Gazi Üniv. Müh. Mim. Fak. Der.
Cilt 26, No 4, 823-832, 2011

J. Fac. Eng. Arch. Gazi Univ.
Vol 26, No 4, 823-832, 2011

BCJR ALGORİTMASI KULLANILAN TURBO KOD ÇÖZÜCÜLERİN FPGA GERÇEKLEŞTİRİMİ

Onur ATAR*, **Murat H. SAZLI****
*Tübitak-UZAY, 06531, ODTÜ Kampüsü, Ankara
**Ankara Üniversitesi Elektronik Mühendisliği, 06100, Tandoğan, Ankara
onur.atar@uzay.tubitak.gov.tr, **sazli@eng.ankara.edu.tr**

(Geliş/Received: 25.01.2011; Kabul/Accepted: 13.07.2011)

ÖZET

Kanal kapasite sınırına yaklaşabilmek amacıyla kullanılan kanal kodlama uygulamalarından en yenisi ve başarılısı olan turbo kodların en zorlu tasarım sorunu, kodlayıcıların bütün olası durumları için hesaplamalar yapan döngülü (iteratif) kod çözücülerin tasarımıdır. Turbo kod çözücülerde kullanılan optimal BCJR (MAP) algoritması, bölme işlemi, üstel ve logaritmik hesaplar gibi karmaşık matematiksel işlemler barındırmaktadır. Bu nedenle, turbo kod çözücülerin gerçeklenmesinde BCJR algoritmasından kaçınılmış ve onun optimal-altı (sub-optimal) türevleri olan Log-MAP ve Max-Log-MAP algoritmaları tercih edilmiştir. BCJR algoritması, önceki çalışmalarda yeniden formüle edilmiş ve FPGA gerçekleştirimine uygun bir yapıya büründürülmüştür. Bu çalışmada, yeniden formüle edilmiş BCJR algoritması gerçeklenmiştir. Donanımda yavaş çalışan karmaşık matematiksel işlemler (bölme, üstel ve logaritmik hesaplar) değer tablolarından okunmuş ve yüksek performanslı hesaplama yapıları oluşturulmuştur. Gerçeklenen sistem, benzetimler ile doğrulanmıştır. Elde edilen BER performansının beklendiği gibi Log-MAP algoritmasından yüksek olduğu gözlenmiştir.

Anahtar Kelimeler: BCJR algoritması, MAP algoritması, Turbo kod çözücü, FPGA

FPGA IMPLEMENTATION OF TURBO DECODERS USING BCJR ALGORITHM

ABSTRACT

The most difficult design issue for turbo codes, which is the most recent and successful channel coding method to approach the channel capacity limit, is the design of the iterative decoders which perform calculations for all possible states of the encoders. BCJR (MAP) algorithm, which is used for turbo decoders, embodies complex mathematical operations such as division, exponential and logarithm calculations. Therefore, BCJR algorithm was avoided and the sub-optimal derivatives of this algorithm such as Log-MAP and Max-Log-MAP were preferred for turbo decoder implementations. BCJR algorithm was reformulated and wrapped into a suitable structure for FPGA implementations at previous works [1]. Reformulated BCJR algorithm is implemented in this work. Complex mathematical operations which run slowly on hardware (division, exponential and logarithm calculations) are read from look-up-tables and high performance calculation structures are established. Implemented system is verified through simulations. It is observed that the BER performance obtained is better than the Log-MAP algorithm as expected.

Keywords: BCJR algorithm, MAP algorithm, Turbo decoder, FPGA

1. GİRİŞ (INTRODUCTION)

Enformasyon Kuramı'nın kurucusu ve en büyük öncüsü olan Claude Shannon, 1948 yılında yayınlanan makalesinde, "kanal kapasitesi sınırı" denilen yeni bir kavram tanımlamış ve veri hızının kanal kapasitesinden fazla olmaması durumunda, güvenilir bir iletişim sağlayabilecek "hata denetim kodlarının" mevcut olduğunu belirtmiştir [2]. Ancak Shannon, bu kodların nasıl oluşturulacağından bahsetmemiş ve böylece, "hata denetim kodlaması" veya "kanal kodlama" olarak bilinen yeni bir alanın doğmasını sağlamıştır. Bu alanda yapılan çalışmaların en büyük amacı, Shannon'ın belirlediği ve daha sonraları

"Shannon Sınırı" olarak anılacak olan teorik kanal kapasite sınırına yaklaşabilmektir. Shannon'ın bu öncü çalışmasından sonra birçok araştırmacı kanal kodlama alanında çalışmış ve verimli kodlama yapıları oluşturmuşlardır. Ancak sadece "turbo kodlar" bir AWGN (Additive White Gaussian Noise) kanalda Shannon sınırının çok yakınına ulaşmayı başarabilmiştir.

1993 yılında bir grup Fransız araştırmacı tarafından icat edilen ve icadı, kanal kodlama alanındaki en büyük başarı olarak kabul edilen turbo kodlar [3], düşük SNR (Signal to Noise Ratio) değerlerinde yapılan iletişimlerde düşük BER (Bit Error Rate) değerleri sağlayabilmektedir. Bugün turbo kodlama, bir 3GPP (Third Generation Partnership Project) ve CCSDS (Consultative Committee for Space Data Systems) standardıdır ve derin uzay iletişimi, radyo iletişimi ve mobil iletişim gibi uygulamalarda yaygın olarak kullanılmaktadır. Turbo kodlama tekniklerinin çeşitli uygulamalarda nasıl kullanılacağıyla ilgili araştırmalar dünya çapında halen icra edilmekte olsa da, turbo kod çözücü yapılarının gerçekleştirimleri daha ilginç bir araştırma konusu olarak dikkat çekmektedir.

Turbo kod çözücülerin FPGA (Field Programmable Gate Array) gerçekleştirimiyle ilgili iki temel kısıt vardır: karmaşıklık ve kod çözme gecikmesi. Turbo kod çözücüler, gerçekleştirimlerini imkansız kılacak kadar karmaşık değildirler. Ancak diğer hata denetim algoritmalarıyla kıyaslandıklarında çok daha karmaşık oldukları görülmektedir. Bu durum başlı başına bir gerçekleştirim zorluğudur. Turbo kod çözücülerin bir diğer özelliği ise döngülü (iteratif) kod çözme yapılarıdır. Döngü (iterasyon) sayısı arttıkça, kod çözücünün hem BER performansı hem de kod çözme gecikmesi artar. BER performansının artması istenilen, kod çözme gecikmesinin artması ise istenmeyen bir durumdur. Turbo kodlamada bilgi, paketler halinde kodlanır ve veri blokları oluşturulur. Veri blokları ne kadar uzunsa, kod çözücünün BER performansı da, kod çözme gecikmesi de o kadar yüksektir. Bu sebeple, BER performansında ciddi düşüşlere sebep olmayacak kadar kısa kod çözme gecikmeleri sağlayan, asgari karmaşıklıkta verimli turbo kod çözücülerin gerçekleştirimi oldukça zorlu bir konudur.

Verimli bir turbo kod çözücü gerçekleştirimi için öncelikle uygun bir kod çözme algoritması seçilmelidir. Orijinal turbo kod çözücülerde BCJR (Bahl, Cocke, Jelinek, Raviv) algoritması kullanılmaktadır [4]. Ancak bu algoritma çarpma, bölme ve logaritma hesaplamaları gibi karmaşık matematiksel işlemler içermektedir. Dolayısıyla mühendisler, FPGA gerçekleştirimi pratik olmayan bu orijinal algoritmayı kullanmaktan kaçınmışlar ve daha düşük BER performansı sağlayan, ancak MAP algoritmasından çok daha basit yapıları olan Log-MAP ve Max-Log-MAP algoritmalarını tercih etmişlerdir [5]. Bugüne kadar yapılan ticari ve akademik gerçekleştirimlerde, karmaşıklığı ve dolayısıyla performansı azaltılmış bu algoritmalar kullanılmıştır.

Turbo kodların icat edildiği yıllarda sahip olunan teknoloji ile BCJR algoritması gibi karmaşık bir yapıyı tek bir tümleşik devrede (IC) gerçeklemek pratik değildi. Ancak günümüz VLSI teknolojisi, sadece bir kanal kodlama algoritmasının değil, bütün bir haberleşme sisteminin dahi tek bir tümleşik devrede gerçeklenmesine olanak sağlamaktadır. Özellikle son yıllarda muazzam bir ilerleme gösteren VLSI teknolojisi ile üretilmiş FPGA'lar; hız, alan ve güç gibi çok önemli parametrelerde ciddi gelişmeler kaydetmekle kalmamış, aynı zamanda karmaşık matematiksel işlemlerin daha verimli bir şekilde gerçeklenmesini de sağlamıştır. Dolayısıyla, bugüne kadar, yüksek BER performansına rağmen gerçekleştiriminden kaçınılan BCJR algoritması, modern FPGA'larda artık pratik bir şekilde gerçeklenebilir olmuştur.

2. TURBO KODLAMA (TURBO CODING)

Bir haberleşme sisteminin en önemli kısımlarından biri, kanal kodlama kısmıdır. Kanal kodlama birimleri, gönderilecek iletiye sistematik olarak bazı artıklıklar ekleyerek, kanal boyunca oluşabilecek bit hatalarının alıcı tarafından düzeltilmesini sağlayabilmektedir. Turbo kodlama, bir kanal kodlama yapısıdır ve bu çalışmanın temelini oluşturmaktadır. Bu bölümde, turbo kodlama ile ilgili kuramsal bilgiler sunulmaktadır.

2.1. Turbo Kodlayıcı (Turbo Encoder)

Bir turbo kodlayıcı, birden fazla sistematik kodlayıcının paralel olarak birleştirilmesinden oluşur. Bu kodlayıcıların her birine "bileşen kodlayıcı/kod çözücü" adı verilir. Eğer bir turbo kodlayıcı iki adet bileşen kodlayıcı ihtiva ediyorsa, bu turbo kodlayıcıya, "iki boyutlu turbo kodlayıcı" adı verilir.
Turbo kodlar orijinal olarak RSC (Recursive Systematic Convolutional) kodlayıcılarla geliştirilmişlerdir. Klasik NSC (Non-Systematic Convolutional) kodlayıcıların yüksek SNR değerlerinde sistematik kodlayıcılara (mevcut durum için RSC kodlayıcılar) göre daha iyi BER performansı gösterdikleri bilinmektedir. Ancak düşük SNR değerlerinde bu durumun tersi gözlenmektedir [3]. Ayrıca RSC kodlayıcılar yüksek kodlama oranlı uygulamalarda SNR değerinden bağımsız olarak NSC kodlayıcılardan daha iyi BER performansı gösterebilmektedirler.

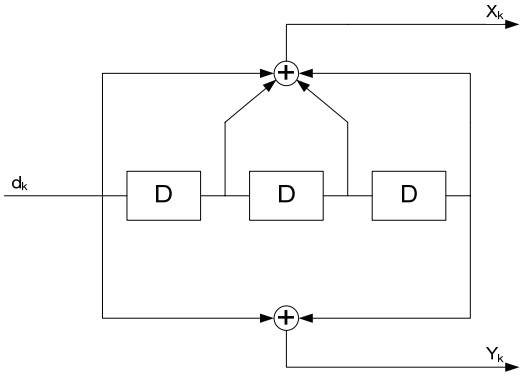
2.2. RSC Kodlayıcılar (RSC Encoders)

RSC kodlayıcıların anlaşılabilmesi için öncelikle NSC kodlayıcıların gözden geçirilmesi gerekmektedir. R=1/2 kod oranlı, cebirsel uzunluğu K ve hafızası v=K-1 olan ikili bir katlamalı kodlayıcı ele alınırsa, bu kodlayıcının k anındaki girişi olan d_k ile bu girişe karşılık gelen kod kelimesi $C_k = (X_k, Y_k)$ arasındaki ilişki aşağıdaki gibi gösterilebilir.

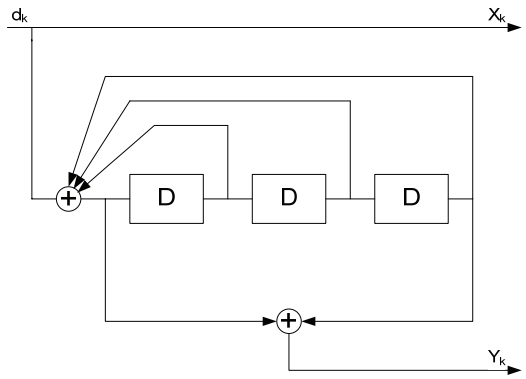
$$X_k = \sum_{i=0}^{K-1} g_{1i} d_{k-i} \quad \text{mod.2} \quad g_{1i} = 0,1 \tag{1}$$

$$Y_k = \sum_{i=0}^{K-1} g_{2i} d_{k-i} \quad \text{mod.2} \quad g_{2i} = 0,1 \tag{2}$$

$G_1 : \{g_{1i}\}$, $G_2 : \{g_{2i}\}$ genellikle oktal biçimde ifade edilen kod üreteçleridir. Şekil 1'de $G_1 = 17$, $G_2 = 11$ ve v=3 parametrelerine sahip bir NSC kodlayıcı görülmektedir.



Şekil 1. Klasik NSC kodlayıcı (Classical NSC encoder)



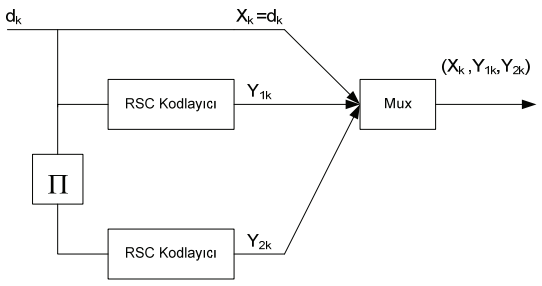
Şekil 2. R=1/2 kod oranlı bir RSC kodlayıcı (Code rate R=1/2 RSC encoder)

R=1/2 kod oranlı ikili bir RSC kodlayıcı ise, NSC kodlayıcıda bir geri besleme döngüsü oluşturulması ve iki çıkıştan birinin giriş biti d_k ile aynı yapılması sayesinde elde edilir. Bir RSC kodlayıcının hafızasının girişi ise enformasyon biti olan d_k değil, yeni bir değişken olan a_k değişkenidir. d_k ile a_k arasındaki ilişki aşağıdaki gibidir.

$$a_k = d_k + \sum_{i=1}^{K-1} \gamma_i a_{k-i} \quad \text{mod.2} \tag{3}$$

$X_k = d_k$ ise $\gamma_i = g_{1i}$, $Y_k = d_k$ ise $\gamma_i = g_{2i}$ olarak tanımlanmıştır. RSC kodlayıcılar, turbo kodlayıcıların bileşen kodlayıcılarıdır. Şekil 2'de bir RSC kodlayıcı görülmektedir.

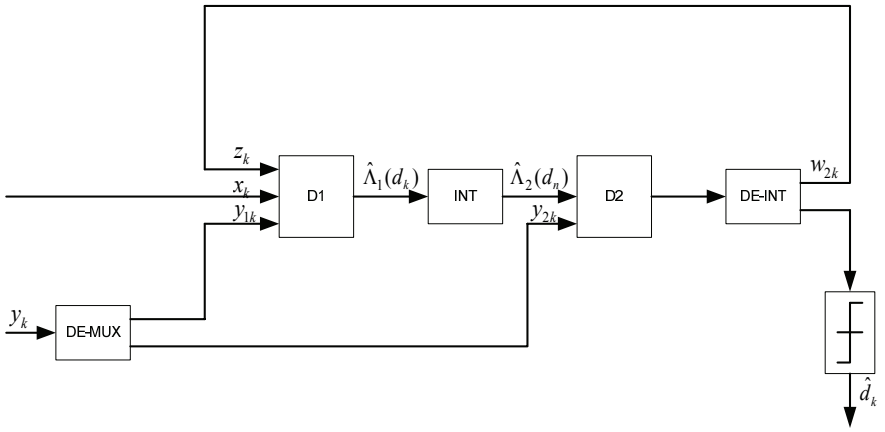
2.3. RSC Kodlayıcıların Paralel Birleşimi (Parallel Concatenation of RSC Encoders)



Şekil 3. Turbo kodlayıcı (Turbo encoder)

Şekil 3'te R=1/3 oranlı bir turbo kodlayıcı görülmektedir. Veri (d_1,d_N) bloklar halinde kodlanmakta ve blok boyutları, karıştırıcının (interleaver) blok boyutu ile belirlenmektedir. Orijinal veri dizisi öncelikle birinci RSC kodlayıcı tarafından kodlanır ve birinci parite dizisi (Y_{11},Y_{1N}) üretilir. Karıştırıcı, orijinal veri dizisini karıştırarak dizi içindeki verilerin yerlerini değiştirir. Yerleri değiştirilmiş yeni veri dizisi ise ikinci RSC kodlayıcı tarafından kodlanır. Böylece ikinci parite dizisi (Y_{21},Y_{2N}) de üretilir. Karıştırma işlemi belirli bir kurala göre olabileceği gibi tamamen rastgele de olabilir. Genellikle, karıştırıcıların uzunlukları ne kadar büyük olursa, turbo kod çözücülerin BER performansları da o kadar iyi olmaktadır. Turbo kod çözücülerde her bir bileşen kod çözücü aynı enformasyon biti üzerinde kod çözme işlemi yapar. Ancak, turbo kod çözücülerde de uygulanan karıştırma işleminden dolayı, bileşen kod çözücüler bu işlemleri farklı sıralara göre uygularlar. Her bileşen kod çözücü, kod çözme işlemi ile elde ettiği bilgiyi diğer bileşen kod çözücüye iletir ve bu süreç döngülü (iteratif) bir şekilde devam eder. Dolayısıyla, blok uzunluğu (veya karıştırıcı uzunluğu) arttıkça, orijinal veri dizisindeki herhangi bir bitin yeri ile karıştırılmış veri dizisindeki aynı bitin yeri arasındaki fark da artacaktır. Bu da, bileşen kodlayıcı tarafından o bit için üretilmiş olan parite bitlerinin daha az ilintili olmasını sağlayacak ve kod çözme işleminin başarımını arttıracaktır.

Turbo kodlayıcılar, enformasyon bitleri için üretilen her iki parite bitini de anahtarlayarak iletebilirler. Bu şekilde üretilen turbo kodlar "1/3 oranlı turbo kodlar" olarak adlandırılır. Başka bir seçenek ise, parite bitlerinin önceden belirlenmiş bir düzene göre anahtarlanmasıdır. Bu işleme "kod delme" işlemi, üretilen koda ise "delinmiş kod" denir. Örneğin, her



Şekil 4. Turbo kod çözücü (Turbo Decoder)

hangi bir zaman aralığında (X_k, Y_{1k}) çifti iletilirken, bir sonraki zaman aralığında (X_{k+1},$Y_{2(k+1)}$) çifti iletilir. Böylece her enformasyon biti için yalnızca bir parite biti üretilir. Böyle turbo kodlara "1/2 kod oranlı turbo kodlar" denir. Kod çözme işlemi de bu kurala uygun olarak yapılmalı ve her zaman aralığında bir parite biti eksik iletildiği için ilgili kod çözücü, parite biti olarak "0" değerini kullanmalıdır.

2.4. Turbo Kod Çözücü (Turbo Decoder)

Şekil 4'te görülen turbo kod çözücü iki adet bileşen kod çözücüden oluşmaktadır. Bu bileşen kod çözücüler döngülü (iteratif) bir yapıda çalışırlar ve bir kod çözücünün ürettiği tahmin diğer kod çözücü tarafından kullanılır.

Turbo kod çözücünün ilk iterasyonunda z_k'lar sıfırlanmıştır. Alınan x_k ve y_k bitleri, bileşen kod çözücülere uygun parite bitlerinin iletilmesi için anahtarlanırlar. Bileşen kod çözücülere sağlanan parite bitleri, kodlayıcıda uygulanan kod delme düzenine göre iletilirler. Eğer kod delme işlemi uygulanmamışsa, her k anında y_{1k} ve y_{2k} bitleri, sırasıyla D1 ve D2 kod çözücülerine iletilirler.

D1 kod çözücüsü x_k ve y_{1k} bitlerini kullanarak d_k biti ile ilgili bir tahminde bulunur. Elde edilen bu ilk tahminden bir bilgi parçası (harici bilgi) üretilir ve bu bilgi uygun bir şekilde karıştırıldıktan (kodlayıcıda kullanılan karıştırıcının aynısı kullanılır) sonra D2 kod çözücüsüne iletilir. D2 kod çözücüsü bu bilgi ile birlikte x_k ve y_{2k} bitlerini kullanarak d_k biti ile ilgili daha iyi bir tahmin üretir. D2 kod çözücüsünün tahmininden elde edilen harici bilgi ise bir sonraki iterasyon için D1 kod çözücüsüne iletilir. Böylece her iterasyonda daha iyi tahminler üretilmiş olur. Genellikle yirmiden daha az iterasyon ile 10^{-5} gibi bir BER değeri elde edilir.

Önceden belirlenmiş iterasyon sayısına ulaşıldığında, D2 kod çözücüsünün d_k biti ile ilgili tahmini, karar vericiye (hard limiter) iletilir. Eşik değeri sıfır olarak ayarlanmış karar verici bu tahmin bilgisini kullanarak turbo kod çözücünün d_k biti ile ilgili kesin kararını (son tahminini) üretir.

3. BCJR ALGORITMASININ YENIDEN FORMÜLASYONU (REFORMULATION OF THE BCJR ALGORITHM)

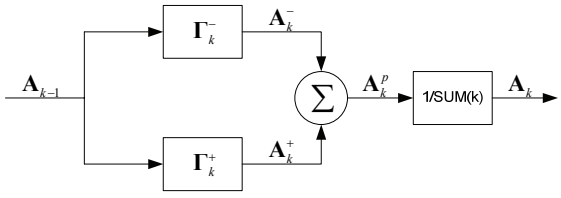
Bu bölümde, BCJR algoritmasının, bazı matris manipülasyonları ile yeniden formülasyonu sunulmaktadır [6]. Daha önce de belirtildiği gibi, çalışma kapsamında yapılan gerçekleştirimin temel aldığı yöntem, bu yeni formülasyondur.

3.1. İleri Ölçütlerin (Alfa Katsayılarının) Hesaplanması (Calculation of the Forward Metrics (Alpha Coefficients))

BCJR algoritmasının ileri ölçütleri, aşağıdaki eşitlikten yola çıkılarak yeniden formüle edilmiştir.

$$\alpha_k(m) = \frac{\sum_{m'}\sum_{i=-1}^{i=+1}\alpha_{k-1}(m')\gamma_1(R_k,m',m)}{\sum_{m}\sum_{m'}\sum_{i=-1}^{i=+1}\alpha_{k-1}(m')\gamma_1(R_k,m',m)} \quad (4)$$

Alfa katsayılarının hesaplanmasıyla ilgili yapılan yeniden formülasyon, algoritmanın gerçekleştiriminde kullanılabilecek bir yapı olarak belirtilmiştir [7]. Şekil 5'te bu yapı görülmektedir.



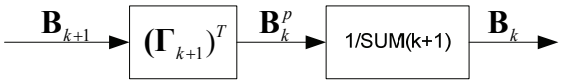
Şekil 5. Alfa katsayılarının hesaplanmasında kullanılan yapı (The structure of the alpha calculator)

3.2. Geri Ölçütlerin (Beta Katsayılarının) Hesaplanması (Calculation of the Backward Metrics (Beta Coefficients))

Geri ölçütlerin hesaplanmasında da, ileri ölçütlerde yapılan yeniden formülasyona benzer bir formülasyon kullanılmaktadır [7].

$$\beta_k(m) = \frac{\sum_{m'}\sum_{i=-1}^{i=+1} \beta_{k+1}(m')\gamma_1(R_{k+1},m,m')}{\sum_{m}\sum_{m'}\sum_{i=-1}^{i=+1} \alpha_k(m)\gamma_1(R_{k+1},m,m')} \quad (5)$$

Beta katsayılarının hesaplanmasıyla ilgili yapılan yeniden formülasyon, algoritmanın gerçekleştiriminde kullanılabilecek bir yapı olarak da belirtilmiştir [7]. Şekil 6'da bu yapı görülmektedir.



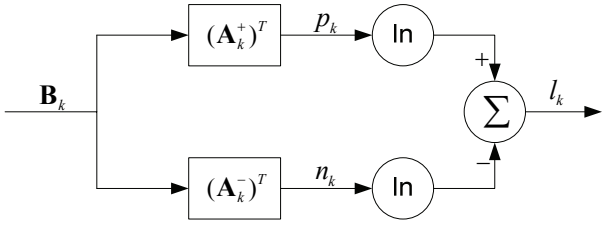
Şekil 6. Beta katsayılarının hesaplanmasında kullanılan yapı (The structure of the beta calculator)

3.3. Logaritmik Benzerlik Oranlarının (LLR) Hesaplanması (Calculation of the Logarithmic Likelihood Ratios (LLR))

Aşağıdaki eşitlik, daha önce tanımlanmış olan $\mathbf{A}$, $\mathbf{B}$ ve $\mathbf{\Gamma}$ matrisleri kullanılarak her d_k biti için LLR (Logarithmic Likelihood Ratio) değerlerinin nasıl hesaplandığını göstermektedir. Her d_k biti ile ilişkili LLR değerleri aşağıdaki gibi hesaplanmaktadır.

$$\Lambda(d_k) = \ln \frac{\sum_{m}\sum_{m'} \gamma_{+1}(R_k,m',m)\alpha_{k-1}(m')\beta_k(m)}{\sum_{m}\sum_{m'} \gamma_{-1}(R_k,m',m)\alpha_{k-1}(m')\beta_k(m)} \quad (6)$$

LLR değerlerinin hesaplanmasıyla ilgili yapılan yeniden formülasyon, algoritmanın gerçekleştiriminde kullanılabilecek bir yapı olarak da belirtilmiştir [7]. Şekil 7'de bu yapı görülmektedir.



Şekil 7. LLR değerlerinin hesaplanmasında kullanılan yapı (The structure of the LLR calculator)

4. BCJR ALGORİTMASI KULLANILAN TURBO KOD ÇÖZÜCÜLERİN FPGA GERÇEKLEŞTİRİMİ (FPGA IMPLEMENTATION OF TURBO DECODERS USING BCJR ALGORITHM)

Turbo kod çözücüler, birçok parametre ile yapılandırılan sistemlerdir. Bu parametrelerin bazılarının seçimi tasarımcının tercihine kalmış, bazıları ise 3GPP standardı tarafından belirlenmiştir [8]. Tablo 1'de gerçekleştirim parametreleri ve bu parametrelerin seçilen değerleri görülmektedir.

3GPP standardı, bir turbo kod çözücü yapısının, R=1/3 kod oranlı ve K=4 cebirsel uzunluğuna sahip olması gerektiğini belirtmiştir. Bu nedenle, bu çalışmada söz konusu parametreler, 3GPP standardının belirttiği şekilde seçilmiştir.

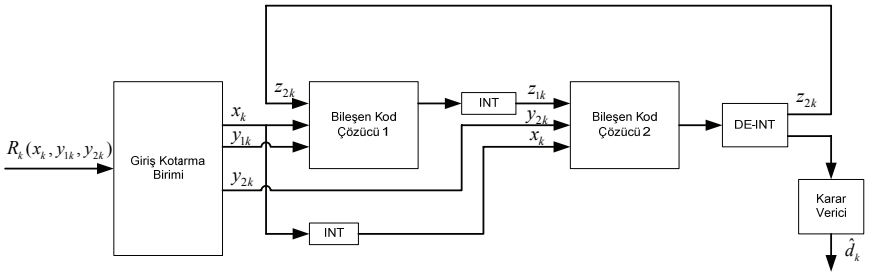
Tablo 1. Tasarım parametreleri (Design Parameters)

Parametre	Değer
Blok Uzunluğu	128-4096
RSC Kodlayıcı Sayısı	2
RSC Kodlayıcıların Cebirsel Uzunluğu	4
RSC Kodlayıcıların Üreteç Matrisi	G={7,5}(oktal)
Kodlama Oranı	R=1/3
Kod Delme	Yok
Kod Çözme Algoritması	BCJR (MAP)
Döngü Sayısı	1-20

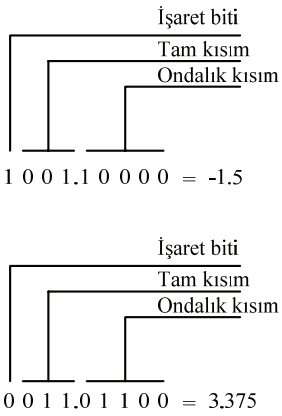
4.1. Sistemin Genel Donanım Yapısı (General Hardware Structure of the System)

Şekil 8'de, gerçeklenen turbo kod çözücünün genel yapısı görülmektedir. $R_k(x_k, y_{1k}, y_{2k})$ giriş dizisi, depolanmak ve bileşen kod çözücülere uygun bir şekilde sağlanmak üzere giriş kotarma birimine iletilir. Giriş kotarma birimi, sistematik x_k bitleri ile y_{1k} ve y_{2k} parite bitlerinin ayrı hafızalarda depolandığı ve gerekli olduğu takdirde bu hafızalardan okunduğu birimdir. Ayrıca detaylandırılmasına gerek duyulmamıştır.

BCJR algoritmasında kullanılan katsayılar ve kanaldan gürültülü bir şekilde gelen veri dizisi, gerçel sayılardır. Dolayısıyla bu sayılar, donanımda sabit-noktalı (fixed-point) olarak ifade edilmektedir. Hem kanaldan gelen veri bitleri için, hem de dahili katsayılar için dokuz bitlik sayılar kullanılmıştır. Bu sayıların kullanımı aşağıdaki gibi detaylandırılabilir.



Şekil 8. Turbo kod çözücünün genel donanım yapısı (General hardware structure of the turbo decoder)



Şekil 9. Gerçel sayıların gösterimi (Representation of real numbers)

Şekil 9'da da gösterildiği gibi, dokuz bitlik sayıların en soldaki biti işaret biti olarak kullanılmaktadır. En soldaki bit "1" ise söz konusu sayı negatif, "0" ise pozitiftir. Bu bitten sonraki üç bit ile sayının tam kısmı gösterilir. Noktadan sonraki beş bit ise, sayının ondalık kısmını göstermek amacıyla kullanılır.

4.2. INT/DE-INT Birimleri (INT/DE-INT Units)

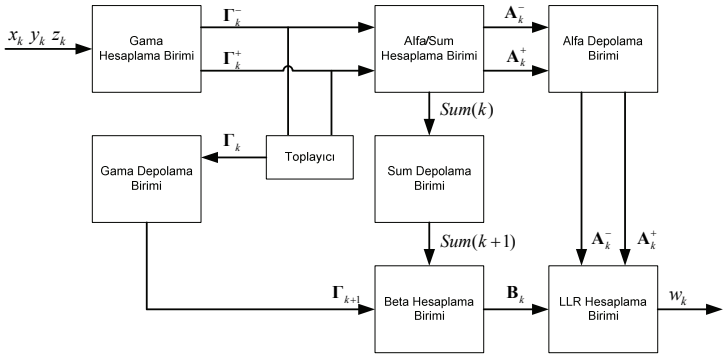
Karıştırıcı birimi (INT), bir veri dizisindeki bitlerin yerlerini değiştirmek için, ters karıştırıcı birimi (DE-INT) ise karıştırıcının işlevinin tersini gerçekleştirmek için kullanılmaktadır. Başka bir deyişle, herhangi bir bitin indis değeri, karıştırma işlevi ile başka bir değere dönüştürülür. Ters karıştırma işlevi vasıtasıyla ise, değeri değiştirilmiş indis, eski değerine kavuşur. Gerçekleştirimde kullanılan INT/DE-INT birimlerinin "indis değeri değiştirme işlevleri", 3GPP belirtiminden alınmıştır. Hem karıştırma, hem de ters karıştırma işlevlerinin, her hangi bir indis değeri için üreteceği değerler önceden bilinmektedir. Dolayısıyla, bu fonksiyonlar için değer tabloları oluşturulmuş ve indis değerleri bu tablodan okunarak değiştirilmiştir. İndis değeri değiştirme işlevleri her blok uzunluğu (karıştırıcı uzunluğu) için farklı değerler üretmektedir. Dolayısıyla, tanımlı her blok uzunluğu için (128, 256, 512, 1024, 2048, 4096) iki adet (bir adet karıştırıcı, bir adet ters karıştırıcı için) değer tablosu oluşturulmuştur. Bu tablolar, FPGA'da hafıza birimleri olarak gerçeklenmiştir ve blok uzunluğu kadar hafıza kaplamaktadır.

4.3. Bileşen Kod Çözücüler (Component Decoders)

Daha önce de belirtildiği gibi, bileşen kod çözücüler, yeniden formüle edilmiş BCJR algoritmasını kullanmaktadırlar [6]. Şekil 10'da bu algoritmanın donanımdaki yapısı gösterilmektedir.

Kanaldan alınan veri dizisi, turbo kod çözücünün giriş kotarma birimi tarafından uygun bir şekilde bileşen kod çözücülere iletilir. Bunun yanında her bileşen kod çözücü, diğer bileşen kod çözücü tarafından üretilen harici bilgileri (z_k) de kullanmaktadır.

Bileşen kod çözücülerde öncelikle, gama hesaplama birimleri vasıtasıyla durum geçiş ölçütleri olan gama katsayıları hesaplanır. Gama katsayıları hesaplanıp gama depolama birimine aktarılırken, aynı zamanda



Şekil 10. Bileşen kod çözücülerin donanım yapısı (Hardware structure of the component decoders)

alfa/sum hesaplama birimine iletilirler. Bu sayede, alfa katsayıları ve Sum(k) değerleri de hesaplanmaya başlar. Beta katsayılarının hesaplanmasının başlayabilmesi için, bütün veri dizisi için alfa katsayıları hesaplanmış olmalıdır. Alfa katsayılarının hesaplanması bittiğinde, sum depolama birimi ve gama depolama biriminden okuma yapılır ve sırasıyla Sum(k+1) değerleri ve $\mathbf{\Gamma}_{k+1}$ matrisi beta hesaplama birimine iletilir. Beta katsayıları beta hesaplama birimi tarafından hesaplanıp LLR hesaplama birimine iletilirken, aynı zamanda alfa depolama biriminden okuma yapılarak ilgili alfa katsayıları da LLR hesaplama birimine iletilir. LLR hesaplama birimi ise $\mathbf{A}_k^+$, $\mathbf{A}_k^-$ ve $\mathbf{B}_k$ matrislerini kullanarak, d_k biti ile ilgili bir tahminde bulunur ve ayrıca w_k değerlerini hesaplayarak diğer bileşen kod çözücüye iletir. Böylece bir döngü (iterasyon) tamamlanmış olur.

4.4. Gama Hesaplama Birimi (Gamma Calculation Unit)

Durum geçiş ölçütlerini hesaplamak amacıyla kullanılan eşitlik aşağıdaki gibidir.

$$\gamma_i(R_k, m', m) = \frac{1}{2\pi\sigma^2}\frac{\exp(z_k/2)}{1+\exp(z_k)}\exp(iz_k/2) \cdot q(d_k = i \mid S_k = m, S_{k-1} = m') \cdot \exp\{\frac{-1}{2\sigma^2}[(x_k - i)^2 + (y_k - Y_k)^2]\} \quad (7)$$

Durum geçiş ölçütlerinin hesaplandığı yukarıdaki eşitlik, üç ifadenin çarpımı biçimindedir. Bu nedenle, bu eşitlikteki üç çarpan, üç ayrı terim olarak ele alınabilir. $\frac{1}{2\pi\sigma^2}\frac{\exp(z_k/2)}{1+\exp(z_k)}\exp(iz_k/2)$ ifadesi birinci terim, $q(d_k = i \mid S_k = m, S_{k-1} = m')$ ifadesi ikinci terim, $\exp\{\frac{-1}{2\sigma^2}[(x_k - i)^2 + (y_k - Y_k)^2]\}$ ifadesi ise üçüncü terim olsun. Bu üç terim birbirleriyle çarpılmış ve durum geçiş ölçütleri sayısal elemanlar vasıtasıyla hesaplanmıştır.

Birinci terimde bulunan $\frac{1}{2\pi\sigma^2}$ ifadesi sabit bir sayıdır ve bir AWGN kanalın SNR bilgileri kullanılarak bir tablodan okunur. Dolayısıyla donanım üzerinde bir işlem yüküne sebep olmamaktadır. Bu sabit sayı ile çarpılan $\frac{\exp(z_k/2)}{1+\exp(z_k)}\exp(iz_k/2)$ ifadesi ise iki farklı tablodan okunabilir. Söz konusu geçiş ölçütü $\mathbf{\Gamma}_k^-$ matrisi için hesaplanıyorsa $i = -1$, $\mathbf{\Gamma}_k^+$ matrisi için hesaplanıyorsa $i = +1$ alınır. Bu nedenle, bu ifade iki farklı durum için tablolardan okunarak $\frac{1}{2\pi\sigma^2}$ sabiti ile çarpılır ve birinci terim hesaplanmış olur. Birinci terimin hesaplanmasında sadece çarpma işlemi kullanılmakta ve exponansiyel hesabı için tablolardan yararlanılmaktadır.

Daha önce de belirtildiği gibi, ikinci terim olan $q(d_k = i \mid S_k = m, S_{k-1} = m')$ terimi de "0" ya da "1" değerlerini alabilir. Bu ifadenin alacağı değerler de önceden belirlenmiştir. Bu sayede, $q(d_k = i \mid S_k = m, S_{k-1} = m')$ teriminin hesaplanmasında da sadece tablolardan yararlanılmaktadır.

Üçüncü terim olan $\exp\{\frac{-1}{2\sigma^2}[(x_k - i)^2 + (y_k - Y_k)^2]\}$ teriminde bulunan $\frac{-1}{2\sigma^2}$ ifadesi de sabit bir sayıdır ve bu ifade de SNR bilgileri kullanılarak bir tablodan okunur. $[(x_k - i)^2 + (y_k - Y_k)^2]$ ifadesindeki Y_k değeri, turbo kodlayıcının $q(d_k = i \mid S_k = m, S_{k-1} = m') = 1$ iken ürettiği parite bitidir ve daha önce de belirtildiği gibi "1" ya da "-1" değerlerini alabilir. Dolayısıyla Y_k değeri de bir tablodan okunarak elde edilir. $[(x_k - i)^2 + (y_k - Y_k)^2]$ ifadesi için yapılan işlemler, çarpma ve toplama işlemleridir. Bu ifadedeki exponansiyel de tablodan okuma yöntemi ile hesaplanmıştır.

Bu üç terimin hesaplanarak birbiriyle çarpılması ile k anındaki durum geçiş ölçütleri hesaplanmış olur. Gama katsayılarının hesabında çarpma ve toplama işlemleri kullanılmıştır. Tablodan yapılan okumalar ise, donanımda hafızadan yapılan okumalar olarak gerçeklenmiştir.

4.5. Alfa ve Beta Hesaplama Birimleri (Alpha and Beta Calculation Units)

Alfa ve beta hesaplama birimleri, BCJR algoritmasının yeniden formülasyonunda da belirtildiği gibi [6], sırasıyla Şekil 5 ve Şekil 6'daki yapılar kullanılarak gerçeklenmiştir.

Bu yapılarda, çarpma ve toplama işlemleri dışında, bölme işlemleri de yapılmaktadır. Donanımda gerçeklenen bölme işlemi, günümüzün yüksek hızlı FPGA'larına rağmen oldukça yavaş çalışmaktadır. Bu nedenle bölme işlemi de, hız kazanmak amacıyla tablolar kullanılarak gerçeklenmiştir. Bölme işleminin tablolar vasıtasıyla hesaplanması aşağıdaki gibidir.

N_1 ve N_2 iki gerçel sayı olsun. Yapılmak istenen işlem ise $\frac{N_1}{N_2}$ olsun. $\frac{N_1}{N_2}$ işlemi, $N_1.\frac{1}{N_2}$ şeklinde de ifade edilebilir. Daha önce de belirtildiği gibi, gerçekleştirimde kullanılan bütün gerçel sayılar, dokuz bitlik sayılar ile gösterilmektedir. Alfa ve beta katsayıları pozitif sayılar olduğu için, işaret bitleri ihmal edilebilir. Bu durumda N_1 ve N_2 sayıları sekiz bitlik sayılar haline gelirler. Dolayısıyla, $N_1.\frac{1}{N_2}$

Tablo 2. R=1/3 oranlı BCJR turbo kod çözücü ile Log-MAP turbo kod çözücünün, 5 iterasyon sonucu elde edilen BER performanslarının karşılaştırılması (BER performance comparison of R=1/3 BCJR turbo decoder and the Log-MAP turbo decoder after 5 iterations)

Blok Uzunluğu	512 Bit		1024 Bit		2048 Bit	
SNR	BCJR	Log-MAP	BCJR	Log-MAP	BCJR	Log-MAP
SNR = 0.5 dB	3.10^{-3}	4.10^{-2}	4.10^{-4}	3.10^{-2}	10^{-4}	2.10^{-2}
SNR = 1 dB	8.10^{-4}	10^{-2}	2.10^{-5}	2.10^{-3}	6.10^{-6}	6.10^{-4}
SNR = 1.5 dB	2.10^{-5}	10^{-3}	10^{-6}	4.10^{-5}	4.10^{-7}	2.10^{-6}
SNR = 2 dB	2.10^{-6}	2.10^{-4}	$<10^{-7}$	10^{-6}	$<10^{-7}$	10^{-7}

Tablo 3. R=1/3 oranlı BCJR turbo kod çözücü ile Log-MAP turbo kod çözücünün, sırasıyla 139 MHz ve 349 MHz çalışma frekansı ile elde edilen toplam veri hızı performanslarının karşılaştırılması (Total throughput performance comparison of R=1/3 BCJR turbo decoder and the Log-MAP turbo decoder at 139 MHz and 349 MHz clock frequency respectively) (Mbit/s)

Blok Uzunluğu	512 Bit		1024 Bit		2048 Bit	
Döngü Sayısı	BCJR	Log-MAP	BCJR	Log-MAP	BCJR	Log-MAP
Döngü = 3	7,87	32,93	7,88	35,58	7,90	37,96
Döngü = 5	4,69	20,68	4,72	22,42	4,74	24,00
Döngü = 7	3,35	15,08	3,36	16,37	3,38	17,54
Döngü = 9	2,61	11,86	2,62	12,89	2,63	13,82

işleminin sonucunun, klasik bölme işlemi kullanılmadan gerçeklenebilmesi için, $\frac{1}{N_2}$ işleminin sonucunun bir tablodan okunması ve okunan değerin N_1 sayısı ile çarpılması gerekmektedir. Söz konusu tabloda 2^8 tane farklı değer vardır ve bu değerler donanım üzerinde bir hafızada tutulurlar. Sekiz bitlik sayılar söz konusu olduğu için, bölme işleminin bu şekilde gerçeklenmesi, kaplanan hafıza alanının azlığı açısından oldukça verimli bir yöntemdir.

4.6. LLR Hesaplama Birimi (LLR Calculation Units)

Şekil 7'de de görüldüğü gibi, LLR hesaplama biriminde de ağırlıklı olarak çarpma ve toplama işlemleri yapılmaktadır. Sekiz bitlik iki sayının çarpımı 16 bitlik bir sonuç vermektedir. Gama, alfa ve beta hesaplama birimlerinde, çarpım sonuçları olan bu 16 bitlik sayılar, sekiz bitlik sayılar olarak nicemlenmiştir. Ancak LLR hesaplama biriminde daha hassas sonuçlar elde edilmesi amacıyla, çarpım sonuçları 10 bitlik sayılar olarak nicemlenmiştir. Söz konusu birimde bu işlemler dışında, donanımda karmaşıklık yaratacak tek işlem logaritma hesabıdır. Şekil 7'deki yapıya göre gerçeklenen LLR hesaplama biriminde, p_k ve n_k sayılarının doğal logaritmaları, tablolardan okunmuş ve $\ln(p_k)-\ln(n_k)$ işleminin sonunu hesaplanmıştır. p_k ve n_k sayıları 10 bitlik sayılar olduğu için, bu sayıların doğal logaritma değerleri 2^{10} elemanlı bir tablodan okunmaktadır. Bu tablo, donanım üzerinde bir hafızada tutulur. Her bileşen kod çözücüde toplam iki logaritma işlemi olduğu için, 2^{10} elemanlı dört tablo kullanılarak bir logaritma hesaplayıcı gerçeklenmiş ve böylece karmaşık logaritma hesabından kaçınılmıştır.

4.7. BCJR Turbo Kod Çözücünün BER Performansı (BER Performance of the BCJR Turbo Decoder)

BCJR algoritmasının, kuramsal olarak Log-MAP ve Max-Log-MAP algoritmasından daha yüksek BER performansı gösterdiği bilinmektedir. Bu çalışmada gerçeklenen BCJR turbo kod çözücünün, Xilinx firmasına ait Log-MAP turbo kod çözücüden [9] daha yüksek bir performans gösterdiği kanıtlanmıştır. Tablo 2'de çalışma kapsamında gerçeklenen BCJR turbo kod çözücü ile Xilinx firmasına ait Log-MAP turbo kod çözücünün BER performanslarının karşılaştırılması görülmektedir. Çalışma kapsamında gerçeklenen BCJR turbo kod çözücünün, Xilinx firmasına ait Log-MAP turbo kod çözücüden daha yüksek BER performansı gösterdiği gözlenmiştir.

Tablo 2'de de görülebileceği gibi, BCJR turbo kod çözücü ile Log-MAP turbo kod çözücülerin BER performansları, SNR değeri arttıkça birbirine yaklaşmaktadır. Bunun sebebi, BCJR turbo kod çözücünün düşük SNR değerlerinde daha yüksek performans göstermesidir. SNR değeri arttıkça hata olasılığı da azalmaktadır. Bu durum, Log-MAP kod çözücünün de performansını arttırmaktadır.

4.8. BCJR Turbo Kod Çözücünün Toplam Veri Hızı Performansı (Total Throughput Performance of the BCJR Turbo Decoder)

Daha önce de belirtildiği gibi, turbo kod çözme işlemi, döngülü bir işlemdir. Her veri bloğu için birden fazla (pratikte altıdan fazla) döngü gerçekleştirilir. Bu sayede turbo kod çözücülerin BER performansları artmaktadır. Ancak bu döngülü yapı toplam veri hızının azalmasına sebep olmaktadır. Tablo 3'te çalışma kapsamında gerçeklenen BCJR

turbo kod çözücü ile Xilinx firmasına ait Log-MAP turbo kod çözücünün toplam veri hızı performanslarının karşılaştırılması görülmektedir.

Çalışma kapsamında gerçeklenen BCJR turbo kod çözücünün toplam veri hızı performansının, Xilinx firmasına ait Log-MAP turbo kod çözücünün performansından düşük olduğu gözlenmiştir. Bu performans düşüklüğünün sebebi, BCJR turbo kod çözücünün azami çalışma frekansının 139 MHz, Log-MAP turbo kod çözücünün azami çalışma frekansının ise 349 MHz olmasıdır. Çarpma işlemi gibi karmaşık matematiksel işlemlere sahip olan BCJR turbo kod çözücü, sadece toplama ve çıkarma işlemi yapılan Log-MAP turbo kod çözücüden daha karmaşık bir donanım yapısı sahip olmakta ve dolayısıyla daha yavaş çalışmaktadır. Ancak toplam veri hızındaki bu performans düşüklüğü, yüksek BER performansı adına ödenebilecek bir bedeldir.

4.9. BCJR Turbo Kod Çözücünün Gerçekleştirim Raporu (Implementation Report of the BCJR Turbo Decoder)

Turbo kod çözücülerde kullanılan kod çözme algoritmaları, karmaşık ve işlem yükü yüksek algoritmalardır. Çarpma ve bölme gibi matematiksel işlemlerin donanımda gerçeklenmesi, donanım elemanlarının sarfiyatını önemli ölçüde arttırmaktadır. Bunun yanında, söz konusu işlemler donanımda yavaş çalışmakta ve algoritmanın toplam veri hızının azalmasına sebep olmaktadır. Log-MAP ve Max-Log-MAP gibi optimal-altı algoritmaların tercih edilmelerinin sebebi de bu işlemlerin, logaritmik bölgede toplama ve çıkarma işlemleri olarak tanımlanmasıdır. Toplama ve çıkarma işlemleri, çarpma ve bölme işlemlerine göre daha az donanım elemanı sarf etmekte ve daha hızlı çalışmaktadır. Ancak optimal BCJR algoritmasının bu anlamda ciddi bir dezavantajı bulunmaktadır. BCJR algoritmasında karmaşık matematiksel işlemler vardır ve bunun sonucunda bu algoritma daha fazla donanım sarf etmekte ve daha yavaş çalışmaktadır. Çalışma kapsamında bu dezavantajın etkisi, çarpma işlemi dışındaki karmaşık işlemlerin donanım elemanları ile değil, değer tabloları, yani hafıza elemanları ile gerçeklenmesi sayesinde azaltılmıştır. Tablo 4 ve Tablo 5'te sırasıyla çalışma kapsamında gerçeklenen BCJR turbo kod çözücü ile Xilinx firmasına ait Log-MAP turbo kod çözücünün gerçekleştirim raporları görülmektedir.

Tablo 4 ve Tablo 5'te de görüldüğü gibi, Xilinx firmasına ait Log-MAP turbo kod çözücünün, donanım elemanları sarfiyatı ve azami çalışma frekansı açısından BCJR turbo kod çözücüden daha üstün olduğu görülmüştür. Ancak bu dezavantaj da, optimal BCJR algoritmasının kullanılabilmesi, dolayısıyla yüksek BER performansı adına ödenebilecek bir bedeldir.

Tablo 4. R=1/3 oranlı BCJR turbo kod çözücünün gerçekleştirim raporu (Implementation report of the R=1/3 rate BCJR turbo decoder)

Seçenekler		
Xilinx FPGA	XC6VLX75T	
LUT/FF Çifti	2264	
Dilim LUT	36254	
Dilim Yazmacı	2404	
Blok RAM (36k)	142	
Blok RAM (18k)	0	
DSP Bloğu	0	
Hız Derecesi	-1	-3
Azami Çalışma Frekansı	130 MHz	139 MHz

Tablo 5. R=1/3 oranlı Log-MAP turbo kod çözücünün gerçekleştirim raporu (Implementation report of the R=1/3 rate Log-MAP turbo decoder)

Seçenekler		
Xilinx FPGA	XC6VLX75T	
LUT/FF Çifti	3765	
Dilim LUT	3712	
Dilim Yazmacı	4062	
Blok RAM (36k)	6	
Blok RAM (18k)	7	
DSP Bloğu	0	
Hız Derecesi	-1	-3
Azami Çalışma Frekansı	285 MHz	349 MHz

5. SONUÇ (CONCLUSION)

Önceki çalışmalarda temelleri atılan [6] turbo kod çözme yapıları, bu çalışmada gerçeklenmiş ve gerçeklenen sistemin, kuramsal kavramlarla tutarlı bir şekilde çalıştığı görülmüştür. Girişte de belirtildiği gibi, gerçeklenen BCJR turbo kod çözücü, ticari bir ürün olan Log-MAP turbo kod çözücü [9] ile kıyaslanmıştır. Bu kıyaslama sonucunda, gerçeklenen BCJR turbo kod çözücünün beklendiği gibi daha yüksek BER performansı gösterdiği gözlenmiştir.

Turbo kodların icat edildiği günden bu yana, kuramsal olarak yüksek BER performansı gösteren BCJR algoritmasının gerçekleştirimi, geçmiş teknolojiler ile bu algoritmanın pratik bir gerçekleştiriminin olmaması sebebiyle hep kaçınılan bir çalışma olmuştur. Ancak günümüz teknolojisi, önemsiz bedeller ödeyerek (önemsiz derecede azalmış toplam veri hızı ve hafıza elemanı sarfiyatı), bu algoritmanın pratik bir şekilde gerçeklenmesine olanak sağlamaktadır.

Bu çalışmada [10]; geleneksel, optimal turbo kod çözme algoritması olan BCJR algoritması kullanılan bir turbo kod çözücünün FPGA gerçekleştiriminin, günümüz VLSI teknolojisi de göz önüne alındığında Log-MAP algoritmasını kullanan turbo kod çözücülere göre daha avantajlı olduğu kanıtlanmıştır.

SEMBOLLER (NOMENCLATURE)

R: RSC kodlayıcının kodlama oranı
K: RSC kodlayıcının cebirsel uzunluğu
v: RSC kodlayıcının hafızası
G: Kod üreteci
d_k: Sistematik bit
C_k: RSC kodlayıcının ürettiği kod kelimesi
X_k: RSC kodlayıcının ürettiği sistematik bit
Y_k: RSC kodlayıcının ürettiği parite biti
R_k: Turbo kod çözücünün aldığı gürültülü kod kelimesi
x_k: Turbo kod çözücünün aldığı gürültülü sistematik bit
y_k: Turbo kod çözücünün aldığı gürültülü parite biti
z_k: Bileşen kod çözücülerin ürettiği harici bilgi
α_k : İleri ölçütler (alfa katsayıları)
β_k : Geri ölçütler (beta katsayıları)
γ_k : Durum geçiş ölçütleri (gama katsayıları)
$\Lambda(d_k)$: İlgili sistematik bitin logaritmik benzerlik oranı
l_k : İlgili sistematik bitin logaritmik benzerlik oranı
m : RSC kodlayıcının mevcut durumu
m' : RSC kodlayıcının önceki durumu

KAYNAKLAR (REFERENCES)

1. Sazlı, M., H., **Neural Network Applications to Turbo Decoding**, Doktora Tezi, Syracuse Universitesi, 2003.
2. Shannon, C. E., "A Mathematical Theory of Communications", **Bell System Technical Journal**, Vol. 27, pp.379-423, 623-656, 1948.
3. Berrou, C., Glavieux, A., Thitimajshima, P., "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes", **Proceedings of IEEE International Conference on Communication**, pp. 1064-1070, 1993.
4. Bahl, L. R., Cocke, J., Jelinek, F., Raviv, J., "Optimal Decoding of Linear Codes for Minimizing The Symbol Error Rate", **IEEE Transactions on Information Theory**, Vol. 20, pp. 284-287, 1974.
5. Robertson, P., Hoeher P., "Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding", **European Transactions on Telecommunications**, Vol. 8, pp. 119-125, 1997.
6. Sazlı, M., H., "Neural Network Implementation of BCJR Algorithm Based on Reformulation Using Matrix Algebra", **IEEE International Symposium on Signal Processing and Information Technology**, pp. 832-837, 2005.
7. Sazlı, M., H., "Neural Network Implementation of BCJR Algorithm", **Digital Signal Processing,** Vol 17; pp. 353-359, 2007.
8. 3GPP, "3GPP Technical Specification", **http://www.3gpp.org**, 2010.
9. Xilinx, Inc., "3GPP Turbo Decoder v4.0 Product Specification", Technical Journal, **http://www.xilinx.com**, 2009.
10. Atar, O., BCJR Algoritması Kullanılan Turbo Kod Çözücülerin FPGA Gerçekleştirimi, Yüksek Lisans Tezi, Ankara Üniversitesi, 2011.