İSTANBUL TEKNİK ÜNİVERSİTESİ

ELEKTRİK – ELEKTRONİK FAKÜLTESİ

FFT ALGORİTMALARININ FPGA ÜZERİNDE GERÇEKLENMESİ

BİTİRME ÖDEVİ

Tuba AYHAN

040040301

Bölümü: Elektronik Mühendisliği

Programı: Elektronik Mühendisliği

Danışmanı: Yrd. Doç. Dr. Müştak Erhan YALÇIN

MAYIS 2008

ÖNSÖZ

Bitirme ödevim boyunca bilgilerinden faydalandığım ve bana çok değerli zamanını ayırarak yardımcı olan Sayın Yrd. Doç. Dr. Müştak Erhan YALÇIN'a sonsuz saygı ve teşekkürlerimi sunarım.

Bu projenin gerçeklenmesinde, bana Gömülü Sistemler Laboratuarında deneyimlerini aktararak yardımcı olan arkadaşlarıma teşekkürlerimi sunarım.

Mayıs 2008 Tuba Ayhan

İÇİNDEKİLER

ÖZET

Ayrık Fourier dönüşümü (Discreet Fourier Transform- DFT), bir işaretin frekans domeni karşılığının, zaman domenindeki ifadesinden daha sık kullanıldığı sayısal işaret işleme uygulamalarında gerekli bir dönüşümdür. Ancak, ayrık Fourier dönüşümünün getirdiği işlem yükü, maliyeti arttırabilir; ya da işlem süresini arttırarak, giriş işaretinin örneklenebileceği en yüksek frekans değerini sınırlayıp, frekans çözünürlüğü düşürebilir. Bu nedenle sayısal işaret işleme uygulamalarında işlem yükü daha az olan ve sayısal işlemci yapısına daha uygun olan Hızlı Fourier Dönüşümü (Fast Fourier Transform- FFT) tercih edilir.

Bu çalışmada, çeşitli hızlı Fourier dönüşüm algoritmaları sahada programlanabilir kapı dizileri (Field Programmable Gate Arrays, FPGA) üzerinde gerçeklenmiştir. Tasarım süresinin kısalığı, tekrar tekrar kullanılabilir olması, test aşamasının kolaylığı ve maliyetinin düşük olması nedeniyle FPGA VLSI (Very Large Scale Integrated Circuit- Çok Geniş Ölçekli Tümdevre) tasarımlarda sıkça kullanılan bir cihazdır.

Bu çalışmada incelenen her bir algoritma için önce o algoritmanın teorik analizi ve MATLAB simülasyonu yapılmış, daha sonra algoritmaya ilişkin devre Verilog HDL ile tasarlanmış, bilgisayar benzetimi yapılmış ve en son yazılan kod FPGA'ya aktarılarak gerçeklenen devre gerçek ortamda çalıştırılıp test edilmiştir.

Bu projede değerlendirilen FFT algoritmalarının genel kullanım biçimleri gözetilerek, her biri için kullanılan toplama ve çarpma alt blokları tekrar tasarlanmış, sabit noktalı, tek duyarlı kayan noktalı veya yarı duyarlı kayan noktalı aritmetik kullanılmıştır. Böylece algoritmaya göre, sonuçların kesinliği, gerçeğe yakınlığı ve FPGA içinde kullanılan alan açısından optimizasyon sağlanmaya çalışılmıştır.

Gerçek ortamda tasarımlar test edilirken, SPARTAN3E geliştirme kiti kullanıldı. Giriş işareti, bir işaret üretecinden alındı ve 8 bitlik analog sayısal dönüştürücü üzerinden FPGA' ya aktarıldı. Alınan 8 bitlik sabit noktalı veri FFT algoritmasında kullanılacak olan forma dönüştürüldü. Dönüşüm sonucu ise, hedef sistemin yapısına uygun bir forma dönüştürüldü. Hedef sistem, dönüşüm sonuçlarının okunacağı VGA gibi bir ekran olabileceği gibi, üzerinde bir işaret işleme algoritması koşulan başka bir FPGA da olabilir. Projede, frekans çözünürlüğü 32 noktaya kadar olan FFT algoritmalarının çıkışı osiloskoptan gözlenmiştir.

Projede 8 bit ADC ve DAC, SPARTAN3E FPGA içeren Starter Kit geliştirme kiti kullanılmıştır. Donanım tanımlama dili olarak Verilog HDL kullanılmış, Xilinx ISE programında derlenmiştir. Benzetim ortamı olarak ise ModelSimXE ve MATLAB kullanılmıştır.

SUMMARY

Discreet Fourier Transform (DFT), is a widely used transform where the frequency domain representation of a signal is more useful than its time domain equivalent. However, the operational cost of discreet Fourier transform increases the device cost; or increased operation time restricts maximum sampling frequency for input sequence. Therefore, frequency resolution decreases. Hence, in applications of digital signal processing, a similar transform whose operation load is less, Fast Fourier Transform (FFT) which is more suitable for digital signal processor architecture is preferred.

During this work, various fast Fourier transform algorithms are realized on Field Programmable Gate Array (FPGA). FPGA is handled for VLSI (Very large Scale Integrated Circuit) designs due to the short design period, being reusable, providing advantages during test step, and low cost.

Each algorithm in this project is firstly searched theoretically and MATLAB simulation of each is analyzed. Then, a circuit concerning that algorithm is designed using Verilog HDL and simulation on computer is made then the circuit is tested on FPGA .

Basic modules such as adders and multipliers are redesigned and single precision floating point, half precision floating point or fixed point arithmetic is used for each algorithm concerning the general usage advantages of the related FFT algorithm. Thus, accuracy of results, reliability or device cost in FPGA is tried to be optimized according to the needs of the algorithm.

For the real time tests, SPARTAN 3E starter kit is used. Input is taken from signal generator and digitalized trough an 8 bit analogue to digital converter. 8 bit fixed point signal is converted to the form that will be used in FFT block if needed. Output of FFT block is then converted to the form that can be used in the target system. Target system can be a screen to read the results, such as LCD as well as another FPGA running another signal processing algorithm. In this project, output of fast fourier transforms with frequency resolutions up to 32 points are screened on oscilloscope.

A starter kit containing SPARTAN 3E FPGA is used. Verilog HDL is used as hardware description language, program is compiled with Xilinx ISE and ModelSim and MATLAB are used as simulation tools.

1. GİRİŞ

Ses ve konuşma işaretlerinin işlenmesi, sonar ve radar uygulamaları, sayısal görüntü işleme, istatistiksel işaret işleme, spektral kestirim, haberleşme sistemleri için işaret işleme, biyomedikal işaret işleme gibi alt dalları bulunan sayısal işaret işleme günümüzde uygulama alanlarını genişletmektedir.

Sayısal işaret işlemeye dair çalışmalar standart bilgisayarlar üzerinde yapılırken, daha sonraları sayısal işaret işlemciler kullanılmaya başlanmıştır. Böylelikle sayısal işaret işleme çalışmalarının pratiğe geçirilmesi kolaylaşmıştır.

Sahada programlanabilir kapı dizileri (Field Programmable Gate Arrays, FPGA), sayısal işaret işleme için kullanılabilecek teknolojilerdendir. Aynı amaç için kullanılabilecek diğer bir araç olan sayısal işaret işlemcilerden programlanma şekli ve mimarisi bakımından çok farklıdır. Öte yandan, ASIC (Application Specific Integrated Circuit- Uygulamaya Özel Tümleşik Devre) teknolojisine de imalat sonrasında yapılandırılabilmesinden dolayı çok yakın değildir.

Bitirme projesi kapsamında sayısal işaret işleme uygulamalarında sıkça kullanılan FFT (Fast Fourier Transform- Hızlı Fourier Dönüşümü) algoritmaları incelenmiştir. Bu projenin amacı her biri farklı uygulamalara sahip FFT algoritmalarının işaret işleme uygulamaları başta olmak üzere diğer çalışmalarda kullanılmak üzere FPGA üzerinde gerçeklenmesidir.

Projenin ilk kısmında sahada programlanabilir kapı dizileri ve çalışmada kullanılan FPGA hakkında bilgi verilmiş, neden bu proje için FPGA'nın seçildiği açıklanmaya çalışılmıştır.

Sonraki kısımlarda ise ayrık Fourier dönüşümü ve hızlı Fourier dönüşümü hakkında genel bilgi verilmiş ve bazı FFT algoritmaları hakkında detaylı bilgi verilmiş ve FPGA üzerinde gerçeklemeleri anlatılmıştır.

2. SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ (FPGA)

Programlanabilir Lojik Aygıt (Programmable Logic Device- PLD) terimi, imalat aşaması bittikten sonra işlev kazandırılan lojik elektronik birimleri için kullanılır. Üretim aşamasında, (yarı iletken üzerine kurma, paketleme…) PLD'leri tanımlanmış bir fonksiyonu yoktur, programlanabilir mantık birimleri içerir ve bu birimler arasındaki bağlantıların ayarlanmasıyla aygıt programlanabilir.

Bu anlamda Salt Oku Bellekler (ROM) ilkel PLD olarak düşünülebilir. Günümüze kullanılan PLD'ler ise 1970 yılında Texas Insturements firmasının 17 giriş 18 çıkıştan ve 8 JK tipi flip floptan oluşan programlanabilir cihazı piyasaya sürmesiyle ortaya çıkmıştır [1]. İlerleyen yıllarda diğer firmaların da benzer cihazları piyasaya sürmeleriyle bellek ve giriş çıkış sayısı bakımından çok daha büyük cihazlar üretildi. 1978'de programlanabilir VEYA dizilerinden oluşan ilk PAL (Programmable Array Logic-programlanabilir mantık dizisi) üretildi [1].

PAL ve PAL benzeri GAL (Generic Array Logic - silinip tekrar programlanabilen PAL), birkaç yüz kapı büyüklüğüne ulaşabilir. Daha büyük mantık devreleri için birkaç PLD'nin programlanabilir bağlantılarla birleştirilmesiyle oluşturulan CPLD (Complex PLD- Karmaşık PLD) ise binlerce kapı içerir.

Bir başka programlanabilir lojik aygıt ailesi ise kapı dizisi teknolojisine dayalı Sahada Programlanabilir Kapı Dizileri (FPGA) dir. FPGA içinde mantık kapıları ve flip-floplardan oluşan lojik birimler kolon veya matrislerde sıralanmıştır. Birimler arasındaki programlanabilir bağlantı ağları sayesinde lojik birimler diğer lojik birimlerle birleşerek daha çok birim ile daha karmaşık işlemler gerçekleştirebilir. Birimler arasındaki bağlantılar kısaldıkça, devrenin çalışabileceği maksimum hız artar. Bu nedenle ara bağlantılar sadece birbirine komşu lojik birimler arasında kurulur. CPLD ve FPGA' ların büyüklüğü kullanılabilir kapı sayısıyla belirtilir. Genelde büyüklük kullanılabilir maksimum iki girişli NAND kapısı sayısına denk gelir.

FPGA ve CPLD lojik elemanların sayısı bakımından büyük tasarımlar yapmaya elverişli olsa da, daha büyük devreleri kurabilmek yeni sorunlar ortaya çıkarmıştır. Çok büyük devrelerde saat işaretinin her bir flip floba aynı anda ulaştırılması

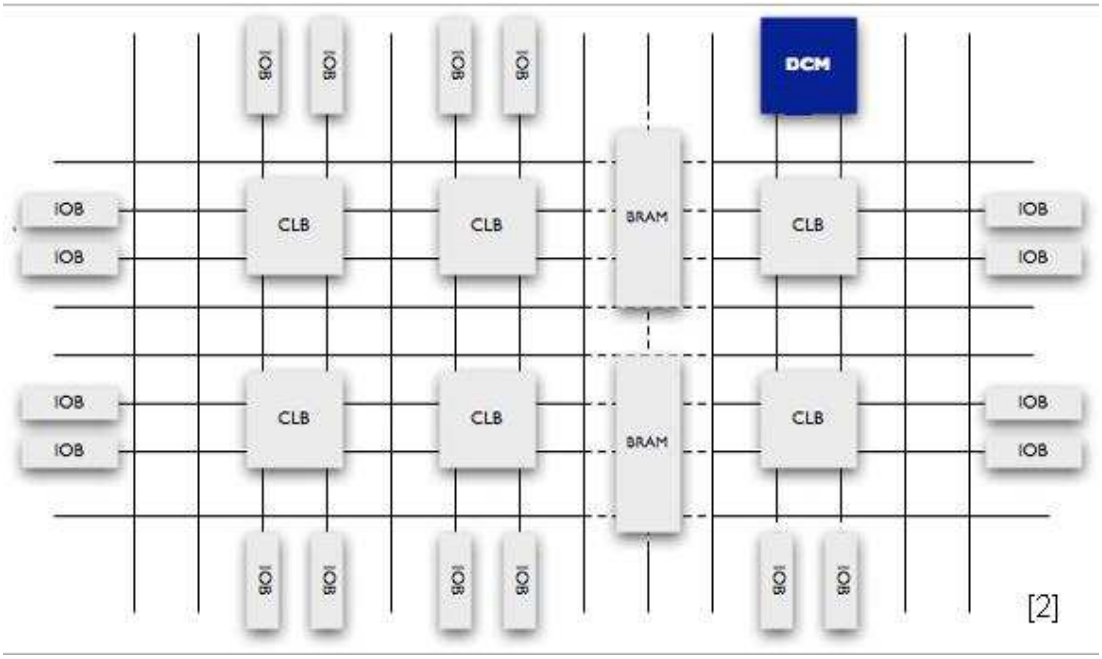
zorlaşmaktadır. Flip floplara farklı zamanlarda saat darbesi gelmesi devrede beklenmeyen davranışlara neden olabileceği için flip flopların saat girişlerindeki kaymayı en aza indirebilmek amacıyla FPGA ve CPLD' lerde saate özel bir evrensel hat kullanılır. Bu hat yüksek hızlı bir veri yoludur.

Devrenin büyük olmasının yarattığı bir diğer sıkıntı da CPLD ve FPGA' ların PAL kadar kolay programlanamıyor oluşudur. Ancak FPGA ve CPLD' lerin yaygılaşmasıyla onlara özel programlama araçları da gelişmiştir.

FPGA ile CPLD arasında büyük farklara yoktur, ikisi arasında seçim genelde tasarımcının tercihine, ekonomik ve ulaşılabilirlik faktörlerine bağlıdır. CPLD daha hızlı ve daha kesin zamanlama özelliklerine sahipken, FPGA' ların kapı yoğunluğu daha fazladır ve sadece üreticinin değil, kullanıcının da programlama yapmasına izin verir bu nedenle FPGA akademik kullanım için daha avantajlıdır.

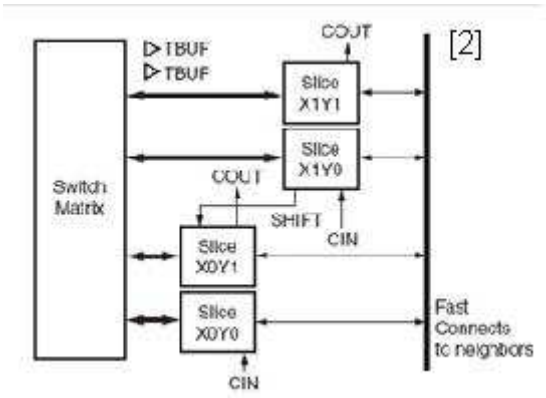
2.1. FPGA Mimarisi

Genel Mimarisi şekil 2.1'de gösterilen FPGA temel olarak üç bloktan oluşur; yapılandırılabilir lojik bloklar (CLB- Configurable Logic Block), giriş çıkış bankları (IOB- Input Output Bank) ve bağlantı blokları. Çeşitli FPGA'larda, çarpma bloğu gibi aritmetik işlem blokları, saat dağıtıcılar (DCM- Digital Clock Manager), bellek blokları (SRAM- Static Random Access Memory) gibi özel bloklar da bulunabilmektedir.



Şekil 2.1: FPGA Genel Yapısı [2]

Boole fonksiyonlarının gerçeklendiği Lojik blok mimarisi şekil 2.2'de gösterilmiştir. Sayısı ve özellikleri cihazdan cihaza değişiklik gösteren lojik blokların her biri 4 veya 6 girişli yapılandırılabilir anahtarlama matrisi (Switch Matrix), ile çoğullayıcı benzeri bir seçici devre ve flip floplar içeren dilimlerden (slice) meydana gelir. Anahtarlama matrisi kombinezonsal mantık devresi elemanı, ötelemeli yazmaç veya RAM olarak yapılandırılabilir.



Şekil 2.2: Yapılandırılabilir Lojik Blok [2]

Farklı sistemlerle daha iyi bağlantı kurulabilmesi için FPGA'lar birçok giriş çıkış standardını sağlayabilecek şekilde üretilmektedirler. FPGA'da giriş çıkışlar gruplar halinde bulunmaktadır. Aynı standardı destekleyen giriş çıkışlar aynı bank (IOB, Input

Output Bank) içinde gruplanırlar. Böylelikle, aynı FPGA, başka arabirimlere gerek kalmaksızın farklı cihazlarla bağlantı kurabilir.

Yapılandırılmış lojik blokların birbirine bağlanmasında ara bağlantı blokları kullanılır. Buradaki yapı PLA yapısına benzer.

Şekil 2.1'de gösterilen genel mimari içinde DCM (Digital Clock Manager) olarak adlandırılan blok, FPGA içindeki tüm flip floplara gönderilen saat işaretinin oluşturulduğu kısımdır, günümüzde FGPA'lar dışarıdan saat işareti uygulanmasına ihtiyaç duymaz.

2.2. FPGA Programlama

FPGA'nın programlanması içindeki lojik blokların yapılandırılması ve ara bağlantıların programlanabilir anahtarlar ile oluşturulmasıyla gerçekleştirilir. Programlanabilir anahtarın gerçeklenme teknolojisi, FPGA'nın programlanma teknolojisini belirler.

Şekil 2.1'deki SRAM (Statik RAM) hücrelerinin kontrol ettiği transistorların programlanarak ara bağlantıları oluşturduğu programlama şekline SRAM programlama teknolojisi adı verilir. SRAM hücreleri besleme kesildiğinde içlerindeki bilgiyi saklayamayacaklarından, bu teknoloji devrenin her açılışında FPGA'nın yeniden programlanmasını gerektirir. Ürün geliştirme aşamasında, ve FPGA içindeki verinin gizli tutulmasını gerektiren şifreleme benzeri uygulamalarda tercih edilir, ayrıca FPGA'nın sistem üzerinde yeniden programlanmasına olanak tanır [2]. Bu nedenle SRAM FPGA içinde büyük alan kaplamasına rağmen, üretilen FPGA'larda SRAM programlama teknolojisinden vazgeçilmemiştir.

Daha küçük alana ihtiyaç duyan antifuse programlama teknolojisinde ise, ara bağlantılar programlama sırasında gerilim uygulanarak oluşturulur. Böylece FPGA'lar tekrar programlanabilme özelliklerini kaybeder. Üretim aşamasında tercih edilen bir programlama teknolojisidir, örnek üretimi ve geliştirme için uygun bir çözüm değildir.

2.3. Bu Çalışmada Kullanılan FPGA ve Geliştirme Kiti Özellikleri

Bu çalışmada içinde SPARTAN 3E serisinden XC3S500E bulunduran SPARTAN 3E geliştirme kiti (Starter Kit) kullanılmıştır.

2.3.1. SPARTAN 3E XCS500E

Bölüm 2.1'de belirtilen temel bloklar dışında SPARTAN 3E ailesi FPGA'larında blok çarpıcılar ve blok RAM bulunmaktadır.

Temel bloklardan olan giriş çıkış banklarına, sadece giriş için kullanılan banklar eklenmiş, komşu banklar arasında DDR (Double Data Rate- İki kat hızlı) flip-floplarının paylaşımı sağlanarak alandan kazanılmış ve tüm giriş çıkış bloklarının gecikmesi programlanabilir hale getirilmiştir. Bu ailenin her bir üyesi 18 giriş çıkış standardını sağlamaktadır. [3]

FPGA içinde 4 adet DCM ve saat hattı bulunmaktadır. Ayrıca saat işaretinin dışardan alınmasına da müsaade edilmiştir.

Diğer bir temel blok CLB, şekil 2.2'de X ve Y indisleriyle gösterilen dörder dilimden oluşur. Her bir dilim ikişer LUT (Look Up Table- referans tablosu) ve tutucu (mandal-latch) veya flip flop olarak kullanılabilecek ikişer adet bellek elemanı bulundurur. LUT 16x1 bellek (RAM 16), 16 bir ötelemeli yazmaç (SRL 16), gerekli durumlarda çoğullayıcı, ve aritmetik blok olarak kullanılabilir. Bu ailede, her CLB birbirine eştir. Bir FPGA ailesinde, FPGA'nın içerdiği dilim (veya CLB) sayısı ayırt edicidir. Tablo 2.1'de bitirme çalışmasında kullanılan XC3S500E için lojik blokların nicel özellikleri verilmiştir.

Tablo 2.1: SPARTAN3E XC3S500E CLB Özellikleri [4]

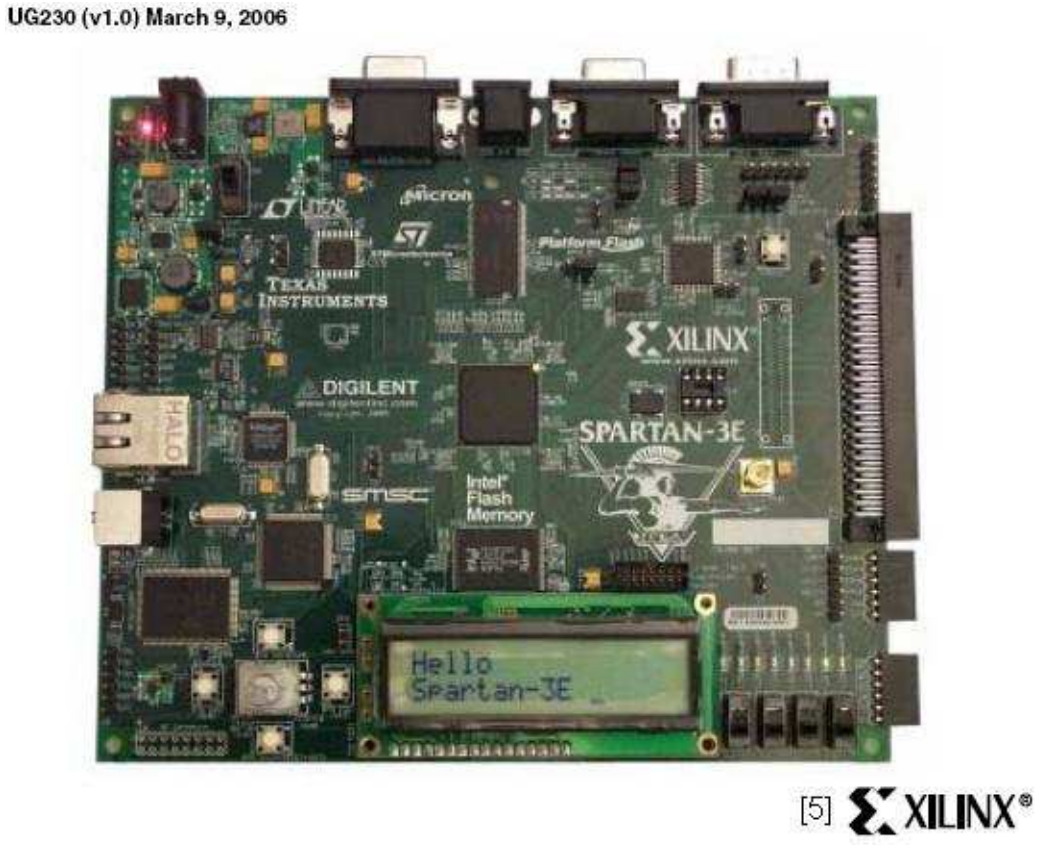
CLB (satır)	CLB (sütun)	CLB (toplam)	Dilim	LUT/Kapan	Eşdeğer Lojik Hücre	RAM 16/SRL	Dağınık RAM
46	34	1,164	4,656	9,312	10,476	4,656	74,496

Aritmetik işlemleri hızlandırmak ve bir dilim içinde bulunan kapan sayısından fazla kapan gerektiren işlemleri, CLB harcamadan yapabilmek için blok çarpıcılar tanımlanmıştır. Her biri 18x18 çarpma gerçekleştirebilen bu bloklardan XC3S500E içinde 20 adet bulunmaktadır [3]. Çarpıcı bloklarının varlığı, büyük çarpma işlemlerinin gerçeklendiği Fourier dönüşümünde, ayrı CLB'lerde bulunan kapanları kullanmadan çarpma işleminin gerçeklenmesini sağlar.

Bir diğer ek blok ise 360K bit adreslenebilir yere sahip RAM bloğudur [3]. Blok RAM, uzunluk ve genişliği farklı şekillerde yapılandırılabilir, çift adres hattı RAM'in çift giriş çıkışlı olmasını sağlar. Blok RAM kullanılan tasarımlarda, RAM'e ulaşım zamanı ve RAM'e yazılacak olan datanın tutulma zamanı ile, cihazın çalışma frekansı arasındaki uyuma dikkat edilmelidir. Blok RAM, gerektiğinde ROM olarak da kullanılabilir. Blokların bir kısmı Fourier dönüşümü gerçeklenirken katsayıları tutmak amacıyla ROM olarak, ve işlemler sırasında kullanmak amacıyla RAM olarak yapılandırılmıştır.

2.3.2. SPARTAN 3E Geliştirme Kiti (STARTER KIT)

Şekil 2.3'de görülen geliştirme kitinin üzerinde, bölüm 2.3.1'de anlatılan FPGA dışında, kullanıcıya ayrılmış giriş-çıkış pinleri, 16 Mbits SPI (Serial Parallel Interface-Seri Paralel Arabirim), paralel NOR Flash, 2 satır 16 karakter LCD ekran, PS/2 fare veya klavye girişi, VGA çıkış portu, çeşitli haberleşme portları (Ethernet, RS232 gibi), 50MHz osilatör, programlama ve çözme için USB girişi, 3 adet Digilent 6-pin konektör, 4 çıkışlı DAC, 2 girişli ADC, anahtarlar, LED'ler, butonlar bulunmaktadır.



Şekil 2.3: SPARTAN 3E Geliştirme Kiti [5]

Bitirme projesinde, FPGA ve programlama üniteleri dışında, 6-pin konektörler ile SPI ADC ve DAC bağlantısı için, LCD ve LED'ler sonuçları gözlemek için, butonlar ile anahtarlar kontrol ve hata ayıklama için kullanılmıştır.

3. AYRIK FOURİER DÖNÜŞÜMÜ

Fourier Analizi, bir fonksiyonun farklı frekanslardaki sinüzoidal fonksiyonların toplamı biçiminde açılımıdır. Periyodik ve ayrık bir işaretin fourier analizi için Ayrık Fourier Dönüşümü (Discrete Fourier Transform- DFT) kullanılır. Ayrık Fourier dönüşümü, bir işaretin frekans spektrumunu bulmak için kullanılabileceği gibi, bir darbe cevabı bilinen bir sistemin frekans cevabının; frekans cevabı bilinen bir sistemin darbe cevabının bulunması için ya da daha karmaşık işlemlerde ara adım olarak kullanılır [6]. Zaman domeninde verilen ayrık bir işretin frekans domenindeki karşılığının DFT ile bulunabilmesi için, işaretin sıfırdan farklı değerlerinin sonlu sayıda olması gerekir. Bu tip bir işaret sürekli bir işaretin uygun şekilde örneklenmesi ile elde edilir.

Ters dönüşümde ise, orijinal işaret tam olarak elde edilemez; çünkü DFT ile elde edilen frekans değerleri (sinüzoidal fonksiyonlar) sonlu sayıdadır ve DFT ile işaretin sadece sonlu bir bölgesi analiz edilmiştir.

N adet gerçek değerden oluşan zaman domenindeki bir x[k] işareti, ayrık Fourier dönüşümünden sonra, N/2 +1 adet kosinüs ve N/2 +1 adet sinüs işareti biçiminde gösterilir. Kosinüs dalgalarının genliği frekans domenindeki işaretin reel kısmına karşı gelirken, sinüs dalgalarının genliği de sanal kısımlara karşı düşer. Giriş işaretinin karmaşık değerlerden oluşması halinde, kompleks DFT, N adet karmaşık değeri frekans domeninde N adet karmaşık değere dönüştürür. DFT baz fonksiyonları şu şekilde bulunur:

$$c_k[i] = \cos(2i\pi k / N) \tag{3.1}$$

$$s_k[i] = \sin(2i\pi k / N) \tag{3.2}$$

Burada $c_k[0]$, giriş işaretinin DC değerini verir, $s_k[0]$ ise her zaman 0 olduğu için dikkate alınmaz, orijinal işaretin tekrar elde edilmesinde etkisi olmaz. Giriş işareti, sinüs ve kosinüs işaretlerinin toplamı olduğuna göre,

$$x[i] = \sum_{k=0}^{N/2} \mathrm{Re}\,\bar{X}[k]\cos(2i\pi k / N) + \sum_{k=0}^{N/2} \mathrm{Im}\,\bar{X}[k]\sin(2i\pi k / N) \tag{3.3}$$

yazılabilir. İfadeler aşağıdaki şekilde genişletilebilir.

$$\mathrm{Re}\,\bar{X}[k] = \frac{\mathrm{Re}\,X[k]}{N/2} \tag{3.4}$$

$$\mathrm{Re}\,\bar{X}[0] = \frac{\mathrm{Re}\,X[0]}{N} \tag{3.5}$$

$$\mathrm{Re}\,\bar{X}[N/2] = \frac{\mathrm{Re}\,X[N/2]}{N} \tag{3.6}$$

$$\mathrm{Im}\,\bar{X}[k] = -\frac{\mathrm{Im}\,X[k]}{N/2} \tag{3.7}$$

Yukarıdaki gibi tanımlanan bir giriş işaretinin ayrık fourier dönüşümü hesaplanmak istendiğinde, sinüs ve kosinüs ifadelerinin karmaşık düzlemde karşılıkları kullanılarak,

$$X(k) = \sum_{n=0}^{N-1} \left(x(n) e^{\frac{-j2\pi}{N} kn} \right) \tag{3.8}$$

yazılabilir.

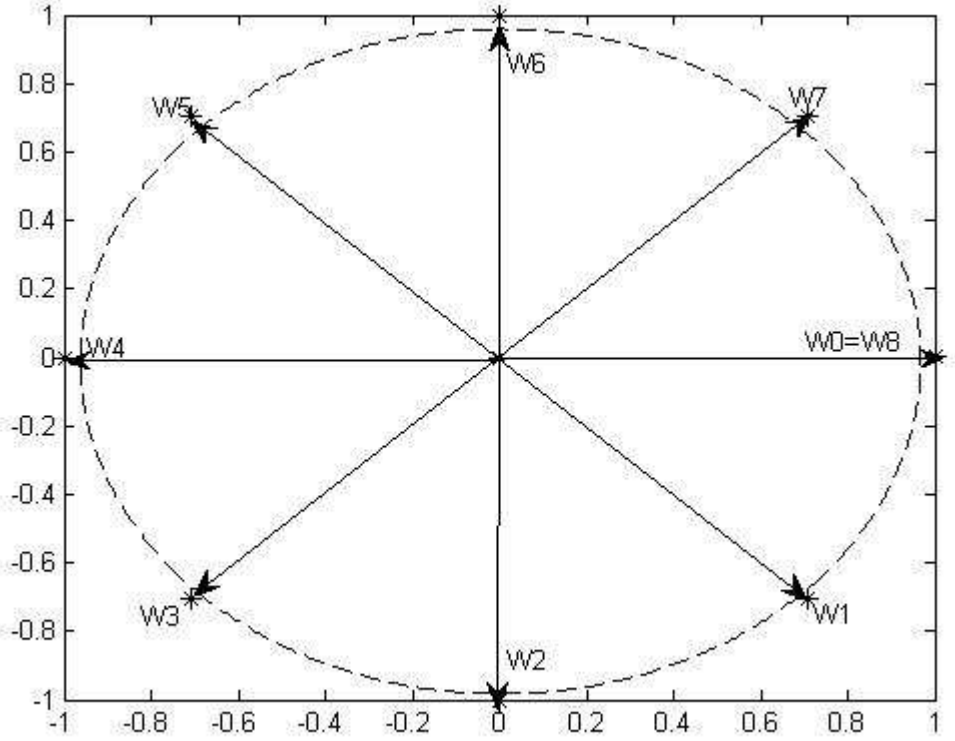
Burada X (karmaşık sayı), x ile gösterilen zaman domenindeki işaretin frekans domenindeki karşılığını ifade eder. Dönüşüm kısaca şu şekilde gösterilebilir:

$$\mathbf{X} = \mathcal{F}\{\mathbf{x}\} \tag{3.9}$$

Frekans domeninden zaman domenine geçerken, ters Fourier dönüşümü kullanılır:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \left(X(k) e^{\frac{j2\pi}{N} kn} \right) \tag{3.10}$$

Denklem (3.8)'de, x(n) teriminin yanında yer alan katsayılar incelendiğinde, katsayıların birim çember üzerinde olduğu görülmektedir. Katsayıların çember üzerindeki dağılımı, nokta sayısı N'ye bağlıdır. N= 8 için şekil 3.1'de katsayıların birim çember üzerinde nasıl dizildiği gösterilmiştir.



Şekil 3.1: Periyodik ve Simetrik Katsayılar

3.1. Ayrık Fourier Dönüşümü Hesaplama Yöntemleri

DFT birbirinden çok farklı 3 yöntemle hesaplanabilir. Anlaşılması en basit olanı, denklem (3.8)'de verilen işlemi doğrudan gerçeklemektir; ancak bu yöntem pratik hesaplamalar için yetersiz kalır. Uygulanabilirliği daha yüksek olan bir yöntem, korelasyon ile verilen işaret içinde bilinen dalga formlarının aranmasına dayanır. Bu çalışmada incelenen yöntem olan hızlı Fourier dönüşümü diğer iki yöntemden daha sık tercih edilen bir yöntemdir [6].

3.1.1. Eşzamanlı Hesaplamalarla DFT

DFT hesaplamak için en basit ancak pratikte uygulanabilirliği en zayıf yöntemdir. Zaman domeninde verilen N değer ile, frekans domeninde olan N adet bilinmeyenin temel matematiksel denklemlerle hesaplanabilmesi için lineer bağımsız N adet denklemin çözülmesi gerekir. N adet değişken, N adet sinüzoidal demektir. Bu N sinüzoidin her birinin bir katsayıyla çarpımının toplamı giriş işaretini verecektir, N nokta için bu toplam yazılırsa N tane N bilinmeyenli denklem elde edilmiş olur. Tek

dezavantajı çok fazla işlem yükü getirmesi değildir, kullanılan sinüzoitler lineer bağımsız olmak zorundadır.

3.1.2. Korelasyonla DFT

Bilinen bir dalga formunun başka bir işaret içinde aranmasıdır. N nokta için DFT işlemi uygulanacaksa, frekans domeninde N/2 sanal, N/2 gerçel değer bulunmalıdır. Korelasyon kullanıldığında, bilinen bir dalga formu ile verilen işaret çarpılır ve elde edilen yeni işaretteki tüm noktaların değerleri toplanır. Elde edilen değer bu iki işaretin birbirine ne kadar benzediğinin bir ölçüsüdür. Kullanılan referans dalga formunun, frekans domenindeki sanal değerleri bulmak için sinüs, gerçel değerleri bulmak için kosinüs olması gerektiği de göz önüne alındığında, aşağıdaki ifadeleri elde ederiz:

$$\mathrm{Re}\{X[k]\} = \sum_{i=0}^{N-1} x[i]\cos(2i\pi i / N) \tag{3.11}$$

$$\mathrm{Im}\{X[k]\} = -\sum_{i=0}^{N-1} x[i]\sin(2i\pi i / N) \tag{3.12}$$

DFT hesaplamada 32 noktaya kadar bu yöntem, daha fazla nokta için Hızlı Fourier Dönüşümü (FFT) kullanılır [7].

4. HIZLI FOURİER DÖNÜŞÜMÜ (FFT)

Bölüm 3.1'de anlatılan DFT'nin fiziksel anlamını matematiksel olarak birebir ifade eden yöntemlerin işlem yükü hesaplama için farklı yöntemler kullanılmasını zorunlu kılmaktadır; çünkü yukarıdaki yöntemler FFT ile karşılaştırıldığında çok yavaş kalmaktadır.

Ancak DFT hesaplamakta kullanılabilecek, işlem yükü görece az her yöntemin FFT olarak değerlendirilip değerlendirilemeyeceği konusunda çeşitli sınıflandırma yöntemlerine gidilmiştir. Bir kısım çoklu indeksleme yöntemleri kullanan algoritmaları FFT algoritması olarak görürken (Burus), bir kısım ise DFT'nin temel işlem yükü olan $O(N^2)$ kadar işlemi, $e^{\frac{j2\pi}{N}}$ ifadesinin simetri, asal katlar gibi özelliklerinden yararlanarak azaltan algoritmaları FFT algoritması sınıfında inceler [6]. İşlem yükünü çeşitli yollarla azaltan algoritmaları göz önüne alındığında, bu tip algoritmaların temelde iki yöntem kullandığını görülür. Bunlardan bir tanesi $(e^{\frac{j2\pi}{N}})_k$ gibi tek boyutlu indekslenmiş katsayıların yerini veya ifadesini değiştirerek DFT tanımında verilen toplama işlemini bir dairesel konvolüsyona dönüştürmektir. Bu yöntem işlem yükünü biraz azaltabildiği gibi, FFT işlemi FIR filtre yapısıyla gerçeklenebilir hale getirilmiş olacağından, gerçeklenebilirliği de arttırmaktadır. Rader, Bluestein Chirp-z ve Winograd I algoritmaları bu tip algoritmalardır [6].

Bir diğer yol ise işlem yükünü oldukça azaltan ve DFT hesaplamaları için sıkılıkla kullanılan çok boyutlu indeksleme yöntemleridir. Cooley-Tukey, Winograd II ve Good-Thomas algoritmaları da bunlara örnek teşkil eder. Bu tip algoritmalar çarpma sayısını azaltanlar ve toplama sayısını azaltanlar olarak gruplanabilmektedir. Çarpma sayısını azaltan algoritmalar ortak bölenlerin bir araya toplanmasına dayanan Cooley-Tukey algoritması gibi algoritmalardır [7]. Toplama sayısının azaltılması için de, Winograd algoritmasında olduğu gibi, katsayıların indeksleri asal çarpanlarına ayrılarak tekrar düzenlenir [7]. Günümüzde işaret işleme için kullanılan DSP ve FPGA gibi araç ve devreler için çarpma işleminin yükü toplama işleminin yükünden

daha fazladır [8]. Çarpma işlemlerinin sayısını azaltan Cooley-Tukey algoritması bu nedenle sıkça kullanılmaktadır [6].

Konvolusyon ile ayrık Fourier dönüşümü hesaplama yöntemleri ile yalnızca belli bir frekans bileşeninin genlik değerini vermesi açısından benzerlik gösteren Goertzel algoritması da hızlı Fourier dönüşümü algoritmaları arasında sayılır [9]. Goertzel Algoritması yalnızca tek bir genlik değeri ile ilgilenildiği zaman, frekans domenineki değerin sanal ve gerçel kısımlarının ayrı ayrı hesaplanması gerekmemesinden yola çıkarak işlem sayısını azaltan bir algoritmadır [10].

4.1. Goertzel Algoritması

FFT algoritmaları elde edilirken, amaç gereksiz işlemlerden kurtulmak, böylece daha ucuz ve daha hızlı işlem yapabilmektir. Bahsedilen gereksiz işlemler, önceden hesaplanmış olan verilerin tekrar hesaplanmalarıdır. Örneğin, bütün bir spektrumun dönüşümü isteniyorsa, simetrilere bağlı olarak yapılan indirgemeler, işlem sayısını yarı yarıya azaltır; ancak, tüm bir spektrumla değil de az sayıda ya da tek bir frekans bileşeni ile ilgileniliyorsa, katsayılardaki simetri işlemlerde azalmaya neden olmamaktadır [10]. Hesaplanan her frekans bileşeninin neden olduğu işlem karmaşıklığı, zaman domenindeki her bir örnek- x_n, hesaba katıldığı için en az O(N) olmalıdır [10]. Ancak, eğer giriş dizisinde aranan frekans veya frekanslar belliyse, kullanılan katsayılar hedef frekansa göre seçilebilir, böylece katsayıdan kaynaklanan karmaşıklık azaltılabilir. X[k] frekans domeninde yer alan hedef bileşen olmak üzere, denklem (3.8)' de $e^{\frac{-j2\pi}{N}kn}$ yerine W_N^{nk} konulduğunda denklem (4.1) elde edilir.

$$X(k)=\sum_{n=0}^{N-1}\left(x(n)W_N^{nk}\right) \tag{4.1}$$

$$X(k)=\sum_{n=0}^{N-1}\left(x(n)W_N^{-Nk}W_N^{nk}\right) \tag{4.2}$$

$$X(k)=\sum_{n=0}^{N-1}\left(x(n)W_N^{(N-n)(-k)}\right) \tag{4.3}$$

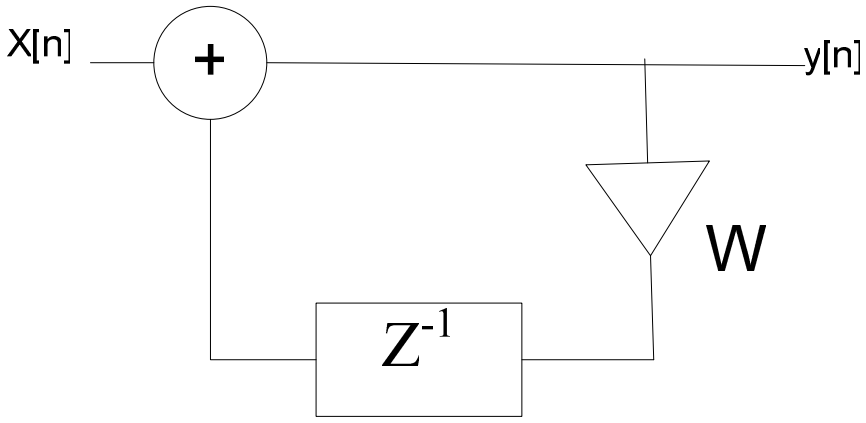
W_N^{-Nk} =1 olduğuna göre, denklem (4.1) tekrar düzenlendiğinde elde edilen (4.2) ifadesinin açılımı yapılarak denklem (4.4) elde edilir.

$$X[k] = x[0] + x[1]W_N^{k} + x[2]W_N^{2k} + x[3]W_N^{3k} + x[4]W_N^{4k} + \ldots + x[N-1]W_N^{(N-1)k} \quad (4.4)$$

Bu nedenle yukarıdaki denklemde verilen katsayılar, hedef frekansa göre seçilip, aynı alınabilir. Polinom Horner kuralına göre özyinelemeli olarak tekrar yazıldığında denklem (4.5)'e ulaşılmaktadır.

$$X[k] = x[0] + W(x[1] + W(x[2] + \ldots + W(x[N-2] + Wx[N-1]))) \quad (4.5)$$

Algoritmanın basit bir FIR (finite impuls response- sonlu darbe yanıtlı) süzgeç yapısı ile gerçeklenebileceği görülmektedir [9].



Şekil 4.1: Goertzel FFT Algoritması Blok Diyagram

Belli bir hedef frekans için, W değeri, kullanılacak nokta sayısı N ve örnekleme frekansı bilindiği halde, $e^{\frac{-j2\pi}{N}kn}$ ifadesinde yerine konarak elde edilmektedir. W karmaşık sayısıyla hesaplanan X[k] değeri de karmaşık bir değer olacaktır; ancak, çeşitli trigonometrik dönüşümlerle X[k] değerinin doğrudan genliğini hesaplamak da mümkündür. Dönüşümü gerçekleştirmek için önceden hesaplanması gereken katsayı (4.6) ile hesaplanır.

$$W = 2.\cos(2.\pi.k/N) \quad (4.6)$$

Burada N nokta sayısı, farklı değerlerde seçilebilir. N değerinin arttırılması, bir örnekleme periyodu içinde işleme sokulması gereken x(n) giriş değerlerinin sayısını arttırdığı için sistemin çalışabileceği maksimum örnekleme hızını düşürecektir. Öte yandan, N değeri aynı zamanda tespit edilebilecek frekans bileşeni sayısına eşit olduğu için nokta sayısının azaltılması frekans çözünürlüğünü düşürmektedir.

Bir işaretin içinde belli bir frekans bileşeninin bulunup bulunmadığının tespiti için kullanılan bu algoritma, DTMF (Dual Tone Multiple Frequency-Çift Tonlu Çoklu

Frekans Kodlama) detektör uygulamalarında tercih edilmektedir [11]. Örneğin DTMF kodlama sisteminde telefon üzerindeki 16 adet tuş (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, #, *, A, B, C, D), ikişer ton ile ifade edilmektedir. Bir karakteri temsil eden iki tondan biri 1kHz in üstündeki 4 frekanstan biri, diğeri 1kHz in altındaki 4 frekanstan biridir. DTMF detektör ile, alınan bir işarette bu sekiz frekans bileşeninden değerinden hangi ikisinin bulunduğunun tespiti yapılır. Goertzel algoritması bu uygulama için tüm işaretin frekans spektrumunu çıkarmaktan daha kolay, aynı zamanda aranan frekans değerleri belli olduğu için daha kesin sonuçlar sunabilmektedir [9].

4.1.1. MATLAB Benzetimi

Goertzel algoritmasının FPGA üzerine geçirmesinden önce, kullanılması gereken giriş ve çıkış uzunlukları ile ağırlıların hassasiyetlerinin belirlenmesi gerekmektedir. Bu nedenle Goertzel algoritmasının Fourier dönüşümü için kullanıldığı uygulamalardaki avantajının, gösterdiği hız ve kesinlik olduğu göz önünde bulundurularak, bir yapı öngörülmüştür. Kesinliğin sağlanabilmesi için kayan nokta aritmetiğinin kullanılması uygun görülmüştür. Telefon üzerindeki tuşları ifade eden sinyallerin bileşenleri tablo 4.1'de verilmiştir. Bu sekiz frekans için denklem (3.6)'ya göre hesaplanan ağırlıklar ise tablo 4.2'de gösterilmektedir. Burada verilen ağırlıklar örnekleme frekansı 8000 Hz kabul edilerek hesaplanmıştır.

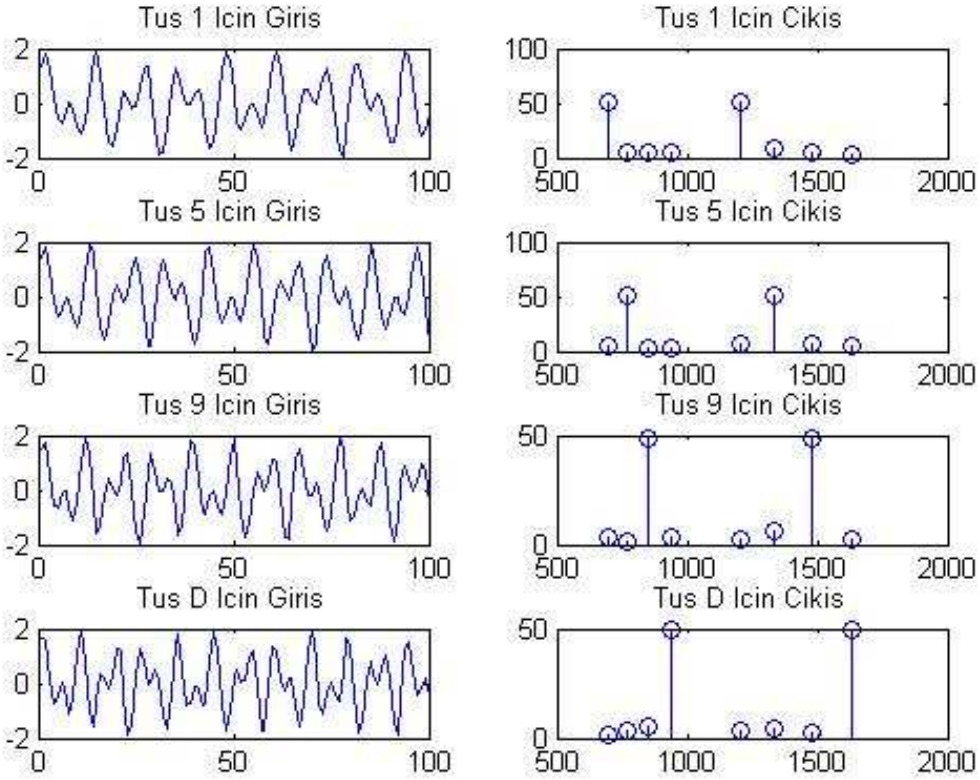
Tablo 4.1: Telefonda Kullanılan İşaretlerin Frekans Bileşenleri [11]

	1209 Hz	**1336 Hz**	**1477 Hz**	**1633 Hz**
669Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Tablo 4.2: DTMF için Frekanslara Göre Ağırlıklar

Frekans (Hz)	W
697	0.8411
770	0.8540
852	0.8625
941	0.8518
1209	0.8918
1336	0.9075
1477	0.9273
1633	0.9440

Telefon tuşlarından 1, 5, 9 ve D için, MATLAB ortamında ideal giriş işaretleri üretilmiş ve bulunan katsayılar ile işleme sokularak, içerdikleri frekans değerlerinin genlikleri bulunmuştur. Şekil 4.2'de gösterilen sonuçlarda, giriş işareti içinde olan bir frekans bileşenine ait genlik değeri, aynı işaret içinde diğer frekanslar için bulunan genlik değerinin yaklaşık 10 katıdır. Giriş işareti [-2 +2] arasında değişirken, frekans genliği [0 50] arasında değişmektedir. 8 bitlik analog sayısal çeviriciden alınan değerin [0-127] arasında olması nedeniyle, frekans genlik çıkışlarının daha fazla büyüyeceği açıktır. İşlem sırasında taşmaları engellemek için kayan noktalı aritmetik kullanılması gerekmektedir.



Şekil 4.2: Goertzel Algoritması ile DTMF İçin MATLAB Benzetimi Sonuçları

4.1.2. Kayan Nokta Aritmetiği

Çok büyük ve çok küçük sayılarla çalışılırken, kayan noktalı gösterim, sabit noktalı gösterime göre daha kesin sonuçlar sağlamaktadır. Ayrıca dinamik erim de arttırılır.

Kayan noktalı gösterimde sayılar işaret, üs *e*, üs tabanı *b* ve anlamlı kısım *s* olmak üzere dört bölümle ifade edilir. (4.7) eşitliğinde bir sayının kayan noktalı temsili gösterilmektedir.

$$x = \pm s \times b^{e} \tag{4.7}$$

İkilik sisteme geçildiğinde, sayıdaki tek işaret biti sayının işaretini verir. Üssün işaretini belirtmek için ise ikiye veya bire tümleme yöntemleri kullanılmaz, ya da işaret biti yazılmaz. Üs belirtilirken gerçek üs bir sayıyla (bias) toplanarak gerçek üssün alacağı en küçük değerin gösterilimi sıfır yapılmış olur. Gösterilen üs negatif olmaz.

Kayan nokta gösteriminde "ANSI/IEEE Std 754-1985" standardı kullanılır. Bu standartta üs yanlı biçimde ifade edilir. Anlamlı sayı kısmı *1.f* şeklinde gösterilir,

burada *f* sayının mantisi, 1 ise saklı bittir. Sayıların uzunlukları, duyarlılıklarını belirler [12]. Std 754 standardında 32 bit uzunluk tek duyarlı gösterime, 64 bit uzunluk ise çift duyarlı gösterime karşı gelmektedir. Bias yalılığı temsil etmek üzere, ikilik tabanda verilen sayının on tabanındaki karşılığı olan x (4.8) denkleminde gösterildiği gibi hesaplanır, tüm bitlerin sıfır olduğu durum onluk tabanda sıfıra karşılık gelmektedir.

$$x = (-1)^{isaret} \times 2^{e-yanlilik} \times 1.f \tag{4.8}$$

1987 yılında genişletilen ANSI/IEEE Std 754 standardında, 16 bit uzunluklu kayan noktalı sayılar yarı duyarlı olarak adlandırılmıştır. Yarı, tek ve çift duyarlı gösterimler için özellikler tablo 4.3'te verilmiştir.

Tablo 4.3: IEEE 754r Standardında Kayan Noktalı Gösterimler

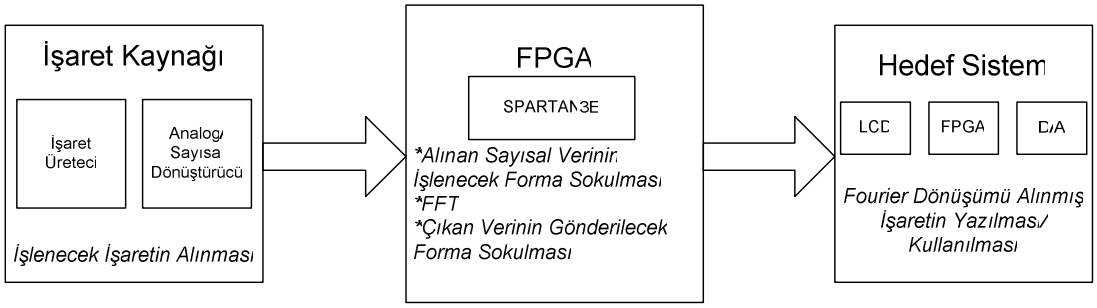
Özellik	Yarı Duyarlı	Tek Duyarlı	Çift Duyarlı
Toplam bit sayısı	16	32	64
Anlamlı bitler (*s*)	10	23	52
Üs bit sayısı	5	8	11
Üs yanlılığı	2^4 -1 = 15	2^7 -1 = 127	2^{10} -1 = 1023
En küçük değer	5.9×10^{-8}	1.2×10^{-32}	2.2×10^{-308}
En büyük değer	6.5×10^{8}	3.4×10^{32}	1.8×10^{308}

Kayan noktalı sayılarla işlem yapmak sabit noktalı sayılarla işlem yapmaktan daha farklıdır. Kayan noktalı sayılarla toplama yapılırken önce sayının mantisi kaydırılarak üsler eşitlenir, sayıların işaretine göre işlem yapılıp sonucun işareti belirlenir, daha sonra sonuç mantisi normalize edilerek 1.f biçimine getirilir, kullanılan duyarlılık ölçüsüne uygun üs ve mantis bit sayısına geri dönülmüş olur [13].

Çarpma işleminde ise üslerin eşitlenmesine gerek yoktur. Mantisler doğrudan çarpılabilir normalize edilerek ve 1.f standardına getirilir. Sonucun üssünün belirlenmesi için üsler toplanır ve yanlılık toplandan çıkarılır. Mantislerin çarpımına göre gerekiyorsa üs arttırılır veya azaltılır [13].

4.1.3. FPGA Üzerinde Gerçekleme

Bilgisayar ortamında benzetimin yapılabilmesi için yalnızca FFT algoritmasının donanım tanımlama dili ile yazılaması yeterlidir; ancak FFT algoritması gerçek ortamda test edilmek istendiğinde, diğer birimlerle bağlantısını ve haberleşmesini sağlamak için FFT modülüne başka modüller de eklenmelidir. FFT algoritması gerçeklenirken sistemin genel halinin göz önünde bulundurulması, FFT modülü üzerinde diğer modüllerden kaynaklanan kısıtlamaların dikkate alınabilmesi için gereklidir. Şekil 4.3'te verilen genel diyagramda, FPGA üzerinde işaret kaynağı ve hedef sistem ile haberleşme işaretleri üretilmesi ve alınması dışında, üç ayrı modül bulunduğu görülmektedir.



Şekil 4.3: Genel Gerçekleme Aşamaları

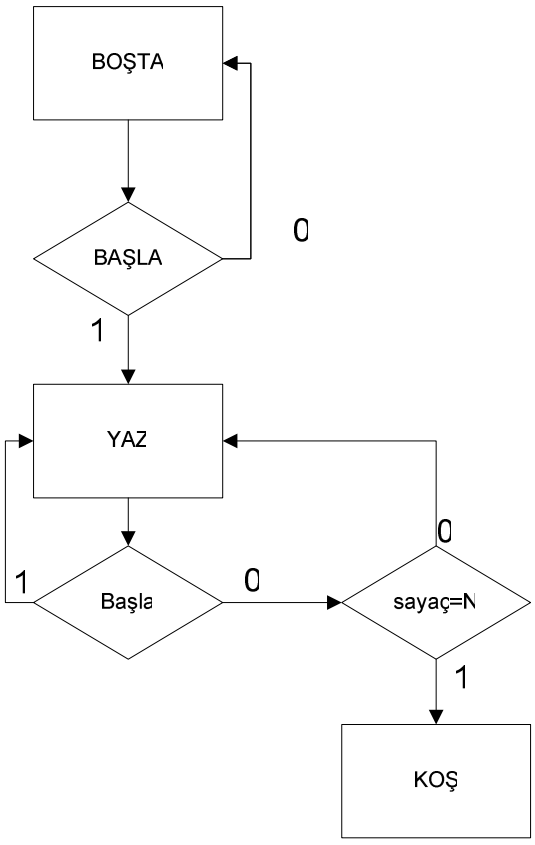
İşaret üretecinden alınan analog işaret, analog sayısal çevirici çıkışında bulunan işaretin alınması ve yeni işaretin gönderilmesi izninin verilmesi ile, hedef sisteme gönderilen işaretin kontrolü geliştirme kiti üzerinde bulunan SPI kontrolör programlanarak yapılmaktadır. SPI kontrolörün hızı, FFT modülü üzerinde bir kısıtlama oluşturmaktadır.

Öncelikle, girişin ve çıkışın tek duyarlı kayan nokta formatında olduğu FFT modülü tasarlanmış ve davranışsal benzetimi yapılmıştır.

N noktadan oluşan giriş dizisine Goertzel algoritması uygulandığında, (3.4) eşitliğinde verildiği gibi, önce x[N-1] değeri önceden belirlenmiş katsayı ile çarpılır, daha sonra da kendinden sonra gelen dizi elemanı ile toplanır. N-1 adet toplama ve N adet çarpma işlemi gerçekleştirildiğinde N nokta da işlenmiş demektir ve böylece ilgilenilen hedef frekansın genliğini veren X[k] elemanının değeri bulunur.

Gerçeklemede, şekil 4.4'te gösterilen üç durumdan bahsedilebilir. Bunlar, BOŞTA, YAZ ve KOŞ durumlarıdır. BOŞTA durumu, işaret gelmesinin ve FFT işlemine başlanmasının beklenildiği durumdur; bu durumda işlem yapılmaz. YAZ durumunda

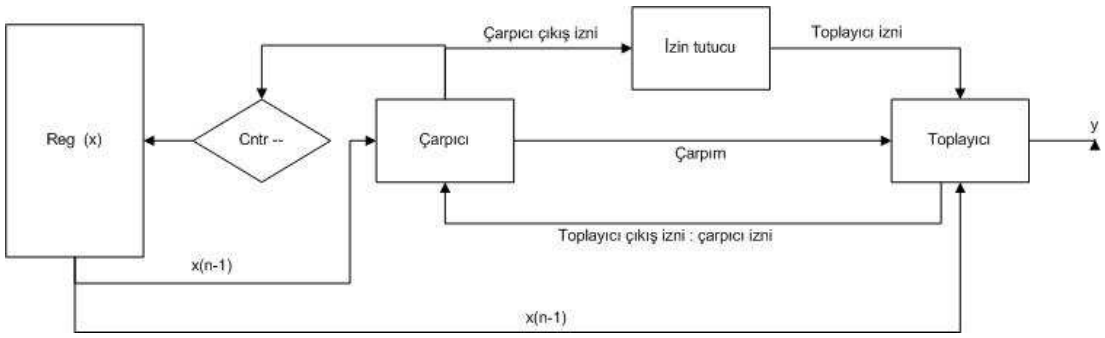
önceden belirtilen dizi uzunluğuna göre (N nokta) bellek oluşturulup, giriş işareti bu belleğe yazılarak gerekli dizi oluşturulur. Bunun alternatifi, belli uzunlukta bir ötelemeli yazmaç kullanarak, devamlı bu belleği güncel tutmaktır; ancak Goertzel algoritmasının kullanıldığı uygulamaları göz önünde bulundurulursa, giriş işaretinin frekans bileşenleri devamlı değişen bir işaret olmadığı anlaşılmaktadır. Örneğin telefonda kullanılan DTMF detektöründe, tuş işaretleri ardı ardına gelen N noktalı paketlerin analizi yapılarak tespit edilebilir. Ötelemeli yazmaç kullanmak bu nedenle gereksizdir. Onun yerine ardı ardına gelen N adet örnekten bir paket oluşturup, bu paketin analizi yapılmaktadır. İlk gelen N noktalı paket YAZ durumunda belleğin ilk halini oluşturur. Burada belleğe yazma hızı, örnekleme hızına uygun olmalıdır. KOŞ durumunda ise, YAZ durumunda oluşturulan N elemanlı dizi Goertzel Algoritmasına göre işlenirken, yeni gelen giriş değerleri de sıradaki N elemanlı paketi oluşturmak üzere belleğe yazılır.



Şekil 4.4: Goertzel FFT Devresi Durum Makinası

Şekil 4.4'te BAŞLA işareti dönüştürülecek yeni bir paketin geldiğini bildirmektedir. Sayaç ise belleğe alınması gereken N adet noktanın tamamlanıp tamamlanmadığının anlaşılması için çalıştırılmaktadır. Yeni bir pakete başlanmadığı sürece, yeteri kadar

nokta belleğe alınana kadar modül YAZ durumunda kalır, N nokta tamamlandığında devre şekil 4.5'te işlevi gösterilen KOŞ durumuna geçer.



Şekil 4.5: Goertzel Algoritması Akış Diyagramı

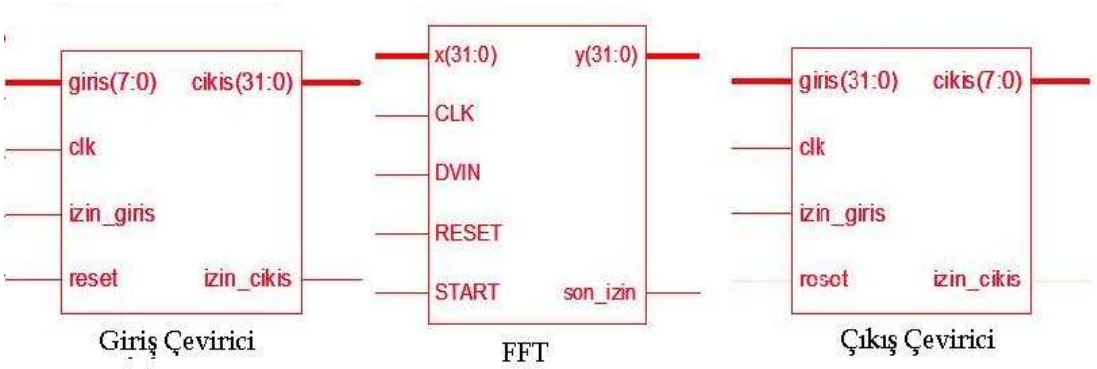
Toplayıcı ve çarpıcı modülleri kullanıldığı zaman toplayıcı ve çarpıcı izinleri ile giriş değerinin değişimi (giriş değerlerinin yazıldığı kütükten değerleri çekmek) şekil 4.5'te özetlenmiştir. N. giriş, giriş değerleri kütüğüne yazıldığında aynı anda çarpıcıya da bir giriş olarak yazılmaktadır. Diğer çarpan önceden belirlenmiş olan ağırlıktır. Bu sırada toplayıcının bir girişi 0 (çarpıcının çıkışı) diğer girişi x (n-1)'dir. Çarpıcı izni en başta (YAZ durumunda) 1 dir, dolayısıyla toplayıcı izni de 1 dir ve çıkış x (n-1)'dir. Çarpıcı izni tekrar 1 olduğunda, çarpım W. x(n-1) olur. Çarpıcı izni, sayıcıyı 1 azaltır ve giriş değerleri kütüğünden x(n-2) değerinin çekilmesini sağlamakta, aynı zamanda da toplayıcıya işlem yapması için izin vermektedir. Sayıcı değeri değiştiği için, çarpıcı izni alınmasıyla, yeni giriş değerinin giriş kütüğünden çekilmesi arasında 1 darbelik fark olacaktır; bu nedenle çarpıcı çıkış izni doğrudan toplayıcı iznine bağlanmamalıdır, 1 saat darbesi geciktirilmektedir. Başka bir deyişle toplayıcının bir girişi (çarpım) yenilendikten 1 saat darbesi sonra toplayıcının ikinci girişi (x) yenilenmekte ve toplayıcı izni alınmaktadır. Toplama işlemi tamamlanmadan veya çarpıcının eski işi bitmeden yeni çarpma işlemi başlatılmamaktadır. Dolayısıyla çarpıcı izninin verilmesi için toplayıcı çıkış izninin ve çarpıcı çıkış izninin verilmiş olması gerekmektedir.

Tüm bu toplama ve çarpma işlemleri tek duyarlı kayan nokta aritmetiğine uygun olarak hazırlanmış toplama ve çarpma bloklarıyla gerçekleştirilmiştir. Toplama ve çarpma işlemleri bölüm 4.1.2'te belirtildiği gibi birden fazla aşama gerektirmektedir, bu nedenle işlemin bitişi bir izin çıkışı ile işaretlenmektedir. Sözü edilen izinler, toplama veya çarpma işleminin sonuçlandığını belirten işaretlerdir.

Sadece FFT devresinin benzetimi, daha önce MATLAB'da alınan sonuçlarla uyum gösterdiğinden FFT modülü diğer modüller ile birleştirilerek geliştirme kiti üzerinde gerçekleme yapılmıştır.

Analog sayısal dönüştürücüden alınan 8 bit uzunluğundaki sabit noktalı değer, FFT bloğunda kullanılacak olan tek duyarlı kayan noktalı gösterime dönüştürülmüştür. Ağırlıklar ise önceden hesaplanabilir olduğundan, FFT devresi içinde bir yazmaçta tutulmaktadır. Çıkış da benzer şekilde tek duyarlı kayan nokta formatından, sekiz bit uzunluklu sabit noktalı hale getirilmiştir.

Benzetim sırasında, dosyaya yazılıp okunabilen değerlerin gerçek ortamda gözlenebilmesi için geliştirilen yol, sayısal analog çevirici girişini FFT modülünün çıkışı belli bir değerin üstündeyse 11111111'e, altındaysa 00000000'a çekmek olmuştur. Sayısal analog çevirici çıkışı ise osiloskoba bağlanarak eğer giriş frekansı hedef frekans ile aynı ise osiloskopta Vdd, değilse toprak seviyesinde işaret gözlenmiştir. Çıkışın kit üzerindeki LED'ler yerineharici bir osiloskopa bağlanmasının nedeni, öncelikle düşünülenin giriş frekansı hedef frekansa yaklaştıkça çıkış değerinde bir artış olup olmadığını tespit etmekti; ancak çıkış aralığının sayısal analog çeviricinin çıkış aralığı yanında çok geniş olması nedeniyle, ufak değişimler osiloskop girişinin besleme veya toprağa sürülmesine neden olmuştur. Bu nedenle sistemde ufak bir değişiklikle hedef frekans ile giriş frekansının örtüşüp örtüşmediği yine osiloskoptan gözlenmiştir.



Şekil 4.6:Goertzel Algoritması İçin Hazırlanan Modüller

Şematik gösterilimi Şekil 4.6'da verilen modüllerin giriş çıkış tanımlamaları ve bağlantıları tablo 4.4'te verilmiştir.

Tablo 4.4: Goertzel Algoritmesı Modülleri Pin Tanımları

İşaret	**Tanım ve Bağlantı**
Giris/Giriş Çevirici	Analog sayısal çeviriciden alınan 8 bit uzunluklu işaret
İzin_giris/Giriş Çevirici	Her bir yeni değeri işaretleyen aktif yüksek giriş.
cikis/Giriş Çevirici	Tek duyarlı kayan nokta formatında zaman domeni işareti (x/FFT)
İzin_cikis/Giriş Çevirici	Her bir yeni kayan nokta değeri işaretleyen giriş çevirici çıkışı
START/FFT	Yeni bir paketle FFT işlemine başlanmasını sağlayan FFT modülü girişi
DVIN/FFT	FFT bloğu için her bir yeni değeri işaretleyen aktif yüksek giriş, (izin_cikis/ giriş çevirici)
x/FFT	FFT modülü girişi
y/FFT	FFT modülü çıkışı
Son_izin/FFT	FFT modülü çıkış izni (izin_giris/çıkış çevirici)
Giris/Çıkış Çevirici	FFT modülünden alınan kayan nokta formatındaki işaret (y/FFT)
İzin_giris/Çıkış Çevirici	Her bir yeni değeri işaretleyen aktif yüksek giriş.
cikis/Çıkış Çevirici	Sayısal analog çeviriciye aktarılacak olan 8 bit uzunluklu işaret
İzin_cikis/Çıkış Çevirici	Her bir yeni çıkışı işaretleyen işaret

Tablo 4.5: Goertzel FFT için FPGA ‘da Kullanılan Alan

Lojik Birimler	**Kullanılan**	**Mevcut**	**Yüzde Kullanılan**
Dilimler	28	4656	%0
Dilimlerdeki Flip-Floplar	32	9312	%0
Dört Girişli LUT’lar	45	9312	%0
IOB’ler	12	232	%5
Saat	2	24	%8

Devre tüm modülleri ile birlikte gerçeklendiğinde FPGA içinde kullandığı dilim ve elemanların dağılımı tablo 4.5’ te gösterilmiştir. Kayan nokta aritmetiği kullanıldığı halde, algoritmanın işlem yükünün çok az olması, FPGA üzerinde devrenin çok küçük bir yer kaplamasını sağlamıştır.

4.2. Rader Algoritması

Rader Algoritması, hızlı fourier dönüşümünü dairesel konvolüsyonla ifade edilmesinden yola çıkar. Rader algoritmasının asal sayı özelliklerini kullanması bu algoritmayı konvolüsyon ile ayrık fourier dönüşümü hesaplama yöntemlerinden ayırır.

N noktada uygulanan algoritma, standart ayrık fourier dönüşümüne göre 2^n işlem kazanç sağlar. Algoritma N nokta DFT işlemini, katsayı indekslerinin yeniden düzenlenmesi ve dairesel konvolüsyon kullanılarak, birkaç toplam işlemi ile birlikte (N-1) nokta DFT işlemine dönüştürür [9].

Toplamanın birleşme özelliğinden yararlanılarak, indekslenmiş terimlerin sırası değiştirilebilir. Rader algoritmasında kullanılan çok boyutlu bir indeksleme değil, asal gruplar içinde çarpma işlemi ile oluşturulan tek boyutlu bir indekslemedir:

$$k = (r^m) \bmod(N) \tag{4.9a}$$

Burada, N asal bir sayı olmak üzere, *r* N asal sayısının grubunda bir sayıdır, öyle ki, gruptaki her bir eleman bu *r* sayısının üssü şeklinde ifade edilebilir. İleride gerçeklemede kullanılacak olan N = 17 ve r = 3 için bu durum sınanacak olursa:

$3^0 \bmod(17) = 1$ (4.9b)

$3^1 \bmod(17) = 3$ (4.9c)

$3^2 \bmod(17) = 9$ (4.9d)

$3^3 \bmod(17) = 10$ (4.9e)

$3^4 \bmod(17) = 13$ (4.9f)

$3^5 \bmod(17) = 5$ (4.9g)

$3^6 \bmod(17) = 15$ (4.9h)

$3^7 \bmod(17) = 11$ (4.9i)

$3^8 \bmod(17) = 16$ (4.9j)

$3^9 \bmod(17) = 14$ (4.9k)

$3^{10} \bmod(17) = 8$ (4.9l)

$3^{11} \bmod(17) = 7$ (4.9m)

$3^{12} \bmod(17) = 4$ (4.9n)

$3^{13} \bmod(17) = 12$ (4.9o)

$3^{14} \bmod(17) = 2$ (4.9p)

$3^{15} \bmod(17) = 6$ (4.9r)

$3^{16} \bmod(17) = 1$ (4.9s)

Görüldüğü gibi N=17 kümesinin her elemanı r=3'ün kuvveti şeklinde yazılabilmektedir.

Asal sayıların bu özelliği, DFT işlemini konvolüsyona çevirmek için kullanılır; (4.1) denkleminde,sıfırdan farklı k yerine eşitlik (4.9)'da verilen k değeri yerleştirildiğinde (4.10) ile verilen eşitliğe ulaşılmaktadır.

$$X((r^{p}) \bmod(N)) = \sum_{m=0}^{N-2} x((r^{-m}) \bmod(N)) W^{rp^{-m}} + x(0)$$

$$= x(0) + x((r^{-l}) \bmod(N)) * W^{r^{l}}, 1 = [0,1,2 \ldots \text{N - 2}] \qquad (4.10)$$

N=5, r=2 ve r^{-1}=3 için örnek bir matris oluşturulmuştur.

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 4 & 1 & 3 \\ 0 & 3 & 1 & 4 & 2 \\ 0 & 4 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \end{bmatrix} \qquad (4.11)$$

Doğrudan DFT gerçeklenebilecek şekilde oluşturulmuş (4.11) matrisinde, satırların yerleri değiştirilmiştir.

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(4) \\ X(3) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 4 & 2 \\ 0 & 2 & 1 & 3 & 4 \\ 0 & 4 & 2 & 1 & 3 \\ 0 & 3 & 4 & 2 & 1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(3) \\ x(4) \\ x(2) \end{bmatrix} \qquad (4.12)$$

Bu yeni düzenlemeyle elde edilen (4.12) ifadesinde, W matrisinde her bir satırdaki sıranın dairesel olarak değişmesi sağlanmıştır. Bu da gerçeklemede kolaylık sağlayacak bir etkendir.

Öte yandan X(0) ise işaretin DC değeri olduğu için bazı uygulamalarda ilgilenilmiyor, veya değeri önceden biliniyor olabilir. Frekans spektrumu çıkarılırken, işaretin DC değeri için N noktanın ortalaması alınması yeterlidir. Bazı uygulamalarda sadece toplamının alınması ile yetinilirken, ayrıca bölme işlemi yapılmamış olur.

4.2.1. MATLAB Benzetimi ve FPGA Üzerinde Gerçekleme

Şekil 4.7'da gösterildiği gibi, dört durum tasarlanmıştır. *Bekle* durumunda *Başla* işareti gelene kadar kalınır. *Başla* işareti geldiğinde, FFT işleminin yapıldığı durumlara geçilir. FFT işlemi iki ayrı durumda gerçekleştirilmiştir.

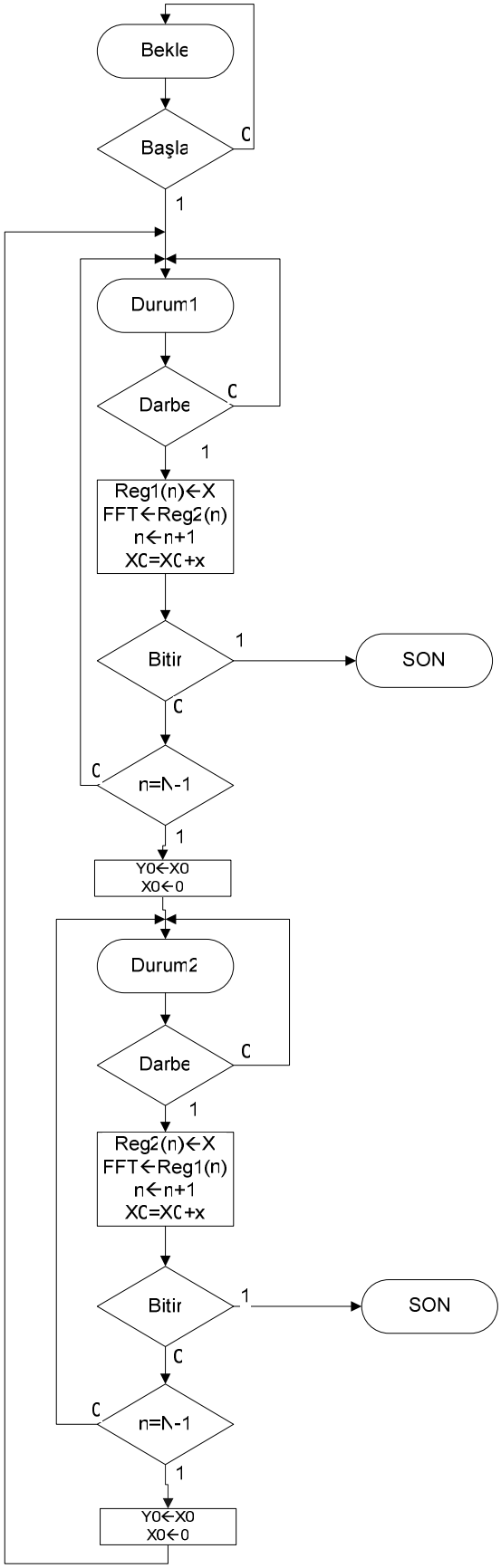
1. İkinci yazmaç dizisindeki verinin dönüşüm işlemi ve gelen yeni verinin birinci belleğe yazılması durumu (Durum 1).
2. Birinci yazmaç dizisindeki verinin dönüşüm işlemi ve gelen yeni verinin ikinci belleğe yazılması durumu (Durum 2).

Durum 1 ve Durum 2 arasındaki geçiş, bir paketteki nokta sayısına ulaşılıp ulaşılmadığı kontrol edilerek sağlanır.

N noktadan oluşan bir paketin ilk elemanı matris çarpımı işleminde kullanılmadığı için belleğe yazılmaz, ancak işaretin DC değerini tutan değişkene eklenir.

Bitir işareti gelene kadar bu iki durum için tanımlanan işlemler yapılmaya devam eder. Bitir işareti geldiğindeyse, belleklere veri yazımı sonlanır ve işlem durdurulur.

Burada, girişteki işaretin yeni bir değer olup olmadığı “darbe” işareti ile kontrol edilir. Darbe olarak tanımlanan aslında örnekleme frekansında bir saat darbesi gibi düşünülebilir, sistemin çalışma saati ise örnekleme frekansından daha yüksek olmalıdır.



Şekil 4.7: Rader Algoritması Akış Diyagramı

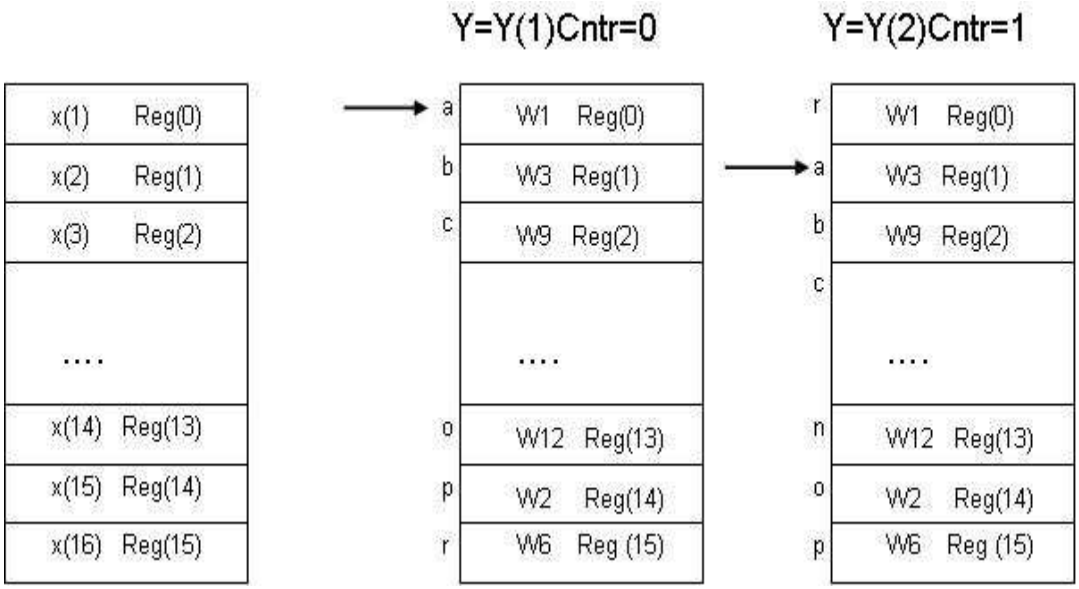
4.2.2. FFT İşleminin Gerçeklenmesi

Yukarıdaki matris çarpımında görüldüğü gibi, x değerleri sıralı olmasa da yerleri sabittir. Öte yandan katsayıların ise yerleri dairesel olarak değişmektedir. Sabit olan bu katsayılar bir bellek dizisinde tutulmakta, ve yerleri değiştirilmemektedir. Ancak katsayılar belleğe yerleştirilirken, matrisin ilk satırındaki sıra kullanılmıştır. N= 5 için [W1 W3 W4 W2], N= 17 için [W1 W3 W9 W10 W13 W5 W15 W11 W16 W14 W8 W7 W4 W12 W2 W6] sırası kullanılmıştır. FFT işlemi sırasında ise, katsayı yazmacından çekilen değerler bir işaretçinin her bir ayrı Y değeri için bir değer yukarıdan başlaması ile matris çarpımında verilen dairesellik sağlanmıştır.

$$y\_reel = REG2[0] * coeffR[r] + REG2[2] * coeffR[sira] + REG2[8] * coeffR[a] + REG2[9] * coeffR[b]\ldots \quad (4.13)$$

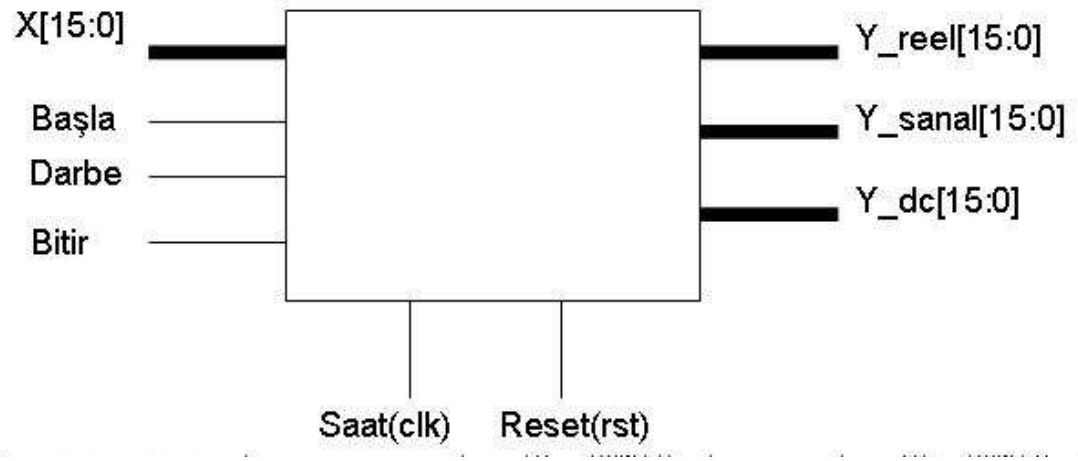
Yukarıda a, b, r ve sira ile tanımlanan değişkenler, katsayı matrisindeki yerleri göstermektedir. r= 0 olduğunda, sira=1 a=2 b=3 olacaktır. Bu durumda matrisin ilk satırı ile giriş işareti çarpılarak y_reel(1) değeri bulunuyormuş gibi düşünülebilir. Bir sonraki işlemde, r, sira, a ve b değişkenleri bir arttırılacak, çarpım ve toplan işlemleri tekrar yapılacak ve y_reel(2) değeri bulunacaktır. Frekans işaretinin reel kısmını bulmak için izlenen tüm yollan sanal kısmının bulunması için de geçerlidir, yalnızca sanal kısmın katsayıları için ayrı bir yazmaç kullanılmıştır.

Değişkenler sabit nokta formatında, çıkış değerleri 16 bit olacak şekilde ayarlanmıştır. Taşma olmaması için 16 bit olarak verilen katsayılar ve giriş değerleri -32 +31 aralığında, 16 bite genişletilmiş sayılardır. Hassaslığın bu kadar düşük olması sonuçları da etkilemektedir.



Şekil 4.8: Dairesel Katsayı Yazmacı

Şekil 4.8'de gösterildiği gibi, yazmaçların içi hiç değiştirilmemiş, toplanan terimler Reg_X(i)*Reg_W(j) şeklinde verilmiştir. Burada i her zaman bir sabit ile tanımlanmışken, j ise i değerine göre a, b, c.. değişkenleri ile tanımlanmıştır. Bu değişkenler, katsayının yazmaçta bulunduğu yeri gösteren indislerdir ve mod (N-1) 'e göre her bir işlemde dairesel olarak artmaktadır. N= 17 seçilmesi, dairesel artışın elde edilmesini kolaylaştırmıştır. Bir kontrol mekanizması konmadan 4 bitlik tanımlanan değişkenler, 16 adet katsayıyı tarayacaktır.



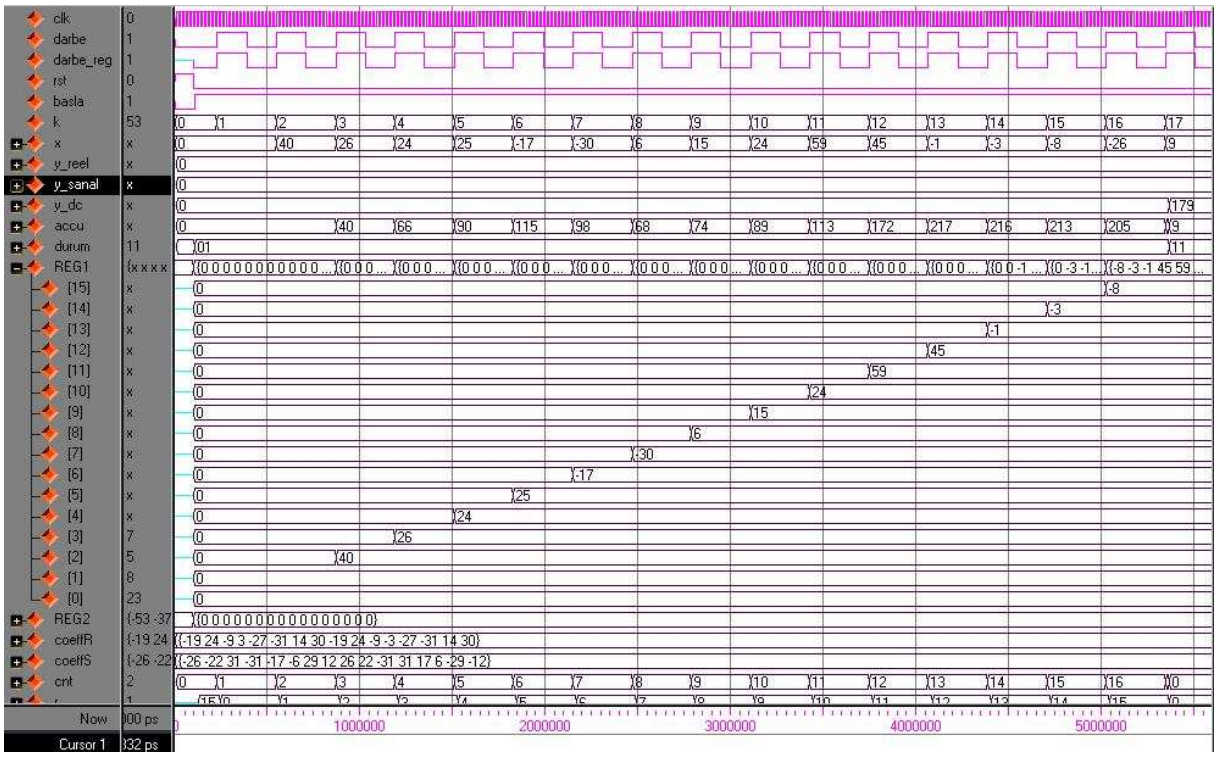
Şekil 4.9: Rader FFT Devresi Şematiği

Sentezlenen FFT devresinin şematiği şekil 4.9 ile, pin açıklamaları ise tablo 4.6 ile verilmiştir.

Tablo 4.6: Rader FFT Devresi Pin Açıklamaları

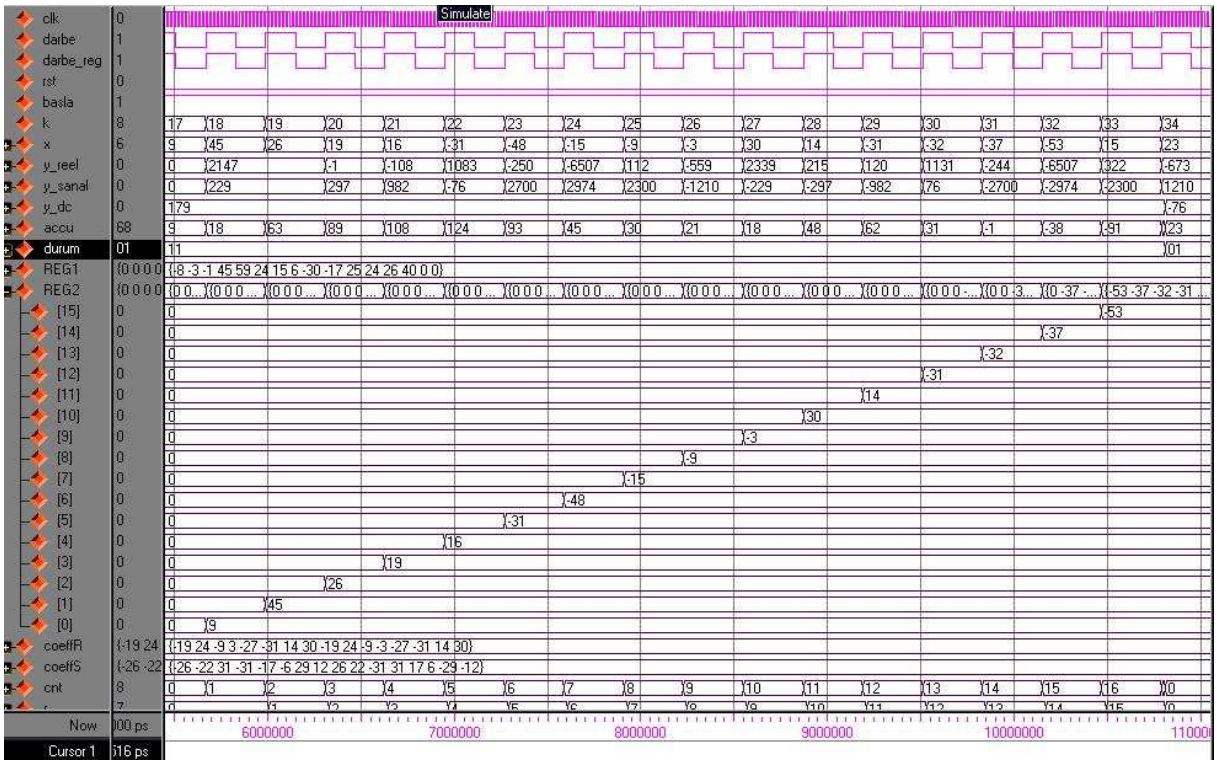
İşaret	**Yön**	**Tanım**
X	Giriş	İşlenecek olan giriş işareti
Başla	Giriş	Girişte işlenecek olan ilk verinin görülmesi ile, sistemin çalışabileceğini gösteren işaret
Darbe	Giriş	Giriş işaretinin örnekleme frekansına bağlı, girişteki işlenecek olan verinin yeni bir değer olduğunu gösteren darbe işareti.
Bitir	Giriş	İşlemin sonlandırılmasını sağlayan işaret
Y_reel	Çıkış	FFT sonucunun reel kısmı
Y_sanal	Çıkış	FFT sonucunun sanal kısmı
Y_dc	Çıkış	İşlenen paketin DC değeri
clk	Giriş	Saat: Yükselen kenar tetiklemeli devre için saat işareti
rst	Giriş	Reset: Asenkron reset, aktif 1

Yazılan devre, MATLAB ile oluşturulan giriş işaretleri ile ModelSim kullanılarak test edilmiştir. Davranışsal benzetim ile durum geçişleri, her bir saat darbesinde yapılan iş, çıkış ve devre içindeki yazmaçlardaki değişim gözlenmiştir.



Şekil 4.10: Durum1 ModelSim Grafiği

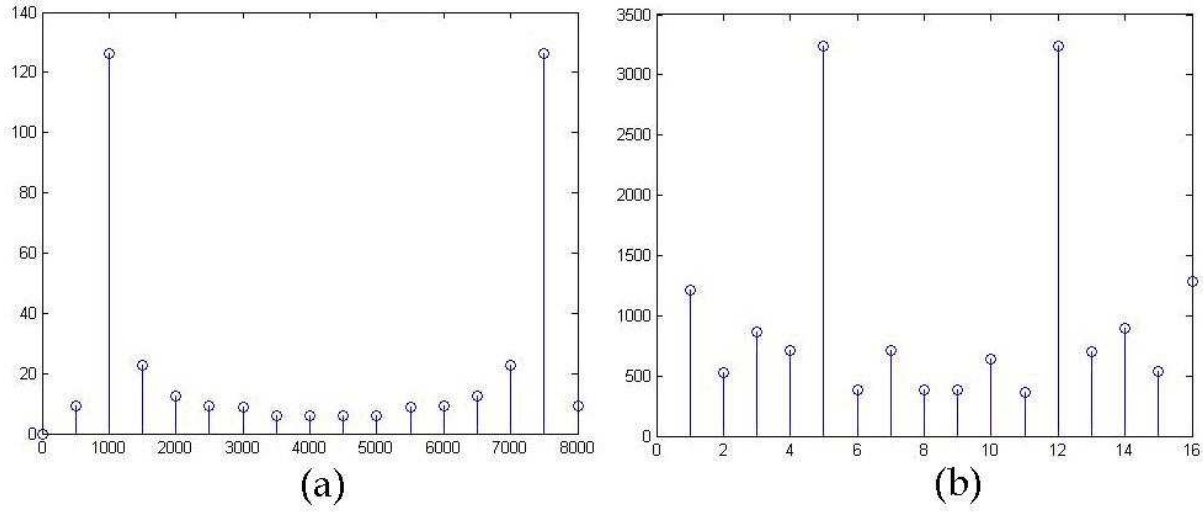
Şekil 4.10 ve şekil 4.11'de sırasıyla Durum1 ve Durum2 için meydana gelen değişimler verilmiştir. Şekillerdeki darbe_reg işareti, darbe işaretinin sıfırdan bire geçtiği yerleri bulmak amacıyla devre içinde oluşturuluş bir işarettir. k ise test dosyasında tanımlanmış, dosyadan okunan giriş değerlerinin sırasını gösterir. Durumların kodlanmasında Gray kodu kullanılmıştır.



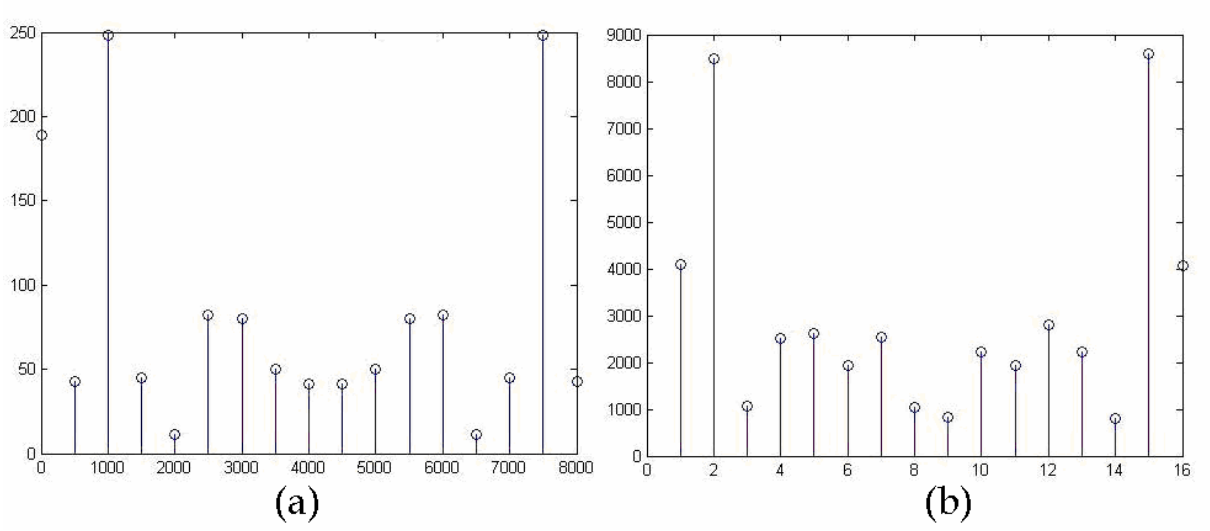
Şekil 4.11: Durum2 ModelSim Grafiği

Simülasyonlarda 1kHz'lik temiz sinüs dalgası ve içinde 200 Hz, 1kHz ve 2,5kHz'lik bileşenler bulunan 8kHz ile örneklenmiş başka bir dalga kullanılmıştır.

Simülasyon sonuçları çizdirilirken, çıkış sırasının frekans skalasındaki sıra ile aynı olmadığı göz önüne alınmış ve aşağıdaki grafikler çizilirken, sıra değiştirilmiştir.



Şekil 4.12: (a) MATLAB ile Elde Edilen 1kHz frekans spektrumu (b) ModelSim Simülasyon Sonucu 1kHz Frekans Spektrumu



Şekil 4.13: (a) MatLab ile elde edilen 200Hz, 1kHz ve 2,5kHz frekans spektrumu (b) ModelSim Simülasyon Sonucu elde edilen 200Hz, 1kHz ve 2,5kHz frekans spektrumu

4.3. Bluestein Chirp- z Algoritması

N noktalı bir DFT işlemi, Fourier dönüşümü yapılacak olan x(n) serisinin, birim çember üzerinde, eşit aralıklarla sıralanmış N noktada z dönüşümünün hesaplanmasına denktir.

Denklem (3.8)'deki genel DFT ifadesi, *nk* yerine (4.14) ifadesi konularak tekrar yazılırsa, denklem (4.15) elde edilir.

$$\frac{-(k-n)^2}{2}+\frac{n^2}{2}+\frac{k^2}{2} \tag{4.14}$$

$$X(k)=e^{\frac{-j\pi}{N}k^2}\sum_{n=0}^{N-1}\left(x(n)e^{\frac{-j\pi n^2}{N}}\left(e^{\frac{j\pi}{N}(k-n)^2}\right)\right) \tag{4.15}$$

Elde edilen ifade, sırasıyla (4.16) ve (4.17) olarak tanımlanan N-1 noktadan oluşan a(n) ve b(n) dizilerinin konvolüsyonunun bir sabit ile çarpılmışıdır. Bu sabit, $e^{\frac{-j\pi}{N}k^2}$, b(k)'ya eşittir.

$$a(n)=x(n)\left(e^{\frac{-j\pi}{N}n^2}\right) \tag{4.16}$$

$$b(n)=\left(e^{\frac{j\pi}{N}n^2}\right) \tag{4.17}$$

Son halde (4.18) denklemi yazılabilir.

$$X(k) = b(k)\sum_{n=0}^{N-1}\left(a(n)b(k-n)\right) \tag{4.18}$$

a(n) ve b(n) ifadeleri, (4.19) ve (4.20)'deki gibi açıldığında, (4.21) ve (4.22)'den yararlanılarak a(n) ifadesi (4.23) olarak değiştirilebilir.

$$b(n) = \cos(\frac{\pi n^2}{N}) + j.\sin(\frac{\pi n^2}{N}) \tag{4.19}$$

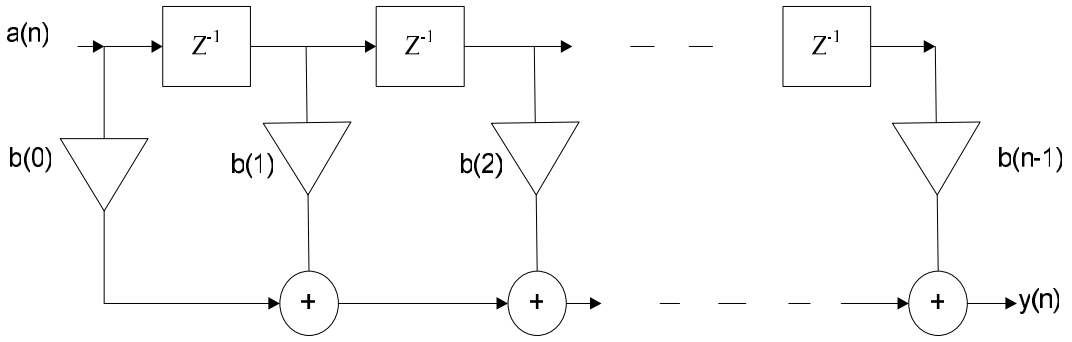
$$a(n) = x(n).(\cos(\frac{\pi n^2}{N}) - j.\sin(\frac{\pi n^2}{N})) \tag{4.20}$$

$$b_{reel}(n) = \cos(\frac{\pi n^2}{N}) \tag{4.21}$$

$$b_{sanal}(n) = \sin(\frac{\pi n^2}{N}) \tag{4.22}$$

$$a(n) = b_{reel}(n).x(n) - b_{sanal}(n).x(n) \tag{4.23}$$

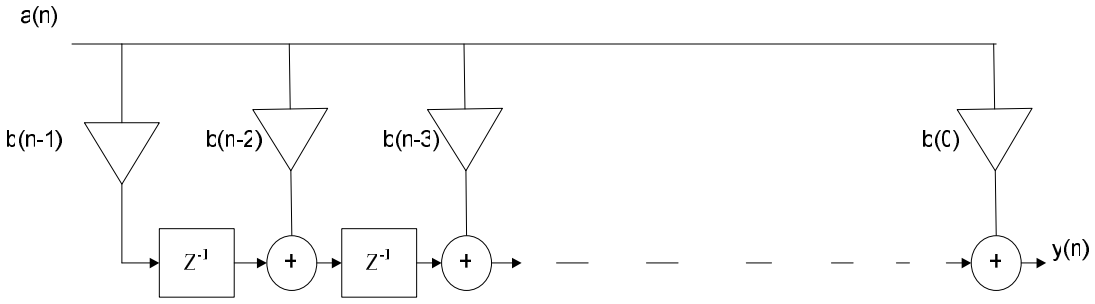
a(n) ve *b(n)* fonksiyonlarının konvolüsyonu, FIR filtre kullanılarak gerçeklenebilmektedir. FIR filtre seçiminde ise, doğrudan (direct) FIR filtre kullanılabileceği gibi, çapraz (transposed) FIR filtre de kullanılabilir. Şekil 4.14' te gösterilen doğrudan FIR gerçeklenirken N satırlık ötelemeli yazmaç kullanılması gerekir, ancak çapraz şekil 4.15 ile gösterilen FIR topolojisi için tek satırlık N adet kütük kullanmak yeterlidir.



Şekil 4.14: Doğrudan FIR

Zaman domeninde bulunan N örnek sayısı ve karşılık gelen K frekans sayısı birbirine eşit seçilmek zorunda değildir, ancak işlem kolaylığı açısından bu ikisi birbirine eşit seçildiğinde, a(n) ve b(n) fonksiyonlarının konvolüsyon ifadesi denklem (4.24)'teki gibi yazılırsa şekil 4.15 ile verilen topolojiye geçilebilir:

$$y(n) = a(n) * b(n) = a(n).b(0) + a(n-1).b(1) + a(n-2).b(2).....a(0).b(n)$$
$$= ((a(n).b(n-1).z^{-1} + a(n).b(n-2)).z^{-1} + ...)z^{-1} + a(n).b(0) \quad (4.24)$$

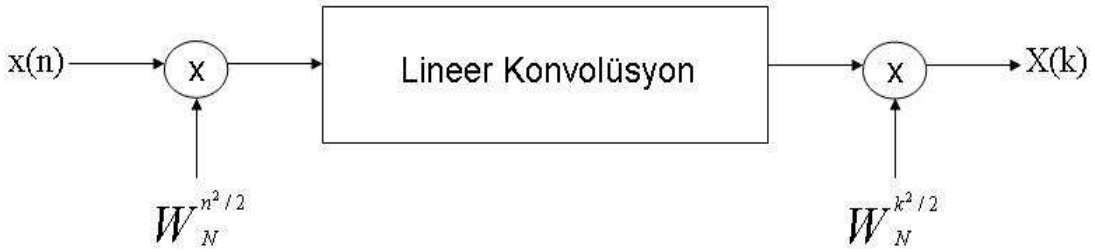


Şekil 4.15: Çapraz FIR

Algoritma üç aşamada gerçeklenebilir:

1. a(n) 'in oluşturulması
2. a(n) ile b(n)'in konvolüsyonu
3. Konvolüsyonun b(k) ile çarpılarak X(k)'nın oluşturulması

Yukarıda maddelenen işlemler, şekil 4.16' daki blok diyagram ile gerçeklenebilir.



Şekil 4.16: Bluestein Chirp-z DFT

Toplamda N uzunluklu bir konvolüsyon işlemi ve 2N adet çarpma gerektiren bu algoritmanın Rader algoritmasına çok benzemektedir. Farkı, Rader algoritmasında olduğu gibi N'in asal olma zorunluluğunun olmamasıdır.

Gerçeklemede ise, b(n) ifadesi bazı n değerleri için işlem yükü getirmeyen 1 ve 0 değerlerini alabildiği için genel hali verilen süzgeç yapıları küçültülebilir. N değerinin sabit olduğu durumlarda, her bir b(n) çarpanı için farklı toplama ve ötelemeli yazmaç yapıları kullanılarak, devrenin kapladığı alan küçültülebilir [9].

4.4. Cooley - Tukey Algoritması

1965 yılında, Cooley ve Tukey, Gauss'un ortya attığı bir algoritmayı FFT hesaplamak için yeniden ele alarak yayınladı [14]. Cooley ve Tukey'in önerdiği algoritma, yalnızca $O(N.\log_2 N)$ işlem gerektirdiğinden, nokta sayısı N büyüdükçe diğer FFT algoritmalarına göre işlem yükünü oldukça azaltmaktadır. Cooley-Tukey algoritması şekil 3.1 ile gösterilen katsayıların periyodikliğine ve simetrisine dayanır. Katsayıların periyodikliğinden denklem (4.25), katsayıların simetrisinden ise denklem (4.26) elde edilir [15].

$$W^k = W^{k+N} \tag{4.25}$$

$$W^k = -W^{k+N/2} \tag{4.26}$$

N nokta DFT işlemi Cooley-Tukey algoritması kullanılarak N, N_1 ve N_2 noktalı DFT işlemlerine bölünebilmektedir [9]. N_1 ve N_2 de aynı şekilde çarpanlarına ayrılarak 2 noktalı DFT işlemine kadar inilebilir. N_1 ve N_2'nin seçiminde teorik olarak bir kısıtlama olmasa da, gerçeklemede kolaylık sağlaması açsından, N ikinin kuvveti olarak seçilip, N/2 noktalı iki DFT işlemine açılacaktır. Bu şekilde açılarak, en son 2 nokta DFT işlemi elde edilecektir. Bu tip bir açılıma taban-2 Cooley-Tukey FFT denir [9]. Optimizasyon için farklı yöntemlerle birleştirilip, katsayıların farklı sıralanmasına müsaade eden Cooley-Tukey algoritmasının bu çalışmada frekansta azaltmalı taban-2 formu kullanılmıştır.

4.4.1. Frekansta Azaltmalı Cooley-Tukey FFT Algoritması

N terimden oluşan x(n) dizisi, terim sayısı eşit iki alt diziye bölünüp, (4.1) ile verilen DFT işlemi yeniden yazılırsa denklem (4.27) elde edilir.

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n) W_N^{nk} + \sum_{n=N/2}^{N-1} x(n) W_N^{nk} \tag{4.27}$$

Denklem (4.27)'de ikinci toplam işleminde n yerine n+N/2 yazılırsa denklem (4.28) halini alır.

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n) W_N^{nk} + W^{kN/2} \sum_{n=0}^{(N/2)-1} x\left(n + \frac{N}{2}\right) W_N^{nk} \tag{4.28}$$

Burada n'ye bağlı olmadığı için toplam dışına çıkarılan terim (4.29) eşitliği ile hesaplanabilir.

$$W^{kN/2} = e^{-jk\pi} = \left(e^{-j\pi}\right)^{k} = (\cos\pi - j\sin\pi)^{k} = (-1)^{k} \tag{4.29}$$

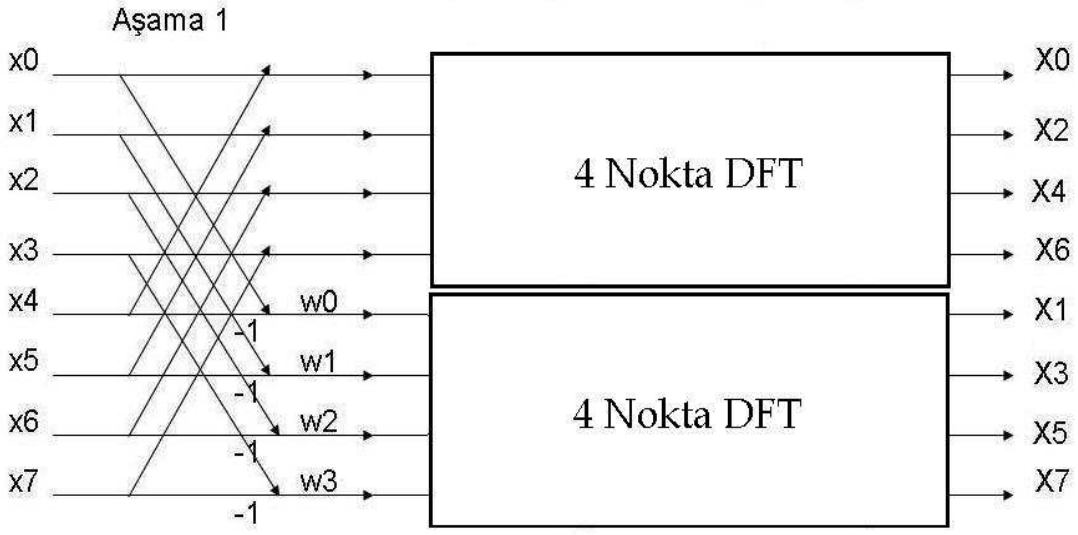
Bulunan katsayı k'nın tek değerleri için -1, çift değerleri için 1 olacaktır. (4.29) ifadesi k'nın çift değerleri için k=2k ve tek değerleri için k=2k+1 olarak ikiye ayrılıp yazıldığında sırasıyla (4.30) ve (4.31) elde edilir.

$$X(2k) = \sum_{n=0}^{(N/2)-1}\left[x(n) + x\left(n+\frac{N}{2}\right)\right]W^{2nk}, k = 0,1,...,\left(\frac{N}{2}\right)-1 \tag{4.30}$$

$$X(2k+1) = \sum_{n=0}^{(N/2)-1}\left[x(n) - x\left(n+\frac{N}{2}\right)\right]W^{n}W^{2nk}, k = 0,1,...,\left(\frac{N}{2}\right)-1 \tag{4.31}$$

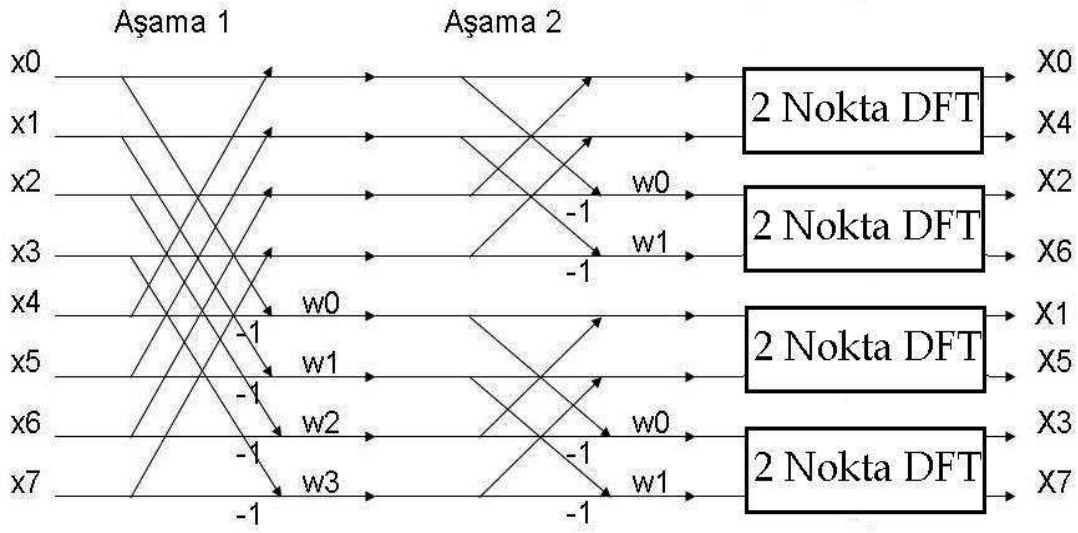
W, nokta sayısı N'nin bir fonksiyonudur, W_N şeklinde gösterilebilir. Katsayıların birim çember üzerindeki hareketi nedeniyle, W^2 de N/2'nin bir fonksiyonu olarak, $W_{N/2}$ biçiminde yazılabilir. $N=2^r$ noktalı DFT işlemi için Cooley Tukey algoritması kullanılarak *r-1* aşamada iki noktalı DFT işlemine indirilebilir. 2 nokta DFT işleminin hesaplanması da bir aşama olarak hesaba katıldığında, toplam *r* aşamada N noktalı DFT işlemi tamamlanmış olur.

8 noktalı DFT işleminin Cooley-Tukey algoritması ile gerçeklenmesinin ilk aşaması Şekil 4.17 ile gösterilmiştir.



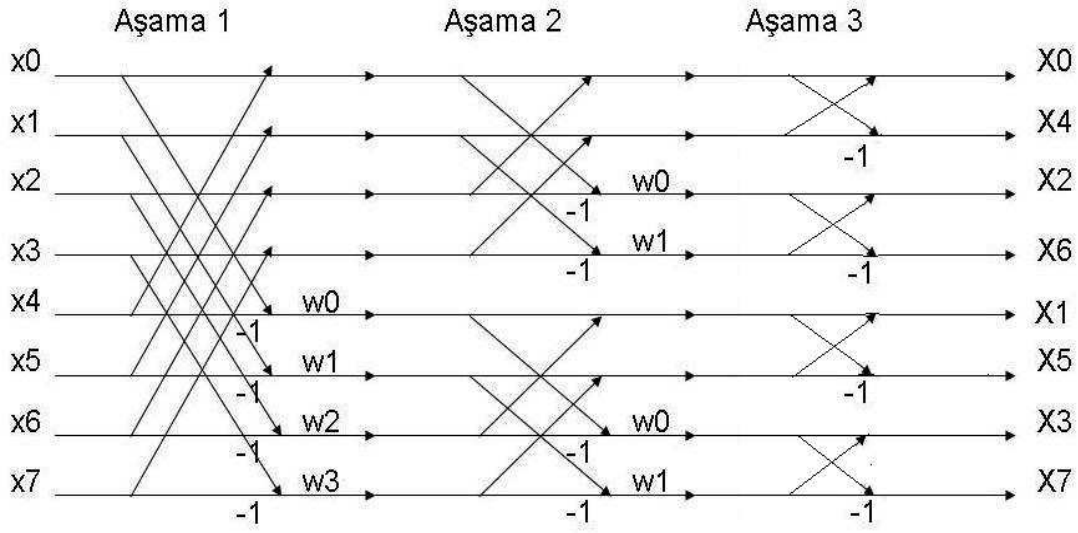
Şekil 4.17: 8 Nokta DFT İçin İlk Aşama

Elde edilen 4 noktalı DFT işlemleri de ikişer adet 2 noktalı DFT işlemine şekil 4.18 ile gösterildiği gibi açılmaktadır.



Şekil 4.18: 8 Nokta DFT İçin İkinci Aşama

İki noktalı DFT işlemine kadar indirgenen N noktalı DFT işleminde son aşama, katsayı kullanılmaksızın gerçekleştirilebilir. 8 nokta DFT hesaplanmasının üç aşaması toplu halde şekil 4.19 gösterilmiştir.

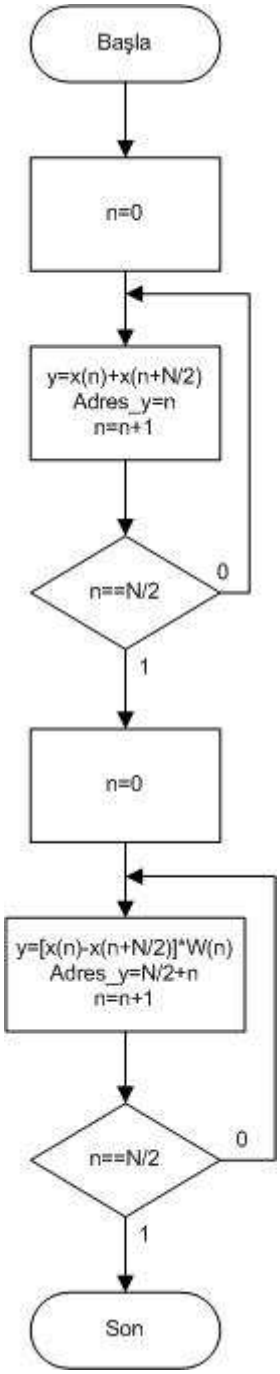


Şekil 4.19: 8 Nokta DFT

Son aşamada bulunan frekans değerleri, sıralı değildir. Elde edilen frekans değerleri ile frekans spektrumu oluşturulmak istendiğinde, frekans değerlerinin yerleri değiştirilmelidir.

4.4.2. MATLAB Benzetimi

Her bir aşamada ele alınan N nokta DFT işlemi için, N/2 tane karmaşık katsayı gerekmektedir. Bir aşamadan bir sonrakine geçebilmek için izlenmesi gereken yol, şekil 4.20'deki akış diyagramı ile verilmiştir.

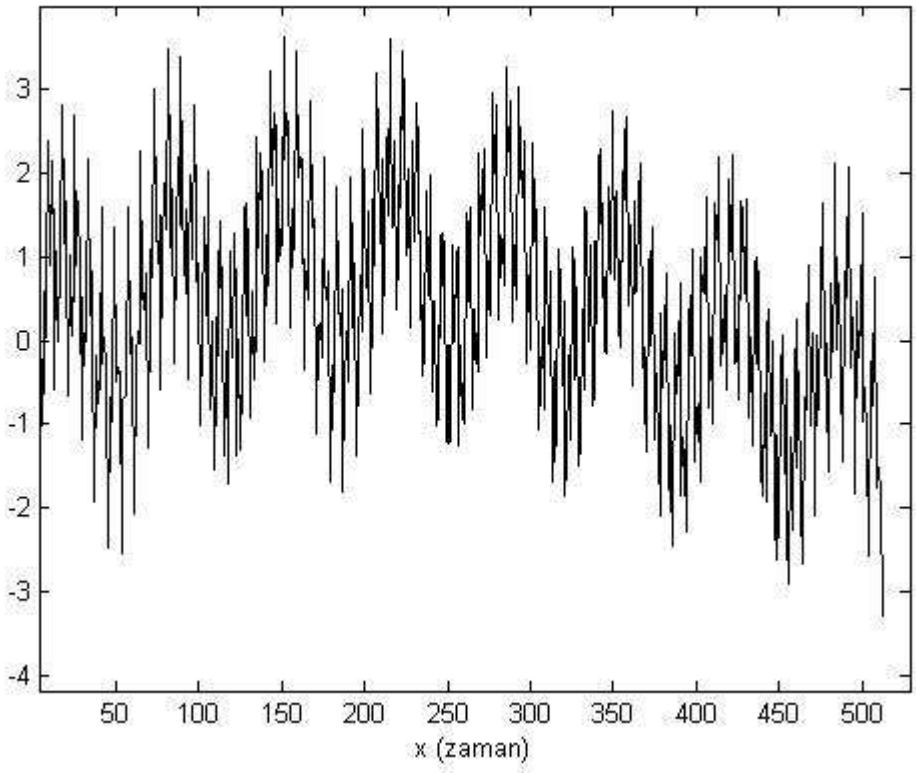


Şekil 4.20: Tek Aşama İçin Cooley-Tukey Akış Diyagramı

Bu diyagram uyarınca, 512 noktalı DFT işlemi, 8 aşamada MATLAB ile hesaplanmıştır. Hesaplamada kullanılan katsayılar ve sinüs dalgası MATLAB ile

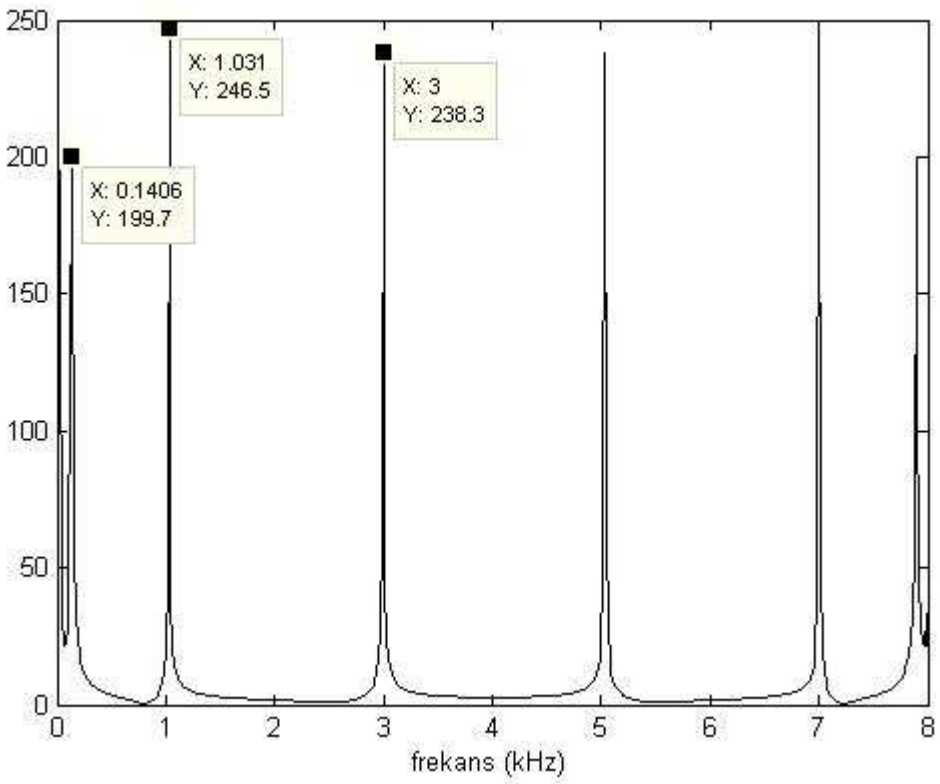
hesaplanabilecek maksimum hassaslık derecesindedir. Aynı giriş işaretinin frekans spektrumu MATLAB'ın sunduğu hazır komutlarla hesaplandığında, sadece Cooley-Tukey algoritması kullanılarak oluşturulan frekans spektrumundan daha temiz bir spektrum elde edilmektedir. Bunun nedeni MATLAB'ın Cooley-Tukey algoritması ile diğer algoritmaların birleşimini kullanarak optimizasyon sağlayan C tabanlı bir FFT bankasından yararlanmasıdır [16]. Kullandığı birleşim 512 noktalı DFT işlemini Coley-Tukey algoritması ile 32 noktalı DFT işlemlerine parçalayıp, bu 32 noktalı DFT işlemini ise 32 ve daha az nokta için daha iyi sonuç üretebilen Rader FFT algoritması ile hesaplamaktır.

512 nokta Cooley Tukey FFT algoritması gerçeklenecektir. Bunun için sekiz aşama gereklidir. Öncelikle aralarında asal üç frekans bileşeninden oluşan 8kHz ile örneklenmiş sinüs işareti şekil 4.21 ile gösterilmiştir.



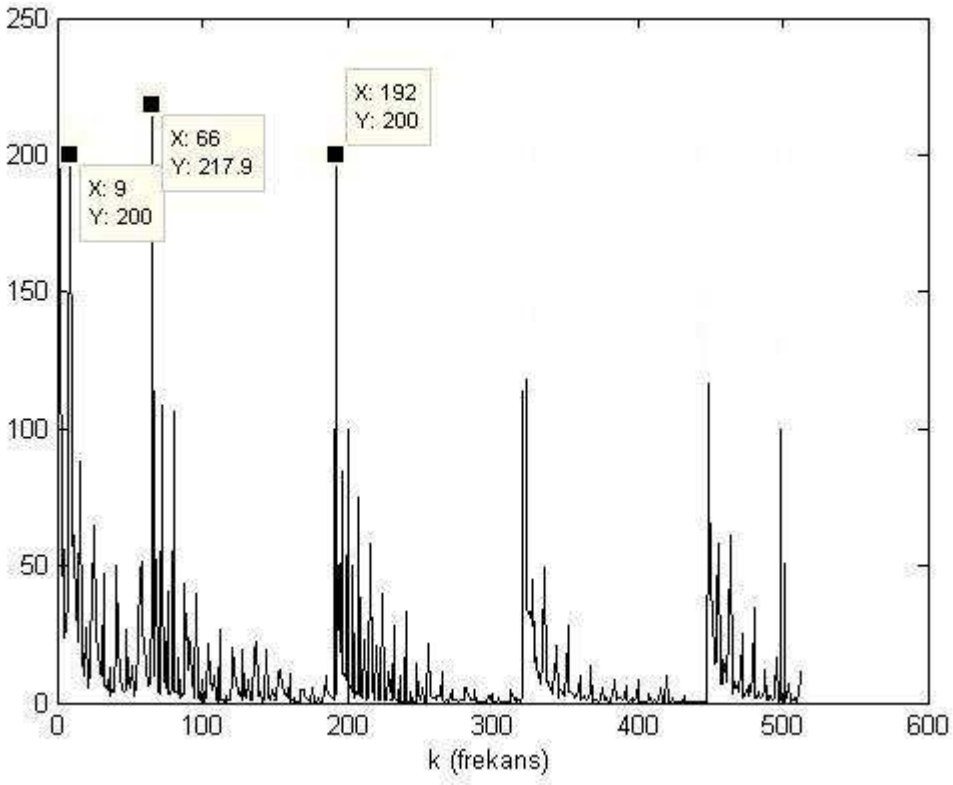
Şekil 4.21: Giriş İşareti Olarak Kullanılan Sinüs Dalgası

Giriş işaretinin MATLAB'ın optimize edilmiş FFT fonksiyonu ile çizilen frekans spektrumu şekil 4.22'de verilmiştir. İşaret 140 Hz, 1,031kHz ve 3kHz bileşenlerinden oluşmaktadır. Bu frekanslara karşı gelen k değerleri sırası ile 9, 66 ve 192'dir.



Şekil 4.22: Giriş İşaretinin Frekans Spektrumu

Şekil 4.20'deki akış diyagramı her bir aşamaya uyarlanarak, sekiz aşama sonunda elde edilen değerler, uygun şekilde yerleştirilerek oluşturulan frekans genlik spektrumu şekil 4.23 ile gösterilmiştir. Beklenildiği gibi, optimize FFT fonksiyonu ile oluşturulan frekans spektrumu ile farklılık göstermekte, frekans bileşenleri çevresinde pikler oluşmaktadır. Ayrıca spektrum Nyquist frekansı etrafında simetrik değildir, giriş işaretinin frekans bileşenleri için genlik değerleri eşit beklenmekte ancak optimize fonksiyonla elde edilen frekans spektrumunda da olduğu gibi, birbirine yakın ve yanal piklerden yüksek olmakla beraber, eşit çıkmamaktadır.



Şekil 4.23: 512 Nokta Cooley-Tukey FFT

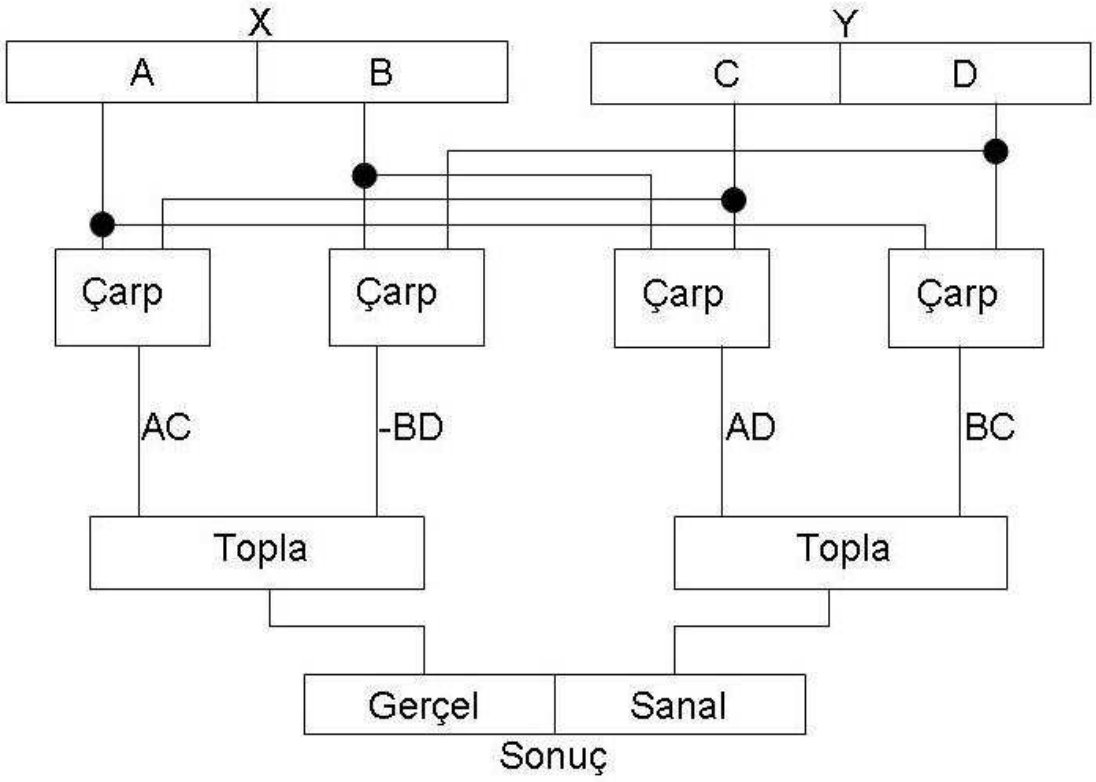
4.4.3. Toplama ve Çarpma Devrelerinin Gerçeklenmesi

X=A+jB ve Y=C+JD şeklinde gösterilen X ve Y karmaşık sayılarının toplamı (4.32) eşitliği ile çarpımları ise (4.33) eşitliği ile bulunur.

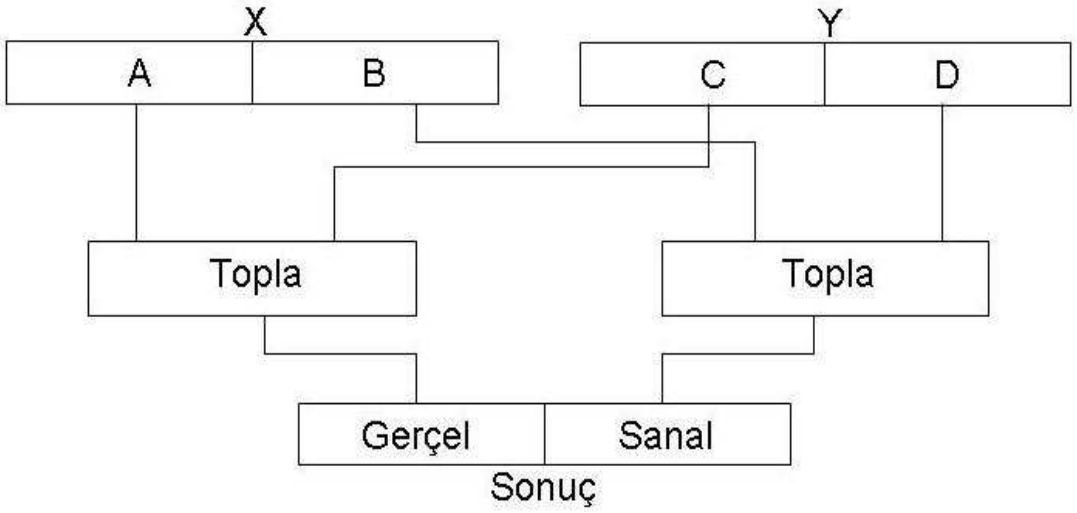
$$X + Y = (A + C) + j(B + D) \tag{4.32}$$

$$XY = (AC - BD) + j(AD + BC) \tag{4.33}$$

Karmaşık çarpma işlemini gerçeklemek için kullanılan blok diyagram şekil 4.24 ile, karmaşık toplama işlemini gerçeklemek için kullanılan blok diyagram ise şekil 4.25 ile verilmiştir.



Şekil 4.24: Karmaşık Sayılarla Çarpma



Şekil 4.25:Karmaşık Sayılarla Toplama

Şekil 4.24 ve şekil 4.25'te gösterilen *Topla* ve *Çarp* blokları, yarı duyarlı kayan nokta gösterimli sayılar için oluşturulmuş toplama ve çarpma bloklarıdır. Bu bloklar Goertzel Algoritması için kullanılan ve işlemleri bölüm 4.1.2'de anlatılan tek duyarlı kayan nokta çarpma ve toplama bloklarının yarı duyarlı kayan nokta gösterilimi için yeniden düzenlenmiş halidir. Tek duyarlı toplama ve çarpma bloklarına göre, kapladığı alan yaklaşık yarı yarıya azalmıştır, bu da karmaşık sayılarla toplama ve

karmaşık sayılarla çarpma modüllerinde birden fazla *Topla* ve *Çarp* bloğu kullanılarak toplama ve çarpma işlemlerinin paralel olarak yapılmasına olanak vermektedir.

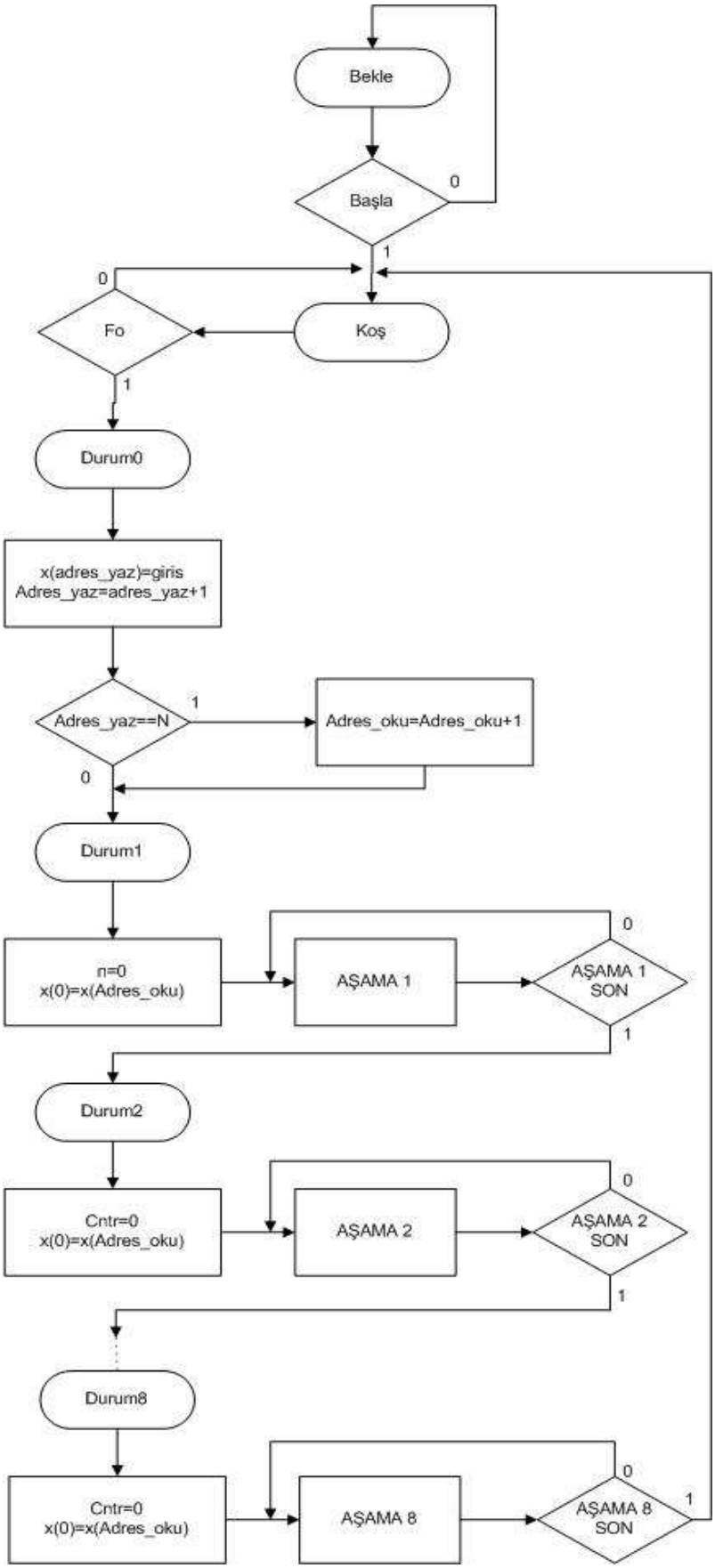
4.4.4. FPGA Üzerinde Gerçekleme

Kullanılan FPGA üzerinde bulunan blok RAM'ler kullanılarak, akış diyagramına göre elde edilen çıkışlar her bir aşama sonrasında kaydedilecektir. Aşama sonuçları dışında, giriş işareti de bir blok RAM'e yazılacaktır. Bu nedenle toplam 9 adet 512 gözlü RAM kullanılması gerekmektedir. SPARTAN 3E içindeki toplam blok RAM kapasitesi de göz önüne alındığında, her bir blok RAM'in genişliğinin 32 bit, uzunluğunun 512 sözcük olması uygun görülmüştür.

Katsayılar da benzer şekilde bir ROM içinde tutulacaktır. Katsayıların karmaşık sayı olması her bir aşama sonunda elde edilen değerlerin de karmaşık olmasına neden olur. Aşama çıkışlarının yazıldığı RAM'lerde sonuçlar gerçel ve sanal olmak üzere iki kısım halinde tutulacaktır. Gerçel ve sanal kısım uzunlukları birbirine eşit ve on altışar bit seçilmiştir. Giriş işareti gerçel olduğu için, giriş işaretinin tutulduğu RAM'in uzunluğunu otuz iki bit seçme zorunluluğu olmasa da, ileride ters FFT işleminin gerçeklenmesinde kolaylık sağlayabilmesi için bu RAM de 32 bit uzunluklu oluşturulmuş; ancak her bir gözün yüksek anlamlı ilk 16 biti giriş işaretine ayrılmış, diğer bitleri sıfır olarak bırakılmıştır. Diğer RAM'lerde ise yüksek anlamlı 16 bit sayının gerçel kısmına, düşük anlamlı 16 bit ise sayının sanal kısmına ayrılmıştır.

Hassalığın sağlanabilmesi için yarı duyarlı kayan nokta aritmetiği kullanılmıştır.

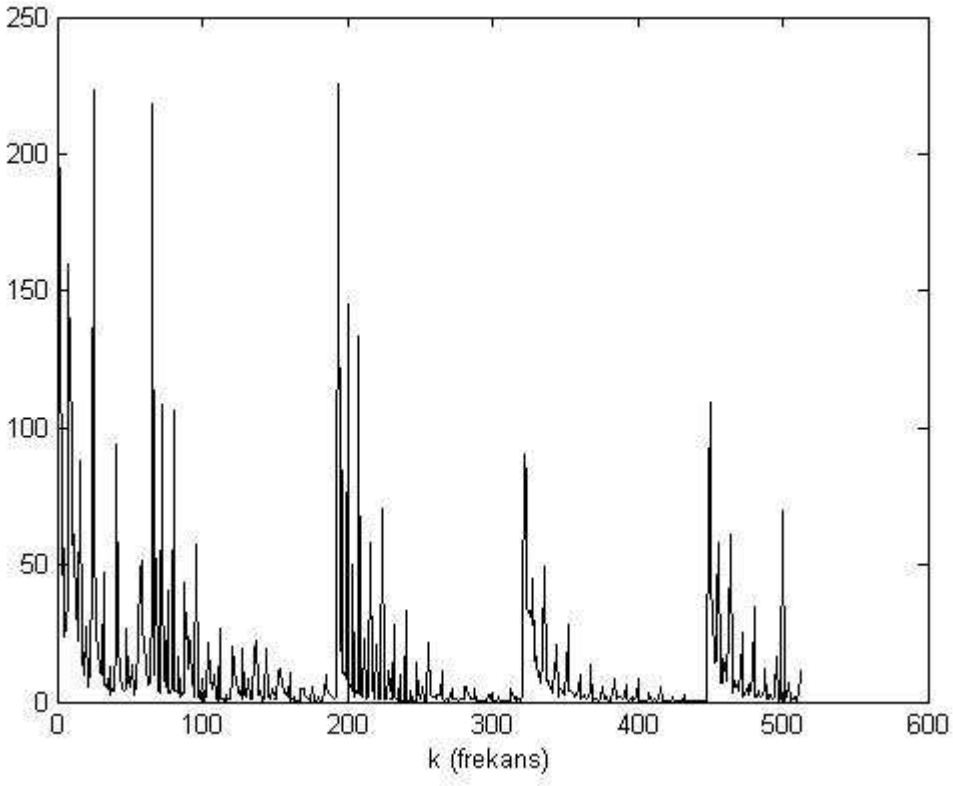
Şekil 4.26'da kullanılan blok diyagram verilmiştir. Burada AŞAMA1, AŞAMA2… olarak verilen işlemlerin her birinde şekil 4.20'de tek bir aşama için gösterilen akış, N değeri değiştirilerek kullanılmaktadır. *Fo*, yeni bir giriş alınıp alınmadığını göstermektedir, örnekleme periyodu ile tetiklenen bir işaret gibi davranmaktadır. Ötelemeli yazmaç yerine Rader algoritmasında olduğu gibi işaretçi kaydırma yöntemi kullanılmıştır.



Şekil 4.26: 512 Nokta Cooley-Tukey FFT Algoritması Blok Diyagramı

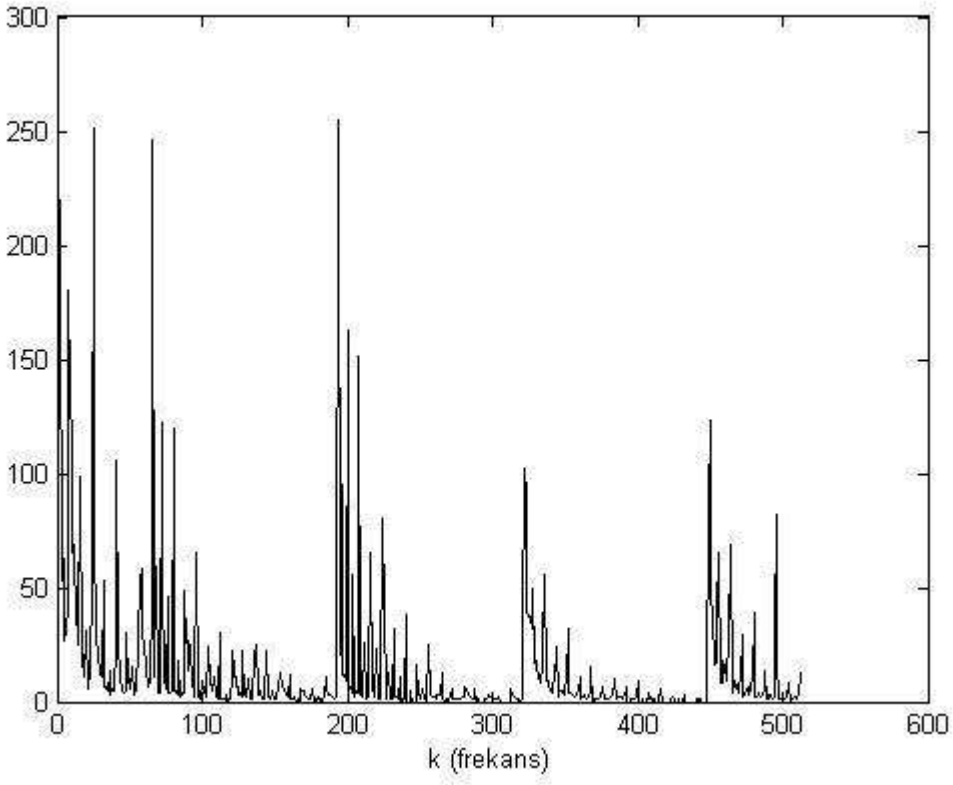
FFT işleminde kullanılacak olan katsayılar, yarı duyarlı kayan nokta hassasiyetine getirilmiş ve ROM'a kaydedilmiştir. FFT bloğu da, 8 aşamanın gerçekleştirildiği, RAM ve adres çözücülerden oluşan 8 ayrı alt blok giriş işaretinin tutulduğu, RAM ile yazma ve okuma adres çözücülerinden oluşan giriş bloğu, durum makinesi ile karmaşık toplama ve çarpma birimlerinden oluşan ana bloktan oluşur. Ana blokta yapılan işlem şekil 4.20'de verilen işlemdir, girişin alındığı RAM (x) ve çıkışın yazıldığı RAM (y) durum makinesi ile belirlenir. Hangi adrese veri yazılacağı ve hangi katsayının işlemlerde kullanılacağı ise aynı alt blokta bulunan adres çözücüler tarafından belirlenir.

Bilgisayar ortamında FFT bloğunun benzetimi yapılmış ve elde edilen çıkış değerleri, doğru şekilde sıralanarak şekil 4.27'de gösterilen frekans spektrumu oluşturulmuştur. MATLAB benzetiminden farklı olarak, yan piklerde artış gözlenmiştir. Ayrıca Nyquist frekansına göre simetrik olması beklenen spektrumda, Nyquist frekansının altında frekans bileşenlerinin genlikleri birbirine yakınken, üstünde frekans genlikleri azalmaktadır.



Şekil 4.27: Cooley-Tukey FFT Bloğu Çıkışı

Analog sayısal dönüştürücüden alınan giriş işaretinin, FFT bloğunda kullanılabilecek forma dönüştürülmesi için bir çevirici blok ve FFT bloğu çıkışının sayısal analog çeviriciye uyumlu forma getirilmesi için bir çevirici blok kullanılmalıdır. Böyle bir çevirici blok kullanıldığında, frekans genliklerinde değişim meydana gelecektir ancak spektrumun genel görüntüsünde büyük bir değişim gözlenmeyecektir. Yeni oluşturulan frekans spektrumu şekil 4.28'deki gibidir.



Şekil 4.28 :Cooley-Tukey Çıkış Çeviricisi Çıkışı

5. KARŞILAŞTIRMALAR

Bitirme çalışması süresince her biri farklı alanlarda sıkça kullanılan dört ayrı FFT algoritması incelenmiş, bunlardan üç tanesi kendilerinden beklenen avantajlar gözetilerek FPGA üzerinde farklı şekillerde gerçeklenmiştir. Bluestein Chirp-z FFT algoritması ise, yapısı itibariyle Rader algoritmasına benzemektedir, Rader algoritması için oluşturulan çarpma bloklarında katsayılara özel değişiklikler yapıldığında, Bluestein Chirp-z algoritmasına geçilebilir. Bu nedenle, Bluestein Chirp-z algoritması FPGA üzerinde gerçeklenmemiştir.

Algoritmalar arasındaki karşılaştırma kriterlerinden biri, algoritmalar gerçeklendiğinde FPGA'nın ne kadarını doldurduğudur. FPGA üzerinde kullanılan alanın ölçüsü olarak kullanılan dilim sayısı gösterilebilir; ancak aynı FPGA üzerinde başka uygulamalar da geliştirilecekse, kullanılan dilim sayısı yanında, blok çarpıcılar, blok RAM'ler gibi FPGA'nın diğer kaynaklarının kullanımı da karşılaştırılmalıdır. Örneğin, dilim kullanımı açısından Cooley-Tukey algoritması FPGA üzerinde çok fazla yük oluşturmasa da, blok RAM ve RAM'e bağlı saat hattı kullanımı yüksektir. FPGA üzerinde kapladıkları alan tablo 5.1 ile verilmiştir.

Gerçeklenen algoritmalardan, FPGA üzerinde en az yer tutan beklenildiği gibi Goertzel algoritmasıdır. FPGA üzerinde kapladığı alan düşünülürse, aynı FPGA paralel olarak yaklaşık 100 frekans değerinin hesaplanması için kullanılabilir, ayrıca FFT blokları seri olarak da kullanılıp, örnekleme frekansı ile devrenin saat frekansının oranına bağlı olarak bu sayı daha da arttırılabilir.

Rader algoritması ile 17 frekans değerine ait genlikler hesaplandığı halde, kapladığı alan Goertzel algoritmasının kapladığı alanın on yedi katından küçüktür. Öte yandan, Rader algoritması, asal sayıda nokta için uygulanabilir olduğundan, gerçeklemede genellenebilirliği düşüktür, nokta sayısı arttıkça hesaplanması güçleşir.

Tablo 5.1: FFT bloklarının FPGA üzerinde kapladıkları alan

Lojik Birimler	**Mevcut**	**Kullanılan**		
		Goertzel	**Rader**	**Cooley-Tukey**
Dilimler	4656	28	714	1088
Dilimlerdeki Flip-Floplar	9312	32	3919	4976
Dört Girişli LUT'lar	9312	45	7357	7088
IOB'ler	232	12	18	12
Saat	24	2	2	4
Blok RAM	20	0	0	9

Ancak, Rader Algoritması Cooley-Tukey algoritması ile karşılaştırıldığında, daha temiz frekans spektrumları verdiği görülmektedir. Nokta sayısının yüksek değerleri için Cooley-Tukey algoritması diğer ikisine göre avantajlıdır, bu algoritma daha kolay genellebildiğinden, uyarlanabilir FFT blokları yapmaya uygundur; fakat bu algoritmanın gerçeklenmesinde blok RAM'ler kullanıldığı sürece, devrenin saat frekansının 50MHz ile sınırlı olduğu unutulmamalıdır. Saat frekansı düştükçe, iki örnek arasında yapılabilecek maksimum işlem sayısı da azalacaktır, bu da erişilebilecek aşama sayınsını aynı zamanda da hesaplanabilecek frekans bileşeni genliğini azaltacaktır. Aşamaların üst üste çalıştırıldığı bir FFT bloğu için, 8kHz örnekleme frekansında en fazla 512 nokta FFT hesaplanabilir.

6. SONUÇLAR VE TARTIŞMA

DSP (Digital Signal Processor- Sayısal İşaret İşlemci) üzerinde defalarca gerçeklenmiş olan FFT algoritmaları bu çalışmada sayısal işaret işleme alanında da kullanımı gittikçe yaygınlaşan [9] FPGA üzerinde gerçeklenmiştir. Farklı hazır FFT bloklarının varlığı, diğer işaret işleme algoritmalarının FPGA üzerinde gerçeklenmesine yardımcı olacaktır. FPGA programlanma şekli nedeniyle, toplama ve çarpma gibi temel işlemlerin yapılandırılmasında DSP'ye göre daha kullanışlıdır. Bu da hız/alan oranının kullanıcı tarafından kontrolüne müsaade ederken, gerçeklenen algoritmanın da daha fazla geliştirilebilmesine olanak tanır.

Bitirme çalışmasında gerçeklenen Goertzel algoritması kullanılarak, aynı cihaz üzerinde farklı frekans değerlerinin tespitini yapan modüller yardımıyla giriş işaretinin frekans spektrumunu oluşturmak gibi çeşitli uygulamalar geliştirilebilir. Daha fazla frekans genliğinin elde edilmesi için diğer modüller yerine, Goertzel FFT modüllerinin ayrı ayrı kullanılması, faz spektrumunun kullanılmadığı uygulamalarda avantajlıdır; doğrudan genlik değeri hesaplandığı için karmaşık sayılarla işlem yapmayı gerektirmez.

32'den daha az nokta için Fourier dönüşümü kullanıldığında, gerek Rader algoritmasında gerekse Bluestein Chirp-z algoritmasında, katsayıların kullanıldığı aritmetik işlemlerinin aynı çarpma ve toplama blokları yerine, katsayıların her biri için özel çarpma blokları kullanılabilir. Böyle oluşturulan devrelerde toplama ya da çarpma yükü azaltılabilir [9].

Daha kesin ve güvenilir frekans spektrumları oluşturabilmek için, bu çalışmada gerçeklenen FFT algoritmaları birbiri ile birleştirilerek, optimize edilmelidir.

KAYNAKLAR

[1] http://www.ti.com

[1] **Maxfield, C.,** 2004. The design warrior's guide to FPGAs, Elsevier, Amsterdam.

[2] http://www.xilinx.com

[3] **PM10855**, 2005. SPARTAN FPGA's

[4] **DS312**, 2005. SPARTAN 3E Datasheet

[5] **UG230,** 2006. Starter Kit Board User guide

[6] **Edelman, A., McCorquodale P., Toledo, S.,** 1999, The Future Fast Fourier Transform, *SIAM J. Sci. Computing,* **20,** 1094–1114.

[7] **Smith, S.,** 2003, Digital Signal Processing, Newnes, USA

[8] **Mano, M.Morris**, 2005, Digital Design, Prentice Hall, USA

[9] **Maeyer, U.,** 2001. Digital Signal Processing With Field Programmable Gate Arrays, Springer, Germany

[10] **Stein, Y.,** 2000. Digital Signal Processing A Computer Science Perspective, John Willey & Sons, Canada

[11]**Stroobach, J.,** 1992. Digital DTMF Tone Detector Kanata, *United States Patent*, No: 5119322, dated: 6.2.1992

[12] **Kahan, W.,** 1997. Lecture Notes on the Status of IEEE 754, Elect. Eng. & Computer Science University of California, USA

[13] **Goldberg, D.,** 1991. What Every Computer Scientist Should Know About Floating-Point Arithmetic, *ACM Computing Surveys*, **23,** 5-48

[14] **Cooley, J. W., Tukey, J. W**., 1965. An algorithm for the machine calculation of complex Fourier series, *Mathematical Computing.* **19**, 297–301

[15] **Chassaing, R.,** 1992. Digital Signal Processing with C and the TMS320C30, John Wiley & Sons, USA

[16] http://www.fftw.org/

ÖZGEÇMİŞ

1987 yılında doğan Tuba AYHAN, Ankara Fen Lisesi'nden 2004 yılında derece ile mezun olduktan sonra lisans öğrenimini İTÜ Elektronik Mühendisliği'nde görmeye başladı. 2006-2007 akademik yılında, K. Leuven Üniversitesi'nde bulunarak lisans öğreniminin bir yılını Belçika'da geçirdi. Birçok kez, İTÜ Yüksek Onur listesine giren Tuba AYHAN, 2004 senesinde ARI ödülüne ve halen devam etmekte olan İTÜ başarı bursuna layık görülmüştür.