Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi
Bilecik Seyh Edebali University Journal of Science
10(1), 76-81, 2023
DOİ: 10.35193/bseufbd.1140235
e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

Araştırma Makalesi -Research Article

Esnek Ardışık-Çıkarımlı Kutupsal Kod Çözücünün FPGA Gerçeklemesi

FPGA Implementation of Flexible Successive-Cancellation Polar Decoder

Muhammet Fatih Sertkaya[1*], Enver Çavuş[2]

Geliş / Received: 08/07/2022 ***Revize / Revised: 12/02/2023*** ***Kabul / Accepted: 19/02/2023***

ÖZET

Kanal kapasitesi, bir kanalın iletilebileceği maksimum bit hızını ifade eder. Kutupsal kodlar, simetrik ikili girişli, hafızasız kanallar için sonsuz blok uzunluğunda kanal kapasitesine erişebilen ilk hata düzeltme kodlarıdır. Kutupsal kodların bu başarımı ile 5. Nesil Yeni Radyo Haberleşme standardında kullanımına karar verilmiştir. Bu çalışmada, farklı blok uzunluklarında ve farklı kod oranlarında kutupsal kodların, ardışık-çıkarım (successive-cancellation) kod çözücü algoritması alanında programlanabilir kapı dizileri (Field-programmable gate array, FPGA) ile gerçeklemesi anlatılmıştır.

Anahtar Kelimeler- Kutupsal Kodlar, FPGA, Kutupsal Kod Çözücü

ABSTRACT

Channel capacity refers to the maximum bit rate at which a channel can be transmitted. Polar codes were the first error correcting codes to achieve infinite block length channel capacity for symmetric binary input, memoryless channels. This achievement of polar codes has been decided to be used in the 5th Generation New Radio Communication standard. In this study, the implementation of the successive-cancellation decoder algorithm of polar codes at different block lengths and different code rates with field programmable gate arrays (FPGA) is explained.

Keywords-Polar Codes, FPGA, Polar Decoder

[1]*Sorumlu yazar iletişim: 185105127@ybu.edu.tr (https://orcid.org/0000-0001-5985-2724)
Elektrik Elektronik Mühendisliği Anabilim Dalı, Ankara Yıldırım Beyazıt Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, Türkiye.
[2]İletişim: ecavus@ybu.edu.tr (https://orcid.org/0000-0002-7203-9700)
Elektrik Elektronik Mühendisliği Anabilim Dalı, Ankara Yıldırım Beyazıt Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, Türkiye.

I. GİRİŞ

E. Arıkan tarafından sunulan kutupsal hata düzeltme kodları [1], simetrik ikili-girişli ayrık hafızasız kanal altında ve sonsuz kod uzunluğunda Shannon kanal kapasitesine erişilebilmektedir. Gelişen teknolojiyle birlikte kablosuz haberleşmede daha hızlı veri alışverişine duyulan ihtiyaç, kutupsal kodların önemini artırmaktadır. Bundan dolayı "5. Nesil Yeni Radyo" standardında kutupsal kodlar kullanılacaktır [2].

Önceki nesillerden farklı olarak 5. nesil yeni radyo ile makinelerin etkin bir şekilde kendi aralarında haberleşmesi öngörülmektedir. Bu durum, sınırlı olan bant genişliğinin daha etkili kullanılmasına sebep olur. Bant genişliğinin etkin kullanılması için kutupsal kodların donanım tasarımının en etkili şekilde gerçeklenmesi gerekmektedir. Kutupsal kodların sonsuz kod uzunluğunda kanal kapasitesine ulaşabilmeleri, donanım gerçeklenmesi açısından karmaşıklığı ve gecikmeyi arttırmaktadır. Bundan dolayı daha düşük kod uzunluklarında performansı arttırmak adına çalışmalar yapılmıştır [3,4].Kutupsal kodların yazılımsal ve donanımsal olarak gerçeklemesi literatürde çeşitli yayınlar tarafından sunulmuştur [5-10].

Bu çalışmada kutupsal kodlara ait ardışık-çıkarım (successive-cancellation, SC) kod çözücüsünün kod uzunluğunun ve kod oranının esnek bir şekilde değişebildiği donanım tasarımı gerçekleştirilmiştir. Kod oranın esnek bir şekilde değiştirilmesinin hedefi, sinyal örnekleme frekansı kriteri olan Nyquist Örnekleme Kriterinden daha hızlı bir şekilde örnekleme (Faster-Than-Nyquist) [11] yapılarak haberleşmenin donanımsal olarak doğrulanması hedeflenmektedir. Bu bağlamda hem kanalın daha verimli kullanılması hem de doğrulamanın sağlıklı yapılabilmesi için donanım üzerinde kod oranının değiştirilmesi çalışma için kritiktir. Bunun yanında kod çözücünün yüksek oranda hafıza elemanına ihtiyaç duyması kaynak kullanımı açısından tasarımları zora sokmaktadır. Bunu engellemek için FPGA içinde bulunan hafıza blokları kullanılarak lojik kaynak tüketim oranını azaltmak ve hafıza bloklu ve lojik hafıza elemanlı olarak ayrı ayrı tasarımı yapılan birimlerin kaynak kullanım karşılaştırılması yapılmıştır.

Bu çalışmanın II. Bölümü'nde sırasıyla kutupsal kodların kodlamasından ve kod çözücü yapısından bahsedilmiştir. SC kutupsal kod çözücü FPGA donanım gerçeklemesi üzerinde III. Bölümde durulmaktadır. Sonuç bölümünde ise bu çalışmanın çıktılarından bahsedilmektedir.

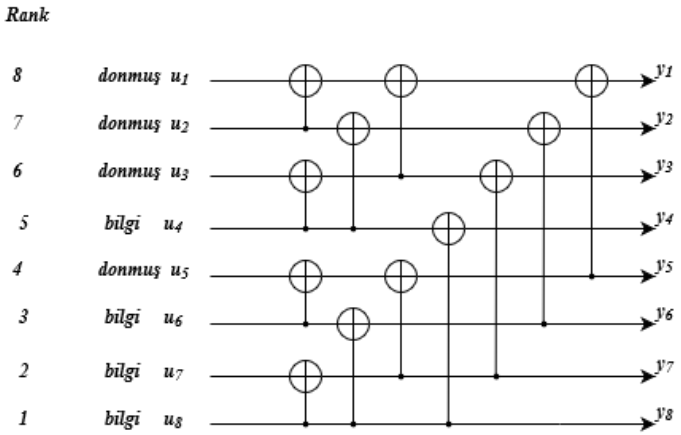
II. KUTUSAP KODLAR

A. Kutupsal Kodlayıcı

Kutupsal kodlar, asimptotik kod uzunluğunda kanal kapasitesine erişebilen hata-doğrulama kodlarıdır. Kod uzunluğu "*N"* olan kodlanmış bitlerin, kanal kutuplaşmasına göre kanaldan geçirebileceği güvenilir bilgi bit sayısı "*K"* ile gösterilir. Şekil 1'de görüldüğü gibi kanallar güvenirliliğine göre sıralanır ve kod oranına göre güvenilir kanallar seçilir. Güvenilmeyen "*N-K"* kadar donmuş bitler, kanaldan geçmez. Böylece kutupsal kodlar "*P (N, K)"*, doğrusal blok uzunluğu "$N = 2^n$*"* ve kod oranı "*R = K/N*" olarak gösterilir. Kutupsal kodların kodlamasını matris çarpması olarak ise:

$$x_0^{N-1} = u_0^{N-1} G^{\otimes n} \tag{1}$$

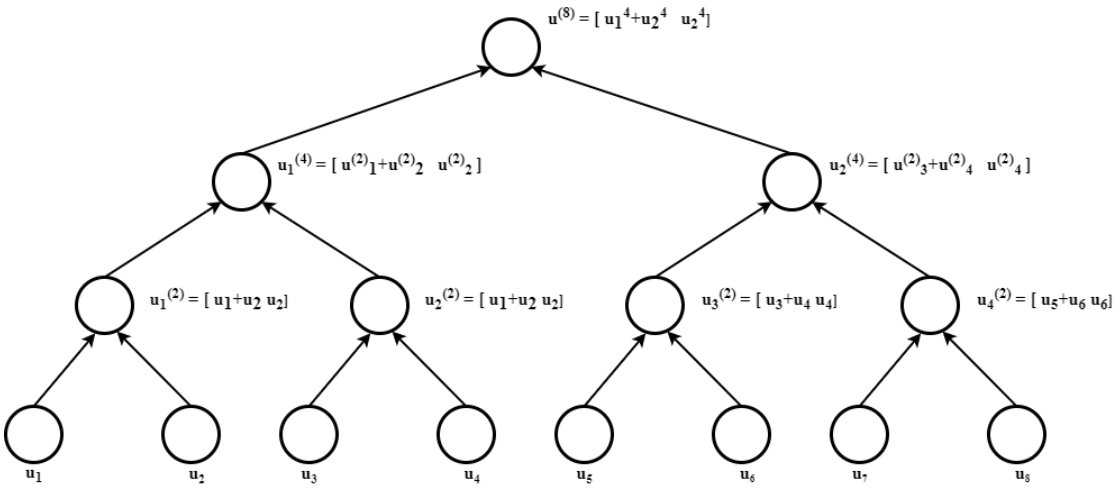
biçiminde gösterebiliriz. Burada, "$u_0^{N-1} = \{u_0, u_1, \ldots, u_{N-1}\}$" girişvektörü, "$x_0^{N-1} = x, x_1, \ldots, x_{N-1}$" kodlanmış vektör ve "$G^{\otimes n}$ " ise "$G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$" matrisinin Kronecker çarpımıdır. "N" uzunluğundaki bir kutup kodu, "*N/2"* uzunluğundaki ikili sıralı kutup kodundan oluşur. Oluşturulan kod çözücü bloğuna dair bir örnek Şekil 2'de "*N=8"*için gösterilmektedir.



Şekil 1. Kanal Güvenirlik Sıralaması, N=8

B. Ardışık-Çıkarım Kutupsal Kod Çözücü

Elde edilen "*N"* tane "**u**" kutupsal kodlu bitler, kanaldan geçerek alıcı kısmında "**y** = $\{y, y_1, \ldots, y_{N-1}\}$" olarak elde edilir. Şimdi "**y**" kodunun çözülmesi hedeflenmektedir. Bu işlem SC kod çözücüyle gerçekleştirilecektir. Tasarlanan kod çözücü mimarisi Şekil 3'te "*N=8"* için gösterilmiştir. SC kod çözücüye "**y"** verildiğinde, sırayla "u_0", sonra "u_1" in değerini şeklinde devam ederek "u_{N-1}" e kadar çıkarılır. Yani, "u_i" nin kod çözümü, "(u_0, u_1,..., u_{i-1})" olarak gösterilir. Kod uzunluğu "*N"* arttıkça ve önceki tüm bitlerin doğru çözüldüğü kabul edildiğinde, "u_i" bitinin doğru çözülme olasılığı "1"e veya "*0,5"*e yaklaşır. Başarılı kod çözme olasılığı "1"e yaklaşan bitlerin oranı, "*N"* arttıkça kanalın kapasitesine doğru eğilim gösterir. Bu polarizasyonun etkisidir. Eğer "u_i" yi tahmin etme olasılığı çok iyi değilse, o zaman değeri hem kodlayıcıda hem de kod çözücüde "0" olarak ayarlanır ve böylece "u_i" aracılığıyla hiçbir bilgi iletilmez. Bu bitlere "dondurulmuş bit" (frozen bit) denir ve herhangi bir bilgi içermez.



Şekil 2. Kutupsal Kodlayıcı İkili Ağaç Gösterimi, N=8

"N" tane olabilirlik oranı (likelihood-ratio, LLR) için kod çözücü "$n = log_2^N$" tane evreden oluşmaktadır. Her evrenin LLR değerlerinin hesaplanması kendinden önceki evrenin LLR değerine bağlıdır. LLR değerleri evreyi ikiye bölüp, her bölmeden iki "LLR" değeri alarak hesaplanır. Sıfırıncı indeksten başlayarak alınan değerler üzerinde "min-sum algoritması" uygulanmıştır. Elde edilen LLR değeri, bir sonraki evreye, evrenin ilk indisinden başlayarak yazılmıştır. Her evre sonucunda "*N/2"* adet LLR değeri elde edilmiştir. Denklem 4'te görüldüğü gibi son evrenin LLR değeri elde edildikten sonra LLR değeri için bit kararı verilmiştir. Eğer bit, dondurulmuş bit konumunda ise direkt "0" kabul edilmiştir, değilse işaretine bakılarak bit değerine karar verilmiştir. Eğer LLR değeri negatif ise bit değeri "1", pozitif ise "0" olarak karar verilmiştir. Bu işlem tasarımda "*f"* fonksiyonu olarak tanımlanmıştır ve şu şekilde ifade edilebilir:

$$L_a^n = f(L_a^{n-1}, L_b^{n-1}) \tag{2}$$

$$f(L_a, L_b) \approx işaret(L_a)\, işaret(L_b) \min(|L_a|, |L_b|) \tag{3}$$

$$u_{\hat{n}}^n = \begin{cases} 0, eğer\ \hat{n}\ donmuş\ bit; \\ 1, başka\ eğer\ L_{\hat{n}}^n < 0; \\ 0, bunun\ dışında. \end{cases} \tag{4}$$

Evrelerin diğer "LLR" değerlerinin hesabı için ikinci bir fonksiyon olan "*g"* fonksiyonu hesabı yapılmıştır. Bu fonksiyona "*f"* fonksiyonunda olduğu gibi bir önceki evrenin LLR değeriyle birlikte kendi evresindeki LLR değerinin bit sonucu da giriş olarak verilmektedir, "*g" fonksiyonunu* şu şekilde tanımlayabiliriz:

$$L_b^{n-1} = g(L_a^{n-1}, L_b^{n-1}, u_a^n) \tag{5}$$

$$g\,(L_a, L_b, \hat{u}_a) \approx L_a(1 - 2\hat{u}_a) + L_b \tag{6}$$

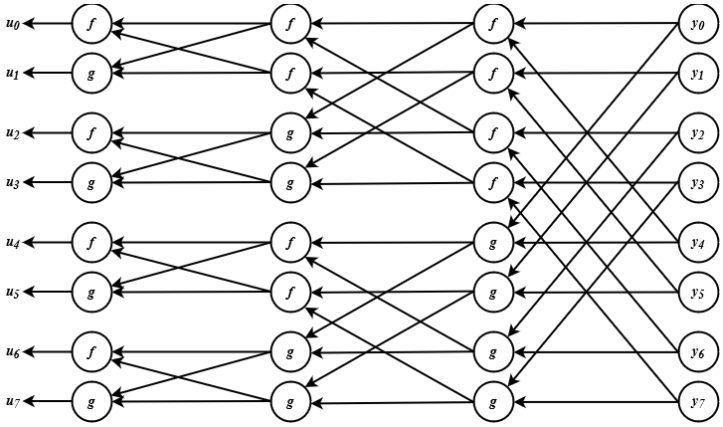
Burada, û ∈ {0,1}.

Bu işlemden sonra yukarıda bahsedilen bit kararı işlemi tekrar uygulanır ve "LLR" değerinin bit kararı verilir. Fakat bu bit kararı işlemleri, sadece son evre "LLR" değerleri için geçerlidir. Son evre dışındaki "LLR" değerlerinin bit kararları kendinden sonraki evrelerin bit değerlerine göre belirlenmektedir. Bu işlem, evrenin indeksteki ilk bit için kendinden sonraki evrenin bit değerlerinin "XOR" sonucudur, ikinci indeksteki bit değeri ise sonraki evrenin aynı indisindeki bit değerinin aynısıdır. Bu işlem şu şekilde gösterilebilir:

$$u_a^{n-1} = u_a^n \oplus u_b^n \tag{7}$$

$$u_b^{n-1} = u_b^n \tag{8}$$

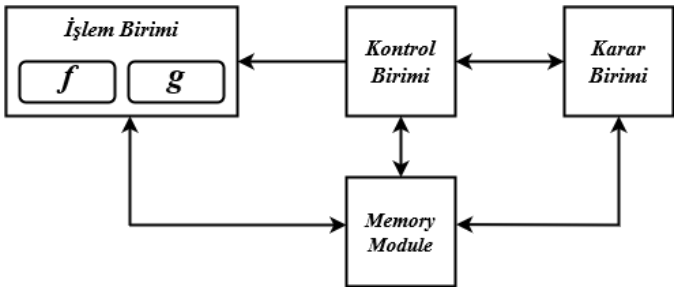
Kod çözüm işlemi "u_N^n " indeksli bit elde edilene kadar devam eder ve edildiğinde işlem son bulur.



Şekil 3. Kutupsal SC Kod Çözücü Mimarisi, N=8

III. ARDIŞIK-ÇIKARIM KUTUPSAL KOD ÇÖZÜCÜNÜN DONANIM GERÇEKLEMESİ

Bu bölümde ardışık-çıkarım kutupsal kod çözücünün FPGA donanım gerçeklemesinin detayları anlatılmıştır. Tasarım konfigürasyon, kontrol, hafıza, işlem ve karar birimlerinden oluşmaktadır. Şekil 4'te en üst seviye şematik tasarımı gösterilmiştir.



Şekil 4. Kutupsal SC Kod Çözücü Şematik Gösterimi

A. Konfigürasyon Birimi

Konfigürasyon birimi polar kod çözücünün giriş bit uzunluğunu, kod oranını ve kod uzunluğunu ayarlamak için kullanılır. Girilen parametreler doğrultusunda uygun donanımını oluşturur. Örneğin bit uzunluğu '8' girildiğinde girişleri '8' bit olarak ayarlar ve arada yapılan işlemlerde taşmayı engelleyecek şekilde bit uzunluklarını ayarlar. Çıkış "1" bit olduğu için çıkış bu durumdan etkilenmemektedir. Bunun yanı sıra kod uzunluğuna göre kullanacağı hafıza elemanlarını üretir. Her derinlik bir hafıza bloğu olduğu için "$Nlog_2^N$" adet hafıza elemanı üretir. Kod oranı ise dondurulacak bit adreslerini seçer. Tüm donmuş bit adresleri başlangıçta üretilmiştir, kod oranına göre seçimini gerçekleştirir.

B. Kontrol Birimi

Kontrol birimi, kanaldan gelen verilerin doğru adreslerle okuma erişimli hafızaya (RAM, random access memory) yazılmasını sağlar. Yazma işlemi bittiğinde veri alma işlemini durdurur ve kod çözme işlemini başlatır ve kontrol biriminin içinde bulunan sonlu durum makinesi sayesinde, her evre "s*"* için gerekli RAM okuma işlemlerini gerçekleştirir. Daha sonra okunan değerler için "f*"* veya "g*"* fonksiyonunu uygulatır ve gerekli RAM'e yazma işlemini gerçekleştirir. Fonksiyonların sonucunda gerekli veriler işlendiğinde, karar verme mekanizmasını çalıştırır ve sonucun yine uygun RAM'e yazılmasını sağlar. Son veri işlendiğinde ise tekrar veri almayı aktif hale getirir ve beklemeye geçer.

C. Hafıza Birimi

Kod uzunluğu "N*"* olan kutupsal kodlar için "$Nlog_2(N)$"tane hafıza elemanına ihtiyaç duyulmaktadır. Bu ihtiyaç "FPGA" içinde bulunan mantık elemanlarını kullanarak oluşturulabilen Distributed RAM yerine FPGA'in içinde hazır donanım olarak bulunan Block RAM'leri kullanarak giderilir. Böylece FPGA içindeki mantık hücre kaynakları (Flip-Floplar ve LUTlar), tasarruflu kullanılmış olurlar. Kullanılan RAM boyutları olabilecek maksimum veri uzunluğu ve paket uzunluğu baz alınarak belirlenmiştir. Çalışmada mesaj uzunluğu maksimum 2048, veri uzunluğu ise 16 bit işaretli olacak şekilde oluşturulmuştur. Daha kısa veri uzunluklarında iki adet yöntemle işlemler devam etmektedir: mesajın işareti dikkate alınarak fazla bitlere kesik atılması veya işaret genişlemesi yapılarak 16 bit mevcut şekilde devam edilmesidir.

D. İşlem Birimi

İşlem birimi, kutupsal kod çözücü için "f*"*, "g*"* fonksiyonlarının ve karar verme modüllerinin gerçeklemesini içerir. LLR domaininde, "f*"* ve "g*"* işlemleri çarpma ve bölme işlemleri içermektedir. Burada, "f*"* ve "g*"* işlemlerinin donanımsal kompleksliği oldukça azdır. Tasarım yapısı Şekil 4'te gösterildiği gibi dallanma şeklinde olduğu için bu işlemler daha anlaşılabilir olması içinden ebeveyn-çocuk ilişkisi şeklinde isimlendirilmiştir [12]. Burada, ebeveyn olan bir düğümden aşağı inen sağ ve sol çocuklar sırasıyla "f*"* ve "g*"* fonksiyonlarını belirtmektedir. "*Sol çocuk*" işlem biriminden iki tane veri beklemekte olup burada gelen verilerin mutlak değerini almanın yanı sıra işaretleri de dikkate almaktadır. Mutlak değerlerinin küçüğününse işaretlerin çarpım sonucuna göre dışarıya vermektedir. Burada çarpma işlemi kompleksliği arttıracağı için, işaretlere "*XOR*" işlemi uygulanmıştır ve aynı sonuç elde edilmiştir. Basit bir toplayıcı ve çıkarıcı ile gerçeklenebilen *"sağ çocuk"* işlem modülü, *"sol çocuk" modülünde* olduğu gibi iki adet veri beklemesinin yanında, bir de "*sol çocuk"* fonksiyonu işlemi sonucu elde edilen karar bitini de giriş olarak almaktadır. Bu karar biti, işlemin toplama ya da çıkarma olacağına karar vermektedir. Eğer bit "1" ise çıkarma, "0" ise toplama yapılmaktadır. Sağ ve sol çocuk işlemleri tamamlandığında ebeveyn işlemi bu iki çocuktan şu iki bilgiyi alarak değerini belirler: ilk bit için sağ ve sol çocuk bilgisinin "*XOR*" sonucu, ikinci bit için sadece sağ çocuğun bit sonucunu alır.

Sol Çocuk İşlemi :

$$f(L_a, L_b) işaret(L_a) işaret(L_b) min(|L_a|, |L_b|) \tag{9}$$

Sağ Çocuk İşlemi:

$$g(\widehat{u_s}, L_a, L_b) \;=\; L_a(-1)^{\widehat{u}_s} + L_b \tag{10}$$

Ebeveyn İşlemi:

$$\widehat{u_1} + \widehat{u_2}, \widehat{u_2} \tag{11}$$

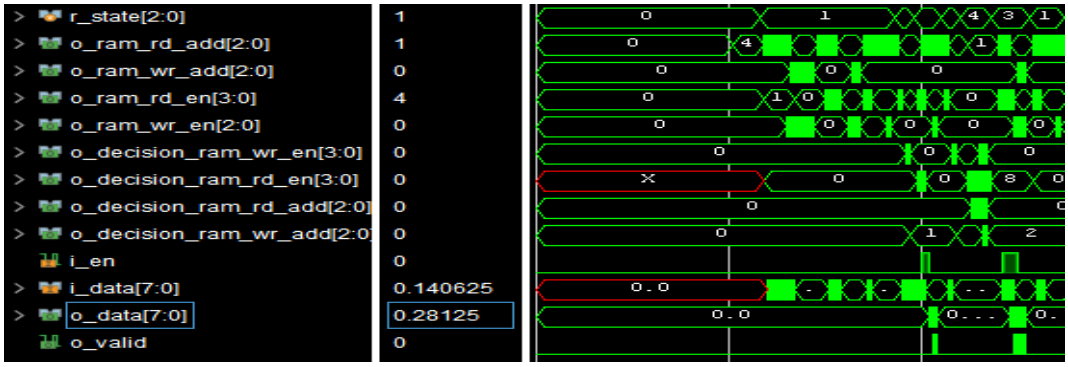
E. Karar Birimi

Karar birimi, kutupsal kod çözücü her son evreye geldiğinde elde edilen LLR değerinin kararını vermektedir. Eğer elde edilen LLR değeri donmuş bit koordinatında ise LLR değerine bakılmaksızın "0" olarak karar vermektedir. LLR değeri bilgi biti koordinatında ise değerin işaretine bakıp pozitif ise "0", negatif ise "1" olarak karar verir. Bunun yanında son evrede olmayan işlemler için her "g*" fonksiyonu* işlemi tamamlandığında daha önce açıklanan karar verme işlemini gerçeklemektedir.

IV. SONUÇ

Bu çalışmada kutupsal kodlar için "SC" kod çözücünün farklı kod uzunlukları ve oranlarında "ZYNQ-7 ZC706 (xc7z045ffg900-2) FPGA Geliştirme kartı" üzerinde çalışma frekansı "200 MHz" olacak şekilde donanımsal gerçeklemesi gerçekleştirilmiş, "N=8*"* için durum değişimleri, birimler arası bazı giriş ve çıkışlar, "RAM" kontrol sinyallerinin simülasyon görüntüsü Şekil 5'te, kaynak kullanımı Tablo 1'de gösterilmiştir. Burada "BRAM" ve lojik elemanları kullanarak iki farklı tasarımın sonuçları incelendiğinde, arada ciddi bir kaynak kullanımı görülmektedir. "FPGA" in kendi lojik kaynaklarıyla oluşturulan tasarımda, kaynakların büyük bir oranı

hafıza birimi için kullanılmıştır. Bu durum, "BRAM" tabanlı hafıza birimi kullanılmasının kaynak tüketimi açısından avantajlı olduğunu göstermektedir. Hafıza biriminin bu ciddi kaynak tüketimi, çalışma frekansının sınırlanmasında önemli bir etkiye sahiptir. Bu yüzden gelecek çalışmalarda hafıza birimi daha optimize hale getirilip, yüksek çalışma frekanslarına çıkılması planlanmaktadır.



Şekil 5. Simülasyon Görüntüsü, N=8

Tablo 1. Polar Kod Çözücünün 200 MHz Çalışma Frekansında BRAM ve Lojik Tabanlı Hafıza Elemanlarının Kaynak Kullanım Sonuçları

	Kod Uzunluğu, N							
	BRAM Tabanlı Tasarım				**Lojik Tabanlı Tasarım**			
	8	**128**	**512**	**2048**	**8**	**128**	**512**	**2048**
LUT	167	266	316	370	380	5073	24135	113780
FlipFlop	146	192	217	249	581	9932	47124	222639
BRAM	4	7	12	19	0	0	0	0

BİLGİLENDİRME

Bu çalışma TÜBİTAK 122E236 numaralı projesi kapsamında desteklenmiştir.

KAYNAKLAR

[1] Arıkan, E. (2009). Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels. *IEEE Transactions on Information Theory*, 55 (7), 996–1009.

[2] 3GPP TSG RAN WG1 Meeting #87. (2016). On the hardware implementation of channel decoders for short block lengths. Reno, Nevada, USA.

[3] Tal, I., & Vardy, A. (2015). List Decoding of Polar Codes, *IEEE Transactions on Information Theory*, 61 (5), 2213-2226.

[4] Tal, I., & Vardy, A. (2011). List Decoding of Polar Codes. International Symposium on Information Theory Proceedings, 31 July - 05 August, St. Petersburg, Russia, 1-5.

[5] Leroux, C., Tal, I., Vardy, A., & Gross, W., J. (2011). Hardware architectures for successive cancellation decoding of polar codes. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 22-27 May, Prague, Czech Republic, 1665-1668.

[6] Pamuk, A. (2011). An FPGA implementation architecture for decoding of polar codes. 8th International Symposium on Wireless Communication Systems, 06-09 November, Aachen, Germany, 1665-1668.

[7] Sarkis, G., Giard, P., Vardy, A., Thibeault, C., & Gross, W., J. (2016). Fast List Decoders for Polar Codes. *IEEE Journal on Selected Areas in Communications*, 34 (2), 318-328.

[8] Dizdar, O., & Arıkan, E. (2016). A High-Throughput Energy-Efficient Implementation of Successive Cancellation Decoder for Polar Codes Using Combinational Logic. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 63 (3), 436-447.

[9] Le Gal, B., Leroux C., & Jego C. (2017). Successive Cancellation Decoder for Very Long Polar Codes. IEEE International Workshop on Signal Processing Systems, 03-05 October, Lorient, France, 1-6.

[10] Arlı, A. Ç., Çolak, A., & Gazi O. (2017). The implementation of a successive cancellation polar decoder on Xilinx System Generator. 24th IEEE International Conference on Electronics, Circuits and Systems, 05-08 December, Batumi, Georgia, 372-376.

[11] Mazo,J. E. (1975). Faster-than-Nyquist signaling. *The Bell System Technical Journal*, 54 (8), 1451– 1462.

[12] A. Balatsoukas-Stimming & M. B. Parizi & A. Burg (2015). LLR-Based Successive Cancellation List Decoding of Polar Codes. IEEE Transactions on Signal Processing, 63(19), 5165-5179.