Hızlı Fourier Dönüşümünün FPGA Uygulamasının SQNR Simülasyonu

SQNR Simulations of Fast Fourier Transform Implementation on FPGA

Burak Kelleci[1], Mehmet Öner[2], Hakan Başaran[2]

[1]Mühendislik ve Mimarlık Fakültesi
Okan Üniversitesi
burak.kelleci@okan.edu.tr

[2]Koç Bilgi ve Savunma Teknolojileri A.Ş.
mehmet.oner@kocsavunma.com.tr, hakan.basaran@ kocsavunma.com.tr

Özet

Hızlı Fourier Dönüşümünün (FFT) FPGA üzerinde sabit noktalı sayılar kullanarak uygulamasının sinyal-kuantalama gürültüsü-oranının (SQNR) Cordic algoritması kullanılarak ve kullanılmadan elde edilen performansı gösterilmiştir. Simülasyon sonuçları SQNR değerinin minimum ve ortalama değerlerinin arasındaki farkın uzunluğu kısa olan FFT için arttığını göstermiştir. 4-noktalı FFT için minimum SQNR değeri ortalama SQNR değerinden 28dB kadar düşük bulunmuştur. Uzunluğu kısa olan FFT için sabit noktalı sayıların kesirli kısmının uzunluğu arttırılarak en kötü durumda bile SQNR performansının sağlanması garanti edilmelidir.

Abstract

Signal-to-Quantization-Noise Ratio (SQNR) performance of Fast Fourier Transform (FFT) Implementation on an FPGA using fixed point numbers with and without Cordic algorithm is presented. Simulation results indicate that the difference between minimum SQNR and mean SQNR values is increased for small length FFT. For 4 point FFT minimum SQNR value is up to 28dB lower than its mean value. For small length FFT, fraction length of fixed point operations should be increased to guarantee SQNR performance for the worst case condition.

1. Giriş

Fourier dönüşümü 1768 yılında Joseph Fourier tarafından sinyallerin sinüzoidal fonksiyonların toplamları cinsinden yazılması olarak önerilmiştir. Yirminci yüzyılın ikinci yarısında sayısal işaret işleme donanımlarının ucuzlaması ve performanslarının kompleks işlemlere izin vermesiyle beraber Fourier dönüşümünün sayısal sistemlere uygulanması üzerine çalışmalar yapılmıştır. Çalışmalar özellikle ayrık zamanlı Fourier dönüşümünün gerektirdiği matematiksel işlemleri azaltma üzerine yapılmıştır. Bu konuda dönüm noktası, parçala ve hesapla mantığına dayalı olan Cooley-Tukey algoritması olmuştur[1]. Algoritma tabanlı ve direkt hesaplamadan daha hızlı olan Fourier dönüşümüne hızlı Fourier dönüşümü (FFT) denilmiştir.

Cooley-Tukey algoritması toplam işlem sayısını, direkt olarak hesaplamanın işlem sayısı olan N^2'den $N\log_2 N$ işlem adedine düşürmüştür. Burada N ayrık Fourier dönüşümünün uzunluğudur. Ayrıca bu algoritma N sayısını 2^v'ye eşit olacak şekilde sınırlamıştır. Burada v sayısı pozitif tam sayı olmak zorundadır. Bu sınırlama algoritmanın problemi her seferinde iki eşit parçaya bölmesinden dolayı oluşmaktadır. Bu yönteme aynı zamanda radix-2 FFT algoritması da denilmektedir. Benzer şekilde sürekli ikiye bölme yerine dörde, sekize ya da onaltıya bölme yöntemleri de kullanılabilir. Bu yöntemlere sırasıyla radix-4 FFT, radix-8 FFT ve radix-16 FFT denmektedir. Dörtden daha yüksek bölme oranları radix algoritmasının karmaşıklığını arttırdığından kazanç beklenildiği kadar yüksek olmamaktadır. 1984 yılında split-radix algoritması önerilmiştir [2]. Bu algoritmanın temel mantığı sinyalin çift tarafı için radix-2 ve tek tarafı için radix-4 kullanılmasıdır.

Literatürde N değerinin asal sayılarının çarpımı olması durumunda hesaplama algoritması Good tarafından önerilmiştir [3]. Daha sonra Good'un algoritmasındaki asal sayı uzunluktaki kısımların FFT hesabını da konvolüsyon kullanarak hesaplanması Winograd Fourier dönüşüm algoritması olarak gösterilmiştir [4]. 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 [5].

Kayan noktalı sayılarla matematiksel işlemler sabit noktalı sayılarla yapılan matematiksel işlemlere göre daha fazla alan ve güç gerektirdiği için matematiksel işlemler sayısal sistemlerde mümkün olduğunca sabit noktalı sayılarla yapılmaktadır. FFT algoritması da aynı şekilde sabit noktalı sayılarla gerçekleştirilmiş ve yuvarlamadan dolayı FFT çıkışındaki kuantalama gürültüsü belirlenmiştir [6, 7].

Kuantalama hatasının gücü kullanılarak rastgele olmayan bir giriş sinyali için FFT çıkışındaki sinyalin SQNR değeri hesaplanabilmektedir [7]. Ancak giriş sinyali pratikte rastgele özelliktedir. Haliyle FFT çıkışında hesaplanan SQNR değeri

de rastgele bir sinyal olacaktır. Bu rastgele SQNR sinyalinin ortalama değeri literatürde belirtilen yöntemlerle hesaplanabilmesine karşın SQNR değerinin en kötü hali hakkında herhangi bir çalışma bulunmamaktadır. Bu çalışmada Monte Carlo analiz yöntemiyle SQNR değerinin en kötü hali radix-2 FFT algoritması için belirlenmiştir. FFT algoritmasının gerektirdiği faz faktörlerinin CORDIC algoritması ile hesaplanmasının etkisi de Monte Carlo yöntemiyle belirlenmiştir.

2. Hızlı Fourier Dönüşümü

Fourier dönüşümü temel olarak N adet kompleks x(n) veri serisini N adet X(k) serisine aşağıdaki eşitliği kullanarak çevirmektedir [7].

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn},\quad 0\le k\le N-1 \tag{1}$$

Burada faz faktörü W_N aşağıdaki şekilde yazılmaktadır.

$$W_N=e^{-j2\pi/N} \tag{2}$$

Eşitlik (1)'den görüldüğü üzere her k değerini hesaplamak için N adet kompleks çarpma (4N gerçek sayı çarpma) ve N-1 adet kompleks toplama (4N-2 gerçek sayı toplama) gerekmektedir. Haliyle N adet k değerini hesaplamak için N^2 adet kompleks çarpma ve N^2-N adet kompleks toplama gerekmektedir. Ayrıca hesaplanan sonuçların tutulması için 2N adet hafıza bölgesine ihtiyaç duyulmaktadır.

Fourier dönüşümünün direkt olarak hesabı faz faktörünün simetri ve tekrarlama özelliklerini kullanmadığından çok verimli olmamaktadır. Literatürdeki radix türü algoritmalar faz faktörlerinin bu özelliklerini kullanarak hesap miktarını azaltmaktadırlar. Bu algoritmalardan en çok kullananı radix-2 algoritması olup, Fourier dönüşümü parçala ve hesapla yöntemiyle tekrar eden hesaplara indirgenmektedir. Bu sayede FFT için gereken kompleks çarpma sayısı $(N/2)\log_2(N)$'e düşmektedir. Gereken kompleks toplama ise $N\log_2 N$ kadar olmaktadır. Ancak her adımda bütün değerleri hafızada tutmak gerektiği için hafıza gereksiniminde bir değişiklik olmamaktadır.

Parçala ve hesapla yönteminde parçalama iki farklı şekilde yapılmaktadır. İlk yöntemde giriş dizisi tek indeks ve çift indeks değerleri olarak ayrılarak hesaplanmaktadır. Bu yönteme zamanda seyreltmeli FFT denmektedir. İkinci yöntemde ise giriş dizisi ortadan ikiyi ayrılarak hesaplanmaktadır. Bu yönteme de frekansta seyreltme denilmektedir. Matematiksel olarak iki yöntemde eşdeğerdir ve çarpma, toplama sayıları ve kullanılan hafıza miktarları bakımından eşdeğerdir.

2.1. Kuantalama Hataları

Eşitlik (1)'de verilen Fourier dönüşümündeki x(n) veri serisinin gerçek ve sanal kısımlarını ve W_N'in bit uzunlukları b olduğu varsayıldığında, her $x(n)W_N$ çarpımı (2b+1) adet bit ile gösterilen gerçek ve sanal kısımlara sahip bir kompleks sayı olur. Bit sayısının sürekli büyümesinin önüne geçmek için her kompleks çarpmadan sonra bit sayısının tekrar b bite düşürüldüğünü varsayalım. Bu durumda her kompleks çarpma için toplam dört adet kuantalama hatası oluşmaktadır. Her DFT'de N çarpma yapıldığından toplamda 4N adet kuantalama hatası oluşacaktır. Eğer kuantalama hatalarının düzgün bir şekilde $-2^{-b}/2$ ile $2^{-b}/2$ arasında dağıldığını, kuantalama hatalarının birbirlerinden ve x(n) sinyalinden bağımsız olduğunu kabul edilirse, her bir kuantalama hatası aşağıdaki şekilde yazılmaktadır.

$$\sigma_e^2=\frac{\Delta^2}{12}=\frac{2^{-2b}}{12} \tag{3}$$

Toplamda 4N adet çarpma olduğundan ve N'in 2^v'ye eşit olduğu varsayılırsa kuantalama hatası

$$\sigma_q^2=4N\sigma_e^2=\frac{N}{3}2^{-2b}=\frac{2^{v-2b}}{3} \tag{4}$$

olmaktadır. Eşitlik (4)'den görüldüğü üzere aynı hassasiyeti sağlayabilmek için N sayısının her dört kat artışı bit sayısının bir adet büyümesini gerektirmektedir.

x(n) sinyali -1, +1 arasında limitli olarak kabul edildiğinde, X(k) sinyali N değerine kadar çıkabilmektedir. Çıkışın -1, +1 arasında limitli olması için giriş sinyalinin 1/N ile çarpılması gerekmektedir. Fakat bu durumda giriş'te kullanılan bit sayısının arttırılarak giriş sinyalinin sinyal-kuantalama gürültüsü-oranın bölmeden dolayı azalmasının engellenmesi gerekmektedir. Bu ölçeklendirme işlemi radix-2 algoritmasında kelebek yapılarına dağıtılabilinir. Bu durumda çıkıştaki kuantalama hatası

$$\sigma_q^2=\frac{2}{3}2^{-2b}\left[1-\left(\frac{1}{2}\right)^v\right]\approx\frac{2}{3}2^{-2b} \tag{4}$$

olmaktadır. v'nin büyük değerleri için bu yaklaşıklık büyük bir hata getirmeyecektir. (4)'den görüldüğü üzere kuantalama hatası N'e bağlı değildir. Ancak sinyal seviyesi N'e bağlı olduğundan sinyal-gürültü-oranı aşağıdaki şekilde yazılmaktadır.

$$\frac{\sigma_X^2}{\sigma_q^2}=\frac{1}{N^2}2^{2b}=2^{2b-v-1} \tag{5}$$

(5) nolu eşitlikteki giriş sinyali rastgele bir sinyal olduğundan kuantalama hatası da rastgele bir sinyal olmaktadır. Haliyle hesaplanan sinyal-gürültü-oranı rastgele bir sinyal olmakta ve (5) nolu eşitlikte bulunan sonuç bu rastgele sinyalin ortalama değeri olmaktadır. Sinyal-kuantalama gürültüsü-oranının minimum değeri ise Monte Carlo analizleri ile belirlenmektedir.

2.2. CORDIC Algoritması

FFT hesabı esnasında faz faktörü W_N'in değerleri gerekmektedir. Eşitlik (2)'de yazılmış olan faz faktörü Euler ilişkisi kullanarak aşağıdaki şekilde yazılmaktadır.

$$W_N^k=\cos(2\pi k/N)-j\sin(2\pi k/N) \tag{6}$$

Radix-2 algoritması için N/2 adet W_N değeri gerekmektedir. Bu değerlerin ROM'da saklanabileceği gibi anlık olarak

hesaplanabilir. Anlık olarak hesaplama özellikle büyük N değerleri için alan bakımında avantajlı olabilmektedir.

Eşitlik (6)'deki sin ve cos fonksiyonları donanımda CORDIC algoritması kullanılarak hesaplanmaktadır [8]. CORDIC algoritması her döngüde sadece bir tablo bakma operasyonu, iki kaydırma operasyonu ve üç adet toplama operasyonu ile gerçek değere yakınsamaktadır. Her CORDIC operasyonu istenilen açının akümülatöre yüklenmesiyle başlamaktadır. Dönme kararı her operasyondan sonra kalan açının işaretine göre yapılmaktadır.

Dönme modunda CORDIC eşitlikleri üç adet eşitlik ile aşağıda gösterilmiştir.

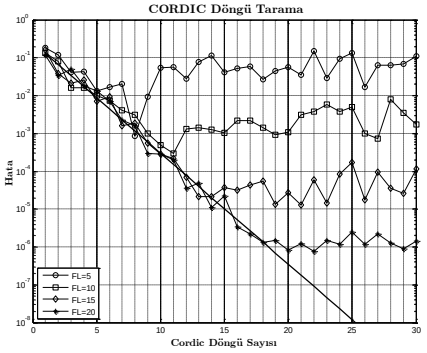
$$\begin{aligned} z_{i+1} &= z_i - d_i \cdot \operatorname{atan}\left(2^{-i}\right) \\ x_{i+1} &= x_i - y_i \cdot d_i \cdot 2^{-i} \\ y_{i+1} &= y_i + x_i \cdot d_i \cdot 2^{-i} \end{aligned} \tag{7}$$

Burada d_i eğer $z_i<0$ ise -1 aksi halde +1 değerini almaktadır. i ise 0,1,...,N-1'e kadar tamsayı değer almaktadır ve N toplam döngü sayısıdır.

3. Simülasyon

3.1. CORDIC Algoritması

Sin ve Cos değerlerini hesaplamak için (7) nolu eşitlikteki döngü başlangış değerleri, z_0=istenilen açı değeri, $x_0=1/A_N$ ve $y_0=0$ olarak alınmaktadır. N döngü sonra $x_n=\cos(\theta)$ ve $y_n=\sin(\theta)$ olmaktadır. Donanımda FFT işlemleri kayan noktalı sayılar yerine kesirli sayılar kullanılarak yapıldığından döngü sayısını kesirli sayının bit uzunluğundan çok daha büyük seçmek avantaj getirmemektedir. Şekil 1'de çeşitli kesirli sayı bit uzunlukları için CORDIC algoritmasının tekrarı simüle edilmiştir. Simülasyon esnasında $\pm\pi/2$ arasında her durum için 1000 adet rastgele açı değeri yaratılmış ve bu değer CORDIC algoritması ile hesaplanmıştır. Şekil 1'de gösterilen FL değeri ise kesirli sayının bit uzunluğunu göstermektedir. Simülasyonlar esnasında taşmayı önlemek ve işaret için ekstradan iki bit daha kullanılmıştır. Görüldüğü üzere döngü sayısının kesirli sayının bit uzunluğunu geçtikten sonra gerçek değerle hesaplanan değer arasındaki hata CORDIC algoritmasının döngüsünden bağımsız olmaktadır. Bu yüzden N sayısı açı için kullanılan kesirli sayının bit uzunluğu kadar seçmek yeterli olmaktadır.



Şekil 1: CORDIC Döngü Tarama

3.2. FFT Algoritmasının Seri Uygulaması

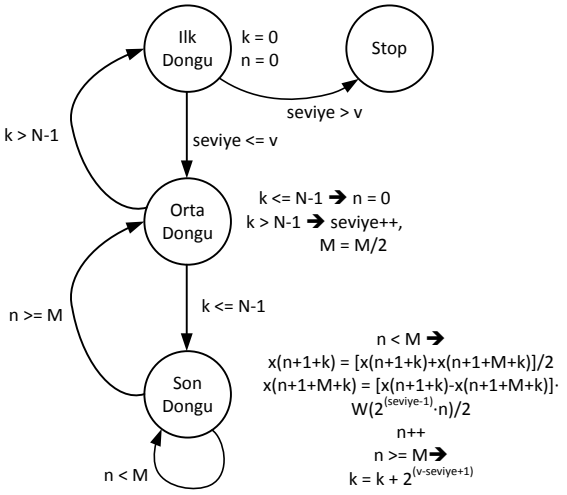
FFT yapısı seri ya da paralel olarak uygulanabilir. Şekil 2'de gösterilen sonlu durum makinası ile gerçeklenen seri yapıda toplama ve çarpma alanlarını azaltmak için her saat periyodunda temel kelebek yapısındaki tek bir hesaplama yapılmaktadır. Fakat seri yapı gerekli çıkışı üretmek için kelebek yapısının derinliğine bağlı olarak çok sayıda saat periyoduna ihtiyaç duyacaktır. Gereken saat periyodu sayısı

$$v \cdot 2^{v-1} + 2^v + 2 \cdot v + 1 \tag{8}$$

olur. v değişkeni kelebek yapısının derinliği olup, FFT sayısı ile arasında $N=2^v$ ilişkisi bulunmaktadır. Eğer gerekli faz faktörlerini anlık olarak hesaplayarak gereken hafıza miktarı azaltılmak istenirse, CORDIC algoritması kullanılması gerekmektedir. CORDIC algoritması döngü tabanlı olduğundan gerekli faz faktörü değerini elde etmek için ekstra saat periyodu gerekecektir. Bu durumda gerekli saat periyodu

$$(niters+1)\cdot v \cdot 2^{v-1} + 2^v + 2 \cdot v + 1 \tag{9}$$

şeklinde yazılıp, niters parametresi CORDIC algoritmasının gereksinim duyduğu döngü adedidir. Bu değeri kullanılan kesirli sayının bit uzunluğu kadar seçmek yeterli olmaktadır. Daha fazla döngü ile elde edilen iyileşme kesirli sayının kuantalama gürültüsünün altında olduğundan döngü sayısını arttırmak Şekil 1'de gösterildiği üzere sinyal-kuantalama gürültüsü-oranı'nı iyileştirmemektedir.



Şekil 2: Seri FFT Uygulamasının Sonlu Durum Makinası

Çizelge 1: Seri FFT Uygulaması için Gereken Saat Frekansı FFT işlem süresi oranı

Gereken Saat Frekansı Oranı								
		Kelebek Derinliği						
		2	4	6	8	10	12	14
FFT Uzunluğu (N)		4	16	64	256	1024	4096	16384
Cordic		15	71	331	1551	7187	32791	147483
Cordic İterasyon Sayısı	1	19	103	523	2575	12307	57367	262171
	2	23	135	715	3599	17427	81943	376859
	4	31	199	1099	5647	27667	131095	606235
	6	39	263	1483	7695	37907	180247	835611
	8	47	327	1867	9743	48147	229399	1064987
	10	55	391	2251	11791	58387	278551	1294363
	12	63	455	2635	13839	68627	327703	1523739
	14	71	519	3019	15887	78867	376855	1753115
	16	79	583	3403	17935	89107	426007	1982491
	18	87	647	3787	19983	99347	475159	2211867
	20	95	711	4171	22031	109587	524311	2441243

3.3. FFT Algoritmasının SQNR Simülasyonları

FFT'nin giriş sinyalinin istatistiksel yapısından dolayı ve kuantalama gürültüsünün de istatistiksel olarak değişmesinden dolayı sinyal-kuantalama gürültüsü-oranı istatistiksel olarak değişmektedir. Eşitlik (4) ve (5)'deki değerler bu değişimin ortalama değerleri olup olabilecek en kötü hal hakkında bilgi vermemektedir. Bu yüzden Monte Carlo simülasyonları ile en kötü hal analizinin yapılması gerekmektedir.

Giriş sinyali olarak da rastgele yaratılmış bir sinyal kullanılmıştır. Her x(n) değeri Gaussian dağılıma sahip varyans değeri bir olan rastgele sayı üreticinin çıktısı kullanılarak yaratılmıştır. Giriş sinyali -1,+1 arasında olması gerektiğinden bu değerlerin dışında gelmiş olan sayılar ret edilmiş ve yerine başka bir sayı ürettirilmiştir. Kullanılan Matlab kodu Ek A'da gösterilmiştir.

Sinyal-kuantalama gürültü-oranı hesabı için FFT hesabı önce ideal IFFT algoritması kullanılarak zaman domenine çevrilmiş ve giriş sinyalinden çıkarılmış ve kuantalama hatası tespit edilmiştir. Sinyalin RMS'inin bu kuantalama hatasının RMS değerine bölünmesiyle sinyal-kuantalama gürültüsü-oranı elde edilmiştir.

Simülasyonlar hem frekansta seyreltme yöntemiyle hem de zamanda seyreltilme yöntemiyle yapılmıştır. Frekansta seyreltme yöntemiyle hesaplanan FFT, zamanda seyreltme yöntemiyle hesaplanan FFT'ye göre çoğunlukla 0 ila 2.3dB daha iyi performans göstermektedir. 1000 adet rastgele üretilmiş sinyalle yapılan simülasyon sonucunda elde edilen en düşük değerler Çizelge 2'de verilmiştir.

Çizelge 3'te frekansta seyreltme yöntemiyle gerçeklenen FFT yapısının SQNR değerlerinin ortalaması verilmektedir. Eşitlik (5)'te gösterildiği üzere kesirli kısmın bit sayısının her artışı SQNR değerini 6dB kadar iyileştirmektedir. FFT nokta sayısı N değerinin her ikiye katlanması SQNR değerini 3dB kadar azaltmaktadır. Çizelge 3'te elde edilen değerler bu eşitlik ile uyumlu çıkmışlardır. Ayrıca çizelge 3'te kalın çizgi ile gösterilen sınırın altında kalan kısım SQNR değerinin 0dB'den daha büyük olduğu durumları göstermektedir. Görüldüğü üzere 32768 noktalı FFT için en azından 9 kesir biti kullanılmalıdır. Pratikte 40dB'den daha iyi performans için en azından 16 kesir biti kullanılmalıdır.

Çizelge 2: Frekansta Seyreltmeli FFT Algoritması ile Zamanda Seyreltmeli FFT Algoritmasının Sinyal-Kuantalama Gürültüsü-Oranı'larının Ortalamalarının Farkı

Frekansta Seyreltmeli FFT Algoritması ile Zamanda Seyreltmeli FFT Algoritmasının SQNR'larının Ortalamalarının Farkı (dB)								
		Kelebek Derinliği						
		2	4	6	8	10	12	14
N		4	16	64	256	1024	4096	16384
Kesirli Kısmın Bit Sayısı	1	0.00	0.19	0.64	0.90	1.05	1.21	1.37
	2	0.01	-0.11	0.00	0.17	0.34	0.46	0.56
	3	-0.01	0.09	-0.05	-0.04	0.03	0.01	-0.01
	4	-0.02	0.20	0.37	0.06	-0.12	-0.13	-0.16
	5	-0.05	0.37	0.86	0.63	0.14	-0.16	-0.25
	6	0.00	0.50	1.13	1.19	0.80	0.18	-0.22
	7	0.00	0.42	1.25	1.56	1.42	0.90	0.18
	8	0.00	0.61	1.34	1.77	1.80	1.53	0.95
	9	0.00	0.56	1.38	1.84	1.99	1.91	1.58
	10	0.00	0.57	1.39	1.86	2.10	2.11	1.97
	12	-0.03	0.61	1.41	1.90	2.15	2.24	2.25
	14	-0.05	0.56	1.42	1.91	2.16	2.26	2.30
	16	-0.02	0.59	1.40	1.89	2.15	2.26	2.31
	18	0.03	0.59	1.41	1.90	2.16	2.26	2.31
	20	-0.04	0.54	1.41	1.89	2.16	2.26	2.30

Çizelge 3: Frekansta Seyreltmeli FFT Algoritmasının Sinyal-Kuantalama Gürültüsü-Oranı'nın Ortalaması

Frekansta Seyreltmeli FFT Algoritmasının SQNR'ın Ortalaması (dB)								
		Kelebek Derinliği						
		2	4	6	8	10	12	14
N		4	16	64	256	1024	4096	16384
Kesirli Kısmın Bit Sayısı	1	-2.65	-9.86	-15.89	-21.74	-27.65	-33.54	-39.41
	2	3.37	-4.44	-10.69	-16.26	-21.88	-27.63	-33.45
	3	9.59	1.49	-5.29	-11.09	-16.43	-21.92	-27.63
	4	15.63	7.34	0.57	-5.68	-11.17	-16.41	-21.87
	5	21.38	13.31	6.54	0.24	-5.78	-11.19	-16.39
	6	27.60	19.34	12.49	6.18	0.09	-5.83	-11.17
	7	33.62	25.34	18.50	12.20	6.08	0.05	-5.85
	8	39.68	31.40	24.48	18.19	12.07	6.03	0.03
	9	45.73	37.37	30.53	24.18	18.08	12.03	6.01
	10	51.55	43.46	36.55	30.19	24.09	18.03	12.01
	12	63.68	55.52	48.61	42.26	36.12	30.07	24.04
	14	75.68	67.48	60.66	54.29	48.16	42.10	36.07
	16	87.94	79.60	72.66	66.32	60.20	54.14	48.11
	18	99.76	91.63	84.70	78.37	72.24	66.19	60.15
	20	111.79	103.56	96.74	90.41	84.29	78.23	72.19

Çizelge 4: Frekansta Seyreltmeli FFT Algoritmasının Sinyal-Kuantalama Gürültüsü-Oranı'nın Minimum Değeri

Frekansta Seyreltmeli FFT Algoritmasının SQNR'ın Minimum Değeri (dB)								
		Kelebek Derinliği						
		2	4	6	8	10	12	14
N		4	16	64	256	1024	4096	16384
Kesirli Kısmın Bit Sayısı	1	-16.19	-15.51	-18.45	-23.06	-28.44	-33.89	-39.64
	2	-9.66	-10.34	-12.84	-17.58	-22.50	-28.00	-33.66
	3	-3.37	-3.91	-7.24	-12.04	-17.13	-22.23	-27.79
	4	-3.40	1.25	-1.70	-6.55	-11.74	-16.73	-22.06
	5	3.90	8.20	4.30	-0.90	-6.44	-11.44	-16.53
	6	6.05	15.15	9.90	5.04	-0.38	-6.11	-11.30
	7	18.77	19.11	16.12	11.20	5.58	-0.23	-5.97
	8	26.87	24.42	22.59	17.04	11.59	5.75	-0.10
	9	30.12	32.27	28.68	23.17	17.46	11.77	5.89
	10	38.89	38.73	34.16	28.95	23.57	17.78	11.87
	12	45.29	50.77	46.35	41.31	35.54	29.82	23.91
	14	46.99	63.08	58.14	53.30	47.64	41.78	35.93
	16	70.45	74.98	69.99	65.17	59.56	53.87	47.97
	18	79.36	86.47	82.16	77.29	71.70	65.95	60.03
	20	92.79	99.32	94.64	89.42	83.70	77.90	72.06

Çizelge 4'te gösterilen SQNR'in minimum değeri ise az noktalı FFT için beklenen her eklenen kesir biti başına 6dB SQNR değerinin artması yada N değerinin iki katına çıkmasıyla SQNR değerinin azalması trendini sağlamamaktadır. Bunun nedeni az noktalı FFT'lerdeki kuantalama hatasının sinyal ile korelasyon göstermesidir. Bu fark çizelge 5'te açık olarak

gözükmektedir. Çok noktalı FFT için ortalama ile minimum değerler arasındaki fark 0dB'ye yaklaşmasına karşın 4 noktalı FFT için 20dB'nin üzerinde fark gözlemlenmiştir.

Çizelge 5: Frekansta Seyreltmeli FFT Algoritmasının Sinyal-Kuantalama Gürültüsü-Oranı'nın Ortalaması ile Minimum Değeri Farkı

Frekansta Seyreltmeli FFT Algoritmasının SQNR'ın Ortalaması ile Minimum Değeri Farkı (dB)								
		Kelebek Derinliği						
		2	4	6	8	10	12	14
N		4	16	64	256	1024	4096	16384
Kesirli Kısmın Bit Sayısı	1	13.55	5.64	2.56	1.32	0.80	0.35	0.24
	2	13.03	5.90	2.15	1.32	0.62	0.37	0.21
	3	12.96	5.39	1.95	0.95	0.71	0.30	0.17
	4	19.03	6.09	2.27	0.87	0.57	0.33	0.19
	5	17.47	5.11	2.24	1.14	0.66	0.25	0.14
	6	21.54	4.19	2.60	1.15	0.47	0.28	0.13
	7	14.85	6.23	2.38	1.00	0.50	0.28	0.13
	8	12.81	6.99	1.89	1.15	0.48	0.28	0.13
	9	15.61	5.10	1.85	1.01	0.62	0.26	0.12
	10	12.66	4.73	2.38	1.23	0.52	0.26	0.14
	12	18.38	4.75	2.26	0.95	0.59	0.24	0.12
	14	28.69	4.40	2.51	0.99	0.51	0.32	0.14
	16	17.50	4.62	2.67	1.15	0.64	0.27	0.14
	18	20.40	5.16	2.54	1.08	0.54	0.24	0.12
	20	19.01	4.25	2.10	1.00	0.59	0.33	0.13

CORDIC algoritması kullanılarak (6) nolu eşitlikte gösterilen faz faktörleri hesaplandığında SQNR değeri direkt hesaplamaya göre pratik SQNR değerleri için 1dB den daha az hata getirmiştir.

Çizelge 6: CORDIC Algoritmasından Dolayı SQNR azalması

CORDIC Algoritmasından Dolayı SQNR Azalması (dB)								
		Kelebek Derinliği						
		2	4	6	8	10	12	14
N		4	16	64	256	1024	4096	16384
Kesirli Kısmın Bit Sayısı	1	0.72	-0.98	-1.98	-2.63	-2.92	-2.99	-2.97
	2	-0.10	0.54	-0.42	-1.43	-1.96	-2.17	-2.22
	3	1.76	1.03	0.87	-0.56	-1.27	-1.49	-1.53
	4	2.79	1.36	3.78	2.02	0.62	0.24	0.27
	5	0.19	0.20	0.22	4.40	1.86	0.44	-0.03
	6	2.51	2.06	1.87	6.87	4.44	1.73	0.50
	7	0.88	0.85	0.67	0.58	8.84	5.14	2.13
	8	0.75	0.74	0.53	0.45	12.26	9.39	4.96
	9	0.71	0.63	0.51	0.41	0.36	14.23	9.67
	10	0.34	0.24	0.16	0.13	0.16	18.13	15.08
	12	1.00	-0.20	-0.62	-0.75	-0.80	-0.83	21.06
	14	0.59	0.35	0.43	0.35	0.33	0.31	0.30
	16	0.97	0.77	0.58	0.52	0.49	0.46	0.45
	18	0.81	0.80	0.57	0.51	0.47	0.46	0.45
	20	0.93	0.61	0.54	0.48	0.46	0.45	0.44

4. Sonuçlar

Hızlı Fourier Dönüşümünün FPGA üzerinde sabit noktalı sayılarla uygulamasının sinyal-kuantalama gürültüsü-oranın (SQNR) Cordic algoritması kullanılarak ve kullanılmadan elde edilen performansı belirlenmiştir. Bunun için FFT yapısı seri olarak uygulanarak gereken toplama ve çarpma sayıları azaltılmıştır.

Simülasyon sonuçları göstermiştir ki kelebek yapısının sayısı arttıkça minimum değer ile ortalama değer arasındaki fark azalmaktadır. Ancak düşük kelebek değerleri için fark yüksek çıkmaktadır. Bunun nedeni kelebek sayısını azalması ile kuantalama hatasının rastgeleliğinin azalmasıdır.

Literatürdeki çalışmalarda SQNR değerinin ortalama değeri hesaplanabilmesine karşın SQNR değerinin en kötü hali hakkında herhangi bir çalışma bulunmamaktadır. Bu çalışma ile SQNR değerinin en kötü hali Monte Carlo analiz yöntemiyle radix-2 FFT algoritması için belirlenmiştir.

5. Teşekkür

Bu çalışma, TÜBİTAK tarafından TEYDEB 1511 destek programı kapsamında 1120187 proje numarası ile desteklenen ve Koç Bilgi ve Savunma Teknolojileri A.Ş. tarafından yürütülen proje kapsamında yapılmıştır. Bu yayındaki hiçbir görüş ve tespit TÜBİTAK'ın resmi görüşü değildir.

6. Kaynaklar

[1] J. W. T. J. W. Cooley, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp*., Cilt 19, pp. 297-301, 1965.

[2] D. H. J. C. S. B. M. T. Heideman, "Gauss and the history of the FFT," *IEEE Acoust. Speech Signal Process. Magazine*, Cilt 1, pp. 14-21, 1984.

[3] H. H. P. Duhamel, "Split-radix FFT algorithm," *Electronic Letters*, Cilt 20, pp. 14-16, 1984.

[4] I. J. Good, "The interaction algorithm and practical Fourier analysis," *J. Roy. Statist. Soc. Ser. B*, Vol. B-22, pp. 372-375, 1960.

[5] S. Winograd, "On computing the discrete Fourier transform," *Proc. Nat. Acad. Sci.*, Cilt 73, pp. 1005-1006, 1976.

[6] L. R. Morris, "A comparative study of time efficient FFT and WFTA programs for general purpose computers," *IEEE Trans. Acoust. Speech Signal Process.*, Vol. ASSP-26, pp. 141-150, 1978.

[7] D. G. M. John G. Proakis, Digital Signal Processing, New Jersey: Prentice Hall, 2007.

[8] J. E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computers*, Vol. EC-8, pp. 330-334, 1959.

Ek A

Giriş için Rastgele Sayı Üretici

Giriş sinyali olarak hem gerçek hem de sanal kısımları ± 1 arasında olan kompleks sayı Matlab'deki Gaussian dağılıma sahip varyansı bir olan rastgele sayı üretici kullanılarak üretilmiştir. Gaussian dağılım teorik olarak $\pm\infty$ arasında olduğundan ± 1 arasında çıkan değerler ret edilip sayı yeniden üretilmiştir. Bu işlem için kullanılan Matlab kodu aşağıda gösterilmiştir.

```
for i=1:length(x)
        while(real(x(i)) < -1 || real(x(i)) > 1 || imag(x(i)) < -1 || imag(x(i)) > 1 )
                x(i)=randn+1i*randn;
        end
end
```