

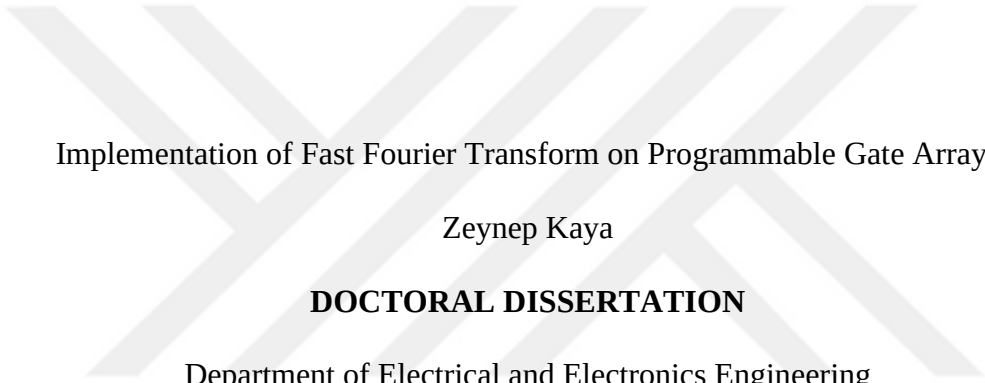
Hızlı Fourier Dönüşümünün Programlanabilir Kapı Dizilerinde Uygulanması

Zeynep Kaya

DOKTORA TEZİ

Elektrik Elektronik Mühendisliği Anabilim Dalı

Nisan 2021



Implementation of Fast Fourier Transform on Programmable Gate Arrays

Zeynep Kaya

DOCTORAL DISSERTATION

Department of Electrical and Electronics Engineering

April 2021

Hızlı Fourier Dönüşümünün Programlanabilir Kapı Dizilerinde Uygulanması

Zeynep Kaya

Eskişehir Osmangazi Üniversitesi

Fen Bilimleri Enstitüsü

Lisansüstü Yönetmeliği Uyarınca

Elektrik Elektronik Mühendisliği Anabilim Dalı

Telekomünikasyon-Sinyal İşleme Bilim Dalında

DOKTORA TEZİ

Olarak Hazırlanmıştır

Danışman: Doç. Dr. Erol Seke

Nisan 2021

ETİK BEYAN

Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kılavuzuna göre, Doç. Dr. Erol SEKE. danışmanlığında hazırlamış olduğum “Hızlı Fourier Dönüşümünün Programlanabilir Kapı Dizilerinde Uygulanması” başlıklı DOKTORA tezimin özgün bir çalışma olduğunu; tez çalışmamın tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; tezimde verdiğim bilgileri, verileri akademik ve bilimsel etik ilke ve kurallara uygun olarak elde ettiğimi; tez çalışmamda yararlandığım eserlerin tümüne atıf yaptığımı ve kaynak gösterdiğimi ve bilgi, belge ve sonuçları bilimsel etik ilke ve kurallara göre sunduğumu beyan ederim. 19/04/2021

Zeynep KAYA

İmza

ÖZET

Son zamanlarda mobil haberleşme teknolojisinin gelişmesiyle aynı bantta birden fazla kullanıcı ve hızlı veri gönderimi ihtiyaçlarından dolayı Dikgen Frekans Bölmeli Çoğullama (OFDM) tekniği kullanılmaya başlanmıştır. OFDM teknolojisi verinin alt kanallara bölünmesi ve her bir alt kanal verisinin birbirine dik taşıyıcılarla tayfa örtüşerek iletişime olanak sağlamasıyla, iletim bandını tek taşıyıcılı sistemlere göre daha etkin bir şekilde kullanır. Ancak, bu tekniğin gerçek zamanlı uygulamalarda başarılı olabilmesi için hızlı ve yüksek verimli işlemciler gereksinim duyulmaktadır. Kablosuz haberleşmede modülasyon tekniği için en etkili yöntemin Hızlı Fourier Dönüşümü (FFT) tabanlı OFDM yaklaşımıdır. OFDM uygulamalarında FFT işlemleri yoğun işlem yükü gerektirmektedir. Bu yüzden FFT işlemlerinin en hızlı şekilde, karmaşıklığı ve kullanılan donanım miktarını azaltacak algoritmalar ile gerçekleştirilmesi önem arz etmektedir. Son zamanlarda Alanda Programlanabilir Kapı Dizileri (FPGA) teknolojisi oldukça gelişmiş olup, karmaşık ve hızlı işlem yapabilme yeteneğinin yanı sıra yeniden programlanabilirlik özelliğiyle gerçek zamanlı FFT uygulamalarında önemli bir seçenek olarak öne çıkmaktadır. FPGA platformlarında FFT algoritmaları uygulanırken hedef, paralel işlem yapabilme özelliğiyle birlikte işlem hızını en üst düzeye çıkarmak ve kullanılan eleman sayısını en aza indirmektir. FPGA'lerin hızlı ve paralel işlem yapabilme özelliği ile geniş bant sayısal iletişim sistemlerinde yer alan gerçek zamanlı FFT uygulamaları başarı ile uygulanmaktadır.

Doktora tez çalışmasında radiks-2 DIF algoritması kullanılarak hafıza-tabanlı FFT çalışmaları yapılmıştır. FPGA içinde bulunan çift-portlu hafıza yapıları kullanılmış ve radiks-2 DIF algoritması ile yeni hafıza adresleme algoritmaları geliştirilmiştir. Hafıza kullanımını ve işlem süresini en aza indirmeyi hedefleyerek bulunan adresleme algoritmalarının VHDL ile uygulaması yapılmıştır. N boyutlu hafıza kullanılarak tasarlanan algoritmalar ile tüm FFT işlemleri tek blok hafıza yapılarında $(\frac{N}{2} + 1)\log_2 N$ ve $\frac{N}{2}(1 + \log_2 N)$, çoklu blok yaklaşımlarda $(\frac{N}{2})\log_2 N$ (ideal çalışma süresi), ve paralellik yaklaşımı ile $(\frac{N}{2})\log_2 N / 2$ çalışma süresinde tamamlanmıştır.

Anahtar Kelimeler: Hafıza-Tabanlı FFT, OFDM, Radiks-2 DIF, Çift-Port, FPGA.

SUMMARY

Recently, with the development of mobile communication technology, usage of Orthogonal Frequency Division Multiplexing (OFDM) technique is increased due to the requirements of efficient bandwidth usage with multiple users within the same band and increase in bandwidth demand. OFDM technology allows bandwidth savings by dividing data into sub-channels and overlapping their carriers by making them orthogonal to each other. However, in order for this modulation technique to be successful in real-time applications, it needs to use fast and efficient processors. The Fast Fourier Transform (FFT) based OFDM application is the most effective method for modulation technique in wireless communication. It is the most computation-intensive part in OFDM applications. Therefore, it is important that FFT operations are performed in the fastest way with algorithms that will reduce the complexity and the amount of hardware used. Lately Programmable Gate Arrays (FPGA) technology, having considerable advancements and reprogrammability feature, plays an important role in real-time FFT applications with its ability to perform complex and fast operations. When implementing FFT algorithms in FPGA platforms, the goal is to maximize the processing speed and minimize the number of elements used with the ability to perform parallel processing. With the fast and parallel processing feature of FPGAs, real-time FFT applications in broadband digital communication systems are successfully applied.

In this doctoral dissertation study, the memory-based FFT studies were performed using radix-2 DIF algorithm. Dual-port memory structures in FPGA were used and new memory addressing algorithms were developed with radix-2 DIF algorithm. Addressing algorithms have been applied with VHDL, aiming to minimize memory usage and processing time. With algorithms designed using N -dimensional memory, all FFT operations were completed in $(\frac{N}{2} + 1)\log_2 N$ and $\frac{N}{2}(1 + \log_2 N)$ in single block memory structures, $(\frac{N}{2})\log_2 N$ (ideal processing time) and $(\frac{N}{2})\log_2 N / 2$ with a parallelism approach in multi-block approaches.

Keywords: Memory-Based FFT, OFDM, Radix-2 DIF, Dual-Port, FPGA.

İÇİNDEKİLER

Sayfa

ÖZET.....	vi
SUMMARY	vii
TEŞEKKÜR.....	viii
İÇİNDEKİLER	ix
ŞEKİLLER DİZİNİ	xii
ÇİZELGELER DİZİNİ	xv
SİMGELER VE KISALTMALAR DİZİNİ	xvi
1. GİRİŞ VE AMAÇ	1
2. LİTERATÜR ARAŞTIRMASI.....	7
3. GENEL BİLGİLER.....	15
3.1. Dikgen Frekans Bölmeli Çoğullama (OFDM)	15
3.1.1. Dikgen Frekans Bölmeli Çoğullama Avantaj ve Dezavantajları	18
3.1.2. Dikgen Frekans Bölmeli Çoğullama Uygulama ve Standartları	20
3.2. Ayırık Fourier Dönüşümü ve Hızlı Fourier Dönüşümü Algoritmaları	21
3.2.1. Cooley-Tukey Algoritmaları.....	23
3.2.1.1. Radiks-2 FFT Algoritması.....	23
3.2.1.2. Radiks-4 FFT Algoritması.....	25
3.2.1.3. Bölünmüş-Radiks (Split-Radix) FFT Algoritması	26
3.2.2. Diğer Hızlı Fourier Dönüşümü Algoritmaları.....	26
3.2.2.1. Winograd Algoritması.....	27
3.2.2.2. Prime Factor Algoritması	27
3.2.2.3. Sabit Geometri Yapısı	28
3.3. Hızlı Fourier Dönüşümü İşlemci Yapılarının Sınıflandırılması	29
3.3.1. Ardışık Düzen FFT Yapıları	29
3.3.1.1. Paralel-Giriş Paralel-Çıkış Ardışık Düzen FFT Yapıları	30
3.3.1.2. Seri-Giriş Seri-Çıkış Ardışık Düzen FFT Yapıları.....	31
3.3.1.3. Seri-Giriş Paralel-Çıkış Ardışık Düzen FFT Yapıları.....	32
3.3.2. Hafıza-tabanlı FFT Yapıları.....	35
3.3.2.1. Tek Hafıza Bloğu-Tabanlı FFT İşlemci Metodu.....	36
3.3.2.2. Çift Hafıza-Tabanlı FFT İşlemci Metodu.....	36
3.3.2.3. Önbellek-Tabanlı FFT İşlemci Metodu.....	37

İÇİNDEKİLER (devam)

Sayfa

3.3.2.4. Radiks-2 Hafıza-Tabanlı FFT Yapıları	37
3.3.2.5. Radiks-4 Hafıza-Tabanlı FFT Yapıları	38
4. ÖNERİLEN YÖNTEMLER	39
4.1. Radiks-2 Hızlı Fourier Dönüşümü Algoritması.....	39
4.2. Çift Portlu Hafıza Yapıları.....	41
4.3. Sabit Çarpanlar ROM'u	43
4.4. Tek Blok Çift-Port Hafıza-Tabanlı Radiks-2 Hızlı Fourier Dönüşümü Uygulamaları.....	44
4.4.1. 16-Noktalı Radiks-2 Hızlı Fourier Dönüşümü Hafıza Adresleme Algoritması.....	47
4.4.2. N-Noktalı Radiks-2 Hızlı Fourier Dönüşümü Hafıza Adresleme Algoritması	51
4.4.3. Radiks-2 Hızlı Fourier Dönüşümü ile N-Noktalı Yeni Hafıza Adresleme Algoritması	56
4.5. Çoklu Blok Çift-Port Hafıza-Tabanlı Radiks-2 Hızlı Fourier Dönüşümü Uygulamaları .	62
4.5.1. Gerçek Çift Portlu Hafıza Yapıları	63
4.5.2. Radiks-2 Çoklu Gerçek Çift Portlu Hafıza-Tabanlı Hızlı Fourier Dönüşümü Adresleme Algoritması	64
4.5.3. Paralel Giriş/Çıkış İşlemleri Sıralı Yerinde Kullanılabilir Radiks-2 Hızlı Fourier Dönüşümü Adresleme Metodu	70
5. BULGULAR VE TARTIŞMA	85
5.1. Tek Blok Çift-Port Hafıza Çalışma Bulguları	85
5.1.1. 16-Noktalı Radiks-2 Hızlı Fourier Dönüşümü Hafıza Adresleme Algoritması Bulguları.....	85
5.1.2. N-Noktalı Radiks-2 Hızlı Fourier Dönüşümü Hafıza Adresleme Algoritması Bulguları.....	86
5.1.3. Radiks-2 Hızlı Fourier Dönüşümü ile N-Noktalı Yeni Hafıza Adresleme Algoritması Bulguları.....	88
5.2. Çoklu Blok Çift-Port Hafıza-Tabanlı Radiks-2 Hızlı Fourier Dönüşümü Uygulamaları Bulguları	90
5.2.1. Radiks-2 Çoklu Gerçek Çift Portlu Hafıza-Tabanlı Hızlı Fourier Dönüşümü Adresleme Algoritması Bulguları	91
5.2.2. Paralel Giriş/Çıkış İşlemleri Sıralı Yerinde Kullanılabilir Radiks-2 Hızlı Fourier Dönüşümü Adresleme Metodu Bulguları	93

İÇİNDEKİLER (devam)**Sayfa**

6. SONUÇ VE ÖNERİLER.....	96
KAYNAKLAR DİZİNİ.....	98



ŞEKİLLER DİZİNİ

<u>Sekil</u>	<u>Sayfa</u>
3.1. OFDM'deki birbirine dik alt taşıyıcılar.....	15
3.2. Dikgen Frekans Bölmeli Çoğullama verici kısmı.....	17
3.3. Her kanalında 16-QAM kullanan 4 alt banta sahip OFDM üretimi	18
3.4. Radiks-2 kelebek yapısıyla 8-noktalı DIT FFT.....	24
3.5. 2-noktalı FFT hesaplayan kelebek yapısı.....	24
3.6. Radiks-4 DIF FFT algoritmasının temel kelebek yapısı.....	25
3.7. Bölünmüş-radiks DIF kelebek yapısı.....	26
3.8. Sabit geometri DIF yapısı.....	28
3.9. R2MDC yapısı.....	30
3.10. R4MDC yapısı.....	30
3.11. R2SDF yapısı.....	31
3.12. R4SDC yapısı.....	32
3.13. R4SDF yapısı.....	32
3.14. Goertzel algoritması yapısı.....	33
3.15. Seri veri girişinde kelebek yapısının tekrarlanan hesaplama yapısı.....	34
3.16. $N=8$ veri için seri-giriş paralel-çıkış FFT pencere metodu.....	34
3.17. Hafıza-tabanlı FFT yapısı.....	35
3.18. Tek hafıza-tabanlı FFT mimari yapısı.....	36
3.19. Çift hafıza-tabanlı FFT mimarisi blok diyagramı.....	36
3.20. Önbellek-tabanlı FFT mimarisi blok diyagramı.....	37
4.1. DIF FFT bölme işlemi gösterimi.....	40
4.2. $N=16$ -noktalı DIF FFT yapısı.....	41
4.3. Tek bir işlem ögesi (PE).....	41
4.4. Çift portlu hafıza blok diyagramı.....	42
4.5. a) $N=16$ için twiddle ROM içerikleri b) karmaşık düzlemdeki çarpanlar.....	44
4.6. Kelebek devre elemanı veri aktarımı ve adresleme kontrolü gösterimi.....	45
4.7. İlk aşamadaki veri işlemleri.....	46

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
4.8. $N=16$ için üretilen adres satırı gösterimi.....	48
4.9. a) Satır adreslerine yerleştirilen p biti ile doğruluk tablosu b) Komütatör model.....	48
4.10. Gerçek adres üretici blok diyagramı.....	49
4.11. İlave saat darbesiyle aşama değişimi yapan VHDL kod örneği.....	50
4.12. $N=16$ için çoklayıcı kullanılarak VHDL ile gerçekleştirilen kombinatoriyal adres değeri üretimi.....	50
4.13. $N=16$ için ta adreslerini üreten VHDL kodu.....	51
4.14. $N=16$ için üretilen sabit çarpanlar ROM adresleri ve üretilen veri hafızası.....	54
4.15. $N=16$ için ta adreslerini a sayacından üreten VHDL kodu.....	55
4.16. İlk aşamadaki veri işlemleri.....	57
4.17. $N=16$ için üretilen sabit çarpanlar ROM adresleri ve üretilen veri hafıza adresleri.....	59
4.18. Tek bir PE işlem ögesi için komütatör devresi ile düz okuma/ yazma gösterimi.....	60
4.19. Tek bir PE işlem ögesi için istisnai durum (ilk $2^s - 2$ işlem) komütatör devresi ile ters okuma ve ters yazma işlemi.....	61
4.20. $N=16$ noktalı FFT için tüm okunan/yazılan hafıza adresleri ve sabit çarpan ROM adresleri gösterimi.....	61
4.21. Gerçek çift portlu hafıza okuma/yazma blok diyagramı.....	63
4.22. FFT okuma/ yazma ve kelebek işlemleri kontrol devresi.....	65
4.23. $N=16$ için üretilen sabit çarpanlar ROM ve üretilen veri hafıza adresleri.....	67
4.24. Tek bir PE işlem ögesi için komütatör devresi.....	68
4.25. $N=16$ noktalı FFT için tüm okunan/yazılan hafıza adresleri ve sabit çarpan değerleri gösterimi.....	69
4.26. $N=16$ için ta adreslerinin $c1$ sayacından VHDL ile üretilmesi.....	69

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
4.27. a) Hafıza-Tabanlı FFT genel yapısı (sol) – Modifiyeli paralel g/ç işlemlili Hafıza-Tabanlı FFT yapısı (sağ) b) Yükleme-işlem-boşaltma kavramsal zamanlama diyagramı c) Son aşama işlemleri ve veri çıkışı (boşaltma) paralel olma zamanlama diyagramı d) Yükleme ve ilk FFT işlemleri paralel olma zamanlama diyagramı e) Yükleme ve boşaltma ile son aşama FFT işlemleri paralel olma zamanlama diyagramı.....	71
4.28. $N=32$ nokta FFT sıralı düzen çıkış ve hafıza adresleme yapısı.....	73
4.29. Adres üretici çoklayıcı bloğu.....	79
4.30. Adres yönlendirme çoklayıcı bloğu.....	81
4.31. Blok seçim biti kullanılarak veri çoklayıcılar aracılığıyla PE ve hafıza blokları arasındaki bağlantı gösterimi.....	81
4.32. Önerilen çalışma için genel blok diyagramı.....	82
5.1. $N=2048$ VHDL modül şeması	93

ÇİZELGELER DİZİNİ

<u>Cizelge</u>	<u>Sayfa</u>
3.1. Yayın standartları.....	20
3.2. Kablosuz Yerel Alan Ağı (WLAN) standardı.....	21
3.3. Kablosuz Yerel Döngü (WLL) standartları.....	21
4.1. $N = 128$ için sayaç bitlerinden okuma adreslerinin oluşturulması.....	78
4.2. $N = 128$ için sayaç bitlerinden yazma adreslerinin oluşturulması.....	79
4.3. $N = 128$ için sabit çarpanlar (tw) üretimi.....	84
5.1. Çeşitli çalışmaların saat darbe sayısı karşılaştırması.....	87
5.2. Hafıza boyutu ve bankası karşılaştırması.....	87
5.3. Optimum değerler ve yapılan çalışmaların saat darbe sayısı ve hafıza miktarı karşılaştırması.....	89
5.4. Çeşitli çalışmaların saat darbe sayısı ve hafıza karşılaştırması.....	89
5.5. Optimum değerler ve yapılan çalışmanın saat darbe sayısı karşılaştırması.....	91
5.6. Çeşitli çalışmaların toplam çalışma süresi, paralel işlem, hafıza ve hafıza bankası karşılaştırması.....	92
5.7. Çeşitli hafıza-tabanlı FFT metotları karşılaştırması.....	95

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler

dB

clk

en

f

GHz

Mbit

MHz

mW

we

Açıklama

Desibel

Clock

Enable

Frekans

Giga Hertz

Mega Bit

Mega Hertz

Mili Watt

Write Enable

Kısaltmalar

ADSL

AMPS

ASIC

CM

CORDIC

CP-OFDM

Açıklama

Sayısal Abone Hattı

Gelişmiş Mobil Telefon Hizmeti

Uygulamaya Özel Tümlşik Devre

Sabit Çarpan

Koordinat Döndüren Sayısal Bilgisayar

Çevrimsel-Önek Dikgen Frekans Bölmeli

Çoğullama

DAB

Dijital Ses Yayını

DDS

Doğrudan Sayısal Sentezleme

DFT

Ayrık Fourier Dönüşümü

DFT-S

Ayrık Fourier Dönüşümü-Yayılm

DVB

Dijital Video Yayını

DVB-T

Karasal Dijital Video Yayını

DSL

Sayısal Abone Hattı

DSP

Sayısal Sinyal İşlemcisi

EDGE

Geliştirilmiş Veri Hızları

eMBB

Gelişmiş Mobil Geniş Bant

FDM

Frekans Bölmeli Çoğullama

SİMGELER VE KISALTMALAR DİZİNİ (devam)

Kısaltmalar

FDMA

FFT

FM

FPGA

GPRS

GSM

HDTV

HSDPA

ICI

IDFT

IFFT

IoT

ISI

LTE

M2M

MDC

MDF

MIMO

MMS

MUX

NMT

NR

OFDM

OFDMA

PAPR

PE

PFA

PTT

QAM

Açıklama

Frekans Bölmeli Çoklu Erişim

Hızlı Fourier Dönüşümü

Frekans Modülasyonu

Alanda Programlanabilir Kapı Dizileri

Genel Paket Radyo Servisi

Mobil Haberleşme için Küresel Sistem

Yüksek Çözünürlüklü Televizyon

Yüksek Hızlı İniş Paketi Erişimi

Taşıyıcılar Arası Etkileşim

Ters Ayrık Fourier Dönüşümü

Ters Hızlı Fourier Dönüşümü

Nesnelerin İnterneti

Semboller Arası Etkileşim

Uzun Vadeli Evrim

Makineler Arası Bağlantı

Çoklu-Yol Gecikmeli Komütatör

Çoklu-Yol Gecikmeli Geribildirim

Çoklu Giriş Çoklu Çıkış

Multimedya Mesaj Servisi

Çoklayıcı

Nordic Mobil Telefon

Yeni Radyo

Dikgen Frekans Bölmeli Çoğullama

Dikgen Frekans Bölmeli Çoklu Erişim

Yüksek Tepe-Ortalama Güç Oranı

İşlem Ögesi

Öncül Faktör Algoritması

Bas ve Konuş

Dördül Genlik Modülasyonu

SİMGELER VE KISALTMALAR DİZİNİ (devam)

Kısaltmalar

QoS

QPSK

RAM

RF

ROM

SC-FDM

SDC

SDF

SDR

TDMA

UMTS

UWB

VHDL

W-CDMA

WiMAX

WLAN

WPANs

Açıklama

IP Ağlarında Yeterli Hizmet Kalitesi

Dördöl Faz Kaydırmalı Anahtarlama

Rastgele Erişimli Bellek

Radyo Frekansı

Salt Okunur Bellek

Tek Taşıyıcılı- Frekans Bölmeli Çoğullama

Tekli-Yol Gecikmeli Komütatör

Tekli-Yol Gecikmeli Geribildirim

Yazılım Tabanlı Radyo

Zaman Bölmeli Çoklu Erişim

Evrensel Mobil Telekomünikasyon Sistemi

Ultra Geniş Bant

Yüksel Hızlı Donanım Tanımlama Dili

Geniş Bantlı Kod Bölmeli Çoklu Erişim

Mikrodalga Erişimi için Dünya Geneline

Uyumluluk

Kablosuz Yerel Alan Ağı

Kablosuz Kişisel Alan Ağı

1. GİRİŞ VE AMAÇ

Kablosuz haberleşme teknolojisi 1970’li yıllarda hücresel iletişim teknolojisinin geliştirilmesi ve Analog FM teknolojisinin kullanımıyla birinci nesil (1G) teknoloji kullanımı ve gelişimi başlamıştır. Bu teknoloji ile analog haberleşme standartları 1980’li yılların sonuna doğru kullanılmış ve 1990’lı yıllarda yerini sayısal haberleşme tekniği olan ikinci nesil (2G) teknolojiye bırakmıştır. Birinci nesil mobil kablosuz iletişim ağı yalnızca analog olarak sesli aramalar için kullanılmaktayken ikinci nesil sayısal iletişim tekniği metin mesajlaşmasını desteklemektedir. Artan kapasite, daha yüksek veri iletim hızı ve multimedya desteği ihtiyaçlarından dolayı 2000 yılında üçüncü nesil (3G) mobil teknolojisi kullanılmaya başlanmıştır. Bu teknolojinin bant genişliği ve kaynak maliyetleri gibi sınırlamalarının üstesinden gelmek için kablosuz mobil internet destekleyen dördüncü nesil (4G) haberleşme teknolojisi ihtiyacını doğurmuştur. Günümüzde mobil veri hizmetlerinin artışı ile hızlı ve yüksek oranda veri iletimi ihtiyaçlarından dolayı çalışmalar yeni nesil kablosuz haberleşme tekniklerinin gelişimi üzerine yoğunlaşmıştır.

1982 yılında sunulan ve 1990’ların başına kadar kullanılan bu teknoloji (1G), yalnızca ses servisi sunmaktadır. Frekans Bölmeli Çoklu Erişim (FDMA) tekniği ile analog veri iletimi sağlayan bu sistemde her kullanıcı için özel bir frekans kullanımı söz konusudur. 30 KHz kanal kapasitesi ile Gelişmiş Mobil Telefon Hizmeti (AMPS), 25 KHz kanal kapasitesi ile Nordic Mobil Telefon (NMT), Bas ve Konuş (PTT) gibi mobil teknolojiler sunulmuştur (Gawas, 2015). Tek bir ülke içerisinde sesli aramaya izin veren, saniyede 2.4 kb veri iletim hızı bulunan ve yalnızca analog işaret kullanan bu sistem, zayıf ses kalitesi ve verimsiz frekans spektrumu dezavantajlarına sahiptir.

2G, Mobil Haberleşme için Küresel Sistem (GSM)’e dayanan ve 1980’lerin sonunda ortaya çıkan ikinci nesil haberleşme teknolojisidir. GSM Zaman Bölmeli Çoklu Erişim (TDMA) ve Frekans Bölmeli Çoklu Erişim (FDMA) metotlarını kullanmaktadır. Ses iletimi için sayısal işaret kullanan haberleşme yöntemi, düşük hızlarda metin ve fotoğraf mesajı gönderimi sağlamaktadır. Temel avantajları şifrelenmiş telefon görüşmesi, frekans aralığı başına daha fazla kullanıcının kullanılmasının sağlayan radyo frekans spektrumunun çok daha verimli kullanımı ve mobil cihazlar için veri hizmetleridir. GSM’deki gelişmeler ile

2.5G olarak bilinen Genel Paket Radyo Servisi (GPRS) ile saniyede 40 kb ve GSM Evrimi için Geliştirilmiş Veri Hızları (EDGE) ile saniyede 384 kb hızında veri transferi sağlanmaktadır.

3G haberleşme teknolojisi, GSM tabanlı yüksek hızlı veri iletimi ile 2000 yılında piyasaya sürülmüştür. Paket anahtarlama kullanılarak 14 Mbps ve daha fazla veriye izin verecek şekilde geliştirilmiştir. Netliğin artırıldığı Geniş Bant Kablosuz Ağı kullanmaktadır. Ayrıca veri hizmetleri, televizyon / video erişimi, Küresel Dolaşım gibi yeni hizmetler sunmaktadır. 15-20 MHz bant genişliğine sahip bu sistem yüksek hızlı internet servisi ve görüntülü sohbet imkânı sunmaktadır. Mobil cihazlar ve mobil telekomünikasyon için kullanılan bir dizi standardı temel alan bu sistemin en yaygın olarak kullanılan standardı Evrensel Mobil Telekomünikasyon Sistemi (UMTS)'dir. Bu standart ticari kullanımlarda Geniş Bantlı Kod Bölmeli Çoklu Erişim (W-CDMA) olarak bilinmektedir. UMTS'nin geliştirilmesiyle saniyede 1.8, 3.6, 7.2, 14.0 Mbit karışından yükleme hızlarında internet erişimi sağlayan Yüksek Hızlı İniş Paketi Erişimi (HSDPA) standardı 3.5G olarak isimlendirilmektedir. HSPA+ ile bu hız 42 Mbit'e ulaşmıştır. Mikrodalga Erişimi için Dünya genelinde Uyumluluk (WiMAX) standardı 50 km'ye kadar etki alanına sahip ve saniyede 75 Mbit hızlara kadar ulaşabilen, IEEE 802.16 bir çeşit kablosuz ağ standardıdır. Bu standartta 3G'den farklı olarak Dikgen Frekans Bölmeli Çoklu Erişim (OFDMA) teknolojisi kullanılmıştır (Cox, 2008).

4G; IP tabanlı, saniyede 100 Mbit ile 1 Gbit'e kadar veri iletim kapasitesi sağlayan servis kalitesi ve yüksek güvenliğiyle herhangi bir zamanda herhangi bir yerde her türlü ağ hizmetini tek bir noktada birleştirebilen haberleşme teknolojisidir. 4G; kablosuz geniş bant erişimi, Multimedya Mesaj Servisi (MMS), görüntülü sohbet, mobil TV, Yüksek Çözünürlüklü Televizyon (HDTV) içeriği, Dijital Video Yayını (DVB), ses ve veri gibi minimum hizmetler gibi IP Ağlarında Yeterli Hizmet Kalitesi (QoS) ve ücret gereksinimlerini karşılamak için geliştirilmiştir. Uzun Vadeli Evrim (LTE), ilk Japonya'da NTT DoCoMo firması tarafından 2004 yılında duyurulan mobil radyo haberleşmesinin en önemli standardıdır. LTE'nin gelişimi, Dikgen Frekans Bölmeli Çoğullama (OFDM), Tek Taşıyıcılı- Frekans Bölmeli Çoğullama (SC-FDM), Çoklu Giriş Çoklu Çıkış (MIMO), Turbo kodlama ve dinamik bağlantı adaptasyon teknikleri ile yapılmıştır (Doufexi. ve Armour, 2005). Özellikle OFDM tekniği, 4G LTE için mükemmel bir dalga şekli seçimi

özelliğindedir. Yüksek spektrum verimliliği sağlaması, mevcut mobil el cihazlarında elde edilebilen işlem seviyeleri ile işlenebilir olması ve işlenebilir ve geniş bant genişliklerini işgal eden yüksek veri hızı akışı ile başarılı bir yöntemdir. Ayrıca seçici solmanın olduğu durumlarda iyi çalışmaktadır. 4G sistemlerin MIMO ve OFDM ile kullanılması ile yüksek veri iletim hızı ve yüksek bit hata oranı başarımları ile bu teknikler kablosuz haberleşme için oldukça önem arz etmektedir. OFDM, Dijital Ses Yayını (DAB), Karasal Dijital Video Yayını (DVB-T), Telsiz LAN (kablosuz yerel ağ (WLAN) IEEE 802.11a ve ETSI/BRAN HiperLAN/2), IEEE 802.16a, IEEE 802.15.3a, Sayısal Abone Hattı (DSL) gibi standartlarda kullanılmaktadır.

Günümüzde 5G Yeni Radyo (NR), yüksek veri hızı, düşük gecikme süresi, enerji tasarrufu, maliyet azaltma, daha yüksek sistem kapasitesi ve devasa cihaz bağlantısı nedeniyle daha popüler hale gelmektedir. NR ve LTE, 5G radyo erişiminin ayrılmaz parçalarıdır. LTE'nin 6 GHz frekansının altında çalışması beklenirken, NR'nin 1 GHz'den 100 GHz'e kadar çalışması öngörülmektedir. 5G NR, LTE ile karşılaştırıldığında, ölçeklenebilirlik OFDM ve mobil geniş bant hizmetlerinin ötesinde çeşitli kullanım durumları (insan merkezli ve makine merkezli) gibi benzersiz özelliklere sahiptir. Bahsedilen 5G NR teknolojisinde radyo dalga formları çoklu erişim teknikleri kullanılarak elde edilmektedir. Çevrimsel-Önek Dikgen Frekans Bölmeli Çoğullama (CP-OFDM) ve Ayrık Fourier Dönüşümü-Yayımlı (DFT-S) OFDM, 5G Gelişmiş Mobil Geniş Bant (eMBB) için doğru seçimlerdir. LTE, uydu-yer bağı iletimleri için, CP-OFDM dalga formu ve yer-uydu hattı iletimleri için DFT-Yayımlı OFDM kullanılmaktadır.

5G'ye giden yolda, mobil iletişim topluluğu NR için birçok dalga formu önerisiyle karşılaşmıştır. Bu önerilerin çoğu sisteme çok-taşıyıcılı dalga formları sağlayan CP-OFDM'in çeşitleridir. 5G ile ilgili yapılan çalışmalarda OFDM yönteminde ufak tefek değişiklikler yapılmak suretiyle; alt taşıyıcı filtreleme veya darbe şekillendirme, alt taşıyıcı gruplarının filtrelenmesi, art arda sembollerin zamanla üst üste binmesine izin verme, çevrimsel öneki sıfırlama veya yerine başka bir dizi kullanma gibi yöntemler mevcuttur. Yüksek hızlı video indirme, oyun, araçtan araca / araçtan altyapıya iletişim, genel hücresel iletişim, Nesnelerin İnterneti (IoT) / Makineler Arası Bağlantı (M2M) iletişimi ve benzerleri de dahil olmak üzere tüm mobil iletişim potansiyel uygulamalar için 5G gerekli performansı sağlayabilmektedir.

OFDM, sayısal verilerin çok kanallı ortamlarda bir radyo kanalı üzerinden verimli ve güvenilir bir şekilde aktarılmasını sağlayan bir modülasyon tekniğidir. OFDM çok sayıda dar bant alt taşıyıcılarla verileri iletmektedir. Bu alt taşıyıcılar, belirli frekans aralıklarında bir spektrum bloğu oluşturmaktadırlar. Taşıyıcıların frekans aralığı ve zaman senkronizasyonu, taşıyıcıların dik olacağı şekilde, yani birbirlerine müdahale etmeyecek şekilde seçilmektedir. Taşıyıcıların birbirine dik olması aynı zamanda Semboller Arası Etkileşimi (ISI) de ortadan kaldırmaktadır. Kısmen örtüşen alt taşıyıcılar için spektral koruyucu bantların yerleştirilmesi, geleneksel Frekans Bölmeli Çoğullamaya (FDM) göre daha yüksek spektral verimlilik sağlamaktadır.

OFDM sisteminde modülatörler yerine Ayrık Fourier Dönüşümleri (DFT)/Hızlı Fourier Dönüşümleri (FFT) kullanılmaktadır. Sürekli artan yüksek veri hızları talebiyle karşı karşıya olan kablosuz iletişim tekniklerinin mevcut durumunda, tek taşıyıcı sistemler, geniş bant kanalının frekans seçiciliği nedeniyle alıcı ucunda eşitleyici tasarımında ciddi karmaşıklıklara neden olan sınırlı çözümler sunmaktadır (Suma ve Kanmani, 2011). Çok taşıyıcılı bir sistem olan OFDM, yeni nesil kablosuz iletişim yöntemleri için etkili bir modülasyon tekniği haline gelmiştir. FFT algoritmaları bu sistemler için hızlı veri işleme gereksinimlerini karşılamaktadır.

FFT, sayısal işaret işleme uygulamalarında sıklıkla kullanılmaktadır ve oldukça önemli bir yere sahiptir. FFT işlemci tasarımında hedef, kullanılan donanımı, işlem süresini ve gecikmeyi en aza indirmektir (Saleh ve Swartzlander, 2012). Hızlı Fourier Dönüşümü ve Ters Hızlı Fourier Dönüşümü (IFFT), DFT işleminin yüksek verimli ve hızlı algoritmalarla hesaplanma biçimidir. DFT, zaman alanındaki işaretleri frekans alanında ifade etmeye yarayan bir yöntemdir. Aynı zamanda bir dalga formunun kendisini oluşturan sinüzoidal bileşenlere dönüştürmesi olarak da tanımlanabilir. FFT alanında geliştirilen çalışmalar hem algoritma hem de uygulama seviyelerinde performans arttırmak için yapılmıştır. Karmaşıklığın azaltılması, güç tüketimini ve donanım miktarını en aza indirmek için oldukça önemlidir (Lee ve Chen, 2009). Bu performans iyileştirici gereksinimler, yazılım uygulamalarının başarısız olduğu yüksek performans gerektiren işlemlerde Uygulamaya Özel Tümlleşik Devreleri (ASIC) ile karşılanmaya çalışılmıştır. Son zamanlarda Alanda Programlanabilir Kapı Dizileri (FPGA) teknolojisi oldukça gelişmiş olup, karmaşık ve hızlı işlem yapabilme yeteneğinin yanı sıra yeniden programlanabilirlik özelliğiyle gerçek

zamanlı FFT uygulamalarında önemli rol oynamaktadır. FPGA platformlarında FFT algoritmaları uygulanırken hedef, verimliliği artırmak ve kullanılan eleman sayısını en aza indirmektir. FPGA'lerin hızlı işlem yapabilme kabiliyeti sayesinde geniş bant sayısal iletişim sistemleri için gerçek zamanlı FFT uygulamalarında başarılı sonuçlar elde edilmektedir.

Kablosuz haberleşmede modülasyon tekniği için en etkili yöntemin FFT tabanlı OFDM uygulaması olduğu gösterilmiştir (Nee ve Prasad, 2000). Ancak, OFDM'in gerçek zamanlı işlemlerde yüksek verim elde edebilmesi için hızlı ve etkili FFT işlemcilerinin kullanılması gerekmektedir. Literatürde bu işlemciler hafıza-tabanlı (memory-based), Önbellek tabanlı (Cache Memory-based), Sıralı (Sequential), Paralel, Paralel yinelemeli (Parallel iterative), Dizi mimari (Array architecture), Ardışık düzen (Pipeline) FFT (Joshi, 2015) olarak sınıflandırılmışlardır. Yapılan çalışmaların Ardışık düzen FFT (He ve Torkelson, 1996) ve Hafıza-tabanlı FFT (Chang vd., 1999) üzerine yoğunlaştırdığı görülmüştür. Ardışık düzen FFT uygulamalarında, hafıza-tabanlı FFT yapılarına oranla daha yüksek performans elde edilmekte ancak daha fazla İşlem Ögesi (PE) yani kelebek devre elemanı kullanılmaktadır. Öte yandan, hafıza-tabanlı mimari, gerçek zamanlı işlemi gerçekleştirmek için giriş ve ara verileri depolamak amacıyla bazı hafıza bloklarına ve yalnızca bir İşlem Ögesine ihtiyaç duyar (Wey vd., 2007). Genel olarak, devre elemanı kullanımı konusunda oldukça başarılı olan FPGA'da uygulaması yapılabilen hafıza-tabanlı FFT yapıları OFDM haberleşme sistemlerinin gerektirdiği son teknoloji gereksinimlerini karşılamaktadır.

OFDM, mobil iletişimde de yaygın olduğundan küçük, verimli ve hızlı donanımın seçimi/ tasarımı kritik bir konudur. Bu sistemlerde FFT en yoğun işlemli kısımdır. Bu yüzden FFT hesaplamalarının en az yer tutacak şekilde ve en az devre elemanı ile gerçekleştirilmesi önemlidir. Tez çalışmasında, FFT uygulamasında minimum (N) hafıza kullanılarak geliştirilen adresleme algoritmalarıyla hızlı işlemci tasarımları yapılmıştır. Uygulama FPGA ile gerçekleştirilmiş ve aşağıdaki parametrelerin en iyilenmesi göz önünde bulundurulmuştur:

1. Hız (saat frekansı)
2. Eleman sayısı (yonga üzerinde kaplanan alan)

3. Devre karmaşıklığı (elemanlar arasındaki bağlantı sayısı)
4. Gecikme (ardışıl FFT işlemleri arasında geçen süre)
5. Kullanım verimliliği (devre elemanlarının kullanım/(kullanım+bekleme) oranı)

Bu parametreler göz önünde bulundurulmuş, OFDM iletişim sistemlerinin en önemli bileşenlerinden biri olan FFT bloğunun FPGA içindeki çift portlu Rastgele Erişimli Bellek (RAM) kullanılarak uygulama yapılmış ve bu hafıza yapılarına uygun yeni hafıza adresleme algoritmaları geliştirilmiştir.

Tez kapsamında yapılan çalışmalar:

- OFDM iletişim sistemi için uygulanabilirliği olan FFT algoritmaları incelenmiş ve hafıza-tabanlı FFT yapıları üzerine çalışmalar yapılmıştır.
- Hafıza-tabanlı FFT yapılarının FPGA ile uygulaması yapılmıştır.
- FFT yapıları için yeni hafıza adresleme algoritmaları geliştirilmiş ve algoritmaya uygun çift portlu RAM'ler kullanılmıştır.

Tez çalışmasında, OFDM tekniğinin en önemli bileşeni olan Hızlı Fourier Dönüşümü işlemlerini radiks-2 algoritması ile hafıza-tabanlı FFT yapıları kullanarak ve bu yapılara uygun olabilecek en hızlı adresleme algoritmaları geliştirilerek FPGA üzerinde uygulaması yapılmıştır. FPGA üzerindeki çift portlu hafızaların hafıza-tabanlı FFT uygulamaları için önemli rol oynadığı ve OFDM için büyük kolaylık ve ilerleme sağladığı görülmüştür. Radiks-2 hafıza-tabanlı FFT metodu ile yeni hafıza adresleme algoritmaları geliştirilmiş ve algoritmaya uygun çift portlu RAM yapıları kullanılmıştır.

2. LİTERATÜR ARAŞTIRMASI

Literatürde birçok bilim insanı FFT işlemlerinin en az hafıza ve en az saat darbesiyle tamamlanması için yeni adresleme algoritmaları geliştirmişlerdir. Bu algoritmalarda amaç kullanılan devre elemanı, güç tüketimi ve işlem süresi bakımından en iyileşmeyi hedeflemenin yanı sıra gerçek zamanlı uygulamalar için devreye sürekli veri akışı sağlayan, paralel işlem yapabilme özelliği ve sıralı düzenli giriş/çıkış verebilen çalışmalar olmaktadır. Tez çalışması kapsamında literatür araştırmasında, OFDM sistemleri için tasarlanmış öncelikli olarak hafıza-tabanlı FFT yapıları incelenmiş, bazı ardışık düzen ve dönüşüm çekirdeği üzerine yapılan çalışmalara da yer verilmiştir. Bu çalışmalar aşağıda özetlenmiştir.

Wey vd. (2007) radiks-2 hafıza-tabanlı FFT yöntemiyle gerçekleştirdikleri çalışmada az donanım kullanarak gecikmeyi en aza indirmeyi ve yüksek çalışma frekansına ulaşmayı hedeflemişlerdir. Yapılan çalışmada amaç çift portlu RAM kullanmak yerine tek portlu RAM ve ön-bellekli kaydedicileri birlikte kullanarak hafıza kullanımını azaltmaktır. Tipik bir hafıza-tabanlı FFT yapısında, 5 adet $N/4$ çift portlu RAM yani toplamda $1.25 N$ hafıza kullanıldığını, N -noktalı FFT hesaplamasında $N/2 + N/2 \log(N)$ döngülük bir gecikme olacağını söylemişlerdir. Tek portlu RAM ile bu gecikmeyi ve hafıza kullanımını azaltacak bir tasarım sunmuşlardır. Tasarımda saat darbesi döngüsünü ikiye bölmüş, yani N -noktalı FFT dönüşümünü $N/2$ saat darbesinde gerçekleştirmişlerdir. Çift portlu hafızalarda okuma/yazma işlemi 1 saat darbesi döngüsünde tamamlanırken, önerilen tek portlu hafızada bu işlem için 2 saat darbesi gerekmektedir. Saat darbesi döngüsünü yarıya indirmek için yani işlemi 1 saat darbesi süresinde gerçeklemek için tek portlu hafıza ile ön-bellekli kaydediciler kullanmışlardır. Böylece N adet hafıza kullanmış ve gecikme süresini $3N/4 + N/4 \log(N)$ 'e indirmeyi başarmışlardır.

Oruklu vd. (2012) radiks-2 ve radiks-4 yöntemleriyle FPGA'de gerçekleştirdikleri bu çalışmada, koordinat döndüren sayısal bilgisayar tabanlı (CORDIC) algoritma kullanarak hafıza gereksinimini ve güç tüketimini azaltmaya yönelik uygulama geliştirmişlerdir. Alışlagelmiş CORDIC tabanlı FFT çalışmalarında dönüşüm çekirdeği açısı değerlerini kaydetmek için Salt Okunur Bellek (ROM) kullanılmaktadır. N -noktalı radiks-2 ve m -bit FFT için, $N/2$ açısı kaydedilmesi ve $mN/2$ bit ROM'a ihtiyaç duyulmaktadır. CORDIC

algoritması öteleme ve toplayıcı elemanlarıyla çarpıcı kullanmayan yinelemeli bir metottur. Çalışmada bu metodu açıcı değerlerinin kaydedilmesini ortadan kaldıran yani hafıza (ROM) kullanmayan, açıcıları akümülatör ile üreten bir uygulama sunulmuştur. Bu yinelemeli algoritmada öteleme ve toplama işlemleriyle birlikte radiks-2 algoritması için 4 adet kaydedici ve 8 adet çoklayıcı (2-to-1 mux) kullanılmıştır. Genel olarak radiks-r FFT için $2r_2$ kaydedici ihtiyacı bulunmaktadır. İyi bir hafıza adreslemesi kontrolü ve akümülatör ile gerçekleştirilen bu tasarımda, radiks-2 m-bit FFT için ihtiyaç duyulan hafıza $5mN/2$ 'den $2mN$ bite indirilmektedir. Bu da hafıza kullanımında %20'lik bir azalma anlamına gelmektedir. Yüksek radiksli uygulamalarda bu değerler, radiks-r için $(3r - 1)mN/r$ 'den $(r - 1)mN/r$ 'ye kadar indirgenmekte yani %33 lük bir azalma anlamına gelmektedir. Radiks-2 ve radiks-4 için STRATIX-III FPGA ile yapılan uygulamada devre elemanları kullanımında az miktarda artış gözlemlenmiş ancak hafıza kullanımında %20 ve buna bağlı olarak güç tüketiminde %15 oranında azalma olmuştur.

ADSL, UWB, DAB, ...vb. gibi kablosuz haberleşme teknolojisi standartları için 128-8192 noktalı DFT hesaplama aralığı gerekmektedir. Bu gibi yüksek FFT hesaplamalarında çarpıcı kullanımının donanım uygulamaları için uygun olmadığını savunan Sung vd. (2010), bu ihtiyacı ortadan kaldırmak için sabit çarpan üretiminde CORDIC algoritma uygulamasını kullanmayı uygun görmüşlerdir. Bu uygulamada 128-noktalı FFT için 16-bit akümülatör kullanılmaktadır. ROM ihtiyacı olmayan CORDIC algoritma ile dönüşüm çekirdeği üretimi 128-noktalı FFT hesaplaması için 2 toplama ve 2 kaydırma, maksimum operasyonda ise 6 toplama ve 2 CORDIC operasyon işlemi gereksinimi bulunmaktadır. Bu algoritma ile fazladan süreye ihtiyaç duyulmamaktadır. Tasarımda split-radiks FFT algoritması kullanılmış ve yerinde kullanılabilir (in-place) hafıza yaklaşımıyla donanım maliyetini azaltma hedeflenmektedir. Split-radiks algoritması FFT'nin tek ve çift kısımlarının ayrı ayrı hesaplanmasıyla elde edilmektedir. Split-radiks 2/8 FFT işlemcisinin kelebek işlemleri hesaplaması için 16 karmaşık toplayıcı, 2 sabit çarpan (CM) ve 4 CORDIC operasyon ihtiyacı bulunmaktadır. Kullanılan sabit çarpan her ne kadar çarpıcı kullanıyor gibi görünse de sadece toplama ve kaydırma işlemlerinden ibaret olmaktadır. Böylece kullanılan çip alanından tasarruf edilmekte ve daha az enerji kullanılmaktadır. Uzunluğu 16-bit olan FFT hesaplaması yapılmış ve 200 MHz'lik performans elde edilmiştir. Ayrıca benzer çalışmalara bakılırsa oldukça yüksek çalışma frekansı elde edildiği görülmüştür.

Radiks-2 DIF kelebek yapısı ile gerçekleştirilen Yang vd. (2012)'nin tasarımı DIT yapısına da uygulanabilmektedir. Çalışmanın amacı alan kullanımını ve güç tüketimini azaltmaktır. Bu amaçla kelebek yapısını gerçeklemede sabit çarpan kullanılmaktadır. Sabit çarpan sayısını ve hafıza kullanımını azaltmak amacıyla ardışık düzen yapısı olan SDF, MDF yapısına tercih edilmektedir. Uygulanan SDF yapısı alt bloklara bölünerek paralel bir yapı oluşturulmuş ve bu uygulamayla enerji tasarrufu sağlanmıştır. N -noktalı FFT M ve L -noktalı iki alt birime bölünmekte, paralel çalışan ve daha az alan gereksinimi olan bir devre gerçekleştirilmektedir. $M = 256$ ve $L = 8$ olmak üzere, bölünmüş paralel hafıza hesaplama yapısıyla 128 ve 2048 noktalı FFT çalışmaları arasındaki kullanılan hafıza miktarı artışı 4 kattan daha az hale getirilmektedir. Çünkü 256 ve 8-noktalı FFT'ler ayrı iki modülde hesaplanmaktadır. Bu uygulama ile iki modül arasında aynı sonucu veren hesaplamalar ortaya çıkmaktadır ve bu sayede her ikisi modül için de tekrar hesaplama gereksinimi ortadan kalkmaktadır. Ayrıca trigonometrik hesaplamalar yaklaşımıyla dönüşüm çekirdeğini saklamak amacıyla ayrıca hafıza kullanmaya gerek kalmamaktadır. 128 ve 2048-noktalı FFT hesaplamaları gerçekleştirilmiştir ve 1.25 MHz- 20 MHz arasında bant genişliğini desteklemektedir. Hafıza kullanımı açısından etkili bir yöntem sunmayan çalışma paralel ardışık düzen yapısıyla az yer kullanımı ve güç gereksinimi sebebiyle tercih edilmektedir.

Hafıza adresleme karışıklığını önlemek amacıyla son zamanlarda yapılan çalışmalardan bir tanesi de sabit geometri (Constant Geometry) çalışmalarıdır. Xing vd. (2017) adresleme karışıklığını önlemek amacıyla hafıza tabanlı FFT yapısını mixed-radiks algoritması ile yeni bir adresleme algoritması önermektedir. Sabit geometri yaklaşımında yerinde kullanılabilir hafıza uygulaması yapılamazken gerçekleştirilen adresleme algoritmasıyla hafıza erişimi karmaşıklığı yaşanmadan bu uygulama yapılabilir. Bu adresleme algoritmasında adres üretici iki ana kısımdan oluşmaktadır. İlk kısımda hafıza bankalarının adreslemesi sayıcı yardımıyla hesaplanıp hafızadaki verileri sırasıyla çıkışa vermektedir. Bu değerlerin uygun yollara dağıtılması ise bir akümülatör sayesinde yapılmaktadır. Kaydırılan veri girdilerini çıkışa yönlendirmek amacıyla barrel kaydırıcı kullanılmaktadır. İkinci kısımda hesaplamaların tamamlanması için vektörler arasında XOR işlemi yapılmaktadır. Son olarak da çıkış verilerinin uygun yerlere yönlendirilmesi için çoklayıcılar kullanılmaktadır. Yapılan paralel FFT çalışmasında sabit geometri yapısı yerinde kullanılabilir hafıza algoritmasıyla hafıza adresleme karışıklığı yaşanmadan $2N$ hafıza kullanılarak yeni bir adresleme algoritması sunulmaktadır.

Ayinala vd. (2013) çalışmalarında yerinde kullanılabilir hafıza ile FFT hesaplamalarını gerçek işaretler için kullanmaktadırlar. Radiks-2 algoritmasıyla karmaşık değerler yerine gerçek ve sanal kısımlar W kelime uzunluğunda birlikte temsil edilmektedir. 32-noktalı gerçek örnekler bu yaklaşım sayesinde 32 örnek yerine 17 örnek üretilmektedir. Kelebek hesaplamaları için 2 karmaşık toplayıcı yerine 2 gerçek toplayıcı kullanmaktadırlar. 2 adet hafıza modülü kullanılmakta, her bir hafızaya N adet W uzunluğunda veri kaydedilmektedir. Ayrıca her bir hafıza modülünde 4 ayrı hafıza bankası kullanılmış ve ardışık düzen FFT uygulamasına benzer adresleme uygulanmaktadır. Hermitian-symmetric IFFT işlemcisi 16-noktalı gerçek dizinin yalnızca 9 giriş örneğiyle kullanımını sağlamaktadır. Benzer çalışmalarda bakıldığında $2W$ olması gereken kelime uzunluğu W 'ye indirgenmiş ve toplamda $N/4W$ kadar hafıza kullanılmaktadır.

Garrido vd. (2014) yüksek hesaplamalı FFT işlemlerinde uygun kelime genişliği, radiks ve paralel işlem yapabilme parametrelerinden dolayı FPGA içindeki IP core'ların kullanımını uygun görmektedirler. IP core'lar MDC tipinde çalışmaktadır. İlave hafıza ihtiyacı olmadan Virtex-6'da VHDL ile uygulaması yapılan çalışmada en fazla 6x536-noktalı FFT hesaplaması yapılabildiği vurgulanmaktadır.

MDC ile yapılan bir başka çalışmada Tsai vd. (2011)'nin çalışmasıdır. Radiks-2, radiks-4 ve radiks-8 algoritmalarının birlikte kullanımıyla 2 paralel PE elemanı kullanan radiks-2'ye göre %50 daha az karmaşık çarpıcı kullanılacağını sunmuşlardır. Ayrıca radiks-4 ile radiks-2'de kullandıkları 2 kelebek devre elemanından birine ihtiyaç kalmayacağını üstelik devrede gecikme gibi durumun olmayacağını iddia etmişlerdir. 4 adet bölünmüş hafıza ile toplamda $2N$ hafıza kullanan bu adresleme algoritmasıyla devreye sürekli akış veri sağlanmaktadır. 64-4096-noktalı MDC FFT ile uygulama sonuçları ile SDF ardışık düzen yapısına göre daha az çarpıcı gereksinimi bulunmaktadır. Devrenin tüm FFT işlemleri için toplam çalışma süresi $N/4\log_8 N$ 'dir.

Luo vd. (2015) güç tüketimini ve hafıza kullanımını azaltmak amacıyla tek portlu, çift portlu ve ön-bellek hafızalarda kullanılabilecek bir adresleme algoritması uygulamaktadır. Hafızanın bölünerek kullanılması benzer çalışmalara kıyasla hafıza kullanımında azalma sağlamak ve adresleme karışıklığının önüne geçmektedir. Örnek olarak 3 aşamada hesaplanan 8-noktalı FFT'nin ilk aşamasında kullanılan (0,4) ve (2,6)

örnekleri hesaplama sonucu ikinci aşamada benzer şekilde (0,2) ve (4,6) hesaplamalarında kullanılmak üzere 2 ve 4 örneklerinin yer değişimi ile tekrar hesaplama yapılmasına gerek kalmamaktadır. Bu adresleme ile bölünmüş hafıza kullanımı tek portlu hafızalarda adresleme karışıklığını çözmektedir. Ancak 1024x32'lik bir hafıza için 2x512x32 olacak şekilde ikiye bölünme işlemi %16.1, 4x256x32 olacak şekilde dörde bölünme işlemi %48.4 kullanım alanının artması anlamına gelmektedir. Kullanım alanının artışı güç tüketimiyle doğru orantılıdır.

Çoklu hafıza kullanımı üzerine yapılan çalışmalardan bir tanesi de Hsiao vd. (2010)'nin mixed-radiks algoritmasıyla yaptıkları uygulamadır. Yerinde kullanılabilir hafıza yaklaşımı ile ikiye bölünen hafıza ile gerekli toplam saat darbesi kullanımı yarıya inmektedir. Çalışmada $2N$ hafıza kullanılmaktadır.

Dönüşüm çekirdeğini ROM hafıza da saklamak yerine Bansal vd. (2014) çalışmalarında CORDIC algoritmayı tercih etmişlerdir. Böylece hem hafıza kullanımından tasarruf edilmiş hem de çarpıcı kullanımı gereksinimi olmamaktadır. Yerinde kullanılabilir hafıza algoritması FFT hesaplama sonuçlarını saklamak için paralel-girişli seri- çıkışlı adresleme ile tek bir N boyutlu RAM kullanılmaktadır. 8-noktalı radiks-2 DIF FFT çalışmasında Koordinat Döndüren Sayısal Bilgisayar (CORDIC) algoritması için 19-bitlik akümülatör kullanılmaktadır. Benzer çalışmalarla kıyaslandığında hafıza kullanımında %23'lük bir azalma olmaktadır. Bu azalma benzerlerinde ulaşılan 257 MHz'lik çalışma frekansını 270 MHz'e çıkarmayı başarmaktadır.

Xia vd. (2017) modüler aritmetik ve çarpıcı kullanarak hafıza-tabanlı FFT çalışması yapmışlardır. Bu çalışmada amaç yüksek radiksli algoritmaların hafıza karmaşıklığına sebep olmayacak verimli bir adresleme algoritması gerçekleştirmektir. 2048-noktalı FFT hesaplamasını MDC paralel hesaplama yapısıyla gerçekleştirmiş ve 122.88 MHz'lik çalışma frekansı elde edilmektedir. Tasarlanan adresleme algoritmasıyla devreye sürekli veri akışı sağlanmakta, ancak benzer çalışmalara oranla fazla sayılabilecek $3N$ hafıza kullanılmaktadır.

Sıralı giriş/çıkış sağlayan çalışmalardan Chang (2008)'in çalışmasında devreye sürekli veri akışı sağlayan Çoklu-Yol Gecikmeli Geribildirim (MDF) ve Tekli-Yol

Gecikmeli Geribildirim (SDF) yöntemleri kullanılmıştır. Chang çalışmasında sıralı düzen elde etmek için verileri sıraya sokan ek devre elemanı (dizi çevirici) kullanmıştır. Bu işlemi yapmak için $N/2$ saat darbesi harcanmaktadır. Ardışık-düzen yapısına örnek olan bu çalışmada FFT işlemleri için $1.5N$ olmak üzere tüm işlemler için $2N$ hafıza kullanılmaktadır. Ayrıca bu yapıların kullanılan yer, devre elemanı ve güç tüketimi açısından hafıza-tabanlı yapılara göre dezavantajı olduğu unutulmamalıdır.

Huang ve Chen (2010) bir radiks-4 ve iki radiks-2 kelebek devre elemanını oluşturduğu radiks-16 FFT çalışmalarında çoklu-blok (32 alt bloktan oluşan) hafıza-tabanlı, 512-noktalı IEEE 802.15.3c (WPANs) standardına uygun sıralı düzen çıkışlı bir çalışma sunmuşlardır. Çalışmada, sıralı düzen çıkış elde edilmesine rağmen, çok sayıda ardışık-düzen PE yapısı kullanılmış ve bu yapıdan kaynaklı 19 olmak üzere toplamda 71 saat darbesi gecikmeye sebep olmuştur. Ardışık düzen PE yapılarının gecikme dezavantajının yanı sıra devrenin kullanım alanı (2.46 mm^2) ve güç tüketimi (103.5 mW) açısından benzer çalışmalara kıyasla verimsiz olduğu görülmektedir.

Huang ve Chen (2012)'in bir başka sıralı düzen çıkış elde ettikleri çalışmalarında 4 PE devre elemanının kullanıldığı 512-nokta radiks-16 FFT ve 16 alt bloktan oluşan hafıza ile yerinde kullanılabilir hafıza-tabanlı ve sürekli akış sağlayan bir algoritma sunmuşlardır. Çıkışta sıralı düzen elde etmek için fazladan hafıza ihtiyacı olmayan bu çalışmada FFT işlemleri yapılırken paralelinde yeni verilerin alınmasıyla, bir önceki çalışmadaki gecikme süresini azaltmışlardır. Tasarımda 32 saat döngüsünde verilerin alındığı ve FFT işlemlerinin yapıldığı (4 PE ile) 72 saat darbelik sürenin son 32 saat darbesinde yeni verilerin alındığı bilgisi verilmektedir. Bir önceki aşama verilerin çıkışa verilmesi esnasında o anki aşamadaki FFT işlemlerine başlanmış olacaktır. Böylece sürekli akış sağlanmakta fakat kullanılan hafıza miktarı $2N$ olmaktadır. Ancak hafıza yazma adreslerinin okuma adreslerinde 7 saat darbesi sonra hesaplanması ve kullanılan ardışık-düzen yapıları devrede toplam operasyon saatinde gecikmeye sebep olmaktadır. Paralel işlem özelliğiyle güç tüketimi (42 mW) ve kullanılan alan (0.93 mm^2) özelliği bakımından bir önceki çalışmalarına göre daha iyidir.

Jo ve Sunwoo (2005) yerinde kullanılabilir adresleme algoritmasıyla (radiks-2/4) karma-radiks FFT ile hafıza tabanlı çalışmalara yeni bir adresleme algoritması sunmuşlardır. Bu çalışmada 4'er alt bloktan oluşan toplamda $2N$ hafıza kullanılmaktadır. Sunulan

adresleme algoritmasıyla sürekli veri akışı mümkün olabilmekte ancak çıkış verilerinin sıralı düzende olmadığı, sıralamak için ek bir devre tasarımı gerektiği görülmektedir. Üstelik yerinde kullanılabilir algoritmanın N boyutlu hafızayla kullanılarak yapılabildiği göz önünde bulundurulduğunda, fazla hafıza kullanımı söz konusudur.

Ma vd. (2015) radiks-2 DIF $N = 256 - 4096$ nokta FFT için yaptıkları hafıza-tabanlı ve sadece gerçek değerli verileri kullandıkları çalışmada, 2 PE devre elemanını paralel kullanarak operasyon saati açısından verim elde etmeyi hedeflemişlerdir. Toplamda N olmak üzere bölünmüş 4 adet çift portlu hafıza bloğu kullanmış ve bu hafızaları karmaşıklık yaratmayan bir adresleme algoritmasıyla kontrol etmişlerdir. Operasyon saati $N/4 (\log_2 N - 1) + 1$ olan bu algoritma devreye sürekli veri akışı sağlayan bir tasarım örneği değildir.

Garrido vd. (2016) 4096- noktalı radiks-4 FFT ile hafıza-tabanlı çalışma sunmuşlardır. Önerilen algoritma $N/4$ boyutta 4 adet bölünmüş hafıza (toplamda N) ve DSP48E'ler (Virtex-5 FPGA'de) ile yalnızca 4096-noktalı FFT için uygulanabilmektedir. Adresleme kontrolleri basit bir sayaç ve komütatör devresi ile yapılabilmektedir. Çift portlu hafızaların kullanımıyla aynı anda okuma/yazma işlemleri yapılabilmekte böylece tüm FFT işlemleri $N/8 (\log_2 N)$ sürede 200 MHz çalışma frekansı ile tamamlanmaktadır. Çalışma 24 bit gerçek ve 24 bit kompleks kelime uzunluğunda verilerle gerçekleştirilmiştir. Toplamda 48 bitlik uzunlukta büyük sayılabilecek verileri işlemede saat frekansını yavaşlatmayacak ve lojik devre elemanı gereksinimini ortadan kaldıracak yüksek frekansta çalışan DSP'lerin kullanımını tercih etmişlerdir.

Dört adet $N/4$ boyutlu bölünmüş hafıza ile yapılan çalışmalardan bir tanesi de radiks-2 FFT algoritmasıyla Ma ve Wanhammar'ın (2000) çalışmalarıdır. Bu çalışmada yerinde kullanılabilir algoritması ve N kadar hafıza kullanımıyla donanım kullanma açısından oldukça verimlidir ancak kelebek giriş/çıkış adres üretiminde 2 adet barrel kaydırıcı kullanılmaktadır.

Mao vd. (2017) hafıza-tabanlı radiks-2 ile gerçekledikleri 2 PE (4 paralel işlem) ve 2 blok hafızanın (her ikisi de alt hafıza bloklarına bölünmüş) ping-pong çalışma prensibiyle sıralı düzen giriş/çıkışlı çalışma sunmuşlardır. Sıralı düzen çıkış elde etmek sürekli veri akışı

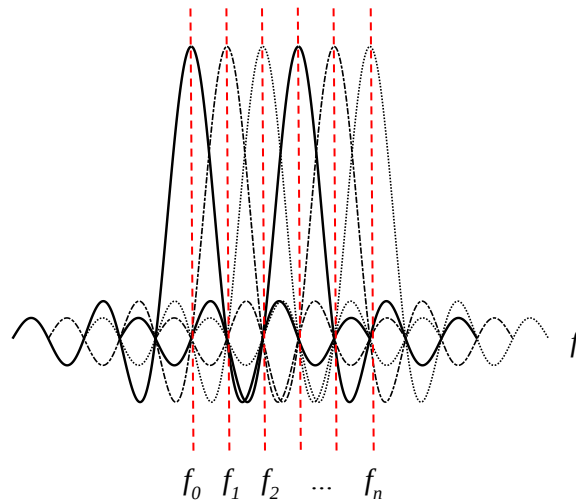
sağlanması için önem arz etmektedir. Bu yüzden çalışmalarının son FFT aşaması $N/4$ (2 PE ile) saat darbesinde tamamlanması gerekirken $N/8$ saat darbesinde tamamlanmıştır. Bu durum çıkış verilerini normal düzene sokmak için yaptıkları yer değiştirme işleminden kaynaklanmadır. Bu işlemin fazladan hafıza kullanmak yerine transistör kullanımı ile yapılabileceğini sunmuşlardır. Örnek olarak $N = 256$ -nokta FFT için 69754 tane transistör kullanılmakta ve bu sayı 4096-nokta için 807034'e ulaşmıştır. Aşamalar arasında da ardışık düzen yapıları kullanılan tasarımın toplam operasyon saati $N/4 (\log_2 N - 0.5) + 1$ ve kullanılan hafıza boyutu $2N + 10$ 'dur. Fazladan kullanılan 10 hafıza ardışık düzen yapıları için kullanılmaktadır.

Sıralı düzen çıkış elde edilen ve böylece gerçek zamanlı FFT işlemler için süreklilik sağlayan çalışmalardan bir tanesi de Tian vd. (2018) radiks-4 FFT ile hafıza-tabanlı çalışmalarıdır. Çalışmada tek portlu hafızaların çift portlu hafızalardan %56 oranında daha az alan kaplaması sebebiyle tek portlu hafıza kullanımını tercih etmişlerdir. $2N$ hafıza kullanımı ve bu hafızaların her birinin 4'er alt bloklara bölünmesi ve her aşamada 'ping-pong' çalışma prensibiyle sıralı düzen çıkış elde ettikleri hafıza adresleme algoritması tasarlamışlardır. Çıkışlarda verileri normal sırayla elde etmek için hafıza yapılarının çıkışında çoklayıcılar kullanılmaktadır. 4 ayrı bölünmüş hafızadan okunan verilerle 4 paralel işlem yapılmakta ve tüm işlemler $N/8 (\log_2 N) - N/4$ saat darbesinde tamamlanmaktadır.

3. GENEL BİLGİLER

3.1. Dikgen Frekans Bölmeli Çoğullama (OFDM)

Frekans Bölmeli Çoğullama (FDM), analog sistemlerde kullanılan çoğullama tekniklerinden biridir. Bu teknik, birçok radyo vericisinin aynı frekans spektrumunda veri göndermesi için kullanılmaktadır. Yöntem, frekans bandını alt-bantlara bölüp, her bir banttı ayrı veri haberleşmesi gerçekleştirmektedir. Her kanaldan aynı anda sadece 1 verici yararlanabilmekte ve tahsis edilen her kanala ait ayrı bir frekans alt-bandı bulunmaktadır. Sayısal sistemlerin sağlamış olduđu avantajlardan tamamen faydalanmak için FDM tekniđi iyi bir çözüm değildir. Bu yüzden sayısal veri modülasyonu özelliđi ile hızlı veri gönderimi yapan Dikgen Frekans Bölmeli Çoğullama (OFDM) tekniđi geliştirilmiştir. FDM gibi, bu teknikte de veri alt-verilere bölünmekte ve her alt-veri birbirine dik taşıyıcıları modüle etmektedir. Böylece yüksek hızlı veri daha düşük hızlı alt-bantlardan iletilmektedir. Şekil 3.1 örnek OFDM sisteminin alt bant tayflarını göstermektedir. Taşıyıcıların birbirine dik olması, alt bant merkezlerinde diđer taşıyıcıların bileşenlerinin 0 olması demektir. Alt bant merkez frekansları arasındaki uzaklık sembol iletim oranları ile doğrudan ilişkilidir ve dikliđin bozulmaması için kontrol edilmesi gerekmektedir.



Şekil 3.1. OFDM'deki birbirine dik alt taşıyıcılar.

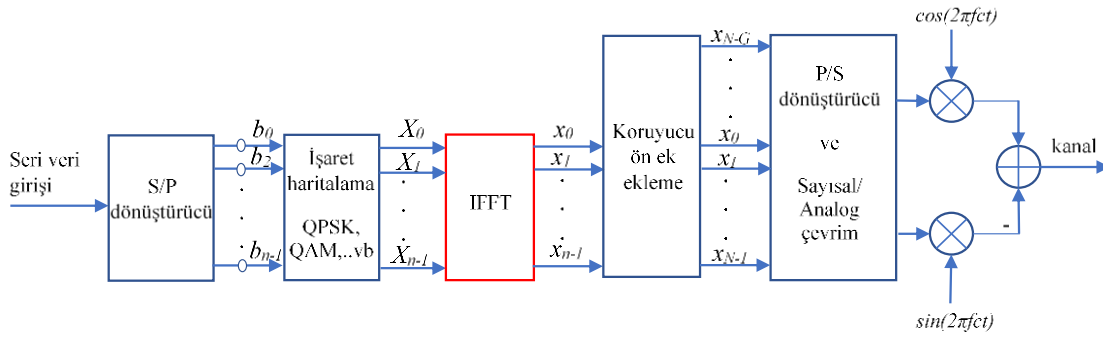
Çok taşıyıcılı modülasyon olarak adlandırılan bu sistem sembol süresini uzun tutup kanal kaynaklı gecikmelerin yeterince büyük olmasını sağlayarak semboller arası etkileşimi (ISI) yok etmekte ya da en aza indirmektedir. Bu nedenle, sembol süresinin küçük olduğu yüksek veri hızı sistemlerinde, veri akışını birçok paralel akışa bölmek, her bir akışın sembol süresini arttırır; böylece gecikme yayılımı sembol süresinin çok küçük bir kısmına denk gelmektedir.

Bir OFDM tasarımında FFT'nin boyutu, çoklu yol olasılığı, Doppler etkisi, tasarım karmaşıklığı/ maliyeti parametreleri göz önünde bulundurularak seçilmelidir. Belirli bir bant genişliği için, büyük bir FFT boyutu seçildiğinde, alt taşıyıcı aralığını azaltacak ve sembol süresini arttıracaktır. Bu, çoklu yol gecikme yayılımına karşı korunmanın daha kolay olmasını sağlamaktadır. Bununla birlikte, azaltılmış bir alt taşıyıcı aralığı, mobil uygulamalarında Doppler yayılımı nedeniyle sistemi taşıyıcılar arası etkileşime (ICI) karşı daha savunmasız hale getirmektedir. Bir OFDM tasarımında gecikme ve Doppler yayılımının etkileri düşünülerek dikkatli dengeleme gerektirmektedir.

OFDM, DFT ve IDFT kullanarak sayısal olarak gerçekleştirilebilmektedir. Bu matematiksel işlemler, veriyi zaman-alan ve frekans-alan arasında dönüştürmek için yaygın olarak kullanılmaktadır. Bu dönüşümler OFDM açısından ilginçtir çünkü dikey alt taşıyıcılara veri eşleme olarak görülebilmektedir. Örneğin, IDFT frekans-alan verilerini almak ve zaman-alan verisine dönüştürmek için kullanılmaktadır. Bu işlemi gerçekleştirmek için IDFT, frekans-alan girdi verisini, belirli frekanslarda sinüzoidal olan dikey temel fonksiyonlarıyla ilişkilendirmektedir. Bu korelasyon, girdi verilerini sinüzoidal temel fonksiyonlara haritalamaya eşdeğerdir.

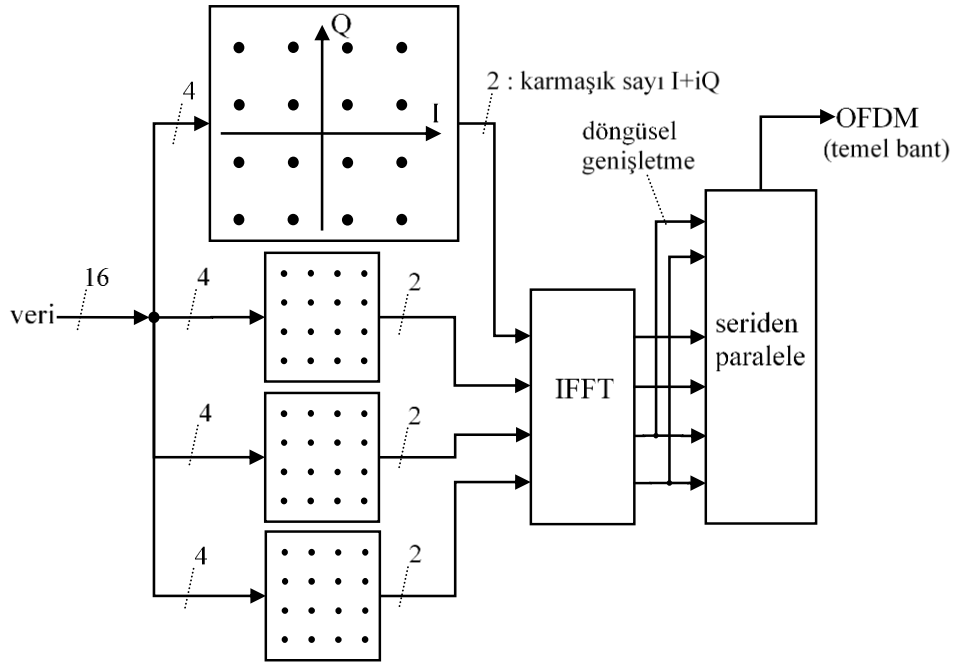
Uygulamada, OFDM sistemleri vericide IFFT alıcıda FFT kullanılarak gerçekleştirilmektedir. OFDM verici kısmında IFFT ve diğer işlemlerin yer aldığı blok diyagram Şekil 3.2'de sunulmuştur. Bir OFDM sistemi, vericideki kaynak sembolleri (örneğin, tek bir taşıyıcı sistemde mevcut olacak QPSK veya QAM simgeleri) frekans alanında gibi işlemektedir. Çünkü kümelenme diyagramındaki her bir sembol aslında ilgili frekanstaki taşıyıcının fazör temsilidir. Bu simgeler, sinyali zaman alanına getiren bir IFFT bloğunun girişleri olarak kullanılmaktadır. IFFT, sistemdeki alt taşıyıcıların sayısı N olmak üzere, N sembol almaktadır. Bu N giriş simgelerinin her biri T saniyelik bir sembol

periyoduna sahiptir. Bir IFFT için temel fonksiyonların N adet dik sinüzoidal olduğu bilinmektedir. Bu sinüzoidallerin her biri farklı bir frekanstadır ve en düşük frekansı sıfır frekansı (DC)'dir. Giriş sembolleri karmaşık olduğu için sembolün değeri hem alt taşıyıcı için sinüzoidal genliğini hem de fazını belirlemektedir. IFFT çıktısı, tüm N sinüzoidallerin toplamıdır. Dolayısıyla, IFFT bloğu, veriyi N dikey alt taşıyıcıyı modüle etmek için basit bir yol sağlamaktadır. IFFT'nin N çıkış örnekleli bloğu, tek bir OFDM sembolü oluşturmaktadır. OFDM sembolünün zaman alanında uzunluğu NT olmakla birlikte, T , IFFT giriş sembol periyodudur.



Şekil 3.2. Dikgen Frekans Bölmeli Çoğullama verici kısmı.

Bazı ilave işlemlerden sonra, IFFT'den sonuçlanan zaman-alan sinyali kanal boyunca iletilmektedir. Alıcıda alınan sinyali işlemek ve frekans alanına getirmek için bir FFT bloğu kullanılmaktadır. FFT çıktısı vericide IFFT'ye gönderilen orijinal fazörlerdir. Kompleks düzlemde çizildiğinde, FFT çıktı örnekleri 16-QAM gibi bir kümeleşme diyagramı oluşturacaktır. Bununla birlikte, zaman-alan sinyali için bir kümeleşme gösterimi mevcut değildir. Kompleks düzlemde çizildiğinde, zaman-alanı sinyali normal şekli olmayan bir dağılım grafiği oluşturmaktadır. Böylece, bir kümeleşme kavramını kullanan herhangi bir alıcı işlemi (sembol dilimlemesi gibi) frekans alanında meydana gelmek zorundadır. Şekil 3.3'te gösterildiği gibi blok diyagram, bir OFDM sisteminde frekans-alan ve zaman alanı arasındaki geçişi göstermektedir.



Şekil 3.3. Her kanalında 16-QAM kullanan 4 alt banta sahip OFDM üretimi (Seke, 2017).

3.1.1. Dikgen Frekans Bölmeli Çoğullama Avantaj ve Dezavantajları

OFDM sisteminin kablosuz haberleşme sistemleri için birçok avantajı bulunmaktadır:

- OFDM modülasyon tekniği birbiri üzerine örtüşen alt taşıyıcılar sayesinde spektrumun etkin şekilde kullanımını sağlamaktadır.
- FFT/ IFFT tekniklerinin kullanımıyla modülasyon ve demodülasyon uygulamaları kolay ve hızlı uygulanabilmektedir.
- OFDM'nin hesaplama karmaşıklığı $O(B \log B T_m)$ olarak gösterilebilir, burada B bant genişliği ve T_m gecikme yayılımıdır. Bu karmaşıklık, $O(B^2 T_m)$ karmaşıklığına sahip standart bir frekans düzenleyici sistemden çok daha düşüktür.
- Bilgi alt taşıyıcılara bölünerek iletiğinden dar bant çentik sönmömlerine karşı tek taşıyıcılı sistemlere göre oldukça dirençlidir.
- Koruyucu ön ek sayesinde semboller arası etkileşimi en aza indirir.

- Derin sönümlemelere maruz kalmış iletilen spektrumun bölümlerinin neden olduğu hata zincirine karşı sağlamlık sağlayabilen, frekans alanında alt taşıyıcılar arasında kodlamayı ve ara eklemeleri (interleaving) kolaylaştırmaktadır. Kanal kodlaması ve serpiştirme kullanımı ile kanalın frekans seçiciliği nedeniyle kaybedilen sembolleri kurtarabilmektedir.

- OFDM, sistemin mevcut kanal koşullarından en iyi şekilde yararlanmasını sağlayan adaptif modülasyon ve kodlama için çok uygundur.

- Kanal dengeleme (equalization) adaptif dengeleme kullanan tek taşıyıcılı sistemlere göre daha kolaydır.

- Sembol süreleri kanalın evre uyumluluk süresine oranla kısa olduğu için yavaş sönümlmeye uğramakta ve sistemin bit hata oranını iyileştirmektedir.

- Alt taşıyıcıların birbirine dik olmasıyla taşıyıcılar arası girişimi (ICI: Inter Carrier interference) engellerken iletim bandının daha verimli kullanılmasını sağlamaktadır.

- Gürültüye karşı iyi koruma sağlamaktadır.

- OFDM, farklı formların birden çok kullanıcı arasında bölünmüş olduğu çoklu erişim şeması olarak kullanılabilir. Bu şemaya OFDMA denir ve mobil WiMAX'tan yararlanılmaktadır. Bu şema aynı zamanda kanal tahsisinde ince ayrıntılar verme olanağı sunmaktadır. Nispeten yavaş zaman ayarlı kanallarda, belirli alt taşıyıcının sinyal-gürültü oranına göre kullanıcı başına veri oranını uyarlayarak kapasiteyi önemli ölçüde arttırmak mümkündür.

- OFDM, dar bant girişimine karşı nispeten sağlamdır çünkü bu tür girişim, alt taşıyıcıların yalnızca bir bölümünü etkilemektedir.

Bu avantajlara rağmen, OFDM teknikleri de çeşitli zorluklarla karşılaşmaktadır:

- OFDM sinyalinin ortalamasının çok üstüne çıkması (zirve yapması) durumunda doğrusallıktan çıkma ve kırılma gibi problemler ortaya çıkmaktadır. Bu durum karşı konulması gereken güç verimsizliklerine neden olabilmektedir.

- OFDM sinyali çok geniş dinamik aralığa sahip bir genliğe benzer bir gürültüye sahiptir, bu nedenle yüksek tepe-ortalama güç oranına (PAPR) sahip RF güç amplifikatörleri gerektirmektedir.

- OFDM sinyallerinin faz gürültüsü ve frekans dağılımına duyarlı olması nedeniyle tasarımın bu kusurlarının hafifletmesi gerekmektedir. Mobil uygulamalarda çok yollu alışı

(multipath) faz kaymasına ve doppler etkisi de frekans kaymasına sebep olmaktadır. Bunun yanında, çok yollu alısta bazı frekanslar kaybolabilmektedir (fading). Faz ve frekans kaymalarına karşı bazı önlemler alınmıştır. Bunlardan ilki, alıcıların referans olarak kullanabileceği bir işaret (bilgi taşımayan sinüs işareti) kullanılması, diğeri ise IDFT ile hesaplanan işaret gönderildikten sonra alıcı tarafından kullanılmayacak işaretlerin atılması ve bu blokların arasını açmak için kullanılan ilave veri gönderilmesi oldukça yaygındır. Rastgele veri göndermek yerine döngüsel genişletme (cyclic extension) kullanmak dalga formunun devamlılığını sağladığından ve taşıyıcı dikliğini bozmadığından tercih edilmektedir.

- Alıcı ve verici arasındaki frekans ve zaman senkronizasyonuna duyarlı bir sistemdir. Taşıyıcı frekanslar arasındaki fark (Carrier Frequency Offset) alt taşıyıcılar arası girişime (ICI) neden olmaktadır.

3.1.2. Dikgen Frekans Bölmeli Çoğullama Uygulama ve Standartları

Dikgen Frekans Bölmeli Çoğullama tekniği LTE, WIMAX, WLAN, DAB, DVB-T ve Asenkron Sayısal Abone Hattı (ADSL) gibi birçok haberleşme sisteminin alt yapısında standart olarak kullanılmaktadır. Bazı standartların parametreleri Çizelge 3.1, Çizelge 3.2 ve Çizelge 3.3'te verilmektedir (Fazel ve Kaiser, 2008).

Çizelge 3.1. Yayın standartları.

	DAB			DVB-T	
Bant Genişliği	1.5 MHz			8 MHz	
Alt Taşıyıcı Sayısı	256 FFT	512 FFT	2k FFT	2 k FFT	8 k FFT
Sembol Süresi	125 us	250 us	1 ms	224 us	896 us
Modülasyon Tipi	D-QPSK			QPSK, 16-QAM, 64-QAM	
Maksimum Veri Hızı	1.7 Mbit/s			31.7 Mbit/s	

Çizelge 3.2. Kablosuz Yerel Alan Ağı (WLAN) standardı.

	IEEE 802.11a
Bant Genişliği	20 MHz
Alt Taşıyıcı Sayısı	64 FFT
Sembol Süresi	4 us
Modülasyon Tipi	BPSK, QPSK, 16-QAM, 64-QAM
Maksimum Veri Hızı	54 Mbit/s

Çizelge 3.3. Kablosuz Yerel Döngü (WLL) standartları.

	IEEE 802.16d, ETSI HIPERMAN	
Bant Genişliği	1.5-28 MHz	
Alt Taşıyıcı Sayısı	256 FFT (OFDM)	2048 FFT (OFDMA)
Sembol Süresi	8-125 us	64-1024 us
Modülasyon Tipi	QPSK, 16-QAM, 64-QAM	
Maksimum Veri Hızı	26 Mbit/s'e kadar (7 MHz'lik kanal kapasitesinde)	

3.2. Ayırık Fourier Dönüşümü ve Hızlı Fourier Dönüşümü Algoritmaları

Kesikli Fourier dönüşümü (DFT) için hızlı algoritmaların geliştirilmesi 1805'te Carl Friedrich Gauss'un 'Pallas' ve 'Juno' asteroitlerin yörüngesini birbirinden ayırmak için yapmış olduğu gözlemlerin olduğu yayımlanmamış çalışmaları ile başlamıştır (Gauss,1866; Heideman vd., 1985). Yöntem, genel olarak modern Hızlı Fourier Dönüşümü (FFT) algoritmasının icadı için kullanılan, James Cooley ve John Tukey tarafından 1965'te yayınlanan yöntemle çok benzerdir. Yöntem üzerine devam eden Cooley ve Tukey'in çalışmaları ile FFT algoritmaları sayısal sinyal işleme uygulamalarında kullanılmaya ve geliştirilmeye başlanmıştır.

DFT, sürekli işaretlerin sonsuz sayıda farklı faz açısındaki sinüzoidallerin toplamı cinsinden ifade edilmesini sağlamak ve bu sinüzoidallerin toplamdaki ağırlıklarını vermektedir. Ağırlıklar işaretin sinüzoidallerle korelasyonu (ya da evrişimi) ile elde edilmektedir. Benzer şekilde Ayırık Fourier dönüşümü de periyodik olduğu varsayılan ayırık işaretlerin ayırık frekanslı ayırık sinüzoidallerin toplamı şeklinde ifade edilmesini

sağlamaktadır. Sinüzoidallerin toplamdaki ağırlıkları, ayrık korelasyon (ya da evirişim) ile elde edilmektedir. Bu ağırlıkları bulma işlemine DFT, ağırlıklar yerine konularak asıl işareti ifade etme işlemine de Ters Ayrık Fourier Dönüşümü (IDFT) denir. Bu dönüşüm, ayrık işaret işleme algoritma ve sistemlerinin analizi, tasarımı, gerçekleştirilmesi ile doğrusal filtreleme, korelasyon analizi ve spektrum analizi gibi uygulamalarda önemli bir rol oynamaktadır.

Basit formdaki DFT, N uzunluktaki x girdi vektörü için N uzunlukta bir çıkış vektörü üretmektedir (3.1).

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{nk}, k = 0, \dots, N-1 \quad (3.1)$$

$W_N = e^{-2\pi j/N}$ temel frekanslı bileşen olmak üzere, W_N^{nk} dönüşümün çekirdeği (kernel) olarak adlandırılmaktadır. Bunlar karmaşık (faz farklı) sinüzoidallerdir. Her bir X_k 'nin hesaplamasında N adet karmaşık çarpıcı ve $N-1$ karmaşık toplayıcı kullanılmaktadır. Böylece N değerli DFT için N^2 karmaşık çarpma ve $N^2 - N$ toplama işlemi gerekmektedir. Çekirdeğin simetri (3.2) ve periyodiklik (3.3) özelliklerinden faydalanılarak daha az sayıda hesap kullanan algoritmalar geliştirilmiştir.

$$\text{Simetri özelliği:} \quad W_N^{k+N/2} = -W_N^k \quad (3.2)$$

$$\text{Periyodiklik Özelliği:} \quad W_N^{k+N} = W_N^k \quad (3.3)$$

DFT $O(N^2)$ algoritmik karmaşıklığa sahip olduğu için doğrudan korelasyon (3.1) yapmak ölçeklenebilir değildir, N büyüdükçe işlem sayısı çok fazla artar. DFT hesaplamaları için verimli algoritmaların geliştirilmesi, bir böl-ve-yönet (divide-and-conquer) yaklaşımını benimsemek durumunda mümkündür. Bu yaklaşım, bir N -nokta DFT'nin ardışık olarak daha küçük DFT'lere ayrılmasına dayanmaktadır. Böylece, denklem (3.1)'den daha düşük karmaşıklığa sahip, FFT algoritmalarına ulaşılmıştır. Adından da anlaşılacağı üzere, FFT bir ayrık Fourier dönüşümünün hızlı bir şekilde hesaplanması için

kullanılan algoritmalarla verilen isimdir. DFT'den farklı bir dönüşüm değildir. FFT, N girdi veri sayısı olmak üzere karmaşıklığı $O(N^2)$ 'den $O(N \log N)$ 'e kadar azaltan bir DFT algoritmasıdır. Logaritma işleminin tabanı, her bir aşamada DFT işleminin kaç bölündüğüdür (Proakis ve Manolakis, 2007) (örneğin N -nokta DFT 2 adet $N/2$ -nokta DFT ile gerçekleştiriliyor ise $\log_2 N$ 'dir).

3.2.1. Cooley-Tukey Algoritmaları

En yaygın olarak kullanılan FFT algoritmalarının başında Cooley-Tukey algoritması gelmektedir. Algoritmada, $N = N_1 N_2$ büyüklüğündeki bir DFT'yi, geleneksel olarak sabit çarpanlar (twiddle) faktörleri olarak adlandırılan karmaşık çarpanların çarpımları ile N_1 ve N_2 boyutlarında daha küçük DFT'lere yinelemeli olarak ayıran bir böl-ve-yönet yaklaşımı algoritmasıdır. En çok bilinen Cooley-Tukey algoritmaları, DFT büyüklüğünün $N = 2^p$ veya $N = 4^p$ (p doğal sayı olmak üzere) şeklinde sınırlandırılması ve böl-ve-yönet yaklaşımıyla radiks-2 FFT ve radiks-4 FFT ve bunların birlikte kullanımıyla gerçekleştirilen karma-radiks (mixed-radix) veya varyantı olan bölünmüş-radiks (split-radix)'tir.

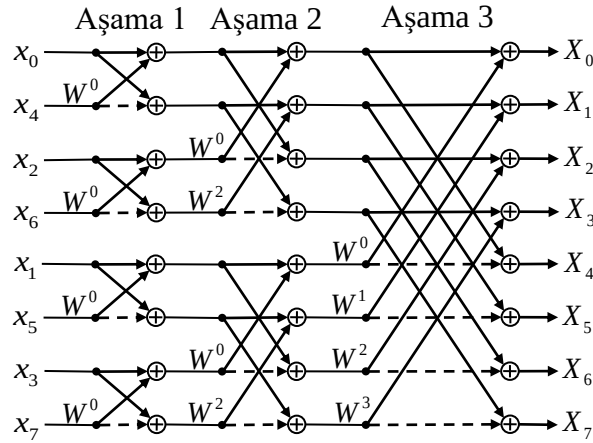
3.2.1.1. Radiks-2 FFT Algoritması

Cooley ve Tukey'in önerdiği (Cooley ve Tukey, 1965) radiks-2 algoritması böl-ve-yönet yaklaşımının ilk uygulandığı oldukça yaygın FFT algoritmasıdır. N -noktalı veri çift ve tek numaralı örnekler olmak üzere 2 adet $N/2$ -nokta dönüşümüne özyinelemeli olarak ayrılmaktadır denklem (3.4). $W_N^2 = W_{N/2}$ eşitliğinden faydalanılarak X_k denklem (3.5)'teki gibi ifade edilebilmektedir. Burada $F_1(k)$ ve $F_2(k)$, $f_1(m)$ ve $f_2(m)$ dizilerinin $N/2$ -noktalı Ayırık Fourier Dönüşümüdür. Bu ayrışma işlemi $N = 2^p$ için, $p = \log_2 N$ sayıda yani 1-nokta dönüşümü elde edilinceye kadar uygulanmaktadır. Toplamda karmaşık çarpma işlemi $N/2 \log_2 N$ 'e, toplama işlemi sayısı ise $N \log_2 N$ 'e kadar azaltılmaktadır.

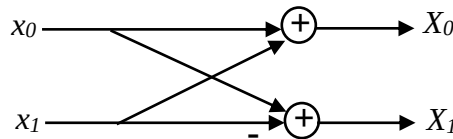
$$X_k = \sum_{m=0}^{N/2-1} x_{2m} W_N^{2mk} + \sum_{m=0}^{N/2-1} x_{2m+1} W_N^{k(2m+1)} \quad (3.4)$$

$$\begin{aligned} X_k &= \sum_{m=0}^{N/2-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{N/2-1} f_2(m) W_{N/2}^{km} \\ &= F_1(k) + W_N^k F_2(k) \quad k = 0, 1, \dots, N-1 \end{aligned} \quad (3.5)$$

Decimation in Time (DIT) ve Decimation in Frequency (DIF) algoritmaları olarak adlandırılan iki farklı radiks-2 FFT algoritması bulunmaktadır. Örneğin $N = 8$ -noktalı ($N=2^3$) DFT, DIT FFT hesaplaması ilk aşamada 4 adet 2-noktalı, ikinci aşamada 2 adet 4-noktalı ve son aşamada 1 adet 8-noktalı dönüşüm olmak üzere toplamada 3 aşamada ifade edilebilmektedir (Şekil 3.4). Her bir aşama $N/2$ adet kelebekten olarak adlandırılan yapılardan oluşmaktadır. Bu bloklar N -noktalı dönüşümü, r radiks olmak üzere, N/r -noktalı dönüşüm haline getirmektedir (Şekil 3.5).



Şekil 3.4. Radiks-2 kelebek yapısıyla 8- noktalı DIT FFT.

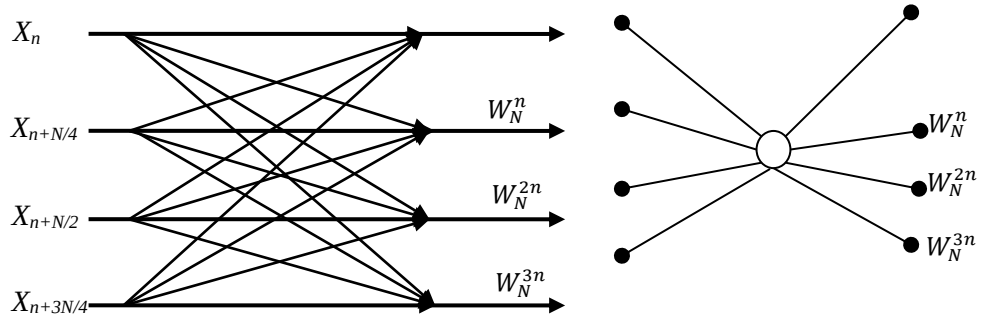


Şekil 3.5. 2- noktalı FFT hesaplayan kelebek yapısı.

3.2.1.2. Radiks-4 FFT Algoritması

Radiks-4 algoritması $N = 4^p$ -noktalı verinin radiks-2 algoritması kullanılarak tasarlanan 4-girişli/çıkışlı kelebek yapılarıyla hesaplanmasıdır. N -noktalı giriş 4 alt aşamaya bölünerek hesaplamalar yapılmaktadır (3.6). Radiks-4 FFT algoritması hesaplaması kelebek yapısı Şekil 3.6'da gösterilmektedir.

$$X_k = \sum_{m=0}^{N/4-1} x_{4m} W_N^{k(4m)} + \sum_{m=0}^{N/4-1} x_{4m+1} W_N^{k(4m+1)} + \sum_{m=0}^{N/4-1} x_{4m+2} W_N^{k(4m+2)} + \sum_{m=0}^{N/4-1} x_{4m+3} W_N^{k(4m+3)} \quad (3.6)$$



Şekil 3.6. Radiks-4 DIF FFT algoritmasının temel kelebek yapısı

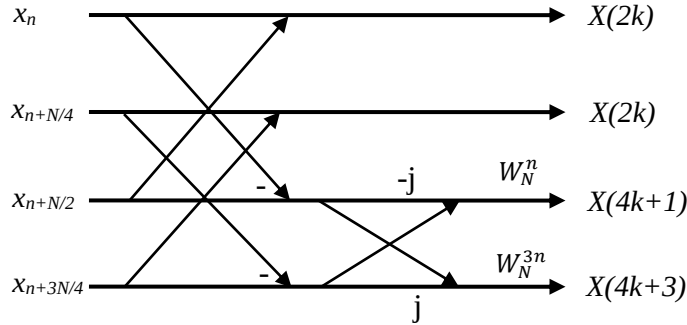
Radiks-4 FFT algoritması her bir aşama $N/4$ kelebekten oluşmak üzere, toplamda p kadar aşamadan oluşmaktadır. Algoritma için hesaplama yükü $3N/8 \log_2 N$ karmaşık çarpma ve $3N/2 \log_2 N$ karmaşık toplama işlemidir. Toplamda %25 oranında çarpıcı sayısında azalma görülmekte, ancak toplayıcı kullanım gereksinimi %50 oranında artmaktadır.

3.2.1.3. Bölünmüş-Radiks (Split-Radix) FFT Algoritması

Ayrılmış-radiks FFT, DFT hesaplamak için uygulanan ve ilk olarak R. Yavne (1968) tarafından bulunan FFT algoritmasıdır. Bölünmüş-radiks ismi 1984 yılında P. Duhamel ve H. Hollmann tarafından verilmiştir. Bu algoritma, Cooley-Tukey FFT algoritmasının radiks-2 ve radiks-4 algoritmalarının birlikte kullanılmasıyla ortaya çıkmıştır. $N/2$ uzunluğunda bir DFT ve $N/4$ uzunluğunda iki DFT'nin kullanımıyla N uzunluğunda bir DFT'nin iteratif uygulaması olarak tanımlanmaktadır. Algoritmada, çift numaralı örnekleri radiks-2 ile tekler ise radiks-4 ile hesaplanmaktadır (3.7).

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} W_N^{k2m} + \sum_{m=0}^{N/4-1} x_{4m+1} W_N^{k(4m+1)} + \sum_{m=0}^{N/4-1} x_{4m+3} W_N^{k(4m+3)} \quad (3.7)$$

Bölünmüş-radiks kelebek yapısı Şekil 3.7'de verilmektedir.



Şekil 3.7. Bölünmüş-radiks DIF kelebek yapısı.

3.2.2. Diğer Hızlı Fourier Dönüşümü Algoritmaları

Cooley ve Tukey'in önerdiği algoritmalar sayesinde FFT hesaplamalarında daha az sayıda çarpıcı ve toplayıcı kullanıldığı belirgindir. Radiks-2 FFT algoritması üzerine yapılan

araştırmalarda Winograd (WFTA) (Winograd, 1978) ve Prime Factor (PFA) (Burrus ve Eschenbacher, 1981) gibi algoritmaların önerildiği ve Cooley-Tukey algoritmasının kelebek bağlantılarının farklı olduğu sabit-geometri (Kwong ve Goel, 2014) yapısı görülmektedir.

3.2.2.1. Winograd Algoritması

Winograd algoritması (Winograd-1976), Cooley ve Tukey tarafından önerilen algoritmayla aynı sayıda toplayıcı kullanmaktadır, ancak çarpıcı sayısı oldukça azalmaktadır. Bu yöntemin avantajı, büyük- N algoritmaları için çarpma sayısının FFT'den daha az bir oranda (N ile) artış sağladığı ve toplayıcı sayısının FFT'de kullanılan sayıdan çok farklı olmamasıdır (birçok N değeri için toplayıcı sayısı aslında daha azdır). N değerinin asal sayıların çarpımı olması durumunda, hesaplama algoritması Good (1960) tarafından önerilmiştir (Good, 1958). Good'un algoritmasındaki asal sayı uzunluktaki kısımların FFT hesabını da konvolüsyon kullanılarak hesaplanması Winograd Fourier dönüşüm algoritması olarak gösterilmiştir. Ancak bu algoritma bilgisayara uygulandığında beklenen performansı sağlayamamış ve FFT karmaşıklığının yeniden tanımlanmasına yol açmıştır. Sadece çarpma sayısına bakmanın yeterli olmadığı aynı zamanda toplama ve hafıza transfer miktarlarına da bakmak gerektiği ortaya çıkmıştır (Morris, 1978).

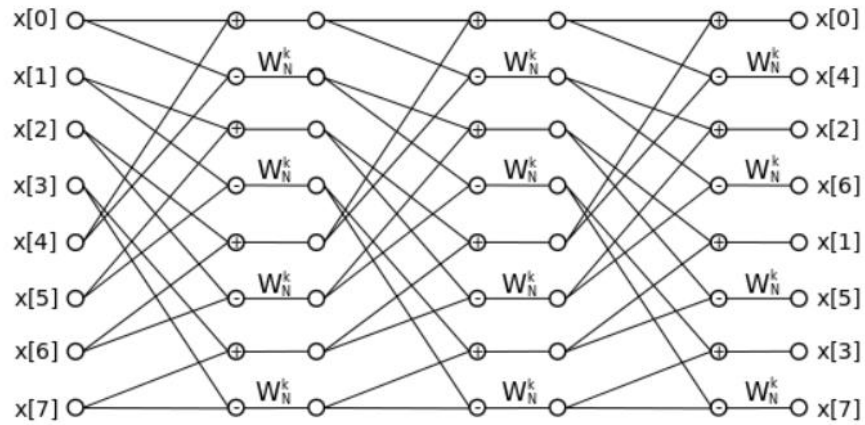
3.2.2.2. Prime Factor Algoritması

Prime Factor algoritması (Kolba ve Parks-1977), karmaşık çarpıcılara ihtiyaç duyulmadan küçük-nokta yapı blokları dizisi ile karakterize edilebilmektedir. Bu yapı blokları dizisi, dönüşüm uzunluğu N 'yi iki sayıya, $N = PQ$ faktörize ederek ve P ve Q noktalı FFT'lerine dayalı N -nokta dönüşümünü hesaplayarak geliştirilmiştir. P veya Q daha fazla faktörlü hale getirilebiliyorsa, örneğin $Q = RS$ olarak ifade edilebiliyorsa, bu durumda Q -noktası dönüşümü iki yapı taşından (R ve S noktası yapı blokları) oluşturulabilmektedir. Bu algoritmanın en büyük kısıtlaması, küçük-nokta yapı bloklarının nispeten asal olmasıdır. Bu durum, ortak faktörlü olamayacakları anlamına gelmektedir.

Algoritma için gerekli olan çarpıcı ve toplayıcı sayısı algoritma yapı blokları için gerekli olanların toplamıdır. Winograd algoritmasıyla karşılaştırılacak olursa, daha fazla çarpıcı, daha az toplayıcı kullanmaktadır, ancak FFT hesaplama hızı daha yüksektir (Burris ve Eschenbacher, 1981).

3.2.2.3. Sabit Geometri Yapısı

Sabit-geometri yapısı Cooley-Tukey'in önerdiği hesaplama sırası yapısının bir alternatifi olarak önerilmiştir. Bu yapıda, FFT hesaplamasının her basamağı özdeş bağlantı yapılarıyla gerçekleştirilmektedir. 8 örneklilik bir radix-2 Cooley-Tukey DIF algoritması sabit geometri yapısıyla Şekil 3.8'de gösterilmektedir.



Şekil 3.8. Sabit geometri DIF yapısı

Bu yapı yerinde-hesaplamaya uygun gibi görünse de, ara hesapların sonuçları henüz ihtiyaç olmayan (sırası gelmeyen) hafıza adreslerini bozacağından ek ihtiyacı doğurmaktadır.

3.3. Hızlı Fourier Dönüşümü İşlemci Yapılarının Sınıflandırılması

Hızlı Fourier Dönüşümü işlemci mimarileri bir kelebek işlem elemanı, saklama birimi olarak farklı büyüklüklerde RAM ve ROM, adres üreteç birimi ve sıralı kontrol birimlerinden oluşmaktadır. RAM, giriş verilerini, ara verileri ve çıkış verilerini saklamak için kullanılmaktadır. ROM'a ise çarpmada kullanılan dönüşümün çekirdeği (twiddle factors) verileri kaydedilmektedir. Adres üretme birimi, kelebek işlemleri verilerini okumak için ve çıktı verileri sonuçlarını RAM'e depolamak için adres üretmektedir. Ardışık kontrol ünitesi ise her birim için kontrol sinyalleri üretmektedir.

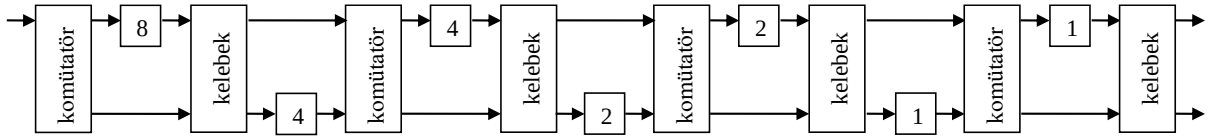
Gerçek zamanlı uygulamalarda hafıza-tabanlı, önbellek hafıza-tabanlı, ardışık, paralel, paralel yinelemeli, dizi mimari ve ardışık düzen (pipeline) olmak üzere çok çeşitli FFT hesaplama algoritmaları bulunmaktadır (Joshi, 2015). Bunlardan en önemli ve en çok tercih edilen iki tanesi ardışık düzen ve hafıza-tabanlı FFT yapılarıdır.

3.3.1. Ardışık Düzen FFT Yapıları

Ardışık düzen FFT yapısı 1970'lerin başında henüz Çok Yüksek Boyutlu Entegrasyon (VLSI) teknolojisi sistem entegrasyon seviyesine gelmeden önce, radar sinyal işleme gibi gerçek zamanlı işaret işleme uygulamaları üzerine doğan ihtiyaçtan dolayı yoğun bir araştırma konusu haline gelmiştir. Ardışık düzen FFT, FFT'yi sıralı bir şekilde hesaplayabilen özel bir algoritmadır. Verinin işlemciden sürekli alınmasıyla gerçek zamanlı uygulamaları başarıyla gerçeklemektedir. Radiks-2 Çoklu-Yol Gecikmeli Komütatör (R2MDC), Radiks-2 Tekli-Yol Gecikmeli Geribildirim (R2SDF), Radiks-4 Çoklu-Yol Gecikmeli Komütatör (R4MDC), Radiks-4 Tekli-Yol Gecikmeli Komütatör (R4SDC) ve Radiks-4 Tekli-Yol Gecikmeli Geribildirim (R4SDF) gibi farklı yaklaşımlara dayanan çeşitli mimariler gerçekleştirilmiştir. Bu mimarilerin her biri paralel veya seri olarak sınıflandırılmakta ve birbirlerine göre avantaj ve dezavantajları bulunmaktadır (He ve Torkelson, 1996).

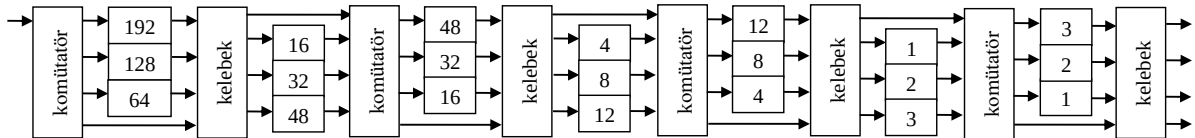
3.3.1.1. Paralel-Giriş Paralel-Çıkış Ardışık Düzen FFT Yapıları

R2MDC: Radiks-2 Çoklu-Yol Gecikmeli Komütatör radiks-2 ardışık düzen FFT uygulaması için en basit yaklaşımlardan biridir. Girişteki veri kelebek yapısının giriş değerleri arasındaki uzaklık gözetilerek ikiye bölünmektedir (Şekil 3.9). Böylece hem kelebek hem de çarpıcı kullanımını %50 oranında azaltmaktadır. $\log_2 N - 1$ çarpıcı, $\log_2 N$ radiks-2 kelebek ve $3/2N - 2$ kaydediciye ihtiyaç duyulmaktadır.



Şekil 3.9. R2MDC yapısı.

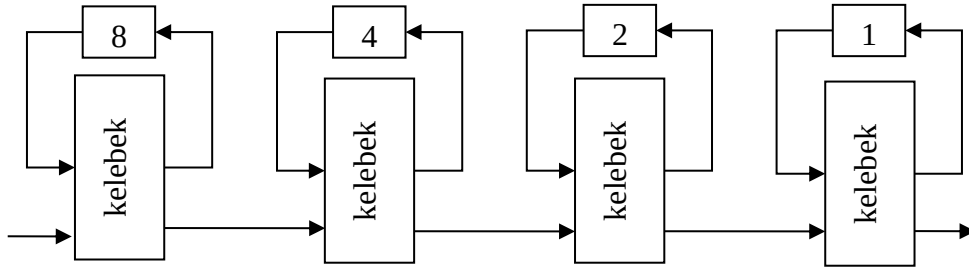
R4MDC: Radiks-4 Çoklu-Yol Gecikmeli Komütatör R2MDC'nin radiks-4 sürümüdür (Şekil 3.10). Bu yapı ilk VLSI uygulamalarında ardışık düzen FFT yapısı için kullanılmıştır. Radiks-4 FFT için ihtiyaç duyulan devre elemanlarının yalnızca %25'ini kullanmakta, bu da ancak 4 FFT'nin eş zamanlı olarak işlendiği bazı özel uygulamalarda mümkün olabilmektedir. $3\log_4 N - 1$ çarpıcı, $\log_4 N$ radiks-4 kelebek ve $5/2N - 4$ kaydedici gereksinimi vardır.



Şekil 3.10. R4MDC yapısı.

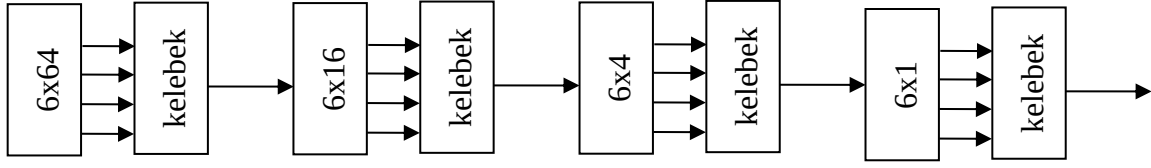
3.3.1.2. Seri-Giriş Seri-Cıkış Ardışık Düzen FFT Yapıları

R2SDF: Radiks-2 Tekli-Yol Gecikmeli Geribildirim yapısı (Şekil 3.11) kelebek çıkışındaki verileri geri besleme ile kaydedicilere depolayarak kelebekleri daha verimli kullanmayı hedeflemektedir. Tüm veriler her aşamada bir çarpanla çarpılmaktadır. Yapıda geciktirici elemanların sayısını azaltmak amacıyla çoklayıcı kullanılmaktadır. Çoklayıcılar Şekil 3.11'deki çizimde kelebek bloklarının içindedir. Çoklayıcı '0' olduğunda kelebek yapısı işlem yapmamakta ve veriler geçmektedir, '1' durumunda ise kelebeğe gelen veriler işlenmektedir. Böylece geri besleme için ihtiyaç duyulan geciktirici eleman sayısı $3N/2$ 'den $N - 1$ 'e inmektedir. R2MDC yapısıyla aynı sayıda kelebek ve çarpıcı kullanılmaktadır.



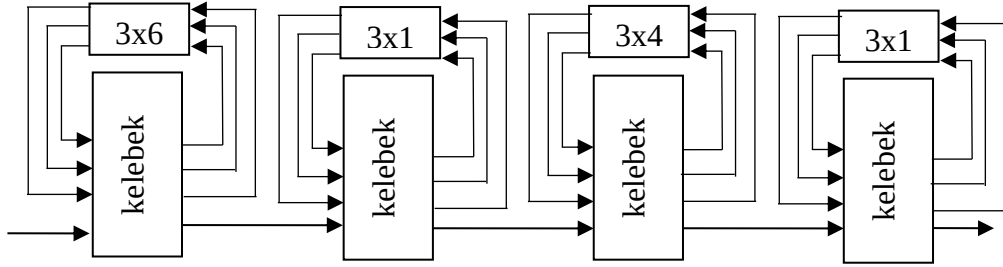
Şekil 3.11. R2SDF yapısı.

R4SDC: Radiks-4 Tekli-Yol Gecikmeli Komütatör yapısı (Şekil 3.12) $1/4$ oranında kullandığı programlanabilir kelebek yapısıyla radiks-4 algoritmasının geliştirilmiş halidir. Çarpıcı sayısı $\log_4 N - 1$ olmakla birlikte R4MDC'de ihtiyaç duyulan $2N - 2$ hafıza miktarını $5/2N - 1$ oranına düşürmektedir. Programlanabilirlik özellikleri nedeniyle kelebek ve geciktirme komütatörü nispeten karmaşık hale gelmektedir. R4SDC son zamanlarda HDTV uygulamalarında ve bu zamana kadar yapılan en büyük tek çipli ardışık düzen FFT işlemcisinde kullanılmıştır.



Şekil 3.12. R4SDC yapısı.

R4SDF: Radiks-4 Tekili-Yol Gecikmeli Geri bildirim, R2SDF yapısının CORDIC iterasyonlarını kullanılarak uygulandığı versiyonudur (Şekil 3.13). Radiks-4 algoritması kullanımından dolayı R2SDF yapısında ihtiyaç duyulan $\log_2 N - 2$ çarpıcı sayısı bu algorithmada $\log_4 N - 1$ 'e düşmektedir. Bununla birlikte, oldukça karmaşık olan ve en az 8 karmaşık toplayıcı içeren radiks-4 kelebeğinin kullanım oranı R2SDF'nin yalnızca %25'i kadar ve daha karmaşık yapıdadır. R2SDF'ye benzer olarak çoklayıcı kullanımı mevcuttur. Çoklayıcı '1' olduğunda, veriler işlenmesi için kelebeğe gönderilmekte, aksi halde veriler $3N/4$ uzunluğundaki gecikme yoluna (ilk aşamada) gönderilmektedir. $\log_4 N - 1$ çarpıcı, $\log_4 N$ radiks-4 kelebek ve $N - 1$ hafıza kullanımı bulunmaktadır.

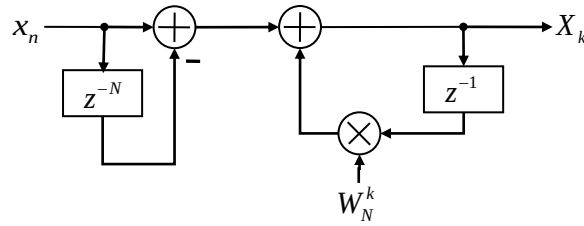


Şekil 3.13. R4SDF yapısı.

3.3.1.3. Seri-Giriş Paralel-Çıkış Ardışık Düzen FFT Yapıları

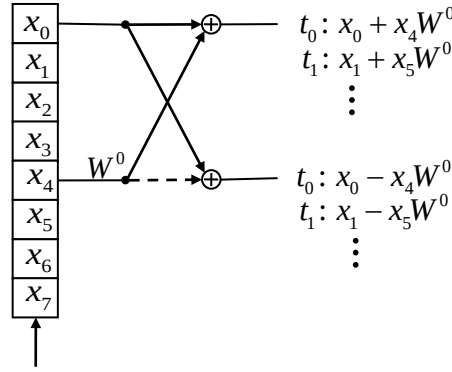
Seri-Giriş Paralel-Çıkış ardışık düzen FFT yapısında giriş verisi seri olarak alınmaktadır. Seri haldeki her bir giriş verisi için FFT katsayısı hesaplanmakta ve N kadar veri ile çarpılarak sonuç çıkışa verilmektedir. Giriş verilerini seri hale getirmek için Goertzel

algoritması (Şekil 3.14) adı verilen ardışık düzen kayan pencere yapısıyla çıkış verileri girişe göre güncellenmektedir (Jacobsen ve Lyons, 2003). Bu tür yaklaşımların gereksinimleri karşılamasına rağmen, çok fazla hesaplamalara yol açtığı, verimliliği azalttığı ve gereksiz yer kullandığı kolaylıkla fark edilebilmektedir. Her bir aşamada ihtiyaç duyulan hafıza miktarı $N/2$ olmakta, $N \log_2 N$ aşama olduğu düşünüldüğünde, toplamda $N \log_2 N / 2$ hafızaya ihtiyaç duyulmaktadır. Kullanılan kelebek sayısı yarıya inmekte, $N - 1$ adet çarpıcı ve $2(N - 1)$ adet toplayıcı kullanılmaktadır. Goertzel algoritması her bir FFT katsayısı için ayrı devre kullandığından sadece sınırlı sayıda FFT katsayısına ihtiyaç duyulduğunda (örneğin 1024-nokta FFT için 10 adet) eleman sayısı açısından verimli olmaktadır.



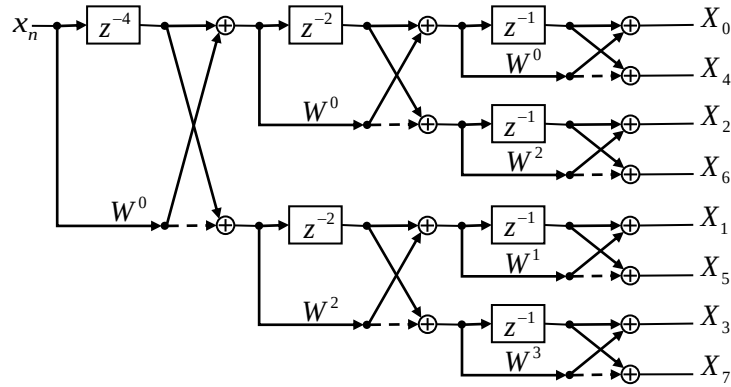
Şekil 3.14. Goertzel algoritması yapısı.

Giriş verileri Şekil 3.15'te görüldüğü üzere seri hale getirildiğinde, $N/2$ zamanda bir yinelenen kelebek yapılarının olduğu görülmektedir. Tüm hesaplanan sonuçlar bir sonraki FFT hesaplaması için gerekli olmaktadır ve bu nedenle hafıza bloklarına geçici olarak saklanması gerekmektedir. Radiks-2 DIT FFT yapısı düşünüldüğünde, kelebeğin 2. aşaması için $N/4$ kadar yapılan toplama ve çıkarma işlemlerinin benzer olduğu görülmektedir.



Şekil 3.15. Seri veri girişinde kelebek yapısının tekrarlanan hesaplama yapısı.

Aynı yaklaşım kelebeğin diğer aşamalarına da uygulandığında Şekil 3.16'daki yapı elde edilmektedir.



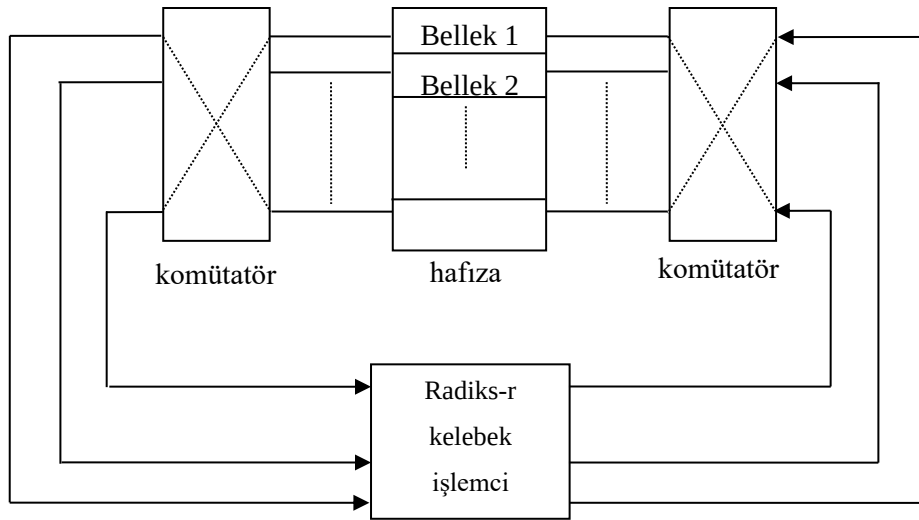
Şekil 3.16. $N=8$ veri için seri-giriş paralel-çıkış FFT pencere metodu (Seke ve Kaya, 2017).

Bu uygulama seri-giriş seri-çıkış ardışık düzen FFT algoritmalarıyla doğrudan karşılaştırılmamalıdır, çünkü bu yaklaşımlar N veri seti için N kadar seri çıktılar üretmektedir. Öte yandan buradaki yaklaşım, girilen kayan pencere içindeki mevcut veriler için eşzamanlı olarak tüm çıkışları sağlamakta ve gelen her örnek için tüm çıktıları hesaplamaktadır. Çoklayıcı ihtiyacı olmayan Seri-giriş Paralel-çıkış ardışık düzen FFT yapısı hızlı spektrum analizi ihtiyacında kullanılabilir.

3.3.2. Hafıza-tabanlı FFT Yapıları

Hafıza-tabanlı yapılar her basamakta işlem sonuçlarını yine hafızaya kaydetmektedirler. Bu mimariler genellikle bir ya da daha fazla işlemci elemanı, hesaplamaya bağlı kelebek, hafıza blokları ve kontrol ünitelerinden oluşmaktadır. Hafıza-tabanlı yapılar, ardışık düzen yapılarının aksine yalnızca bir kelebek işlemcisi ile de kullanabilmektedir. Bu özellikleriyle donanım maliyetini oldukça düşürmektedirler. Ancak, tek bir kelebek işlemci elemanı her bir saat darbesinde bir hesap yapmakta, tüm sistemin işlem hızını azaltmaktadır. Bu dezavantaj yüksek radiks algoritmaları kullanılarak giderilebilmektedir. Sonuç olarak, sistem saat darbesinin sıklığını artırarak istenilen hızlara ulaşılabilmektedir.

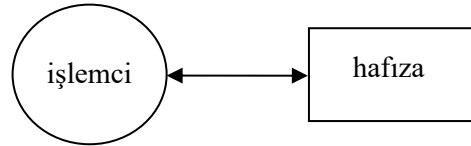
Kelebek işlemcisinin yanı sıra hafıza-tabanlı mimariler ana hafıza, bir adres üretici ve komütatörlerden oluşmaktadır (Şekil 3.17). Kelebek giriş ve çıkışları ana hafızayla bağlanmaktadır. Komütatör, hafızadan veri okunduktan ve geri hafızaya yazıldıktan sonra hesaplanan FFT değerlerini doğru düzene sokmak için kullanılmaktadır.



Şekil 3.17. Hafıza-tabanlı FFT yapısı.

3.3.2.1. Tek Hafıza Bloğu-Tabanlı FFT İşlemci Metodu

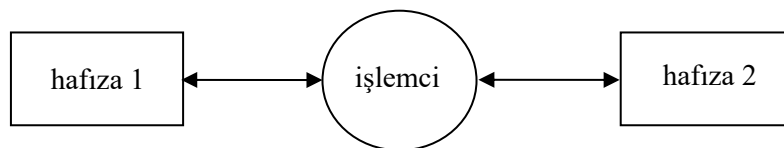
Tek-hafıza tabanlı FFT mimarileri hafıza-tabanlı yapıların en basitidir. Yalnızca bir hafıza bloğuna sahip olan bu yapılar, yeni FFT işlem verilerinin, önceki FFT işlem verileri tarafından kullanılan, aynı yerde saklandığı bir bellek adresleme stratejisi ile çalışmaktadır. Veri alışverişi her aşamada işlemci ve hafıza arasındaki çift yönlü veri iletimi yoluyla sağlanmaktadır. Tek-hafıza tabanlı FFT mimarisi blok diyagramı Şekil 3.18’de gösterilmektedir.



Şekil 3.18. Tek hafıza-tabanlı FFT mimari yapısı.

3.3.2.2. Çift Hafıza-Tabanlı FFT İşlemci Metodu

Çift hafıza-tabanlı FFT mimarilerinde, her iki hafıza da işlemci elemanına iki ayrı çift yönlü veri iletim yoluyla bağlanmaktadır. Hafızaların her biri kelebek girişi ile kelebek çıkışı arasında işlevlerini değiştirmektedir. Veri girişi bir hafızadan diğerine işlemci elemanı aracılığıyla iletilmekte, dönüşüm tamamlanana kadar da bu işlem ve tersi devam etmektedir. Bu, giriş verilerinin hafızadan okunması ve çıkış sonuçlarını aynı anda hafızaya yazması anlamına gelmektedir. Şekil 3.19, çift bellek tabanlı mimarinin şematik diyagramıdır.



Şekil 3.19. Çift hafıza-tabanlı FFT mimarisi blok diyagramı.

3.3.2.3. Önbellek-Tabanlı FFT İşlemci Metodu

Şekil 3.20’de şematiği gösterilen önbellek-tabanlı mimari bir işlemci, küçük bir önbellek ve bir ana bellekten oluşmaktadır. Çalışma mekanizması olarak tek bellek mimarisine benzemektedir, ancak fazladan bir önbelleği bulunmaktadır. Bu mimarinin kullanım amacı, ana belleğe erişim hızını artırmak böylece enerji verimliliğini artırmak ve FFT uzunluğunun fazla olması durumunda güç tüketimini azaltmaktır. Fazladan donanım ve kontrol karmaşıklığı sebebiyle çok fazla tercih edilmemektedir.



Şekil 3.20. Önbellek-tabanlı FFT mimarisi blok diyagramı

3.3.2.4. Radiks-2 Hafıza-Tabanlı FFT Yapıları

Radiks-2 kelebek yapısına dayalı uygulanan hafıza-tabanlı FFT yapıları yerinde kullanılabilir (in-place) hafıza adreslemesi ve hafıza erişimi verimini artırabilmek için hafıza bankaları yapısı kullanmaktadır. Ana hafızayı 2 tane $N/2$ yerleşim bankasına bölmede kullanılan, yalnızca radiks-2 algoritması için adres üretici ihtiyacı bulunmaktadır. Yerinde kullanılabilir hafıza adresleme yapısında kelebek çıkışındaki, hafızadan okunan ve aynı anda aynı hafıza adresine yazılan veriler kaydedilmektedir. Bu uygulamayla N -noktalı verinin okuma/yazma esnasında karışıklık yaşamadan r bankalı hafızaya yazılması işlemi gerçekleştirilmektedir. Karmaşık çarpıcı sayısı 1, hesaplama döngüsü sayısı ise $N/2 \log_2 N$ ’dir.

3.3.2.5. Radiks-4 Hafıza-Tabanlı FFT Yapıları

Yüksek hızlara ulaşmak için radiks-4 algoritması kullanılmaktadır. Radiks-2 algoritmasına benzer olarak yerinde kullanılabilir hafıza yapılarını kullanmaktadır, ancak adres üretici ve kontrol birimleri çalışma prensipleri farklıdır. İşlemci elemanı içinde karmaşık çarpıcı sayısı 3, hesaplama döngüsü sayısı ise $N/4\log_4 N$ 'dir.



4. ÖNERİLEN YÖNTEMLER

4.1. Radiks-2 Hızlı Fourier Dönüşümü Algoritması

Ayrık Fourier dönüşümü (DFT) sinyal işlemede temel ve sıklıkla kullanılan yöntemlerden biridir. DFT, devreye karmaşık sayılı çarpanlar (denklem (4.1)) ile girişteki her bir verinin çarpılıp toplanmasıyla hesaplanmaktadır (denklem (4.2)).

$$W_N^{nk} = \exp(j2\pi nk/N) \quad (4.1)$$

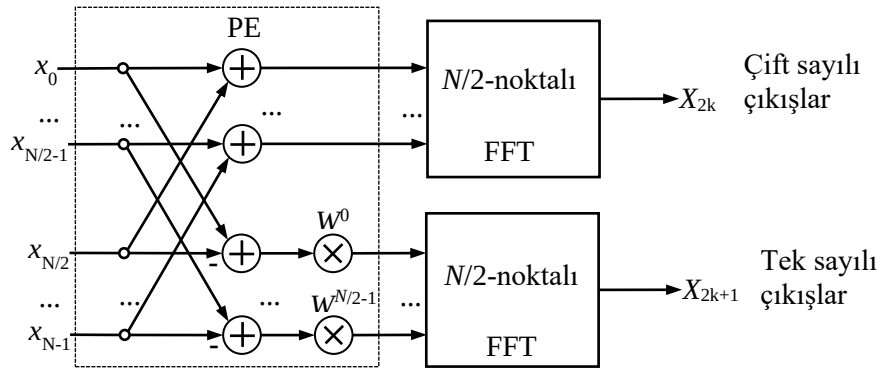
$$X_k = \sum_{n=0}^{N-1} x_n W_N^{nk}, k = 0, \dots, N-1 \quad (4.2)$$

Cooley ve Tukey (Cooley ve Tukey, 1965) DFT’de N^2 olan hesaplama karmaşıklığını $N/2 \log_2 N$ ’e indiren bir Hızlı Fourier Dönüşümü yöntemi sunmaktadır. Bu yöntem ile hafıza-tabanlı yapılardan ardışık düzen yapılara kadar birçok yazılım ve donanım uygulamaları için hesaplama algoritmaları geliştirilmiş ve uygulanmıştır.

Radiks-2 algoritması, N -noktalı DFT’yi iki adet $N/2$ -noktalı DFT’ye bölmekte ve 2-noktalı DFT hesaplaması yapılana kadar bölme işlemi devam etmektedir (denklem (4.3)).

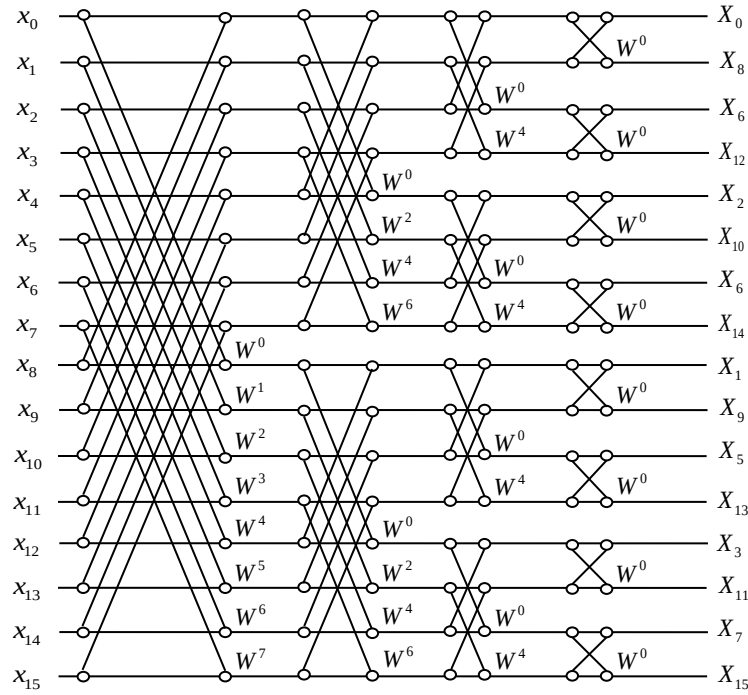
$$X_k = \sum_{m=0}^{N/2-1} x_{2m} W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} x_{2m+1} W_{N/2}^{mk} \quad (4.3)$$

Denklem (4.3)’e göre FFT bölme işlemleri yapılmış ve verilerin çıkışa verilmiş gösterimi Şekil 4.1’de sunulmaktadır.

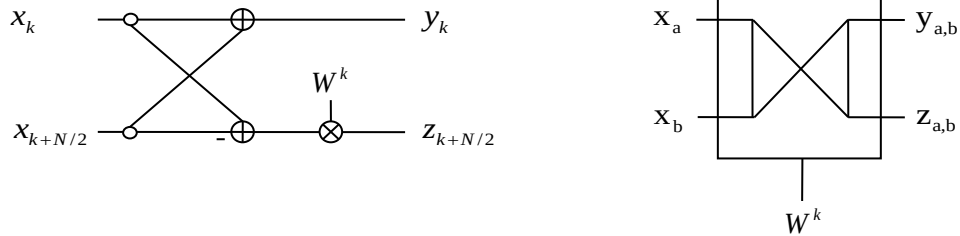


Şekil 4.1. DIF FFT bölme işlemi gösterimi.

İşlem sayısı azaldığı için (toplama ve çarpma işlemleri), hesaplamalar daha hızlı yapılmakta ve dolayısıyla Hızlı Fourier Dönüşümü adı verilen ve ancak DFT'nin hızlı hesaplama biçimi olan yöntem kullanılmaktadır. Bölme işlemi için, her bir alt bölümdeki noktaların sayısı, radiks olarak adlandırılan r 'nin katı olmalıdır. m tam sayı olmak üzere, eğer $N = r^m$ ise tüm FFT, $N \log_r N / r$ r -noktalı FFT'ler ile uygulanabilir. Radiks-2 FFT'nin DIT veya DIF olmak üzere iki şekilde uygulama biçimi mevcuttur. Bu uygulamalar aynı sonucu vermekte, ancak biçimsel olarak farklılık göstermektedirler. Örnek olarak, radiks-2 $N = 16$ giriş noktalı DIF FFT hesaplama yapısı Şekil 4.2'de verilmektedir. Bu 2-noktalı FFT yapısı işlem ögesi (PE) olarak adlandırılmakta ve tüm PE'ler toplama-çıkarma ve bir çarpıcı içeren kelebek şeklinde görünen bağlantılarından oluşmaktadır. Tüm PE'ler kelebeğin alt bacağındaki sabit çarpan haricinde aynıdır. Bu yapı, Şekil 4.3'te gösterilen tek bir PE'yi kullanan özyinelemeli/tekrarlayan algoritmalara uygundur. r arttıkça, algoritmik karmaşıklık azalmaktadır ancak veri gönderim yolları daha karmaşık bir hal almaktadır.



Şekil 4.2. $N = 16$ -noktalı DIF FFT yapısı.

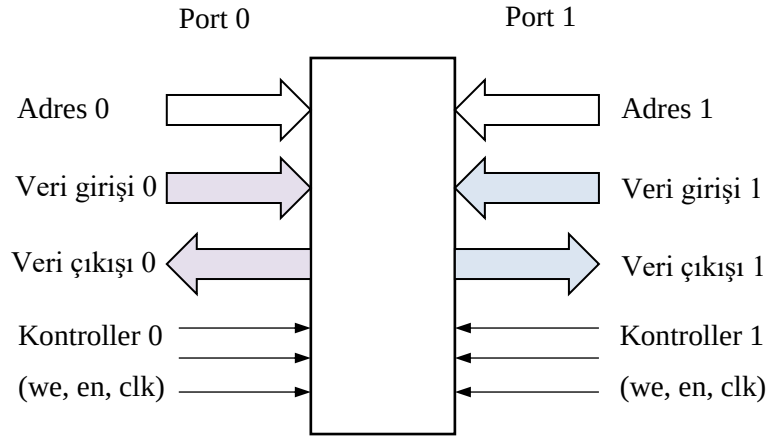


Şekil 4.3. Tek bir işlem ögesi (PE).

4.2. Çift Portlu Hafıza Yapıları

Çift portlu hafıza bloklarının kullanımı FPGA ile yapılan çalışmalarda geniş yer kaplamaktadır. İkinci portun kullanımı ilave adres çözücü devresi gerektirmesine rağmen, devredeki gecikmeyi yani fazladan saat darbesi gereksinimini ortadan kaldıracağı için, aynı anda yazıp okuma özelliği olan çift portlu hafıza kullanımı tercih edilmektedir. Aynı yere yazılıp okunmadan kaynaklanabilecek problemleri gidermek için birçok adresleme

algoritması tasarlanmıştır. Bu hafızalar özellikle gerçek zamanlı uygulamalarda kullanışlı olmaktadır. Çift portlu hafıza bloğu çalışma prensibi, ortada tek bir fiziksel hafıza bloğu olmak üzere, Şekil 4.4 ile daha kolay canlandırılabilir. Her iki port birbirinden bağımsız olarak aynı fiziksel hafızaya erişebilmektedir. Ancak, her iki port aynı adrese aynı anda eriştiğinde problemler çıkabilmektedir.



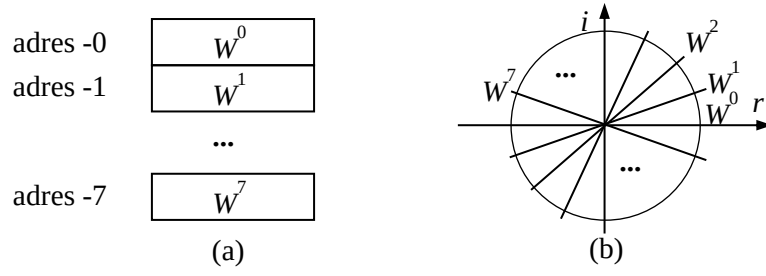
Şekil 4.4. Çift portlu hafıza blok diyagramı.

Çift portlu hafızaların farklı uygulamalardaki ihtiyaçlarına göre çeşitli çalışma modları bulunmaktadır. Bu modlar “önce okuma”, “önce yazma” ya da “değişiklik yok” modlarıdır. Her iki portun da aynı saat frekansı ile kontrol edilmesi ve porttan okunan verilerin geçersiz olmaması sebebiyle yapılan çalışmalarda “önce okuma” modu kullanılmaktadır. Yeni gelen veriyi belirlenen adresteki hafızaya yazarken aynı anda aynı hafızada bulunan bir önceki veriyi okuyarak işlem gerçekleştirilmektedir. Bu işlem bir sonraki saat darbesi gelene kadar hafıza çıkışlarında bulunan geçici belleklerde (buffer) verinin saklanmasıyla sağlanmaktadır. FPGA içerisindeki RAM’lerde hafıza çıkışlarındaki geçici bellekler sayesinde fazladan hafıza kullanımına ihtiyaç duyulmamaktadır. Ancak hafızaya yazılacak olan verinin boyutuna göre hafıza çıkışındaki geçici belleklerin ihtiyacı karşılamaması veya var olmaması durumunda devre maliyeti çok küçük olacak şekilde fazladan geçici bellek kullanımı olasıdır.

4.3. Sabit Çarpanlar ROM'u

FFT algoritmalarında bir sabit çarpan (twiddle factor), Cooley-Tukey FFT algoritmasının kelebek işlemlerinde verilerle çarpılan trigonometrik sabit katsayılarından herhangi birisidir. Bu terim ilk defa 1966'da Gentleman & Sande tarafından kullanılmıştır (Gentleman ve Sande, 1966). Birçok araştırmacı hafıza gereksinimini azaltmak amacıyla sabit çarpanlar ROM'u (twiddle ROM) üzerine çalışma yapmışlardır. Bu çalışmalar aslında doğrudan sayısal sentez (DDS) sinüzoidal işaret üretimi metodu ile paralel olup, hafıza kullanımını en aza indirmeyi hedeflemektedir (Sansaloni vd., 2007; De Caro ve Strollo, 2005). Hiçbir azaltma uygulanmadığı durumda, $N/2$ kadar bellek (ROM) gerekmektedir. Hafıza kullanımını azaltmak için dönüşüm çekirdeğinin simetri ($Re\{W^t\} = -Re\{W^{\frac{N}{2}-t}\}$ ve $Im\{W^t\} = Im\{W^{\frac{N}{2}-t}\}$) eşitlikleri ile bu değerlerin yarısının diğer yarısından hesaplanabileceği fark edilebilmelidir. Bu dönüşümün yapılmasıyla, asıl değerleri elde etmek için hafızadan okunan karmaşık değerın gerçık kısmının işaretinin değıştirilmesi gerekmektedir. Az veri ile işlem gerektiren FFT'ler için, karşılaştırma, çıkarma, işaret değışikliği, vb. için fazladan kullanılacak devre göz önüne alındığında tercih edilecek bir uygulama değildir. Ancak işlem yapılacak veri miktarı arttıkça FFT hesaplamalarında böyle bir yöntemle ROM hafıza alanından büyük ölçüde tasarruf ettirecektir. Daha büyük veri işleyen FFT'ler için, gerçık ve sanal kısımların değıştirilmesini içeren ve veri genişliğinde çarpıcılar gerektiren 4-kat simetriyi kullanma düşünülebilir. Daha fazla veri işlenen FFT'ler için bile sabit çarpanlar, tek bir karmaşık sabit saklanması gerektiren, yinelemeli metod kullanılarak hesaplanabilmektedir

Sabit çarpanlar (twiddle factors) ROM'u, devre elemanlarına karmaşık sabit çarpıcılar sağlamaktadır. Sabit çarpanlar belleğinde hafıza kullanımını en aza indirmek hedeflenir. Hafıza kullanımını azaltma dönüşüm çekirdeğinin simetri özelliği kullanılarak yapılmaktadır (Şekil 4.5). Hafıza içerikleri Şekil 4.5.a'da ve $r-i$ karmaşık düzleminde ilgili noktaları gösteren birim çemberi Şekil 4.5.b'de verilmiştir. Hiçbir azaltma uygulanmadığı durumda, $N/2$ kadar hafıza gerekmektedir.



Şekil 4.5. a) $N=16$ için twiddle ROM içerikleri **b)** karmaşık düzlemdeki çarpanlar

Bu tez çalışmasında FFT hesaplama kısımlarına ağırlık verildiği için deneysel uygulamalarda gerekli olan tüm çarpıcılar $N/2$ büyüklüğündeki hafızada saklanmıştır.

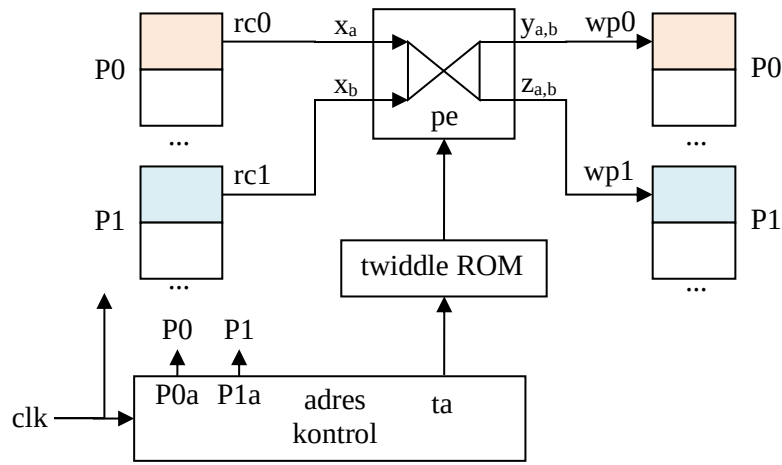
4.4. Tek Blok Çift-Port Hafıza-Tabanlı Radiks-2 Hızlı Fourier Dönüşümü Uygulamaları

Literatür çalışmaları sonucunda, tez çalışmasında OFDM sistemi için FPGA ile en az hafıza gereksinimi olan radiks-2 hafıza-tabanlı FFT metodu çift portlu RAM kullanılarak gerçekleştirilmiştir. Çift portlu hafıza bloklarının kullanımı FPGA ile yapılan çalışmalarda geniş yer kaplamaktadır. Devredeki gecikmeyi yani fazladan saat darbesi gereksinimini ortadan kaldıracağı için, aynı anda yazıp okuma özelliği olan çift portlu hafıza kullanımı tercih edilmiştir. Aynı yere yazılıp okunmadan kaynaklı olası çakışma problemlerini gidermek için yeni adresleme algoritmaları geliştirilmiştir. Bu hafızalar aynı anda okuma ve yazma özellikleriyle gerçek zamanlı uygulamalarda kullanışlı olmaktadır.

Çift portlu hafızaların farklı uygulamalardaki ihtiyaçlarına göre çeşitli çalışma modları bulunmaktadır. Çalışmada kullanılacak mod ise yazmadan önce okuma (read-before-write) modudur. Yeni gelen veriyi belirlenen adresteki hafızaya yazarken aynı anda aynı adreste bulunan bir önceki veriyi okuyarak işlem gerçekleştirilmektedir. Bu işlem bir sonraki saat darbesi gelene kadar hafıza çıkışlarında bulunan geçici belleklerde (buffer) verinin saklanmasıyla sağlanmaktadır. FPGA içerisindeki RAM'lerde hafıza çıkışlarındaki geçici bellekler sayesinde fazladan hafıza kullanımına ihtiyaç duyulmamaktadır. Ancak

hafızaya yazılacak olan verinin boyutuna göre hafıza çıkışındaki geçici belleklerin ihtiyacı karşılamaması veya var olmaması durumunda devre maliyeti çok küçük olacak şekilde fazladan geçici bellek kullanımı olasıdır.

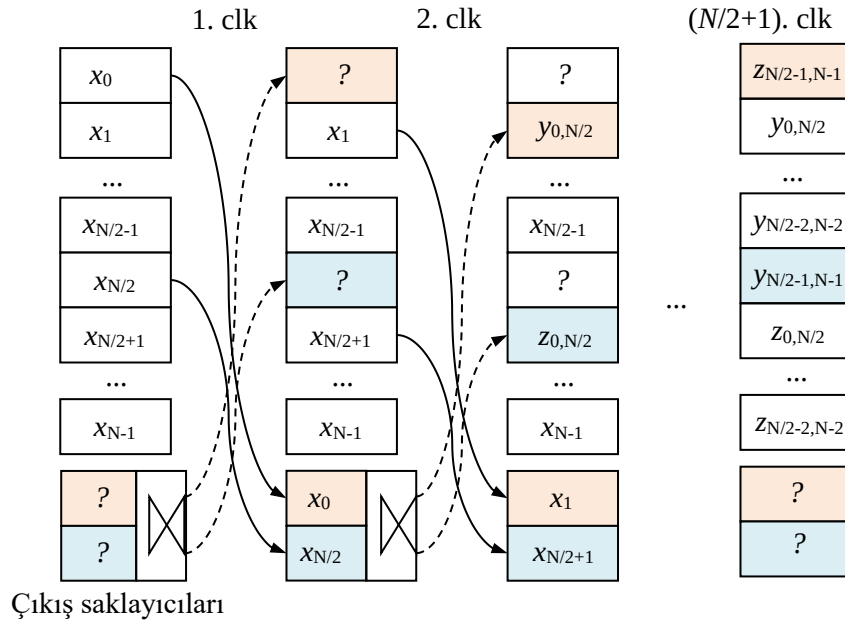
Önerilen yöntemler ile bir kelebek devresinin tek saat darbesi süresinde eşzamanlı hafızadan okuma ve hafızaya yazma işlemi gerçekleştirilebilir. Kelebek devre elemanları bağlantıları ve adres kontrolünü gösteren blok Şekil 4.6’da gösterilmiştir.



Şekil 4.6. Kelebek devre elemanı veri aktarımı ve adresleme kontrolü gösterimi.

Bölüm 4.4.1 ve 4.4.2’deki yöntemler için giriş verinin hafızaya yazılma adresleri ve çıkıştaki veri hafıza pozisyonları aynıdır. Bu uygulamalar için ilk FFT aşaması için hafıza okuma/yazma işlemleri Şekil 4.7’de gösterilmektedir. Giriş verileri $(x_0, \dots, x_{N-1})$ önceden $(0, \dots, N-1)$ hafıza pozisyonlarına yazılmıştır. Geçici bellekler ise alakasız başlangıç değerleri içermektedir. Adres hatları başlangıçta üzerinde çalışılacak veri adreslerini taşımaktadır. İlk saat darbesiyle geçici belleklerdeki hesaplanan alakasız değerler hafıza bloğunun ilk adresine yazılmakta, aynı zamanda bu adresteki var olan değerler okunarak geçici belleklere kaydedilmekte ve adres bir sonraki değeri okuma pozisyonu için güncellenmektedir. Bir sonraki saat darbesinden önce kelebek devre elemanları çıkışı değerleri bir sonraki hafıza adresine yazılmak için hazır bulunmaktadır. Bu okuma/ yazma işlemleri ilk FFT aşaması için tüm işlemler tamamlanana kadar devam etmektedir. Ancak,

sonuçlar aslında gerçek yerleri yerine bir sonraki hafıza adresine yazılmaktadır. Böylece ilk FFT aşaması tamamlandığında hafıza adresleri $Z_{N/2-1,N-1}, y_{0,N/2}, y_{1,N/2+1}, \dots, Z_{N/2-2,N-2}$ ara verileriyle doldurulmaktadır. Eğer veriler alışlageldik yerinde kullanılabilir (in-place) prensibiyle kaydedilmiş olsaydı $(0, \dots, N-1)$ hafıza adreslerine $y_{0,N/2}, y_{1,N/2+1}, \dots, Z_{N/2-2,N-2}, Z_{N/2-1,N-1}$ şeklinde sıralanmış olacaktı. Bu kaydırma yaklaşımıyla her bir hesaplama işlemi tek saat darbesinde tamamlanmaktadır.



Şekil 4.7. İlk aşamadaki veri işlemleri.

İlk aşama $N/2 + 1$ saat darbesinde hesaplandıktan sonra, ikinci aşama aynı çalışma prensibini kullanarak $N/4$ pozisyonundaki değerleri okumaya başlamaktadır. Bu aşama 0 ve $N/2$ 'den başlayan ilk aşama gibi olmamakta, değerler 1 ve $N/2 + 1$ adreslerinden başlamakta, son değer ise 0 adresinde bulunmaktadır. Bu nedenle, adres ayarlaması ve ilk aşamaya benzer olarak ilk değer çiftini okumak için bir saat döngüsü kadar süre harcanmaktadır.

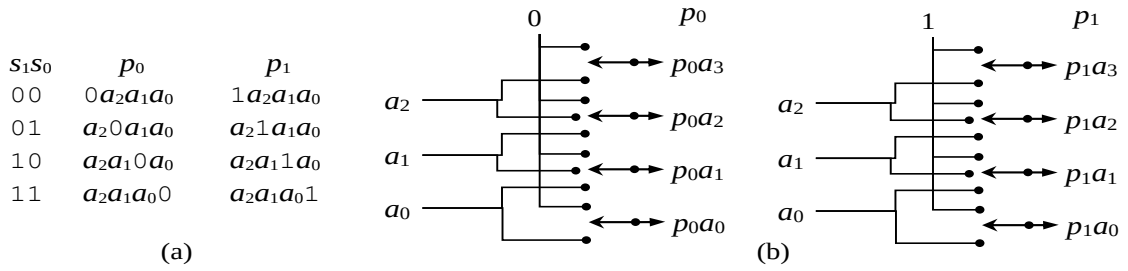
4.4.1. 16-Noktalı Radiks-2 Hızlı Fourier Dönüşümü Hafıza Adresleme Algoritması

Şekil 4.6’da blok şemada gösterilen adres kontrol devresinde ikili sayı tabanında sayaçlar kullanılmaktadır. Bunlar, 0’dan $N/2 - 1$ ’e kadar olan bir aşama için işlem sayısını saymak üzere $\log_2(N) - 1$ uzunluğundaki a ve 0’dan $\log_2(N) - 1$ ’e kadar aşama sayısını saymak üzere $\text{ceil}(\log_2(\log_2(N)))$ bit s sayıcılarıdır. Şekil 4.8 bu sayaçların her bir FFT hesaplama aşamasındaki değerlerini göstermektedir. N uzunluğunda hafıza adreslemesi için $\log_2 N$ bit gereksiniminden ve her işlem için iki adres ihtiyacından dolayı ilave bir adres biti ihtiyacı bulunmaktadır. İlave bit, farklı s değerleri için farklı a pozisyonlarına yerleştirilmektedir. Örnek olarak, $N = 16$ için 4 adres bitine ihtiyaç duyulmaktadır. 3 bit a , ve 2 bit ($\text{ceil}(\log_2(\log_2(16))) = 2$) s için kullanılmaktadır. p bit olarak adlandırılan ve s değerine göre yeri belirlenen 4. adres biti, devre elemanına gönderilen verinin alt ve üst girişlerine olacak şekilde adresleme ayrımını yapmaktadır. İlk aşama için a bitleri 000’den 111’e kadar saymaktadır. $a_2 a_1 a_0$, a ’nın uygun indislenmiş bitleri olmak üzere, işlemci elemanının her iki bacağı için adres girişleri $0a_2 a_1 a_0$ ve $1a_2 a_1 a_0$ olmaktadır. s sayacının bu adres satırlarıyla birlikte hesaplanmasıyla gerçek hafıza adresleri elde edilmektedir. İlk aşama için (toplamda 8 hesap) tüm hesaplamalar yapıldıktan sonra p ikinci aşama hesaplamaları için bir sağ bite kaydırılmakta ($a_2 p a_1 a_0$) ve tıpkı ilk aşamadaki gibi saymaktadır. Benzer şekilde 3. ve son aşama için satır adresleri $a_2 a_1 p a_0$ ve $a_2 a_1 a_0 p$ olmaktadır. p dışındaki bitler sadece basit bir ikili sayaç olarak çalışmaktadır ve adreslemede herhangi bir karışıklık oluşmamaktadır. p bitinin pozisyonu aşama sayacı olarak kullanılan $s_1 s_0$ bitleriyle kontrol edilmektedir. Gerçek adresleme ile önerilen tasarımda kullanılacak satır adresleri ilişkisini gösteren çizelge Şekil 4.8’de verilmiştir.

$s_1 s_0$ aşama sayacı bitleri, p_0 ve p_1 her iki port için satır adreslerini temsil etmek üzere, çift portlu hafıza giriş ve çıkış adresi üretimi doğruluk tablosunu gösteren şematik Şekil 4.9’da gösterilmektedir. Satır adreslemesi işlemleri yerinde kullanılabilir algoritma prensibiyle yapılmaktadır. Bu algoritmada, M adresinden okunan değer gerekli hesaplamalar yapıldıktan sonra tekrar M adresine yazılmaktadır. Önerdiğimiz adres üreticinde ise tek fark yazma işleminin bir sonraki adres olan $M + 1$ adresine yapılmasıdır. Eklenen fark (offset) her bir aşama için farklılık göstermekte, aslında s ’ye eşit olmaktadır.

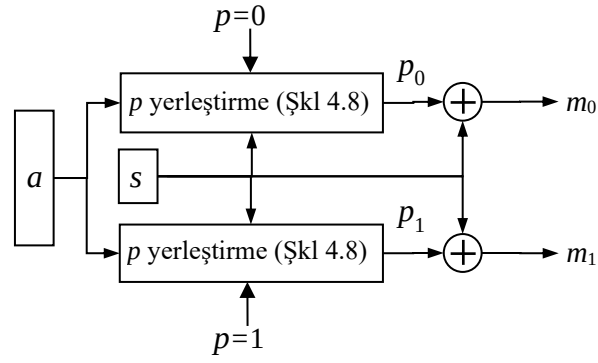
		aşama 0		aşama 1		aşama 2		aşama 3		
$s_1s_0 =$		00		01		10		11		
p		$a_2a_1a_0$	m_0	p	m_0	p	m_0	a	p	m_0
saat döngüsü ↓	port 0 adres ↑	0000	0000	0000	0001	0000	0010	0000	0011	0011
		0001	0001	0001	0010	0001	0011	0010	0101	0101
		0010	0010	0010	0011	0100	0110	0100	0111	0111
		0011	0011	0011	0100	0101	0111	0110	1001	1001
		0100	0100	1000	1001	1000	1010	1000	1011	1011
		0101	0101	1001	1010	1001	1011	1010	1101	1101
		0110	0110	1010	1011	1100	1110	1100	1111	1111
		0111	0111	1011	1100	1101	1111	1110	0001	0001
		m_1	m_1	m_1	m_1	m_1	m_1	m_1	m_1	
port 1 adres ↓	1000	1000	0100	0101	0010	0100	0001	0100	0100	
	1001	1001	0101	0110	0011	0101	0011	0110	0110	
	1010	1010	0110	0111	0110	1000	0101	1000	1000	
	1011	1011	0111	1000	0111	1001	0111	1010	1010	
	1100	1100	1100	1101	1010	1100	1001	1100	1100	
	1101	1101	1101	1110	1011	1101	1011	1110	1110	
	1110	1110	1110	1111	1110	0000	1101	0000	0000	
	1111	1111	1111	0000	1111	0001	1111	0010	0010	

Şekil 4.8. $N=16$ için üretilen adres satırı gösterimi.



Şekil 4.9. a) Satır adreslerine yerleştirilen p biti ile doğruluk tablosu b) Komütatör model.

Önerilen devreyi gerçeklemenin birçok yolu bulunmakla birlikte komütatör modeli ile gerçekleştirme örneği Şekil 4.9'da verilmiştir. Gerçek hafıza için çift port okuma/yazma adresleri m_0 ve m_1 ise, tasarımda bu adresler $m_0 = p_0 + s$ ve $m_1 = p_1 + s$ olmaktadır (Şekil 4.10).



Şekil 4.10. Gerçek adres üretici blok diyagramı.

Bir aşamadaki son okuma işlemi tamamlandıktan sonra, adres sayacı değeri devre elemanı çıkışlarını yazmak üzere 000 olmaktadır. Bu işlem $N = 2$ 'nin kuvveti olması durumunda alışlageldik ikili sayaç ile otomatik olarak yapılmaktadır. Sayaç değeri 1...1 iken 1 artırıldığında değer en başa (0...0) dönmektedir. Son yazma pozisyonundaki değer beklendiği gibi okunur, ancak yeni aşamaya geçilirken geçersiz bir okuma gerçekleşmektedir. Bu durum fazladan 1 saat darbesinin devreye girdiği andır. Son değerler yazılırken (ve gereksiz okuma işlemi gerçekleşirken) adres sayacında değişiklik yapılmamalı ancak bir sonraki aşama için doğru değerlerin okunması amacıyla p bitinin konumu değiştirilmelidir. Açıkçası, gelecek saat darbesiyle yazılan veriler geçersizdir. Aslında bu pozisyonundaki değer aynı anda okunacağı için geçerli değerler kaybolmamaktadır. Özetle, her aşama başlangıcında fazladan bir okuma gerçekleştiği zaman yazma işlemi yapılmaktadır. Bu özel durumdan sonra daha önce de bahsedildiği üzere kalan tüm aşamalar için okuma ve yazma işlemleri benzeri şekilde gerçekleştirilmektedir.

Çalışmanın FPGA üzerinde VHDL ile uygulanması oldukça basittir. Verilen örnek VHDL kodunda (Şekil 4.11) bayrak sinyali olarak kullanılan `retry` ilk başta `false` olarak tanımlanmaktadır. Şekil 4.12 hafıza adreslerinin hesaplanmasını gösteren VHDL kodlamasıdır.

```

if (a=(others=>'0')) then
  if (retry) then a <= a+1;
  else s <= s+1; end if;
  retry <= not(retry);
else a <= a+1; end if;

```

Şekil 4.11. İlave saat darbesiyle aşama değişimi yapan VHDL kod örneği.

```

p0 <= ('0',a(2),a(1),a(0)) when s="00" else
      (a(2),'0',a(1),a(0)) when s="01" else
      (a(2),a(1),'0',a(0)) when s="10" else
      (a(2),a(1),a(0),'0');
p1 <= ('1',a(2),a(1),a(0)) when s="00" else
      (a(2),'1',a(1),a(0)) when s="01" else
      (a(2),a(1),'1',a(0)) when s="10" else
      (a(2),a(1),a(0),'1');
m0 <= p0+s;
m1 <= p1+s;

```

Şekil 4.12. $N=16$ için çoklayıcı kullanılarak VHDL ile gerçekleştirilen kombinatoriyal adres değeri üretimi.

Tez çalışmasında kullanılan sabit çarpanlar (twiddle factors) ROM'u, devre elemanlarına karmaşık sabit çarpıcılar sağlamaktadır. Şekil 4.5'te gösterildiği üzere $s = 00$ olduğunda $N/2$ hafıza için sabit dönüşüm çekirdeği değerleri a adres sayıcısı yardımıyla hafızaya kaydedilmektedir ve twiddle ROM adres sinyali ta olarak isimlendirilmektedir. Genel olarak ta , a 'nın en az ağırlıklı bitlerinin (LSB) sıfırlanmasıyla hesaplanabilmektedir. $s = 0$ için a 'nın en az ağırlıklı birinci biti, $s = 1$ için en az ağırlıklı iki biti sıfırlanmaktadır. Ve bu işlem diğer aşamalar için benzer şekilde devam etmektedir. Son aşama için $uzunluk(a) = genişlik(s)$ eşitliği sebebiyle ta 'nın tüm bitleri sıfır olmaktadır (Şekil 4.13). Ayrıca, $W^k = W^{N/2-k}$ eşitliğinden bu değerlerin yarısının diğer yarısından kolaylıkla hesaplanabileceği ortadadır. ROM büyüklüğü $N/4$ olduğunda, gerçek sürücü adresi bulmak için $N/4$ 'ten büyük olan ta' adresine $ta = N/2 - ta'$ adres dönüşümü uygulanmaktadır. Bu dönüşümün yapılmasıyla, asıl değerleri elde etmek için hafızadan okunan karmaşık değerlerin gerçek kısmının işaretinin değiştirilmesi gerekmektedir. Az sayıda veri içeren FFT'ler için, karşılaştırma, çıkarma, işaret değişikliği için fazladan kullanılacak

devre göz önüne alındığında tercih edilecek bir uygulama değildir. Ancak büyük verili FFT hesaplamalarında böyle bir yöntemle ROM hafıza alanından büyük ölçüde tasarruf ettirecektir. Bu tez çalışmasında sabit çarpanlar $N/2$ boyutundaki ROM hafızada saklanmıştır.

```
ta <= (a(2), a(1), a(0)) when s="00" else
      (a(2), a(1), '0') when s="01" else
      (a(2), '0', '0') when s="10" else
      ('0', '0', '0');
```

Şekil 4.13. $N=16$ için ta adreslerini üreten VHDL kodu.

4.4.2. N-Noktalı Radiks-2 Hızlı Fourier Dönüşümü Hafıza Adresleme Algoritması

Bu çalışmada, OFDM için gerekli ($N = 8192$ noktaya kadar uygulaması yapılmış) tüm FFT'ler için uygulanabilen yeni bir hafıza adresleme algoritması geliştirilmiş ve radiks-2 hafıza-tabanlı FFT metodu uygulanmıştır. Hafıza-tabanlı terimi genellikle verilerin (ve ara verilerin) bellek bloklarında tutulduğu ve hesaplamalar için gerektiğinde geri alındığı anlamına gelmektedir. Alışlagelmiş belleklerde, okuma/yazma veri yollarının sayısında bazı karışıklık/sınırlamalar bulunmaktadır. Ancak birçok araştırmacı (Johnson, 1992; Hidalgo vd., 1999; Chen vd., 2014; Xia vd., 2017; Hsiao vd., 2010; Jo. ve Sunwoo, 2005; Baek ve Choi, 2008) bu karışıklık/sınırlamaların üstesinden gelmek için ayrı adresleme devresine (adres kod çözücüleri) sahip bölünmüş bellek blokları kullanımını uygun görmüşlerdir.

Çoklu blok yaklaşımlarda (Johnson, 1992; Hidalgo vd., 1999; Chen vd., 2014; Xia vd., 2017; Hsiao vd., 2010; Jo. ve Sunwoo, 2005; Baek ve Choi, 2008; Anonim, 2007; Anonim, 2017), çoklu okuma/yazma işlemleri eşzamanlı olarak gerçekleştirilmekte aynı zamanda daha yüksek radiksli algoritmalar uygulanabilmektedir. Ancak bu uygulamalar ek devre karmaşıklığına sebep olabilmektedir. Önerilen algoritmaların çoğu (Ma vd., 2015;

Ayinala vd., 2013), daha büyük FFT boyutları için kaynak sınırlamalarına varabilecek performans iyileştirmesi için FFT boyutundan daha büyük bellek kullanmaktadır. Bu tez çalışmasında tek blok hafıza için en düşük kaynak ve en düşük operasyon saati sayısı olan bir FFT işlemci tasarımı yapılmıştır. Aynı zamanda, VHDL dili kullanılarak tasarımda işlemcinin genişletilebilir ve kolayca değiştirilebilir olması hedeflenmiştir. Bu nedenle birçok FPGA’da bulunan tek blok çift-port hafıza kullanımı tercih edilmiştir. r -noktalı FFT’nin tek bir saat darbesinde tamamlanması hedeflendiğinde tek blok kullanımından ve aynı anda iki okuma işlemi yapılacağından $r = 2$ ile sınırlı olması dezavantajı bulunmaktadır.

Önerilen yöntemde hafıza okuma/yazma işlemleri Bölüm 4.4’te verilmiştir. Bir kelebek devresi işlemleri tek saat darbesi süresinde, hafızadan eşzamanlı okuma ve hafızaya yazma işlemi yapılarak tamamlanmaktadır. Tek blok hafıza ile tüm FFT işlemlerinin en az saat döngüsünde hesaplanması için, daha önce hesaplanmış verilerin belleğe yazılma işlemi yapılırken yeni verilerin okunması gerekmektedir. Tek blok hafıza ile okuma ve yazma işlemleri aynı hafıza adresinden yapılabildiği için adresler okuma/yazma işlemleri esnasında (aynı saat darbesinde) güncellenmeli, değerler bir sonraki hafıza pozisyonları için tekrar hesaplanmalıdır. Hesaplanan değerler bir sonraki okuma pozisyonuna yazılırken aynı anda bu pozisyonadaki çıkış saklayıcılarında kaydedilen verilerin okuma yapılmasıyla okuma/yazma işlemi tek saat darbesinde yapılmış olur. Bahsi geçen okuma yazma işlemini gösteren adresleme şeması Şekil 4.6’da gösterilmektedir.

Tek saat döngüsünde, çift portlu hafızanın P0 ve P1 portlarından iki değer okunur ve bu değerler mevcut PE çıkışları ile doldurulur. FPGA içerisindeki çift portlu hafızaların okuma ve yazma için aynı hafıza pozisyonlarını kullanma zorunluluğundan dolayı okuma/yazma hafıza pozisyonları aynı olmalıdır. Yazılan değerler çıkış saklayıcılarındaki okunan veriler değil, bir önceki hesaplanan verilerdir.

FFT $s = 0$ aşaması hafıza okuma/yazma işlemleri Şekil 4.7’de verilmektedir. $s = 0$ için işlemler $N/2 + 1$ saat darbesinde hesaplandıktan sonra, $s = 1$ aşamasında aynı çalışma prensibini kullanarak $N/4$ pozisyonadaki değerler okumaya başlanmaktadır. Bu aşama 0 ve $N/2$ ’den başlayan $s = 0$ aşaması gibi olmamakta, değerler 1 ve $N/2+1$ olarak belirtilen adreslerden başlamaktadır. Bu yüzden adres ayarlaması yapılması ve ilk çiftleri okumak için

fazladan bir saat döngüsü harcanmak zorundadır. Bu fazladan saat döngüsüne sadece bir sonraki aşamaya taşınırken değil, aynı zamanda yazma pozisyonları okunduğunda da gereklidir. Her aşama s 0'dan başlayan aşama numarası olmak üzere, her biri $N/2^{s+1}$ kelebekten oluşan 2^s adet kelebek kümesi bulunmaktadır. Örneğin, Şekil 4.2'de, 0 aşamasında 8 kelebekten oluşan 1 adet kelebek kümesi, 1 aşamasında 4 kelebekten oluşan 2 adet kelebek kümesi, vb. bulunmaktadır. Bir kümedeki son yazma adresleri sonraki kümedeki ilk okuma adresleriyle aynı olmadığından, ek bir saat döngüsü gerektiren bir adres ayarı olmalıdır. Bu durumun tek istisnası, her kümenin sadece bir kelebeğe sahip olduğu ve hesaplanan değerlerin 1 yerine 2 sonraki bellek konumlarına yazılmasına izin verdiği son aşamadır. Yani, her PE çıkışı bir sonraki kelebek giriş veri pozisyonlarına kümeler arası fazladan saat döngüsü ihtiyacı olmadan yazılabilmektedir.

Bu yaklaşımla tüm FFT aşamaları $\frac{N}{2} \log_2 2N = \frac{N}{2} (1 + \log_2 N)$ saat döngüsünde tamamlanmaktadır. Aslında, tek-blok hafıza algoritma çalışmaları oldukça azdır. PE kullanımı (utilization) fazla veri ile gerçekleştirilen FFT'ler için %100'e yakındır.

Şekil 4.6'da blok şemada gösterilen adres kontrol devresinde ikili sayı tabanında sayaçlar kullanılmaktadır. 0'dan $\log_2(N) - 1$ 'e kadar sayan aşama değeri s , 0'dan $2^s - 1$ 'e kadar sayan kelebek kümesi değeri c ve bir küme içerisindeki 0'dan $N/2^{s+1} - 1$ 'e kadar sayan işlem numarası a ile isimlendirilen 3 adet sayaç ile ifade edilmektedir. Tüm bu değerler $\lceil \log_2 N \rceil$ bit kaydedicide/sayaçta saklanmaktadır. Şekil 4.14 16-noktalı FFT için her hesaplama adımında bu sayaçların değerini göstermektedir. a_0 ve a_1 PE'nin üst ve alt bacaklarının hafıza port adres satırlarını ifade etmektedir. ta ise sabit çarpanlar (twiddle) ROM'u içinde her pozisyon için saklanan W^t sabit çarpan adreslerini ifade etmektedir. Şemada, satır değerleri, sayaç darbesinin yükselen kenarında, sütun başlıkları tarafından uygun şekilde gösterildiği gibi, sayaçların s, a, a_0, a_1 ve ta değerlerini göstermektedir. $ta = x$, PE'nin önceden hesaplanmış değerinin yazılacağı ve aynı zamanda fazladan/gereksiz bir okuma yapılacağı yerleri göstermektedir. Bu değerler kullanılmadığı için önemsiz olarak nitelendirilmekte, yani "x" (önemsiz) olarak işaretlenmektedir. x 'in hemen altındaki satırlarda, yeni değerler yeni kümeden okunur, ancak aynı zamanda yazılan değerler anlamsızdır (daha önce okunan alakasız değerler için PE çıktıları). Bu alakasız değerler, 0'lar ile değiştirilerek yazılabilir, fakat sonraki işlemlerde gerçek değerlerle değiştirileceği

için gerekli değildir. Bununla birlikte, bu noktalarda, adres a_1 'de yazılı değer, aslında bir sonraki okumada gerekli olan değer üzerine yazılmaktadır. Bu nedenle, aynı zamanda okunan değer, daha sonra kullanılmak üzere saklanmalıdır. Örneğin, $s = 1$ aşamasında $a_1 = 9$ 'dan okunan değer bir sonraki işlem için saklanmalıdır. Bu, önerilen FFT mimarisinin tamamında (hafıza adres kod çözücüler hariç) bir veri komütatörünün (mux) ihtiyaç duyulduğu tek yerdir.

Her aşamada aşama sonlarındaki çıkış değerleri pozisyonlara yerleştirilmiş (tüm pozisyonlar değişmiş) ve son değer 0 hafıza pozisyonuna yerleştirilmiş olacaktır. Bu durum bir sonraki pozisyonda adreslere uygun fark değerleri eklendiği sürece problem oluşturmamaktadır. Şekil 4.14 a_0 ve a_1 'e fark değerleri eklenmiş halini göstermektedir. Fark değeri aslında s değeriyle aynı olmaktadır. Sayıcıları saklamak yerine aritmetik işlemleri (toplama) kullanarak a_0 ve a_1 adreslerini hesaplamak, gerçek veri genişlikleriyle karşılaştırıldıklarında daha küçük olduğu için ve toplama işlemleri için sentezleyiciler basit kombinatoryal lojik devreleri kullandıkları için daha kullanışlıdır.

s = 0						s = 1						s = 2						s = 3					
c	a	a ₀	a ₁	ta		c	a	a ₀	a ₁	ta		c	a	a ₀	a ₁	ta		c	a	a ₀	a ₁	ta	
0	0	0	8	0		0	0	1	5	0		0	0	2	4	0		0	0	3	4	0	
0	1	1	9	1		0	1	2	6	2		0	1	3	5	4		1	0	5	6	0	
0	2	2	10	2		0	2	3	7	4		0	2	4	6	x		2	0	7	8	0	
0	3	3	11	3		0	3	4	8	6		1	0	6	8	0		3	0	9	10	0	
0	4	4	12	4		0	3	5	9	x		1	1	7	9	4		4	0	11	12	0	
0	5	5	13	5		1	0	9	13	0		1	2	8	10	x		5	0	13	14	0	
0	6	6	14	6		1	1	10	14	2		2	0	10	12	0		6	0	15	0	0	
0	7	7	15	7		1	2	11	15	4		2	1	11	13	4		7	0	1	2	0	
0	7	8	0	x		1	3	12	0	6		2	2	12	14	x		0	0	3	4	0	
						1	3	13	1	x		3	0	14	0	0							
												3	1	15	1	4							
												3	2	0	2	x							

Şekil 4.14. $N=16$ için üretilen sabit çarpanlar ROM adresleri ve üretilen veri hafızası.

Açıkçası, gerçekleştirilen adresleme devresi başka yollarla da gerçekleştirilebilir. Kaydediciler FPGA'ler için oldukça düşük maliyetlidirler. Bu nedenle, tek bir sayaca sahip olmak ve ondan tüm diğer sinyalleri üretmek yerine, kombinatoryal devrelerin azaltılması ve sonuç olarak operasyon frekansının iyileştirilmesi için kendi sayaçlarının ve ilgili mantığın tüm çoklu-bayt mantık sinyallerine atanması tercih edilmektedir. Tasarım kriterleri ne olursa olsun, PE kombinatoryal mantığının, karmaşık toplayıcıları içereceği ve istenen hassasiyete sahip karmaşık çarpıcılar içereceği için muhtemelen tüm FFT tasarımı için kritik olduğu söylenebilmektedir. Bu nedenle, daha yüksek saat frekansları için PE geliştirilmedikçe, tasarımın geri kalanının çalışma frekansı üzerinde hiçbir etkisi olmamasına gerek olmadığı sonucuna varılabilmektedir. Başlangıçtan son aşamadaki son kelebeğin tamamlanmasına kadar geçen saat darbesi sayısı ve kullanılan hafıza miktarı benzer yapıdaki FFT işlemci mimarileriyle karşılaştırılmış, bulgular Bölüm 5'te çizelgeler (Çizelge 5.1, Çizelge 5.2) halinde sunulmuştur.

Şekil 4.2 ve Şekil 4.14, $s = 00$ ve ROM boyutu $N/2$ olduğunda a sayacıyla kolaylıkla adreslenebilen hafızaya kaydedilmiş sabitleri göstermektedir. Bu hafızanın içeriği, Şekil 4.5a'da, r -i kompleks düzlemindeki birim daire üzerindeki karşılık gelen noktalarla birlikte gösterilmektedir (Şekil 4.5b).

$s = 00$ olduğunda, ta (sabit çarpanlar ROM'u adres sinyali) a 'ya eşit olmaktadır. Genel olarak, ta , en az ağırlıklı $\log(N) - 1$ bitinin s kere a 'nın sola kaydırılmasıyla hesaplanmaktadır. Yani, $s = 0$ ise $ta = a$, $s = 1$ ise $ta = 2a$, $s = 2$ ise $ta = 4a$, vb.. şeklinde devam etmektedir. Son aşama için $uzunluk(a) = genişlik(s)$ durumundan dolayı ta 'nın tüm bitleri 0 olmaktadır. Devreye sabit çarpanları üreten ta adreslerinin VHDL ile hesaplanması Şekil 4.15'te gösterilmiştir.

```
ta <= (a(2), a(1), a(0)) when s="00" else
      (a(2), a(1), '0') when s="01" else
      (a(2), '0', '0') when s="10" else
      ('0', '0', '0');
```

Şekil 4.15. $N=16$ için ta adreslerini a sayacından üreten VHDL kodu.

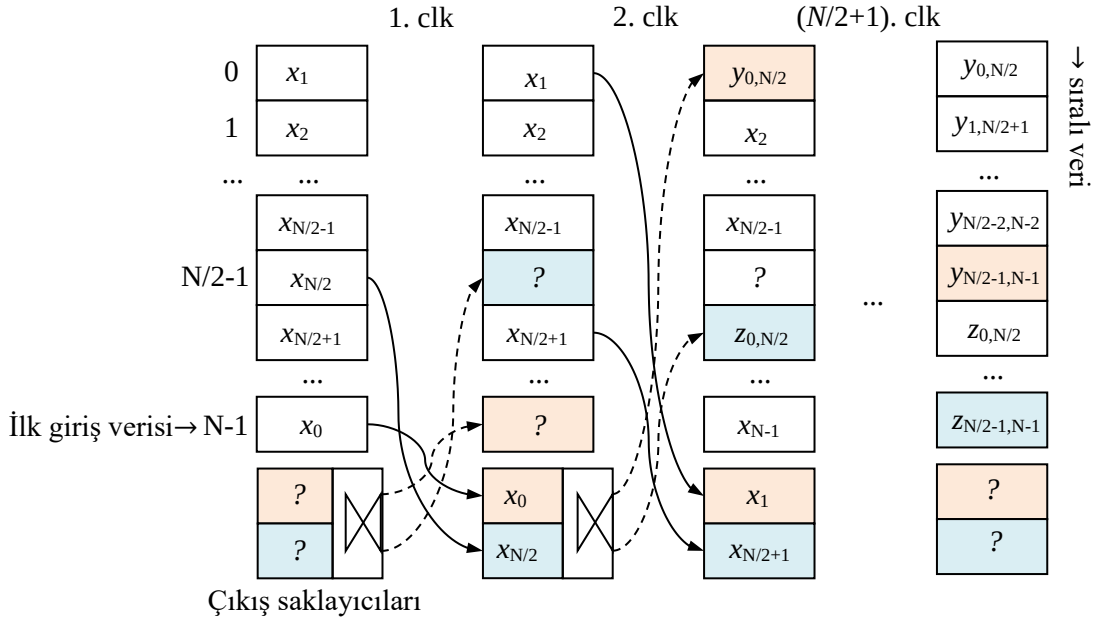
Bu tez çalışmasında, deneysel uygulamalarda gerekli olan tüm çarpıcılar $N/2$ büyüklüğündeki hafızada saklanmıştır.

4.4.3. Radiks-2 Hızlı Fourier Dönüşümü ile N-Noktalı Yeni Hafıza Adresleme Algoritması

Bu çalışma Bölüm 4.4.1 ve 4.4.2’de bahsedilen çalışmalar ile hafıza okuma/yazma açısından benzer özellikler göstermekte, ancak ilk başta hafızaya yazılan verilerin sıralaması ve adresleme algoritması açısından farklılık göstermektedir. Öyle ki, tasarımda tek blok çift-port hafıza yapıları kullanılmakta, kelebek devre elemanı veri aktarımı ve adresleme kontrolü ile tek bir PE okuma/yazma işlemleri Bölüm 4.4’te anlatılmaktadır. Bu çalışmada giriş verilerinin hafıza pozisyonlarına yazılması önceki çalışmalardaki gibi $(0, 1, \dots, N - 1)$ adreslerine sırayla değil, $(N - 1, 0, 1, \dots, N - 2)$ hafıza pozisyonlarına olmaktadır. Yani, ilk veri hafızanın son adresine yazılırken, ikinci veri ise hafızanın ilk adresine yazılmaktadır. Böylece ilk FFT aşaması tamamlandığında, hesaplanmış veriler hafıza adreslerinde sırayla yazılmış ve $(0, \dots, N - 1)$ hafıza pozisyonlarında konumlanmaktadır.

İlk FFT aşaması için hafıza okuma/yazma işlemleri Şekil 4.16’da gösterilmektedir. Giriş verileri $(x_0, \dots, x_{N-1})$ önceden $(N - 1, 0, 1, \dots, N - 2)$ hafıza pozisyonlarına yazılmaktadır. Adres satırları ilk başta 0’dan $N - 1$ ’e olan veriler yerine bir önceki verileri taşımaktadır. Geçici bellekler ise rastgele başlangıç değerleri içermektedir. İlk saat darbesiyle geçici belleklerdeki hesaplanan alakasız değerler hafıza bloğunun $N - 1$. ve $N/2 - 1$. adresine yazılmakta, aynı zamanda bu adresteki var olan değerler okunarak geçici belleklere kaydedilmekte ve adres bir sonraki değeri okuma pozisyonu (0. ve $N/2$.) için güncellenmektedir. Bir sonraki saat darbesinden önce kelebek devre elemanları çıkışı değerleri bir sonraki hafıza adresine yazılmak için hazır bulunmaktadır. Bu okuma/ yazma işlemleri ilk FFT aşaması için tüm işlemler tamamlanana kadar devam etmektedir. İlk başta değerlerin bir önceki hafıza adreslerine yerleştirilmesiyle ilk aşama işlemleri tamamlandığında sonuçların gerçek/olması gereken hafıza pozisyonlarında bulunması sağlanmaktadır. Böylece ilk FFT aşaması tamamlandığında $(0, \dots, N - 1)$ hafıza adresleri $y_{0,N/2}, y_{1,N/2+1}, \dots, z_{N/2-2,N-2}, z_{N/2-1,N-1}$ ara verileriyle doldurulmuş olmaktadır. Bu

yaklaşım ile her bir hesaplama işlemi tek saat darbesinde tamamlanmakta ve ilk aşama işlemleri $N/2 + 1$ saat darbesinde tamamlanmaktadır.



Şekil 4.16. İlk aşamadaki veri işlemleri.

İlk aşama $N/2 + 1$ saat darbesinde hesaplandıktan sonra, ikinci aşama aynı çalışma prensibini kullanarak $N/4$ pozisyonundaki değerleri okumaya başlamaktadır. Bu aşama ilk aşamada verilerin olması gereken hafıza pozisyonlarından bir önceki pozisyona yerleştirilmesi ve hesaplanan değerlerin bir sonraki pozisyona yazılmasıyla 0. ve $N/4$. adreslerden başlamaktadır. Bir sonraki aşamaya geçildiğinde adres ayarlamasının tekrar yapılması yani ilk çiftlerin okunması için fazladan bir saat döngüsü harcanmaktadır. Bu fazladan saat döngüsüne sadece bir sonraki aşamaya taşınırken değil, aynı zamanda yazma pozisyonlarındaki verilerin okunması için de ihtiyaç duyulmaktadır.

Her aşamada, s 0'dan başlayan aşama numarası olmak üzere, her biri $N/2^{s+1}$ adet kelebekten oluşan 2^s tane kelebek kümesi bulunmaktadır. Bu kelebekler hafızanın iki ayrı adresini temsil eden alt ve üst olmak üzere a_0 ve a_1 olarak isimlendirilen bacaklara sahiptirler. Kelebek bacakları arasındaki ofset değerleri (a_0 ve a_1 arasındaki adres farkı)

mevcut aşamadaki küme içerisindeki $N/2^{s+1}$ kelebek sayısına eşit olmaktadır. Örnek olarak, Şekil 4.2’de şematikte gösterildiği üzere $N = 16$ -noktalı FFT için $s = 0$ aşamasında $N/2^{s+1}$ ile hesaplanan 8 kelebek, 2^s hesaplanan ile 1 adet kelebek kümesi olduğu görülmektedir. Yine $s = 1$ aşamasında aynı hesaplama yöntemleriyle kelebek sayısı 4, küme sayısı ise 2 adettir. Bahsi geçen hesaplama yöntemleri her aşama için aynı şekilde uygulanmaktadır. Ofset olarak adlandırılan okunan/yazılan kelebek alt ve üst bacak adresleri arasındaki fark aşamadaki küme içerisindeki kelebek sayısına eşit ($N/2^{s+1}$) ve her aşama için farklılık göstermektedir.

Bu yaklaşımla FFT hesaplamaları, tek aşama kelebek işlemleri için $\frac{N}{2}$ olmak üzere tüm aşamalar için $\frac{N}{2} \log_2 N$ saat döngüsünde tamamlanmaktadır. Ek olarak, aşama sonlarında fazladan okumanın yapıldığı ve aynı zamanda bir sonraki aşama için adres ayarlama yapılması gerektiği durumlarda da fazladan bir saat darbesi kullanımı söz konusudur. Fazladan saat darbesi tüm FFT aşamaları için $\log_2 N$ ’dir. Böylece tüm FFT aşamaları toplamda $(\frac{N}{2} + 1) \log_2 N$ saat döngüsünde tamamlanmaktadır. Bu değer, bir önceki çalışmada yapılan $\frac{N}{2} (1 + \log_2 N)$ saat döngüsünde tamamlanan ve literatürde tek bir PE ve tek-blok hafıza kullanan algoritmalar arasındaki en düşük saat döngüsü değeridir.

Şekil 4.6’da blok şemada gösterilen adres kontrol devresinde ikili sayı tabanında sayaçlar kullanılmaktadır. Sayaçlar 0’dan $\log_2(N) - 1$ ’e kadar sayan aşama değeri s , 2^s ile belirlenen kelebek kümesi değeri c ve sabit çarpanlar değeri ta ’dır. Şekil 4.17 16-noktalı FFT için her hesaplama adımında bu sayaçların değerini göstermektedir. a_0 ve a_1 PE’nin üst ve alt bacaklarına bağlı olan hafıza portlarının adres satırlarını, ta ise sabit çarpanlar (twiddle) ROM’u içinde her pozisyon için saklanan W^t sabit çarpan adreslerini ifade etmektedir. Şemada, satır değerleri, saat darbesinin yükselen kenarında, sütun başlıkları tarafından uygun şekilde gösterildiği gibi, s, c, a_0, a_1 ve ta gibi parametrelerin değerlerini de göstermektedir. $ta = x$, PE’nin önceden hesaplanmış değerinin yazılacağı ve aynı zamanda fazladan/gereksiz bir okuma yapılacağı yerleri göstermektedir. Bu değerler kullanılmadığı için önemsiz olarak nitelendirilmekte, yani “ x ” (önemsiz) olarak ifade edilmektedir.

Her aşamada aşama sonlarındaki çıkış değerleri pozisyonlara yerleştirilmiş (tüm pozisyonlar değişmiş) ve son değer 0. hafıza pozisyonuna yerleştirilmiş olacaktır. Aritmetik işlemleri (toplama) kullanarak a_0 ve a_1 adreslerini hesaplamak, basit kombinatoriyal lojik devrelerin kullanımıyla oldukça kolaydır.

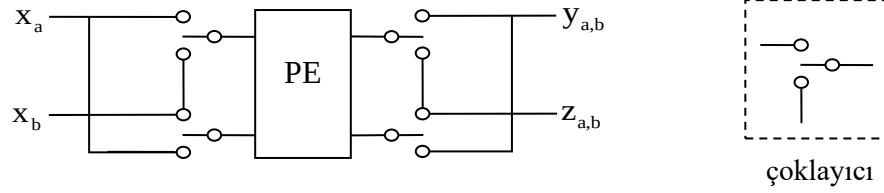
$s = 0$				$s = 1$				$s = 2$				$s = 3$			
c	a_0	a_1	ta	c	a_0	a_1	ta	c	a_0	a_1	ta	c	a_0	a_1	ta
0	15	7	0	0	0	4	0	0	0	2	4	0	0	1	0
0	0	8	1	1	8	12	0	1	4	6	4	1	2	3	0
0	1	9	2	0	1	5	2	2	8	10	0	2	4	5	0
0	2	10	3	1	9	13	2	3	12	14	0	3	6	7	0
0	3	11	4	0	2	6	4	0	1	3	0	4	8	9	0
0	4	12	5	1	10	14	4	1	5	7	0	5	10	11	0
0	5	13	6	0	3	7	6	2	9	11	4	6	12	13	0
0	6	14	7	1	11	15	6	3	13	15	4	7	14	15	0
0	7	15	x	0	0	4	x	0	0	2	x	0	0	1	x

Şekil 4.17. $N=16$ için üretilen sabit çarpanlar ROM adresleri ve üretilen veri hafıza adresleri.

İlk aşama için hafızadan okuma/yazma işlemleri ofset değerleri $N/2$ olan pozisyonlardan alınarak ve bir sonraki adreslere yazılarak gerçekleştirilmektedir. İlk başta değerlerin bir önceki hafıza pozisyonlarına yerleştirilmesi ve ilk aşama işlem sonuçlarının bir sonraki hafıza pozisyonlarına konumlandırılmasıyla, aşama sonunda değerler olması gerektiği yerde bulunmaktadır. Sonuçların yazıldığı hafıza pozisyonları bir sonraki adres pozisyonları olduğu için hesaplamalar basit bir sayaç ile oldukça kolay yapılmaktadır.

İlk aşama haricindeki aşamalar için hesaplanan değerlerin bir sonraki okuma/yazma pozisyonu adresi $N/2^s$ ve $N/2^s + 1$ ile hesaplanmaktadır. Bu okuma/yazma işlemi her aşama için $N/2^s + 1$ sonraki pozisyonlara düz ve $N/2^s$ pozisyonlara ters olarak gerçekleştirilmektedir. Düz ve ters okuma/yazma, kelebeklerin a_0 ve a_1 bacaklarından okunan, hesaplaması yapılan değerlerin olması gereken pozisyon adreslerine ya da birbirlerinin yerine yazılması anlamına gelmektedir. Bu işlem, halihazırda FPGA'da bulunan

çoklayıcıların (mux) devrede komütatör görevi üstlenmesiyle yapılmaktadır. Tek bir PE işlem ögesinde komütatör devresi kullanımı gösterimi Şekil 4.18’de verilmiştir. Komütatörlerin ‘0’ olması durumunda düz, ‘1’ olması durumunda ise ters okuma/yazma işlemi yapılmaktadır.

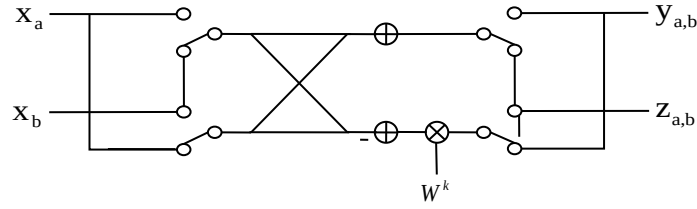


Şekil 4.18. Tek bir PE işlem ögesi için komütatör devresi ile düz okuma/yazma gösterimi.

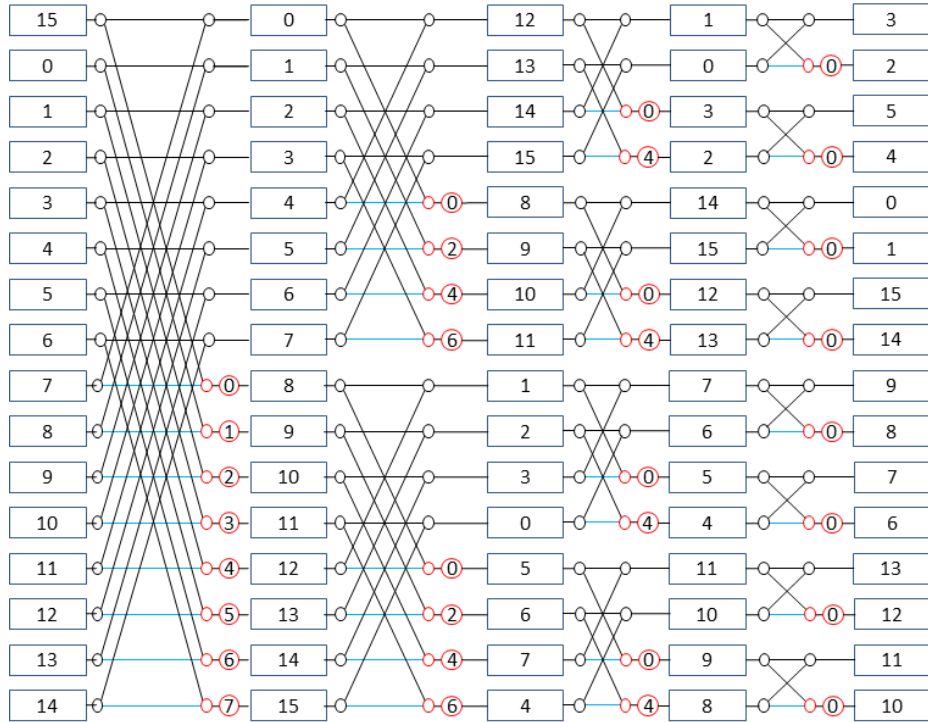
Aşama ilk okuma/yazma işlemleri 0 ve $N/2^{s+1}$ adreslerdeki değerler alınarak ters yazma işlemiyle başlamaktadır. Hangi aşamada ve hangi durumlarda düz veya ters yazma işlemi yapılacağı ise yine ilgili aşamadaki kelebek küme sayısı ile belirlenmektedir. Düz yazma işlemi yalnızca ilgili aşamadaki yazma işlemi sırasının kelebek küme sayısına eşit ve katları olduğu durumlarda gerçekleşmektedir. Örnek olarak, 16-noktalı FFT birinci aşama için küme sayısının (2^s hesaplaması ile) 2 olduğu bilinmektedir. Bu durumda sadece 2. yazma ve onun katlarında (2, 4, 6...) düz yazma işlemi yapılmakta onun haricindeki hafıza yazma işlemleri tersten yapılmaktadır. Bu kural aynı şekilde son yazma işleminde de uygulanabilmekte ancak düz yazma prensibine göre $N/2^s + 1$. adrese değil ilk hesaplama işlemlerinin yapıldığı 0 ve $N/2^{s+1}$. adreslere yazılmaktadır.

Ters yazma işlemi her aşama için farklılık gösteren 2^s ve katları işlemleri hariç tüm hafıza yazma işlemlerinde uygulanmaktadır. Ancak, bu durumun bazı istisnaları mevcuttur. Bu istisnalar, aşama başlarında ters yazma işlemi yapılması gereken ilk $2^s - 2$ işlemin ters olarak okunup FFT hesaplamaları yapıldıktan sonra tekrar ters yazma işleminin yapılmasıdır. Bu işlem PE işlem ögesi öncesinde ve sonrasındaki komütatörlerin ‘1’ olmasıyla gerçekleşmektedir. Bu durum yalnızca ilk düz yazma işlemi gerçekleşene kadar olmakta, ilk yazma işlemi gerçekleştikten sonra uygulanmamaktadır. Ters yazma işlemi yapılması gereken ancak istisna olan bu durumların bir başka istisnası ise aşama ilk

işlemlerinde kullanılan 0. sabit çarpan değeri yerine en son sabit çarpan değerinin kullanılmasıdır. Hafızadan okunan değerlerin ters olarak alındığı, hesaplama yapıldıktan sonra yine ilgili pozisyonlara ters olarak yazıldığını gösteren şematik Şekil 4.19'da verilmiştir. Ayrıca, $N=16$ noktalı FFT için tüm okunan/yazılan hafıza adresleri ve sabit çarpan değerlerini gösteren şematik Şekil 4.20'de sunulmaktadır.



Şekil 4.19. Tek bir PE işlem ögesi için istisnai durum (ilk $2^s - 2$ işlem) komütatör devresi ile ters okuma ve ters yazma işlemi.



Şekil 4.20. $N=16$ noktalı FFT için tüm okunan/yazılan hafıza adresleri ve sabit çarpan ROM adresleri gösterimi.

Kaydediciler FPGA'ler için oldukça düşük maliyetli olduğundan tek bir sayaç kullanmak veya o sinyalden diğer sinyalleri üretmek yerine ilgili mantığın tüm çoklu-bayt mantık sinyallerine atanması tercih edilmiş, böylece kombinatöryel devrelerin azaltılması ve işlem frekansının iyileştirilmesi sağlanmıştır.

N -noktalı FFT için, gerçek zamanlı çalışmalarda depolanması veya hesaplanması gereken $N/2$ karmaşık çarpıcılar bulunmaktadır. Bu çalışmada temel yarım simetri uygulanmaktadır. Yani, herhangi bir sıkıştırma uygulanmamakta ve N -nokta FFT için, $N/2$ boyutunda bellek (ROM) gerekmektedir.

ta (sabit çarpanlar ROM'u adres sinyali) aşama başlarında 0'dan başlayan sayaç olmak üzere yazma işlemi sayısı ilgili aşamadaki küme sayısına ($c = 2^s$) eşit olduğu zaman değeri c kadar artırılmaktadır. Yalnızca istisnai durum olan aşama başlarında ters yazma işlemi yapılması gereken ilk $2^s - 2$ işlemin ters olarak okunup FFT hesaplamaları yapıldıktan sonra tekrar ters yazma işleminde ta değeri c 'ye eşit olmaktadır. $2^s - 2$ okuma/yazmadan sonra ta yine 0 değerinden başlamakta ve c kadar yazma işleminden sonra artırılmaktadır. Anlatılan yöntem ile ta sabit çarpan sayacının artırım değerleri kolaylıkla belirlenebilmektedir.

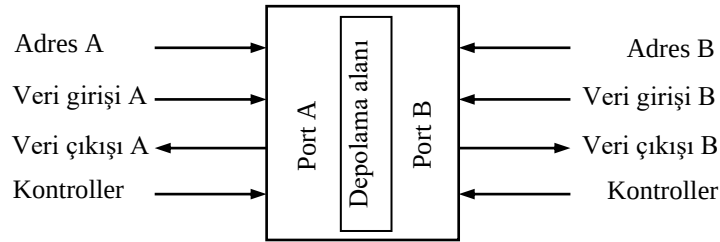
4.5. Çoklu Blok Çift-Port Hafıza-Tabanlı Radiks-2 Hızlı Fourier Dönüşümü Uygulamaları

Çoklu blok yaklaşımlarda, çoklu okuma/yazma işlemleri eşzamanlı olarak gerçekleştirilebilmekte aynı zamanda daha yüksek radikslı algoritmalar uygulanabilmektedir (Chen vd., 2014; Xia vd., 2017; Hsiao vd., 2010). Ancak bazı uygulamalarda birden fazla PE kullanılmakta, fazladan devre elemanı kullanımına ve ek devre karmaşıklığına sebep olabilmektedir. Önerilen algoritmaların çoğunda $2N$, gerçek değerli veriler için (Ma vd., 2015; Ayinala vd., 2013) N boyutta hafıza kullanılmaktadır ve bu durum daha büyük FFT boyutları için kaynak sınırlamalarına varabilecek performans iyileştirmesi için FFT boyutundan daha büyük bellek kullanımıyla sonuçlanmaktadır. Çok blok yaklaşımlarda çift

portlu hafıza ve tek bir PE kullanılarak tüm FFT işlemlerini N kadar hafıza ile gerçeklemek mümkündür. Tez çalışmasında yerinde kullanılabilir yaklaşımla iki adet $N/2$ hafıza ile gerçekleştirilen en düşük kaynak (toplamda N) ve en düşük operasyon saati sayısı ($N/2 \log_2 N$) olan FFT işlemci tasarımı gerçekleştirilmiştir. Ayrıca, normal düzen çıkışlı ve paralel işlem yapan adresleme algoritması ile verilerin giriş ve çıkış süreleri dahil ($N/2 \log_2 N/2$) operasyon saatinde tamamlanmıştır.

4.5.1. Gerçek Çift Portlu Hafıza Yapıları

Gerçek çift portlu hafıza bloklarının kullanımı FPGA ile gerçekleştirilen çalışmalarda geniş yer kaplamaktadır (Cheng ve Keshab, 2007; Garcia vd., 2007; Ma vd., 2015; Luo vd., 2014). Basit çift portlu hafıza modunda, her okuma ve yazma işlemi için özel bir adres portu mevcuttur. Herhangi bir karmaşıklık oluşmadan yazma işlemi A portundan yapılırken B portundan verilerin okuma işlemi yapılabilir. Gerçek çift portlu hafıza modunda ise, okuma ve yazma işlemi için iki ayrı adres portu bulunmaktadır. Bu modda, veriler A portu veya B portunun adresine yazılabilmekte veya okunabilmektedir (Şekil 4.21).



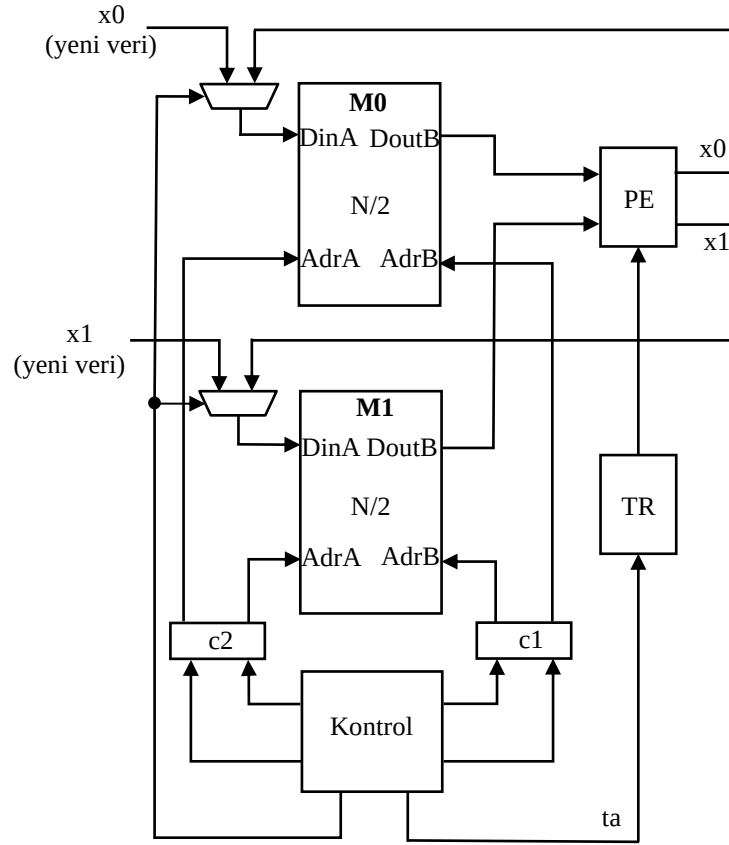
Şekil 4.21. Gerçek çift portlu hafıza okuma/yazma blok diyagramı.

Gerçek çift portlu hafızaların çeşitli çalışma modları bulunmaktadır. Bu modlar “önce okuma”, “önce yazma” ya da “değişiklik yok” modlarıdır. Her iki portun da aynı saat frekansı ile kontrol edilmesi ve porttan okunan verilerin geçersiz olmaması sebebiyle yapılan çalışmalarda “önce okuma” modu kullanılmaktadır.

4.5.2. Radiks-2 Çoklu Gerçek Çift Portlu Hafıza-Tabanlı Hızlı Fourier Dönüşümü Adresleme Algoritması

Çoklu blok radiks-2 FFT çalışmasında toplamda N olmak üzere, bölünmüş ($N/2$) hafıza kullanımı tercih edilmiştir. Başlangıçta verilerin ilk yarısı ($0, \dots, \frac{N}{2} - 1$) birinci hafıza, diğer yarısı ise ($\frac{N}{2}, \dots, N - 1$) ikinci hafıza pozisyonlarına yerleştirilmektedir. Tasarlanan adresleme algoritması gereği her iki hafızanın da A portları yalnızca veri yazma, B portları ise yalnızca veri okuma işlemi için kullanılmaktadır. Tek bir kelebek devre elemanı söz konusu olduğundan ve aynı anda iki okuma/yazma işlemi yapıldığından kelebek alt ve üst bacak adreslerinin farklı hafızalarda olması gerekmektedir. Kelebek işlemleri yapılacak veriler hafızaların B portlarından okunmakta ve sonuçlar aynı adresli A portlarına yazılmaktadır. Geliştirilen adresleme algoritmasıyla son aşama FFT okuma işlemleri yapılırken eş zamanlı olarak aynı adreslere yeni verilerin yazılma işlemi de yapılabilmektedir. Böylece yeni gelen verilerle yapılacak FFT hesaplamaları için fazladan zaman ihtiyacı ortadan kalkmakta ve bir önceki FFT hesaplamalarının tamamlanmasıyla yeni FFT işlemlerine başlanabilmektedir. Ancak normal düzenli çıkış elde edilemediğinden verilerin daha önceden bir hafızaya alınması ihtiyacı doğmaktadır. İki adet basit sayaç ($c1$ ve $c2$) ile hafıza adreslerinin hesaplandığı, hafıza okuma/ yazma ve kelebek işlemleri kontrol devresini gösteren blok diyagram Şekil 4.22’de verilmektedir.

FPGA içerisindeki gerçek çift portlu Rastgele Erişimli Belleklerde (RAM) verilerin hafızadan okunma ve yazılma adreslerinin aynı olma zorunluluğu bulunmamaktadır. Bu özellikten faydalanılarak yeni adresleme algoritması tasarlanmıştır. Önerilen yöntem ile bir kelebek devresinin tek saat darbesi süresinde eşzamanlı hafızadan okuma ve hafızaya yazma işlemi gerçekleştirilmektedir. Bölünmüş hafızaya dayalı FFT’nin en az saat döngüsünde hesaplama işlemi, daha önce hesaplanmış verilerin aynı anda belleğe yazılırken işlenecek yeni verilerin okunması ile sağlanmaktadır.



Şekil 4.22. FFT okuma/ yazma ve kelebek işlemleri kontrol devresi.

İlk FFT aşaması için giriş verileri $(x_0, \dots, x_{N/2-1})$ ve $(x_{N/2}, \dots, x_{N-1})$ önceden $M0$ ve $M1$ hafızalarına sırasıyla yazılmaktadır. İlk saat darbesiyle $M0$ ve $M1$ 'in ilk hafıza pozisyonlarındaki değerler okunmaya başlanır. Bir sonraki saat darbesinden önce kelebek devre elemanları çıkış değerleri bir sonraki hafıza adresine yazılmak için hazır bulunmaktadır. Aynı zamanda adres değerleri sıradaki okuma pozisyonu için güncellenmektedir. Bu okuma/ yazma işlemleri ilk FFT aşaması için tüm işlemler tamamlanana kadar devam etmektedir. Bu yaklaşımla her bir hesaplama işlemi tek saat darbesinde ve ilk aşama işlemleri $N/2$ saat darbesinde tamamlanmaktadır. İlk aşama işlemlerinin $N/2$ saat darbesi süresinde tamamlanmasıyla, diğer aşama işlemlerine fazladan saat darbesi ihtiyacı olmadan başlanabilmekte, aynı çalışma prensiyle ve aynı sürede tamamlanmaktadır. Yapılan işlemler ve adresleme hesaplaması aynı ancak işlem yapılacak hafıza adres pozisyonları farklılık göstermektedir. Bu durum hem DIF FFT yapısına (Şekil 4.2) göre kelebek bacakları arasındaki ofset değerlerinin farklılığından hem de bu bacakların

iki ayrı hafızadan okunma zorunluluğundan kaynaklanmaktadır. Bu zorunluluk sebebiyle bazı hafıza yazma pozisyonlarında komütatör devresi kullanılmaktadır.

Bulunan adresleme algoritmasıyla FFT hesaplamaları, tek aşama kelebek işlemleri için $\frac{N}{2}$ olmak üzere tüm aşamalar için $\frac{N}{2} \log_2 N$ saat döngüsünde tamamlanmaktadır. Gerçek zamanlı uygulamalarda sürekli veri akışı olmakta ve bu verilerin sırasıyla hafıza pozisyonlarına yerleştirilmesi ve çıkışta da normal sırayla olması önem arz etmektedir. Ancak radiks-2 FFT DIF yapısına göre çıkışlar ters-bit düzeniyle verilmektedir. Yapılan çalışmada geliştirilen adresleme algoritmasıyla çıkışlar sıralı düzenle verilmemekte ancak yeni gelen verilerin önceden farklı bir hafızaya yerleştirilmesiyle son aşama işlemleri yapılırken ilgili pozisyonlara yazılması sağlanmaktadır. İlk FFT işlemlerinden sonra yeni gelen verilerin ilk FFT son aşama işlemlerinde hafıza adreslerine yerleştirilmesiyle bir sonraki FFT işlemlerinde hafızaya verilerin yazılması için gereken süre ortadan kalkmaktadır. Elde edilen bulgular, bir önceki çalışmada elde edilen tek bir PE ve tek blok hafıza ile gerçekleştirilen $(\frac{N}{2} + 1) \log_2 N$ saat döngüsünde tamamlanan çalışmadan oldukça iyidir.

Şekil 4.22’de blok şemada gösterilen kontrol devresinde ikili sayı tabanında sayaçlar kullanılmaktadır. Sayaçlar 0’dan $\log_2(N) - 1$ ’e kadar sayan aşama değeri s , kelebek üst ve alt bacak adreslemelerini kontrol eden $c1$ ve $c2$ ve sabit çarpanlar değeri ta ’dır. Hafızaların iki ayrı okuma adresi ra_0 ve ra_1 ve hafızaya yazma adresleri wa_0 ve wa_1 , $c1$ ve $c2$ sayaçları ile üretilmektedir. Her aşamada her biri $N/2^{s+1}$ kelebekten oluşan 2^s kelebek kümesi bulunmaktadır.

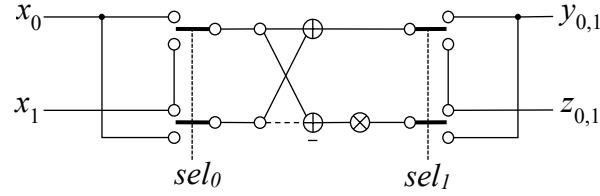
Şekil 4.23 16-noktalı FFT için her hesaplama adımında bu sayaçların değerini göstermektedir. ra_0 ve ra_1 PE’nin üst ve alt bacaklarına bağlı olan hafıza portlarının okuma adres satırlarını, wa_0 ve wa_1 ilgili aşamadaki yazma ve bir sonraki aşama okuma adreslerini, ta ise sabit çarpanlar (twiddle) ROM’u içinde her pozisyon için saklanan W^t sabit çarpan adreslerini ifade etmektedir. Satır değerleri, saat darbesinin yükselen kenarında, sütun başlıkları tarafından uygun şekilde gösterildiği gibi, s , ra_0 , ra_1 , wa_0 , wa_1 , ve t gibi parametrelerin değerleri de gösterilmektedir.

$s = 0$					$s = 1$					$s = 2$					$s = 3$				
ra_0	ra_1	wa_0	wa_1	ta	ra_0	ra_1	wa_0	wa_1	ta	ra_0	ra_1	wa_0	wa_1	ta	ra_0	ra_1	wa_0	wa_1	ta
0	8	0	8	0	0	12	0	12	0	0	14	0	14	0	0	15	0	15	0
1	9	1	9	1	1	13	1	13	2	1	15	15	1	4	14	1	14	1	0
2	10	2	10	2	2	14	14	2	4	12	2	12	2	0	12	3	12	3	0
3	11	3	11	3	3	15	15	3	6	13	3	3	13	4	2	13	2	13	0
4	12	12	4	4	8	4	8	4	0	8	6	8	6	0	8	7	8	7	0
5	13	13	5	5	9	5	9	5	2	9	7	7	9	4	6	9	6	9	0
6	14	14	6	6	10	6	6	10	4	4	10	4	10	0	4	11	4	11	0
7	15	15	7	7	11	7	7	11	6	5	11	11	5	4	10	5	10	5	0

Şekil 4.23. $N=16$ için üretilen sabit çarpanlar ROM ve üretilen veri hafıza adresleri.

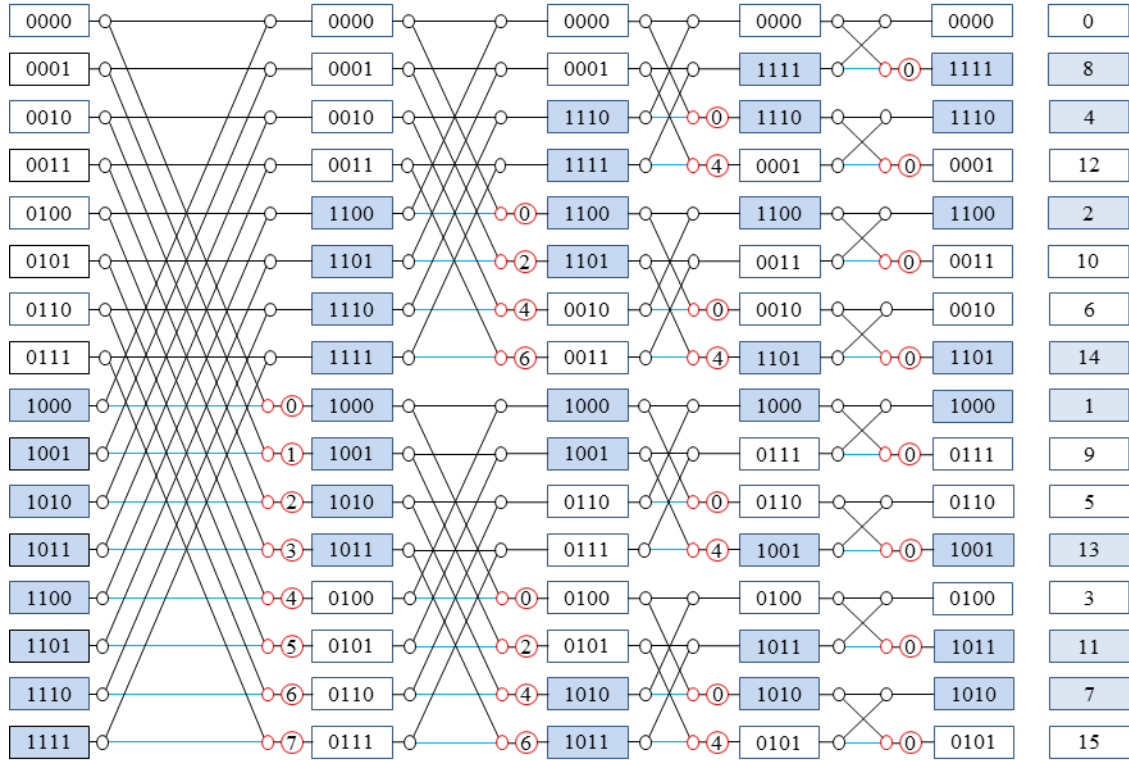
İlk aşama işlemleri giriş verilerinin sırasıyla hafıza pozisyonlarına yerleştirildikten sonra ilk okuma adreslerinin hesaplanmasıyla başlamaktadır. Tüm aşamalar için kelebek üst bacak okuma pozisyonu ra_0 , her aşama için 0 'dan $\frac{N}{2} - 1$ 'e sayan $c1$ sayacının değerini taşımaktadır. Alt bacak okuma değeri ra_1 ise ra_0 'ın s kadar en ağırlıklı bit değerinin tersine çevrilmiş halidir. Aslında her aşama en ağırlıklı bitin değeri hangi hafızanın kullanıldığı hakkında da bilgi vermektedir. Öyle ki, bulunan adresleme algoritmasında '0' ilk hafızayı, '1' ise ikinci hafızayı temsil etmektedir. Sayıcıları saklamak yerine aritmetik işlemleri (toplama) kullanarak ra_0 ve ra_1 adreslerini hesaplamak, gerçek veri genişlikleriyle karşılaştırıldıklarında daha küçük olduğu için ve toplama işlemleri için sentezleyiciler basit kombinatoriyel lojik devreleri kullandıkları için daha kullanışlıdır. Yazma adresleri wa_1 ve wa_1 , ilk $N/2^{s+2}$ yazma pozisyonu için okuma adresleriyle aynı olmakta ancak son $N/2^{s+2}$ yazma pozisyonunda adres değerleri birbirleriyle yer değiştirmektedir. Yer değiştirme işlemlerinin yapıldığı hafıza adresleri Şekil 4.23'te renkli olarak ifade edilmiştir. Yazma adreslerinin ters yapılmasından dolayı bir sonraki aşama okuma işleminin de ters yapılması zorunluluğu doğmakta ve komütatör devresi kullanımı söz konusu olmaktadır. Ters okuma/yazma işlemleri daha önce de belirtildiği üzere bir sonraki aşama işlemleri yapılırken okuma/yazma adreslerinin farklı hafızadan olması gerekliliğindendir. Yazma adresleri $c2$ sayacıyla üretilmekte ve bu sayaç aynı zamanda son aşama işlemlerinde okunan verilerin yerine yeni verilerin yazılması işlemi için de kullanılmaktadır. Adres değişim

işlemleri FPGA’lerde halihazırda bulunan komütatör devrelerinin kullanılmasıyla kolaylıkla yapılabilmektedir. Bu devre *sel* kontrol sinyalleriyle aktif hale getirilmektedir (Şekil 4.24).



Şekil 4.24. Tek bir PE işlem ögesi için komütatör devresi.

Açıkçası, bahsi geçen adresleme devresini farklı algoritmalarla ile gerçeklemek de mümkündür. Kaydediciler FPGA’ler için oldukça ucuz maliyetlidirler. Bu nedenle, tek bir sayaca sahip olmak ve ondan tüm diğer sinyalleri üretmek yerine, kombinatoriyal devrelerin azaltılması ve sonuç olarak operasyon frekansının iyileştirilmesi için kendi sayaçlarının ve ilgili mantığın tüm çoklu-bayt mantık sinyallerine atanması tercih edilmektedir. Tasarım kriterleri ne olursa olsun, PE kombinatoriyal mantığının, karmaşık toplayıcıları içereceği ve istenen hassasiyete sahip karmaşık çarpıcılar içereceği için muhtemelen tüm FFT tasarımı için kritik olduğu söylenebilmektedir. Bu nedenle, daha yüksek saat frekansları için PE geliştirilmedikçe, tasarımın geri kalanının çalışma frekansı üzerinde fazla etkisi olmadığı sonucuna varılabilmektedir. Başlangıçtan son aşamadaki son keleşin tamamlanmasına kadar geçen saat darbesi sayısı benzer yapıdaki FFT işlemci mimarileriyle karşılaştırılmaktadır. Benzer şekilde, çip üzerindeki kullanılan alanın büyük bir kısmı büyük N için beklendiği gibi hafıza tarafından işgal edildiğinden, verimli bellek kullanımı (ayrık kaydediciler hariç) aynı şekilde karşılaştırılabilmektedir. Önerilen tasarım okuma/yazma adresleri ikilik sistemde hesaplanmış gösterimi Şekil 4.25’te verilmiştir.



Şekil 4.25. $N=16$ noktalı FFT için tüm okunan/yazılan hafıza adresleri ve sabit çarpan değerleri gösterimi.

Tasarımda devreye sağlanan sabit çarpanlar ta (sabit çarpanlar ROM'u adres sinyali) aşama başlarında 0'dan başlayan okuma adreslerini belirleyen $c1$ sayacıyla hesaplanmaktadır. Aşamanın değişmesi ile sabit çarpanlar ikilik sayı sistemindeki sayaç bitlerinin en az ağırlıklı s kadar bitinin '0' olmasıyla oluşturulur. 16-noktalı FFT için devreye sağlanan sabit çarpan değerlerini kontrol ta 'nın VHDL ile gösterimi aşağıda verilmiştir (Şekil 4.26).

```
ta <= (c1(2), c1(1), c1(0)) when s="00" else
      (c1(2), c1(1), '0') when s="01" else
      (c1(2), '0', '0') when s="10" else
      ('0', '0', '0');
```

Şekil 4.26. $N=16$ için ta adreslerinin $c1$ sayacından VHDL ile üretilmesi.

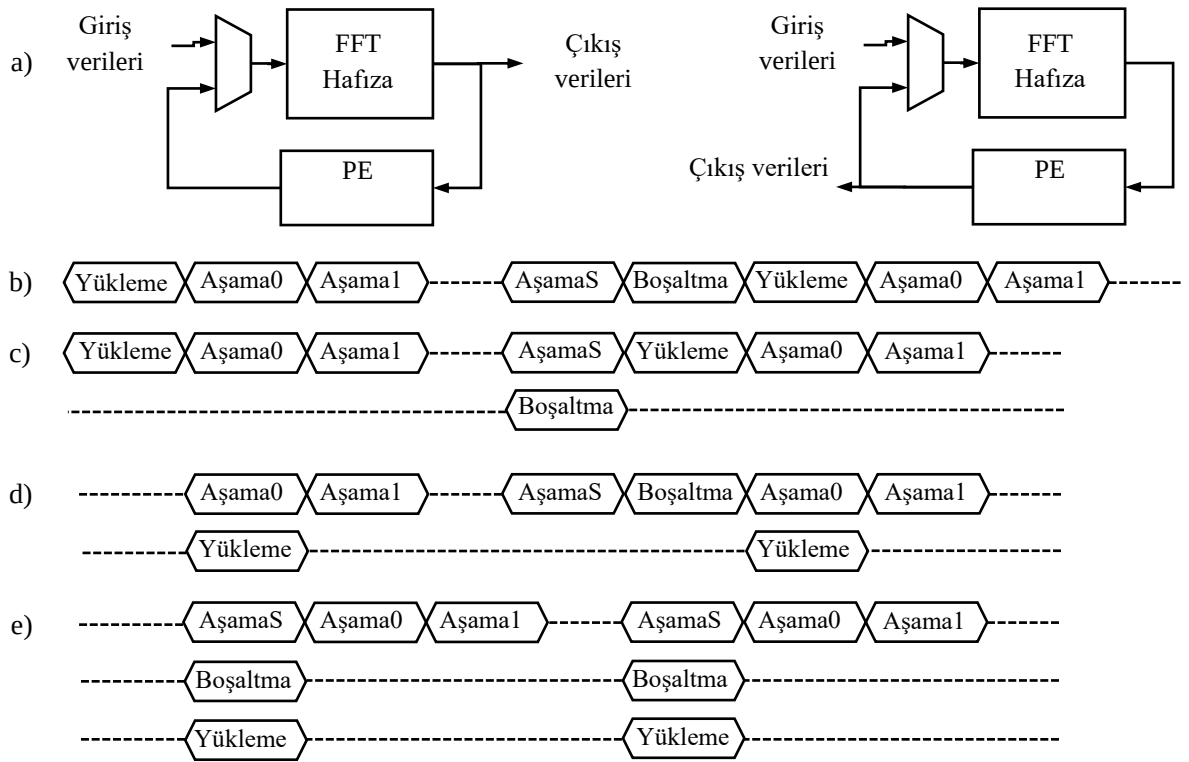
4.5.3. Paralel Giriş/Çıkış İşlemleri Sıralı Yerinde Kullanılabilir Radiks-2 Hızlı Fourier Dönüşümü Adresleme Metodu

Çoklu blok radiks-2 FFT çalışmasında toplamda N olmak üzere, bölünmüş ($N/2$) hafıza kullanımı tercih edilmiştir. Başlangıçta veriler iki ayrı hafıza bloğuna sırayla yazılmaktadır. Tasarlanan adresleme algoritması gereği hafızaların A ve B portları hem veri okuma hem de yazma işlemi için kullanılmaktadır. Tek bir kelebek devre elemanı söz konusu olduğundan ve aynı anda iki okuma/yazma işlemi yapıldığından kelebek alt ve üst bacak adreslerinin farklı hafızalarda olması gerekmektedir. Bu çalışmada giriş verilerinin taşınabildiği ve son aşama işlenirken önceki hesaplamaların sonuç verilerinin paralel olarak okunabildiği basit, karmaşık değerli bir radiks-2 bellek tabanlı Hızlı Fourier Dönüşümü mimarisi önerilmiştir. Bunun için, ara verileri normal sıralı adresleme sırasına koyan bir adresleme şeması kullanılmakta, böylece FFT çıktı verileri bit ters sırada değil, normal düzen sırasında olmaktadır. Bu nedenle, son aşamadaki işleme elemanının iki okuma adresi, giriş verilerinin birinci ve ikinci yarısından olma özelliğine sahiptir, böylece işlem sıralı bir şekilde yapıldığında, son aşama işlemleri sırasında yeni giriş verileri yüklenebilmektedir. Böyle bir yaklaşım, sıralı çıktı gerektiren uygulamalar için önemli sayıda saat döngüsü tasarrufu sağlamakta ve paralel işlem yapabilme özelliğiyle yeniden sıralama için ek kaynak gerektirmemektedir.

Tek bir PE devre elemanı ile radiks-2 hafıza-tabanlı FFT işlemlerinin tamamlanması için gereken süre, önceden verilerin hafızalara yerleştirilmesi ile (optimum çalışma süresi) $N/2 \log_2 N$ saat darbesidir. Ancak, son FFT işlemlerinin tamamlanmasıyla çıkış verilerinin alınması ve bir sonraki FFT işlemlerinin başlaması için yeniden verilerin hafıza pozisyonlarına yerleştirilmesi ek süre ihtiyacını doğurmaktadır. N_{bh} bölünmüş hafıza bloğu sayısı olmak üzere, tüm veri giriş ve çıkış işlemleri için toplamda $2N/N_{bh}$ saat darbesi gerekmektedir. Giriş ve çıkış verileri işlemlerinin paralel hareketi sayesinde bu sürenin N/N_{bh} 'ye azaltılması mümkündür. Böylece sürekli akış gerçek zamanlı uygulamalarda toplam zaman ihtiyacı $N/2 \log_2 N + N/N_{bh}$ operasyon saatidir. Son aşama FFT aşama veri yazma ve bir sonraki FFT ilk aşama veri okuma işlemlerinin de paralelleştirilmesiyle daha fazla saat darbesi tasarrufu sağlanmaktadır.

Bu çalışmada önerilen adresleme algoritması ile, FFT son aşama işlemleri veri hareketi (giriş/çıkış) işlemleriyle paralel gerçekleştirilmekte bağımsız tüm FFT döngüleri arasında beklemeyi ortadan kaldırmakta ve genel performansı artırmaktadır. Önerilen radiks-2 FFT adresleme algoritması FFT boyutu $N = 2^{2n+1}$ ($N=32, 128, 512, \dots$) boyutuna genişletilebilmekte ve tüm FPGA'lerde uygulanabilmektedir.

Birçok araştırmacı paralellik özelliğini geliştirerek çeşitli hafıza adresleme şemaları önermiş ve kaynak kullanımını azaltan, performans iyileştirici çalışmalar sunmuşlardır. Şekil 4.27'de temel hafıza-tabanlı ve çeşitli paralellik yaklaşımları zamanlama diyagramları sunulmuştur.

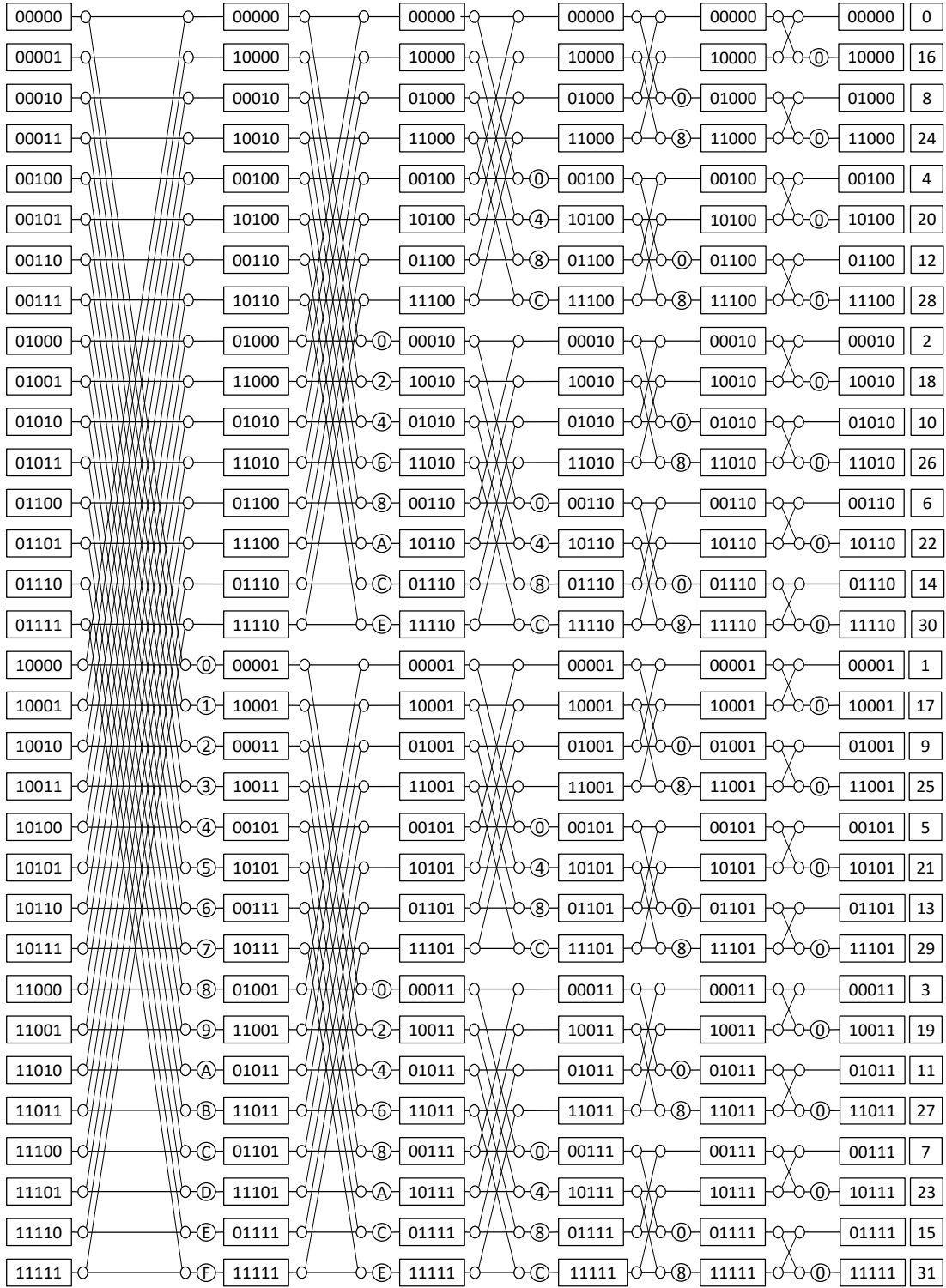


Şekil 4.27. a) Hafıza-Tabanlı FFT genel yapısı (sol) – Modifiyeli paralel g/ç işlemli Hafıza-Tabanlı FFT yapısı (sağ) **b)** Yükleme-işlem-boşaltma kavramsal zamanlama diyagramı **c)** Son aşama işlemleri ve veri çıkışı (boşaltma) paralel olma zamanlama diyagramı **d)** Yükleme ve ilk FFT işlemleri paralel olma zamanlama diyagramı **e)** Yükleme ve boşaltma ile son aşama FFT işlemleri paralel olma zamanlama diyagramı.

Şekil 4.2’de verilerin okunduğu hafıza adresine yazılmasıyla çıkış verilerinin ters-bit düzeninde üretildiği görülmektedir. Ters-bit düzeninde çıkışın elde edilmesi, hesaplanan sonuçların doğal sıralı olarak okunmasına ve aynı anda yeni verilerin doldurulmasına izin vermemektedir. Bu nedenle, çıktı verilerini normal sıralı adres sırasında üreten bir adresleme şeması, giriş/çıkış ve FFT işlemlerinin paralelleştirilmesi için yardımcı olmaktadır. Hem giriş hem de çıkış verileri aynı hafıza adresinde olduğunda, çıkış verileri normal düzen sırasında okunurken giriş verileri FFT belleğe doldurulabilir. Bu nedenle, çıkış verilerini normal sıralı adres düzeninde üreten bir adresleme şeması, giriş/çıkış ve FFT işlemlerini paralel hale getirmek için yardımcı olacaktır. Hem giriş hem de çıkış verileri aynı sırada olduğunda, çıkış verileri doğal sırada okunurken giriş verileri belleğe doldurulabilmektedir. Birçok araştırmacı paralellik özelliğini geliştirerek çeşitli hafıza adresleme şemaları önermiş ve kaynak kullanımını azaltan, performans iyileştirici çalışmalar sunmuşlardır. (Tian vd., 2018). Önerilen adresleme şeması $N = 32$ -noktalı radiks-2 FFT için Şekil 4.28’de verilmektedir. Şemada gösterildiği üzere hem giriş hem de çıkış verileri sırası hafıza adreslerinin normal sırasıyla aynıdır. Bu yöntem ile ilk iki aşama sonunda verilerin sıralı düzen elde edilmesiyle son iki aşama işlemleri oldukça kolay yapılmaktadır. Bu çalışmada önerilen yöntem, FFT son aşama işlemlerini veri hareketi (giriş/çıkış) işlemleriyle paralelleştirerek bağımsız tüm FFT döngüleri arasında beklemeyi ortadan kaldırmakta ve genel performansı artırmaktadır.

Çalışmada sunulan “kendiliğinden sıralı yerinde kullanılabilir” algoritma, kısıtlı donanıma sahip FPGA’ler için yazılımsal çözüm sunmaktadır. Kendiliğinden sıralanabilir algoritma şemasında $S = \log_2(N)$ adres bitleri sayısını ve radiks-2 FFT için aşama numarasını belirlemektedir. Radiks-2 kelebek yapısına göre iki ayrı okuma adresi olan R_0 ve R_1 , aşama numarası ($s = 0 \dots S - 1$) tarafından belirlenen ve yalnızca bir bitin değişiklik göstermesiyle denklem (4.4)’e göre hesaplanmaktadır.

$$\begin{aligned} R_0 &= a_{S-1}a_{S-2} \cdots a_1a_0 \\ R_1 &= b_{S-1}b_{S-2} \cdots b_1b_0 \end{aligned} \quad (4.4)$$



Şekil 4.28. $N=32$ nokta FFT sıralı düzen çıkış ve hafıza adresleme yapısı.

Örnek olarak, DIF FFT uygulamalarında tek bit farklı okuma adresleri ilk aşamada $0a_{S-2} \cdots a_1a_0$ ve $1a_{S-2} \cdots a_1a_0$, ikinci aşamada $a_{S-1}0 \cdots a_1a_0$ ve $a_{S-1}1 \cdots a_1a_0$ olmakta ve

sonraki aşamalarda benzer şekilde devam etmektedir. Ancak bu adreslerden okunan verilerin hesaplama sonuçlarının yazıldığı hafıza adresleri farklı olan bitin simetrisinde bulunan en az ağırlıklı bit (LSB) ile değişmesiyle bulunmaktadır. Yani ilk aşama yazma adresleri $a_0 a_{s-2} \dots a_1 \mathbf{0}$ ve $a_0 a_{s-2} \dots a_1 \mathbf{1}$, ikinci aşama yazma adresleri $a_{s-1} a_1 \dots \mathbf{0} a_0$ ve $a_{s-1} a_1 \dots \mathbf{1} a_0$ olacaktır. Okuma ve yazma adreslerindeki düzenlemeler ile $S_h = \text{int}(S/2)$ aşama sonrasındaki tüm aşama işlemlerinde veriler okunduğu adreslere yazılabilmektedir. Ancak $s \leq S_h$ için, yazma konumlarındaki veriler henüz okunmamış ve kullanılmamış olabileceğinden, bu permütasyonlar aşama tamamlandıktan sonra gerçekleştirilmek zorundadır. Permütasyonlar, bu kendi kendini sıralayan yerinde algoritmanın naif donanım uygulamalarında her zaman ek saat döngüleri gerektirecektir.

Tüm radiks-2 PE operasyonları için $(S - 1 - s)$. bit pozisyonlarında farklılık gösteren iki ayrı hafıza adresinden iki ayrı okuma yapılmaktadır. Kalan $S - 1$ adres bitleri $(0, 1, \dots, N/2)$ arasında değer alan b sayacı tarafından üretilmektedir. Farklılaşan bit değerleri “0” ve “1” in b sayacı tarafından üretilen adres değerlerinin arasına uygun pozisyonlara yerleştirilmesiyle okuma adresleri elde edilmektedir. Yazma adresleri (W_0 ve W_1), bu okuma adreslerinin s inci ve $(S-1-s)$. bitleri değiştirilerek oluşturulmaktadır. Bu bitler, okuma adresi oluşturma sırasında yerleştirildikleri için mevcut değerleri bilindiğinden, sadece uygun bit değerleriyle değiştirilebilmektedir.

Bir yazılım uygulamasında bellekten birden fazla sıralı okuma yapılabilen ve bu “tek bir yineleme adımı” olarak adlandırılmaktadır. Öte yandan, “tek adım” genellikle donanım uygulamalarında “tek bir saat döngüsü” anlamına gelmektedir. Bu nedenle, donanım uygulamalarında, bellek bloklarının kullanılabilir bağlantı noktalarının sayısı da her zaman dikkate alınmalıdır. Örneğin, (Marti-Puig, 2009)’un tartışılan sıralı g / χ yöntemi, tek bir PE işlemi için iki bellek okuma ve iki bellek yazma gerektirmekle birlikte okuma ve yazma için farklı bellek adreslerine sahip olduğu görülmektedir. Bu, bir donanım uygulamasında tek bir saatte bir PE kelebek işlemi gerçekleştirilecekse, minimum dört portlu bir bellek bloğuna ihtiyaç duyulduğu anlamına gelmektedir ki bu tasarımcıların ulaşamayacağı bir kaynaktır. Bu nedenle tasarımcılar, çoklu bellek bloklarını birden çok bellek bloğu kullanarak oluşturmalı ve olası erişim çakışmalarını önlemek için bağlantı noktalarını dikkatlice yönetmelidir.

Hafıza kullanan sayısal tasarım çalışmalarında dikkate alması gereken bazı sınırlamalar bulunmaktadır:

1. Önce-Okuma (Read-First) modu kullanılmadıkça bir hafıza bloğundan yapılan okuma/yazma sayısı o hafızanın port sayısını geçmemelidir.
2. Aşama geçişlerindeki sınırlamalara (bir önceki aşamanın son verilerini yazarken ve aynı anda o aşamadaki ilk verileri okuma işlemi yapılırken), belleğe erişme sırası esnasındaki çakışma problemlerine dikkat edilmesi gerekmektedir.
3. Eğer g/ç işleminin paralel olması amaçlanıyorsa, PE operasyon sırası bir DIF FFT'nin (veya bir DIT FFT'nin ilk aşamasındaki veri giriş sırası) son aşamasındaki veri çıkış sırası ile belirlenmelidir.
4. Aynı saat darbesinde PE okuma ve yazma işlemlerinin yapılabilmesi mümkün olmadığından yazma işleminin okuma işleminden önce ardışık düzen (pipeline) kaydediciler kullanılmıyorsa bir saat darbesi sonrası, tek bir ardışık düzen kaydedici kullanılıyorsa iki saat darbesi sonrası yapılması unutulmamalıdır.

Bu adresleme algoritmasında performans nedenleriyle, karmaşık toplayıcıları izleyen pipeline kaydediciler kullanılmıştır. Böylece yazma işlemleri kullanılan pipeline yapısı sayısı kadar saat darbesi gecikmeli yapılabilmektedir. Denklem (4.5) pipeline kaydedicilerin kullanıldığı ve denklem (4.4)'e göre hesaplanmış olan, üssel ifadelerin saat darbesi numarasını gösterdiği, ilk aşama okuma ve yazma adreslerini göstermektedir.

$$\begin{aligned} R_0^t &= 0a_{S-2} \cdots a_1 a_0, R_1^t = 1a_{S-2} \cdots a_1 a_0 \\ W_0^{t+2} &= a_0 a_{S-2} \cdots a_1 0, W_1^{t+2} = a_0 a_{S-2} \cdots a_1 1 \end{aligned} \quad (4.5)$$

Yazma işlemi okuma işleminden iki saat darbesi gecikmeli yapılmaktadır. $a_0 = 0$ ise $R_0^t = W_0^{t+2}$ ve $a_0 = 1$ ise $R_1^t = W_1^{t+2}$ 'dir. Yazma adresindeki verilerin kaybolmaması için yazma işlemi yapılmadan önce okunması gerekmektedir. Bir sonraki saat darbesinde bu veriler okuma için hazır bulunmaktadır. Koruma için okunan bu verilerin okuma ve a_0 bitinin yer değiştirmesiyle hesaplanan yazma adresleri denklem (4.6)'te verilmektedir.

$$\begin{aligned} R_0^{t+1} &= 0a_{s-2} \cdots a_1 \bar{a}_0, R_1^{t+1} = 1a_{s-2} \cdots a_1 \bar{a}_0 \\ W_0^{t+3} &= \bar{a}_0 a_{s-2} \cdots a_1 0, W_1^{t+3} = \bar{a}_0 a_{s-2} \cdots a_1 1 \end{aligned} \quad (4.6)$$

$\{R_0^t, R_1^t, R_0^{t+1}, R_1^{t+1}\}$ dörtlü okuma ve $\{W_0^{t+2}, W_0^{t+3}, W_1^{t+2}, W_1^{t+3}\}$ dörtlü yazma kümeleri aslında aynı adreslere sahiptirler. PE'ler aynı $a_{s-2} \cdots a_1$ adres bitli çiftler olarak ele alındığında kalan PE'ler bu adreslerden bağımsız olarak hareket edebilmektedir. İlk aşamadaki ilk iki PE örnek olarak gösterilecek olursa, okuma ve yazma adresleri $R_0^t = 0 \dots 0$, $R_1^t = 1 \dots 0$, $R_0^{t+1} = 0 \dots 1$, $R_1^{t+1} = 1 \dots 1$, $W_0^{t+2} = 0 \dots 0$, $W_1^{t+2} = 0 \dots 1$, $W_0^{t+3} = 0 \dots 1$ ve $W_1^{t+3} = 1 \dots 1$ 'dir. Adreslerdeki “...” ifadesi aynı değeri taşımaktadır. Tüm olası MSB ve LSB kombinasyonlarına sahip adresler (bunlardan 4 tanesi) işlendiğinden (okunup yazıldığından), kalan bitlerin tümü aynı olduğundan, PE'lerin geri kalanının bu adreslerdeki verilerle hiçbir ilgisi yoktur.

S_h aşamasına kadar tüm aşamalar bitlerin konumları s ve $S - 1 - s$ 'e bağlı olarak aynı prensiple çalışmaktadır. PE çiftlerinin belirli bir aşamada gerçekleştirilme sırası, yalnızca iki blok hafızayı gerektirecek şekilde seçilebilmektedir. $s < S_h$ aşamaları için PE çiftleri okuma/yazma adreslerinin genel formülü denklem (4.7)'de verilmektedir.

$$\begin{aligned} R^{t+2}: x0yBz, x1yBz, W^{t+2}: uAw0v, uAw1v \\ R^{t+3}: x0y\bar{B}z, x1y\bar{B}z, W^{t+3}: u\bar{A}w0v, u\bar{A}w1v \end{aligned} \quad (4.7)$$

$\{0,1\}$ olarak gösterilen bitler ve büyük harfler $\{A,B\}$, merkez bit (ler) ten eşit uzaklıkta konumdadır. Kalan $\{x,z,u,v\}$ dizisi 0'dan $S_h - 1$ aşamasına kadar, $\{y,w\}$ $S - 2$ 'deki bit pozisyonlarını temsil etmektedir. m_0 olarak isimlendirilen ve okuma ve yazma adreslerinde p. bite karşılık gelen bit hafıza seçimi için kullanılmaktadır. $N/2$ boyutlu iki ayrı hafıza kullanımı söz konusu olduğundan kendiliğinden sıralanabilir algoritmalarda p 'nin en az bir aşamada $\{0,1,X,\bar{X}\}$ pozisyonuna karşılık gelmesi yani bir hafıza için üç erişimde bulunması söz konusudur. Örneğin, eğer $B = 0$ ise, denklem (4.7)'deki yazma adresi 0 ve 1 olmakta, bu da bir bellek bloğuna üç erişimde bulunmak demektir. Benzer bir çakışma $B = 1$ olduğunda da meydana gelmektedir. Bu nedenle, (Chu ve George, 1998;

Johnson ve Burrus, 1984) tarafından açıklanan orijinal algoritmayı kullanan donanım uygulamasının yalnızca iki bellek bloğu ile yapılamayacağını kanıtlamaktadır. Diğer yandan, kendinden sıralanabilir algoritma uygulamaları iki bellek bloğu ile birkaç değişiklik yapılarak gerçekleştirilebilir. Bunlar:

1. Yazma permütasyonlarında kullanılmayan tek bit olduğu için orta adres biti bellek bloğu seçim biti olarak kullanılmalıdır. Bu durumda uygulama tek boyutlu adreslere sahip FFT'ler için sınırlıdır (örneğin 32, 128, 512, 2048, ...).

2. Ekstra saat döngüleri (bir pipeline kaydedici kullanıldığında iki adet), $s = S_h$ aşamasının başlangıcına ve bitimine, bu aşamadaki adresleme stratejisi değişiklikleri ve bellek bloğu seçim biti etkilendiği için eklenmelidir.

3. Son aşamanın doğal işlem sırasına göre yapılması gerektiğinden, hafıza bloğu seçim biti kontrol edilemez. Bu nedenle, veri girişi, veri çıkışı ve son aşama işlemenin paralel olarak yapılabilmesi için blok hafızalarının önce okuma (READ_FIRST) modu etkinleştirilmelidir.

Adresleme algoritması tüm yapılan işlemleri aşağıdaki gibi sıralayabiliriz:

1. 0'dan $S - 1$ 'e aşama numarasını sayan s sayacı ve her aşamada $b_{S-2} \dots b_0$ bitleri için 0'dan $N/2 - 1$ 'e kadar b sayacı kullanılmaktadır.

2. b_1, b_{S_h} ile değiştirilmektedir, böylece sonraki PE çifti adreslerinin farklı bellek bloklarından gelmesi sağlanmaktadır. Bu son aşamada için gerekli değildir çünkü son aşama g / φ paralelliği elde etmek için doğal sırayla gerçekleştirilmelidir. $s > S_h$ aşamalarında bu işlem gerekli değildir.

3. PE çiftlerinin adreslerinin ardışık olarak üretilmesi gerektiğinden, LSB (b_0) bit (b_s) ile değiştirilmektedir. Bu son aşamada yapılmamaktadır.

4. b 'nin değiştirilmiş / yeniden sıralı bitlerini kullanarak, bellek adresleri R_0, R_1, W_0 ve W_1 daha önce tarif edildiği gibi oluşturulmaktadır (0 ve 1 uygun konumlara ekleyerek). W_0^{+2} ve W_1^{+2} kaydediciler kullanılarak W_0 ve W_1 'in geciktirilmesiyle elde edilmektedir.

5. Tüm R_0, R_1, W_0^{+2} ve W_1^{+2} 'deki a_{S_h} biti hafıza arayüz devresinde blok seçiciler olarak kullanılmaktadır. Bu adresler haricindeki bitler, a_{S_h} tarafından hafıza bloklarının adres girişlerine yönlendirilmektedir.

Yukarıda maddeler halinde verilen algoritma çalışma prensibi sayısal devre uygulama (sayaçlar ve adımlar) biçimi aşağıda verilmektedir:

Sayaç: $b = b_{N_b-1} \dots b_{S-2} b_{S-3} \dots b_0$ sayacı toplamda $N_b = S - 1 + \text{ceil}(\log_2 S)$ bit kullanmaktadır. $b_{N_b-1} \dots b_{S-1}$ bitleri (0'dan $(\log_2 S - 1)$ 'e kadar) aşama numarasını belirlemek üzere $b_{S-2} b_{S-3} \dots b_0$ bitleri ise hafıza adreslerini üretmek için kullanılmaktadır.

Adımlar: Tüm bu adımlar çoklayıcıların (multiplexers) yönetiminin kolay olması sebebiyle adres biti başına bir çoklayıcı kullanılarak yönetilmektedir. $N = 128$ - noktalı FFT örnekleri için, R_0 ve R_1 bitleri her aşama için Çizelge 4.1'de verilmiştir. Burada x , R_0 için 0 ve R_1 için 1'dir. Çoğu a_i biti, s aşamasına bağlı olarak en fazla iki farklı sayaç bit değerinden birini, 0 veya 1 sabitini varsaymaktadır. Bu, her adres biti için ($N = 128$ için üç seçim biti) $\text{ceil}(\log_2 S)$ seçim bitlerine sahip bir çoklayıcı gerektirmektedir. Bununla birlikte, a_i için çoklayıcı girişi $i > S_h$ olduğunda b_{i-1} ve $i \leq S_h$ olduğunda b_i 'dir. Çizelgeyi bir HDL tasarımına gömmek, çoklayıcı bloğunu büyük ölçüde basitleştirmektedir. Çizelgedeki b_0 , b_1 ve x konumları hariç (gri arkaplan), kalan yerler son aşama haricinde sayacın kalan bitleri ile serbestçe doldurulabilmektedir.

Çizelge 4.1. $N = 128$ için sayaç bitlerinden okuma adreslerinin oluşturulması.

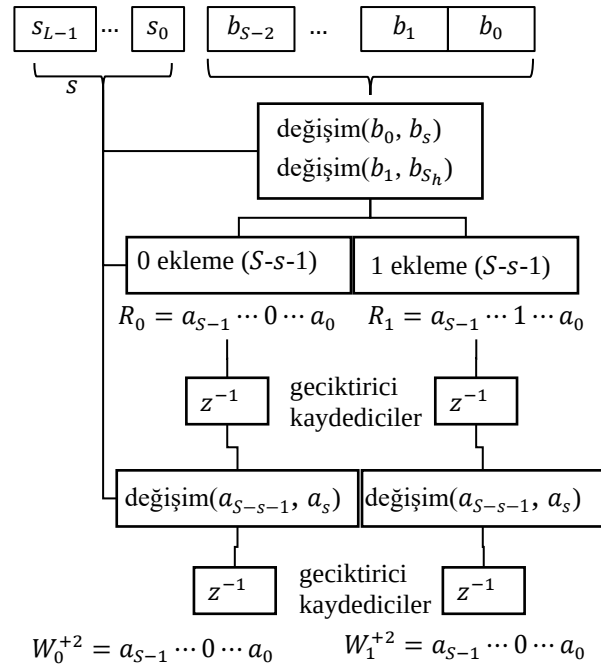
s	a_6	a_5	a_4	r	a_2	a_1	a_0
0	x	b_5	b_4	b_1	b_3	b_2	b_0
1	b_5	x	b_4	b_1	b_3	b_0	b_2
2	b_5	b_4	x	b_1	b_0	b_3	b_2
3	b_5	b_4	b_3	x	b_0	b_2	b_1
4	b_5	b_4	x	b_1	b_0	b_3	b_2
5	b_5	x	b_4	b_1	b_3	b_0	b_2
6	x	b_5	b_4	b_3	b_2	b_1	b_0

Benzer şekilde, yazma adresi yapılandırması Çizelge 4.2'de $N = 128$ için verilmektedir. Okuma adreslerine benzer olarak, x , W_0 için 0 ve W_1 için 1'dir. Okuma ve

yazma formülasyonu arasındaki benzerlik, sayısal tasarım uygulamasını oldukça basitleştirmektedir. Öte yandan, çoklayıcı yöntemi, kombinatoriyal gecikmeyi sınırlı tutmaktadır. Tüm değişim işlemlerinin ve eklemelerin çoklayıcı blokları tarafından kontrol edildiği genel algoritmik adres üretme devresi Şekil 4.29’da verilmektedir. Oluşturulan adresler daha sonra kaydediciler kullanılarak iki saat periyodu ile geciktirilmektedir.

Çizelge 4.2. $N = 128$ için sayaç bitlerinden yazma adreslerinin oluşturulması.

s	a_6	a_5	a_4	$w.$	a_2	a_1	a_0
0	b_0	b_5	b_4	b_1	b_3	b_2	x
1	b_5	b_0	b_4	b_1	b_3	x	b_2
2	b_5	b_4	b_0	b_1	x	b_3	b_2
3	b_5	b_4	b_3	x	b_0	b_2	b_1
4	b_5	b_4	x	b_1	b_0	b_3	b_2
5	b_5	x	b_4	b_1	b_3	b_0	b_2
6	x	b_5	b_4	b_3	b_2	b_1	b_0



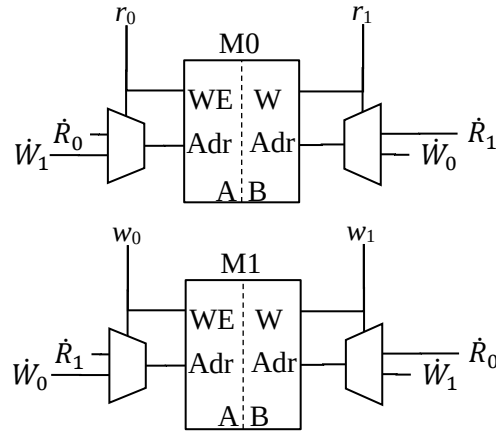
Şekil 4.29. Adres üretici çoklayıcı bloğu.

Oluşturulan yazma adresleri, ardışık düzen kaydedicileri kullanılarak iki saat periyodu ile geciktirilmelidir. Kaynak tasarrufu sağlamak amacıyla, yazma adreslerini çoklayıcı kullanarak sayaç bitlerinden oluşturmak yerine okuma adreslerinden (R_0 ve R_1) üretmek mantıklıdır. Şekil 4.29'da şematik gösterimi olan bu algoritma HDL tasarımcılarına oldukça esneklik ve kolay kontrol seçeneği sunmaktadır.

Hafıza Bloğu Seçimi: Tüm R_0, R_1, W_0^{+2} ve W_1^{+2} içindeki a_{S_h} hafıza bloğunu seçmek için kullanılmaktadır. Bu bitler r_0, r_1, w_0 ve w_1 olarak adlandırıldığında, kalan bitler hafıza bloklarının fiziksel adreslerini temsil etmektedir. Formüle edilmiş ifadeyle, bahsi geçen hafıza bloğu seçim bitlerini çıkarılmış haliyle denklem (4.8) bize fiziksel adresleri ifade etmektedir. Adresler formülde yaklaşık işareti ($\sim$) kullanılarak temsil edilmektedir. w_0 ve w_1 , gecikmeli (iki saat periyodu ile) yazma adreslerinden çıkarıldığında fiziksel adres bitlerinin sayısı artık $S - 1$ olmaktadır.

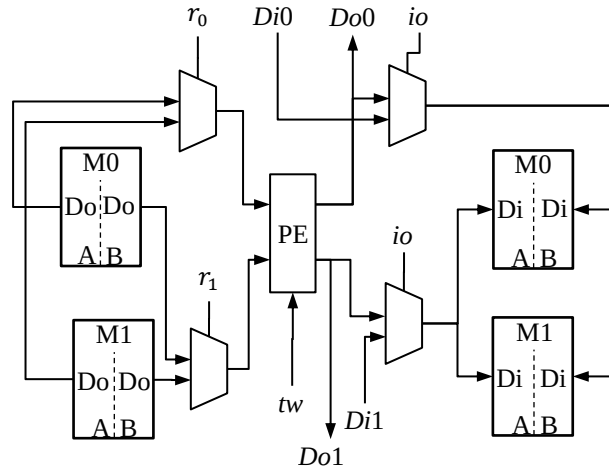
$$\begin{aligned} \dot{R}_0 &= \tilde{R}_0, \dot{R}_1 = \tilde{R}_1 \\ \dot{W}_0 &= \tilde{W}_0^{+2}, \dot{W}_1 = \tilde{W}_1^{+2} \end{aligned} \quad (4.8)$$

r_0, r_1, w_0 ve w_1 bitleri için olası değerler analiz edildiğinde, $s < S - 1$ için yalnızca üç olası kombinasyon olduğu görülmektedir; 0011, 1100 veya 0101, burada her bit hafıza blok numarasını göstermektedir (son aşama hariçtir). Örneğin, 0011, verilerin çift portlu 0 hafıza bloğunun (M0) $\dot{R}_0$ ve $\dot{R}_1$ adres portlarından okunduğunu, 1 hafıza bloğu (M1) $\dot{W}_0$ ve $\dot{W}_1$ adres portlarına yazılacağını göstermektedir. w_1 ve w_0 her zaman sırasıyla r_0 ve r_1 'in tersi olduğundan, yalnızca son ikisini üretmek ve bunları blok seçimi için adres yönlendirme için kullanmak yeterlidir. Çift portlu hafıza bloklarının adres girişlerini besleyen çoklayıcıların girişlerini düzenleyerek, tüm $N = 2^{2n+1}$ -noktalı FFT'ler için geçerli olan Şekil 4.30'da gösterilen adres yönlendirme devresi kolayca elde edilmektedir. WE sinyali Adres üretme devresi tarafından üretilir ve bellek yazma işlemlerine izin verildiğinde yükselir (high), aşama başlarında (2 saat darbesi) düşük (low) pozisyonundadır.



Şekil 4.30. Adres yönlendirme çoklayıcı bloğu.

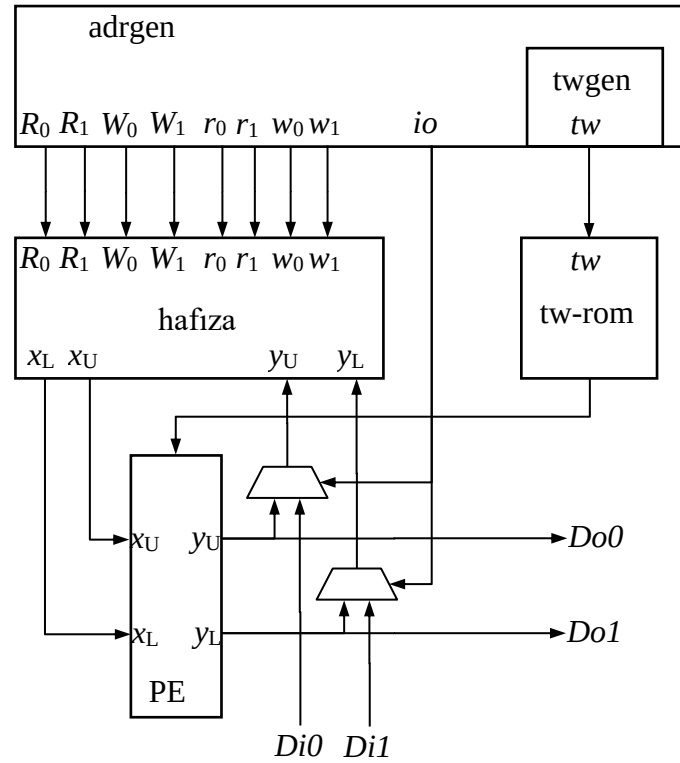
Yalnızca üç olası r_0 , r_1 w_0 ve w_1 bit kombinasyonuna sahip olmanın rahatlığı, Çizelge 4.1 ve Çizelge 4.2’de görüldüğü gibi yalnızca hafıza seçiminde kullanılan sayacın b_1 biti ve x çiftlerinden meydana gelmektedir. Bu kolaylık benzer şekilde veri yönlendirmede de görülmektedir (Şekil 4.31).



Şekil 4.31. Blok seçim biti kullanılarak veri çoklayıcılar aracılığıyla PE ve hafıza blokları arasındaki bağlantı gösterimi.

Şekil. 4.31'deki devre son aşama işlemleri yapılırken aynı zamanda yeni verilerin alınmasına izin veren giriş çoklayıcılarını göstermektedir. $Do0$ ve $Do1$, son aşamada PE'nin çıkışlarından okunan sürekli çıkış verilerinin birinci ve ikinci yarısını temsil etmektedir. Çıkışlar sıralı olduğundan bu veriyi kullanacak devreye doğrudan verilebilmektedir. Benzer şekilde, $Di0$ ve $Di1$ verileri, yalnızca son aşamada aktif olan io sinyali tarafından kontrol edilen çoklayıcılar aracılığıyla belleğe kaydedilmektedir.

Tasarımın ihtiyaca göre değiştirilebilmesi için adres oluşturma, bellek bloğu, sabit çarpanlar oluşturma ve PE birimleri VHDL uygulamasında ayrı modüllerle hazırlanmıştır. Her biri $N/2$ boyutunda 2 bellek bankasından oluşan bellek bloğu, işlem sırasında veri sıralamasına sahip olmak için hem giriş hem de çıkış çoklayıcıları gerektirmektedir. Bahsedilen çalışmanın genel blok diyagramı Şekil 4.32'de sunulmuştur.



Şekil 4.32. Önerilen çalışma için genel blok diyagramı.

Literatürdeki kendinden sıralanan algoritmaların çok azı sabit çarpanlar üretiminden ve hatta hiçbirini donanım uygulamalarından bahsetmektedir. Bu çalışmada, $S - 1$ boyutundaki sabit çarpan indeksleri S boyutundaki R_0 'ın bitlerinin yer değiştirmesiyle üretilmektedir (Denklem 4.9).

$$tw = lsb_{S-1}\{\dots a_{s+1}a_s \mathbf{a}_{S-s-1} \dots \mathbf{a}_{S-1}0_0 \dots 0_{s-1}\} \quad (4.9)$$

lsb_{S-1} operatörü R_0 'ın en az ağırlıklı $S - 1$ bitini almakta ve s kadar sona '0' eklemektedir. Koyu renkte yazılan bitler ters sırada (reverse order) R_0 'ın s kadar en fazla ağırlıklı bitleridir. Örneğin, N=128-noktalı FFT ($S = 7$) için $R_0 = a_6a_5a_4a_3a_2a_1a_0$ 'dır. $s = 2$ aşamasında $a_6a_5a_4a_3a_2$ 'nin en ağırlıklı 2 biti ters düzende $\mathbf{a_5a_6}$ 'nın yanına '00' eklenmesiyle twiddle adresi $a_6a_5a_4a_3a_2\mathbf{a_5a_6}00$ olarak bulunmaktadır. Son olarak $tw = a_3a_2\mathbf{a_5a_6}00$ elde etmek için en sağdaki 6-bit çıkarılır. Açıkça ki, önceden $a_6a_5a_4$ ($s+1$ en ağırlıklı bitler)'in çıkarılacağı bilindiğinden, bu bitleri ilk adımda hariç tutmak pratiktir.

tw , W^{+2} adresinin hafıza adres girişinde belirmesiyle aynı anda PE girişinde belirmelidir. tw 'nin R_0 adresinden üretilmesi sebebiyle sabit çarpan adreslerinin üretiminde ek bir gecikme ya da adres üretimi söz konusu değildir. $s = 0$ 'da sabit çarpanlar R_0 'a eşit olduğundan bu hesaplamalardan ayrı tutulmalıdır. Sabit çarpanlar üretimi Çizelge 4.3'te verilmiştir.

Çizelge 4.3, N=128 için tw 'nin nasıl inşa edildiğini göstermektedir. Burada, $N/2$ boyutlu sabit çarpanlar ROM'unun, bu tez çalışmasında sabit çarpanlar ROM'un optimizasyonuna veya yinelemeli çarpımlar yoluyla sabit çarpan faktörlerinin oluşturulmasına değil, doğal sıralama ve paralellik üzerine odaklanması nedeniyle kullanıldığı varsayılmaktadır.

Çizelge 4.3. $N = 128$ için sabit çarpanlar (tw) üretimi.

s	$tw (R_0 = a_6 a_5 a_4 a_3 a_2 a_1 a_0)$
0	$a_5 a_4 a_3 a_2 a_1 a_0$
1	$a_4 a_3 a_2 a_1 \mathbf{a_6} 0$
2	$a_3 a_2 \mathbf{a_5} \mathbf{a_6} 00$
3	$\mathbf{a_4} \mathbf{a_5} \mathbf{a_6} 000$
4	$\mathbf{a_5} \mathbf{a_6} 0000$
5	$\mathbf{a_6} 00000$
6	000000

5. BULGULAR VE TARTIŞMA

5.1. Tek Blok Çift-Port Hafıza Çalışma Bulguları

Tez çalışmasında tek blok çift-port hafıza ile radiks-2 yöntemiyle tek bir PE kullanarak 3 ayrı adresleme algoritması geliştirilmiş ve FPGA (çift-port hafıza olan tüm FPGA'lerde uygulanabilir) üzerinde VHDL ile uygulaması yapılmıştır. Tek blok çalışmalar literatürde oldukça azdır. Çalışmalarda olabilecek en düşük hafıza olan N büyüklüğünde hafıza kullanılmıştır. Hafıza ve tek bir PE ile devre elemanı sayısını minimumda tutmak amaçlanmıştır. Yapılan uygulamalarda fazladan saat darbesi kullanımı söz konusu olmuştur. Geliştirilen adresleme algoritmalarıyla fazladan saat darbesi ihtiyacını en aza indirmek ve toplam süreyi optimuma yakın hale getirilmeye çalışılmıştır. Hafıza adresleme algoritmalarının sonuçları ve benzer çalışmalarla karşılaştırmaları alt bölümlerde verilmiştir.

5.1.1. 16-Noktalı Radiks-2 Hızlı Fourier Dönüşümü Hafıza Adresleme Algoritması Bulguları

Bölüm 4.4.1'de anlatılan hafıza adresleme algoritması 16-noktalı FFT için sınırlı olmaktadır. Algoritmada hafıza çıkışlarındaki çıkış saklayıcıları kullanılarak verilerin okuma/yazma işlemleri bu saklayıcılar yardımıyla tasarımda tek hafıza kullanımını sağlamıştır. Ancak aşama sonlarında yazma işlemi yapılırken ek bir saat darbesine ihtiyaç duyulmuş ve bu ek saat darbesi bir sonraki aşama için adres ayarlama yapılması için de gereklidir. Bu yaklaşımla FFT hesaplamaları, tek aşama kelebek işlemleri için $\frac{N}{2}$ olmak üzere tüm aşamalar için $\frac{N}{2} \log_2 N$ saat döngüsünde tamamlanmaktadır. Fazladan saat darbesi tüm FFT aşamaları için $\log_2 N$ 'dir. Böylece tüm FFT işlemleri $(\frac{N}{2} + 1) \log_2 N$ operasyon saati süresinde tamamlanmaktadır.

5.1.2. N-Noktalı Radiks-2 Hızlı Fourier Dönüşümü Hafıza Adresleme Algoritması Bulguları

Bu tez çalışmasında (Bölüm 4.4.2.) tek bir hafıza bloğu kullanan basit bir radiks-2 FFT yöntemi uygulaması yapılmıştır. Önerilen yaklaşım VHDL kullanılarak Xilinx FPGA'lerde bulunan çift portlu RAM kullanılarak test edilmiştir. Minimum bellek boyutu ve en basit mimariyi hedeflediğimizden, radiks-2 ve tek bellek bloğuyla tasarımlar gerçekleştirebilir bulunmaktayız. Sonuç olarak, saat darbe sayısı yüksek radiks ve çoklu hafıza blokları kullanan diğer karşılaştırılabilir algoritmalara oranla daha yüksek orandadır. Daha yüksek radiks ve çoklu hafıza blokları, daha az gecikme elde etmek için yüksek devre karmaşıklığı ve karmaşık algoritmalar anlamına gelmektedir. Bu nedenle, yaklaşımımızın en büyük avantajı, ihtiyaç duyulan hafıza boyutunun azlığı ve tasarımın sadeliği olmaktadır. Literatürde FFT boyutuna eşit bellek büyüklüğünü kullanan tek bloklu çift portlu hafıza tabanlı FFT mimarisi bulunmamaktadır. Yine de Çizelge 5.1 ve Çizelge 5.2'de vermiş olduğumuz veriler yaklaşımımızı saat darbesi, hafıza büyüklüğü ve hafıza blokları açısından yeni algoritmalar kullanan benzer çalışmalarla karşılaştırmaktadır.

Bu yaklaşımla tüm FFT aşamaları $\frac{N}{2} \log_2 2N = \frac{N}{2} (1 + \log_2 N)$ saat döngüsünde tamamlanmaktadır. Bu değer literatürde tek bir PE ve tek-blok hafıza kullanan algoritmalar arasında oldukça iyidir. Aslında, tek-blok hafıza algoritma çalışmaları oldukça nadirdir. PE kullanımı büyük FFT'ler için %100'e yakındır. Bu değerden sapmalar kümeler arasında harcanan fazladan saat darbesi kaynaklıdır.

Önerilen yaklaşımın, donanım kaynaklarının az olduğu ve gecikmenin temel sorun olmadığı, ancak yazılım tabanlı yaklaşımların gecikme gerekliliklerini yerine getiremediği uygulamalarda kullanılabilir.

Çizelge 5.1. Çeşitli çalışmaların saat darbe sayısı karşılaştırması.

Önerilen			[1]		[2]*[3]*		[4]	
Boyut	Döngü	<i>r</i>	Döngü	<i>r</i>	Döngü	<i>r</i>	Döngü	<i>r</i>
512	2560	2	-	-	-	-	640	mix
1024	5632	2	1024	2/8	2560	4	2560	4
2048	12288	2	2048	4/8	22528	2	6144	2/8
4096	26624	2	4096	8	4096	8	12288	4
8192	57344	2	10240	2/8	106496	2	28672	2/4

[1]: Hsiao vd., 2010

[2]: Johnson, 1992

[3]: Hidalgo vd.,1999

[4]: Jo ve Sunwoo, 2005

-: verilmemiş

*: [1]'den alıntı

Çizelge 5.2. Hafıza boyutu ve bankası karşılaştırması.

Önerilen		[1]	[2] [3] [4]	[5]
Hafıza	N	2N	2N	2N
Bankalar	1	4/8	2+	5/7

[1]: Hsiao vd., 2010

[2]: Johnson, 1992

[3]: Hidalgo vd.,1999

[4]: Jo ve Sunwoo, 2005

[5]: Chen vd., 2014

5.1.3. Radiks-2 Hızlı Fourier Dönüşümü ile N-Noktalı Yeni Hafıza Adresleme Algoritması Bulguları

Bu çalışmada (Bölüm 4.4.3.) tek bir hafıza bloğu kullanan yeni bir radiks-2 FFT yöntemi sunulmaktadır. Önerilen yaklaşım VHDL kullanılarak Xilinx marka FPGA’larda bulunan çift portlu RAM kullanılarak test edilmiştir. Minimum bellek boyutu ve en basit mimari hedeflenmiş, bu hedef doğrultusunda radiks-2 FFT ve tek bellek bloğu kullanılarak yeni adresleme algoritması üretilmiştir. Tüm FFT işlemleri için harcanan saat darbe sayısı düşük noktalı FFT işlemleri için fazla avantaj sağlamamakta, ancak yüksek noktalı FFT işlemlerinde önem arz etmektedir. Geliştirilen algoritma ile en düşük bellek boyutu olan N adet hafıza kullanılmakla birlikte tüm FFT aşamaları için harcanan saat darbesi sayısının her bir FFT işlemi için bir saat darbesinden daha az sürede hesaplanamayacağı göz önünde bulundurularak, optimum değere (tüm FFT işlemleri için $\frac{N}{2} \log_2 N$) yakın olduğu görülmüştür. Bir önceki yapılan çalışma ile karşılaştırıldığında, kullanılan hafıza miktarı aynı kalmış, ancak fazladan kullanılan saat darbesi sayısı azalmıştır. Bu tez çalışmasının bir önceki çalışma ve optimum değerler ile kullanılan hafıza miktarı ve saat darbesi sayısı karşılaştırılması Çizelge 5.3’te verilmiştir. Kullanılan hafıza boyutu, miktarı ve saat darbesi sonuçlarının, literatürdeki diğer benzer çalışmalarla karşılaştırılmış hali ise Çizelge 5.4’te yer almaktadır. Daha yüksek radiks ve çoklu hafıza blokları, daha az gecikme elde etmek için yüksek devre karmaşıklığı ve karmaşık algoritmalar anlamına gelmektedir. Bu nedenle, yaklaşımımızın en büyük avantajı, ihtiyaç duyulan hafıza boyutunun azlığı ve tasarımın sadeliği olmaktadır. Literatürde FFT boyutuna eşit bellek büyüklüğünü kullanan tek bloklu çift portlu hafıza tabanlı FFT mimarisi bulunmamaktadır.

Bu yaklaşımla FFT hesaplamaları, tek aşama kelebek işlemleri için $\frac{N}{2}$ olmak üzere tüm aşamalar için $\frac{N}{2} \log_2 N$ saat döngüsünde tamamlanmaktadır. Ek olarak, aşama sonlarında fazladan okumanın yapıldığı ve aynı zamanda bir sonraki aşama için adres ayarlama yapılması gerektiği durumlarda da fazladan bir saat darbesi kullanımı söz konusudur. Fazladan saat darbesi tüm FFT aşamaları için $\log_2 N$ ’dir. Böylece tüm FFT aşamaları toplamda $(\frac{N}{2} + 1) \log_2 N$ saat döngüsünde tamamlanmaktadır. Bu değer, bir önceki

çalışmada yapılan $\frac{N}{2} (1 + \log_2 N)$ saat döngüsünde tamamlanan ve literatürde tek bir PE ve tek-blok hafıza kullanan algoritmalar arasındaki en düşük saat döngüsü değeridir.

Çizelge 5.3. Optimum değerler ve yapılan çalışmaların saat darbe sayısı ve hafıza miktarı karşılaştırması.

Optimum	Boyut	512	1024	2048	4096	8192
	Döngü	2304	5120	11264	24576	53248
[1]	Boyut	512	1024	2048	4096	8192
	Döngü	2313	5130	11275	24588	53261
[2]	Boyut	512	1024	2048	4096	8192
	Döngü	2560	5632	12288	26624	57344

[1]: Kaya ve Seke, 2020

[2]: Seke ve Kaya, 2018

Çizelge 5.4. Çeşitli çalışmaların saat darbe sayısı ve hafıza karşılaştırması.

	Hafıza	Döngü	r	Hafıza bankaları	Hafıza
Hsiao vd., 2010	512	-	-	4/8	2N
	1024	1024	2/8		
	2048	2048	4/8		
	4096	4096	8		
	8192	10240	2/8		
Chen vd., 2015	512	640	mix	5/7	2N
	1024	2560	4		
	2048	6144	2/8		
	4096	12228	4		
	8192	28672	2/4		
Ayinala vd., 2013 ¹	512	1152	2	4	N
	1024	2560			
	2048	5632			
	4096	12288			
	8192	-			
Radhouane vd., 2000	512	2304	2	2	2N
	1024	5120			
	2048	11264			
	4096	24576			

	8192	53248			
Lo vd., 2001	512	2306	2	2	2N
	1024	5122			
	2048	11266			
	4096	24578			
	8192	53250			
Baek ve Choi, 2008	512	2304	2	2	2N
	1024	5120			
	2048	11264			
	4096	24576			
	8192	53248			
[1]	512	2313	2	1	N
	1024	5130			
	2048	11275			
	4096	24588			
	8192	53261			

-: verilmemiş

¹: yalnızca gerçek değerli veriler kullanılmış

[1]: Kaya ve Seke, 2020

5.2. Çoklu Blok Çift-Port Hafıza-Tabanlı Radiks-2 Hızlı Fourier Dönüşümü Uygulamaları Bulguları

Tez çalışmasının önceki bölümünde (5.1.) radiks-2 hafıza-tabanlı FFT ile N adet tek blok çift-port hafıza kullanılarak üretilen adresleme algoritmaları uygulama sonuçları verilmiştir. Bu çalışmalarda uygulanan adresleme algoritmalarıyla tüm FFT işlemleri $\frac{N}{2}(1 + \log_2 N)$ ve $(\frac{N}{2} + 1)\log_2 N$ saat döngüsünde tamamlanmaktaydı. Yapılan çalışmalar sonrasında elde edilen operasyon süresi değerleri, oldukça az görülen tek blok ve N kadar hafıza kullanan çalışmalar arasında en iyileridir. Ancak, yüksek noktalı FFT işlemleri için operasyon saatinin azaltılmasının önemli olduğu düşünülerek bu parametreyi olabilecek en düşük operasyon saati olan $N/2 \log_2 N$ 'e indirmeye hatta paralel işlemler ile daha kısa sürede yapmak hedeflenmiş, yeni adresleme algoritmaları ile toplam operasyon saati süresinde iyileştirme çalışmaları yapılmıştır. Bu algoritmalar ile gerek hafıza kullanımı gerekse toplam saat darbesi bakımından optimum değerlere ulaşılmış, sonuçlar ve benzer çalışmalar ile karşılaştırmaları sunulmuştur.

5.2.1. Radiks-2 Çoklu Gerçek Çift Portlu Hafıza-Tabanlı Hızlı Fourier Dönüşümü Adresleme Algoritması Bulguları

Bölüm 4.5.1’de algoritma tasarımı anlatılan çalışmanın sonuçları ve diğer çalışmalarla karşılaştırılması bu bölümde verilmiştir. Önerilen yaklaşım VHDL kullanılarak Xilinx FPGA’lerde bulunan gerçek çift portlu RAM kullanılarak test edilmiştir. Minimum bellek boyutu ve en basit mimari hedeflenmiş, bu hedef doğrultusunda radiks-2 FFT ve bölünmüş bellek bloğu kullanılarak yeni adresleme algoritması üretilmiştir. Geliştirilen algoritma ile en düşük bellek boyutu olan N adet hafıza kullanılmakla birlikte tüm FFT aşamaları için harcanan saat darbesi sayısının her bir FFT işlemi için bir saat darbesinden daha az sürede hesaplanamayacağı göz önünde bulundurularak, optimum değere (tüm FFT işlemleri için $\frac{N}{2} \log_2 N$) ulaşılmıştır. Bir önceki yapılan çalışmalar ile karşılaştırıldığında, kullanılan toplam hafıza miktarı açısından aynı (fakat bölünmüş hafıza kullanılmakta), ancak fazladan kullanılan saat darbesi sayısı ortadan kaldırılmıştır. Bu çalışmanın bir önceki çalışma ve optimum değerler ile kullanılan hafıza miktarı ve saat darbesi sayısı karşılaştırılması Çizelge 5.5’te verilmiştir. Kullanılan hafıza boyutu, miktarı ve saat darbesi sonuçlarının, literatürdeki diğer benzer çalışmalarla karşılaştırılmış hali ise Çizelge 5.6’da yer almaktadır. Daha yüksek radiks ve çoklu hafıza blokları, daha az gecikme elde etmek için yüksek devre karmaşıklığı ve karmaşık algoritmalar anlamına gelmektedir. Bu nedenle, yaklaşımımızın en büyük avantajı, ihtiyaç duyulan hafıza boyutunun azlığı ve tasarımın sadeliği olmaktadır.

Çizelge 5.5. Optimum değerler ve yapılan çalışmanın saat darbe sayısı karşılaştırması.

	Boyut	512	1024	2048	4096	8192
Optimum	Döngü	2304	5120	11264	24576	53248
Önerilen	Döngü	2304	5120	11264	24576	53248
[1]	Döngü	2313	5130	11275	24588	53261
[2]	Döngü	2560	5632	12288	26624	57344

[1]: Kaya ve Seke, 2020

[2]: Seke ve Kaya, 2018

Çizelge 5.6. Çeşitli çalışmaların toplam çalışma süresi, paralel işlem, hafıza ve hafıza bankası karşılaştırması.

Yaklaşım	Radiks	Paralel İşlem	Hafıza	Banka	Döngü	Gözlem
[1]	4	4	N	4	$\frac{N}{8}(\log_2 N)$	Kelebek işlemleri DSP48E’de gerçekleştirilmiş
[2]	4	4	N	4	$\frac{N}{8}(\log_2 N)$	Yüksek miktarda mux, barrel öteleyici ekstra kaydediciler kullanılmış
[3]	4/2	4	2N	4	$\frac{N}{8}(\log_2 N)$	Sürekli akış
[4]	2	4	N	4	$\frac{N}{4}(\log_2 N - 1) + 1$	Yalnızca gerçek-değerli veriler
[5]	2	2	N	4	$\frac{N}{2}(\log_2 N)$	Okuma/Yazma adresleri için ilave Barrel kaydırıcı
Önerilen	2	2	N	2	$\frac{N}{2}(\log_2 N)$	

[1]: Garrido vd., 2016

[2]: Xiao vd., 2009

[3]: Jo ve Sunwoo, 2005

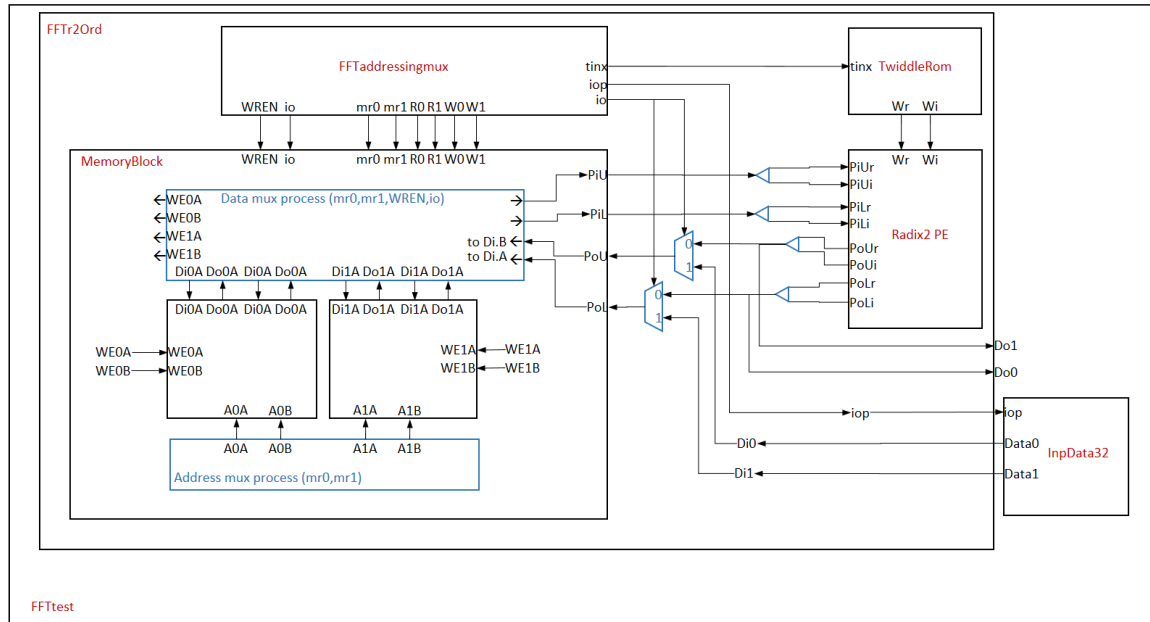
[4]: Ma vd., 2015

[5]: Ma ve Wanhammar, 2000

Yapılan çalışmanın, donanım kaynaklarının az olduğu ve gecikmenin temel sorun olduğu, gerçek zamanlı uygulama gerekliliklerini yerine getiren uygulamalarda kullanım alanı bulabileceği sonucuna varılmıştır. Tasarlanan adresleme algoritmasıyla hem donanım hem de zaman açısından optimum değerlere ulaşılmıştır ancak, çıkış verileri normal düzende değildir. Radiks-2 DIF FFT yapısında da giriş düzeni normal ancak çıkışlar ters-bit (bit-reversed) düzeniyle olmaktadır. Birçok araştırmacı bu durumu görmezden gelmiş ya da son aşama adresleri için ayrıca bir barrel kaydırıcı devresi kullanımını uygun görmüşleridir. Çünkü literatürde birçok çalışmada adresleme algoritması tasarımı üzerine yoğunlaşmış ve verileri normal düzene sokmak için ek hafıza ya da devre elemanı kullanmaktan kaçınmamışlardır. Ancak unutulmamalıdır ki bu tarz işlemler ek süre gerektirmekte aynı zamanda gerçek zamanlı uygulamalar için uygun olmamaktadır. Normal giriş ve çıkış düzeniyle yapılan tasarımlar gerçek zamanlı uygulamalarda önem arz etmektedir.

5.2.2. Paralel Giriş/Çıkış İşlemleri Sıralı Yerinde Kullanılabilir Radiks-2 Hızlı Fourier Dönüşümü Adresleme Metodu Bulguları

Bölüm 4.5.3'te hafıza adresleme algoritması verilen, sıralı giriş/çıkış ve son aşamada paralellığe sahip radiks-2 FFT çalışması VHDL ile kodlanmış ve çift portlu blok hafızaları olan çeşitli FPGA'larda testleri yapılmıştır. Tasarımın önceden hafızaya yazılmış verilerle ve gerçek zamanlı verilerle $N=32, 128, 512, \dots$ noktalı FFT'ler için uygulaması yapılmıştır. $N=2048$ noktalı FFT için VHDL modül şeması Şekil 5.1'de verilmiştir. Tasarımın ihtiyaca göre değiştirilebilmesi için adres oluşturma, bellek bloğu, sabit çarpan faktörü oluşturma ve PE birimleri ayrı modüllerle uygulanmaktadır. Her biri $N/2$ boyutunda 2 bellek bankasından oluşan bellek bloğu, işlem sırasında veri sıralamasına sahip olmak için hem giriş hem de çıkış çoklayıcıları gerektirmektedir. Geliştirilen algoritma ile en düşük bellek boyutu olan N adet hafıza kullanılmakla birlikte tüm FFT aşamaları için harcanan saat darbesi sayısının her bir FFT işlemi için bir saat darbesinden daha az sürede hesaplanamayacağı göz önünde bulundurularak, tüm FFT işlemleri için $\frac{N}{2} \log_2 N/2$ saat darbesi değerine ulaşılmıştır.



Şekil 5.1. $N=2048$ VHDL modül şeması.

Sunulan yöntemin temel avantajı normal sıralı düzen giriş ve çıkış verileri ile paralel işlem yapabilme özelliğidir. Çizelge 5.8’de verilen diğer tüm yöntemler, FFT devresinin bekleme durumunda olduğu sırada veri yükleme/boşaltma döngüsü gerektirmektedir. Bununla birlikte, yöntem, ayrı yükleme/boşaltma yapan eşdeğer yöntemlere kıyasla en az $N/2$ saat döngüsü tasarrufu sağlayan son aşama işlemeye paralel olarak veri yükleme/boşaltma yapmaktadır. Adresleme yöntemi, iki PE’nin paralel olarak çalıştığı şemalara oldukça kolay uygulanabilir. Daha yüksek radiks mimarilerinin FFT’yi tamamlamak için daha düşük saat döngüleri sağladığı açıktır. Bu nedenle, aynı çizelgedeki farklı radiks FFT’leri karşılaştırmak tamamen doğru değildir. Sonuçta, daha yüksek radiksli FFT’ler, çarpanlar gibi daha fazla donanım kaynağı kullanmaktadır. Biraz adil bir karşılaştırma için, çizelgede veri doldurma ve sonuç okuma için gerekli saat döngüleri eklenmiştir. Çizelgedeki karşılaştırılan çalışmaların hiçbiri veri doldurma ve sonuç okuma için bir mekanizma sağlamamaktadır. Bellek blokları paralel olarak doldurulabildiğinden ve okunabildiğinden (doğal sırada olmasa da), saat döngüleri için formülasyon da buna göre değiştirilmiştir.

Temel optimize edilmemiş VHDL kodu, 160 MHz maksimum saat frekansı ile Artix-7 (sg-3) ve 144 MHz maksimum saat frekansı ile Zinq (sg-3) için sentezlenmektedir. Bunlar etkileyici hızlar olmasa da temel tasarımda ek ardışık düzen kaydedicilerinin veya diğer hız iyileştirme yöntemlerinin kullanılmadığına dikkat edilmelidir. Çalışmanın detaylı VHDL sentez raporuna göre $N=512$ nokta için sentezlenmiş FFT bileşenlerinin 395 adet flip-flop ve 143 adet çoklayıcı kullandığı görülmüştür.

Önerilen yöntemin temel avantajı, giriş ve çıkış verilerini sıralı düzen elde etmekten ziyade bu özellik sayesinde paralellik özelliğiyle tüm FFT işlemlerinin optimum sürede daha kısa sürede tamamlanmasıdır. Çizelge 5.8’de literatürdeki en iyi radiks-2 ve radiks-4 hafıza-tabanlı FFT çalışmaları ve önerilen yöntemin karşılaştırması verilmiştir. Karşılaştırması yapılan çalışmalar FFT hesaplamasının beklemede olduğu veri yükleme/boşaltma için ek saat darbeleri gerektirmektedir. Önerilen yöntem, son aşamada veri yükleme/boşaltma işlemlerinin paralel yapılabilmesi özelliğiyle bu işlemler için ayrı saat darbesi gerektiren diğer çalışmalardan $N/2$ saat darbesi daha kısa sürede çalışmaktadır.

Çizelge 5.7. Çeşitli hafıza-tabanlı FFT metotları karşılaştırması.

Yaklaşım	r	#bankalar/M	Döngü	Giriş/Çıkış Paralelliği	N=2048 döngüleri
[1]	2	$4 / N$	$N/2 \log_2 N$	Hayır	12288
[2] ¹	2,4	$4 / 2N$	$N/8 \log_2 N$	-	3840
[3]	4	$4 / N$	$N/8 \log_2 N$	Hayır	3840
[4]	2	$2 / N$	$N/2 \log_2 N$	Hayır	13312
[5] ²	2	$4 / N$	$N/4 (\log_2 N - 1) + 1$	Hayır	6145
[6]	4	$4 / N$	$N/8 \log_2 N$	Hayır	3840
Önerilen ³	2	$4 / N$	$N/2 \log_2(N/2)$	Evet	10240

¹: radiks-2 ve radiks-4 birlikte kullanılmaktadır.

²: Yalnızca gerçek değerli veriler kullanılmaktadır.

³: Giriş/Çıkış veri saat darbesini içermektedir.

[1]: Ma ve Wanhammar, 2000

[2]: Jo ve Sunwoo, 2005

[3]: Xiao vd., 2009

[4]: Xiao vd., 2008

[5]: Ma vd., 2015

[6]: Garrido vd., 2016

6. SONUÇ VE ÖNERİLER

Tez çalışmasında tek blok ve bölünmüş hafıza yapılarıyla hafıza adresleme algoritması tasarlanmış ve uygulamaları yapılmıştır. Bu çalışmalarda radiks-2 DIF algoritması ile hafıza-tabanlı FFT metodu kullanılmış ve geliştirilen hafıza adresleme algoritması ile tek bir PE ve en düşük hafıza miktarı olan N adet hafıza kullanılmıştır. İlk tasarlanan çalışma tek blok hafıza kullanılarak geliştirilen adresleme algoritması ile yalnızca 16-noktalı FFT'ler için uygulanabilirlik, $(\frac{N}{2} + 1)\log_2 N$ operasyon saatinde çalışma özelliklerine sahiptir. OFDM için gerekli tüm FFT'ler için uygulanabilir olması için yeni hafıza adresleme algoritması geliştirilmiş ve tek blok hafıza kullanılarak üretilen algoritma $(\frac{N}{2} + 1)\log_2 N$ saat darbesi çalışma süresine sahiptir. Bölüm 4.4.3'te verilen çalışmada fazladan kullanılan saat darbesi azaltılmış ve $\frac{N}{2}\log_2 2N = \frac{N}{2}(1 + \log_2 N)$ sürede çalışan bir hafıza adresleme tasarımı elde edilmiştir. Bu çalışma $\frac{N}{2}(\log_2 N)$ olan optimum çalışma süresine oldukça yakın bir değer olmakla birlikte oldukça az görülen tek bloklu çalışmalar arasında en az hafıza kullanımı ve en iyi çalışma süresi özelliklerine sahiptir.

Literatürde birçok araştırmacı FFT işlemlerinde hızlı ve verimli işlemcilerin yapılması üzerine çalışmalar yapmaktadır. Bu doğrultuda FFT işlemlerini yapmak için toplam sürenin azaltılması, optimum süreye ulaşılması hatta paralel işlem yapılarak bu sürenin de altına inilmesi mümkündür. Bu doktora tezinde, çoklu blok yaklaşımlarla oldukça başarılı sonuç veren iki adet hafıza adresleme algoritması sunulmuştur. Bölüm 4.5.2'de anlatılan çalışmada 2 adet $N/2$ hafıza kullanılarak optimum çalışma süresi olan $N/2 \log_2 N$ değerine ulaşıldığı görülmektedir. Algoritma OFDM için gerekli tüm FFT'ler için uygulanabilmektedir. Yine aynı şekilde (2 adet $N/2$) ve paralelleştirme işlemleriyle optimum süreden daha kısa sürede tüm FFT işlemlerini gerçekleştiren Bölüm 4.5.3'te algoritma mantığı detaylı olarak sunulan tez çalışmasında, oldukça kısa sürede, en az hafıza ile ve sıralı düzen çıkış özellikleriyle oldukça başarılı sonuçlar elde edilmiştir. Bu çalışmada son aşama verileri alınırken bir sonraki işlemleri için gerekli yeni veriler paralel olarak alınabilmekte ve tüm işlemler $N/2 \log_2 N/2$ sürede tamamlanmış olmaktadır. Sunulan adresleme algoritması OFDM için gerekli, gerçek zamanlı işlemlerde kullanılabilir özelliği ile tüm noktalı FFT'ler için uygulanabilmektedir.

Bu doktora tezinde OFDM'in en önemli ve yoğun işlem kısmı olan FFT işlemleri hafıza-tabanlı metodu kullanılarak olabilecek en kısa sürede ve en az hafıza kullanan algoritmalar tasarlanarak uygulaması FPGA üzerinde VHDL kullanılarak uygulaması yapılmıştır. Elde edilen bulgular doğrultusunda tek blok hafıza ile yapılan çalışmalarda fazladan saat darbesi ihtiyacı olduğu görülmüştür. Bu durum optimum çalışma süresinde tamamlanması beklenen ve gerçek zamanlı çalışmalar için engel teşkil etmektedir. Çoklu blok çalışmaların tasarlanan algoritmalar ile çalışma süresi açısından daha iyi sonuçlar verdiği elde edilen bulgular ile kanıtlanmıştır. Algoritmanın normal düzende giriş ve çıkış verileri özelliği ile verilerin çıkıştan alınırken aynı zamanda yeni verilerin alınması paralel işlem olarak adlandırılmış ve bu özellik ile literatüre değer katmıştır. Böylece tüm FFT işlemleri optimum süreden daha kısa zamanda gerçekleştirilmekte ve sürekli akış gerektiren çalışmalar için oldukça avantajlı hale gelmektedir. Paralel çalışma ve kullanılan hafızanın minimum olması bundan sonraki yapılacak çalışmalar için örnek olacaktır. Çoklu blok hafıza, paralellik özelliği ve normal giriş/çıkış veri düzeni ile yüksek radiksli (radiks-4/8/16/25 vb.) uygulamalar yapılması mümkündür.

KAYNAKLAR DİZİNİ

- Anonim, 2007, Altera. DFT/IDFT Reference Design; [Online] Available 2018: <http://www.altera.com>.
- Anonim, 2017, Xilinx. LogiCORE IP Discrete Fourier Transform V9.0; [Online]. Available 2018: <http://www.xilinx.com>.
- Ayinala, M., Lao, Y., Parhi, K. K., 2013, An in-place FFT architecture for real-valued signals. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(10), 652-656.
- Baek, J., Choi, K., 2008, New address generation scheme for memory-based FFT processor using multiple radix-2 butterflies. In 2008 International SoC Design Conference (Vol. 1, pp. I-273). IEEE.
- Bansal, P., Dhaliwal, B. S., Gill, S. S., 2014, Memory-efficient Radix-2 FFT processor using CORDIC algorithm. In 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE) (pp. 1-5). IEEE.
- Burrus, C., Eschenbacher, P., 1981, An in-place, in-order prime factor FFT algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(4), 806-817.
- Chang, C. H., Wang, C. L., Chang, Y. T., 1999, A novel memory-based FFT processor for DMT/OFDM applications. In 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258) (Vol. 4, pp. 1921-1924). IEEE.
- Chang, Y. N., 2008, An efficient VLSI architecture for normal I/O order pipeline FFT design. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(12), 1234-1238.
- Chen, J., Hu, J., Lee, S., Sobelman, G. E., 2014, Hardware efficient mixed radix-25/16/9 FFT for LTE systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(2), 221-229.
- Cheng, C., Keshab K. P., 2007, High-throughput VLSI architecture for FFT computation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 54 (10) 863-867.
- Chu, E., George, A., 1998, FFT algorithms and their adaptation to parallel processing. *Linear Algebra Appl.*, 284, 95-124.
- Cooley, J. W., Tukey, J. W., 1965, An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90), 297-301.

KAYNAKLAR DİZİNİ (devam)

- Cox, C., 2008, Essentials of UMTS. Cambridge University Press. Ms. Anju Uttam Gawas, An Overview on Evolution of Mobile Wireless Communication Networks: 1G-6G, JRITCC, VOLUME 3 ISSUE 5, MAY2015.
- De Caro, D., Strollo, A. G., 2005, High-performance direct digital frequency synthesizers using piecewise-polynomial approximation. IEEE Transactions on Circuits and Systems I: Regular Papers, 52(2), 324-337.
- Doufexi, A., Armour, S., 2005, Design Considerations and Physical Layer Performance Results for a 4G OFDMA System Employing Dynamic Subcarrier Allocation. In PIMRC (Vol. 5, pp. 357-361).
- Fazel, K., Kaiser, S., 2008, Multi-carrier and spread spectrum systems: from OFDM and MC-CDMA to LTE and WiMAX. John Wiley & Sons.
- García, J., Michell, J. A., Ruiz, G., Burón, A. M., 2007, FPGA realization of a Split Radix FFT processor, In VLSI Circuits and Systems III (Vol. 6590, p. 65900P). International Society for Optics and Photonics.
- Garrido, M., Acevedo, M., Ehliar, A., Gustafsson, O., 2014, Challenging the limits of FFT performance on FPGAs. In 2014 International Symposium on Integrated Circuits (ISIC) (pp. 172-175). IEEE.
- Garrido, M., Sánchez, M. Á., López-Vallejo, M. L., Grajal, J., 2016, A 4096-point radix-4 memory-based FFT using DSP slices. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25(1), 375-379.
- Gauss, C. F., 1866, Nachlass: Theoria interpolationis methodo nova tractata. Carl Friedrich Gauss Werke, 3, 265-327.
- Gawas, A. U., 2015, An overview on evolution of mobile wireless communication networks: 1G-6G. International Journal on Recent and Innovation Trends in Computing and Communication, 3(5), 3130-3133.
- Gentleman, W. M., Sande, G., 1966, Fast Fourier Transforms: for fun and profit. In Proceedings of the November 7-10, 1966, fall joint computer conference (pp. 563-578).
- Good, I. J., 1958, The interaction algorithm and practical Fourier analysis. Journal of the Royal Statistical Society: Series B (Methodological), 20(2), 361-372.
- He, S., Torkelson, M., 1996, A new approach to pipeline FFT processor. In Proceedings of International Conference on Parallel Processing (pp. 766-770). IEEE.

KAYNAKLAR DİZİNİ (devam)

- Heideman, M. T., Johnson, D. H., Burrus, C. S., 1985, Gauss and the history of the fast Fourier transform. *Archive for history of exact sciences*, 265-277.
- Hidalgo, J. A., López, J., Arguello, F., Zapata, E. L., 1999, Area-efficient architecture for fast Fourier transform. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(2), 187-193.
- Hsiao, C. F., Chen, Y., Lee, C. Y., 2010, A generalized mixed-radix algorithm for memory-based FFT processors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(1), 26-30.
- Huang, S. J., Chen, S. G., 2010, A green FFT processor with 2.5-GS/s for IEEE 802.15. 3c (WPANs). In *The 2010 International Conference on Green Circuits and Systems* (pp. 9-13). IEEE.
- Huang, S. J., Chen, S. G., 2012, A high-throughput radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15. 3c systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(8), 1752-1765.
- Jacobsen, E., Lyons, R., 2003, The sliding DFT. *IEEE Signal Processing Magazine*, 20(2), 74-80.
- Jo, B. G., Sunwoo, M. H., 2005, New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(5), 911-919.
- Johnson, H. W., Burrus, C. S., 1984, An In-Order, In-Place Radix-2 FFT. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 473-476.
- Johnson, L. G., 1992, Conflict free memory addressing for dedicated FFT hardware. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(5), 312-316.
- Joshi, Shubhangi M., 2015, FFT architectures: a review. *International Journal of Computer Applications* 116.7.
- Kaya, Z., Seke, E., 2020, A novel addressing algorithm of radix-2 FFT using single-bank dual-port memory, *Circuit World*, DOI: 10.1108/CW-06-2020-0108
- Kwong, J. Y., Goel, M., 2014, U.S. Patent No. 8,819,097. Washington, DC: U.S. Patent and Trademark Office.
- Lee, Y. H. G., Chen, C. I. H., 2009, Dynamic kernel function fast Fourier transform with variable truncation scheme for wideband coarse frequency detection. *IEEE Transactions on Instrumentation and Measurement*, 58(5), 1555-1562.

KAYNAKLAR DİZİNİ (devam)

- Luo, H. F., Liu, Y. J., Shieh, M. D., 2014, Efficient memory-addressing algorithms for FFT processor design. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 23(10), 2162-2172.
- Ma, Y., Wanhammar, L., 2000, A hardware efficient control of memory addressing for high-performance FFT processors. IEEE transactions on signal processing, 48(3), 917-921.
- Ma, Z. G., Yin, X. B., Yu, F., 2015, A novel memory-based FFT architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm. IEEE Transactions on Circuits and Systems II: Express Briefs, 62(9), 876-880.
- Mao, X. B., Ma, Z. G., Yu, F., Xing, Q. J., 2017, A continuous-flow memory-based architecture for real-valued FFT. IEEE Transactions on Circuits and Systems II: Express Briefs, 64(11), 1352-1356.
- Marti-Puig, P., 2009, Two Families of Radix-2 FFT Algorithms With Ordered Input and Output Data. IEEE Signal Processing Letters, 16, (2), 65–68.
- Morris, L., 1978, A comparative study of time efficient FFT and WFTA programs for general purpose computers. IEEE Transactions on Acoustics, Speech, and Signal Processing, 26(2), 141-150.
- Nee, R. V., Prasad, R., 2000, OFDM for wireless multimedia communications. Artech House, Inc.
- Oruklu, E., Xiao, X., Saniie, J., 2012, Reduced memory and low power architectures for CORDIC-based FFT processors. Journal of Signal Processing Systems, 66(2), 129-134.
- Proakis, J. G., Manolakis, D. G., 2007, Digital Signal Processing—Principles, Algorithms & Applications”, Pearson Education, Prentice Hall, 2007.
- Saleh, H. H., Swartzlander, E. E., 2012, FFT implementation with fused floating-point operations. IEEE Annals of the History of Computing, (02), 284-288.
- Sansaloni, T., Pérez-Pascual, A., Torres, V., Valls, J., 2007, Scheme for reducing the storage requirements of FFT twiddle factors on FPGAs. The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, 47(2), 183-187.
- Seke, E., 2017, VHDL Örnekleriyle Sayısal Haberleşmeye Giriş, kitap- Seçkin Yayıncılık.
- Seke E., Kaya Z., 2017, Serial-in-Parallel-out FFT. 5th International Conference on Advanced Technology Sciences.

KAYNAKLAR DİZİNİ (devam)

- Seke, E., Kaya, Z., 2018, Single Bank Dual-Port Memory-Based FFT for Field Programmable Gate Arrays. Eskişehir Technical University Journal of Science and Technology A-Applied Sciences and Engineering, 19(4), 796-804.
- Suma, M. N., Kanmani, B., 2011, Developments in orthogonal frequency division multiplexing (OFDM) system-a survey. In 2011 Second Asian Himalayas International Conference on Internet (AH-ICI) (pp. 1-4). IEEE.
- Sung, T. Y., Hsin, H. C., Cheng, Y. P., 2010, Low-power and high-speed CORDIC-based split-radix FFT processor for OFDM systems. Digital Signal Processing, 20(2), 511-527.
- Tian, Y., Hei, Y., Liu, Z., Shen, Q., Di, Z., Chen, T., 2018, A modified signal flow graph and corresponding conflict-free strategy for memory-based FFT processor design. IEEE Transactions on Circuits and Systems II: Express Briefs, 66(1), 106-110.
- Tsai, P. Y., Lin, C. Y., 2010, A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 19(12), 2290-2302.
- Wey, C. L., Lin, S. Y., Tang, W. C., Shiue, M. T., 2007, High-speed, low cost parallel memory-based FFT processors for OFDM applications. In 2007 14th IEEE International Conference on Electronics, Circuits and Systems (pp. 783-787). IEEE.
- Wey, C. L., Lin, S. Y., Tang, W. C., 2007, Efficient memory-based FFT processors for OFDM applications. In 2007 IEEE International Conference on Electro/Information Technology (pp. 345-350). IEEE.
- Winograd, S., 1978, On computing the discrete Fourier transform. Mathematics of computation, 32(141), 175-199.
- Xia, K. F., Wu, B., Xiong, T., Ye, T. C., 2017, A memory-based FFT processor design with generalized efficient conflict-free address schemes. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25(6), 1919-1929.
- Xiao, X., Oruklu, E., Saniie, J., 2008, An Efficient FFT Engine With Reduced Addressing Logic. IEEE Transactions on Circuits and Systems II: Express Briefs, 55, (11), 1149–1153.
- Xiao, X., Oruklu, E., Saniie, J., 2009, Fast memory addressing scheme for radix-4 FFT implementation. In 2009 IEEE International Conference on Electro/Information Technology (pp. 437-440). IEEE.

KAYNAKLAR DİZİNİ (devam)

- Xing, Q. J., Ma, Z. G., Xu, Y. K., 2017, A novel conflict-free parallel memory access scheme for FFT processors. IEEE Transactions on Circuits and Systems II: Express Briefs, 64(11), 1347-1351.
- Yang, C. H., Yu, T. H., Markovic, D., 2011, Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example. IEEE journal of solid-state circuits, 47(3), 757-768.

