KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

100GBİT 256 NOKTA RADİX-4 FFT'NİN FPGA İLE GERÇEKLENMESİ

GÖKHAN POLAT

KOCAELİ 2014

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

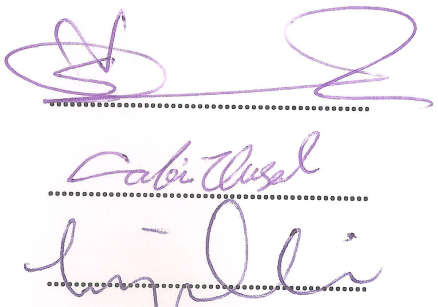
100GBİT 256 NOKTA RADİX-4 FFT'NİN FPGA İLE GERÇEKLENMESİ

GÖKHAN POLAT

Yrd.Doç.Dr. Sıtkı ÖZTÜRK
Danışman, Kocaeli Üniv.

Doç.Dr. Cabir VURAL
Jüri Üyesi, Marmara Üniv.

Doç.Dr. Kemal GÜLLÜ
Jüri Üyesi, Kocaeli Üniv.



Tezin Savunulduğu Tarih: 21.01.2014

ÖNSÖZ ve TEŞEKKÜR

Günümüzde kullanıcıların veri ihtiyaçları giderek artmaktadır. Kullanıcıların veri transferi ihtiyaçlarını karşılamak adına klasik bakır kablo yeterli gelmemeye başladığından fiber hatlar tercih edilmeye başlanmıştır.

Optik hatlarda yüksek hızlı veri iletimi dezavantaj olarak bazı bozunumları beraberinde getirir. Optik iletimde 2 adet önemli bozunum söz konusudur: Kromatik Saçılım ve Polarizasyon Mod Saçılımı.

Kromatik saçılım düzeltmesi için kullanılan yöntemlerden biri olan optik ızgara tabanlı saçılım düzelticiler yerine Sayısal İşaret İşleme yöntemleri ile düzeltme avantajları sebebi ile popüler hale gelmeye başlamıştır. Sayısal yöntemlerden ise frekans düzleminde düzeltme yöntemleri tercih edilmektedir. Frekans düzlemine geçiş için FFT algoritmaları kullanılmaktadır; fakat birim zamanda hat üzerinden akan veri miktarının çok yüksek, verilerin devamlı olması sebebi ile standart hızlardaki seri yapıda çalışan FFT algoritmaları çözüm olarak yetersiz kalmaktadır. FFT tasarımının paralelleştirilmesi gerekmektedir.

Bu çalışmada optik ağlarda veya yüksek hız gerektiren uygulamalarda kullanılmak üzere 9 saat (clock) işareti gecikmeli 256 nokta Radix-4 DIF FFT tasarımı yapılarak FPGA'de gerçeklenmiş ve test edilmiştir.

Yüksek lisans tez danışmanım Yrd. Doç.Dr. Sıtkı Öztürk'e, tasarımlarımı gözden geçiren ve alternatif metotlar arayışımda bana yol gösteren değerli hocam Yrd. Doç.Dr. Mehmet YAKUT'a, tasarım konusundaki fikirlerimi dinleyen ve tecrübelerini esirgemeyen İbrahim Etem ÜPÇİN'e, tüm sorularımı sabırla cevaplayan Ahmet KALE'ye, öğrenim hayatım boyunca bana her türlü maddi ve manevi desteklerini esirgemeyen, beni bugünlere getiren aileme, bilgiye kıymet veren ve benimle paylaşan herkese gönülden teşekkürlerimi sunarım.

Ocak - 2014 Gökhan POLAT

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER DİZİNİ ve KISALTMALAR

Kısaltmalar

ADC	:	Analog to Digital Converter (Analogdan Sayısala Dönüştürücü)
ASIC	:	Application Specified Integrated Circuit (Uygulamaya Özel Tümleşik Devre)
CD	:	Chromatic Dispersion (Kromatik Saçılım)
CLB	:	Configurable Logic Block (Düzenlenebilir Lojik Blok)
CMA	:	Constant Modulus Algorithm (Sabit Büyüklük Algoritması)
CPLD	:	Complex Programmable Logic Device (Karmaşık Programlanabilir Lojik Aygıt)
DCF	:	Dispersion Compensating Fiber (Saçılım Düzeltici Fiber)
DIF	:	Decimation In Frequency (Frekansta Örnek Seyreltme)
DIT	:	Decimation In Time (Zamanda Örnek Seyreltme)
DFT	:	Discrete Fourier Transform (Ayrık Fourier Dönüşümü)
DP-QPSK	:	Dual Polarization Quadrature Phase Shift Keying (Çift Polarizasyon Dörtlü Faz Kaydırmalı Anahtarlama)
DSP	:	Digital Signal Processing (Sayısal İşaret İşleme)
EEPROM	:	Electronically Erasable Programmable Read-Only Memory (Elektrikle Silinebilir Programlanabilir Sadece Okunabilir Bellek)
FFT	:	Fast Fourier Transform (Hızlı Fourier Dönüşümü)
FPGA	:	Field Programable Gate Array (Alan Programlanabilir Kapı Dizisi)
IP CORE	:	Intellectual Property Core (Fikir Mülkiyetli Çekirdek)
ISE	:	Integrated Software Environment (Entegre Yazılım Ortamı)
JTAG	:	Joint Test Action Group (Ortak Test Eylem Grubu)
LUT	:	Look Up Table (Taramalı Tablo)
MAE	:	Mean Absolute Error (Ortalama Mutlak Hata)
PAL	:	Programmable Array Logic (Programlanabilir Dizi Lojiği)
PLA	:	Programmable Logic Array (Programlanabilir Lojik Dizisi)
PMD	:	Polarization Mode Dispersion (Polarizasyon Modu Saçılımı)
PROM	:	Programmable Read Only Memory (Programlanabilir Sadece Okunabilir Bellek)
ROM	:	Read Only Memory (Sadece Okunabilir Bellek)
SPLD	:	Simple Programmable Logic Device (Basit Programlanabilir Lojik Aygıt)
VHDL	:	VHSIC Hardware Description Language (VHSIC Donanım Tanımlama Dili)
WDM	:	Wavelength Division Multiplexing (Dalga boyu Bölümlemeli Çoğullama)

100GBİT 256 NOKTA RADİX-4 FFT'NİN FPGA İLE GERÇEKLENMESİ

ÖZET

Optik bir bozulma olan Kromatik Saçılım, günümüzde fiber filtreler yerine; Sayısal İşaret İşleme yöntemleri ile yeniden programlanabilir donanımlar olan FPGAler kullanılarak düzeltilebilmektedir. Frekans düzleminde, Kromatik Saçılımın sebep olduğu bozulmaları düzeltmek, Fourier dönüşümü gerektirir. Optik hatlarda veri aktarım hızı çok yüksek olduğu için, kullanılacak FFT algoritması paralel bir mimariye ihtiyaç duyar. Piyasadaki çözüm olarak sunulan FFT IP korlarının bu ihtiyacı istenilen hızlarda tam olarak karşılamaması bu projenin geliştirilmesini tetiklemiştir.

Bu tezde yapılan çalışmanın amacı; yüksek hız gerektiren uygulamalar için 256 nokta Radix-4 100 Gbit/s tamamen paralel FFT algoritmasının donanımsal olarak gerçeklenmesidir. Tez sonunda geliştirilen algoritma FPGA'e gömülerek test edilmiştir.

Anahtar Kelimeler: FFT, FPGA, Kromatik Saçılım, Optik, Paralel Mimari, Sayısal İşaret İşleme

FPGA IMPLEMENTATION OF 100GBIT 256 POINT RADIX-4 FFT

ABSTRACT

Chromatic dispersion is an optical distortion, which nowadays can be compensate with Digital Signal Processing methods by using reconfigurable hardware as FPGAs, instead of fiber filters. Correction of distortion in frequency domain caused by chromatic dispersion requires Fourier transform. FFT algorithm requires a parallel architecture because of data transfer speed is so high in optical transmission lines. FFT IP cores offered as solution on market can't exactly satisfy these requirements at desired speed; this reason triggered the development of this project.

The purpose of the work in this thesis, realization of full parallel 256 point Radix-4 100Gbit/s FFT algorithm on hardware, for applications which requires high speed. As result of thesis, the developed algorithm has been implemented and tested on FPGA.

Keywords: Chromatic Dispersion, Digital Signal Processing, FFT, FPGA, Optic, Parallel Architecture

GİRİŞ

İhtiyaçlar buluşları doğurur. İletişim ve haberleşme geçmişten günümüze insanlığın en büyük ihtiyacı olmuştur. Bu konuda yapılan onlarca çalışma sonrası bilim kurgu filmlerini aratmayacak haberleşme aletleri günlük hayatımızın bir parçası olmuş, hatta hayatımızın vazgeçilmezleri haline gelmiştir.

Haberleşmedeki bu gelişim toplumun her kesimince benimsendikçe ve kullanılmaya başlandıkça, gün içerisinde muazzam miktarlarda veri akışı gerçekleşmektedir. Önceden sınırlı sayı ve kişide bulunan telefonlar yerine her kişinin yanında mobil bir telefon taşıması, hatta kullanılan mobil telefonlarla konuşma yapmak yerine video, resim veya internet gibi yoğun bilgi taşıyan transferlere ihtiyaç duyulması, kişi başına düşen veri akışını hızla artırmaktadır. Veri akışı kişi bazından toplum düzeyine çıkartılarak incelenirse, günlük veri aktarımının büyük oranda arttığı gözlenmektedir.

Toplumun önceye nazaran yüksek veri aktarım ihtiyacı, bizi mevcut teknolojinin yetersizliği konusunda düşünmeye sevk etmiştir. Dolayısı ile yüksek hızlı veri iletimi için kullanılan bakır hatlarda iletim yerine optik hatlara yönelim başlamıştır.

Optik hatlar, bakır hatlara göre birçok avantaja sahiptir. Ne yazık ki her avantaj bazı dezavantajları beraberinde getirir. Bazı bozunumlar Bakır hatlarda elektriksel iletim için söz konusu değilken, optik iletim için söz konusudur.

Optik bozunumlardan biri olan Kromatik Saçılım(CD), DCF gibi optik bazı filtreler ile düzeltilebilir ama DCF fazladan kayıplara sebep olur. Bu kayıpların etkilerini düzeltme için kullanılacak optik yükselteçler hat üzerindeki gürültüyü de yükseltir. Bu durumdan kurtulmak için bir alternatif metot CD etkisini elektriksel hat üzerinde düzeltmeye çalışmaktır [1]. Haykin'e göre "İşaret işleme, eldeki problemin fiziğinin altında yatan öncelikli ve kazanılan bilgiyi matematiğin eşsiz yeteneği ile başarılı bir şekilde birleştirmek için en iyi yoldur" [2]. CD, etkisi fiziksel olarak modellenebilir olmasından dolayı DSP metotları ile düzeltilebilir.

DSP metotlarından frekans düzleminde düzeltme sıklıkla tercih edilir. Zaman düzlemindeki işaretimizi frekans düzlemine geçirmek için FFT işlemine ihtiyaç duyulmaktadır. Problemin can alıcı kısmı CD etkisini düzeltmek için kullanılacak FFT algoritmasını optik hatlar için gerçek zamanlı şekilde gerçekleştirmektir. Piyasada FFT işlemi için bulunan IP korlar, kaynak kullanımını minimumda tutmak için seri çalışma mantığında tasarlanmıştır. Bu sebeple istenilen hızlara çıkamamaktadır. Paralel yapıda çalışan bir tasarım kaçınılmaz olmaktadır. Özellikle paralel işlem gerektiren uygulamalarda yaygın olarak tercih edilen FPGA'ler bu işe çok uygundur.

Yapılmış çalışmalar ve piyasadaki mevcut çözümler incelendiğinde, FFT için kullanılan IP CORE'ların seri mimariye sahip olması sebebi ile hız açısından yetersiz olduğu görülmüştür. En yakın çalışma ise ALCATEL'in ürettiği özel gelişme ortamı olan TCHATER bordu üzerinde koşan 128 nokta 20GS/s'lık FFT kodudur.

Bu tezde, FFT işleminin temel prensipleri incelenerek, yüksek hızlar gerektiren uygulamalar için 9 saat işareti gecikmeli 256 nokta Radix-4 DIF FFT algoritması FPGA'de gerçeklenmiştir ve testleri yapılmıştır.

Birinci bölümde; optik hatlar üzerindeki haberleşme sistemleri, optik bozulmalar, düzeltim metotları ve ihtiyaç duyulan mimari tasarımdan bahsedilecektir.

İkinci Bölümde DFT, FFT, Radix gibi kavramlardan bahsedilerek temel taşları incelenecektir. Tasarlanan paralel FFT koru için temel tasarım prensipleri anlatılacaktır.

Üçüncü bölümde gömülü sistemler ve kullanılacak donanımlar anlatılacaktır.

Dördüncü bölümde tasarım ve yöntem detayları anlatılacaktır. Tasarım sırasında karşılaşılan problemler incelenecektir.

Beşinci bölümde ise sentez sonuçları ve test sonuçları paylaşılacaktır. Çalışma üzerine yapılabilecek geliştirmeler için önerilerde bulunulacaktır.

1. OPTİK HABERLEŞME

Bu bölümde genel olarak optik hatlar incelenecektir. Optik hatların, bakır hatlar ile karşılaştırılması yapılacak, avantajları üzerinde durulacaktır. Optik hatların sebep olduğu bazı bozunumlardan bahsedilecek ve en önemli etkilerden olan CD düzetmesi incelenecektir.

1.1. Optik Hatlar

İnsanoğlunun haberleşme serüveni incelendiğinde, optik haberleşme ilkler arasında yer alır. Ateş ve dumanla haberleşme optik haberleşmenin en temel örneklerindendir. Buna rağmen; elektriğin kullanılabilir hale gelişi, haberleşmeyi de etkisi altına almıştır. Bu sayede haberleşme teknolojilerinin atası sayılan telgraf ve telefon, elektriksel haberleşmenin önünü açmıştır. Elektriksel iletim hayatımızda önemli bir yer edinmiştir ve edindiği yeri korumaktadır.

Telekomünikasyon alanında bireysel ve kurumsal ihtiyaçların giderek artması, elektriksel hatların bant genişliği açısından yetersiz gelmeye başlaması, insanları alternatif iletim hatları araştırmaya yöneltmiştir. Optik hatlar ile haberleşme bu boşluğu doldurmaya adaydır.

Optik haberleşme tarihi incelendiğinde, telefonun mucidi Graham Bell ses işaretlerini ışık yoluyla ileten "Photophone" adlı bir cihaz geliştirmiş fakat havada iletilen ışığın dış etkenlerden aşırı etkilenmesi üzerine bu uygulamadan vazgeçmiştir. 1975'te ABD hükümeti Cheyenne Mountain'da bulunan NORAD karargâhındaki bilgisayarları elektronik gürültüyü azaltmak amacı ile fiber ile bağlamaya karar vermiştir. 1977'de ise 2 km uzunluğundaki ilk fiber telefon iletişim hattı Chicago'da 672 ses kanalıyla kullanılmaya başlanmıştır [3]. Böylelikle fiber hat kullanımı hayatın bir parçası olmaya başladı. Diğer iletim hatlarına göre üstün birçok özelliği olmasından dolayı vazgeçilmez bir hal aldı.

Optik hatlar, elektriksel hatlara göre; daha uzun mesafelere iletim, daha geniş bant genişliği, daha hızlı veri iletimi, daha az kayıp ve hammadde olarak bakıra göre daha

ucuz olan cam (yani temelinde silisyum) kullanımı dolayısı ile daha düşük maliyet getirdiğinden günümüzde elektriksel iletimin yerini almaktadır.

Optik hatların diğer avantajları:

- Elektromanyetik darbelerden etkilenmezler,
- Fiziksel boyutları küçük ve hafiftirler,
- Tek bir fiber içinden birçok farklı dalga boyu ile iletim yapılabilir,
- Topraklama problemli yoktur,
- Dış etkenlere karşı dayanıklıdır (Yağmur, sıcaklık, radyasyon v.s).

Optik hatlar, metropollerde hat döşeme ve hatları uç uca ekleme zorluğu gibi bazı dezavantajlara da sahiptir.

1.2. Optik Hat Bozunmaları

Fiber hatlar birçok yönden diğer iletim hatlarına göre avantajlı olmasına rağmen mükemmel değillerdir. Işığın doğasından kaynaklanan kendine özgü bozunumlara uğrarlar. Bu bozunumlar temelde doğrusal ve doğrusal olmayan olmak üzere iki ana başlıkta incelenebilir.

1.2.1. Doğrusal olamayan optik bozulmalar

Optik hatlar üzerindeki doğrusal olmayan bozulmalar; Kerr etkisi ve saçılma etkisi olmak üzere iki alt kategoride incelenebilir. SBS (Stimulated Brillouin scattering), SRS (Stimulated Raman scattering) saçılma etkisi kaynaklı bozulmalardır. SPM (Self-phase modulation), XPM (cross phase modulation) ve FWM (four wave mixing) Kerr etkisi kaynaklı bozulmalardır [4].Bu bozulmalar:

- SBS, optik işaret ile ses dalgalarının etkileşimden kaynaklanan saçılımdır.
- SRS, ışığın ve silisyumlu moleküllerin titreşimlerinden kaynaklanan saçılmadır.
- SPM, camın kırılma indeksinden kaynaklanır ve darbe işaretinin genişlemesine sebep olur.
- XPM, kırılma indeksinden kaynaklanır, tek hat üzerinden birden çok dalga boyu ile iletim yapılırken (WDM kullanılırken) bir ışığın dalga boyunun diğer bir dalga boyu üzerindeki faz bozulmasına sebep olur.

- FWM, farklı dalga boyundaki işaretlerin; kırılma indeksinden kaynaklı olarak farklı frekans varyasyonlarının etkileşimi yüzünden işarette fazladan dalga boyu üretilmesine sebep olur.

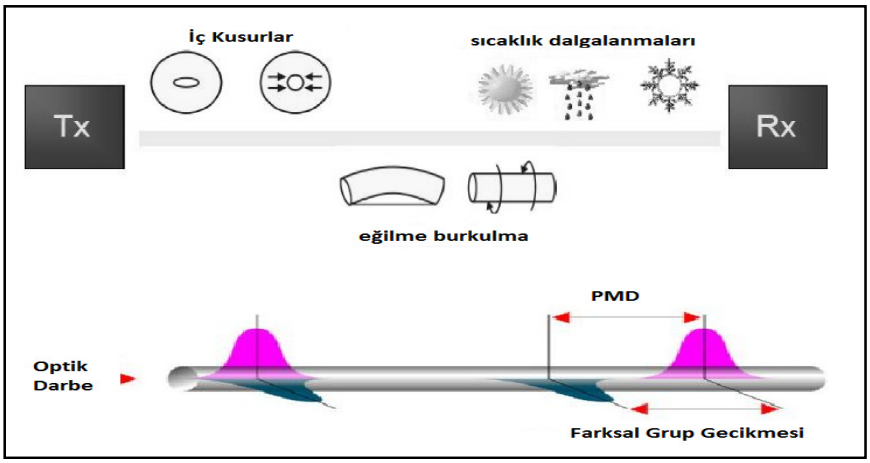
1.2.2. Doğrusal optik bozulmalar

Optik üzerindeki doğrusal bozulmalar, polarizasyon mod saçılımı (PMD) ve kromatik saçılma (CD) olmak üzere iki temel başlıkta incelenebilir.

1.2.2.1. Polarizasyon mod saçılımı (PMD)

Hattı verimli kullanmak için tek bir fiber hat üzerinden, birbirine dik olan iki farklı polarizasyonda bilgi taşınabilir. Bu taşımanın sorunsuz gerçekleşmesi için, fiber çekirdeği mükemmel silindirik yapıda olmalıdır. Fakat pratikte bu mümkün değildir. Bükülme, kıvrılma, sıcaklık ve mekanik gerilim gibi fiziksel etkiler fiber çekirdeğinin yapısını daha da bozar [1,5]. Bu bozulmalar yüzünden, aynı anda farklı polarizasyonlar üzerinden verici tarafından yollanan işaretler alıcı tarafına aynı anda ulaşmaz. Polarizasyonların farklı gecikmelere maruz kalmasından dolayı oluşan bu etkiye PMD (Polarizasyon Mod Saçılımı) denir. Birimi $\mathrm{ps}/\sqrt{\mathrm{km}}$ şeklinde gösterilebilir [1, 5, 6].

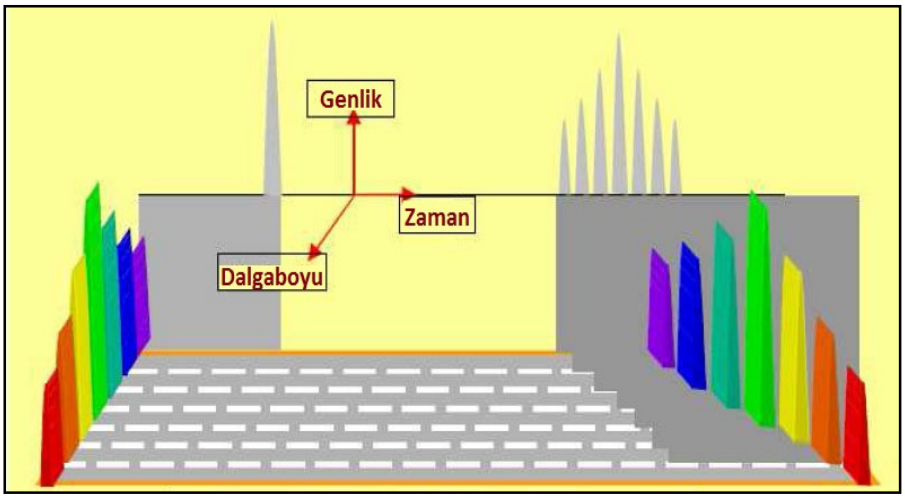
Şekil 1.1'de fiber hattın mekanik ve çevresel şartlar sebebi ile çekirdeğinde oluşan şekil bozulmalarından kaynaklanan PMD etkisi görülmektedir. Vericiden aynı anda iki farklı polarizasyondan yollanan optik işaretlerin PMD'a maruz kalması sebebi ile alıcı tarafına farklı zamanlarda ulaştığı görülmektedir.



Şekil 1.1. PMD oluşumu [1]

1.2.2.2. Kromatik saçılım (CD)

Optik kaynaklar tek bir frekansta değil bir frekans bandında enerji yayarlar. Farklı dalga boylarındaki ışık, doğası gereği farklı hızlarda hareket eder. İletilen işaretin farklı spektral bileşenleri arasında gecikme oluşur. Dolayısı ile gönderici tarafından yollanan işaret alıcı tarafına ulaştığında yayılma etkisi gösterir [1, 7-13]. Yayılan darbeler birbirleri arasında etkileşime sebep olur. Bu yayılma etkisi aslında işarette faz bozulmasıdır [9]. Şekil 1.2'de aynı anda farklı dalga boylarında yollanan optik işaretlerin, farklı hızlara sahip olması sebebi ile alıcıya farklı anlarda ulaşması ile oluşan CD etkisi gösterilmektedir.



Şekil 1.2. CD oluşumu [11]

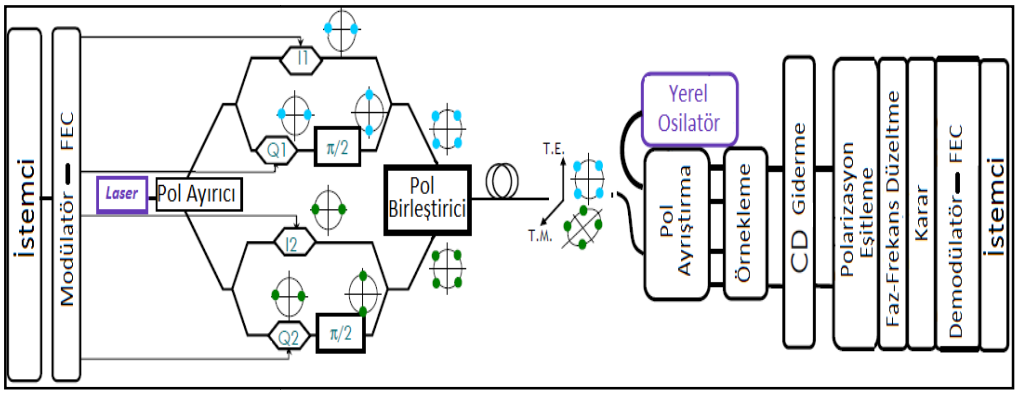
CD, grup hızı bozulması olarak da ifade edilir, birimi ps/(nm x km) ile verilir. CD, malzeme saçılımı ve dalga kılavuzu saçılımından oluşmaktadır [7], uzaklık ile doğrudan ilişkilidir. CD transfer fonksiyonu Denklem (1.1)'de, grup hızından kaynaklı bozulma parametresi ise Denklem (1.2)'de verilmiştir;

$$H(f) \cong e^{j\frac{\pi\lambda^2 DL}{c}f^2} \tag{1.1}$$

$$D = -\frac{2\pi c}{\lambda^2}\frac{d^2\beta}{d\omega^2} = \frac{2\pi c}{v_g^2\lambda^2}\frac{dv_g}{d\omega} \tag{1.2}$$

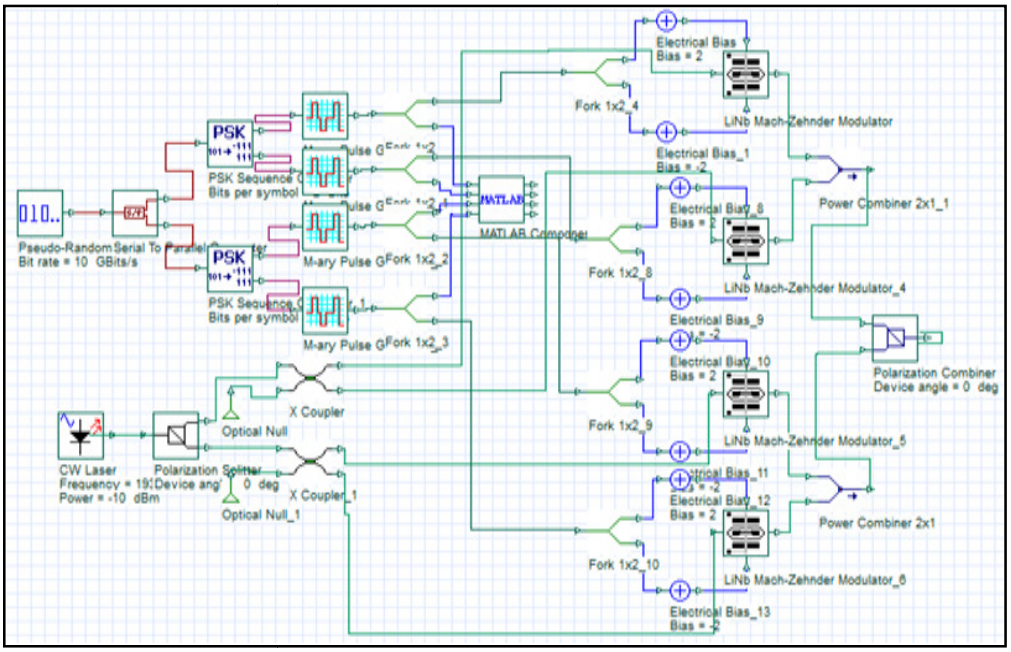
Denklemde λ: dalga boyu(nm), L: iletim hattı uzunluğu (km), D: CD parametresi(ps/nm.km), c: ışık hızı (m/s), B bant genişliği (Hz), β: propagasyon sabiti ve v_g=dω/dβ grup hızıdır [12,13].

Verici tarafından gönderilen işaretin hat üzerinde kromatik saçılıma maruz kalması sonucu; alıcı tarafta oluşan bozulma etkisinin görülüp anlaşılması için standart bir optik haberleşme sistemi incelenmelidir. Kullanılan optik haberleşme yapısı Şekil 1.3'de verilmiştir.

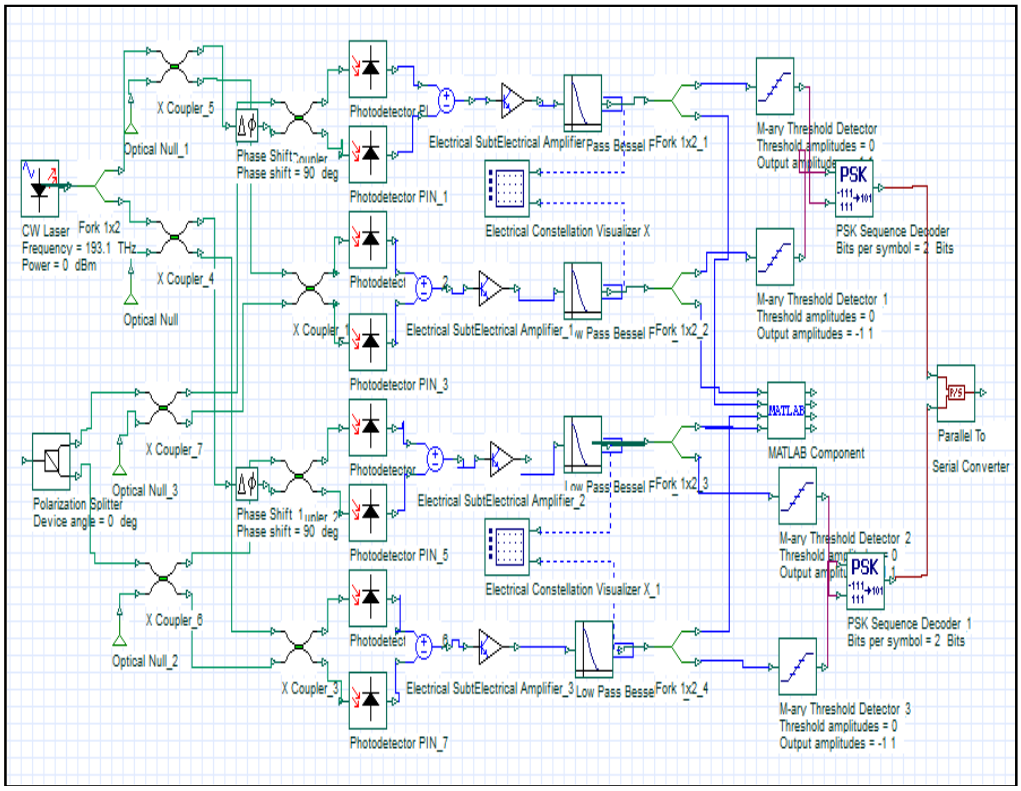


Şekil 1.3. DP-QPSK Optik Verici Alıcı [14]

Şekil 1.3'de verilen DP-QPSK iletimi yapan haberleşme sisteminin çalışma yapısını daha iyi anlamak adına simülasyon yapılması gerekmektedir. "OPTISIM" adlı simülasyon programı kullanılmış ve şekildeki sisteme uygun olarak haberleşme sistemi kurulmuştur. Kurulan verici blokları Şekil 1.4'de, alıcı blokları ise Şekil 1.5'de detayları ile gösterilmektedir.

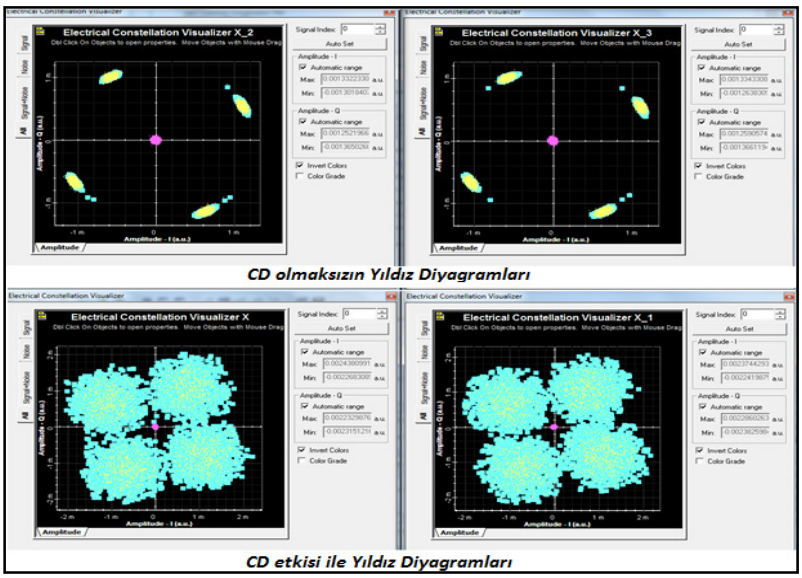


Şekil 1.4. DP-QPSK OPTISIM Verici Kısmı



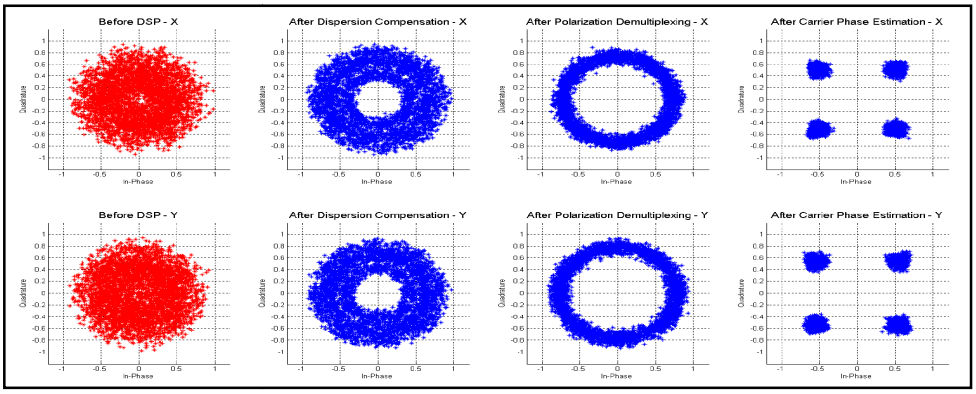
Şekil 1.5. DP-QPSK OPTISIM Alıcı Kısmı

Sistem üzerinde sadece CD etkisini görmek için diğer bozunumlar hesaba katılmadan yaklaşık 100 km için simülasyon yapıldığında yıldız diyagramları Şekil 1.6 verilmiştir.



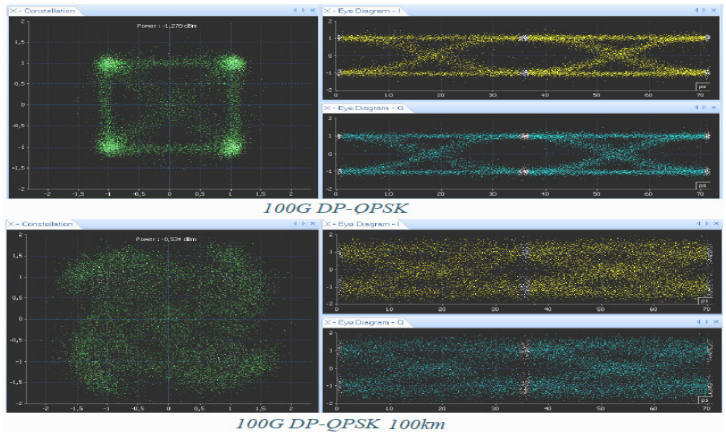
Şekil 1.6. CD etkisinin Yıldız Diyagramlarında görünüşü

CD-PMD-Faz Bozulmasına maruz kalan bir işaretin alıcıda sırası ile düzletilmesi Şekil 1.7de gösterilmiştir.



Şekil 1.7. Bozulan optik işaretin alıcıda sırası ile düzeltilmesi [15]

Kromatik saçılımın etkisi, göz diyagramında Şekil 1.8'de görülmektedir.

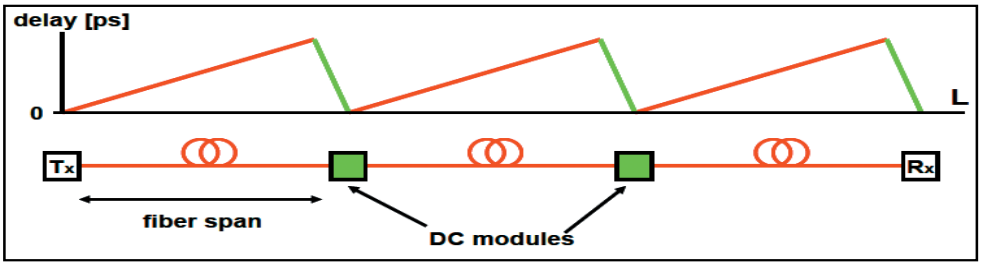


Şekil 1.8. Göz Diyagramlarında CD etkisi [16]

1.3. CD Düzeltmesi

Bu tezin çıkışını tetikleyen konu, kromatik saçılım düzeltmesidir. Telafi edilmesi gereken en önemli optik bozulmalardan biri kromatik saçılımdır. CD doğrusal bir etkidir, dolayısı ile kararlıdır. Yani etkileri tahmin edilebilir ve kontrol edilebilir [10]. CD düzeltmesi hem elektriksel hem de optik olarak yapılabilir.

Optik düzlemde düzeltme yöntemi olarak DCF (Saçılım Düzeltici Fiber) kullanılır. Bu yöntemde hat üzerinde belli noktalara yerleştirilen negatif saçılım oluşturacak filtreler kullanılır. Şekil 1.9'da DCF modüllerle CD düzeltmesi şematik olarak verilmektedir.



Şekil 1.9. Fiber hat üzerinde DCF [10]

Kabul görmüş yöntemlerden olan DCF, hat üzerinde fazladan kayıplara sebep olur. Bu kayıpları önlemek için optik yükselteçlere ihtiyaç vardır. Fakat bu yükselteçler işaretle birlikte işaret üzerindeki gürültüyü de yükseltir [1]. Bu yöntem sistemi olumsuz olarak etkiler.

Elektriksel düzeltme yöntemleri incelenecek olursa, düzeltme için zaman düzlemi yöntemleri veya frekans düzlemi yöntemleri seçilebilir.

Zaman düzleminde kullanılan düzeltme yöntemleri;

- Savory Metodu
- Adaptif Filtreler Metodu
- CMA Metodu
- LMS Metodu

Frekans düzleminde kullanılan düzeltme yöntemi;

- Hızlı Fourier Dönüşüm Metodu

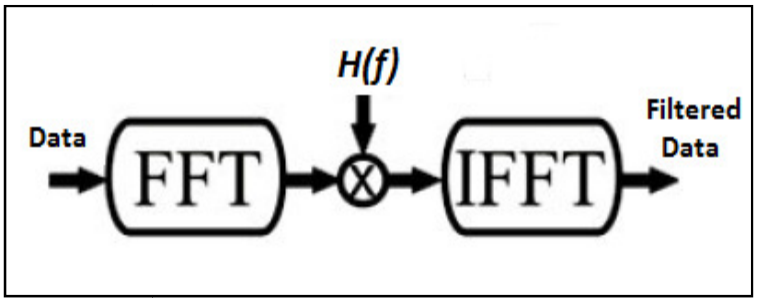
CD düzeltmesinde yapılan çalışmalar incelendiğinde, frekans düzleminde FFT metodu ile düzeltme uygulamalarının popüler bir metot olduğu ve sıklıkla tercih edildiği görülmektedir.
Bu çalışmanın çıkış noktası olan kromatik saçılım düzeltmesi için FFT metodu üzerinde durulacaktır.

1.3.1. FFT metodu ile CD düzeltmesi

CD etkisi bir faz bozulmasıdır. Bu bozulmayı düzeltmek için FFT metodunu kullanıldığında; öncelikle işaretlerimizin zaman düzleminden frekans düzlemine geçirilmesi gerekmektedir. Frekans düzlemine geçirilen işaretimizi, CD sistem yanıtının tersi (Denklem (1.3)) ile çarpmak; gerekli faz düzeltmesini sağlayacaktır.

Düzeltilen işaretin ise IFFT si alınarak tekrar zaman düzlemine geçilmesi ile CD düzeltmesi gerçekleşmiş olacaktır. Şekil 1.10 anlatılan işlemlerin şematiği verilmiştir. Denklem (1.3) aşağıdaki gibidir;

$$H(f) \cong e^{-j\frac{\pi\lambda^2 DL}{c}f^2} \tag{1.3}$$

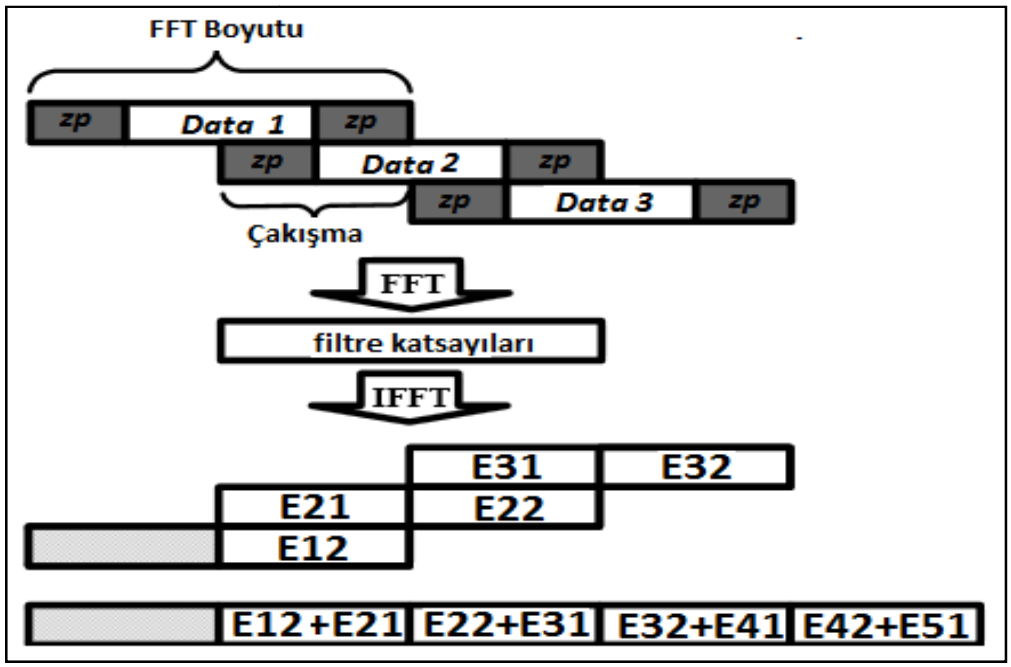


Şekil 1.10. FFT CD Düzeltme Bloğu Şematiği [1]

FFT söz konusu olduğunda uzun diziler ile çalışmak büyük bir dezavantajdır. Çünkü; FFT nokta sayısı arttıkça sistem karmaşıklığı artmaktadır. Bundan dolayı küçük dizi pencerelerine bölerek çalışmak daha kolay olacaktır. Bu seferde pencerelerin birbirinden bağımsız değerlendirilmesinden kaynaklı hatalar ortaya çıkacaktır. Bu hataları engellemek için çakıştır-topla (overlap-add) veya çakıştır-sakla(overlap-save) yöntemleri kullanılır. Şekil 1.11'de çakıştır-topla yönteminin şematik gösterimi bulunmaktadır.

N nokta FFT için Çakıştır-Topla metodunun basamakları [7,17];

- t anında N/2 adet dizi elemanı bilgi penceresini oluşturur.
- t anında bilgi penceresinin her iki yanına N/4 adet sıfır ekleme (zero padding) yapılır.
- t anında mevcut N elemanlı pencerenin FFT'si alınır katsayılar ile çarpılır ve IFFT'si alınarak hafızada tutulur.
- t+1 anında yukarıdaki işlemler tekrarlanır.
- Son olarak IFFT'si alınan t anı dizisinin son N/2 örneği ile t+1 anı dizisinin ilk N/2 örneği toplanır.
- Bu işlem her yeni gelen veri için benzer şekilde devam eder.



Şekil 1.11. Çakıştır Topla (Overlap-Add) Metodu [7]

2. FOURİER DÖNÜŞÜMÜ

Optik bir haberleşme sisteminin incelendiği bölüm 1'de, sistemde fiber hattın sebep olduğu bozulmalar anlatılmıştır. Bu optik hat bozulmalarından CD etkisine odaklanılmıştır. İncelemeler sonucu CD etkisinin hat üzerinde sebep olduğu bozulmaların düzeltilmesi için kullanılan yöntemler araştırılmıştır. Bu araştırmaların sonucu olarak frekans düzleminde FFT Metodu ile CD düzeltme metoduna ulaştırmıştır.

Bu bölümde FFT metodunu daha verimli kullanabilmek adına, Fourier dönüşümü, kullanım alanları, Ayrık Fourier Dönüşümü, Hızlı Fourier Dönüşümü incelenecektir.

2.1. Fourier Dönüşümü ve Tarihi

Meraklı insanlar, tarihin başından beri doğayı, doğanın içindeki nesneleri incelemişlerdir. İncelemeleri sırasında bulduklarını birbirleri ile paylaşmış ve geleceğe taşımak istemişlerdir. Bunun için doğanın ortak dili olan matematiği seçmişlerdir.

İnsan yaradılışı gereği incelemelerini zaman düzleminde yapmıştır, bulgularını zaman düzlemi üzerinde incelemiştir. Fakat doğadaki çoğu olay veya varlık yapısı itibari ile frekans düzlemine incelenmeye mahkûmdur. Bazı araştırmacılar bu gerçeğe çok yaklaşmış olduğu halde bulamamıştır. Bazı salgın hastalıklar tarihte belli periyotlarda ortaya çıkar, gökyüzü cisimleri belli periyotlarda hareket eder, hatta tüm maddelerin yapı taşı sayılan atomlar belli frekanslarda titreşirler. Fourier dönüşümü, özet olarak zaman düzlemindeki bilgiyi frekans düzleminde dönüştürür, insanlara bilgiyi incelemek için farklı bir bakış açısı sağlar.

Fourier dönüşümünden önce Jean Baptiste Joseph Fourier tarafından bulunmuştur. 1798 yılında Napolyon Mısır'a giderken, Fourier'i de yanına alarak ve onu bilim heyetinin başına atamıştır. Mısır'da iken her dahi gibi araştırdığı konuya takıntılı hale gelmiştir. Isı taşınımının matematiksel esasları üzerine araştırmalar yapmaya başlamıştır. Isı üzerine çalışmalarda bulunurken garip alışkanlıklar edinmiştir. Çölün

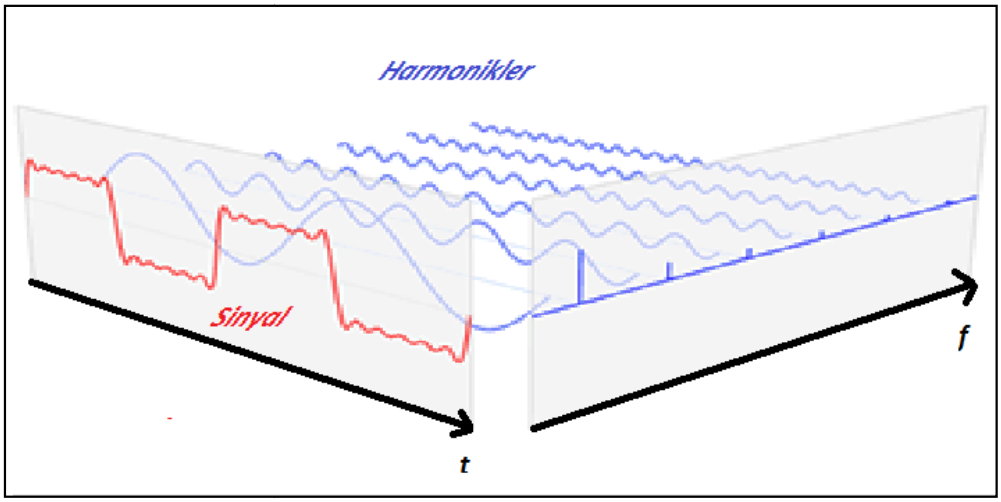
sıcağının sağlık için en iyi bir ortam olduğuna inanmıştı. Bu nedenle bir mumya gibi örtünüyor, çöl sıcağı kadar sıcak odalarda kalın kıyafetlerle oturuyordu. Ziyaretçileri bulunduğu ortamların rahatsızlık verecek kadar sıcak olduğundan bahsediyordu [18]. (Bazı kaynaklara göre sıcaklıktan kaynaklanan damar çatlaması sebebi ile öldüğü söylenir, hatta Mısır seferine katılmasının sebebi sıcak havanın cazibesindendir.)

Mısır'da bulunduğu zaman boyunca, ısıya olan tutkusu onu bu konuda bir denklem türetmeye sürüklemiştir. Üzerinde düşündüğü sorun bile yeterince karışıktı; Fourier, gemi çapasına bağlı zincir halkasının ısıtılırken, ısı akışının zincirin geri kalanına ve çapaya yayılışının denklemini çözmek istemişti. Düşüncesi ısının düzensiz dağılımı, halkanın etrafındaki birçok sinüs bileşeninin frekansı ile anlatılabilirdi. Önermesi şu şekildedir; maksimum sıcaklık ve sinüs bileşenlerinin harmoniklerin pozisyonu, sıcaklık dağılımının düzensiz Fourier dönüşümü aracılığı ile türetilebilir [19].

Fourier'in ısı çalışmaları üzerine ortaya çıkan Fourier Dönüşümü; ele alınan işaretin frekans davranışını inceler. Doğadaki tüm periyodik fonksiyonlar sinüs ve kosinüs fonksiyonlarının toplamları veya farkları şeklinde gösterilebilir. İfadelerin sinüs ve kosinüsler yerine, karmaşık üslü sayıların toplamı olarak gösterilmesine; Fourier serisi gösterimi denir. Harmonik analizi olarak ta adlandırılır. Fourierin bu buluşu kendisinden daha önce Euler ve Lagrange tarafında da bulunmuş, fakat ikisi de bu formüllerin süreksiz periyodik fonksiyonlara uygulanamayacağını düşünmüştür. Fourier bu seri açılımının her periyodik fonksiyona uygulanabileceğini iddia etmiştir. Fourier bu formülü iki yüzeyi farklı ısılarda olan katı bir cismin sıcaklık dağılımını hesaplamak için kullanmış, yoğun işlem çabası gerektirdiğinden ve yaklaşık sonuç verdiğinden verimli olmadığını düşünmüştür [18].

Bu matematiksel dönüşüm formülünün başlarda değeri çok anlaşılmamasına rağmen günümüzde geniş bir kullanım alanı vardır. Biyolojide DNA iki sarmallarının şekli X, ışını kırınım teknikleri (1962), virüs gibi karmaşık canlı analizi FFT sayesinde bulunmuştur [18]. Ses filtreleme, görüntü işleme, haberleşme, müzik kullanım alanlarında bazılarıdır.

Şekil 2.1'de periyodik bir işaretin zaman ve frekans düzlemindeki bileşenleri verilmiştir.



Şekil 2.1. Periyodik işaretin frekans bileşenleri [20]

Fourier dönüşümü sürekli ve ayrık fourier dönüşümü olarak 2 ana başlıkta incelenebilir.

2.2. Ayrık Fourier Dönüşümü (DFT)

Mikroişlemcilerin ilk kullanım alanlarından biri matematiktir. Karışık hesapların yapılması sırasında mikroişlemcilerin kullanılması makul bir yöntemdir. Fourier dönüşümü gibi uzun soluklu ve zorlu matematiksel işlemlerin, insanlar yerine mikroişlemcilere yaptırılması; hem hesap sırasında insan kaynaklanan hataları yok edecek hem de işlem süresini kısaltacaktır.

Günlük hayatta gözlemlediğimiz veriler sürekli zamanlı ve sürekli genliklidir, yani analogtur. Mikroişlemcilerin sürekli zaman işaretleri ile çalışabilmesi için işaretlerin zamanda örneklenmesi gerekmektedir. Fourier dönüşümü sonsuz uzunlukta bir dizi için tanımlı olduğundan periyodik ve ayrık bir işaretin frekans analizi için Ayrık Fourier Dönüşümü (AFD veya DFT) metodu kullanılmaktadır. (DFT sonlu örneğe uygulanacağından el ile çözümlerde de kolaylık sağlar.)

Sürekli zaman Fourier dönüşümü Denklem (2.1)'deki gibi iken ayrık Fourier dönüşümü Denklem (2.2)'deki gibidir;

$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-jwt}\, dt \tag{2.1}$$

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}} = \sum_{n=0}^{N-1} x(n) W_N^{kn} \tag{2.2}$$

Ayrık Fourier Dönüşümünün çözümü için temel olarak 3 yöntemden bahsedilebilir [21];

- Eşzamanlı hesaplamalarla DFT;

Zaman düzlemindeki N değer ile frekans düzlemindeki N tane bilinmeyenin temel matematiksel denklemlerle hesaplanabilmesi için doğrusal bağımsız N tane denklemin çözülmesi gerekir. N adet değişken, N adet sinüzoidal demektir. N sinüzoidin her birinin bir kat sayıyla çarpımının toplamı giriş işaretini verecektir. N nokta için bu toplam yazılırsa N tane N bilinmeyenli denklem elde edilmiş olur.

- Korelasyonla DFT

İşaretin içerisinde bilinen dalga formu olan sinüs ve kosinüs bileşenlerinin aramasıdır. Bilinen dalga formlarımızdan sinüs sanal, kosinüs gerçek değerleri bulmamızı sağlar.

- Hızlı Fourier Dönüşümü (HDF- FFT)

DFT'nin bazı temel özelliklerni kullanarak (tek ve çift simetri özelliği) işlem yükünü azaltmaya yarayan bir hesap yöntemidir. Detaylarına bir sonraki bölümde değinilecektir.

2.3. FFT

DFT metodu, işaretlerin frekans düzleminde incelenmesine olanak sunmasına rağmen işlem yükü açısından verimli bir yöntem değildir. Bu algoritma zamanla farklı kullanıcılar tarafından optimize edilmeye çalışılmıştır. DFT yönteminde bazı matematiksel özelliklerden faydalanarak kısa yollar kullanılması sonucunda FFT metodu ortaya çıkmıştır. Zaman içinde birden fazla kişi konu üzerinde kafa yorduğu için çeşitli yaklaşımlardan bahsedilebilir. Bazıları çoklu indeksleme yöntemleri, bazıları DFT'nin simetri ve asal katların özelliklerinden faydalanmıştır [20-22];

- Dairesel konvolüsyon temelli yöntemler;
  - Rader Algoritması
  - Bluestein (Chirp-z) Algoritması
  - Winograd I Algoritması

- Çok boyutlu indeksleme temelli yöntemler;
  - Winograd II Algoritması
  - Good-Thomas Algoritması
  - Cooley-Tukey Algoritması

Çok boyutlu indeksleme metotları toplama sayını azaltan veya çarpma sayısını azaltan yöntemler olarak ikiye ayrılır. Bu çalışmada FFT algoritması FPGA üzerine gerçekleştirilecektir. FPGA'de çarpma işlemleri toplama işlemlerine göre daha fazla işlem yükünü beraberinde getirdiğinden çarpma sayısını azaltan yöntemlerden olan Cooley-Tukey algoritması kullanılacaktır(yöntemler arası çeşitli karşılaştırmalar Tablo 2.1 verilmektedir [23]).

Tablo 2.1. FFT yöntemlerinin karşılaştırılması [23]

Algoritmalar	**Maliyet**	**Karmaşıklık**	**Çalışma Frekansı**	**Yapılan İşlem**
Cooley-Tukey	Düşük	Düşük	Düşük	N noktalı DFT'yi daha küçük parçalar halinde çözer
Winograd	Normal	Normal	Normal	$(Z^N - 1)$ İfadesini çeşitli polinomlar şeklinde çarpanlara ayırır
Rader-Brenner	Normal	Normal	Düşük	N noktalı DFT'yi $N = 2^t$ dönüşümü ile çözer
Brunn	Yüksek	Düşük	Normal	DFT üzerinde gerçel katsayıları hesaplar
Asal Çarpan (Good-Thomas)	Düşük	Yüksek	Yüksek	Sadece N_1 ve N_2 aralarında asal olduğu durumlarda DFT'yi yeniden ifade edilerek uygulanır

İşlem karmaşıklığı bakımından incelendiğinde DFT N^2 adet karmaşık çarpma, $N(N-1)$ adet karmaşık toplama işlemi getirmektedir [24]. FFT ise $(\frac{N}{2}\log_2 N)$ adet karmaşık çarpma, $N\log_2 N$ adet karmaşık toplama işlemi gerektirmektedir (İşlem yükü incelenirken, parametre olarak çarpma işlemleri temel alınmaktadır. Çünkü çarpma işlemleri, toplama işlemlerine göre daha karmaşık yapıdadır ve beraberinde daha fazla işlem yükü getirmektedir.). İşlem yükünü açıklayan karşılaştırma Tablo 2.2'de verilmiştir.

Tablo 2.2. $O(N^2)$ ile $\frac{N}{2}\log_2 N$ işlem yükü karşılaştırması [25]

	DFT'nin Hesap Yükü		FFT'nin Hesap Yükü	
N	**Karmaşık Çarpma N^2**	**Karmaşık Toplama N(N-1)**	**Karmaşık Çarpma $(N/2)\log_2 N$**	**Karmaşık Toplama $(N)\log_2 N$**
2	4	2	1	2
8	64	56	12	24
32	1024	922	80	160
64	4096	4022	192	384
128	16384	16256	448	896
2^{10}	1048576	1047522	5120	10240
2^{20}	~10^{12}	~10^{12}	~10^{7}	~$2x10^{7}$

Bir örnek üzerinden anlatmak gerekirse; işlem karmaşıklığını karşılaştırmak için karmaşık çarpma sayıları baz alınabilir. Eğer 1MFLOP (saniyede 1 milyon floating point işlem yapabilen) bir makine kullanılırsa ve N=2^{20} seçilirse;

- $O(N^2) \approx 10^{12}$ işlem, dolayısı ile 10^6 saniye, yani yaklaşık 11.5 gün sürer.
- $(\frac{N}{2}\log_2 N) \approx 10^7$ işlem, dolayısı ile 10 saniye sürer.

FFT kavramlarından bahsederken atlanmaması gereken kavramlardan ikisi DIT (Decimation In Time) ve DIF (Decimation In Frequency)'dır. FFT işlemleri yapılırken, sürecin hızlandırılması için işlemler gruplandırılır. Bu gruplama yapılırken işlem yapılacak örnekler zamanda seyreltiliyor ise DIT, frekansta seyreltiliyor ise DIF adını alır.

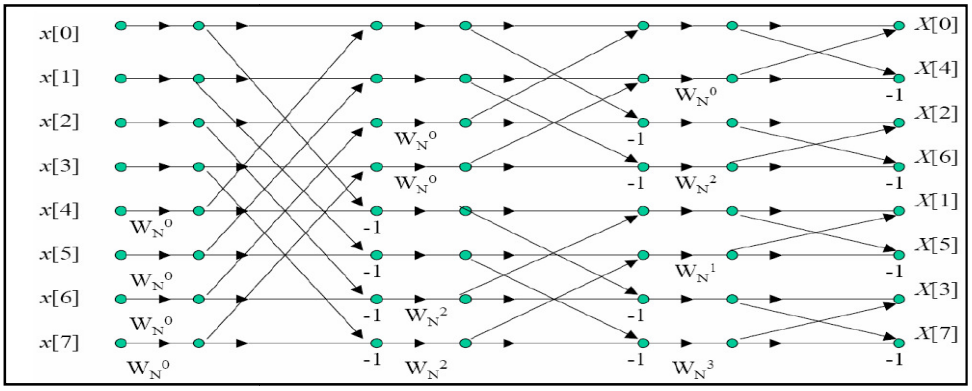
DIT FFT işlemi yapılırken temel metot, örnekleri tek ve çift sayılı örneklere ayırmaktan geçmektedir. Örneğin N=8 için, x[n] = [x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7]] dizisi tek ve çift örneklerine ayrılır. Çift örnekler xçift[n] = [x[0], x[2], x[4], x[6]] , tek örnekler ise xtek[n] = [x[1], x[3], x[5], x[7]] dizileri ile gösterilir. Çift örnekler kendi arasından tekler ve çiftlere olarak xçt[n] =[x[0], x[4]], xçç[n] = [x[2], x[6]], şeklinde ayrılır. Benzer şekilde tek örnekler tekler ve çiftlere xtt[n] = [x[1], x[5]], xtç[n] =[x[3], x[7]], şeklinde alt dizilere ayrıştırılır. Daha büyük N değerleri içinde en temel ikiliye kadar tekler ve çiftler şeklide ayrıştırma yapılmalıdır.

DIT işlemleri DFT'de gösterilecek olursa; Denklem (2.2)'deki x(n) ifadesi her biri eşit uzunlukta x(2m) şeklinde çift ve x(2m+1) tek parçalara ayrılırsa yeni ifade;

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x(2m) W_N^{2mk} + \sum_{m=0}^{\frac{N}{2}-1} x(2m+1) W_N^{(2m+1)k} \quad (2.3)$$

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x(2m) W_{N/2}^{mk} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} x(2m+1) W_{N/2}^{mk} \quad (2.4)$$

şekline dönüşmektedir [26]. N=8 iken FFT-DIT yapısı Şekil 2.2 gösterildiği gibidir.



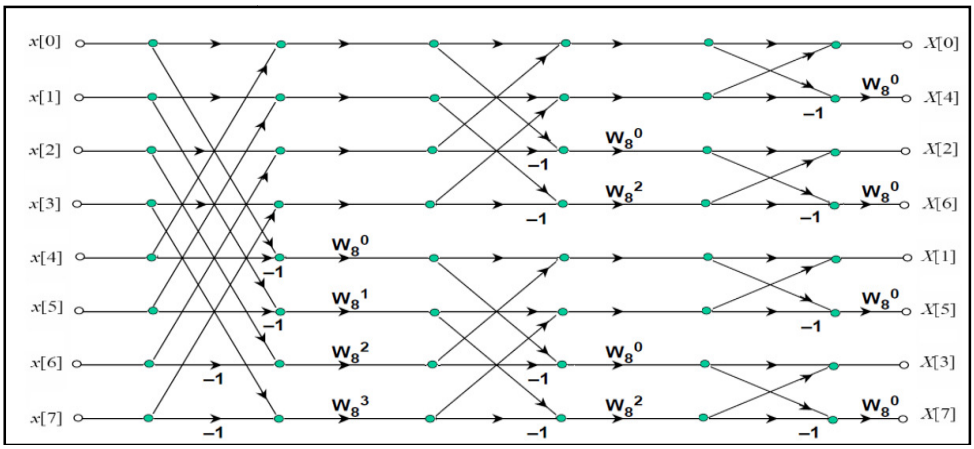
Şekil 2.2. 8 nokta FFT için DIT [27]

DIF işlemleri DFT üzerinden gösterilecek olursa; Denklem (2.2)'deki ifade kullanılarak;

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + \sum_{n=N/2}^{N-1} x(n) W_N^{kn} \quad (2.5)$$

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2}) W_N^{kn} \quad (2.6)$$

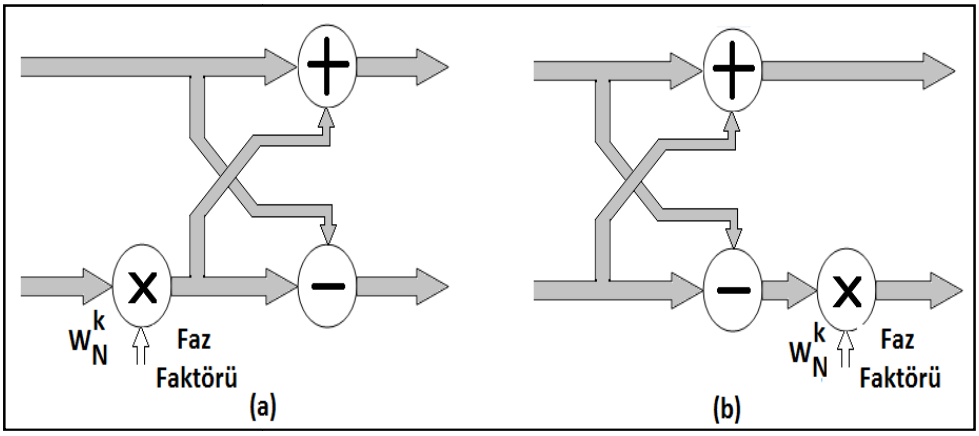
şekline dönüşmektedir [28]. N=8 iken FFT-DIF yapısı Şekil 2.3 gösterildiği gibidir.



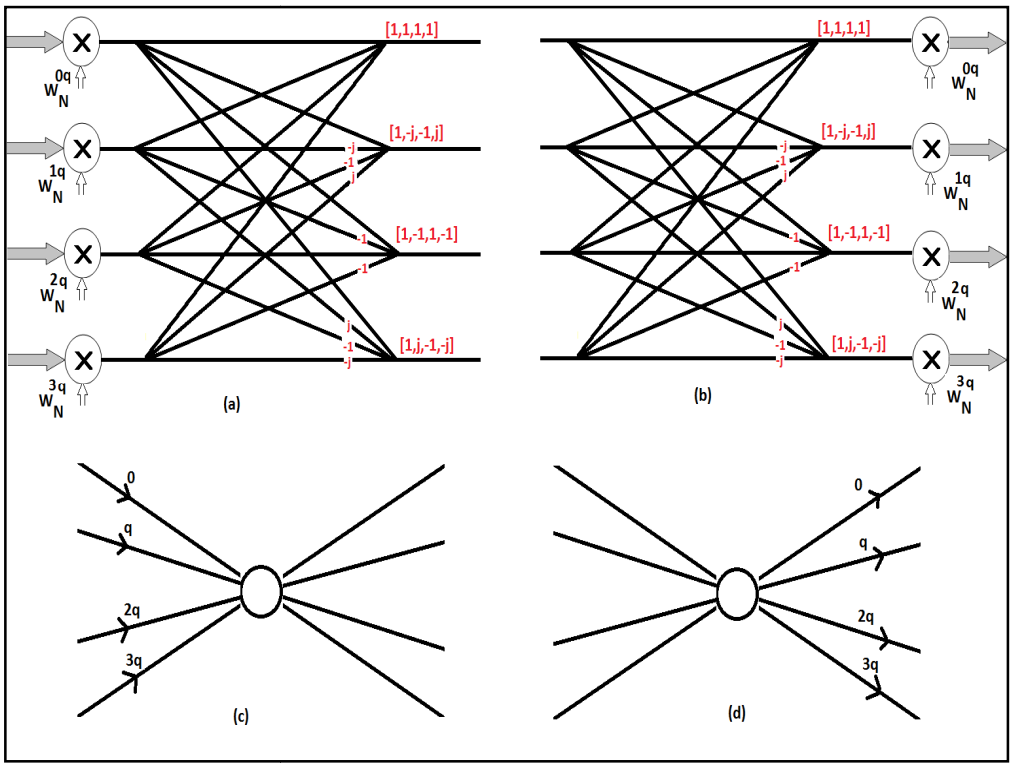
Şekil 2.3. 8 nokta FFT için DIF [27]

2.3.1. FFT de radix kavramı

FFT işlemi gerçekleştirilirken işlemleri kolaylaştırmak için yapılan gruplamalar Radix olarak adlandırılır. Özel olarak Radix-2 yapısı Butterfly (kelebek), Radix-4 yapısı Dragonfly olarak adlandırılır. Radix-2 için Butterfly yapıları Şekil 2.4 verilmiştir.



Şekil 2.4. a)Radix-2 DIT Butterfly b)Radix-2 DIF Butterfly



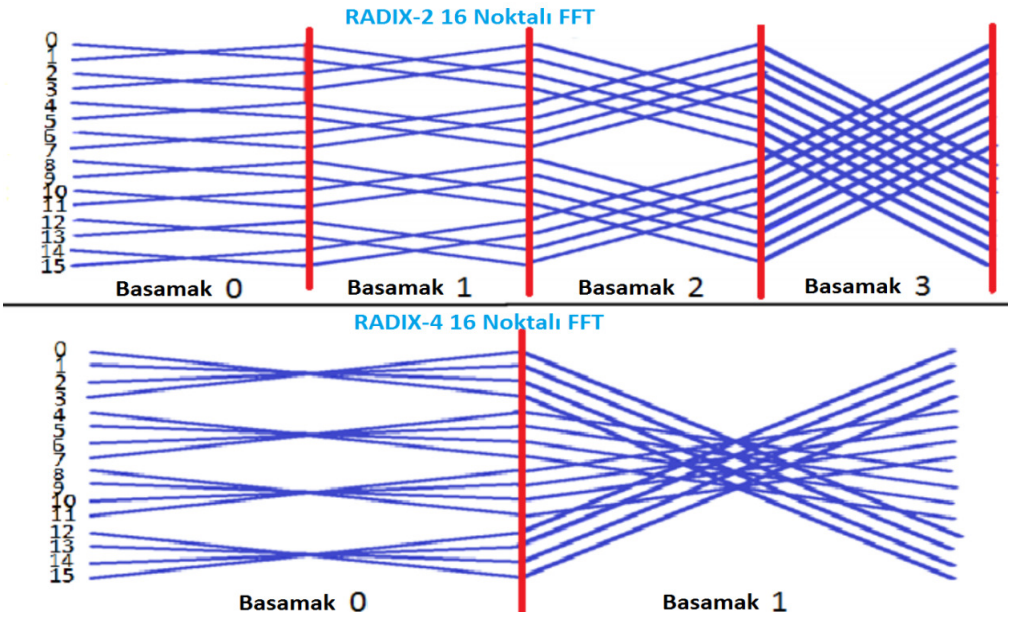
Şekil 2.5. a)Radix-4 DIT Dragonfly b)Radix-4 DIF Dragonfly c)DIT Dragonfly kısaltılmış şekli d)DIF Dragonfly kısaltılmış şekli

N nokta FFT işlemi yapılırken seçilen Radix sayesinde tüm işlemlerin kaç basamakta hesaplanılacağı r=Radix, N=FFT nokta sayısı iken,

$$\text{FFT işlem basamak sayısı} = \log_r N \tag{2.7}$$

formülü ile hesaplanmaktadır.

Yüksek Radix kullanımı FFT işleminin daha az basamakta çözülmesine olanak tanır. Radix-4 yerine Radix-8, Radix-16 gibi yapıların kullanımı basamak sayısını azaltmasından dolayı tercih sebebi gibi görünmesine rağmen, yüksek Radix işlem karmaşıklığını arttırdığı için tercih edilmemektedir. Radix-2 ve Radix-4 arasındaki basamak karşılaştırması Şekil 2.6'da verilmiştir.



Şekil 2.6. 16 Nokta FFT için Radix-2 ve Radix-4 Basamak karşılaştırması [29]

FFT için Radix-2 ve Radix-4 uygulamalarda en fazla tercih edilen yöntemlerdir. Radix-8 yöntemi ise nadiren tercih edilmektedir. Çünkü Radix yükseldikçe karmaşıklık çok artmasına rağmen performans artışı beklentileri karşılamamaktadır [29]. Donanım üzerinde yüksek karmaşıklıktaki algoritmaları gerçeklemek, işleri fazlasıyla zorlaştırmaktadır.

Radix-2 DIF işlemleri için genelleştirilen ifadeler Şekil 2.7, Radix-4 DIF işlemleri için genelleştirilen ifadeler Şekil 2.8 verilmektedir [30].

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} [x(n) + x(n + \frac{N}{2})]$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} [x(n) - x(n + \frac{N}{2})]$$

Şekil 2.7. Radix-2 DIF genel ifade

$$X(4k) = \sum_{n=0}^{\frac{N}{4}-1} \left[x(n) + x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) + x(n+\frac{3N}{4})\right] W_N^0 W_{\frac{N}{4}}^{kn}$$

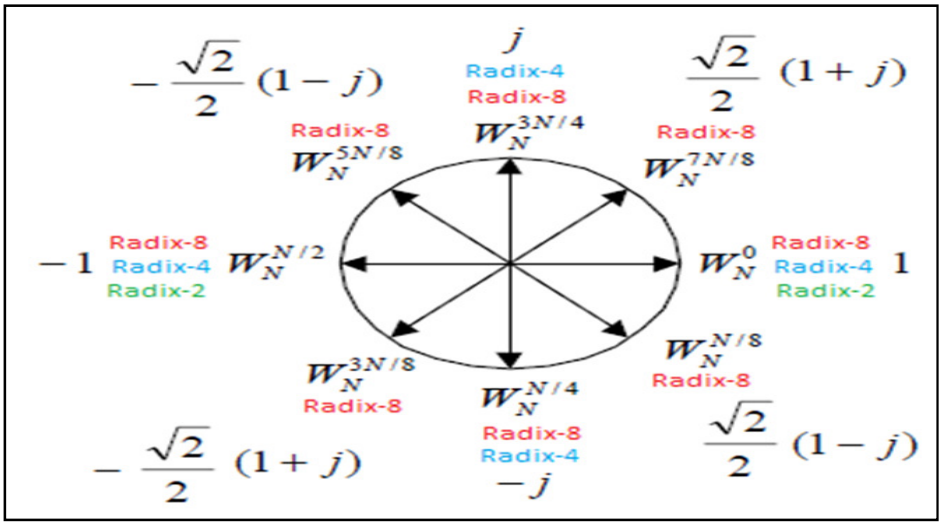
$$X(4k+1) = \sum_{n=0}^{\frac{N}{4}-1} \left[x(n) - jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) + jx(n+\frac{3N}{4})\right] W_N^n W_{\frac{N}{4}}^{kn}$$

$$X(4k+2) = \sum_{n=0}^{\frac{N}{4}-1} \left[x(n) - x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) - x(n+\frac{3N}{4})\right] W_N^{2n} W_{\frac{N}{4}}^{kn}$$

$$X(4k+3) = \sum_{n=0}^{\frac{N}{4}-1} \left[x(n) + jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) - jx(n+\frac{3N}{4})\right] W_N^{3n} W_{\frac{N}{4}}^{kn}$$

Şekil 2.8. Radix-4 DIF genel ifade

FFT işlemlerinde ayrıştırmalar sırasında oluşan katsayılar faz faktörü veya twiddle faktörü olarak adlandırılır ve birim çember üzerinde rahatlıkla gösterilebilirler. Oluşan faktörlerden bir tanesi Radix yöntemine bağımlı olarak oluşur, bir diğeri ise FFT nokta sayısı ile doğrudan ilişkilidir. Radix'e bağımlı olan faz faktörleri Şekil 2.9 üzerinde verilmiştir. Bu çarpanların getireceği işlem yükünden kurtulmak çok önemlidir. Çünkü kaç nokta FFT kullanılırsa kullanılsın, her Radix düğümünde bu çarpanlar olacaktır.



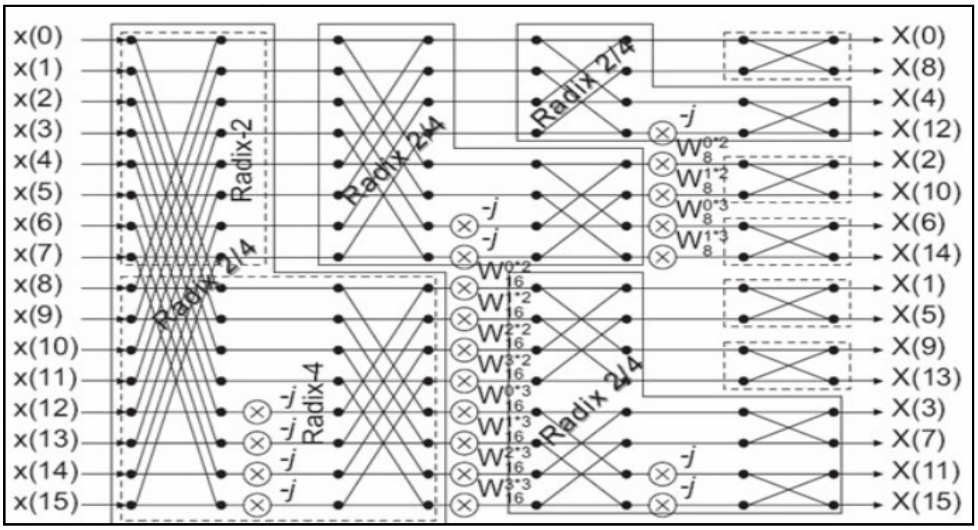
Şekil 2.9. Birim çember üzerinde Radix-2, Radix-4 ve Radix-8 faz faktörleri

Radix-2 birim çember üzerinde sadece W_N^0 ve $W_N^{N/2}$ bileşenleri bulundurur. Birim çemberdeki bu faz bileşenleri 0 ve 180 açılarına karşılık gelir. Bu açıların sin ve cos değerlerine bakıldığında, 0 açısı sin(0)=0 ve cos(0)=1 değerlerine karşılık gelir. Benzer şekilde 180 açısı için sin(180)=0 ve cos(180)=-1 değerlerine karşılık gelir. Sonuç olarak Radix-2 faz faktörleri birim çember üzerinde [1, -1] ile ifade edilebilir. Bu katsayılarla çarpma işlemi, bir toplama ve bir çıkartma işlemine dönüşmektedir. Bu da işlem yükü ciddi oranda azalmaktadır.

Benzer şekilde Radix-4'de faz faktörü bileşenleri; W_N^0, $W_N^{N/4}$, $W_N^{N/2}$ ve $W_N^{3N/4}$ tür. 0, 90, 180 ve 270 açılarına karşılık gelir. Sin ve cos açılımları incelendiğinde Radix-4 faz faktörleri [1, -j, -1, j] olmaktadır. Bu ifadelerde Radix-2'ye ek olarak sanal çarpanlar eklenmiş gibi görünse de bazı yer değiştirme işlemleri sayesinde sanal kısımlar sadece katsayı olarak değerlendirilebilmektedir (İşlemin detaylarına bölüm 5.4'de detaylı olarak değinilecektir.). [1, -j, -1, j] katsayıları ayrıştırma işlemleri sırasında W_N^{kn} ifadesindeki üstel "k" değerine göre değişmektedir. İfade gerçekte $[1^{(k)}, -j^{(k)}, -1^{(k)}, j^{(k)}]$ halindedir. Ayrıştırma sırasında k=0 için [1, 1, 1, 1], k=1 için [1, -j, -1, j], k=2 için [1, -1, 1, -1] ve k=3 için [1, j, -1, -j] değerlerini almaktadır. Katsayıların ayrıştırma sırasındaki bu kullanımı Şekil 2.8'verilmektedir.

Radix-8 kullanılmak istenirse faz faktörü bileşenleri W_N^0, $W_N^{N/8}$, $W_N^{N/4}$, $W_N^{3N/8}$, $W_N^{N/2}$, $W_N^{5N/8}$, $W_N^{3N/4}$ ve $W_N^{7N/8}$ olacaktır. Bunlar ise; 0, 45, 90, 135, 180, 225, 270 ve 315 açılarına denk gelecektir. Bu açılardan 45, 135, 225, 315 değerleri $\sqrt{2}$ katsayısı getirecektir. Radix-8'den gelen $\sqrt{2}$ katsayısını, Radix-2'den gelen [1, -1] ve Radix-4'den gelen [1, -j, -1, j] katsayıları ile karşılaştırdığımızda işlem yükünün ciddi oranda artacağını görülmektedir.

Split-Radix olarak adlandırılan diğer bir FFT algoritması da mevcuttur. Algoritma, farklı Radix yöntemlerini birlikte kullanarak ayrıştırma işlemleri gerçekleştirme temeline dayanmaktadır. Kullanılan basamak kavramı bu yöntemde ortadan kalkar. Çünkü basamaklar iç içe girmiştir. İşlem maliyetini azaltan bir algoritmadır. Radix-2 ve Radix-4 gibi rutin işlemleri takip etmediği için karmaşıklığı yüksektir. Örnek bir Split-Radix ayrıştırması Şekil 2.10 verilmiştir.



Şekil 2.10. Split-Radix [31]

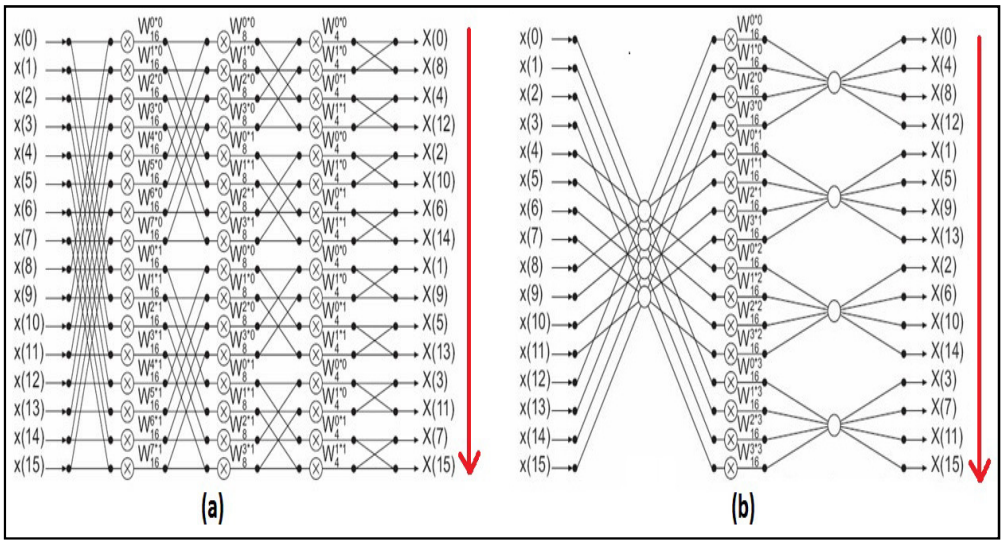
Bölüm 2.3 Tablo 2.2'de DFT ile FFT arasında işlem maliyeti karşılaştırmaları Radix-2'ye göre yapılmıştır. Şekil 2.11'de işlem maliyetleri Radix-2, Radix-4, Radix-8 ve Split-Radix karşılaştırmaları görülmektedir. Karşılaştırmalar yapılırken bazı tablolar karmaşık çarpma ve toplama miktarları üzerinden hazırlanmışken, bazı tablolar gerçek sayılar üzerinden çarpma ve toplama işlemleri ile ifade edilmiştir. Her bir karmaşık çarpma işlemi 4 gerçek çarpma ve 2 gerçek toplama işlemi ile ifade edilebilir. Her bir karmaşık toplam işlemi 2 gerçek toplama işlemi ile ifade edilebilir [32].

	Gerçel Çarpımlar				Gerçel Toplamlar			
N	Radix 2	Radix 4	Radix 8	Split Radix	Radix 2	Radix 4	Radix 8	Split Radix
16	24	20		20	152	148		148
32	88			68	408			388
64	264	208	204	196	1,032	976	972	964
128	712			516	2,504			2,308
256	1,800	1,392		1,284	5,896	5,488		5,380
512	4,360		3,204	3,076	13,566		12,420	12,292
1,024	10,248	7,856		7,172	30,728	28,336		27,652

Şekil 2.11. Radix işlem maliyetleri karşılaştırması [33]

Sıralı olarak verilen girişlere rağmen seçilen Radix'e göre çıkışların sırası değişmektedir. Çıkışta görülen yeni sıralama rastgele gibi görünmesine rağmen bir düzene sahiptir. Kullanılan Radix tabanına göre terslenmiş değerler çıkış

sıralamasında görülmektedir. 16 nokta FFT için girişleri düzgün sırada verilen bir dizinin Radix-2 ve Radix-4 yöntemlerine göre çıkış sıralamaları Şekil 2.12'da verilmektedir. Çıkış sıralamalarının taban düzeyinde terslenmiş halleri Şekil 2.13'de verilmiştir.



Şekil 2.12. a)16 nokta FFT Radix-2 b) 16 nokta FFT Radix-4 [31]

Giriş Sırası			Çıkış Sırası			
			Radix-2		Radix-4	
On Tabanında Gösterim	İki Tabanında Gösterim	Dört Tabanında Gösterim	On Tabanında Gösterim	İki Tabanında Gösterim	On Tabanında Gösterim	Dört Tabanında Gösterim
0	0000_2	00_4	0	0000_2	0	00_4
1	0001_2	01_4	8	1000_2	4	10_4
2	0010_2	02_4	4	0100_2	8	20_4
3	0011_2	03_4	12	1100_2	12	30_4
4	0100_2	10_4	2	0010_2	1	01_4
5	0101_2	11_4	10	1010_2	5	11_4
6	0110_2	12_4	6	0110_2	9	21_4
7	0111_2	13_4	14	1110_2	13	31_4
8	1000_2	20_4	1	0001_2	2	02_4
9	1001_2	21_4	9	1001_2	6	12_4
10	1010_2	22_4	5	0101_2	10	22_4
11	1011_2	23_4	13	1101_2	14	32_4
12	1100_2	30_4	3	0011_2	3	03_4
13	1101_2	31_4	11	1011_2	7	13_4
14	1110_2	32_4	7	0111_2	11	23_4
15	1111_2	33_4	15	1111_2	15	33_4

Şekil 2.13. Çıkış sıralamasının taban tersleme yöntemi ile elde edilişi

Şekil 2.13'de görüldüğü gibi 16 nokta Radix-2 FFT için 2 numaralı çıkış incelendiğinde (dizi "0" ile başladığı için 1. indis olarak ifade edilir.) iki tabanında "0001" ile ifade edilmektedir ve tüm rakamlar tersten dizildiğinde "1000" şekline dönüşür, on tabanında 8. indisi gösterir. Yani 2 numaralı çıkış 8. indisi gösterecektir.

3. GÖMÜLÜ SİSTEMLER

1948'de transistörlerin icadı ile insanlık tarihi en önemli dönemlerinden birine girmiştir. Transistörlerin boyutları kısa zamanda ciddi oranda küçülmüştür. Transistörler bir araya getirilerek tümleşik devreler oluşturulmuştur. Ciddi hesapların veya işlerin, insanlar yerine daha az hata yapan elektronik kontrollü makinelere bırakılma fikri zaman içinde cazibesini arttırmıştır. Bu fikir gömülü sistem olarak adlandırılan teknolojilerin geliştirilmesine ön ayak olmuştur.

Genel maksatlı bir sistemden farklı olarak, özel olarak tanımlanan işleri yerine getirmek için tasarlanan sistemler gömülü sistemler olarak tanımlanmaktadır. Bu sistemler sayesinde cihazların boyutu küçülür, maliyeti azaltılır ve optimize edilerek daha yüksek hızlara çıkabilir.

Kayda değer ilk gömülü sistem MIT Instrumentation Laboratory'de Charles Stark Draper tarafından geliştirilen Apollo Guidance Computer olmuştur. İlk kitlesel gömülü sistem üretimi 1961 yılında Minuteman füzesi için yapılan Autonetics D-17 rehber bilgisayarı olmuştur [34].

Tümleşik devrelerin ucuzlaması sayesinde gömülü sistemler kendilerine geniş çalışma alanları bulmaya başlamıştır. Hesap makinelerinden başlayarak çeşitli askeri projeler, motor kontrolü, ATM'ler, ölçüm cihazları, haberleşme aygıtları ve bunun gibi bir çok alanda kullanılmaktadır. Geniş bir kullanım alanına sahiptirler.

Kullanıcı ihtiyaçlarına ve maliyetlerine göre farklı donanımlar geliştirilme başlandı. Ortaya çıkan bazı gömülü sistem donanımları;

Basit Mikrodenetleyiciler: 8 veya 16 bit işlem kapasitesine sahip, birçok temel işlevi yerine getiren entegrelerdir. Diğer türlere nazaran ucuzdurlar. PIC (Mikrochip), MSP430(TI), AVR-8 (Atmel) bunlardan bazılarıdır.

ARM: 32 bit işlem kapasitesine sahip özelleşmiş bir mikrodenetleyicidir. Bazı işlemleri daha kolay veya daha hızlı yapabilmesi için özelleşen alt birimleri

mevcuttur. STMicroelectronics, TI, NXP gibi birçok firma, farklı çevresel birimleri ve özellikleri bulunan ürün aileleri üretilmiştir.

DSP: Özel olarak sayısal işaretleri işlemeye yarayan alt birimlere sahip bir mikrodenetleyicidir.

FPGA: İçindeki donanımsal yapı kullanıcı tarafından şekillendirilebilen özelleşmiş yeniden programlanabilir entegrelerdir. Kullanıcıya büyük esneklik sağlar ve paralel çalışma kabiliyetine sahiptir.

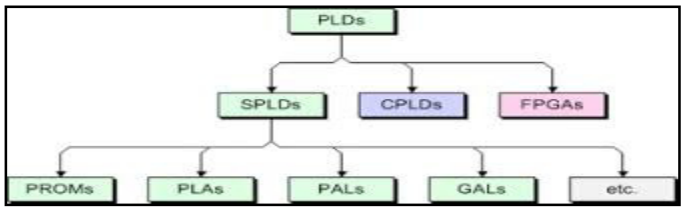
ASIC: İhtiyaç üzerine özelleşmiş sisteme ait tasarımın entegre olarak üretilmiş halidir. Gereksiz hiçbir alt birim içermez.

Hibrit yapılar: Kullanıcılar yukarıda bahsedilen donanımların bir tanesi ile ihtiyaç duyduğu işlemleri gerçekleştiremeyebilir. Tek bir işlemcinin yetersiz kaldığı yerler için çeşitli hibrit yapılar kullanır. Örneğin: OMAP=ARM+DSP, ZYNQ=FPGA+ARM , ASIC+ARM yapıları gibi.

3.1. Programlanabilir Entegreler

Kullanılan gömülü sistem donanımı, geliştirme sürecinde birden fazla programlamaya ihtiyaç duymaktadır. Programda yapılan her bir düzenleme ve düzeltme işleminin, simülasyon yerine donanım üzerindeki davranışlarını incelemek istersek; donanımın programlanabilir olmalıdır. Bu sayede maliyet düşecektir.

İlk programlanabilir cihazlar olan ROM'lar üzerinde değişiklik yapılması için imalat sürecinde değişikliğe gidilmesi gerekmekteydi. Bu yöntem o günkü ihtiyaçları kısmen karşılayabilse bile pahalı bir süreçti ve hata tespiti sadece üretim yapıldıktan sonra ortaya çıkmaktaydı. Bu sebepler PROM teknolojisinin ve türevlerinin gelişimini tetikledi. Şekil 3.1'de programlanabilir entegreler ailesi verilmiştir.



Şekil 3.1. Programlanabilir Entegreler

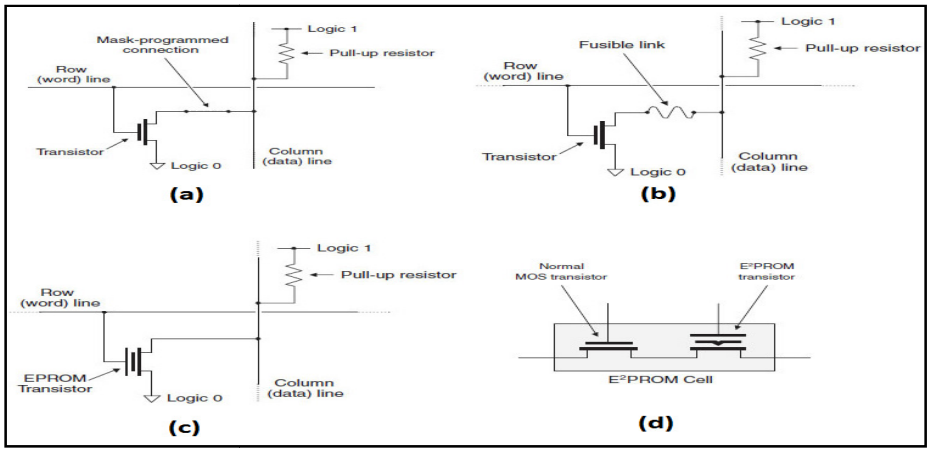
3.1.1. SPLD (Simple Programmable Logic Device)

SPLD'ler kapasiteleri en düşük, bunun sonucu olarakda en ucuz programlanabilir lojik entegrelerdir. SPLD'ler bu özelliklerinden ötürü sadece 7400 serisi TTL ürünlerine göre tercih edilirler. SPLD'lerdeki her hücrenin bir diğeri ile direk olarak bağlantısı vardır.

3.1.1.1. PROM (Programmable Read Only Memory)

Yeni bilgiler içeren bir ROM için üretim sürecinin beklemesi maliyet, zaman ve üretim gibi birçok süreç bakımından uygunsuz bir süreçti. Bu fikirden yola çıkarak üretilen PROM entegreleri bilgisayar yardımı ile bir kez programlanabiliyordu. Bu sayede üretim süreci beklenmiyor, yeni bilgiler içen bellek ünitesi için yeni baştan üretim sürecine girilmiyordu. Programlama yapılırken, entegre içerisindeki kullanılmayacak bağlantılar üzerinden yüksek akım geçirilerek yakılıyordu. PROM teknolojisi ROM mantığına göre bir çığır açmış olsa da yeterli gelmemekteydi.

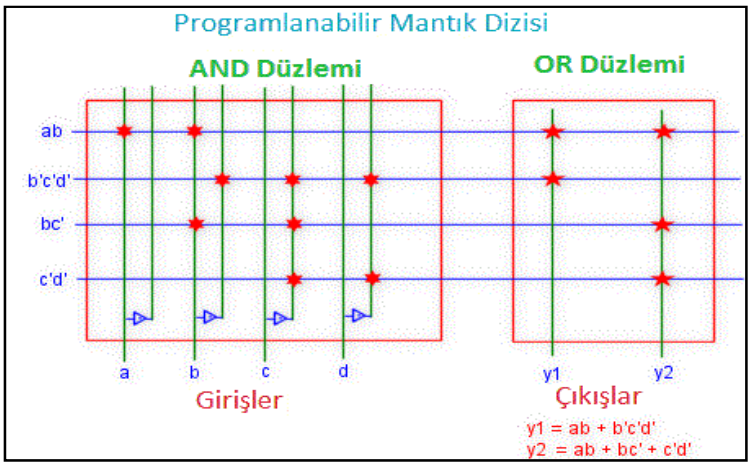
Bir kez programlanabilen PROM yerini elektriksel olarak tekrar yazılabilen ve silinebilen EEPROM teknolojisine bıraktı.



Şekil 3.2. a)PROM b)PROM c)EPROM d)EEPROM [35]

3.1.1.2. PLA (Programmable Logic Array)

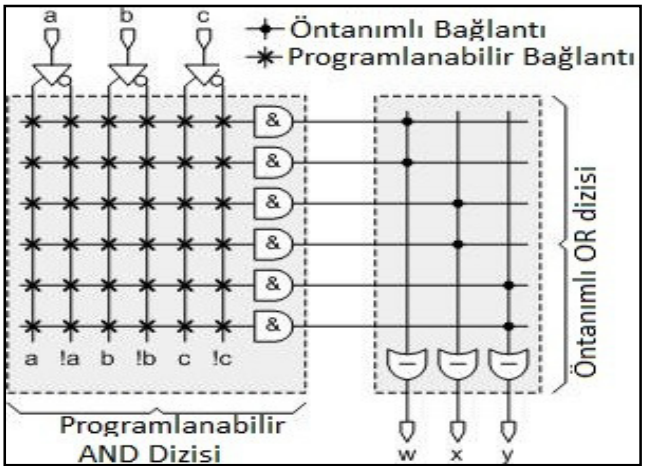
PROM teknolojisinin hız olarak yetersiz oluşu PLA teknolojisinin gelişmesine sebep olmuştur. PLA içyapı olarak AND, OR ve NOT kapılarından oluşmaktadır. Arzu edilen tüm tasarımlar bu mantık kapıları kullanılarak gerçekleştirilmek zorundadır. Şekil 3.3'de PLA şematiği görülmektedir.



Şekil 3.3. PLA ve örnek uygulaması [36]

3.1.1.3. PAL (Programmable Array Logic)

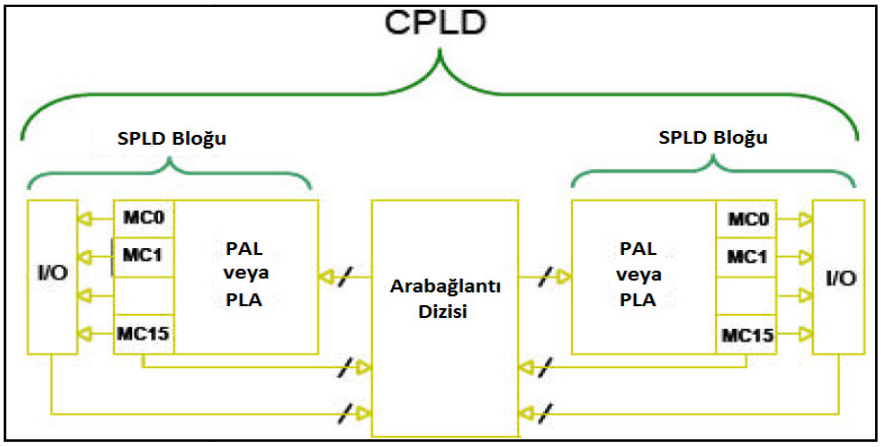
PAL teknolojisi PLA teknolojisinin geliştirilmesiyle elde edilmiştir. AND ve NOT kapılarıyla OR kapısının işlevi gerçekleştirilebileceği için OR kapısı sayısı azaltılmıştır, OR kapısında açılan yerlere XOR, Flip-Flop ve Multiplexer gibi yapılar yerleştirilmiştir. Bu sebeple birçok fonksiyonu PLA'lara göre daha hızlı işleyebilmektedir.Şekil 3.4'de PAL şematiği görülmektedir.



Şekil 3.4. PAL yapısı [36]

3.1.2. CPLD (Complex Programmable Logic Device)

Artan kullanıcı ihtiyaçların sebebi ile programlanabilir lojik yapılar (PAL ve PLA) birleştirilerek daha karmaşık ve işlevselliği yüksek CPLD'ler ortaya çıkmıştır. Şekil 3.5'de CPLD şematiği verilmiştir.



Şekil 3.5. CPLD [36]

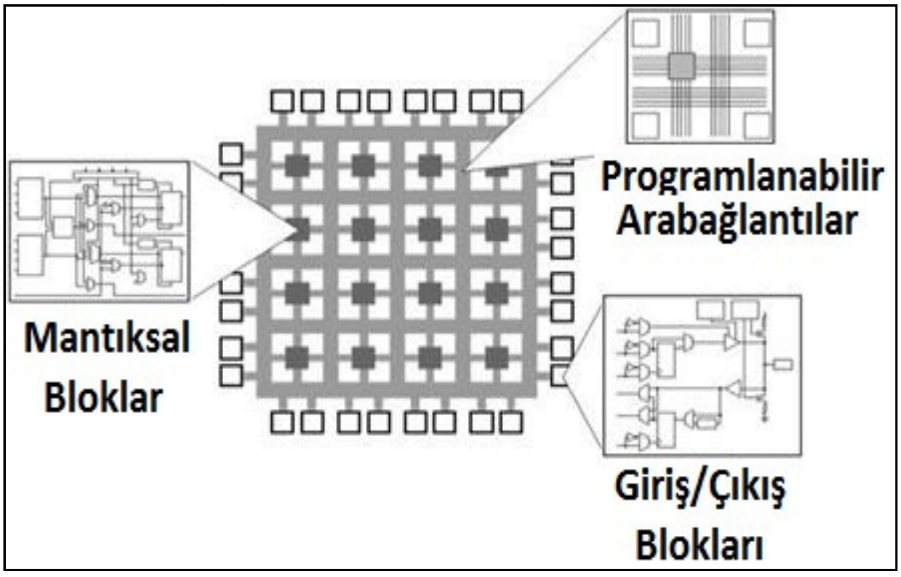
3.1.3. FPGA (Field Programmable Gate Array)

Piyasadaki PLD ve ASIC arasındaki boşluğu doldurmak amacı ile üretilen milyonlarca mantıksal kapıyı bünyesinde barındıran entegrelerdir. Kaynak yeterli geldiği takdirde her türlü dijital tasarım FPGAler üzerinde gerçeklenebilir.

İçyapısında temel bileşen olarak;

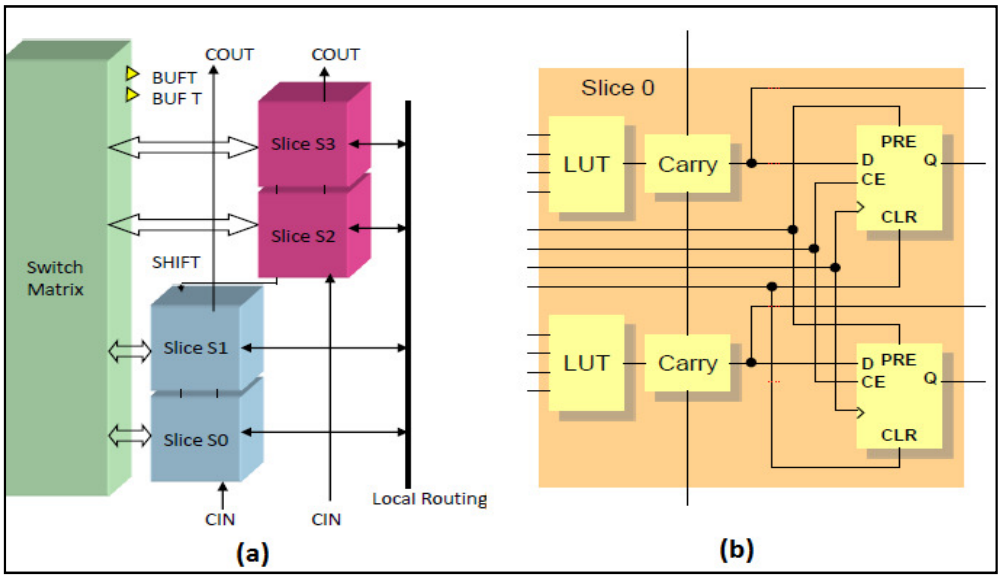
- CLB adlı düzenlenebilir mantık blokları
- Giriş- Çıkış blokları
- Ara bağlantı blokları

Olmak üzere 3 temel bloktan oluşur. Şekil 3.6'de FPGA iç yapısı gösterilmiştir.



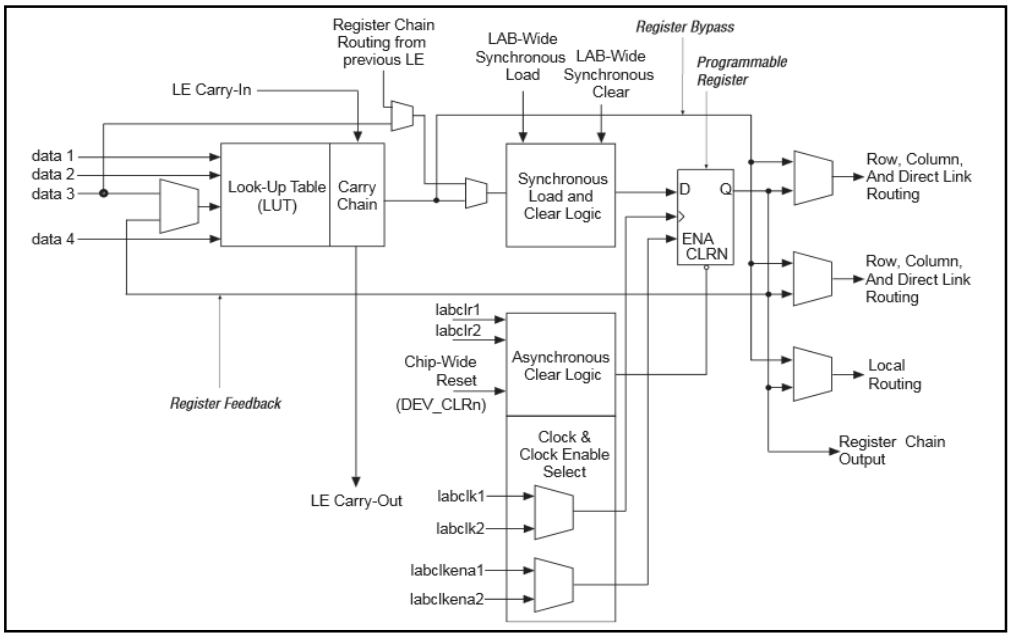
Şekil 3.6. FPGA [36]

CLB blokları Slice'lardan oluşmaktadır. Çeşitli firmaların marka ve modellerine göre kaç Slice içerdiği katalog bilgisinden bakılmalıdır. Aynı CLB içerisindeki Slice'lar birbirleri ile yerel arabağlantılara sahiptir. Her bir Slice LUT, Carry Logic, Flip-Flop'lar içerir. Şekil 3.7'de CLB, Şekil 3.8'de LE yapısı gösterilmektedir.



Şekil 3.7. a)CLB b)Slice [36]

Bazı firmalar Slice kavramı yerine LE ifadesini kullanır.



Şekil 3.8. Altera LE [36]

Temel olarak CLB, I/O ve arabağlantıları içeren FPGA'ler günümüzde birçok ek özelliği içinde barındırmaktadır. Bunlardan bazıları;

- DCM veya MMCM: FPGA entegresi içerisindeki saat işaretinin kararlı, istenilen hızlarda ve fazlarda üretilmesini sağlayan bloktur.
- DSP Slice veya DSP48: FPGA içerisinde yüksek işlem gücü gerektiren çarpma işlemlerinin kolaylıkla yapılması ve filtre tasarımının kolayca gerçeklenmesini sağlayan yalnızca çarpma işlemi yapan özelleşmiş bloklarıdır.
- Transceiver: Yüksek hızlı seri iletişim kanallarıdır. FPGA ile dış ortam arasında yüksek hızlı veri akışını sağlamakla görevli bloklardır. (GTX≈ 5.5Ghz, GTH≈ 11.2Ghz, GTZ≈ 2.8Ghz).
- PCIe (Peripheral Component Interconnect express) arayüzü: Günümüzde genellikle bilgisayarla donanım arasında hızlı haberleşme gerektiren durumlar için kullanılan PCIe için hazırlanmış bloklarıdır.
- Block RAM: Hafıza işlemlerinin yoğun olarak kullanılmasını sağlayan özel hafıza bloklarıdır.

Piyasada bulunan farklı FPGA üreticilerinin farklı modellerde üretimleri vardır. Bu yelpazeden uygun ürünü seçmek başlı başına bir iştir. Bu çalışmada piyasada FPGA konusunda önder kabul edilen Xilinx firması tercih edilmiştir.

Sadece firma seçimi, uygun FPGA'i seçmek için yeterli değildir. Seçilen firmanın daha önceden bahsedildiği gibi ek özellikleri arasından uygun seçimler yapılmalıdır. Her bir ek özellik beraberinde ek bir maliyet getirmektedir.

Xilinx firması ürünleri Spartan ve Virtex olarak ikiye ayrılmaktaydı. Spartan serisi genellikle düşük mantık blok sayısına sahip olduğu için maliyeti düşürmek adına seri üretime girecek projelerde tercih edilmektedir. Virtex serisi ise genellikle yüksek mantık blok sayısına sahiptir ve ek özellikleri beraberinde getirir. Bu sebeple proje geliştirme süreçlerinde veya çok yüksek işlem gücü gerektiren uygulamalarda tercih edilmektedir. Xilinx firması 7 serisi ürünleri çıkması ile beraber aralarında uçurumlar olan Spartan ve Virtex serilerinin arasına bir ara ürün yerleştirme ihtiyacı duymuştur. Spartan serisini iki ayrı seriye ayırmıştır. Düşük mantıksal ünitelere sahip Artix, orta düzey işlemler için Kintex serileri oluşturulmuştur. Virtex serisi en yüksek seri

olarak yerini korumaktadır. Piyasada yapılan birçok projenin tasarımda DSP veya ARM bazı işlemlerin üstesinde FPGA'lere göre daha kolay geldiği için yardımcı işlemci olarak kullanılır. (Örneğin floating point sayılarda bir bölme işlemi için DSP veya ARM ile C dilinde 1 satır kod yeterli gelirken, FPGA'lerde IP CORE kullanılsa bile koru yönetmek için onlarca satır kod yazılmalıdır.) FPGA yeterli kaynağa sahipse sanal bir işlemci olan Microblaze yardımcı işlemci olarak kullanılabilir fakat sanal olduğu için gerçek donanıma sahip bir işlemcinin performansını sağlamaz. Piyasadaki bu açıklığın farkına varan Xilinx firması içerisinde donanımsal olarak ARM bulunan etrafında mantıksal birimlerle çevrili bir hibrit işlemci olan ZYNQ serisini üretmiştir.

Tez konusu, optik hatlara uygulanacak bir filtre tasarımını içerdiğinden uygun alıcı-verici seçimi kritiktir. Filtre tasarımı sırasında çarpım işlemlerinin kolaylıkla yerine getirmek için uygun sayıda DSP48 bloğuna ihtiyaç duyulacaktır. Tasarım genel itibari ile çok yüksek sayıda mantıksal üniteye ihtiyaç duyacaktır. Firmanın Xilinx Virtex-6 565T adlı ürünü uygun görünmektedir. Virtex-6 ailesine ait genel bilgiler Tablo 3.1 verilmiştir.

Tablo 3.1. Xilinx Virtex-6 Ailesi Özellikleri [37]

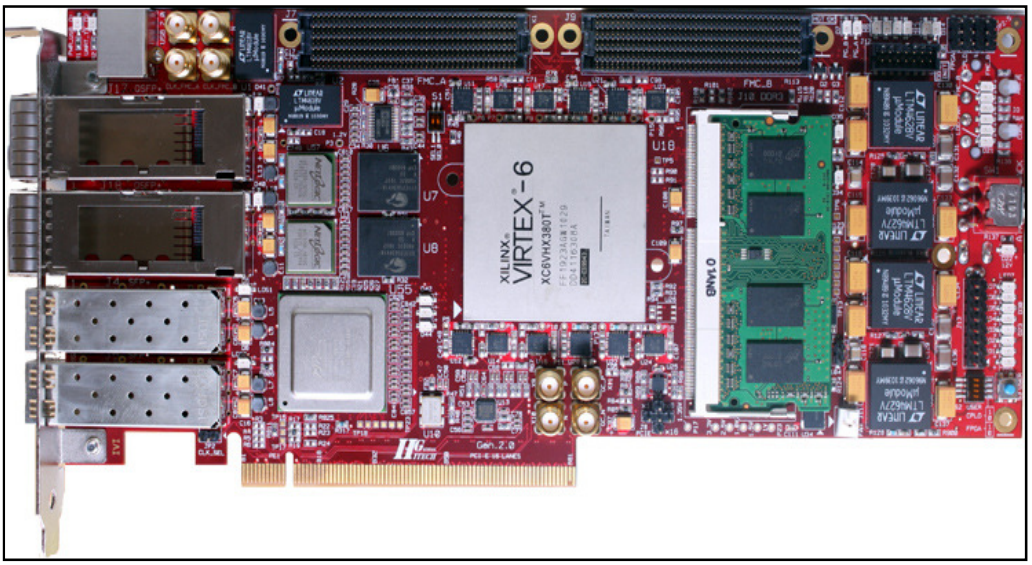
Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP48E1 Slices(2)	Block RAM Blocks			MMCMs(4)	Interface Blocks for PCI Express	Ethernet MACs(5)	Maximum Transceivers		Total I/O Banks(6)	Max User I/O(7)
		Slices(1)	Max Distributed RAM (Kb)		18 Kb(3)	36 Kb	Max (Kb)				GTX	GTH		
XC6VLX75T	74,496	11,640	1,045	288	312	156	5,616	6	1	4	12	0	9	360
XC6VLX130T	128,000	20,000	1,740	480	528	264	9,504	10	2	4	20	0	15	600
XC6VLX195T	199,680	31,200	3,040	640	688	344	12,384	10	2	4	20	0	15	600
XC6VLX240T	241,152	37,680	3,650	768	832	416	14,976	12	2	4	24	0	18	720
XC6VLX365T	364,032	56,880	4,130	576	832	416	14,976	12	2	4	24	0	18	720
XC6VLX550T	549,888	85,920	6,200	864	1,264	632	22,752	18	2	4	36	0	30	1200
XC6VLX760	758,784	118,560	8,280	864	1,440	720	25,920	18	0	0	0	0	30	1200
XC6VSX315T	314,880	49,200	5,090	1,344	1,408	704	25,344	12	2	4	24	0	18	720
XC6VSX475T	476,160	74,400	7,640	2,016	2,128	1,064	38,304	18	2	4	36	0	21	840
XC6VHX250T	251,904	39,360	3,040	576	1,008	504	18,144	12	4	4	48	0	8	320
XC6VHX255T	253,440	39,600	3,050	576	1,032	516	18,576	12	2	2	24	24	12	480
XC6VHX380T	382,464	59,760	4,570	864	1,536	768	27,648	18	4	4	48	24	18	720
XC6VHX565T	566,784	88,560	6,370	864	1,824	912	32,832	18	4	4	48	24	18	720

Notes:
1. Each Virtex-6 FPGA slice contains four LUTs and eight flip-flops, only some slices can use their LUTs as distributed RAM or SRLs.
2. Each DSP48E1 slice contains a 25 x 18 multiplier, an adder, and an accumulator.
3. Block RAMs are fundamentally 36 Kbits in size. Each block can also be used as two independent 18 Kb blocks.

Seçilen ürün olan Virtex-6 565T Virtex ailesine ait en üst düzey üründür. Bu ürünün genel özellikleri;

- Yaklaşık yarım milyon mantık bloğu
- 912 adet 36Kbitlik Block RAM
- 18 adet MMCM
- 4 adet PCIe arayüzü
- 48 adet GTX Transceiver (yaklaşık 5,5GHz)
- 24 adet GTH Transceiver (yaklaşık 11.2GHz)
- 865 adet DSP48 (Donanımsal çarpma bloğu)
- 720 adet giriş çıkış

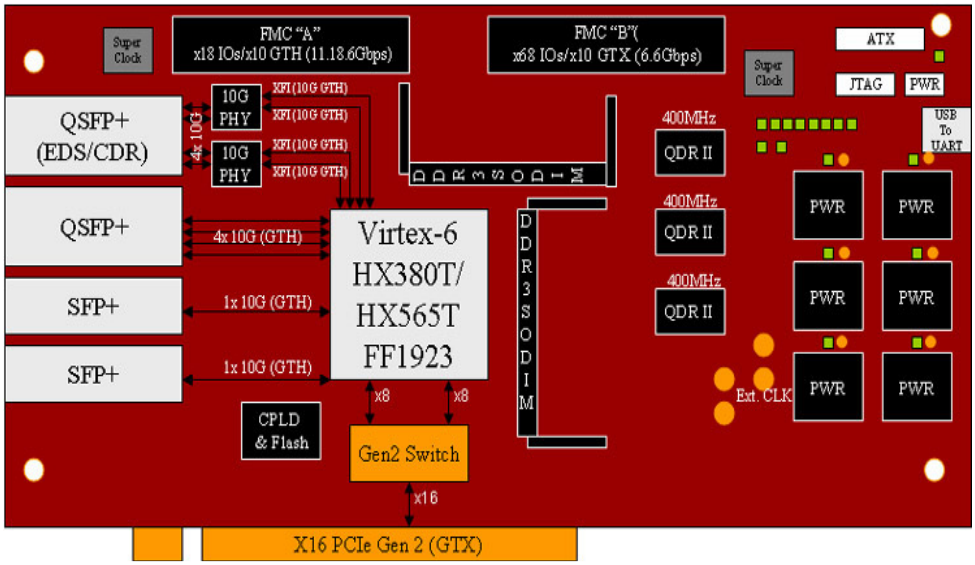
Seçilen FPGA tek başına bir şeyler ifade etmeyeceği için şetçiğimiz FPGA'i barındıran uygun bir geliştirme platform da seçilmelidir. Bu işlemciye sahip optik arayüzler için hazırlanmış piyasadaki en uygun hazır ürün Hi-TECH GLOBAL adlı firmaya ait HTG-V6HXT-X16PCIE isimli geliştirme platformudur. Geliştirme ve testler sırasında bu platformdan faydalanılmıştır. Bu platform Şekil 3.9'da, platformun sahip olduğu özellikler Şekil 3.10'da verilmiştir.



Şekil 3.9. HTG-V6HXT-X16PCIE

Bu platform istenilen özellikleri sağlayan bir FPGA'e sahip olduğu için seçilmiştir. Yüksek yeteneklere sahip bir geliştirme kartı olmasına rağmen, kullanıcı optik veriyi aynı zamanda birçok farklı arayüzden almak zorunda kalacaktır. Kullanıcı sadece bu iş için bir platform kullanmak isterse bu platform yetersiz kalacağından kendi tasarımını yapması gerekecektir. Dolayısı ile bu kart sadece ilgili kodların

geliştirilmesi ve testleri için kullanılacaktır. Bu derece yüksek özelliklere ve yeteneklere sahip bir geliştirme kartının neden projeye tam olarak uymadığına tasarım ve yöntem kısmında detaylı olarak değinilecektir.



Şekil 3.10. HTG-V6HXT-X16PCIE Platform içeriği

4. TASARIM VE YÖNTEM

Bu tezin ilk bölümünde; günümüzdeki yüksek veri aktarım ihtiyacından kaynaklanan bir problemin varlığından ve bu problemin çözümünün optik haberleşme ile giderilebileceğinden bahsedilmiştir. Bakır hatlar yerine optik hatları tercih etmemiz, bizleri optiğin kendine has problemlerinden biri olan CD bozulma ile karşı karşıya getirmiştir. Probleme ait çözümlerden biri sayısal işaret işleme yöntemidir. Zamanda konvolüsyon işlemi yerine frekansta çarpım daha kolay olduğu için frekans düzlemindeki filtreleme işlemi tercih edilmiştir. Bu çözüm için ihtiyaç duyulan algoritma ise FFT'dir.

Bölüm 3'de FFT algoritmasını verimli ve uygun bir şekilde kullanabilmek adına detayları işlenmiştir. Seçilen yöntemlerden kaynaklı farklılıklar ve yöntemlerin getirdiği işlem yükleri incelenmiştir. Sonuç olarak Radix-4 DIF FFT yönteminin bu çalışma için uygun olduğuna karar verilmiştir.

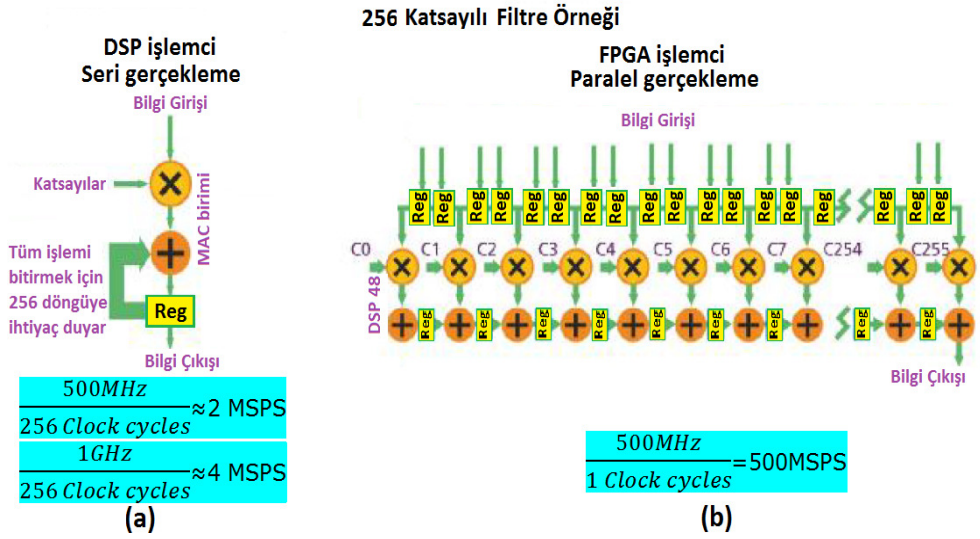
Bölüm 4'de ise incelenen ve seçilen FFT algoritmasını teorik aşamalardan uygulama aşamasına geçirmek için gerekli olan donanımların seçilmesi için genel olarak gömülü sistem teknolojileri incelenmiştir. Yüksek hızlarda bir problemin çözümü için kullanılacak yöntemin sistemle aynı hızlarda çalışabilmesi için kullanılması gereken donanım FPGA olarak seçilmiştir.

Bu bölümde; kullanılacak donanım olarak neden FPGA seçilmiştir, seçilen FPGA geliştirme kartı olan HTG-V6HXT-X16PCIE yerine nasıl bir tasarım olmalıydı, seçilen Radix-4 DIF FFT algoritmasının teoriden uygulamaya geçirirken karşılaşılan problemler ve çözüm için sergilenen yaklaşımlar gibi detaylara değinilecektir.

4.1. FPGA Donanımının Tercih Sebebi

Birçok donanım arasından FPGA'ler paralel işlem yapabilmesi sebebi ile tercih edilmektedir. Bu özelliği başlı başına tercih sebebidir. Paralel mimari tasarımı yapılabilmesi sayesinde düşük saat frekanslarında çalışsa bile çok yüksek hızlarda işlem gerçekleştirilebilir.

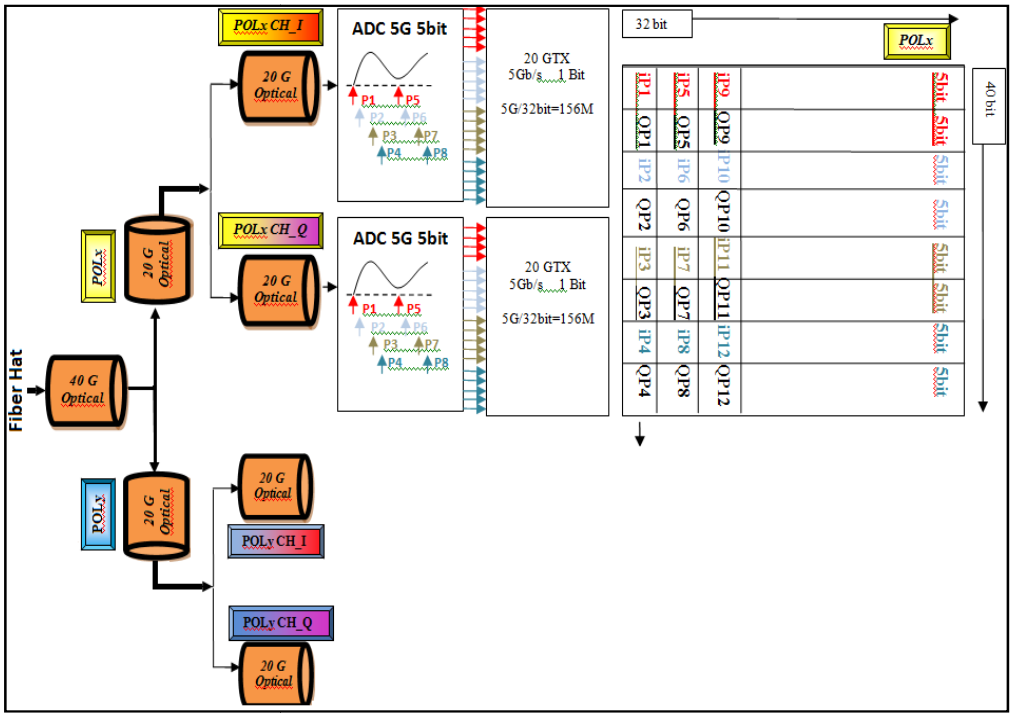
Şekil 4.1a'da gösterildiği gibi 256 örneğin çeşitli katsayılar ile çarpılmasını içeren bir filtre işleminde DSP donanımı kullanılacak olursa; DSP sadece bir çarpma ünitesi barındırdığı için 256 örnek, sırası ile çarpıcı birime giriş yaparken ilgili katsayılar birer birer çarpıcı ünitesinin diğer girişini besleyecektir. Dolayısı ile 256 örnek 256 saat işaretinde işlenebilecektir. Eğer FPGA teknolojisi tercih edilecek olursa Şekil 4.1.b'de verildiği gibi 256 adet çarpma ünitesi paralel olarak kullanılacak, her çarpma ünitesinin bir girişi ilgili katsayı ile sabit bir şekilde beslenebilecektir ve girişten aynı anda verilen 256 adet bilgi 1 saat işaretinde işlenerek sonuca aktarılabilecektir. Sonuç olarak DSP işlemcisinin çalışma frekansı daha yüksek olsa bile FPGA'in işlem kapasitesinin yanında düşük kalacaktır.



Şekil 4.1. a)DSP (seri) b)FPGA (paralel) [38]

4.2. Optik Hat için Kullanılması Gereken Donanım Tasarımı

Bu tez çalışmasında CD düzeltmesi sırasında kullanılacak sayısal filtrenin önemli bir parçası olan FFT algoritmasıdır. Her ne kadar işlemler FPGA içinde yani sayısal olarak geçecek olsa da optik işaretin FPGA içerisine alınması gerekmektedir. Proje, optik haberleşme için gerçekleştirilecek haberleşme donanımı olarak düşünüldüğü vakit, özel bir tasarım gerekecektir. Bu tasarımın maddi olarak desteklenmedikçe ürün olarak üretilmesi başlı başına bir problemdir. Bu tezde yalnızca uygun donanım üzerinde çalışacak FFT algoritmasının geliştirmesi amaçlandığı için hazır ürün olarak HTG-V6HXT-X16PCIE seçilmiştir. Fakat yinede uygulama için gereken tasarımın yapılabilirliğinin ispatlanması gerekmektedir.

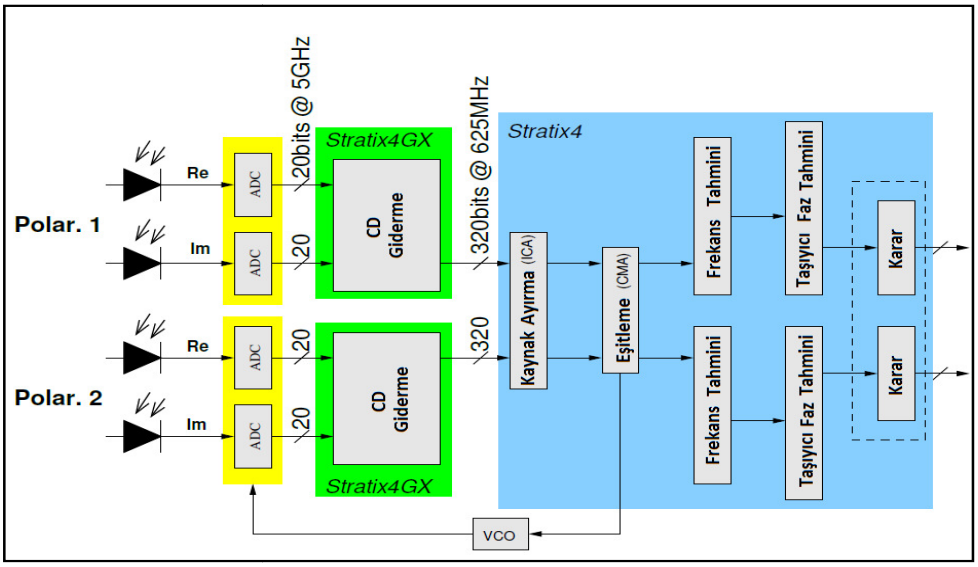


Şekil 4.2. 40G optik işaretin FPGA içerisine alınması

Şekil 4.2'de görüldüğü üzere; 40 Giga örneklik veri optik hattan gelir. Bu optik veri, polarizasyon ayırıcı sayesinde 20 Giga örneklik veriler halinde gruplanır. Gruplardan her biri hem gerçel hem sanal iki kısımdan oluşmaktadır. Bu analog bilgiler zaman-serpiştirmeli olarak adlandırılan farklı fazlarda örnekleme yapan 5 Giga örnekleme hızında 5 bitlik ADC'lerden geçirilir. Sayısala dönüştürülen 5 GHz hızındaki veriler, Xilinx FPGA'lerde GTX (hızlı iletim kanalları) [39] ile FPGA içine alınarak sayısallaştırılmış olmaktadır.

GTX'le ADC'lerin hızından dolayı 5Gbit/s hızında kullanılmak istenirse ve kullanım modu olarak 32bitlik çalışma modu seçilirse, FPGA 156MHz (5000Mhz/32bit=156,25Mhz) hızında çalışmak zorunda olacaktır. 40 GTX eş zamanlı olarak kullanılmalıdır. 1280 bit (32bit*40GTX=1280) aynı anda elimize geçecektir.

Şekil 4.2'de tasarlanan donanım, bu çalışmada hazırlanmamış olsa bile benzeri konsept projeler piyasada bulunmaktadır. Örnek olarak Alcatel-Lucent firmasının optik geliştirme platformu olarak hazırladığı TCHATER (Şekil 4.3) veya Ekinops firmasının benzer ürünleri vardır.



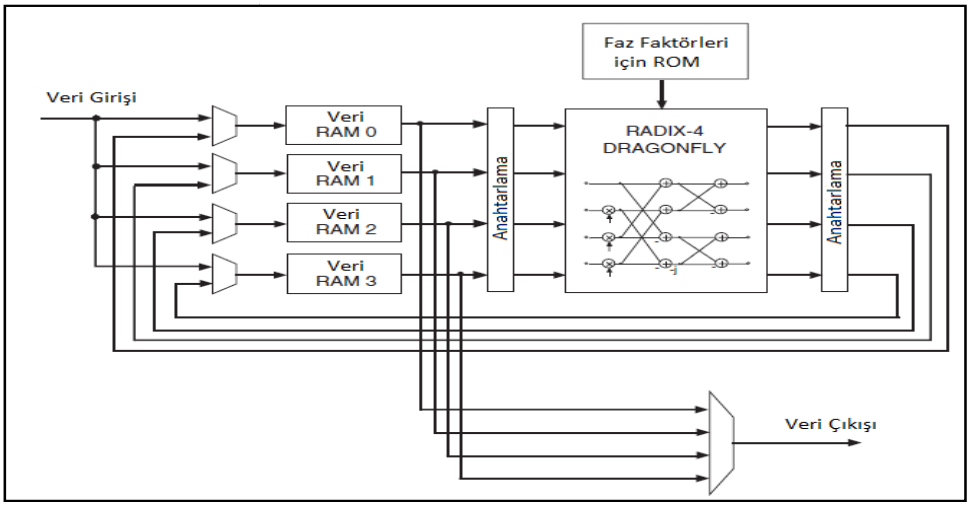
Şekil 4.3. Alcatel-Lucent TCHATER platformu [40]

Kullanılacak geliştirme kartının baştan tasarlanması yerine, FFT algoritmamızı çalıştırabileceğimiz piyasada bulunan hazır çözümlerden seçmek doğru bir karardır. Piyasadaki ürünler incelendiğinde en uygun geliştirme kartının Hi-TECH GLOBAL adlı firmaya ait HTG-V6HXT-X16PCIE kartı olduğu görülmüş ve bu kartın kullanımına karar verilmiştir.

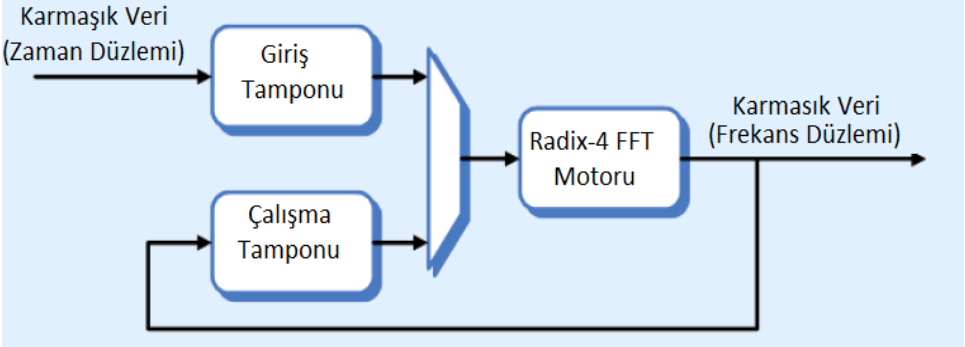
4.3. Paralel Tasarım İhtiyacı

FFT algoritmasını gerçeklemek üzere donanım olarak FPGA seçilmiştir. Bölüm 4'te FPGA'lerin bu iş için neden uygun olduğundan bahsedilmiştir. Fakat bir algoritma geliştirilirken takip edilecek adımlardan en önemlisi, seçilen uygun donanım için piyasada çeşitli çözümlerin bulunup bulunmadığını araştırmaktır. Seçilen donanım FPGA ise IP CORE adı verilen, hazır algoritmaları gerçekleştiren bloklar mevcuttur.

FFT birçok işlem sırasında kullanılabilecek önemli bir algoritma olduğundan piyasada bu konu ile ilgili bazı çözümler bulunmaktadır. Kullanılacak olan donanımın üretici olan Xilinx firmasının kendi kod geliştirme ortamında bile, FFT IP koru bulunmaktadır (Şekil 4.4). Piyasadaki bir başka FFT çözümü ise Commsonic firmasına aittir (Şekil 4.5). Piyasa birçok farklı firmanın çözümlerini bulmak mümkündür.



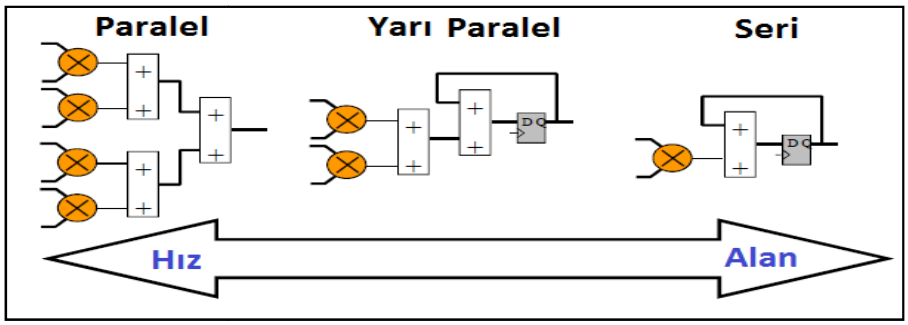
Şekil 4.4. Xilinx FFT IP CORE [41]



Şekil 4.5. Commsonic FFT IP CORE [42]

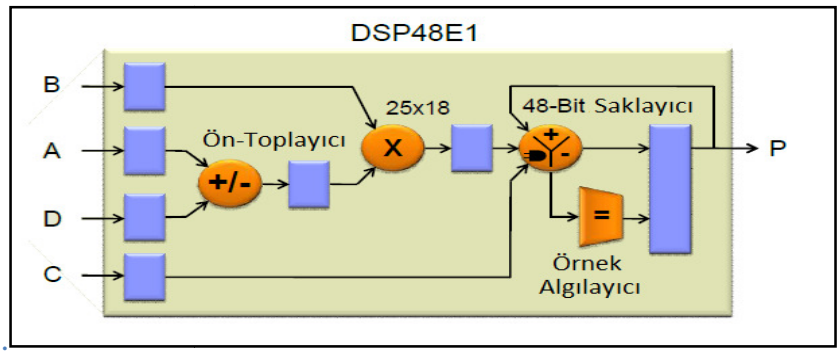
Sadece piyasadaki bu iki çözümün incelenmesi bile problemin kaynağının aydınlatılmasına yardımcı olacaktır. Piyasadaki FFT çözümleri genel kullanım için tasarlanmıştır. Genel kullanım söz konusu olduğunda mevcut kaynakların verimli kullanılması ise kritik derecede önemlidir. Yüksek hız gerektiren uygulamalar paralel bir yapıya ihtiyaç duyar ve bu sebeple yüksek alan kaplarlar. Göreceli olarak düşük hız gerektiren uygulamalar ise daha düşük alan kaplar. Kullanılacak yapılar imkân var ise ortak kullanım bloklarına dönüştürülebilir. Şekil 4.6'da paralel tasarımın getirdiği hız avantajı, seri tasarımın getirdiği alan avantajı verilmiştir.

Şekil 4.4 ve Şekil 4.5'de görüldüğü gibi FFT işleminin can alıcı noktası olan çarpım işlemlerinin yapılacağı Radix-4 için ortak bir blok kullanılmıştır.

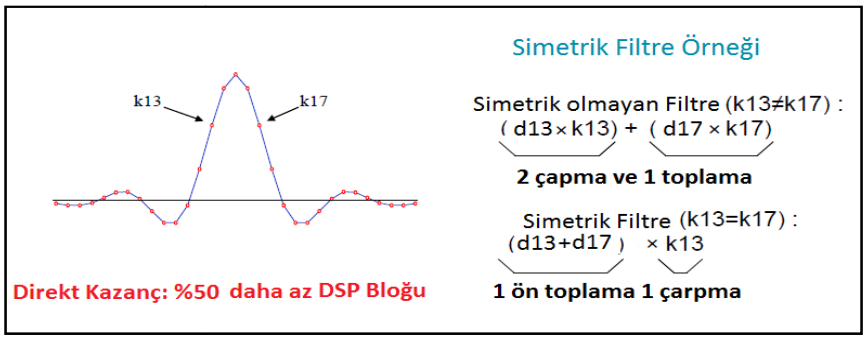


Şekil 4.6. Seri Paralel Karşılaştırması [38]

Çarpma işlemleri FPGA'ler için zorluk derecesi yüksek işlemlerdir. Günümüzde ki FPGA'lerin bu zorluğu aşmak için tasarımlarına eklediği hazır çarpıcı donanımlarından bölüm 4.1.3'te bahsetmiştik. Xilinx firmasının FPGA'lerindeki hazır çarpıcılar DSP48 blokları olarak adlandırılır. Bu bloklar gerçekte içlerinde çarpıcıdan fazlasını barındırırlar. Şekil 4.7'de görüldüğü gibi bir adet ön toplayıcı bloğa eklenmiştir. Böylece filtre yapıları gerçekleştirilirken simetrik katsayılar ile çarpım yapılmadan önce ilgili değerler toplanılarak işlem yükü yarıya indirilebilmektedir, bu kazanç Şekil 4.8'de gösterilmiştir.



Şekil 4.7. DSP48 Çarpma bloğunun içyapısı [38]



Şekil 4.8. DSP48 ön toplayıcının sağladığı kazanç [38]

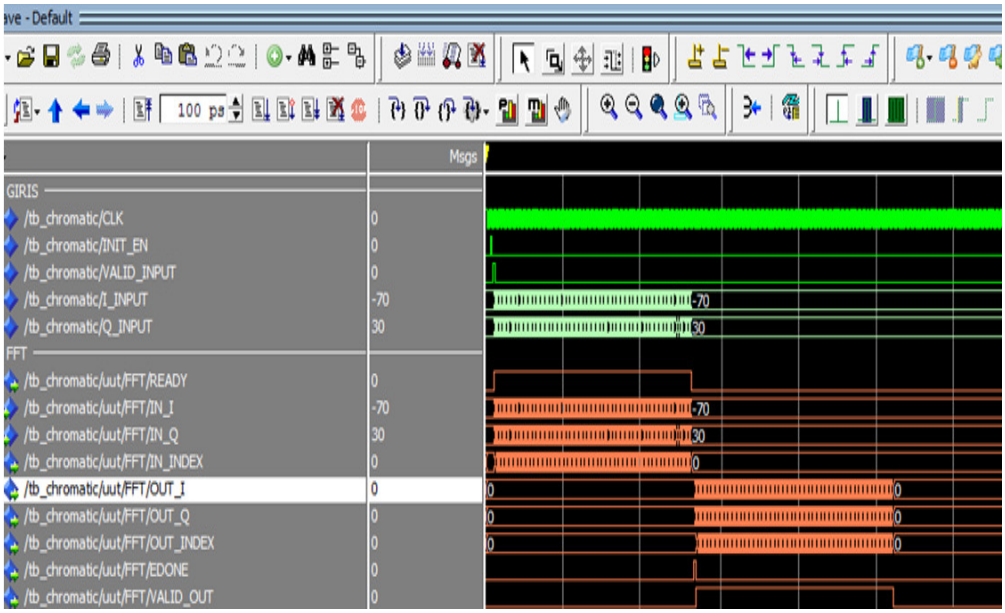
Şekil 4.8'de simetrik filtrenin 13. ve 17. katsayıları eş değerlere sahiptir. Sonuç hesaplanırken (k13 x deger13) + (k17 x deger17) işlemi yapılırsa 2 çarpma ve 2 toplama yapılacaktır. Ön toplayıcı kullanılırsa ifade; (deger13+deger17)x k13 şeklinde 1 ön toplama ve bir çarpma yapılarak daha kısa şekilde gerçekleştirilebilir.

256 nokta Radix-4 FFT tasarım aşamalarında ilk olarak Xilinx firmasının çözüm olarak sunduğu IP CORE denenmiştir. İlk deneme için geliştirme ortamında 128 nokta FFT bloğu hazırlanmıştır. Kod sentezlenmiştir. Harcanan kaynaklar Şekil 4.9'de verilmiştir.

Specific Feature Utilization:				
Number of Block RAM/FIFO:	2	out of	912	0%
Number using Block RAM only:	2			
Number of BUFG/BUFGCTRLs:	1	out of	32	3%
Number of DSP48E1s:	40	out of	864	4%

Şekil 4.9. 128 Nokta FFT için kaynak kullanımı

Daha önceden rastgele seçilen bir giriş dizisinin MATLAB üzerinde FFT'si hesaplanmıştır. Hazırlanan FFT bloğuna aynı girişler uygulanarak sonuçlar (Modelsim – Questa yazılımı kullanılarak) simülasyon üzerinde gözlemlenmiştir. Simülasyon akışı Şekil 4.10'da verilmiştir.



Şekil 4.10. 128 nokta FFT için simülasyon akışı

Şekil 4.10'da test vektörü bilgilerinin girdi olarak verilmeye başlandığını gösteren darbe işareti(VALID_INPUT) ile birlikte sırası ile 128 giriş verisi FFT bloğuna uygulanmış, 128. ve son veri ile birlikte blok, çıkış bilgilerini sağlamaya başlamıştır.

Şekil 4.4'de mimarisi gösterilen seri şekilde çalışan IP CORE'un simülasyon akışı Şekil 4.10'da incelenmiştir. Bölüm 5.2 Şekil 4.2 ayrıntılı olarak anlatıldığı üzere optik hattan bir saat işaretinde FFT'si alınacak tüm değerleri aynı anda alınacaktır. Xilinx'in mevcut çözümü kullanılacak olursa bir saat işaretinde alınan tüm veriler 128 saat işaretinde FFT bloğuna alınacak ve 128 saat işaretinde tüm çıkış bilgilerini elde etmemizi sağlayacaktır. 128 saat işareti boyunca IP CORE hesaplama yaptığı için meşgul olacak dolayısı ile optik hattan her bir saat işaretinde gelen, FFT'si hesaplanacak grup halindeki verileri işleyemeyecektir.

Bu sorunu aşmak için IP CORE'lardan paralel şekilde 128 tane kullanmak bir çözüm gibi görülebilir. Fakat Şekil 4.9'daki sentez sonucu kaynak kullanımı incelenirse görülecektir ki; 40 adet DSP48 bloğu kullanılmaktadır. 128 adet FFT IP CORE'u kullanılırsa 40*128=5120 adet DSP48 bloğu gerekecektir. Kullandığımız Xilinx marka ve serisinin en üst ürünlerinden olan Virtex-6 565T FPGA bünyesinde sadece 864 adet DSP48 koru bulundurmaktadır. Bu çözüm kaynak sınırlaması sebebi ile kullanılamamaktadır. Tekrar hatırlatılmalı ve dikkat çekilmelidir ki; yapılan karşılaştırma 128 nokta FFT için kaynak kullanımı olmasına rağmen bize 256 nokta FFT gerekmektedir. Yaklaşık olarak 2 katı kaynak ihtiyacı olacaktır.

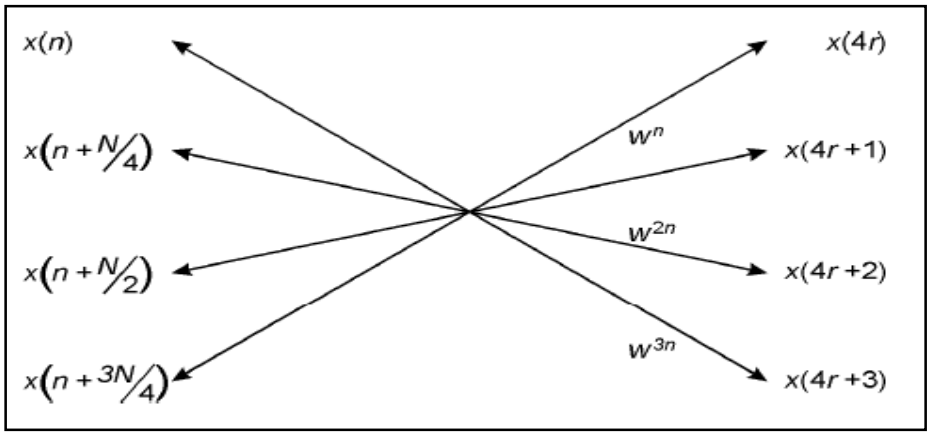
FFT probleminin özel bir tasarım gerektirdiği ve piyasada sunulan standart çözümlerle aşılmayacağı aşikârdır. Kullanıcıların her gün karşılaşmadığı bir probleme çözüm olarak, piyasadaki IP CORE üretici firmalarından hazır ürün beklemesi pek olası görünmemektedir. Bu sebepler, mevcut problem doğrultusunda daha hızlı çalışması gereken bir kor üretimini gerektirmektedir. Paralel tasarımların, seri tasarımlara üstünlüğü göz önünde tutulduğunda, tamamen paralel 256 nokta Radix-4 DIF FFT korunun tasarlanması gerekmektedir.

4.4. Radix-4 DIF

Bu bölümde FFT tasarımında kullanılacak Radix-4 DIF yöntemi anlatılacaktır. Radix-4 donanım uygulamalarında mevcut yaklaşımların ve çalışmaların incelenmesi

tasarım sırasında bize yol gösterecektir.

FFT işleminin teknik detayları Bölüm 3.3'te incelenmişti. 256 nokta FFT için Denklem (3.7)'de formül kullanılırsa; basamak= $\log_4 256$ ifadesinden işlemin 4 basamakta gerçekleşeceği görülmektedir. Her bir basamak 64 adet farklı Dragonfly düğümünden oluşacaktır. Şekil 2.5'de gösterilen Radix-4 DIF için dragonfly yapısı ayrıntılı olarak Şekil 4.11'de verilmiştir.



Şekil 4.11. Radix-4 DIF FFT için dragonfly düğümü [44]

TI firmasının DSP donanımları için hazırlamış olduğu uygumla dokümanı ([44]) detaylı olarak incelendiğinde, dragonfly düğümü için girişler gerçel ve sanal olarak;

x(n)=xa+jya (4.1)

x(n+n/4)=xb+jyb (4.2)

x(n+n/2)=xc+jyc (4.3)

x(n+3n/4)=xd+jyd (4.4)

Benzer şekilde çıkışlar;

x(4r)=xa'+jya' (4.5)

x(4r+1)=xb'+jyb' (4.6)

x(4r+2)=xc'+jyc' (4.7)

$x(4r+3)=xd'+jyd'$ (4.8)

Son olarak Faz faktörleri ise;

$W^n = Wb = Cb+j(-Sb)$ (4.9)

$W^{2n} = Wc = Cc+j(-Sc)$ (4.10)

$W^{3n} = Wd = Cd+j(-Sd)$ (4.11)

Şeklinde ifade edilir. Yukarıdaki denklemler kullanılarak Dragonfly düğümüne göre yeniden düzenlenirse;

$xa' = xa+xb+xc+xd$ (4.12)

$ya' = ya+yb+yc+yd$ (4.13)

$xb' = (xa+yb-xc-yd)Cb-(ya-xb-yc+xd)(-Sb)$ (4.14)

$yb' = (ya-xb-yc+xd)Cb+(xa+yb-xc-yd)(-Sb)$ (4.15)

$xc' = (xa-xb+xc-xd)Cc-(ya-yb+yc-yd)(-Sc)$ (4.16)

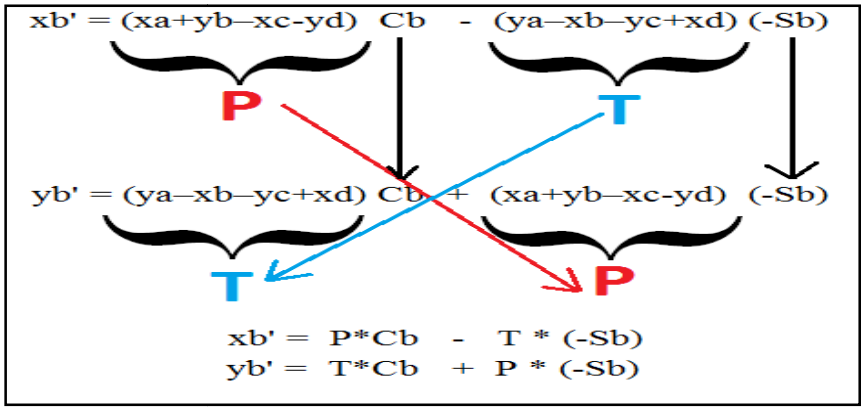
$yc' = (ya-yb+yc-yd)Cc+(xa-xb+xc-xd)(-Sc)$ (4.17)

$xd' = (xa-yb-xc+yd)Cd-(ya+xb-yc-xd)(-Sd)$ (4.18)

$yd' = (ya+xb-yc-xd)Cd+(xa-yb-xc+yd)(-Sd)$ (4.19)

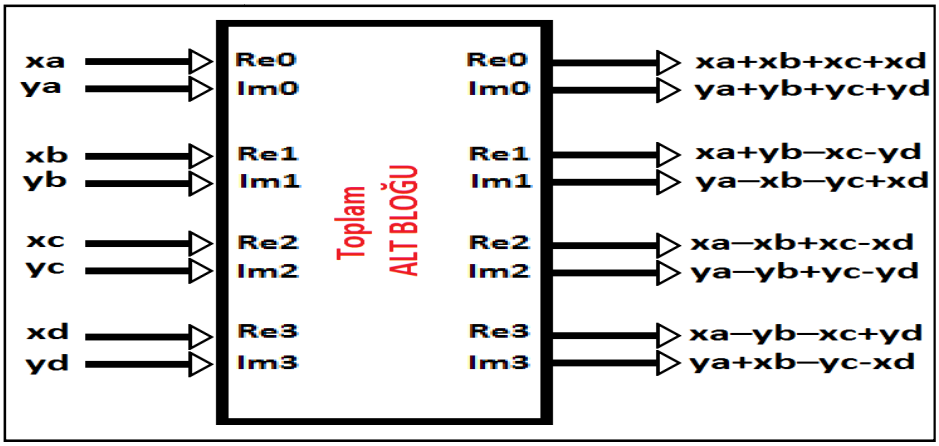
Denklem (4.12) ve (4.13) incelenecek olursa ilk faz faktörünün W^0 olduğu görülmektedir. $\mathrm{Sin}(0) = 0$ olduğu için denklemlerdeki sanal kısımlar yok olur. $\mathrm{Cos}(0) = 1$ olmasından dolayı sadece toplama işlemleri yapmak bu ifadelerin hesaplanması için yeterli olacaktır. Dolayısı ile her dragonfly düğümünde ilk dal çarpım işlemi yapılmaksızın hesaplanabilecektir. Dragonfly düğümündeki diğer dallar incelenecek olursa ((4.14) – (4.15), (4.16) – (4.17) ve (4.18) – (4.19) denklem grupları) 90, 180, 270 dereceye denk gelen çarpanların ifadelerle kaynaştığı ve işlem yükü getirmediği rahatlıkla anlaşılır. Bu denklemler çiftlerinden herhangi biri incelenecek olursa (örneğin; Denklem (4.14) ve (4.15)) faz faktörleri her iki denklemde aynıdır. Hesaplanan kat sayılar ise sadece yer değiştirmiştir (Şekil 4.12).

İfadeler ortak kısımlara sahip olduğundan, tasarım sırasında bu bloklar bir kez kullanılarak alandan kazanç sağlanacaktır.



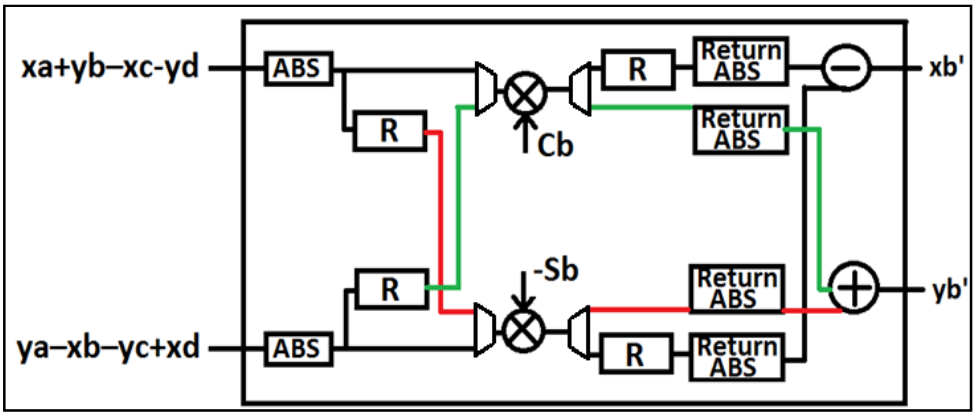
Şekil 4.12. Radix-4 ortak ifadeler

Dragonfly düğümleri incelenecek olursa, 4 adet değeri elde etmek için 4'ü gerçel 4'ü sanal toplam 8 giriş gerekmektedir. Tüm denklemler için kullanılacak ifadeler bu 8 girişin çeşitli şekillerde toplam veya farklarının o düğüme ait faz faktörleri ile çarpımlarından oluşacaktır. Her düğüm toplamların ve çarpımların yapıldığı 2 farklı alt bloktan oluşmaktadır.



Şekil 4.13. Toplam Bloğu Mimarisi

FPGA'ler kullanılarak toplama işlemlerinin hesaplandığı bloğun mimarisi Şekil 4.13'te verilmiştir. Bu blokta her dragonfly düğümünde yapılacak çeşitli toplamların tüm sonuçları tek adımda hesaplanmaktadır.



Şekil 4.14. Çarpma Bloğu Mimarisi

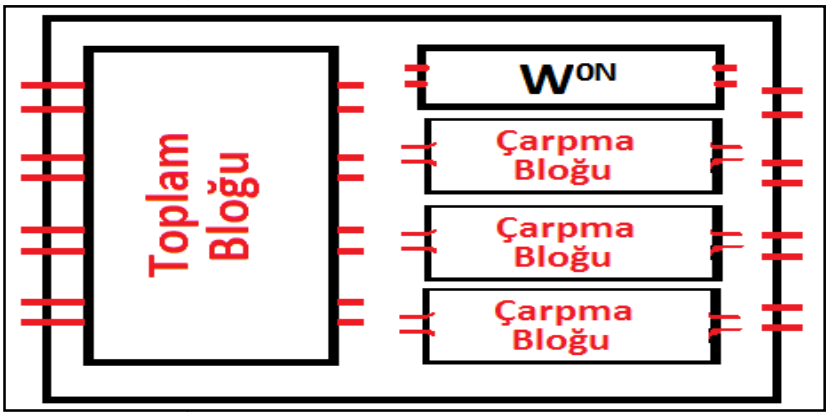
Çarpım işlemlerinin gerçekleştirildiği bloğun yapısı Şekil 4.14'de verilmiştir. Bu bloğun çalışma yapısı açıklanırken denklem çiftlerinden Denklem (4.14) ve Denklem (4.15) kullanılmıştır. Blokta yapılması gereken işlemler Şekil 4.12 verilmiştir. Yapılacak işlemleri özetlemek gerekirse sırası ile;

- Bloğa giren her girişin mutlak değeri alınır.
- Girişlerin işaretlerinin kaybolmaması için değerler bir yazmaçta saklanır.
- Girişlerden her biri ilgili faz faktörü ile çarpılırken girişlerin bir kopyası bir yazmaçta saklanır.
- Çarpılan ve hesaplanmış olan değerler bir yazmaca yazılırken, beklemedeki saklanan kopya değerler çaprazlanarak diğer faz faktörleri ile çarpılır.
- Girişteki işaret değerlerini tutan yazmaçtaki bilgiye göre sayıya işareti tekrar kazandırılır.
- Çarpım sonrasında formüldeki toplam ve farklar işlemleri uygulanarak değerler hesaplanmış olur.

Bu blokta dikkat edilecek en önemli nokta, çarpma işlemleri FPGA'ler için zorlayıcı bir süreç olduğundan; gerçekleştirilen çarpım birimi 2 kez kullanılmaktadır. Dolayısı ile diğer blokların hızına yetişmesi için 2 katı hızda çalıştırılmak zorunda olmasıdır. Bölüm 5.2'de çalışma frekansı için yeter olarak öngörülen 156MHz bu çarpım işlemini gerçekleştirmesi için 312MHz olmak zorundadır.

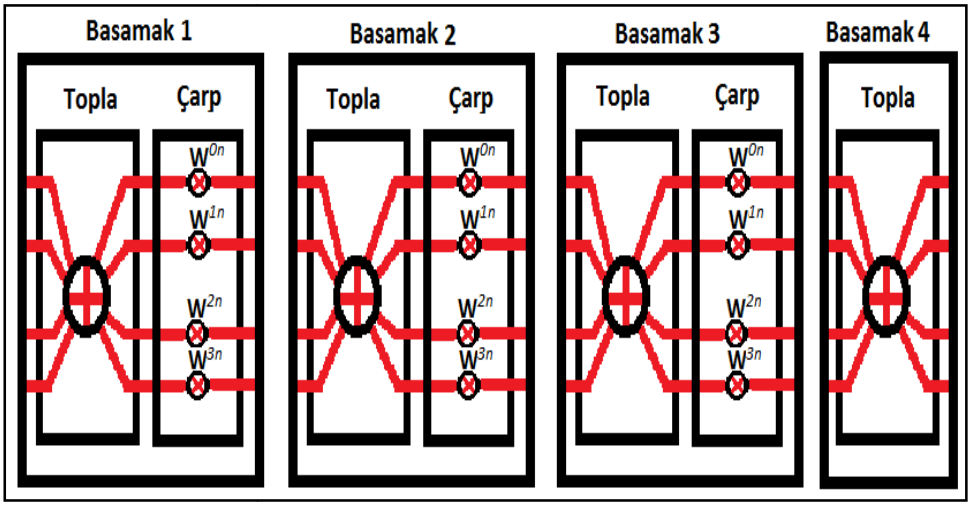
Bir toplama ve 3 çarpma alt bloğu kullanılarak bir dragonfly düğümü gerçekleştirilebilir. Dragonfly düğümü için gerçekleştirilen düğüm için mimari genel hatları ile Şekil 4.15'te verilmiştir. Çarpama blokları haricinde W^{0N}için özel bir blok

da tasarımda yer almaktadır. Her ne kadar daha önceki bölümlerde W^{0N} için çarpma yapılmadığından bahsedilse de, geliştirilen metotta normalizasyon amacı ile kaba bir çarpım işlemi yapılmak zorundadır. Bu normalizasyon işlemi ile ilgili detaylar bölüm 5.5'de ayrıntılı olarak anlatılmaktadır.



Şekil 4.15. Dragonfly düğüm mimarisi

256 nokta FFT işlemi geçekleştirilirken 4 basamak olacağından bahsedilmişti. Her bir basamak 64 adet Dragonfly düğümü bulunduracaktır. Her bir basamağı Dragonfly yapıları yerine toplam ve çarpım alt sistemleri şeklinde ifade etmek hem hiyerarşiyi daha düzenli tutmamıza, hem de sistemde oluşabilecek hatayı analiz ederken kontrol edilecek bölümleri doğru tespit etmeye olanak sağlayacaktır. Genel hatları ile mimari tasarım Şekil 4.16'da gösterilmiştir.



Şekil 4.16. 256 nokta FFT mimarisi

Şekil 4.16'da dikkat çeken nokta basamak 4 bloğunda çarpma bloğuna ihtiyaç duyulmamasıdır. FFT ile ilgili incelenecek olursa son basamaktaki çarpma işlemlerinin W_N^0 çarpanına denk geldiği görülmektedir. Bu çarpandan dolayı son basamak sadece toplama (ve çıkartma) işlemleri ile gerçekleştirilebilir.

4.5. Faz Faktörü ve Normalizasyon

256 nokta FFT işlemi için çarpma bloklarında kullanılması gereken faz faktörlerinin hesaplanması gerekmektedir. Denklem (3.2)'de $W_N^{kn} = e^{\frac{-j2\pi kn}{N}}$ ifadesinde N=256 yerine yazılırsa, ifade $W_{256}^{n} = e^{\frac{-j2\pi n}{256}}$ halinde olacaktır. Faz faktörlerinin Euler formülü ile ifadesi Denklem (4.20)'de verilmiştir,

$$W_N^n = \cos\left(\frac{2\pi n}{N}\right) + j.\sin\left(\frac{2\pi n}{N}\right) \tag{4.20}$$

Denklem (4.20)'den görüleceği gibi her bir faz faktörü birim çember üzerindeki bir değerden ibarettir. Bu değer hem sanal hem de gerçel kısımlardan oluşmaktadır. 256 nokta FFT için Radix-4 denklemleri (Şekil 2.8'de verilen denklemler) uygulanınca görülecektir ki birim çember üzerindeki tüm değerler kullanılmayacaktır.

Basamak 1 için W_{256}^{kn} incelendiğinde; k=1 değerini alırken n değeri 0'dan 63'e kadar değerleri alacaktır. Sırası ile dizilen 64 dragonfly düğümlerinde birinci dallar W_{256}^0 değerini alacaktır. İkinci dallar W_{256}^{1n} ifadesinden dolayı 0'dan başlayarak 63'e kadar 1'er 1'er artan değerleri alacaktır. Üçüncü dallar W_{256}^{2n} ifadesinden dolayı 0'dan başlayıp 126'ya kadar 2'şer 2'şer artan değerleri alacaktır. Dördüncü ve son dallar W_{256}^{3n} ifadesinden gelen 0'dan başlayarak 189'a kadar 3'er 3'er artan değerleri alacaktır. Kısaca birinci basamakta;

- Birinci dallar = [$W_{256}^0, W_{256}^0, W_{256}^0 \ldots\ldots W_{256}^0, W_{256}^0, W_{256}^0$]
- İkinci dallar = [$W_{256}^0, W_{256}^1, W_{256}^2 \ldots\ldots W_{256}^{61}, W_{256}^{62}, W_{256}^{63}$]
- Üçüncü dallar = [$W_{256}^0, W_{256}^2, W_{256}^4 \ldots\ldots W_{256}^{122}, W_{256}^{124}, W_{256}^{128}$]
- Dördüncü dallar = [$W_{256}^0, W_{256}^3, W_{256}^6 \ldots\ldots W_{256}^{183}, W_{256}^{186}, W_{256}^{189}$]

İkinci basamak için W_{256}^{kn} incelendiğinde; k=4 değerini alırken n değeri 0'dan 16'ya kadar değişecektir. Önceki basamaktaki gibi 64 dragonfly olmasına rağmen, gruplamalar değişmiştir. Birinci dallar benzer şekilde W_{256}^0 değerini alacaktır. İkinci

dallar W_{256}^{4n} gruplamasını kullanacaktır. Üçüncü dallar W_{256}^{2x4xn} ifadesinden dolayı W_{256}^{8n}, dördüncü dallar W_{256}^{3x4xn} ifadesinden dolayı W_{256}^{12n} şeklinde kullanılacaktır. İkinci basamakta;

- Birinci dallar = [W_{256}^{0}, W_{256}^{0}, W_{256}^{0}……W_{256}^{0}, W_{256}^{0}, W_{256}^{0}]
- İkinci dallar = [W_{256}^{0}, W_{256}^{4}, W_{256}^{8}……W_{256}^{52}, W_{256}^{56}, W_{256}^{60}]
- Üçüncü dallar = [W_{256}^{0}, W_{256}^{8}, W_{256}^{16}……W_{256}^{104}, W_{256}^{112}, W_{256}^{120}]
- Dördüncü dallar = [W_{256}^{0}, W_{256}^{12}, W_{256}^{24}……W_{256}^{156}, W_{256}^{168}, W_{256}^{180}]

Üçüncü basamak için W_{256}^{kn} incelendiğinde; k=16 değerini alırken n değeri artık 0'dan 4'e kadar değişecektir. Birinci dallar W_{256}^{0} değerlerini alırken ikinci dallar W_{256}^{1x16xn} yani W_{256}^{16n} , üçüncü dallar W_{256}^{2x16xn} yani W_{256}^{32n} , dördüncü dallar ise W_{256}^{3x16xn} yani W_{256}^{48n} değerlerini alacaktır. Üçüncü basamakta;

- Birinci dallar = [W_{256}^{0}, W_{256}^{0}, W_{256}^{0}, W_{256}^{0}]
- İkinci dallar = [W_{256}^{0}, W_{256}^{16}, W_{256}^{32}, W_{256}^{48}]
- Üçüncü dallar = [W_{256}^{0}, W_{256}^{32}, W_{256}^{64}, W_{256}^{96}]
- Dördüncü dallar = [W_{256}^{0}, W_{256}^{48}, W_{256}^{96}, W_{256}^{144}]

Bu ifadeler Şekil 4.17'de görsel olarak da verilmiştir, Fakat 256 nokta FFT çok fazla değişken içerdiği için detaylar net olarak görüntülenememektedir.

Yapılan hesaplamalar ile 256 nokta FFT için en yüksek W_{256}^{189} ifadesinin kullanılacağı hesaplanmıştır. Bazı değerlerinse hiçbir basamakta kullanılmayacağı görülmektedir. Kullanılsın veya kullanılmasın ifadeler birim çemberin 256 eşit parçaya bölünmesiyle hesaplanmaktadır.

Birim çember üzerindeki değerler Denklem (4.20)'ye göre incelendiğinde; gerçek kısmı $\cos\left(\frac{2\pi n}{256}\right)$ ve sanal kısmı $\sin\left(\frac{2\pi n}{256}\right)$ terimlerinden oluşmaktadır. Sinüs ve kosinüs ifadelerinden en büyük değer olarak "1" gelirken, en düşük değer olarak "0" değeri gelmektedir. Asıl problem "1" ve "0" değerleri arasındaki katsayılardır. Bu değerler kayan noktalı (floating point) sayılar kullanılarak ifade edilmek zorundadır.

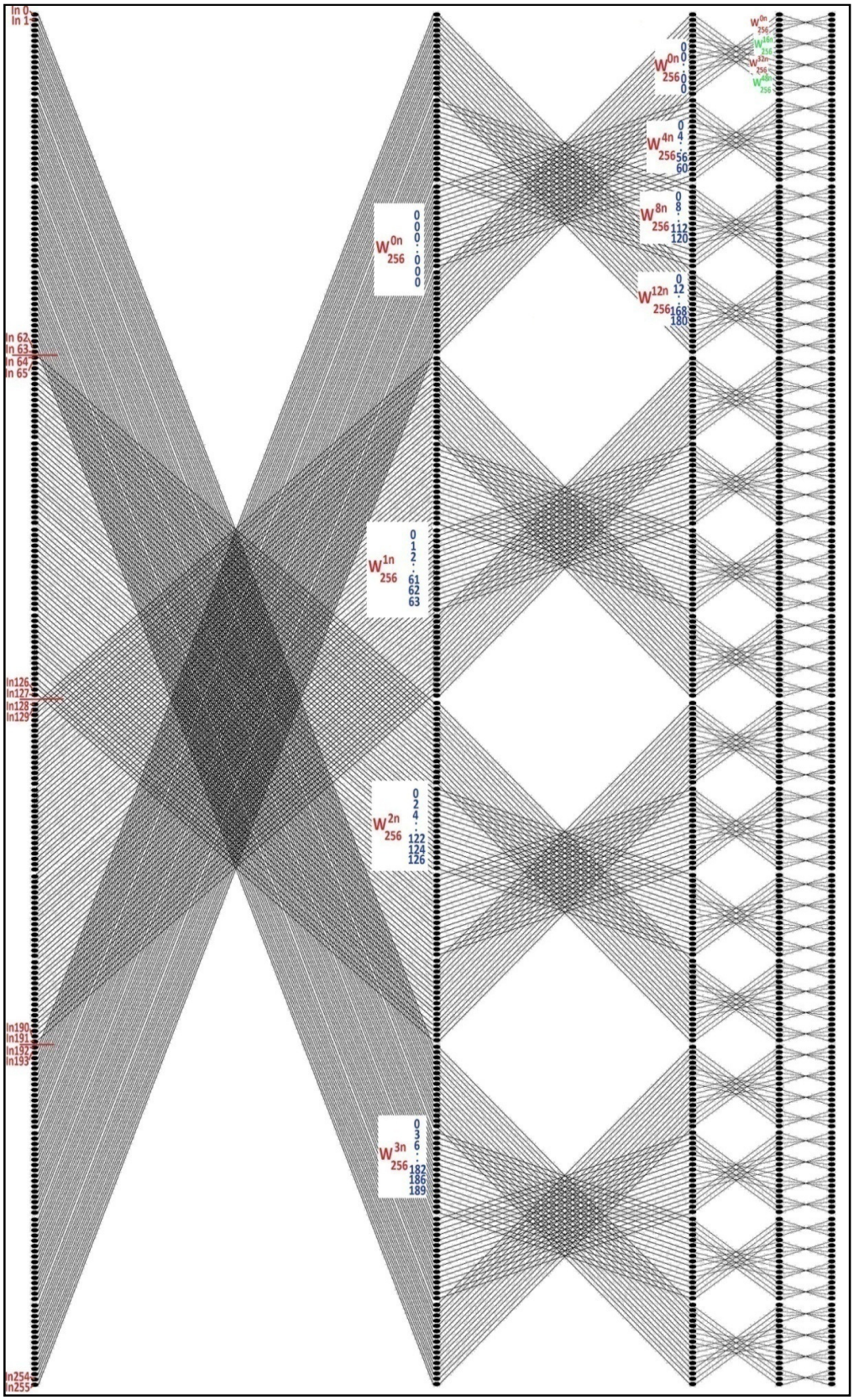
Tasarlanan sistem optik hat iletim hızlarına uyum sağlaması gerektiğinden yüksek hızda çalışmalıdır. Kayan noktalı sayılar ile işlem yapmak oldukça zordur. Optik hat hızlarında, 256 nokta FFT gibi yüksek işlem yükü barındıran bir algoritmayı kayan

noktalı sayıları kullanarak gerçeklemek için FPGA üzerinde yeterince kaynak yoktur. Kayan noktalı sayılar kullanmak yerine sabit noktalı (fixed point) sayıları kullanarak işlemleri yapmak, ihtiyaç duyulan donanım gereksinimlerini azaltacak ve veri işleme süresini kısaltacaktır [45].

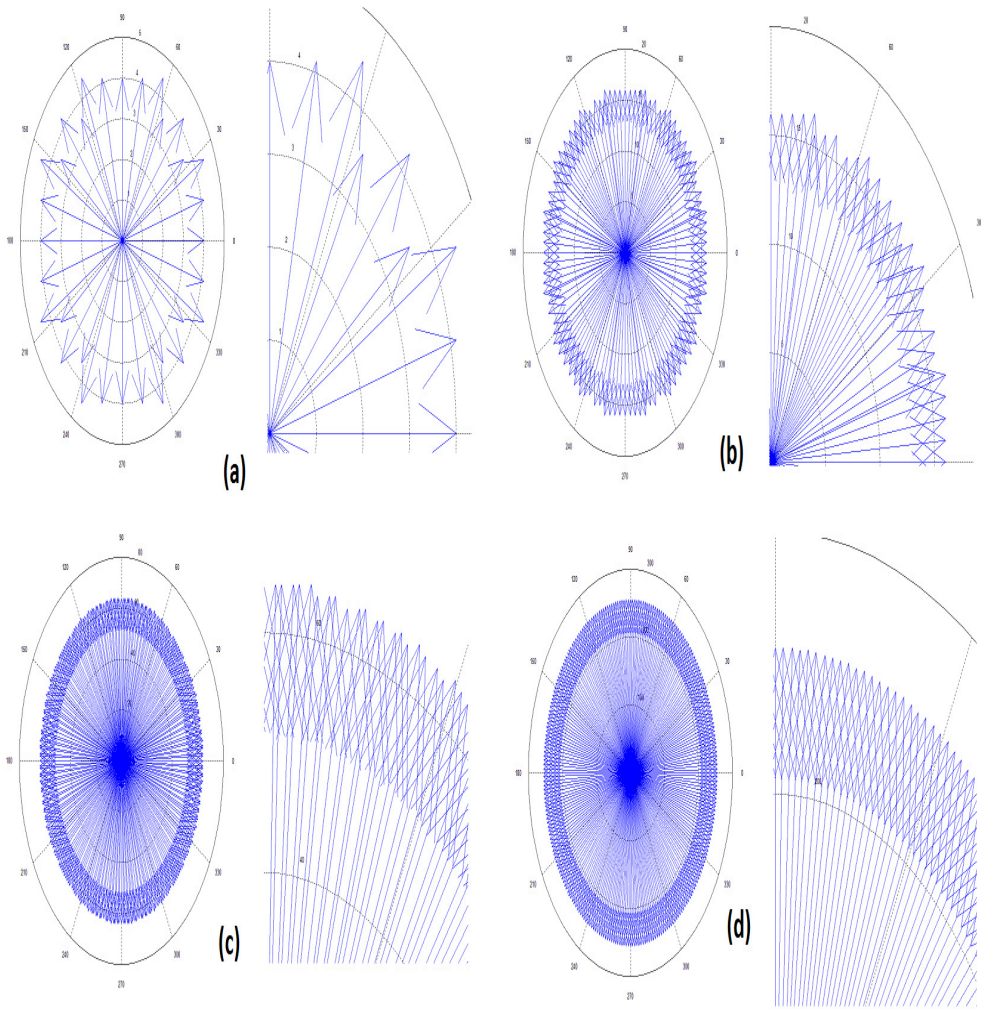
Ondalık sayılarla çalışmak yerine, tüm değerlerin bir kat sayı ile genişletilerek normalize edilmesi işlemleri kolaylaştıracaktır. Bu işin dezavantajı ondalık sayıların bir yöntem (flooring-aşağı en yakın tam sayıya yuvarla, ceiling-yukarı en yakın tam sayıya yuvarla, round-en yakın tam sayıya yuvalara ve truncation- virgülden sonrasını atarak yuvarla) ile yuvarlanması sonucu oluşacak hatalardır. Yuvarlama hataları haricinde dikkat edilecek bir başka problem ise normalizasyon sonucunda çarpılan değerlerin bit düzeyindeki genişleyişidir. 256 nokta FFT için 4 basamağın 3'ünde çarpım işlemleri mevcuttur. Her çarpım sonucunda değer bit düzleminde genişlemektedir. Örneğin 1 bit ile ifade edilen "1" sayısı, 2048 yani 2^{11} ile normalize edilecek olursa artık 11 bit ile ifade edilebilecektir. Her basamaktaki faz faktörü ile çarpımda oluşan bit genişlemesi sonucu kaynaklar yetersiz gelmeye başlayacaktır. Bu genişlemeyi engellemek için her FFT basamağında, seçilen yuvarlama yöntemlerinden biri kullanılır ve her basamaktaki genişleme kontrol altına alınmış olur.

Bit genişlemesi probleminin üstesinden gelinebilmesi için normalizasyon değeri oldukça küçük seçilmeye çalışılırken, 256 parçaya bölünen birim çemberde ifade edilen iki ardışık W_{256}^{n} değeri ayırt edilebilir şekilde farklı olmalıdır. Yetersiz genişletme yapıldığında, yuvarlama sonrasında ardışık değerler üst üste bineceğinden veya tüm ardışık değerler arası mesafe aynı olmayacağından hata oranı artacaktır. Farklı genişletme oranları için faz faktörlerinin değerlerinin karşılaştırması yapılmıştır. Bu karşılaştırma görsel olarak Şekil 4.18'de gösterilmiştir. Faz faktörlerinin farklı normalizasyon değerleri için sayısal karşılaştırması Tablo 4.1'de gösterilmiştir.

Uygulamaya bağlı olarak normalizasyon değerleri değiştirilebilir. Yani hata toleransı yüksek olan bir uygulama için, daha fazla yuvarlama hatasına izin vermek adına daha düşük bir normalizasyon değeri seçebilir. Düşük normalizasyon sonucunda ise, bit genişlemesi azalacak dolayısı ile lojik kaynak kullanımı azalacaktır.



Şekil 4.17. 256 nokta FFT şematiği



Şekil 4.18. Normalizasyon katsayısı a)4 b)16 c)64 d)256 iken faz faktörleri

Şekil 4.18 ve Tablo 4.1'de görüldüğü gibi küçük normalizasyon değerleri kullanılacak olursa; 256 eş parçaya ayrılması beklenen birim çember, eş olmayan parçalara ayrılacaktır ve çözünürlük yetersiz kaldığından birçok nokta üst üste gelecektir. Ondalık basamaktan en yakın tam sayıya yuvarla yöntemi kullanılırsa ve sabit noktalı sayılarla tanımlama yapılırsa çözünürlük arttıkça noktaların birim çembere dağılımı daha homojen bir hale gelecektir.

Tablo 4.1 karşılaştırma değerleri incelenirse; 16 ile normalizasyon sonucunda n=0 ve n=1 değerleri aynı nokta üzerinde olacaktır. Yani n=1 değeri ile çarpımın ayırt ediciliği kalmayacaktır. Ancak 1024 ile normalizasyon yapıldığında, değerlerin ayırt edici olmaya başlayacağı görülecektir.

Tablo 4.1. Faz faktörlerinin farklı değerlerle normalizasyonu

		16 ile Normalizasyon	64 ile Normalizasyon	256 ile Normalizasyon	1024 ile Normalizasyon
n=0	1,000 + 0,000i	16+0i	64+0i	256+0i	1024+0i
n=1	0,999 - 0,024i	16-0i	64-2i	256-6i	1024-25i
n=2	0,998 - 0,049i	16-1i	64-3i	256-13i	1023-50i
n=3	0,997 - 0,073i	16-1i	64-5i	255-19i	1021-75i
n=4	0,995 - 0,098i	16-2i	64-6i	255-25i	1019-100i
n=5	0,992 - 0,122i	16-2i	64-8i	254-31i	1016-125i
n=6	0,989 - 0,146i	16-2i	63-9i	253-38i	1013-150i
n=7	0,985 - 0,170i	16-3i	63-11i	252-44i	1009-175i
n=8	0,980 - 0,195i	16-3i	63-12i	251-50i	1004-200i
n=9	0,975 - 0,219i	16-4i	62-14i	250-56i	999 -224i
n=10	0,970 - 0,242i	16-4i	62-16i	248-62i	993 -249i

Her basamakta faz faktörü ile çarpma işlemi için 1024 ile normalizasyon yapıldığında, 10 bitlik genişleme olacaktır. 256 nokta FFT sırasında 3 basamak faz faktörleri ile çarpma getirdiğinden toplamda 30 bitlik genişleme olacaktır. 256 basamaklı FFT işlemi sonunda 30 bitlik kırpma yapılacak olursa; FFT işlem sonucu hata oranı düşük olacaktır, fakat yüksek bitli sayılarla işlem yapılmak zorunda olduğundan mantıksal kapılar yeterli gelmeyecektir. Her basamakta 1024 (2^{10}) ile normalizasyon yapıldıktan sonra değerler belli oranda kırpılırsa, bit genişlemesi daha sınırlı olacaktır.

Faz faktörleri ile ilgili olarak tasarımın şekillenmesini sağlayan asıl önemli nokta; Faz faktörlerinin her dragonfly düğümü ve düğümdeki her dal için önceden hesaplanabilir sabitler olmasıdır. Şekil 4.17 de gösterildiği gibi tasarlanan, 256 nokta Radix-4 DIF FFT için W_{256}^{n} kullanılarak tüm noktalarda çarpma işleminde kullanılacak faz faktörleri hesaplanabilir. Dolayısı ile bahsedilen çarpma işlemlerindeki çarpanlardan biri değişken, biri sabit olacaktır. FPGA'ler için zorlu işlemler olan çarpma işlemlerinin üstesinden gelinirken bu önemli özellik kullanılacaktır.

4.6. Çarpma İşlemi için Uygulanan Yöntem

Çarpma işlemi sonuçları gündelik hayatta çeşitli yollar ile kolayca bulunabilir olmasına rağmen, FPGA gibi bir donanım ile bu işlemi gerçekleştirirken zorluklarının farkına varılacaktır. Hazır bir çarpma donanımı yoksa çarpma işlemi defalarca toplama yapmaktan ibarettir. Defalarca kelimesi "C" gibi bir dilde "for"

olarak yorumlanacaktır. Fakat "for" döngüsü işlemi donanımsal olarak yorumlanacak olursa sıralı bir mantığı yani seri kullanılan bir mimariyi belirtmektedir. Seri mimari bölüm 5.3'de bahsedildiği gibi optik hızlarda kullanım için yeterince hızlı değildir.

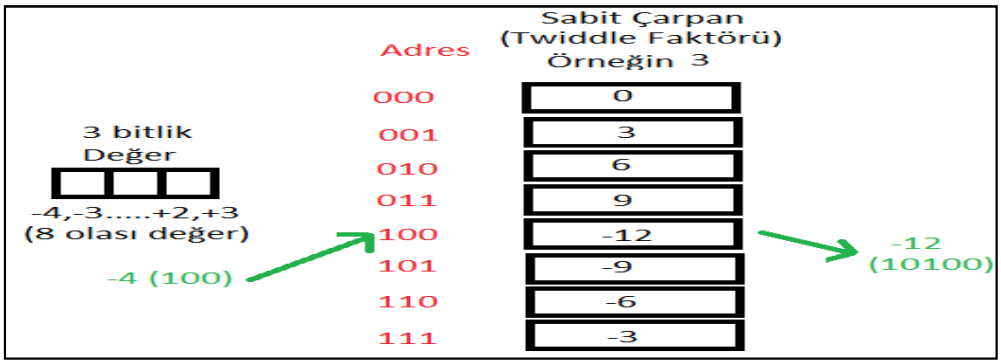
Bölüm 5.3'de Xilinx Virtex-6 565T FPGA'ler için DSP48 adlı hazır donanımlardan bahsedilmiştir. Bu donanımsal birimler, çok hızlı bir şekilde çarpma işlemlerini gerçekleştirebildiği halde, FPGA içerisinde sınırlı kaynağa sahiptir. Bu değerli kaynakları başka fonksiyonlar için yedekte bekletmek daha makul bir çözüm olacaktır. Hem tasarımda DSP48 kullanıldıktan sonra, donanım olarak farklı bir FPGA markasına ait benzer özel çarpma donanımı kullanıldığı zaman tasarım değiştirilmek zorunda kalacaktır.

Tasarıma başlanmadan önce problemin detaylı şekilde incelenmesi ve problemin düzgünce tanımlanması her proje açısından önemlidir. En temel yapıtaşı niteliğindeki alt bloklarda oluşan hatalar, hiyerarşik açıdan incelendiğinde üst bloklarda birikmeli hatalara sebep olacaktır. Benzer şekilde alt bloklar tasarlanırken, kaynak açısından mantıksal kapıların verimli kullanımı sayesinde üst bloklarda ciddi alan kazanımları göze çarpacaktır. Örneğin alt blok tasarlanırken fark edilerek tasarımdan çıkartılan bir yazmaç sayesinde, 1024 adet alt bloğun birleşmesinden oluşan bir üst blokta 1024 yazmaç kazanç sağlanacaktır.

256 nokta FFT işleminde çarpma işlemi incelenirken, çarpanlardan biri olan faz faktörlerinin sabit ve önceden belirlenebilir olması büyük kolaylık sağlayacaktır. Böylece işlemler tek bir değişkene bağımlı hale gelmiş olacaktır.

Sabit ile çarpma işlemi için akla gelen ilk yöntem hafıza temelli bir mimaridir. Değişken sabit bir çarpan ile çarpıldığı zaman sadece belli değerleri alabilmektedir. Eğer bu değerler önceden hesaplanarak FPGA içerisindeki Blok RAM'lere yazılırsa, her yeni gelen değerde hesap yapmak yerine tablodan uygun karşılığı çekmek, makul bir çözüm olacaktır. Hafıza temelli çarpım örneği Şekil 4.19'da verilmektedir. Mevcut çalışmalar incelendiğinde, literatürde benzer çalışmalar göze çarpmaktadır [46]. Fakat bu yöntem incelendiğinde, bazı problemler yüzünden mevcut tasarımımız için uygun olmadığı anlaşılmıştır. 256 nokta FFT için her basamakta normalizasyon, çarpma ve toplamlar yüzünden genişleyen değerler iki tabanında daha fazla sayıda bitlerle ifade edilmeye başlayacaktır. Daha fazla sayıda bit kullanımı demek ise daha

yüksek miktarda hafıza ünitesine ihtiyaç duymak demektir. FPGA içerisinde yer alan blok RAM'ler yeterli gelmemektedir ve blok RAM'lerin istenildiği şekilde bit bölümlemeleri ile kullanılamadığından dolayı ihtiyaç duyulan hafıza alanından daha fazlası gerekmektedir.



Şekil 4.19. Hafıza temelli çarpma yöntemi

Tasarım için gereken hafıza alanını hesaplama yapmak gerekirse; girişler 5 bit olarak kararlaştırılmıştır. 256 nokta FFT'nin birinci basamağını, toplam ve çarpım alt blokları olarak ayrılmıştır. Birinci basamak toplam bloğu 5 bit işaretli 4 sayının dragonfly düğümüne göre toplandığında en fazla 7 bit ile ifade edilebilir.

Birinci basamaktaki çarpım blokları , W_{256}^{0n}, W_{256}^{1n}, W_{256}^{2n} ve W_{256}^{3n} ifadelerinde n=0 dan n=63'e kadar değerler alacaktır. W_{256}^{0} değerleri hesaba katılmaz ise geriye 189 adet değer kalmaktadır. Her değer hem sanal hem gerçel kısımları içereceği için 189x2=378 çarpım işlemi gerekecektir. Her çarpım işlemi için 1024 yani 10 bit ile genişletilerek, 9 bitlik kırpma işlemi yapılırsa 1 bitlik genişlemesine imkân tanınarak 8 bit halinde sonuçlar barındıracaktır.

Özet olarak birinci basamakta;

- 7 bitlik giriş değerleri için 2^7 adet hafıza adresi gerekecektir,
- Bu hafıza adresleri 8 bitlik değerler saklayacaktır.
- Benzer çarpım işlemleri için 378 çarpım gerekecektir.

Bu değerler için ihtiyaç duyulan hafıza hesaplamaları yapılırsa;

- Birinci basamakta: $378x8x2^7$=387.072 bit.
- İkinci basamakta: $360x12x2^{11}$=8.847.360 bit
- Üçüncü basamakta: $288x16x2^{16}$=301.989.888 bit gerekecektir.

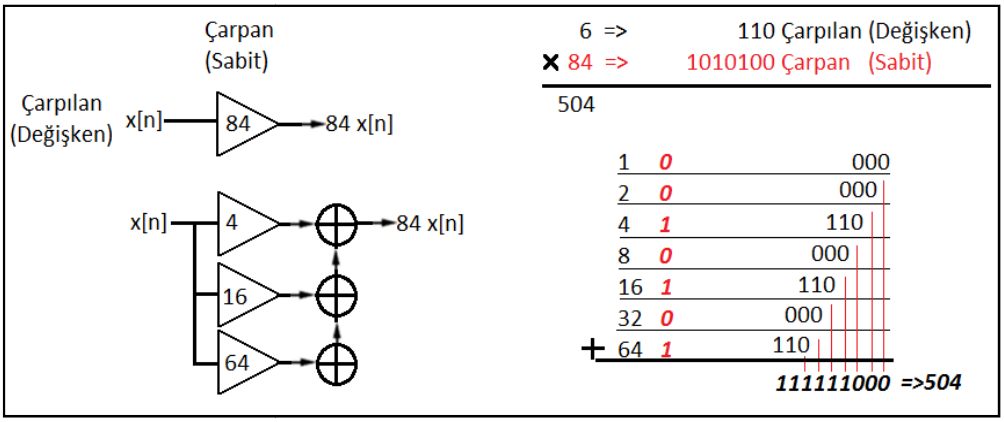
Toplamda 311.224.320 bit yani yaklaşık 311MB blok RAM gerekmektedir. Virtex-6 565T özellikleri Tablo 3.1'den incelenecek olursa 1824 adet 18K'lık blok RAM'e sahiptir. Yani yaklaşık 32Mbit hafıza alanı sağlamaktadır. Yöntem bu şekilde değerlendirildiği zaman bile yetersiz kalmaktadır. Blok RAM'lerin sadece bloklar halinde kullanılması ise bu yöntemi tamamen kullanılmaz hale getirecektir. Örneğin 64 adet adres bölgesinde değer bulunduran ve her adres hücresinde bir bitlik bilgi içeren 64 bitlik bir tablo 18K'lık bir bloğu tamamen kullanacaktır.

Kullanılan Virtex-6 565T'nin hafızası yetersiz kaldığı için FPGA dışarısına bağlanacak hafıza üniteleri ilk bakışta çözüm olarak görünebilir. Fakat optik bir hattan devamlı veri akışı olduğu düşünülecek olursa, FPGA dışarısındaki bir belleğe erişim için haberleşme protokol komutları bile sistemi yavaşlatmaya yetecektir. İstenilen hızlarda çalışmanın en uygun yolu, FPGA içerisindeki direk erişimli hafıza ünitelerini kullanmaktan geçmektedir.

Çarpma işlemini gerçekleştirmek için en temel yöntemlerden biri de kaydır-topla olarak anılan yöntemdir [47-50]. Üstelik çarpanlardan birini olan faz faktörünün önceden hesaplanabilir bir sabit olması işlemi kolaylaştırmaktadır. Sabitlerin bitlere dönüştürülmesinin ardından, sadece içerisindeki "1" ifadelerin için işlemlerin yapılacak olması ise kaynak kullanımını azaltacaktır.

Yöntemde; sabit çarpan değeri iki tabanında yazılır. Basamak değeri "0" ise; "0" çarpmanın yutan elemanı olduğundan işleme tabi tutulmaz, eğer basamak değeri "1" ise; "1" çarpmanın etkisiz elemanı olduğundan herhangi bir çarpma işlemi yapmadan olduğu gibi yazılır ve basamak değeri kadar sola kaydırılır. FPGA'ler için bit kaydırma işlemleri, n kadar sola yapıldığında 2^n ile çarpma, kaydırma sağa yapıldığında 2^n'e bölme anlamına gelmektedir. Birçok donanım için problemli işlemlerin başında gelen kaydırma işlemi FPGA'ler ile çok kolay gerçekleştirilebilmektedir. Sonuç olarak FPGA'ler için zor olarak kabul edilen çarpma işlemleri, FPGA'ler için basit olan kaydırma ve toplama işlemleri ile kolaylıkla tamamlanacaktır.

Kaydır topla metodu ile çarpma işlemi Şekil 4.20'de detaylı olarak verilmiştir. Şekilden anlaşılacağı üzere tüm çarpma işlemleri 2^n ifadesine göre yapılmaktadır.



Şekil 4.20. Kaydır-Topla metodu ile çarpma

4.7. Benzetim, Kullanıcı Arayüzü ve Otomatik Kod Üretimi

Problem öncelikle alt bloklara ayrılmış, tüm alt blok problemleri derinlemesine incelenmiş ve hepsi ile ilgili uygun çözümler geliştirilmiştir. En uygun çözümlerin teoriden pratiğe geçirilerek kodlaması gerekmektedir. Projenin genel hatların düşünüldüğünde, FFT basamakları, toplama alt bloğu, çarpma alt bloğu ve tüm dragonfly düğümleri onbinlerce satır kodun varlığından bahsedilebilir.

FPGA uygulanması için VHDL kodu yazılmaya başlandığında, işlemin, benzer iskelet yapıların (dragonfly düğümleri, çarpım alt blokları) birleşiminden meydana geldiği izlenimi oluşmaktadır. Fakat tamamen paralel bir mimari kullanılacağından her dragonfly düğümündeki çarpma alt bloğu için kendine has bir mimari kurulmak zorundadır. Çünkü her faz faktörü için kaydır topla yöntemine göre kaydırılacak değerler farklı olacaktır.

FPGA'ler için VHDL kodlaması yapıldıktan sonra, çok uzun kodlar için en hızlı bilgisayarlar bile kullanıldığında sentezleme işlemi uzun sürmektedir. Geliştirme sürecinde olası hatalar sonucunda bir satır kodun bile değiştirilmesi ise ISE'nin (Xilinx kod geliştirme ortamı) tüm sentezi baştan yapmasına sebep olacaktır. Xilinx'in yeni kod geliştirme ortamı Vivado sayesinde parçalı sentezleme de yapılabilmektedir.

En önemli aşamalardan biri, yapılan tasarımın bir benzetim ortamında denenmesidir. Bir önceki paragrafta bahsedildiği gibi fikrin direk olarak FPGA üzerinde denenmesi

(sentez ve yerleştirme) işlemleri uzatacaktır. FPGA üzerinde yapılacak hesaplamaların öncelikli olarak oluşturulan bir MATLAB benzetiminde gerçekleştirilmesi, analiz aşamasında avantaj sağlayacaktır.

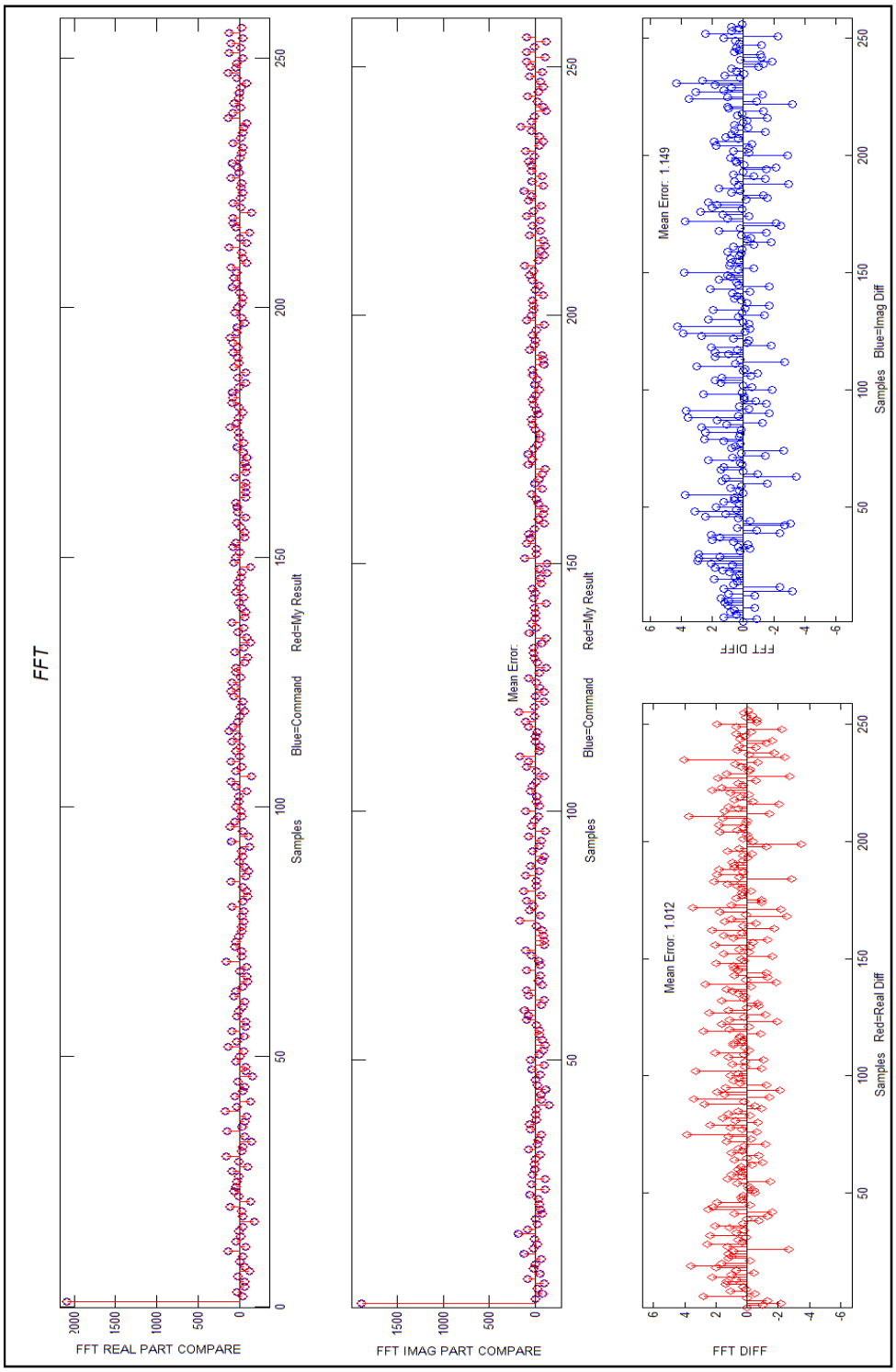
FPGA'de yapılacak işlemlerin öncelikli olarak MATLAB benzetimde gerçekleştirilmesi sırasında önemli bir detay vardır. Bilgisayar ortamında kullanılan MATLAB işlemlerini genellikle kendi hazır kütüphaneleri ile gerçekleştirir ve kayan noktalı sayı aritmetiği kullanır. FPGA üzerinde çalışacak VHDL kodunda ise tüm işlemler bit düzleminde yapılmaktadır. Bu sebeple FPGA'ler için uygun çözüm olarak kullanılması düşünülen yöntemler ile ilgili yeni kütüphaneler geliştirmek bu çalışma için kaçınılmaz olacaktır.

Bit düzlemindeki uygun kütüphaneler geliştirilerek 256 nokta FFT işlemi için gereken tüm alt blokların benzetimi gerçekleştirilmiş ve test edilmiştir. Ardından alt blokların birleşerek oluşturduğu üst mimari içinde testler yapılmıştır. FPGA benzetimi için geliştirilen kütüphaneler ile MATLAB'ın orijinal kütüphanelerine göre FFT testleri yapılmış ve sonuçlar karşılaştırılmıştır. Alınan benzetim sonuçları Şekil 4.21 üzerinde görülmektedir.

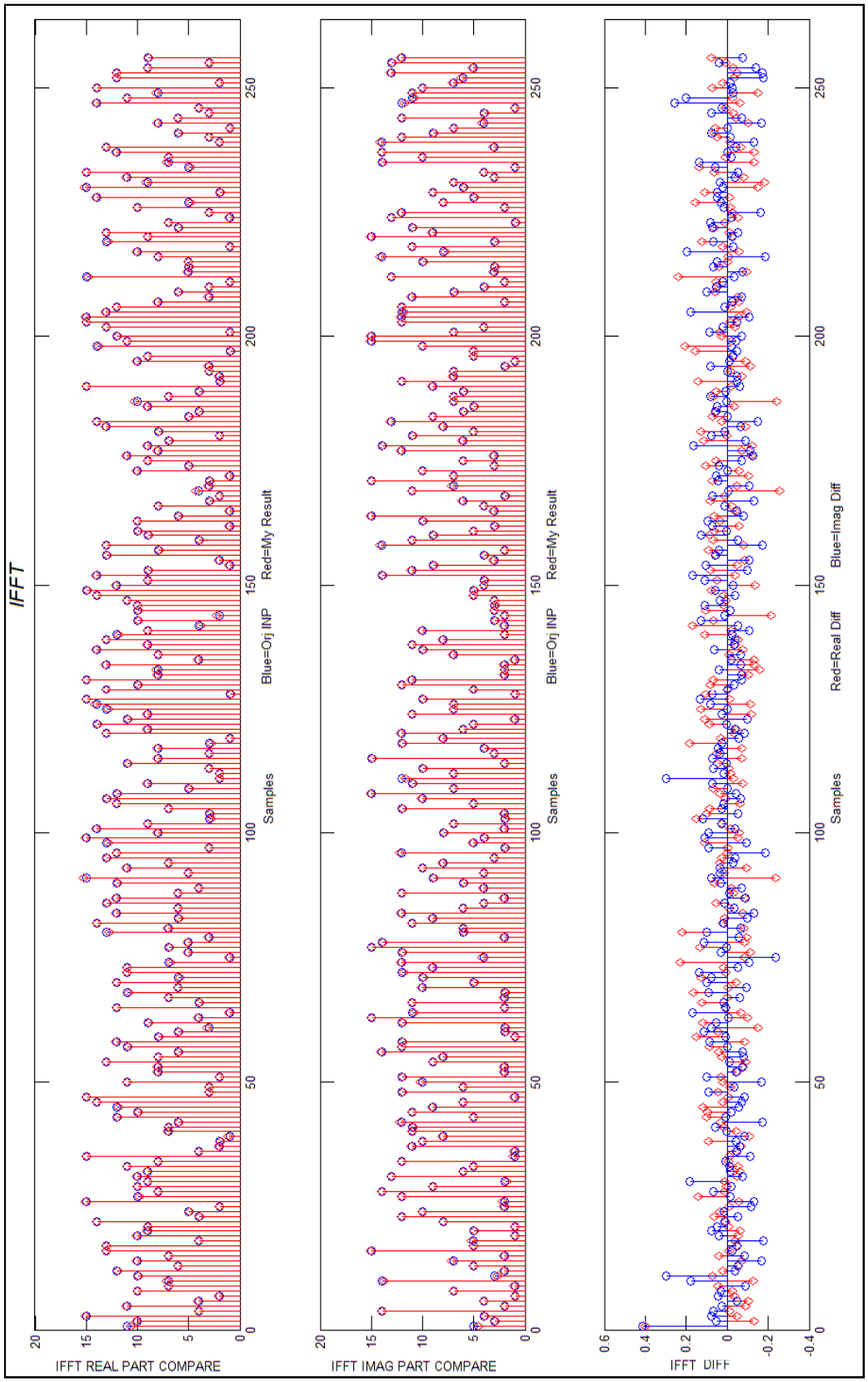
Şekil 4.21'de MATLAB orijinal kütüphaneleri ile hesaplanan FFT mavi renk ile FPGA benzetimi FFT sonucu kırmızı ile verilmiştir. Üst grafik gerçel kısımların karşılaştırması, ortadaki grafik sanal kısımların karşılaştırması ve en alttaki kırmızı renk ile çizdirilen grafik MATLAB sonuçları ile FPGA benzetim kütüphanesi sonuçlarının gerçel kısımlarının farkı, mavi kısım ise sanal kısımların farkını ortaya koymaktadır. Üstteki iki grafikte sonuçların örtüştüğü görülmektedir. Alt kısımdaki fark grafiklerinden anlaşıldığı üzere en büyük fark "± 5"'i geçmemektedir. Ortalama mutlak hata ise 1.1 civarında olmaktadır. Yuvarlama hataları ve sabit nokta aritmetiği için hata oranı kabul edilebilir düzeydedir.

Bilindiği gibi FFT'si alınan bir işaretin IFFT'si alındığında işaretin ilk hali elde edilmektedir. Şekil 4.22'de ilk grafikte mavi kısım MATLAB kütüphanesi ile sonuçları, kırmızı kısım FPGA benzetim sonuçlarını verilmektedir. Gerçel kısımların karşılaştırmasında ifadelerin örtüştüğü görülmektedir. Ortadaki grafikte aynı karşılaştırma sanal kısımlar için yapılmıştır. Son grafikte kırmızı gerçel, mavi sanal

farkları göstermektedir. Sonuç olarak farklar ±"0,6" hata ile orijinal işarete benzemektedir.



Şekil 4.21. MATLAB FFTsi ile Benzetim kütüphanesi FFT'sinin karşılaştırması



Şekil 4.22. MATLAB IFFTsi ile Benzetim kütüphanesi IFFT'sinin karşılaştırması

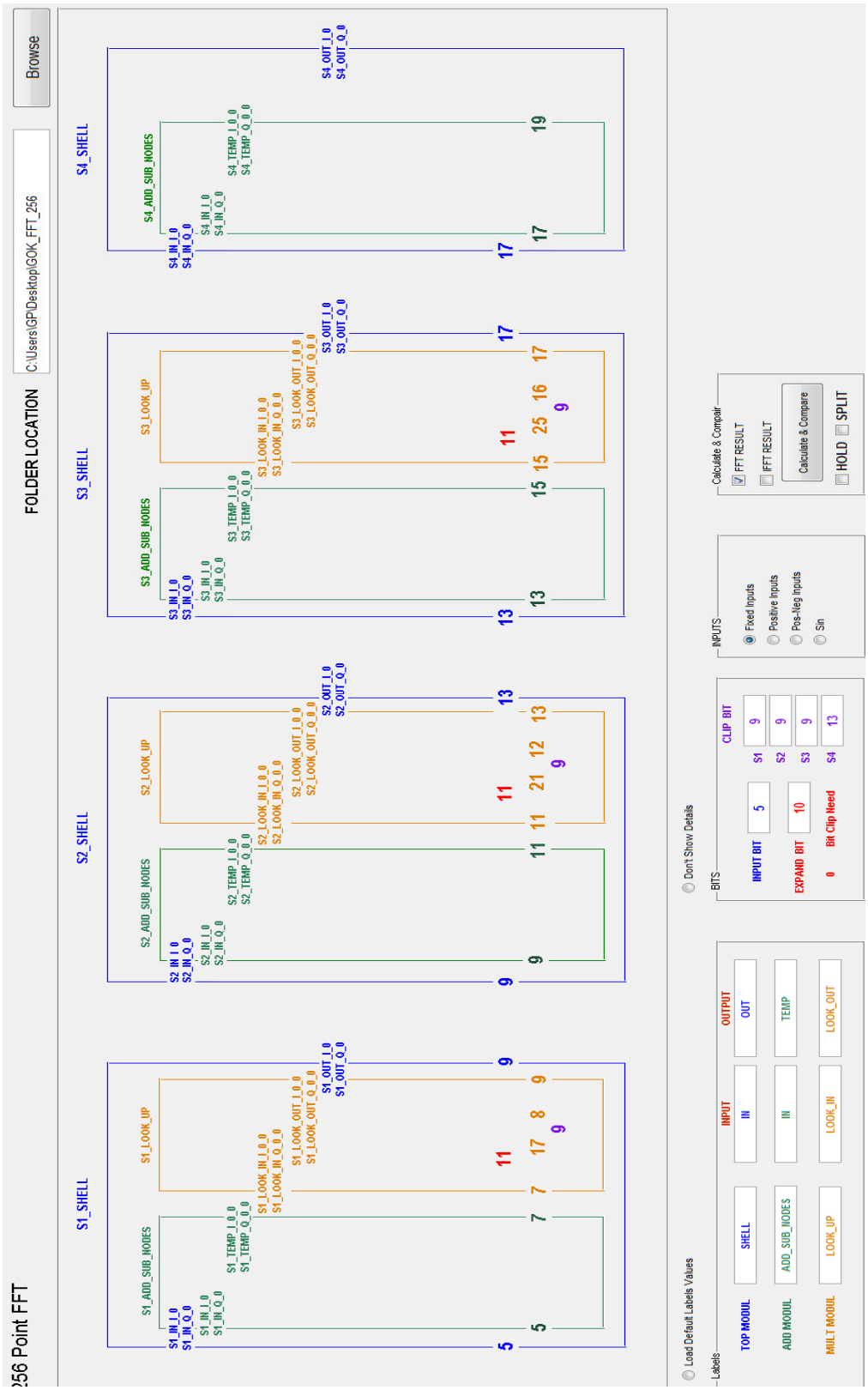
MATLAB'da FPGA benzetim kütüphanesi oluşturulmuştur. Bu sayede testlerin gerçekleştirilmesi oldukça kolaylaşmıştır. Herhangi bir kullanıcının MATLAB kodları ile uğraşmadan bazı karşılaştırmaları yapabilmesi için bir kullanıcı ara yüzü oluşturulmuştur. Bu arayüzden farklı giriş bitleri seçilebilir, farklı sayıda bit kırpılarak yuvarlama hata oranları değiştirilebilir. Kullanıcı grafik arayüzü Şekil 4.23'te gösterilmiştir. Kullanıcı grafik arayüzünde alt bloklar olan toplam ve çarpım blokları gösteren şekiller bulunmaktadır. Alt blokların birleşerek oluşturdukları üst mimariler kabuk olarak adlandırılmıştır. Her blok sonrası bit düzlemindeki sayıların genişleyişini görmek, değerlerin değiştirilmesi için gereken müdahalelerin yapılmasına kolaylık sağlamaktadır.

Arayüz oluşturulduktan sonra farklı normalizasyon değerleri, farklı giriş bit genişlikleri kullanılarak karşılaştırmalar yapılmıştır. Bu sayede farklı kullanıcılar bu tasarımı kendi istedikleri şekilde yeniden yapılandırabilecektir. Örneğin daha fazla hata toleransı olan bir uygulama için daha az kaynak tüketimi sağlayabilecektir. Tablo 4.2'de farklı normalizasyon değerleri ve giriş bitlerine göre ortalama hata değerleri gösterilmiştir.

Tablo 4.2. Farklı değerler için ortalama mutlak hata (MAE) tablosu

Giriş Bit Genişliği / Normalizasyon Katsayısı	5bit		6bit		7bit		8bit	
	Gerçel	Sanal	Gerçel	Sanal	Gerçel	Sanal	Gerçel	Sanal
1024	1.111	1.175	1.142	1.213	1.193	1.202	1.198	1.280
512	1.099	1.179	1.160	1.210	1.183	1.376	1.281	1.405
256	1.164	1.179	1.316	1.325	1.501	1.542	2.140	1.955
128	1.223	1.308	1.734	1.695	2.423	2.365	4.319	4.185
64	1.597	1.678	2.727	2.747	5.066	5.022	9.797	9.064
32	2.635	2.631	5.155	4.325	10.196	8.867	18.516	18.131
16	3.446	3.944	7.138	6.804	14.800	14.065	28.260	29.093
8	8.387	8.382	16.461	16.912	36.352	35.018	63.428	64.237

Bu projede ortaya çıkan en önemli yaklaşımlardan biri yazılacak VHDL kodunun otomatik olarak yazdırılmasıdır. MATLAB'da geliştirdiğimiz FPGA benzetim kütüphaneleri sayesinde yapılması gerek tüm işlemler FPGA davranışlarına göre hesaplanmaktadır. Tüm değerler hesaplanmışken bunları kullanılacak kodlar haline getirmek, bahsedilen onbinlerce satır kod yazmak çok daha kolay olacaktır. Yapılacak rutin işlemler için değiştirilmesi gereken kod parçacıkları, MATLAB FPGA benzetim kütüphanesi tarafından hesaplanacak ve kodlamada ilgili kısma yerleştirilecektir.



Şekil 4.23. Kullanıcı arayüzü

Farklı uygulamalar için bazı parametreler değiştiğinde, onbinlerce kodun en azından binlerce satırı değişmek zorunda kalacaktır. Oysaki kullanıcı arayüzü ile birlikte çalışan otomatik kod yazma süreci sayesinde kullanıcı ekranından sadece parametrelerinin girilmesi yeterli olacaktır. Girilen parametreler ile ilgili karşılaştırmalar ve sonuçlar kullanıcı için uygun ise; kullanıcı tarafından belirtilen bir dizine tüm VHDL kodları oluşturulmaya başlanacaktır. 256 nokta FFT işlemi için VHDL'de sadece giriş çıkışların tanımlanması için (512 giriş – 512 çıkış) 1024 satır kod gerektirmektedir. Benzer şekilde kaydır topla yöntemi için normalize edilen değerlerin iki tabanında karşılığının hesaplanarak kaydırma birimlerinin oluşturulması ve bu işlemlerin her bir dragonfly düğümü için defalarca tekrarlanması gerektiği düşünüldüğünde kodların otomatik olarak MATLAB'a yazdırılması kullanıcı için büyük kolaylıklar sağlayacaktır. Onbinlerce satır kodun üretilmesi yaklaşık olarak 6sn sürmektedir.

5. SONUÇLAR ve ÖNERİLER

Bu tezde, hızlı uygulamalar için gerçek zamanlı FFT problemi analiz edilmiştir. Problem, alt birim problemlerine ayrılmış ve her problem için uygun çözümler araştırılmıştır. Problemler için mevcut çözümler gözden geçirilmiş ve uygun görülen çözümlere kendi bakış açımızla eklentiler yapılmıştır.

Geliştirilen çözüm ve yaklaşımların tutarlılığını ispat etmek için önce MATLAB ortamında FPGA benzetim kütüphanesi oluşturulmuştur. Benzetim kütüphanesi sayesinde FPGA üzerinde gerçekleştirilmesi gereken işlemler aynı tutarlılıkta MATLAB'da hızlıca gözden geçirilme imkânı sunmaktadır. Geliştirilen FPGA benzetim kütüphanesi ışığında küçük temel farklılıklar barındıran rutin işlemler için script mantığı ile VHDL kodlama yapılmış. Otomatik kod oluşturma sayesinde herhangi bir değişikliği tüm sisteme uygulamak Matlab kodunda bir kod parçasının değiştirilmesinden ibaret olacaktır. Onbinlerce satır koda uygulanması gereken kod değişikliği saatler yerine saniyeler içerisinde düzeltilebilmektedir.

VHDL kodlar oluşturulduktan sonra Xilinx ISE ile sentezlenmiştir. Testbench'ler oluşturularak Questa (eski adı ile Modelsim) adlı simülatör ile test edilmiştir. Sonuçlar MATLAB hesaplamaları ile karşılaştırılmıştır. Kodların FPGA üzerindeki çalışması ChipScope adlı ISE eklentisi ile test edilmiştir.

Kendine has bir probleme çözüm için üretilen ve piyasada benzeri bulunmayan bir uygulama olan tamamen paralel, 9 saat işareti gecikmeye sahip (9 clock latency) 256 nokta Radix-4 DIF FFT çalışması gerçekleştirilmiştir.

Bu bölümde sonuçlar ile ilgili raporlar sunulacak ve ileride ki çalışmalar için bazı önerilerde bulunulacaktır.

5.1. Sentez, Questa ve ChipScope Sonuçları

Tezin en kritik kararı başlangıçta verilen tamamen paralel tasarım kararıydı. Bu kararın sonucu olarak daha fazla sayıda mantıksal ünite kullanılacağı öngörülmüştü. Yani farklı bir ifade ile kodun FPGA'de daha geniş alan kaplayacağı daha önceden

biliniyordu. Bu karar ihtiyaç duyulan hızlara çıkmak için verilmesi gereken bir tavizdir.

Tablo 5.1. Xilinx Sentez raporu

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	123406	708480	17%
Number of Slice LUTs	213700	354240	60%
Number of fully used LUT-FF pairs	82598	254508	32%
Number of BUFG/BUFGCTRLs	2	32	6%

```
Timing Summary:
---------------
Speed Grade: -2

  Minimum period: 2.977ns (Maximum Frequency: 335.914MHz)
  Minimum input arrival time before clock: 7.304ns
  Maximum output required time after clock: 0.659ns
```

VHDL kodu Virtex-6 565T FPGA için Xilinx ISE ile sentezlendiğinde sentez raporu Tablo 5.1 verilmiştir. Sentez sırasında optimizasyon için herhangi bir ayar yapılmamıştır. Kodun sadece sentezlenmesi hızlı sayılan bir bilgisayar ile (Intel Core i7-3970X @ 3.50GHz işlemci ve 32GB 2400MHz DDR3 RAM'e sahip) 3 saate yakın sürmektedir.

Tablo 5.1'deki sentez sonuçları incelendiğinde yarım milyon mantıksal ünitenin yaklaşık olarak yarısı kullanılmıştır. DSP48 kaynak kullanımı yoktur. Sentez zamanlama sonuçlarına göre; bölüm 5.4'te ihtiyaç duyulduğu belirtilen 312MHz clock hızına erişmek mümkündür

Sentez sonrasında ilk doğrulama aşaması sayılacak Questa simülasyonu yapılmıştır. Daha önceden MATLAB'da üretilen ve FFT'si hesaplanan giriş test vektörü Testbench kodu olarak eklenmiştir. 256 nokta tamamen paralel FFT işlemi için toplamda 512 giriş (256 sanal, 256 gerçel) gerekmektedir. Benzer şekilde 512 adet çıkış bulunmak zorundadır. Bunların tamamının ekranda rapor olarak sunulmasına görsel açıdan çok zor olacağı için aşamalar ve karşılaştırmalar sınırlı sayıda giriş ve çıkış için verilmiştir.

MATLAB'ın kendi FFT komutu ile (kayan noktalı sayılar kullanılarak) belirli giriş değerleri için hesaplama yapılmış, ardından MATLAB'da oluşturduğumuz FPGA benzetim kütüphanesi ile (sabit noktalı sayılar) FFT hesabı yapılmış ve sonuçlar karşılaştırmıştır. Karşılaştırma sonuçları Tablo 5.2'de verilmiştir.

Tablo 5.2. MATLAB ve FPGA benzetim kütüphanesi karşılaştırması

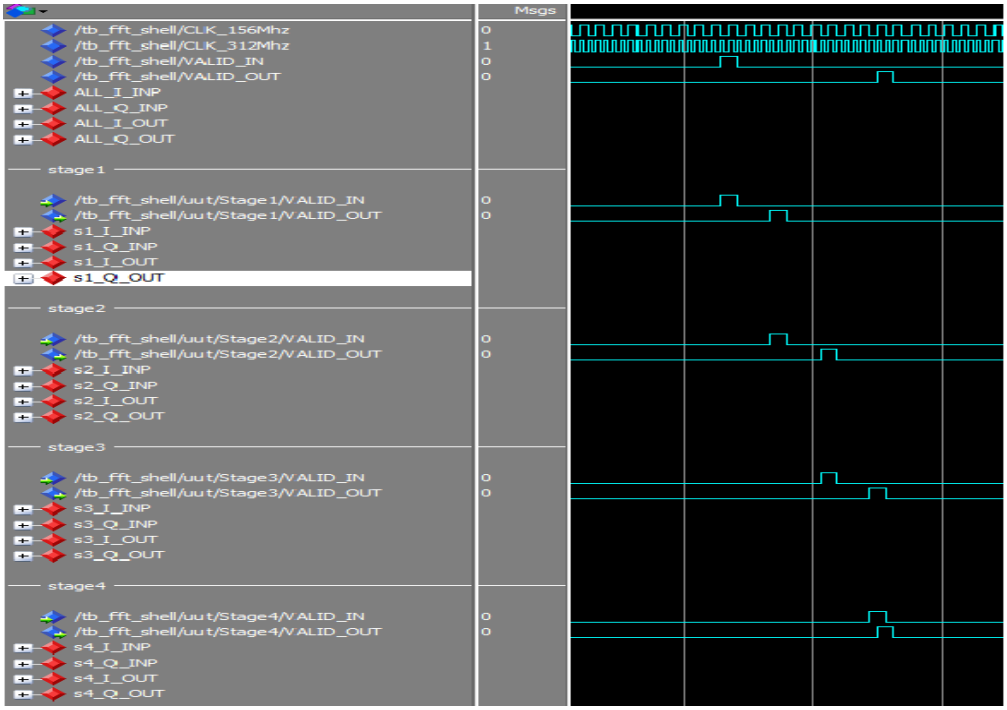
	Questa Sonuçları		Matlab Sonuçları	
İndeks	**Re**	**Im**	**Re**	**Im**
1	2082	1884	2082	1884
2	-39	2	-39,9843	1,122015
3	33	-77	30,81792	-75,7036
4	-56	-23	-57,2995	-22,5662
5	-53	-95	-53,0205	-94,1363
6	20	85	22,77702	85,59045
7	-114	-45	-114,459	-45,7134
8	-55	24	-53,938	25,03714
9	2	0	2,099795	1,2372205
10	-37	59	-36,7657	-58,0585

Tablo 5.2'de giriş test vektörü uygulanan Testbench kodu çalıştığında Questa simülasyon sonuçları kaydedilmiş ve MATLAB'ın FFT komutu sonuçları ile karşılaştırılmıştır. Sonuçlar oldukça yakındır. Hatalar, her basamakta normalizasyon sonrası yuvarlama işleminden dolayı oluşmuştur. 256 değeri aynı anda tablolarda göstermek zor olacağı için karşılaştırmalar sınırlı örnek üzerinden yapılmıştır.

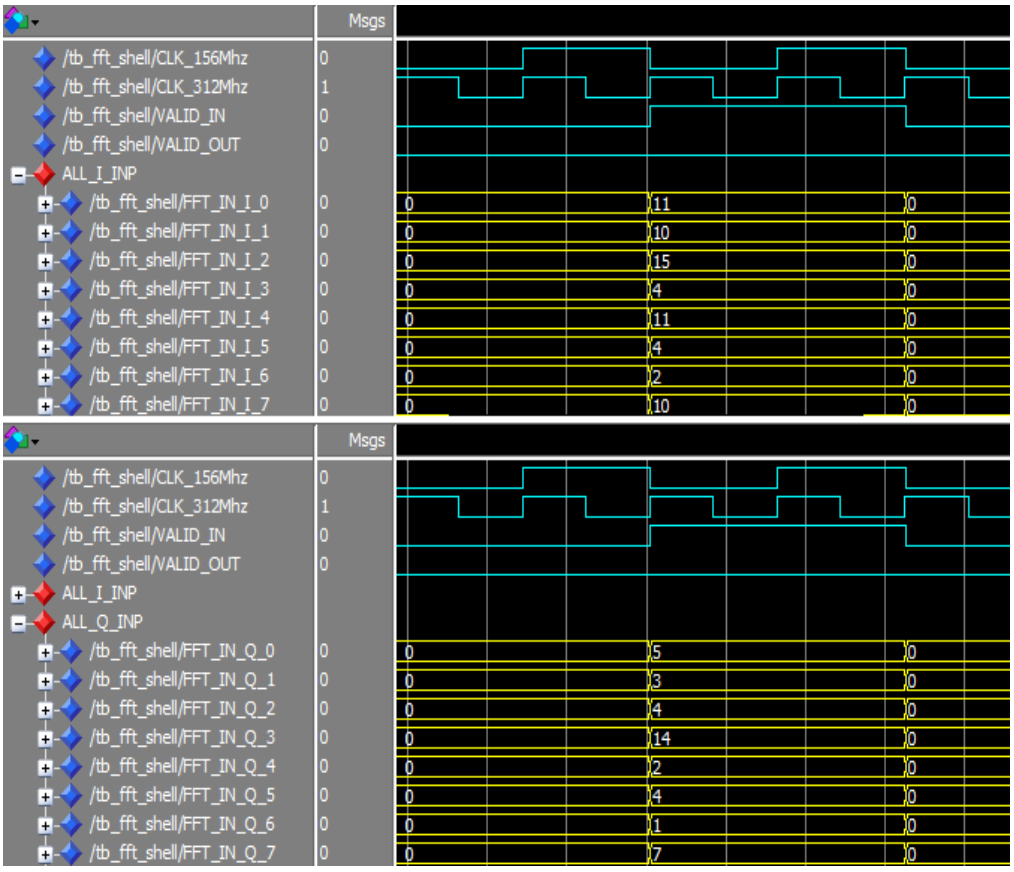
Tasarım sırasında olası bir hatayı kolay analiz etmek için her alt bloğa kendine özgü Testbench kodu yazılmıştır. Her basamak ve üstelik her basamaktaki toplam ve çarpım bloklarının sonucu sırasıyla kontrol edilmiştir.

Şekil 5.1'de görüldüğü gibi giriş test vektörü verildikten 9 saat işareti sonrasında çıkışlar değerleri alınabilmektedir (9 clock latency). Mevcut mimari kesintisiz şekilde işlem yapabilmektedir (pipeline).

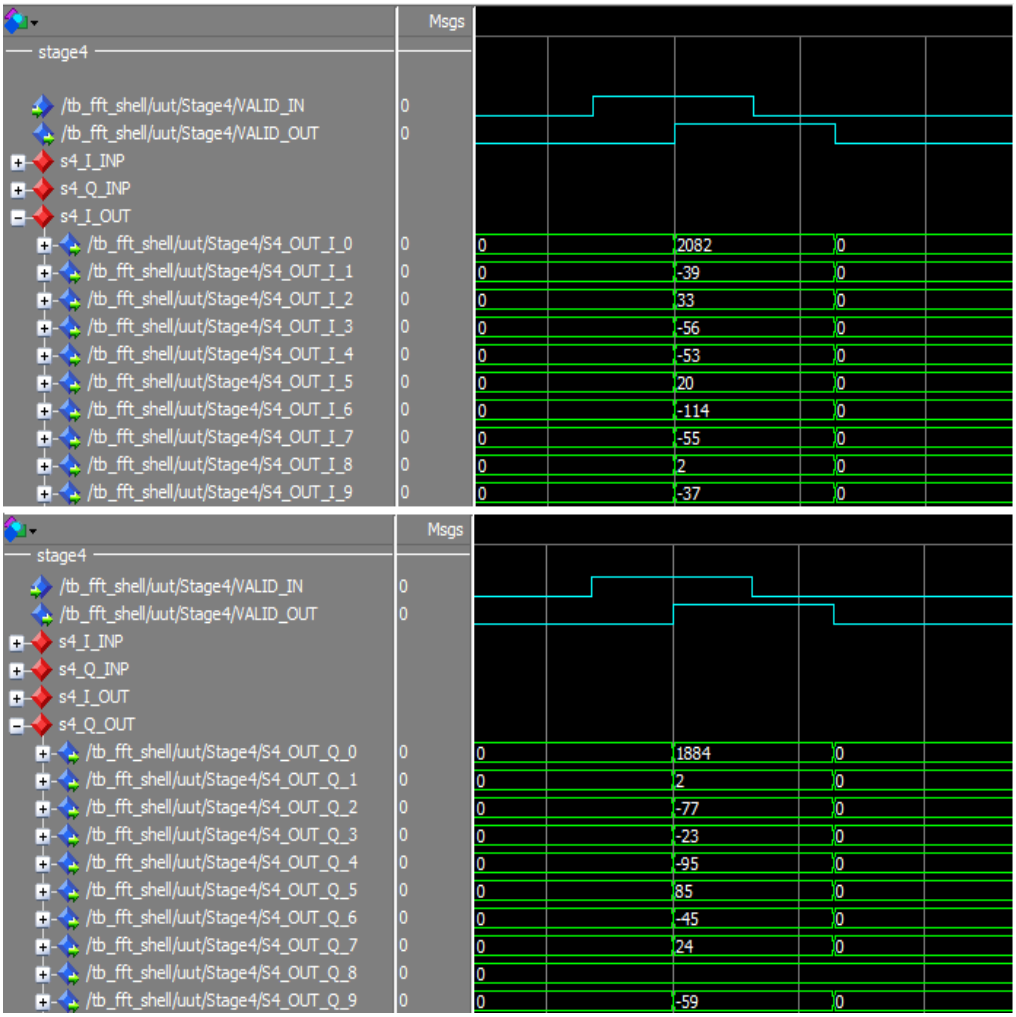
Şekil 5.2'deki girişler test vektörü olarak tasarlanan FFT koruna uygulandığında Şekil 5.3'teki çıkışlar elde edilmiştir. Sonuçlar Tablo 5.2'de verilen MATLAB FPGA benzetim kütüphanesinin sonuçları ile tamamen örtüşmektedir. Sonuçlar hem 256 nokta FFT korunun istenildiği gibi çalıştığını göstermektedir, hem de FPGA benzetim kütüphanesi sayesinde sentez aşamasına gelmeden MATLAB kullanıcı arayüzünde ile gerekli hesaplamaların ve karşılaştırmaların yapılabileceğini göstermektedir.



Şekil 5.1. Tüm basamakları gösteren Questa çıktısı



Şekil 5.2. Questa Gerçel ve Sanal giriş değerlerin uygulanışı

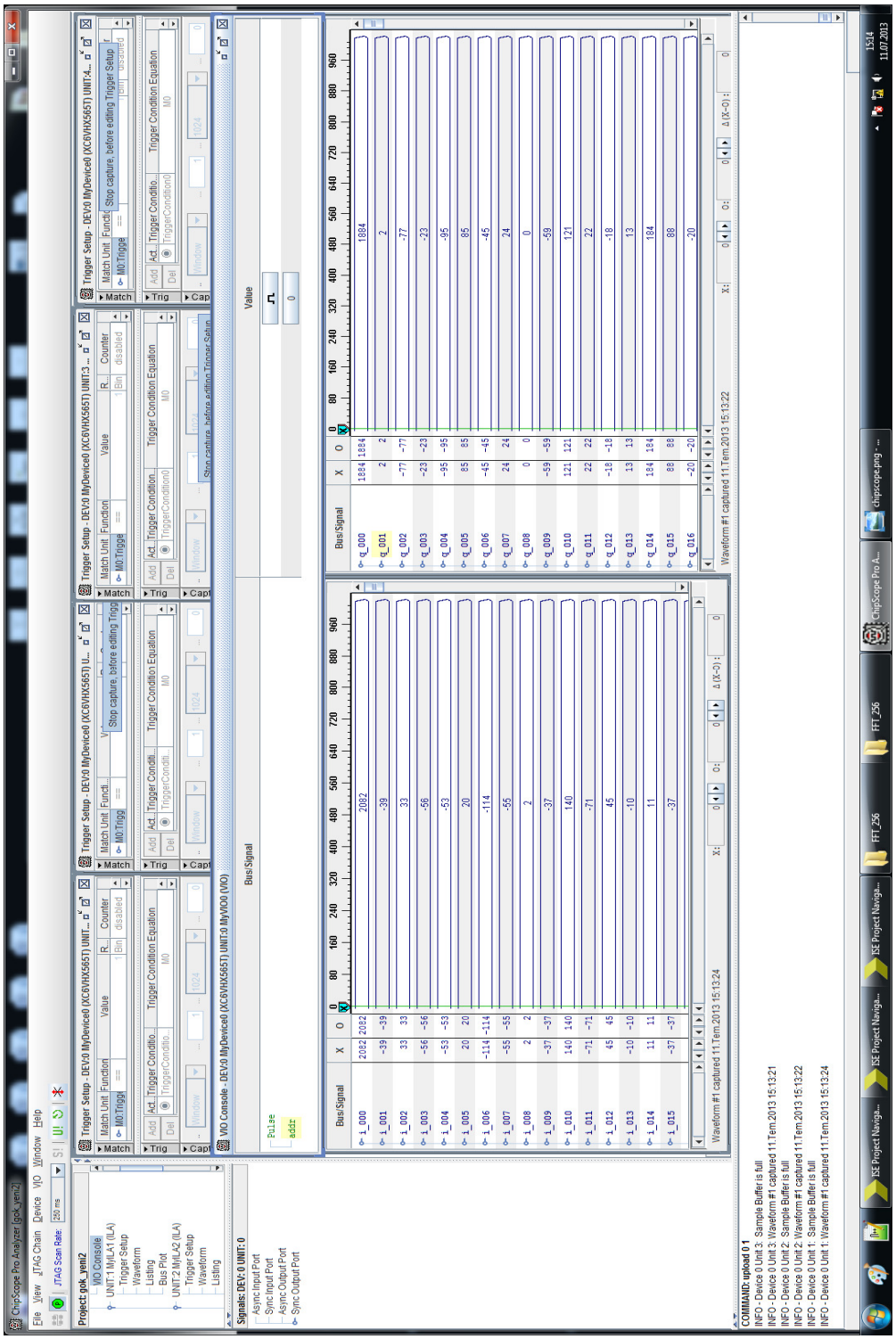


Şekil 5.3. Questa Gerçel ve Sanal çıkış değerleri

Xilinx ISE'de sentezlenen ve üstelik Questa ile simülasyonu yapılan ve doğrulanan kodlar beklenildiği gibi çalışmayabilir. Emin olmak için yapılması gereken; VHDL kodların FPGA üzerinde çalışma sırasında doğruluğunun ispat edilmesidir. Bu doğrulama süreci en zorlu süreçlerden biridir. En basit donanım tasarımlarına bile dışarıdan bir Logic-Analyser ile bağlanmak oldukça can sıkıcı bir süreçtir. Bu çalışma için konuşmak gerekirse 19 bitlik 512 adet çıkışı gözlemlemek için yapılacak Logic-Analyser bağlantıları günlerce vakit alabilir. Xilinx bu problemi çözebilmek adına ChipScope adını verdiği bir sistem geliştirmiştir [51].

ChipScope uygulaması, FPGA içerisine gömülerek Block RAM'lere FPGA'in kaynakları imkân verdiği sürece örnekleri kayıt etmektedir. Bu sayede dışarı

herhangi bir bağlantı yapılmadan sadece JTAG vasıtası ile Block RAM'ler üzerindeki değerler okunmakta ve çalışan FPGA üzerindeki değerler görünmektedir.



Şekil 5.4. ChipScope 256 nokta FFT çıkışları

Şekil 5.4'daki ChipScope sonuçlarının hem Questa simülasyonu hem de MATLAB FPGA benzetim kütüphane sonuçları ile birebir uyuştuğu görülmektedir.

Sonuç olarak bu çalışmada geliştirilen kodlar sayesinde düşünülen tasarım gerçekleştirilmiştir. Tasarım beklenildiği gibi çalışmıştır.

5.2. Öneriler

Bu çalışmanın başlangıcında uygun donanım ve çözümler araştırılmıştır. Araştırma sonucunda karar verilen donanımlar üzerinden bazı varsayımlar yapılarak tasarıma başlanmıştır. Tasarım sırasında Moore yasası göz önünde bulundurulursa projenin yapım aşamasında bile daha modern çözümlerin piyasaya çıkacağı aşikardır. Çalışmada mevcut zamanın en iyi FPGA'i kabul edilen Xilinx Virtex-6 565T kullanmaya başladıktan 3 ay sonra Xilinx firması Virtex-7 serisini piyasaya sunmuştur. Mevcut FPGA'de yarım milyon mantık kapısı bulunurken, 28nm teknolojisi ile üretilen Virtex-7 serisinde bu rakam 2.5 milyon kapıya ulaşmıştır. Benzer şekilde mevcut FPGA'de 6.6GHz'lik 48 GTX ve 11.2GHz 24 GTH hızlı alıcı-vericisi bulunurken Virtex-7 serisindeki 28GHz'lik alıcı-vericilerden (GTZ) 96 adet bulunmaktadır.

5 GHz zaman-serpiştirmeli ADC'ler (20Gsample/s) yerine Fujitsu firmasının ürettiği 56Gs/s lik ADC'ler tercih edilebilir [52].

Çalışmada VHDL kodları donanım bağımsız şekilde oluşturulmuştur. Herhangi bir IP CORE kullanılmadığından, MATLAB kullanıcı arayüzü sayesinde daha fazla giriş bitleri veya daha düşük yuvarlama hatalarını seçilebilir. Bu sayede Tablo 5.2'ye göre daha doğru sonuçlar almak mümkün olacaktır, ama karşılığında oluşturulan VHDL kodları FPGA'de daha fazla mantık kapısı kullanacaktır. Fakat yeni FPGA serileri için bu problem teşkil etmeyecektir.

Çarpım metodu için kullanılan kaydır topla metodu için oluşturulan kısımda değerler olduğu, gibi kullanılmıştır. Örneğin 1023 değeri ile çarpma yapılırken on adet kaydırma ve toplama işlemi yapılmıştır (MATLAB'da bu şekilde hesap yapmak kolay olduğu için). Kullanıcı 1023 ifadesini "1024-1" şeklinde yorumlayabilir ve işlemi 1 kaydır 1 topla işlemine dönüştürebilir. Fakat her değer için bu karşılaştırmayı yaparak uygun çözümü bulmak zaman alacaktır ve MATLAB

otomatik kod yazma sürecine bu işlemi öğretmek ciddi bir süreç olacaktır. Eğer bu işlem için yeterince zaman ayrılırsa tüm faz faktörleri için uygulandığında ciddi miktarda mantık kapısı kullanımı azalacaktır.

Sonuç olarak gelişen teknoloji ve yeni daha gelişmiş ürünler ile yapılan çalışma daha da iyileştirilebilir.

KAYNAKLAR

[1] Mussolin M., Digital Signal Processing Algorithms for High-Speed Coherent Transmission in Optical Fibers, Ms Thesis, Università degli studi di Padova Facoltà di Ingegneria, 2010.

[2] Haykin S., Signal processing: where physics and mathematics meet, *IEEE Sig. Proc. Mag.*, 2001, **18**, 6–7.

[3] http://www.megep.meb.gov.tr/mte_program_modul/moduller_pdf/Fiber%20Optik.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[4] Agrawal G. P., Nonlinear Effects in Optical Fibers, The Institute of Optics, http://www.imedea.uib.es/~salvador/coms_optiques/addicional/agrawal/NLOF.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[5] Laferriére J., Lietaert G., Taws R., Wolszczak S., *Reference Guide to Fiber Optic Testing*, 2nd ed., JDSU, France, 2011.

[6] http://lunainc.com/wp-content/uploads/2012/08/22pmdweb.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[7] Xu T., Digital Dispersion Equalization and Carrier Phase Estimation in 112-Gbit/s Coherent Optical Fiber Transmission System, Licentiate Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2011.

[8] Zhang X., Digital Signal Processing for Optical Coherent Communication Systems, Licentiate Thesis, Technical University of Denmark, Department of Photonics Engineering, Lyngby, Denmark , 2012.

[9] Sharifian S., Chromatic Dispersion Compensation By Signal Predistortion: Linear And Nonlinear Filtering, Ms Thesis, Chalmers University Of Technology, Göteborg, Sweden, 2010.

[10] Chauvel G., Dispersion in Optical Fibers, Anritsu Corporation, http://www.ausoptic.com/Alltopic/Download/Disp_in_Opt_Fibers_PMD_CD.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[11] Gajewski J., Przywecki M., Deliverable D.4.4: Events Proceedings, *Porta Optica study*, POS-15-005, 61, 2007.

[12] Ishihara K., Frequency-Domain Equalization for High-speed Fiber-Optic Transmission Systems, NTT Network Innovation Laboratories, http://www.mobile.ecei.tohoku.ac.jp/COE/workshop_2010_04/pdf/ishihara.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[13] http://www.fiberoptic.com/fiber_characterization/pdf/chromatic_dispersion.pdf , (Ziyaret tarihi: 05 Aralık 2013)

[14] Rival O., Morea A., Efficiency gain from elastic optical networks, Alcatel-Lucent Bell Labs, http://www.greentouch.org/uploads/documents/Morea-Rival_ACP2011VxOptNP110018.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[15] http://optiwave.com/?wpdmact=process&did=MTI0LmhvdGxpbms=, (Ziyaret tarihi: 05 Aralık 2013)

[16] Škoda P., Radil J., Vojtěch J., Hůla M., 100G Coherent System Interoperability, *TERENA Networking Conference*, Reykjavík, 21-24 Mayıs 2012.

[17] http://www.comm.utoronto.ca/~dkundur/course_info/real-time-DSP/implementation/Kundur_Lab3_FFT_Convolution_6437.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[18] http://tr.wikipedia.org/wiki/Jean-Baptiste_Joseph_Fourier, (Ziyaret tarihi: 05 Aralık 2013)

[19] http://www.yalescientific.org/2010/12/fourier-transform-natures-way-of-analyzing-data/, (Ziyaret tarihi: 05 Aralık 2013)

[20] http://en.wikipedia.org/wiki/Fourier_transform, (Ziyaret tarihi: 05 Aralık 2013)

[21] Ayhan T., FFT Algoritmalarının FPGA Üzerinde Gerçeklenmesi, Lisans Tezi, İstanbul Teknik Üniversitesi Elektrik – Elektronik Fakültesi, İstanbul, 2008.

[22] Burrus C.S., Fast Fourier Transform, CONNEXIONS, http://www.freeinfosociety.com/media/pdf/2804.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[23] Pavan Kumar K. M., Jain P., Ravi Kiran S., Rohith N., Ramamani K., FFT Algorithms: A Survey, *The International Journal Of Engineering And Science (IJES)*, 2013, **2**, 22-26.

[24] McKeown Mark, FFT Implementation on the TMS320VC5505,TMS320C5505, and TMS320C5515 DSPs, *Texas Instruments*, SPRABB6B, 2010.

[25] http://course.ee.ust.hk/elec214/13spg-Palomar/notes/Elec3100_2012_Ch3.3_FFT_Final.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[26] Hudrouss A. A., FFT Decimation In Time, Islamic University Gaza, http://site.iugaza.edu.ps/aliafana/files/2010/03/decimation_in_time.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[27] http://twins.ee.nctu.edu.tw/courses/dsp_07/Chap9-FFT.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[28] Hudrouss A. A., FFT Decimation In Frequency, Islamic University Gaza, http://site.iugaza.edu.ps/aliafana/files/2010/03/decimation_in_frequency.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[29] http://www.ece.ucdavis.edu/~bbaas/281/slides/Handout-.fft2.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[30] http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html, (Ziyaret tarihi: 05 Aralık 2013)

[31] http://www.ece.uvic.ca/~elec499/2004a/group05/html/background.html, (Ziyaret tarihi: 05 Aralık 2013)

[32] Punskaya E., Fast Fourier Transform (FFT), Cambridge, http://www-sigproc.eng.cam.ac.uk/~op205/3F3_3_Fast_%20Fourier_Transform.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[33] Kenneth E. Barner, The Fast Fourier Transform; Radix–2 and Radix–4 Algorithms, Dept. of Elect. and Comp. Engineering University of Delaware, http://www.eecis.udel.edu/~barner/courses/eleg305/lecture_notes/L17.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[34] http://tr.wikipedia.org/wiki/G%C3%B6m%C3%BCl%C3%BC_Sistem_%28_Sabitlenmi%C5%9F_Sistem_%29 , (Ziyaret tarihi: 05 Aralık 2013)

[35] http://mouloudrahmani.com/electrical/Digital/FPGAFundamentals.html, (Ziyaret tarihi: 05 Aralık 2013)

[36] Başak S., FPGA ile Gömülü Sistem Tasarımına Giriş, Çizgi Tagem, http://www.cizgi-tagem.org/vfiles/lms_sayfa/3058/fpgagiris.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[37] http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[38] Gallagher S., Mapping DSP Algorithms Into FPGAs, Xilinx Inc. http://www.ieee.li/pdf/viewgraphs/mapping_dsp_algorithms_into_fpgas.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[39] http://www.xilinx.com/support/documentation/user_guides/ug366.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[40] Dinechin F., Tanguy J. M., A 128-Tap Complex FIR Filter Processing 20 Giga -Samples/s in a Single FPGA, *44th Asilomar Conference on Signals*, Systems & Computers, systems & Computers, California, 7-10 Kasım 2010.

[41] http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf , (Ziyaret tarihi: 05 Aralık 2013)

[42] http://www.commsonic.com/downloads/sd0001.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[43] http://www.xilinx.com/support/documentation/ip_documentation/dsp48_macro_ds754.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[44] C. Wu, Implementing the Radix-4 Decimation in Frequency (DIF) Fast Fourier Transform (FFT) Algorithm Using a TMS320C80 D0 DSP, *Texas Instruments*, SPRA152, 1998.

[45] Ferizi A., Hoeher B., Jung M., Fischer G. & Koelpin A, Design and Implementation of a Fixed-Point Radix-4 FFT Optimized for Local Positioning in Wireless Sensor Networks, *9th International MultiConference on Systems, Signals and Devices*, Chemnitz, 20-23 Mart 2012.

[46] Hsiao C. F., Chen Y., and Lee C. Y., A Generalized Mixed-radix Algorithm for Memory-based FFT Processors, *IEEE Trans. Circuits and Systems II, Express Briefs*, 2010, **57**, 26-30.

[47] Pasca B., FPGA Multipliers, Université de Lyon, France, http://webdali.univ-perp.fr/raim11/slides/slide-bpasca.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[48] Lefèvre V., Multiplication by an Integer Constant, *Instıtut Natıonal De Recherche En Informatıque Et En Automatıque*, INRIA/RR-4192, 2001.

[49] http://users.utcluj.ro/~baruch/book_ssce/SSCE-Shift-Mult.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[50] Ramezani S., Bilgisayar Aritmetik Algoritmalarının Fpga Üzerinde Gerçeklenmesi, Lisans Tezi, İstanbul Teknik Üniversitesi Elektrik – Elektronik Fakültesi, İstanbul, 2008.

[51] http://www.xilinx.com/support/documentation/ip_documentation/chipscope_ila.pdf, (Ziyaret tarihi: 05 Aralık 2013)

[52] Dedic I., 56Gs/s ADC: Enabling 100GbE, Fujitsu, http://www.fujitsu.com/downloads/MICRO/fme/dataconverters/OFC-2010-56Gss-ADC-Enabling-100GbE.pdf, (Ziyaret tarihi: 05 Aralık 2013)

KİŞİSEL YAYIN ve ESERLER

1- Kale A., Sarı F., **Polat G.**, Radar Dönüş Ekolarının Simülasyon Ortamında Üretimi ve Radar Alıcı Kütüphanesi, *5. Ulusal Savunma Uygulamaları Modelleme ve Simülasyon Konferansı (USMOS' 2013)*, Ankara, 11-12 Haziran 2011.

2- **Polat G.**, Öztürk S., Yakut M., Design and Implementation of 256 Point Radix-4 100 Gbit/s FFT Algorithm into FPGA for High Speed Applications, *IEEE Computer Society* (hakem değerlendirmesinde).

ÖZGEÇMİŞ

1987 yılında İstanbul'da doğdu. İlk ve orta öğretimini Eczacıbaşı İlköğretim Okulunda tamamladı. Lise öğrenimini Tuzla Behiye Dr. Nevhiz Işıl Anadolu Lisesinde tamamladı. Kocaeli Üniversitesi Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği Bölümü'nden 2010 yılında Elektronik ve Haberleşme Mühendisi olarak mezun oldu. 2010 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda Yüksek Lisans öğrenimine başladı. Halen Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda öğrenci olup mezun olma aşamasındadır.