPU kullanımına göre daha avantajlıdır. Eğer çok fazla işlemci birimi olan yüksek güç tüketimli GPU'ları kullanıyor olsaydık FPGA'lara göre daha iyi başarım elde edebilirdik. Fakat bu GPU'ları mobil sistemlerde kullanmak imkânsızdır.

6. GEÇMİŞ ÇALIŞMALAR

Literatürde heterojen sistem ve FPGA tabanlı, enerji kazancına yönelik FIR filtre tasarımları oldukça çok sayıda bulunmaktadır. [10]'de yapılan çalışmada tap sayıları 8'den 128'e kadar değişken olan FIR Hilbert transform filtreleri Virtex-5 FPGA aracı üzerinde dijital ultrason görüntüleme amaçlı kullanılmıştır. 64-taplı bir FIR filtre örneği üzerinden gidilmiş ve bunun güç tüketimi 814 miliwat olarak hesaplanmıştır. Gömülü sinyal işleme işleri için kullanılması planlanan çok-üniteli heterojen bir sistem [11]'de verilmiştir. Sistem performansının arttırılması için dört adet Virtex-5 FPGA cihazı bir adet 8641D Freescale™ işlemcisine bağlanmıştır. Sistem FPGA'larda 550 MHz ve CPU'da 1.5 GHz'lik bir performansa ulaşabilmiştir. [12]'te yapılan çalışmada göğüs kanseri tedavisi için göğüs yapı resimleri oluşturulmasına yönelik bir çalışmada FPGA tabanlı bir veri toplama sistemi geliştirilmiştir. Sayısal filtreleme işlemlerinde zaman bazlı yaklaşım olan metot yazılımsal metoda göre 6.9 kat performans artışı sağlarken frekans bazlı yaklaşımda 2.9 katlık başarım artışı sağlanmıştır. FPGA üzerinde çalıştırılacak sinyal işleme uygulamaları için [13]'te yapılan çalışmalarda uygulamaya özel bir işletim sistemi ve çok işlemcili bir sistem tasarlanmıştır. FPGA platformu olarak Virtex-II Pro ve işlemci birimleri olarak Xilinx'in soft olarak sağladığı Microblaze işlemcisi seçilmiştir. Yapılan çalışmalar sonrasında sistem FFT algoritmalarını başarıyla çalıştırmıştır. [14]'te yapılan çalışmada ise yine FPGA tabanlı gömülü bir çok çekirdekli sistem oluşturulmuştur. GSM sinyal işleme uygulamaları OpenMP ve Pthread ara yüzleri kullanılarak karşılaştırılmıştır.

[15]'da yapılan çalışmada FPGA'da karışık-tabanlı FFT çekirdeği tasarlanmış ve GPU ile güç ve başarım karşılaştırılması yapılmıştır. GPU'da FFT analizi için CUDA'nın CuFFT kütüphanesi kullanılmıştır. GPU FPGA'dan daha başarılı çıkmasına rağmen FPGA'ların daha az güç tükettiği gözlemlenmiştir. Gerçek zamanlı gömülü görüntü işleme uygulaması için FPGA ve GPU'lar [16]'de karşılaştırılmıştır. Yapılan deneyler sonrasında GPU FPGA'dan daha başarılı çıkmıştır. GPU platformu olarak NVIDIA'nın GTX 280 ekran kartı ve FPGA platformu olarak da Xilinx'in Virtex-5 ürünü seçilmiştir. Daha fazla FPGA ve GPU arasındaki başarım ve enerji tüketimine yönelik karşılaştırma çalışmaları [17-21]'de bulunabilir.

Daha önceki çalışmaların sayısı çok olsa da yapılan karşılaştırmalar genellikle aşırı güç tüketen ve çok güçlü GPU ve FPGA platformları için yapılmıştır. Gömülü sistemleri hedef alan çalışmalarda bile araştırmacılar çok güç tüketen sistemleri kullanarak sonuçlar almışlardır. Bu tezde yapılan çalışmada ise mobil platformlarda çalıştırılacak olan FIR filtre uygulamaları için iki adet az güç tüketimine yönelik

olarak üretilmiş sistem ele alınmıştır. Yapılan çalışmaların çoğunda GPU'lar FPGA'lardan daha başarılı olarak gözükse de araştırmacılar genellikle aşırı güç tüketen ve çok güçlü GPU'lar seçerek bu karşılaştırmaları yapmışlardır. Bu tez çalışmasında ise GPU platformu olarak üzerinde düşük profilli Vivante GC2000 grafik işlemcisi olan Freescale'in i.MX 6Quad bordu seçilmiştir. Daha önceki çalışmalara nazaran Spartan 3A DSP FPGA, heterojen sistemden daha başarılı çıkmıştır.

7. SONUÇLAR

Sinyal işleme uygulamalarının bulunduğu mobil ya da genel amaçlı sistemlerde FIR filtreler bu sistemlerin vazgeçilmez elemanlarıdır. FIR filtreler çarpma işlemlerinin birbirinden bağımsız ve paralel olarak yapılabilme avantajına sahiptir. FIR filtre uygulamalarında throughputu arttırmak ve başarım artışı sağlamak için paralel mimarilere sahip olan FPGA ve GPU'lar çarpma işlemlerinin paralelleştirilmesi için tercih edilmektedirler. Batarya süresini arttırmak için düşük enerji harcaması mobil sistemler için kaçınılmaz bir gereksinimdir. Bu yüzden mobil sistem üzerinde çalışacak donanım ve yazılımlarda başarımın yanında düşük enerji harcaması da dikkate alınmalıdır. Literatürdeki daha önceki çalışmalara baktığımızda bu çalışmaların yüksek enerji harcamasını göz ardı ederek çok güçlü FPGA ve GPU'ları ele alarak karşılaştırma yaptıklarını görmekteyiz.

Bu tez çalışmasında çeşitli filtre seviyelerinde bulunan FIR filtre uygulamalarının enerji harcamaları ve başarım yönlerinden karşılaştırmak üzere düşük güç tüketimli bir Spartan 3A DSP FPGA'sı ve mobil uygulamalara özel olarak tasarlanmış düşük profilli bir GPU seçilmiştir. Hem FPGA platformu hem de GPU platformu için 3 farklı mimari tasarımı tasarlanmış ve uygulanmıştır. Sonuçlar bize Spartan 3A DSP FPGA'nın GPU sistemine göre daha yüksek performanslı ve daha az güç tüketimli olduğunu göstermektedir. Daha önceki çalışmalara baktığımızda genellikle GPU'ların FPGA'ları başarım yönünden geride bıraktığını görmekteyiz. Fakat bu araştırmalarda mobil platformlara uygun olmayan çok enerji harcayan çok güçlü GPU'lar kullanılmıştır. Bu tarz karşılaştırma araştırmalarında en önemli noktalardan birisi de karşılaştırılan platformların birbirine yakın profillerde olmasıdır, aksi takdirde adil bir karşılaştırma yapılması imkânsızdır.

KAYNAKLAR

[1] S. Winder, "Analog and digital filter design," Elsevier Science, Second Edition, 1997.

[2] J. W. Cooley, J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, 1965.

[3] T. C. Denk and K. K. Parhi, "Synthesis of folded pipeline architectures for multirate DSP algorithms," *IEEE Transactions of Very Large Scale Integration Systems*, Vol. 6, No. 4, pp. 595-607, Dec. 1998.

[4] Xilinx, Spartan 3A DSP FPGA Family Data Sheet, Oct. 2010.

[5] Xilinx, XtremeDSP DSP48A for Spartan-3A DSP FPGAs User Guide, Jul 2008.

[6] P. Longa and A. Miri, "Area-efficient FIR filter design on FPGAs using distributed arithmetic," in *Proc. Int. Symp. on Signal Processing and Information Theory*, 2006.

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

[8] Freescale, i.MX 6Quad User Guide, Nov. 2012.

[9] Xilinx, XPower Estimator User Guide, Jan. 2012.

[10] M. A. Hassan, A. M. Youssef, and Y. M. Kadah, "Embedded digital signal processing for digital ultrasound imaging," in *Proc. IEEE Radio Science Conference*, pp. 1-10, 2011.

[11] W. Changrui, C. Fan, and C. Huizhi, "A high-performance heterogeneous embedded signal processing system based on serial RapidIO interconnection,". in *Proc. IEEE Int. Conf. on Computer Science and Information Technology,* 2010.

[12] M. Birk, S. Koehler, M. Balzer, M. Huebner, N. V. Ruiter, and J. Becker, "FPGA-based embedded signal processing for 3-D ultrasound computer tomography," *IEEE Trans.on Nuclear Science*, vol. 58, issue 4, part 1, pp. 1647–1651, 2011.

[13] F. M. Vallina, N. Jachimiec, and J. Saniie, "Multiprocessor and operating system design for signal processing on an FPGA," in *Proc. IEEE Int. Conf. on Electro/Information Technology*, 2007.

[14] T. Liu, Z. Ji, Q. Wang, and S. Zhu, "Research on efficiency of signal processing on embedded multicore system," in *Proc. Int. Conf. on Pervasive Computing Signal Processing and Applications,* 2010.

[15] B. Duan, W. Wang, X. Li, C. Zhang, P. Zhang, and N. Sun, "Floating-point mixed-radix FFT core generation for FPGA and comparison with GPU and CPU," in *Proc. Int. Conf. on Field-Programmable Technology,* 2011.

[16] A. Pacholik, M. Müller, W. Fengler, T. Machleidt, and K. H. Franke "GPU vs FPGA: Example application on white light interferometry," in *Proc. Int. Conf. on Reconfigurable Computing and FPGAs,* 2011.

[17] D. Llamocca, C. Carranza, and M. Pattichis, “Seperable FIR filtering in FPGA and GPU implementations: Energy, performance, and accuracy considerations,” in *Proc. Int. Conf. on Field Programmable Logic and Applications*, Sept. 2011.

[18] K. Pauwels, M. Tomasi, J. Diaz, E. Ros, and M.M. Van Hulle, “A comparison of FPGA and GPU for real-time phase-based optical flow, stereo, and local image features,”*IEEE Transactions on Computers*, July 2012.

[19] S. Asano, T. Maruyama, and Y. Yamaguchi, “Performance comparison of FPGA, GPU, and CPU in image processing,” *IEEE Trans. Image Processing*, vol. 16, no. 3, pp. 879-884, Mar. 2007.

[20] Y. Zhang, Y.H. Shalabi, R. Jain, K.K. Nagar, and J.D. Bakos, “FPGA vs. GPU for sparse matrix vector multiply,” *Int. Conf. on Field-Programmable Technology*, Dec. 2009.

[21] S. Che, J. Li, J.W. Sheaffer, K. Skadron, and J. Lach, “Accelerating compute-intensive applications with GPUs and FPGAs,” *Symposium on Application Specific Processors*, June 2008.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : AYKENAR, Mehmet Burak

Uyruğu : T.C.

Doğum tarihi ve yeri : 10.07.1987 Çanakkale

Medeni hali : Evli

Telefon : 0 (505) 223 64 57

Faks : 0 (312) 292 42 90

e-mail : mbaykenar@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Y. Lisans	TOBB ETÜ/Bilgisayar Müh.	2013
Lisans	ODTÜ/Elektrik-Elektronik Müh.	2011

İş Deneyimi

Yıl	Yer	Görev
2012-...	ROKETSAN A.Ş.	Mühendis
2011-2012	TOBB Ekonomi ve Teknoloji Üniversitesi	Araştırma Görevlisi

Yabancı Dil

İngilizce (TOEFL iBT -105)

Arapça (Orta Seviye)

Yayınlar

Aykenar, M.B.; Ozgur, 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, Mehmet Burak, "Analysis of FPGA based recursive and non-recursive digital filters according to hardware cost and performance," Signal Processing and Communications Applications Conference (SIU), 21st , April 2013.

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.