BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TURBO KOD ANALİZİ VE FPGA UYGULAMASI

DENİZ POLAT

YÜKSEK LİSANS TEZİ
2010

TURBO KOD ANALİZİ VE FPGA UYGULAMASI

ANALYSIS OF TURBO CODE AND FPGA APPLICATION

DENİZ POLAT

Başkent Üniversitesi

Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin

ELEKTRİK-ELEKTRONİK Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2010

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma, jürimiz tarafından **ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI 'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Üye (Danışman) : Yrd. Doç. Dr. Mustafa Doğan

Üye : Yrd. Doç. Dr. Hamit Erdem

Üye : Yrd. Doç. Dr. Mustafa Sert

ONAY

Bu tez 15/06/2010 tarihinde, yukarıdaki jüri üyeleri tarafından kabul edilmiştir.

..../06/2010

Prof.Dr. Emin AKATA

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

TEŞEKKÜR

Yazar, bu çalışmanın gerçekleşmesinde katkılarından dolayı, aşağıda adı geçen kişilere içtenlikle teşekkür eder.

Sayın Yrd. Doç. Dr. Mustafa Doğan'a (tez danışmanı), çalışmanın sonuca ulaştırılmasında ve karşılaşılan güçlüklerin aşılmasında her zaman yardımcı ve yol gösterici olduğu için,

Sayın Yrd. Doç. Dr. Hamit Erdem'e ve Sayın Yrd. Doç. Dr. Mustafa Sert' e değerli görüş ve katkılarından dolayı,

Sayın Enis Ungan 'a FPGA ile ilgili katkıları için.

ÖZ

TURBO KODLAMA VE FPGA DE GERÇEKLEŞTİRİLMESİ

Deniz Polat
Başkent Üniversitesi Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Haberleşme sistemlerinin temel amacı, maliyeti mümkün olduğunca düşük verimliliği ise yine mümkün olduğunca yüksek teknolojiler ve algoritmalar geliştirmektir. Bunun için de mesaj paketlerinin en iyi şekilde kodlanmasını, gürültüden minimum etkilenmesini ve mesaj paketinin alıcıda minimum hatayla alınmasını sağlamak gerekmektedir. Shannon limitinin teorik olarak bir haberleşme sisteminde elde edilebilecek en yüksek verimliliği belirlediği 1950'li yıllardan bu yana gibi birçok iletişim algoritması geliştirilmiştir. Bu algoritmalar içinde en başarılı ve teoride en yüksek kodlama kazancını sağlayan algoritma Turbo Kodlama yöntemi olmuştur. İletişim sistemleri için, kodlama kazancı ve Shannon limiti en önemli başarım ölçütleridir.

Bu tez çalışmasında FPGA tabanlı haberleşme sistemleri için Turbo kod uygulaması yapılmıştır. Turbo kod çözücü yapısı, MAP algoritması kullanılarak gerçekleştirilmiştir. Yapılan kodlamanın başarımı, sinyal-gürültü ve hata oranları ile değerlendirildi. Pratik uygulaması FPGA devresi üzerinde gerçekleştirilip, FPGA devresi olarak, Altera firmasının DE1 kiti kullanılmıştır. Bu kod çözme algoritmasının başarımları ve sonuçları detaylı olarak açıklanmış ve incelenmiştir. Turbo kodlama yapısının, bu uygulamada kullanılan FPGA teknolojisiyle de uyumu ve verimliliği ayrıca incelenmiştir. Turbo kodlamanın başarımını değerlendirmek için yeni yaklaşımlar geliştirilmiş, kodlama başarımını arttırmak için kullanılan harmanlayıcının kodlama başarımı üzerindeki etkisi incelenmiştir.

ANAHTAR SÖZCÜKLER: Turbo kodlama, FPGA, Harmanlayıcı, VHDL, MAP
Danışman: Yrd. Doç. Dr. Mustafa Doğan, Başkent Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü

ABSTRACT

The main aim of communication systems is to develop technologies and algorithms, that have relatively low computational cost and high efficiency. Therefore, optimal encoding and transmission with minimum noise and error rates should be achieved. After Shannon's limit was determined theoretically, various communication algorithms have been developed. This limit provides the highest efficiency that can be obtained in a communication system. Turbo Code algorithm has been the most successful one and provided the highest encoding gain in theory among these algorithms. Encoding gain and Shannon limits that will be deemed as a reference for communication systems are important performance measures.

In this study, the FPGA-based turbo codes for communication systems has been implemented. Turbo decoder structure,is carried out by using the MAP algorithm. Signal-to-noise ratio and error rates are main performance measures to compare. Practical application is developed on FPGA circuit, namely the Altera DE1 FPGA kit. The performance results of the decoding algorithm are analyzed in details. The integration level and co-efficiency of Turbo coding structures and FPGA technology have been studied as well. To evaluate the performance of turbo coding, new approaches are developed, and used to improve the coding performance with different interleavers. The sturucture of interleavers and their impact on coding performance are also investigated.

KEY WORDS: Turbo coding, FPGA, Interleaver, VHDL, MAP

Advisor: Asst. Prof. Dr. Mustafa Dogan, Baskent University, Department of Electrical and Electronics Engineering

İÇİNDEKİLER LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Şekil **Sayfa**

ÇİZELGELER LİSTESİ

Çizelge **Sayfa**

SİMGELER VE KISALTMALAR LİSTESİ

r	Kod Oranı
k	Kodlanmamış Bit Sayısı
n	Kodlanmış Bit Sayısı
K	Kısıt Uzunluğu
FPGA	Sahada Programlanabilir Kapı Diziler (Sahada Programlanabilir Kapı Dizileri)
IEEE	Institute of Electrical and Electronics Engineering(Elektrik Elektronik Mühendisleri Enstitüsü)
LUT	Look-up Tablosu
MAP	En Büyük Sonsal Olabilirlik (Maximum a Posteriori Probability)
SOVA	Yumuşak Çıktılı Viterbi Algoritması (Soft Output Viterbi Algoritm)
BCJR	Bahl Cocke Jelinek Raviv
Bps	Saniye Başına Bit (Bits Per Second)
dB	Decibel
AWGN	Toplanır Beyaz Gauss Gürültüsü (Additive White Gaussian Noise)
BER	Bit Hata Oranı (Bit Error Rate)
SNR	Sinyal Gürültü Oranı(Signal Noise Ratio)
MUX	Çoğaltıcı (Multiplexer)
PLD	Programlanabilir Mantık Yapıları (Programmable Logic Devices)
PROM	Programlanabilir Rastgele Giriş Hafızası (Programmable Random Access Memory)
CMOS	Bütünleyici Metal Oksit Yarıiletken (Complementary Metal Oxide Semiconductor)
ISP	Sahada Programlanabilir (In System/Circuit Programmable)

1.GİRİŞ

Haberleşme sistemlerinin temel amacı, verimliliği en yüksek ama bunun karşılığında da maliyeti en düşük yüksek teknolojiler ve algoritmalar geliştirmektir. Bunun için de mesaj paketlerinin en iyi şekilde kodlanmasını, iletim sırasında gürültüden minimum etkilenmesini ve mesaj paketinin alıcıda minimum hatayla alınmasını sağlamak gerekmektedir. Shannon limitinin teorik olarak bir haberleşme sisteminde elde edilebilecek en yüksek verimliliği belirlediği 1950'li yıllardan bu yana kanal kodlama üzerine çok çesitli algoritmalar geliştirilmiştir. Bu algoritmalar içinde en basarılı ve teoride en yüksek kodlama kazancını sağlayan kod yöntemi, Turbo kodlama yöntemi olmuştur. Örnek olarak, turbo kodlama günümüzde 3G, 4G ve IEEE 802.16 (WiMAX) uygulamalarında zorunlu bir standart olarak kullanılmaktadır.

Haberleşme sistemlerinin vazgeçilmez bir unsuru olan kanal kodlama konusu üzerinde çok sayıda çalışma yapılmıştır. Günümüzde konu üzerinde yoğun bir şekilde çalısmalar yapılmaya devam edilmektedir. Özellikle de telsiz haberleşme sistemlerinde son derece önemli bir yer teşkil eden kanal kodlama işleminin amacı, başarımı en yüksek kod algoritmaları geliştirmektir. Bu amaca yönelik olarak çok çesitli kanal kodlama teknikleri ve algoritmaları geliştirilmiştir [1, 2].

Kanal kodlama haberleşme sistemlerinin temelini oluşturmaktadır. Amaç; kodlanmış verilerin kanala iletilmesinde ve alıcıda alınmasında minimum veri kaybıyla tekrar elde edebilmektir. Turbo kodlar da bu amaçla düşünülmüş ve ortaya çıkarılmış bir kodlama sistemidir. İlk kez Forney tarafından geliştirilen ve yayınlanan "Birleştirilmis Kodlama Şeması" iki veya daha fazla kodun birbiriyle bağlanmasını anlatır. Bu bağlanma sonucunda ortaya çıkan uzun kodlar hata doğrulama kapasitesine sahiptir. Ayrıca bu sonuç kodları daha karmaşık kod çözücüye izin verecek bir yapıya bürünmüş durumdadır. Bir turbo kod bir veri dizisinin kodlanması için kodlayıcı ve bu birleştirilmiş kod dizisinin çözülmesi için tekrarlı algoritmayı içeren bir yapıdır [3].

Kodlama konusunda referans kabul edilen ve teoride en iyi kodlama kazancı ve başarımlarının alındığı Shannon kuramının yayınlandığı 1948 yılından günümüze kadar birçok kodlama teknikleri denenmiş ve uygulanmıştır. Ancak bu kodlama teknikleri çok küçük kanal kapasitelerinin verimli kullanımına neden olmuştur.

Turbo kodların 1993'te keşfedilmesiyle bir kanalın band genişliği çok büyük kodların taşınmasına ve hata denetimine izin verecek ölçüde büyümüştür. 1993 yılında Berrou, Glavieux ve Thitimajshima tarafından turbo kodların bulunmasından sonra, sayısal haberleşme ve veri kayıt sistemleri alanında başarım analizi, tasarım, gerçekleme ve uygulamaya yönelik olarak birçok araştırma yapılmıştır [4, 5].

Turbo kodlama 17 yıllık bir geçmişi olan bir kanal kodlama tekniğidir. Son zamanlarda bu teknik geliştirilerek çok daha başarılı kodlama algoritmaları geliştirilmiştir. Haberleşme sektöründe önde gelen ve standartlaşan kodlama yöntemlerinden biri haline gelmiştir. Bu teknik en son NASA'nın Mars ile iletişimini sağlamak için kullanılmıştır.

Herhangi bir kodlama yönteminin amacı güvenli ve bilgi kaybı olmadan haberleşme, sınırlı band genişliğini kullanarak ekonomik davranmaktır. Shannon'dan önce iletilen bilgide kayıp olmaksızın bir kodlama yapılamayacağı, yapılsa bile büyük bir kanal kapasitesine sahip olması gerektiği öngörülmekteydi.

Shannon'un kanal kodlama konusunda devrim sayılacak buluşundan sonra ise kanal kapasitesini çok zorlamadan sadece ana veriye yine kendisinden türetilmiş olan, denetim bitleri eklenerek, hem güvenli hem de veri kaybı olmadan iletimin olabileceği görüldü. Turbo kodlama Shannon teorisinde ulaşılabilecek bütün limitlerin kesin olarak ulaşıldığı, hem güvenilirlik hem de veri doğruluk açısından en başarılı kodlama tekniklerinden birisidir.

Turbo kodlar bir harmanlayıcı tarafından birleştirilmiş giriş ve çıkış kodlarından oluşur. Bu yapı özellikle uydu haberleşmesinde kullanılmaktadır. Turbo kod gibi

birleşik bir kod oluşturmanın temel felsefesi daha az hata oranlı ve basit bir kodlama yapısında bir kod çözücüsü oluşturabilmektir. Bir turbo kod çözücü ya en büyük sonsal olasılık (Maximum a Posteriori Probability, MAP) algoritması yada yumusak çıkışlı viterbi algoritması (Soft Output Viterbi Algorithm, SOVA) kullanarak, kod çözücüde giriş dizisinden yeni kodlar oluşturulmasını sağlarlar [6].

Kodlanmış mesajın çözülmesinde kullanılan önemli algoritmalardan MAP(BCJR) algoritması incelenmistir. MAP algoritması kullanarak gerçekleştirilen kanal kodlamanın da hata oranları oldukça düşük değerlerde çıkmıştır. Bit hata oranları da bu kodlamanın başarısının ölçülebileceği en önemli ölçüttür. Bu açıdan gerçekleştirilen uygulamalar BER analizinde, ilgili literatüre göre daha iyi sonuçlar elde edilmiştir [7, 8, 9].

Bu tez çalışmasında, birinci bölümde, turbo kodlama ve bu kodlamada kullanılan algoritmalar incelenmiştir [10-15]. İkinci bölümde ise kod çözücü yöntemi incelenmiştir. Ayrıca, harmanlayıcının etkisi araştırılmıştır. Son bölümde ise kodlama yöntemi donanımsal olarak FPGA üzerinde gerçekleştirilmiştir. Ayrıca, FPGA üzerindeki başarısı ölçülmüştür [16].

2.KOD YAPILARI

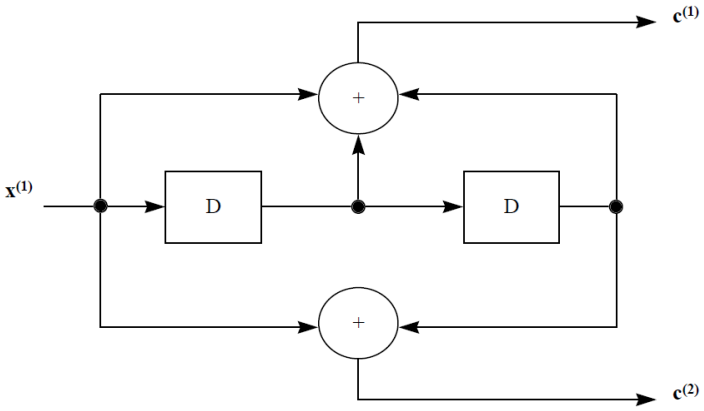
Günümüzde birçok çeşit kodlama yöntemi kullanılmaktadır. Bu kısımda, Turbo kodlama için kullanılan evrişimli kodların yapıları ve işleyiş kuralları açıklanmıştır.

2.1 Doğrusal Blok Kodlar

Doğrusal blok kodlar, kodların doğrusal olarak birleştirilmesiyle ortaya çıkar. Bu tezde anlatılan Turbo kodların yapısında, birden çok evrişimli kodlayıcı bulunmaktadır. Evrişimli kodlar ve Turbo kodlar doğrusal blok kodlardandır.

2.2 Evrişimli Kodlar ve Gösterimleri

Evrişimli kod (Convolution code) iki farklı parametre ile tanımlanır. Kod oranı (Code rate) ‘r=k/n’ ve kısıt uzunluğu (constraint length) ‘K’ . Burada k evrişim kodlayıcıya giren mesaj bitlerinin sayısıyken, n evrişim kodlayıcıdan çıkan kodlanmış bitlerin sayısıdır. Kısıt uzunluğu ile ifade edilen K, evrişim kodlayıcıya giren bir bilgi bitinin çıkış kodunu kaç adım boyunca etkilediğini gösterir. Şekil 2.1’de görüldüğü gibi K=2, r=1/2 değerlerine sahip bir evrişimli kodlayıcı (Convolutional encoder) örneği verilmiştir. Burada, D ile gösterilen blok, giriş yapan bitin kaydırılma (shifting) işlemidir. Her bir D, bitin bir defa kaydırıldığını gösterir. Verilen şekilde, x(t) girişi, c(t)’ler ise çıkışı temsil eder.



Şekil 2.1 Evrişimli Kodlama Gösterimi

2.2.1 Bağlantı Gösterimi

Bir önceki sayfada yer alan Şekil 2.1 ‘de görülen gösterim, evrişimli kodlayıcı giriş ve çıkışları arasındaki dürtü yanıtı (impulse response) aşağıdaki çizelgede gösterilmiştir. Bu çizelge, Bağlantı Gösterimi olarak adlandırılır.

Çizelge 2.1 Giriş – Çıkış Bağlantı Gösterimi

Giriş dizisi	1	0	0
Çıkış Dizisi	11	10	11

Dürtü yanıtı kullanılarak [1 0 0] girişine karşılık gelen evrişimli kodlama sonucu ortaya çıkan çizelge aşağıda gösterilmiştir.

Çizelge 2.2 Evrişimli Kodlayıcının Giriş – Çıkış Bağlantı Gösterimi

Giriş	Çıkış				
1	11	10	11		
0		00	00	00	
1			11	10	11
İkili sistemde toplam	11	10	00	10	11

2.2.2 Polinom Gösterimi

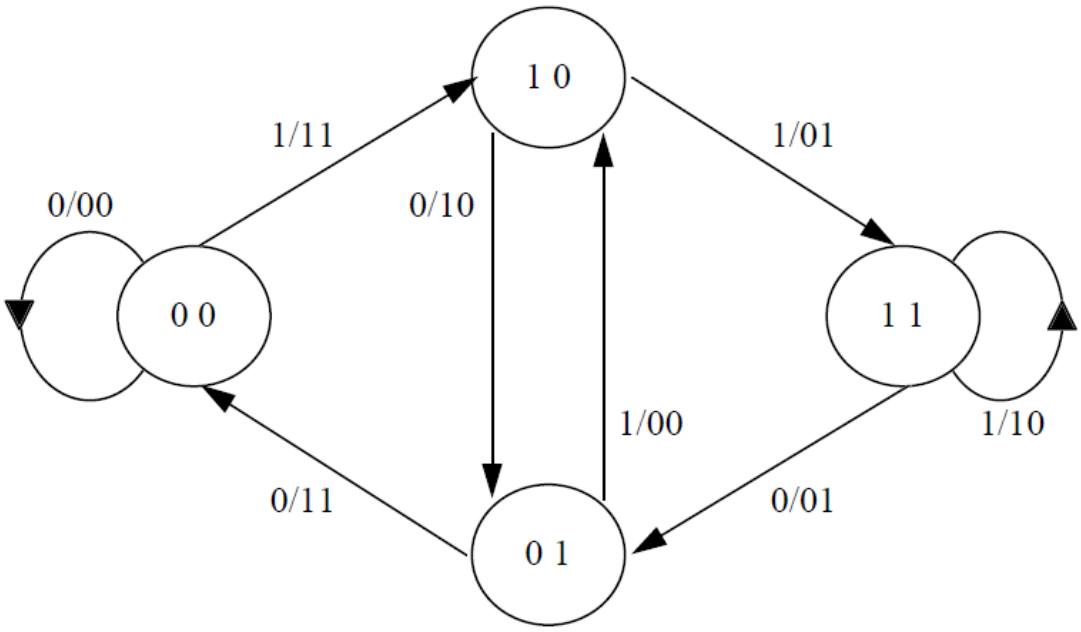
Bazı durumlarda evrişimli kodlayıcının çıktıları polinomlarla gösterilir. Buradaki evrişimli kodlayıcının polinom gösterimi aşağıdadır.

C_1 yolu için $g_1(x)=1+X+X^2$

C_2 yolu için $g_2(x)=1+X^2$

2.2.3 Durum Gösterimi

Evrişimli kodlayıcılar, sınırlı hafızaya sahip oldukları için sonlu durumlu makine(finite state machine) olarak açıklanabilir. Burada Şekil 2.1'de örneği verilen kodlama yapısının durum gösterimi ifade edilir. Bu gösterimde '00' durumundan başladığımızı varsayarsak giriş bitimiz'1' olduğunda çıktımız '11' olmaktadır. Bir sonraki adımda '10' durumuna ilerlemiş oluruz. Buradaki '10' durumu aynı zamanda iki yazmacın çıktısını belirtir. Bu şekilde devam ederek tüm çıktılar kolay bir biçimde hesaplanabilir. Durum gösterimi, döngülü (iterative) bir algoritma yazılmasına büyük uygunluk göstermektedir.

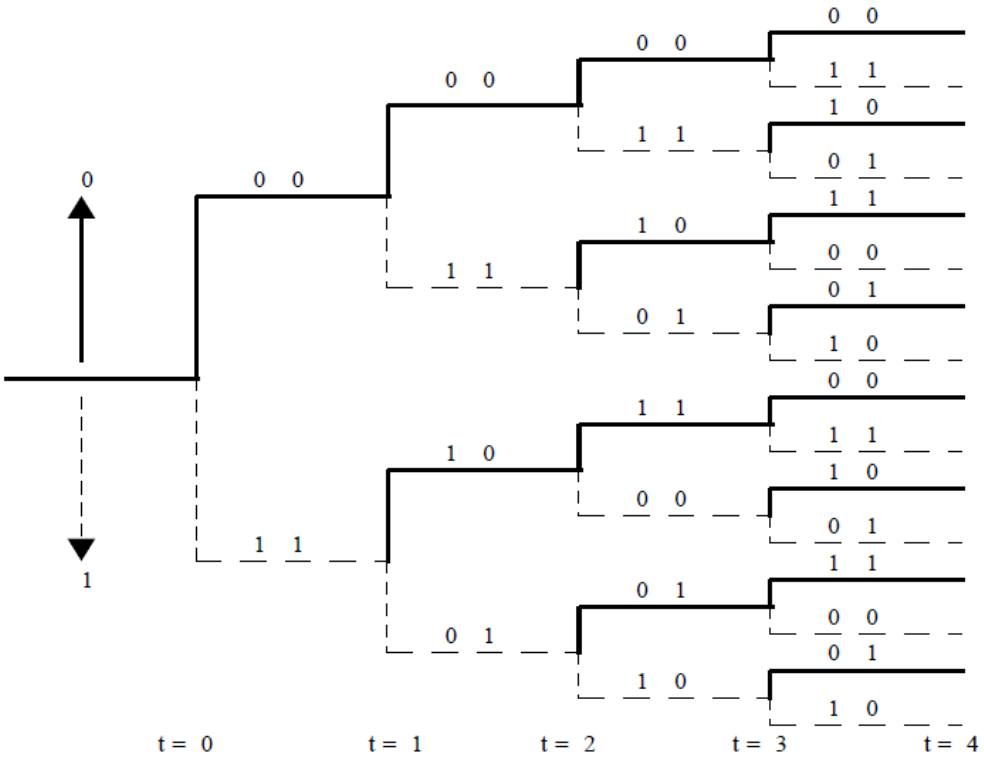


Şekil 2.2. Evrişimli Kodlayıcının Durum Gösterimi

2.2.4 Ağaç Şema Gösterimi

Ağaç şeması (Tree diagram) oluşabilecek tüm durumları gösteren bir ağaç yapısıdır. Burada Şekil 2.1'de örneği verilen kodlama yapısının ağaç şeması ifade edilmektedir. Şekil 2.3'de bu evrişimli kodlayıcının ağaç yapısı bulunmaktadır. Bu ağaç yapısı yukarıda bulunan durum yapısına zaman bilgisinin eklenmiş halidir. Kesikli çizgi ile ifade edilen yol giriş bitinin '1' olduğu zaman seçilir. Giriş biti '0' olduğu zaman diğer yol seçilmelidir. Bu şekilde ilerliyerek evrişimli kodlamanın

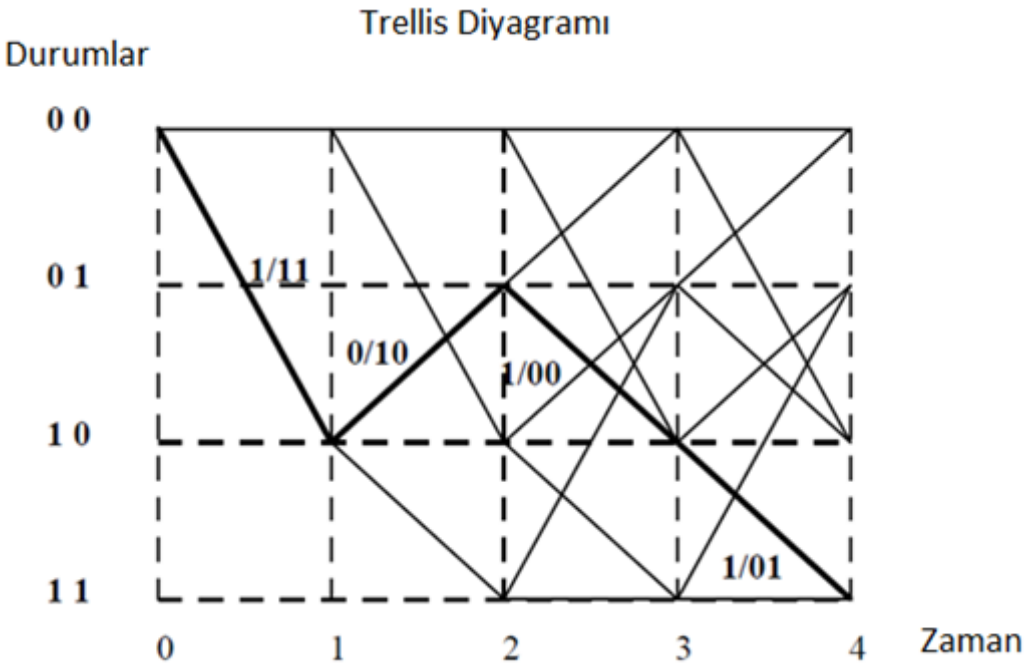
hangi zamanda hangi çıktıyı verdiği görülebilir. Burada dikkati çeken bir diğer özellik 3. adımdan sonra ağaç şemasının üst kısmı ile alt kısmının birbirinin aynısı olmasıdır. Genel formül olarak K+1'inci adımda bu özellik kendini göstermektedir. Burada K kısıt uzunluğunu ifade eder ve daha öncede değinildiği gibi bu örnekte K=2 dir. Bu ağaçta genel olarak 4 durum vardır. Bunlar [00], [10], [01] ve [11] durumlarıdır. Bunlara karşılık olarak a, b, c ve d sembolleri bir sonraki gösterimde kullanılacaktır.



Şekil 2.3. Evrişimli Kodlayıcının Ağaç Şema Gösterimi

2.2.5 Trellis Şeması Gösterimi

Bu şema evrişimli kodlamanın çözülmesi açısından oldukça önemli bir gösterimdir. Şekil 2.4'de evrişimli kodlayıcımızın trellis şeması verilmiştir. Trellis şemaları arka arkaya eklenerek haberleşme sırasında oluşan durumlar gözlenebilir.



Şekil 2.4. Evrişimli Kodlayıcının Trellis Şema Gösterimi

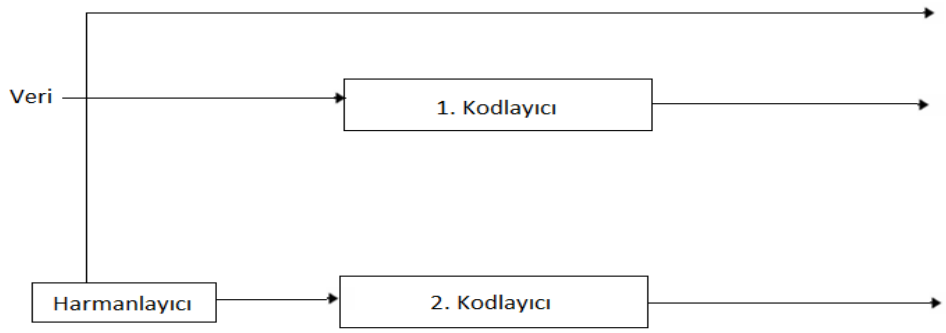
Şekil 2.4, durumların ve bu durumların gerçekleştiği zamanlar arasındaki bağlantıları göstermektedir. a/bc gösterimlerinde ise;
a, giriş bitlerini; bc ise, çıkış bitlerinin durumunu açıklamaktadır.

3. TURBO KODLAMA

İlk olarak 1993 senesinde Berrou tarafından tariflenen klasik Turbo kodlar, mükemmele yakın kod çözücü başarımları sayesinde büyük ilgi toplamıştır. Turbo kod, klasik diğer kodların daha da gelişmiş halidir ve IEEE 802. 16 (WIMAX) ve DVB-RSC gibi bugünün haberleşme standartlarında yaygın olarak kullanılmaktadır. Bu kodlar, klasik kodlara kıyasla daha iyi hata düzeltme yeteneğine sahip olmakla birlikte çözücü acısından daha fazla işlemsel karmaşa içermektedir. Bu çalışmada, turbo kod çözücünün bütün yönleriyle araştırılmasına ve MAP(maximum a Posteriori) algoritmasının açıklanmasına yönelik çalışılmıştır.

3.1 Turbo Kodlama Temelleri

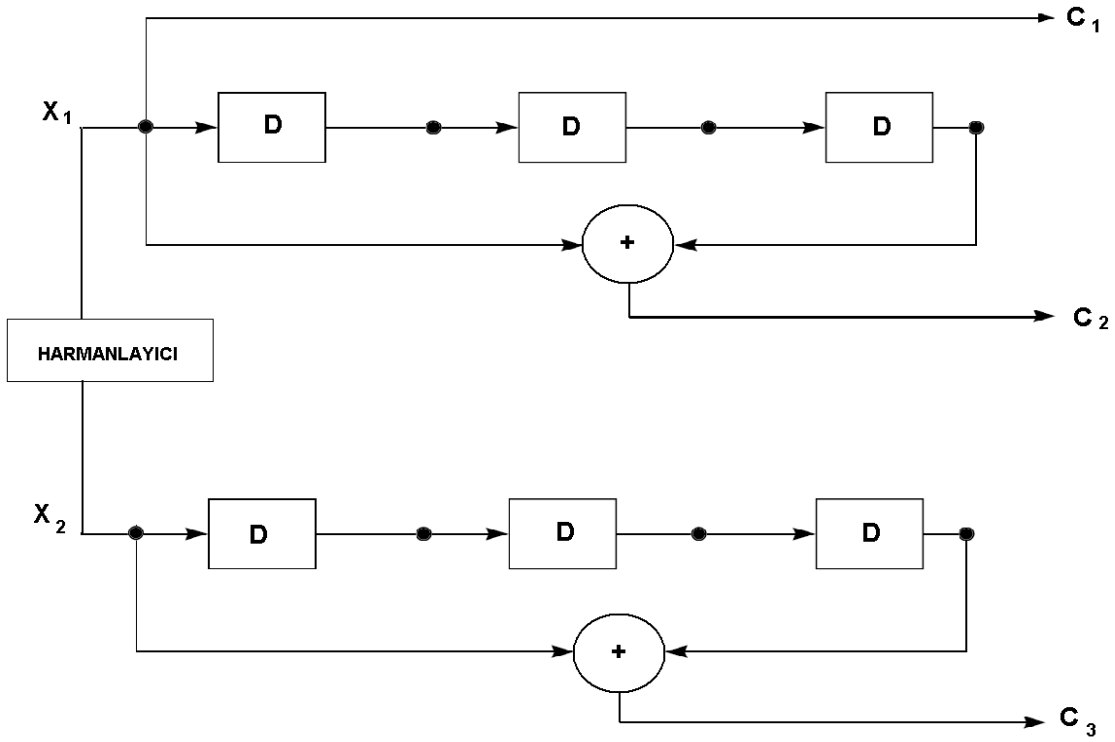
Bu kodlar basit kodlama yöntemiyle kodlanırlar. Kodlayıcı temelde 1 bit alıp 3 bit üretmektedir. Üretilen bu 3 bitten ilki asıl giriş biti olup 2. bit, 1. kodlayıcının ürettiği eşlik biti ve 3. bit ise 2. kodlayıcının ürettiği eşlik bitidir. 1/3'lük kodlayıcı Şekil 3.1 ile verilmiştir. Birinci ve ikinci basit kodlayıcılar özdeş seçilmişlerdir. İki kodlayıcı arasına hata patlamasının etkilerini azaltmak için veri dizisinin boyutunda bir rastgele serpiştirici yerleştirilmiştir. Her bit böylece 3 bit ile gösterilmekte olup kodlayıcı çıkışında paralelden seriye dönüştürme işlemi yapılıp iletim kanalına verilmektedir. Burada, ikinci kodlayıcıdan önce bulunan harmanlayıcı, hata düzeltme işlevi için konulmuştur. Bu konu ileri bölümlerde daha detaylı olarak incelenmiştir.



Şekil 3.1. Turbo Kodlayıcının Genel Yapısı

Turbo kodlama tekniği, yapısal olarak ardışık kodlama tekniğinin geliştirilmiş bir versiyonu ile kod çözme için kullanılan bir kod çözme algoritmasından ibarettir. Turbo kodlamanın temel unsurlarından biri olan ardışık kodlama, ilk olarak Forney tarafından tasarlanmış bir tekniktir. Bu teknik, iki ya da daha fazla basit kodlayıcının, yüksek kod kazancına ulaşabilmek için paralel yada seri birleştirilmesinden oluşmaktadır. Netice de ortaya çıkan bu konular, çok uzun kodların hata düzeltme kabiliyetlerine sahip olmakla birlikte onlara göre daha makul sayılabilecek karmaşıklıkta bir kod çözme yapısına sahiptirler.

Turbo kodlayıcının genel yapısı şekil 3.1'de verilmiştir. Her iki kodlayıcıda aynı veriyi alır. Fakat ikinci kodlayıcı giriş verisi harmanlayıcıdan geçtikten sonra oluşan yeni dizilimli veriyi alır. Turbo kodlarının rastgele gibi görünmesini sağlayan harmanlayıcı işlemidir.



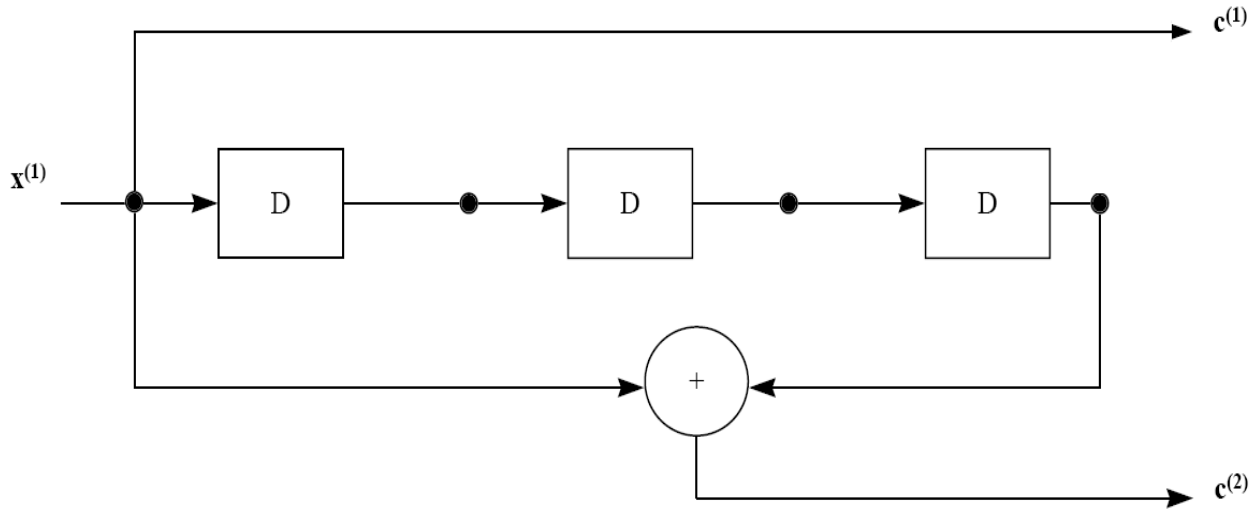
Şekil 3.2. Turbo Kodlayıcının Yapısı

3. 2 Basit Kodlayıcıların Yapısı

Şekil 3.3'te, turbo kodlayıcının özdeş olan basit kodlayıcılarından bir örnek bulunmaktadır. Burada girişe gelen veri biti iki çıktı olarak kodlayıcıdan çıkmaktadır. İlki, girişe uygulanan giriş bitinin kendisiyken, ikinci çıkış biti ise geçtiği süreçler sonrasında kodlanmış olan bittir.

Burada girişte sadece bir bit ve çıkışta da iki bit olduğu için, bu basit kodlayıcının kodlama oranı ½ dir. Kodlama oranı, k/n olarak gösterilir. Burada k giriş sayısı, n ise kodlayıcının çıkış sayısıdır. Kodlama oranının gösterimi ise r dir. Yani, kodlama oranı r=k/n olarak gösterilir.

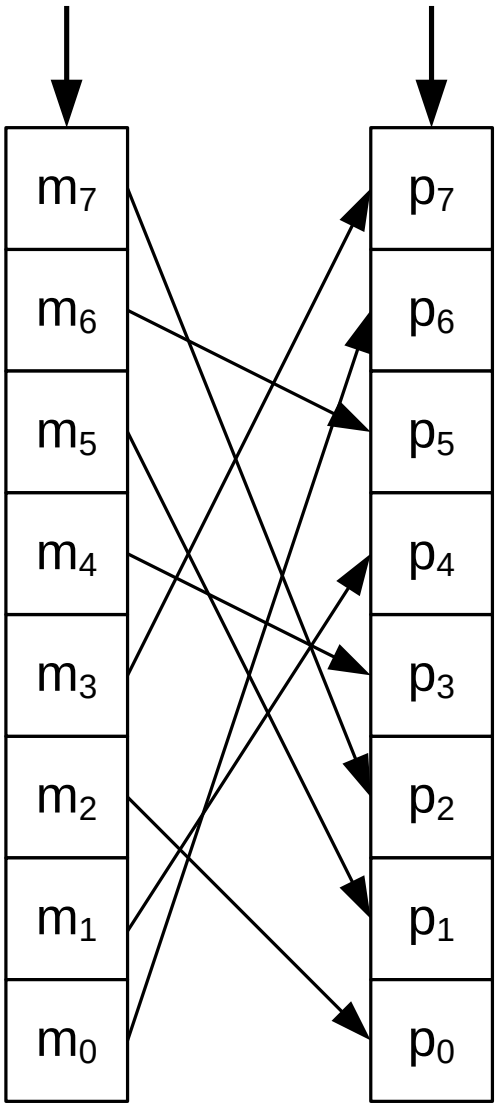
Burada D ile gösterilenler ise adım sayılarıdır. Girilen verinin kaç kere öteleneceğini gösterir. Bu sayı K ile gösterilir. Yani, K değeri girişin değişime uğradığı adım sayısıdır. Bu şekilde K sayısı üçe eşittir.



Şekil 3.3. Basit Kodlayıcının Genel Yapısı

3.3 Harmanlayıcı Yapısı

Harmanlayıcı, basit anlamda, bir dizideki bitlerin yerlerini rassal olarak değiştiren bir yapıdır.

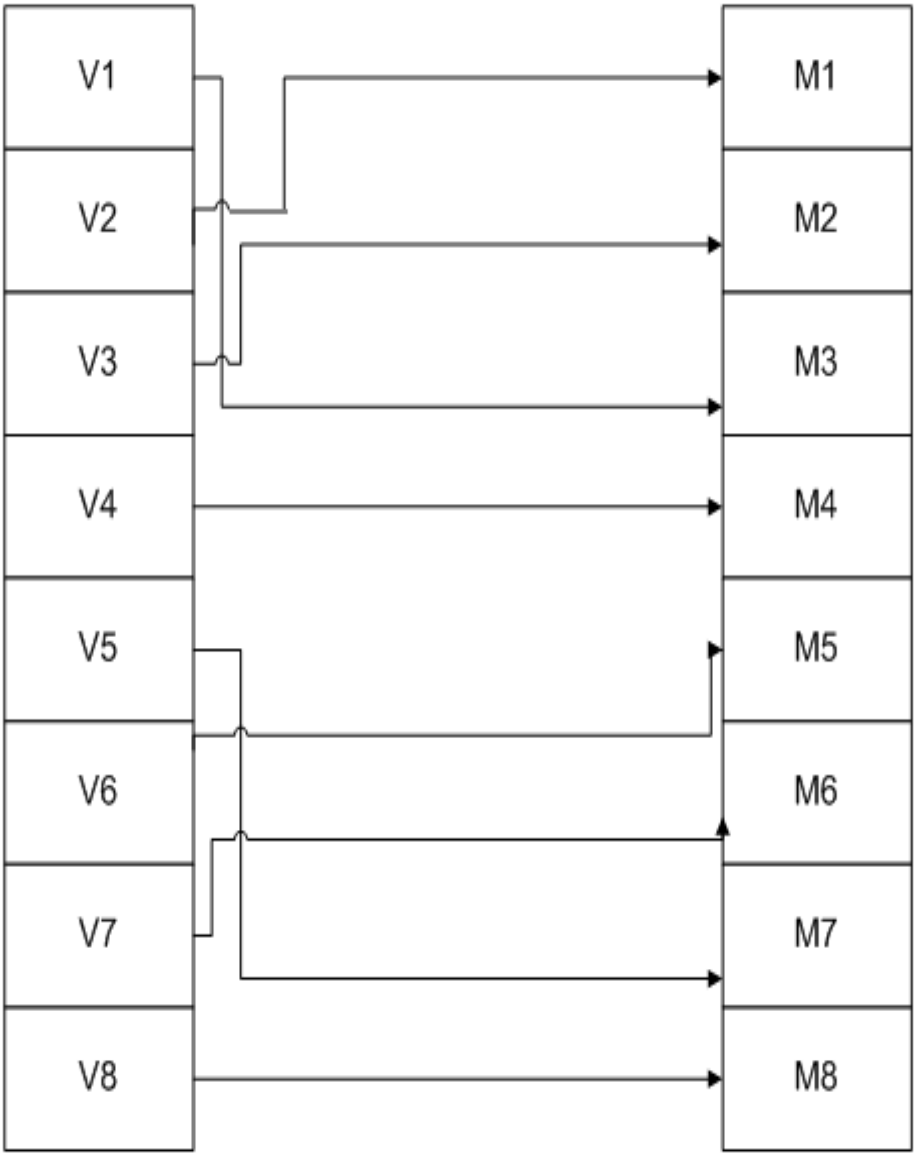


Şekil 3.4. Harmanlayıcı Yapısı

Harmanlayıcının yapısı, Şekil 3.4'de de görüldüğü gibi rastgeledir. Hangi sıradaki bitin harmanlayıcıdan hangi sırada çıkacağı tamamen rastgele seçilmektedir. Bu çalışmada, harmanlayıcıların sonuçlara etkisini karşılaştırabilmek için iki adet farklı harmanlayıcı yapısı kullanılmıştır.

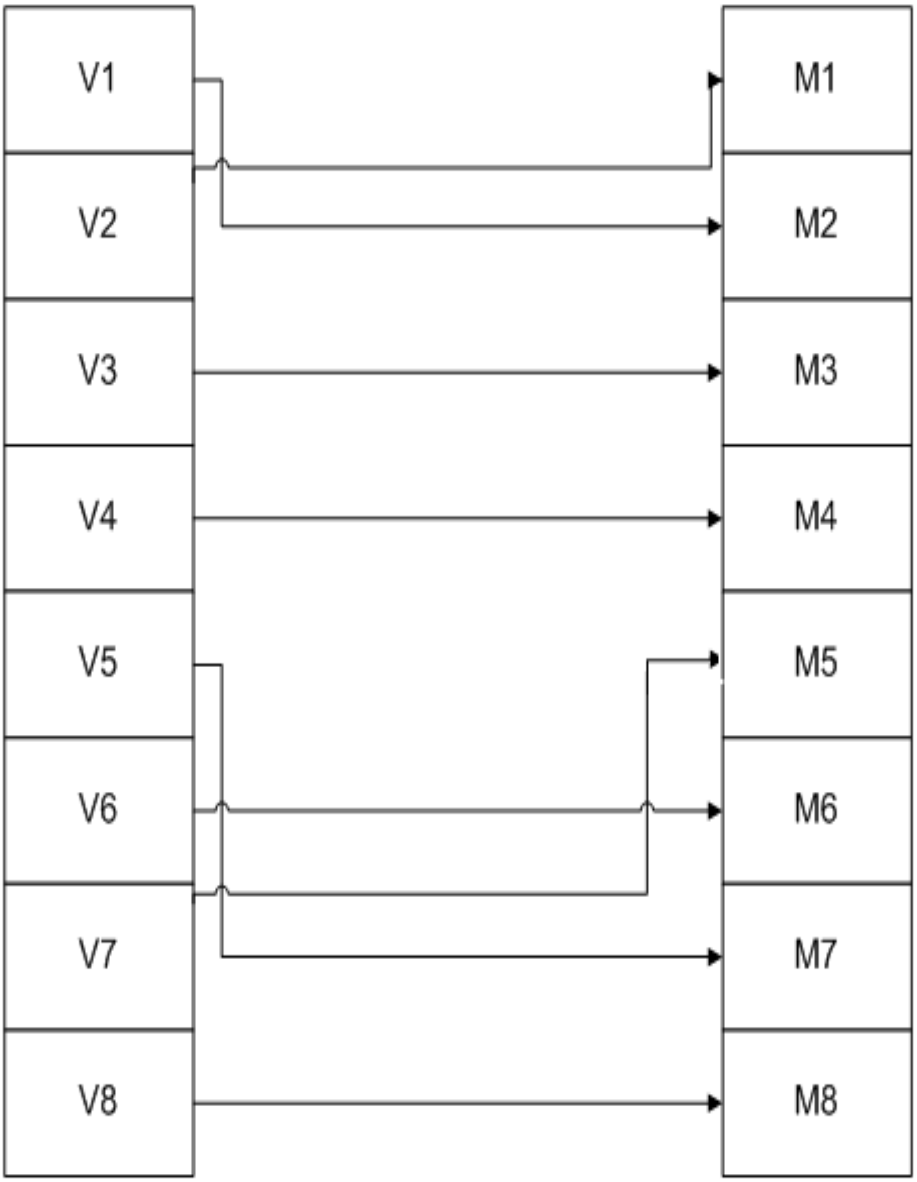
Bunlar ;

Harmanlayıcı Yapısı A için,



Şekil 3.5 Harmanlayıcı Durum A Yapısı

Harmanlayıcı Yapısı B için,

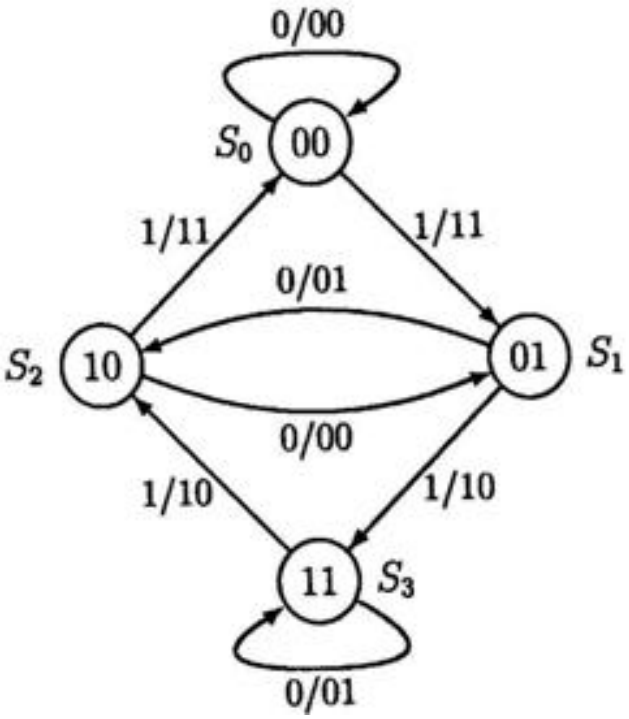


Şekil 3.6 Harmanlayıcı Durum B Yapısı

Bu iki harmanlayıcı durumu olmak üzere iki farklı dizilim kullandım. Günümüzde; bu harmanlayıcılardan ayrı olarak birçok farklı özelliklere sahip harmanlayıcılar bulunmaktadır.

3.4 Kafes Sonlandırılması

Turbo Kod oluştururken kullanılan bir diğer önemli yapı, kafes sonlandırılmasıdır. Girişteki ilk veriden itibaren sona kadar uzayan kod dizileri kafes sonlandırılması algoritmasıyla şekillenmektedir. Bir kafes şeması, bütün giriş ve çıkış dizileri ve durum geçişleri şemasından faydalanarak oluşturulur.

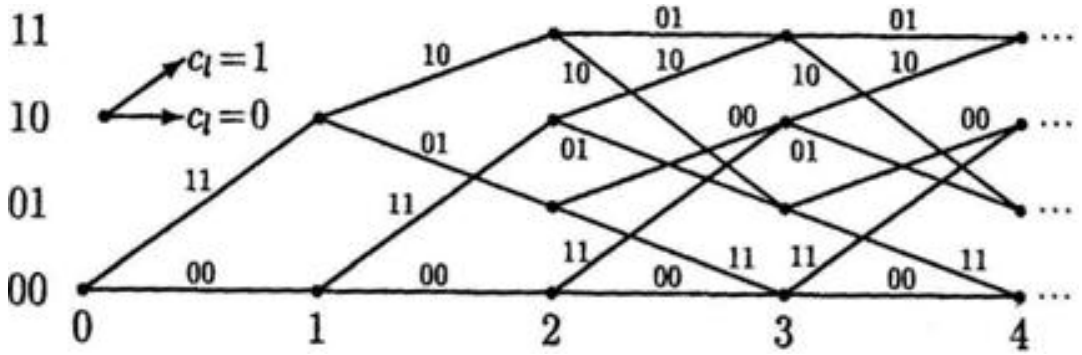


Şekil 3.7. Sistematik Olmayan Standart (2, 1) Kodlayıcı için Durum Şeması

Şekil 3.7'de gösterilen şemada bütün sıfır durumları (00) ve her bir geçiş safhaları genişletilerek elde edilmiştir. İlk mesaj verisi yazmaca geldiğinde, kodlayıcı eğer giriş sembolü 1 ise (10) durumu, eğer giriş sembolü 0 ise (00) olan bir çıkışa yönlenmiş olur. Bir sonraki sembol yazmaca geldiğinde, kodlayıcı şu olası 4 durumdan birine yönlenecektir; (00), (01), (10) ve (11). Bu olası durum çıkışları her bir yeni kayma sırasında katlanarak artacaktır.

Her durum karşılığından iki farklı sembole iki dallanmış yayılma vardır. Ayrıca her bir yeni durum oluşurken iki dallı düğüm birleşir. Yazmaçtan çıkan sembol 0 durumu ise geçişlerde dalların alt kısımlarında, sembol durumu 1 ise geçişlerde en

üstteki dallarda sıralanır. Oluşan bütün bu dallanmalı yapı çıkışta sınıflandırılmıştır.



Şekil 3.8. Evrişimli Kodlayıcı (Şekil 2.1) için Kafes Şeması

Çıkış kod dizileri kafes şemasındaki yollar izlenerek bulunur. Örneğin giriş dizisi c=(11001) için çıkış dizisi kafes şemasından okunursa, v= (1110101111) ifadesi bulunur. Sonuç olarak (n, k) gibi bir kod için kafeste 2k dallı bir seçenek vardır ve çıkış dizisi bu 2k gibi farklı dalların durumlarından biridir. Bir turbo kodlayıcı her bir blok veri için bir blok kodlayıcısı gibi çalışır. Ancak bileşen kodlar tekrarlı bir yapıda olduğu için sıfır kuyruk verisi iletiminde kafes şemasının aynen uygulanması doğru olmaz. Kafes sonlandırılması, kodlayıcıyı bütün sıfır durumları için yönlendirir. Her bir blok sonunda gelecek her bir yeni blok sıfır durumundan başlar.

3.5 Hamming Uzaklığı

Hamming uzaklığı aynı uzunluktaki iki dizenin farklı elemanlarının sayısıdır. Hamming uzaklığı adı Richard Hamming'in adından gelir. Hamming iki ve daha fazla bitin değişmesi problemini uzaklık olarak tanımlamıştır. Hamming bu uzaklığı ve gönderilen mesajdaki bilgi oranını olabildiğince artırmaya çalışmıştır. 1940'lar boyunca var olan kodlar üzerinde önemli gelişmeler gerçekleştiren birkaç kod algoritması geliştirmiştir. Bütün sistemlerin anahtar noktası, denetim bitlerinin amacı bütün bitlerin ve verinin olabildiğince doğru iletilmesini ve güvenliğini sağlamaktır.

Bu tezde harmanlayıcı yapısı, hamming uzaklığını arttırmak için kullanılmıştır. Harmanlayıcı ne kadar rassalsa, hamming uzaklığı o kadar büyüktür ve hatalı veri iletimi okadar az olur.

3.6 MAP (Maximum a Posteriori) Algoritması (BCJR)

BCJR algoritmasının ismi, bu algoritmayı bulan kişilerin isimlerinden gelmektedir. 1974'te Bahl, Cocke, Jelinek ve Raviv bir gürültülü ayrık belleksiz kanal (DMC, discrete memoryless channel) üzerinden gözlemlenen Markov kaynağın durumlarının ve geçişlerinin, sonsal olasılıklarını (a posteriori probabilities) özyineli olarak tahmin eden bir algoritma sunmuşlardır. Bu sunumun ardından da bu algoritma BCJR algoritması veya Forward-Backward algoritması olarakta bilinir.

Sözü edilen makalede yazarlar, bu problemin özel bir durumu olarak doğrusal kodların (katlamalı ve blok) kod çözümünü ve bu algoritmanın doğrusal katlamalı ve doğrusal blok kodların kod çözümünde kullanılabilir bir algoritma olduğunu göstermişlerdir. BCJR algoritması katlamalı ve blok kodlara doğrudan uygulanabilir olmasına rağmen, artan karmaşıklık nedeni ile Viterbi algoritmasına göre tercih edilen bir algoritma olmamıştır. Çünkü, Viterbi algoritması, yapı itibari ile MAP algoritmasına göre işlem yükü bakımından çok daha kolay bir algortimadır. Viterbi algoritmasının tercih edilmesinin bir diğer nedeni ise "özyineli kod çözmenin (iterative decoding)" veya "önyükleyici kod çözmenin (bootsrap decoding)" o tarihlerde öneminin yeterince anlaşılamamış olmasıdır. Sonraki yıllarda, "sıralı kodlara" (concatanated codes) olan ilginin artması ve "turbo kodların" gelişi, BCJR algoritmasını daha popüler yapmıştır. Turbo kod çözücülerin bileşen kod çözücüleri ve diğer sıralı kodların kod çözümünde döngülü kod çözme şemaları için en fazla kullanılan algoritma olmasını sağlamıştır. Bu da BCJR algoritmasının kullanılabilirliğine ve özgün algoritmanın değiştirilmesine gerek olmaksızın yumuşak çıkışlar üretme yeteneğine atfedilebilir. BCJR ismi bu algoritmayı ilk olarak geliştirenlerin (Bahl, Cocke, Jalinek, Raviv) adlarından ileri gelmektedir. Bu algoritma MAP algoritmasının verimli bir gerçeklemesi olup, mucitlerinin anısına BCJR algoritması olarak bilinmektedir.

Viterbi algoritması yumuşak çıkışlar üretme yeteneğine sahip değildir ve bu yüzden orijinal hali ile döngülü kod çözücülerde kullanıma uygun değildir. Hagenauer ve Hoeher daha az karmaşık ama en uygun olabilecek olan SOVA (Yumuşak Çıkışlı Viterbi Algoritması) olarak isimlendirilen bir alternatif önermişlerdir. SOVA'nın azaltılmış hesap yüküne karşılık, BCJR algoritması ve onların optimal altı türevleri turbo kod çözücülerde ve diğer döngülü kod çözücülerde daha fazla tercih edilmekte ve kullanılmaktadır.Literatürde MAP yada BCJR algoritması olarak da bilinen ilk olasılıkların yumuşak girişli, sonsal olasılıkların ise sert kararlı olduğu, tekrarlı oldukça kullanışlı bir algoritmadır. BCJR algoritması hata olasılığını minimum düzeyde tutar ve bir APP (A Posteriori Probability) üretir. Evrişimli ve blok kodlar için en uygun kod çözme algoritması olduğu gösterilmiş ancak turbo kodun yapısından ileri gelen tekrarlamalı algoritmasına uygun hale getirilmek için yeni bir düzenleme ile geliştirilmiştir.

MAP algoritması günümüzde, turbo kodlama için kullanılan en önemli çözücü algoritmadır. İletimin güvenilirliği açısından en iyi sonuçları MAP algoritması vermektedir. Turbo kodlamada, verinin kodlanması ve kanaldan doğru bir şekilde veri alınması aşamasında henüz MAP algoritmasının doğruluk oranlarını geçecek bir algoritma bulunmamaktadır. Bunun nedeni de, bu algoritmanın olasılıklar hesabını maksimum kesinlik payları ile yapmasıdır. En küçük bir hata ihtimalinde bile, sistem aldığı veriyi iptal etmekte ve bu veriyi yollayıcıdan tekrar istemektedir. Bu tür özellikler turbo kodlamanın neden bu kadar üstün olduğunu göstermektedir. Ancak bu iletişimde verimlilik kaybı gibi gözükse de, paralel yapıdaki kodlayıcı - kod çözücü ve harmanlayıcının birlikte tasarlanmış olması, bu kaybı hissedilmeyecek oranlarda gerçekleştirir.

3.6.1 Log-Likelihood Oranı(Logaritmik Olabilirlik Oranı)

MAP algoritması, logaritmik olabilirlik oranını kullanarak işlem yapmaktadır. Bütün matematiksel işlemleri bu oran üzerinden hesaplamaktadır. MAP algoritması için kullanılacak metriklerin hesaplanmasında $L(Uk) = \ln [P(Uk=+1) / (P(Uk=-1)]$ ifadesi kullanılır. Burad P ifadeleri, olasılık kavramı olarak kullanılmıştır. Burada Uk bilgi bitidir. k zamanındaki U değerini belirtir. Burada kanaldan alınan yada kanala

verilen 0 ve 1 bitleri için -1 ve +1 metrikleri kullanılır. 0 değeri -1, 1 değeri ise +1 ile ifade edilmektedir. Ayrıca, bu yöntem hata düzelme için en çok kullanılan yöntemdir. En önemli özelliği çok hassas olmasıdır.

L(Uk) = ln [P(Uk=+1) / (P(Uk=-1)] ifadesini adım adım açarsak,
L(Uk) = ln [P($Y\binom{N}{1}$, Uk=+1) / P($Y\binom{N}{1}$, Uk=-1)] ifadesini buluruz. Bu formül, aldığımız bitler ile bilgi bitlerinin arasındaki ilişkiyi verir.

Teorik olarak;

1-Başlangıç durumu

2- Bitiş durumu

3-Giriş biti

bu üç bilgiden ikisi biliniyorsa hatası çok az bir çözüm yapabiliriz.

P(Uk=+1, $Y\binom{N}{1}$) ile P(s', s, $Y\binom{N}{1}$) gösterimleri olasılık kavramı içerisinde aynıdır. Çünkü, burada s' metriği ilk durumu, s metriği ise son durumu belirtir.

$Y\binom{N}{1}$ değerini $Y\binom{k-1}{1}$, Yk, $Y\binom{N}{k+1}$ olarakta yazabiliriz. Bu metrikleri ;

$Y\binom{k-1}{1}$ = Ygeçmiş (Yp),

Yk =Yşimdi (Yk),

$Y\binom{N}{k+1}$ =Ygelecek (Yf) olarak adlandırabiliriz.

Böylece formülümüz ; P(s', s, Yp, Yk, Yf) olarak şekillenmiştir.

Önceki gelen ve şimdiki verilerı bildiğimize göre, gelecek veri olarak adlandırdığımız verinin olasılık hesabı yapılacaktır. Yani formülümüz; P(Yf |

s’,s,Yp,Yk). P(s’,s,Yp,Yk) olarak güncellenmiş olur. Burada da gelecek olan verinin kestirim hesabı geçmiş ve şimdiki verilerden bağımsız olduğu için, bu formüldeki s’, Yp ve Yk değerleri hesabımızda etkisiz eleman olurlar ve bu nedenden dolayı silinebilirler. Bu şekilde formülümüzün son hali ;

P(Yf | s)P(s’, s, Yp, Yk) şeklindedir.

Yukarıda yaptığımız işlemleri, formülümüzün ikinci kısmınada uygularsak;

P(s’, s, Yp, Yk) = P(s, Yk | s’, Yp) . P(s’, Yk) ;
P(Yf | s) P(s, Yk | s’, Yp) P(s’, Yp) formülünü elde ederiz. Burada ilk kısım gelecek olan verinin, son olan kısım geçmişteki durumu, ortadaki kısım ise geçiş durumun simgeler.

MAP kod çözücü algoritmasının 3 adet metriği vardır.

Bunlar;

$\alpha_{k-1}(s')$ = P(s’, Yp) ,

β_k(s) = P(Yf | s) ve

¥$_k$(s’, s) = P(s, Yk | s’, Yp) ‘dir

Sonuç olarak formülümüz; P(s’, s, Y) = $\alpha_{k-1}(s')$. β_k(s) . Y_k(s’, s) şeklini almıştır.

Log-likelihood oranı, L(Uk) ise;

$$\sum_{Uk=+1} \alpha_{k-1}(s') \cdot \beta_k(s) \cdot ¥_k(s', s) \Big/ \sum_{Uk=-1} \alpha_{k-1}(s') \cdot \beta_k(s) \cdot ¥_k(s', s)$$

olarak bulunur.

Son olarak verilerın ne kadar dallanabileceğini, yani verilerin ne ölçüde bozulabileceğini hesaplayan branch metrik adı verilen bir hesap daha bulunmaktadır. Branch metriğin formülü;

$\delta_k^{i,m}$ = 0.5exp($x_k u_k^i + y_k v_k^{i,m}$) olarak kullanılmaktadır. Burada 0.5 değeri toplamın ortalaması için kullanılmaktadır. Parantez içindeki diğer veriler ise giriş ve çıkış verilerının farklı durumlara göre(m) ve farklı zamana göre (k) sayısal değerleridir.

Ölçütlerin İsimleri

$\alpha_{k-1}(s')$ = İleri Ölçütü (Forward Metric)

β_k(s) = Geri Ölçütü (Backward Metric)

¥$_k$(s', s) = İletim Ölçütü (Transmision Metric)

$\delta_k^{i,m}$ = Dallanma Ölçütü (Branch Metric)

Markov süreci özelliklerini taşıyan ayrık bir kaynağın t anındaki durumu St ve aynı andaki çıkısı Xt ile gösterilsin. t anından t' anına kadar geçen sürede bu durum dizisi, $S_t^{t'}$ = St, St+1, …, St' ve kaynak çıkışları ise, $X_t^{t'}$= Xt, Xt+1, …, Xt' olsun. Buna göre M tane farklı durum içeren kaynağın, m durumundan m' durumuna geçişi, olasılık işlevi olarak, P_t(m\ m')= P(S_{t-1} =m\ S_t = m') şeklinde ifade edilebilir. Yani S_t ve S_{t-1} şeklinde tanımlanan iki durum, tek bir X_t çıkışını göstermektedir. Buna göre, X_t çıkışının gerçekleşme olasılığı q_t(X \ m', m) = P(X_t =X \ S_{t-1} =m', S_t =m) eşitliği ile S_t ve S_{t-1} durumlarına bağlı olarak ifade edilir

Ayrık belleksiz bir kanal için, başlangıç durumu, S_0=0 olan ve τ zaman sonra, S_τ = 0 durumuna ulaşan bir kaynak, X_1^τ çıkışlarını üretsin. Bu çıkış değerleri, kanaldaki gürültüden dolayı, algılayıcıda Y_1^τ şeklinde elde edilir.

MAP algoritması, gözlemlenen Y_1^τ değerlerini kullanarak, P(S_t = m \ Y_1^τ)ve P(S_{t-1}= m', S_t =m \ Y_1^τ) sonsal olasılıklarını bulmaya çalışır. Λ_t(m) = P(S_t =m, Y_1^τ) ve σ_t(m', m) = P(S_{t-1} = m', S_t =m, Y_1^τ) şeklinde tanımlanan ortak olasılıklar, P(Y_1^τile bölünürse, yukarıda tanımlanan durum ve geçiş olasılıkları için sonsal olasılıklar elde edilir.

BCJR algoritması yapısı gereğince, t anındaki durum değeri ile o ana kadar gözlemlenen çıkış değerlerinin ortak olasılığı, $\alpha_t(m)$ = P(S_t = m, Y_1^T) eşitliği ile; t anındaki durum bilinirken o andan sonra gözlemlenecek çıkış değerleri koşullu olasılığı, β_t(m)= P(Y_{t+1}^T \ S_t = m) olarak tanımlanır. Aynı zamanda, t-1 anındaki durum değeri bilinirken, t anındaki durum değeri ve o andaki çıkış değerinin olması olasılığı, P(S_t = m, Yt \ S_{t-1}= m') ile ifade edilir.

Tanımlanan bu olasılık değerleri kullanılarak sonsal olasılıklar hesaplanır. Bu sonsal olasılıkların bulunabilmesi için öncelikle ortak olasılıkların hesaplanması gerekir. Markov süreçlerinin özellikleri ve yukarıda tanımlanan α, β ve Y değerleri kullanılarak, durum sonsal olasılığı için, Λ_t(m) = P(S_t =m, Y_1^T) eşitliği aşağıda gösterildiği gibi elde edilir.

Λ_t(m) = P(S_t =m, Y_1^T)
= P(S_t =m, Y_1^T, Y_{t+1}^T)
= P(Y_{t+1}^T\ S_t = m, Y_1^T) P(S_t =m, Y_1^T)
= P(Y_{t+1}^T\ S_t = m) P(S_t =m, Y_1^T)
=β_t(m) $\alpha_t(m)$

Geçiş sonsal olasılıkları için σ_t(m', m)= P(S_{t-1} = m', S_t =m, Y_1^T) eşitliği ise;

σ_t(m', m) = P(S_{t-1} = m', S_t =m, Y_1^T)
= P(S_t = m, S_{t-1}=m', $Y_1^{t-1}, Y_t, Y_{t+1}^T$)
= P(Y_{t+1}^T) \ S_t = m, S_{t-1}=m', Y_1^{t-1}, Y_t) P(S_t = m, S_{t-1}=m', Y_1^{t-1}, Y_t)
= P(Y_{t+1}^T \ S_t = m) P(S_t = m, Y_t \ S_{t-1}=m', Y_1^{t-1}, Yt)P(S_{t-1}=m', Y_1^{t-1})
= β_t(m) ¥$_t$(m', m) $\alpha_{t-1}(m')$

şeklinde ifade edilir.

İleri yöndeki olasılıklar, $\alpha_t(m)$; geri yöndeki olasılıklar ise, β_t(m) değerlerinin hesaplanmasıyla elde edilir. Buna göre, toplam olasılık kuramı ve Markov süreçleri özellikleri kullanılarak, $\alpha_t(m)$ = P(S_t = m, Y_1^T) eşitliği özyineli olarak;

$\alpha_t(m)$ =$\sum_{m'}$P(S_{t-1} = m', S_t = m, Y_1^{t-1}, Y_t)

$= \sum_{m'} \mathrm{P}(S_t = \mathrm{m}, Y_t | S_{t-1} = \mathrm{m}', Y_1^{t-1}) \mathrm{P}(S_{t-1} = \mathrm{m}', Y_1^{t-1})$

$= \sum_{m'} \mathrm{P}(S_t = \mathrm{m}, Y_t | S_{t-1} = \mathrm{m}') \mathrm{P}(S_{t-1} = \mathrm{m}' | Y_1^{t-1})$

$= \sum_{m'} ¥_t(\mathrm{m}', \mathrm{m}) \alpha_{t-1}(m')$

ve $\beta_t(\mathrm{m}) = \mathrm{P}(Y_{t+1}^{\mathrm{T}} \mid S_t = \mathrm{m})$ eşitliğide yine aynı şekilde özyineli olarak;

$\beta_t(\mathrm{m}) = \sum_{m'} \mathrm{P}(S_{t+1} = \mathrm{m}', Y_{t+1}, Y_{t+2}^{\mathrm{T}} \mid S_t = \mathrm{m})$

$= \sum_{m'} \mathrm{P}(Y_{t+2}^{\mathrm{T}} | S_{t+1} = \mathrm{m}', Y_{t+1}, S_t = \mathrm{m}) \mathrm{P}(S_{t+1} = \mathrm{m}', Y_{t+1} | S_t = \mathrm{m})$

$= \sum_{m'} \mathrm{P}(Y_{t+2}^{\mathrm{T}} | S_{t+1} = \mathrm{m}') \mathrm{P}(S_{t+1} = \mathrm{m}', Y_{t+1} | S_t = \mathrm{m})$

$= \sum_{m'} \beta_{t+1}(\mathrm{m}') ¥_{t+1}(\mathrm{m}, \mathrm{m}')$

şeklinde ifade edilir.

3.6.2 MAP ile SOVA Algoritmalarının Farkı

Günümüzde Turbo kodlama için kullanılan en temel iki adet algoritma bulunmaktadır. Bunlarda biri MAP diğeri ise, SOVA algoritmasıdır. Sova algoritmasını Viterbi bulmuştur. Viterbi yönteminin temeli; girişe gelen veriyi tamamen ve tek seferde alarak hesaplanması ve bunun sonucunda bir çıktı elde etmesidir.

Günümüzde; MAP ve SOVA algoritmalarının kullanım oranları yaklaşık olarak aynıdır. Bunun nedeni, SOVA algoritmasının daha az güvenilir olmasına karşın, MAP algoritmasına göre çok daha kolay gerçekleştirilen bir algoritma olmasıdır. MAP algoritmasının tercih edilmesinin sebebi ise, SOVA'ya göre çok daha zor gerçekleştirilir bir algoritma olmasına karşın, sonuçlar açısından çok daha güvenilir bir yöntem olmasıdır.

SOVA ve MAP algoritmalarının başlıca farkı aşağıda verilmiştir;

SOVA, Viterbi algoritmasını kullanır. Bu algoritma giriş yapan verinin bütününü ele alır ve bunun üzerinden işlem yapar.

MAP yöntemi ise giriş yapan verinin her bir bit değeri üzerinden işlem yapar. Buda SOVA algoritmasını MAP algoritmasına göre çok daha kolay bir yöntem yapar ancak MAP algoritması çok daha ayrıntılı işlemler yaptığı için, SOVA algoritmasına göre daha güvenilir ve daha doğru sonuçlar üretir. Bu farklardan dolayı, bu araştırmada MAP algoritmasının kullanması tercih edilmiştir.

3.6.3 Shannon Limiti

Toplanır beyaz gauss gürültülü (AWGN) kanallarda ne kadar kodlama kazancı elde edileceği merak edilmiş ve bu konuda araştırmalar yapılmıştır. Claude Shannon 1948'te "A Mathematical Theory of Communication" adlı makalesinde bir iletim kanalının kapasitesini aşağıdaki denklemle ifade etmiştir .

$$C = W \log_2(1 + E_S / N_O) \text{ bps (bit/saniye)}$$

Burada;

W, kanaldaki band genişliğidir ve Hertz olarak ifade edilir. E_S / N_O değeri ise, sinyal gürültü oranıdır. C ise, kanal kapasitesidir. Shannon limiti teorik anlamda bir kodlama kanalında elde edilecek en yüksek kodlama kazancını gösterir. Shannon limiti, teorik olarak kanal kapasitesinin çok düşük bir kısmında hata kodları kullanılarak kullanıcıda en az hatayla verinin elde edilebileceğini göstermiştir. Teoride Shannon limiti -1.59 dB 'dir. Bu değere günümüzde hala ulaşılabilen çok az sayıda çalışma bulunmamaktadır. Bu limiti elde eden çalışmalardan bazıları, NASA'nın en gelişmiş şartlarda yaptığı Mars ile haberleşme uygulamalarıdır.

Shannon limitine yaklaşabilen en başarılı çalışmalardan birisi 1993 yılında Berrou, A. Glavieux and P. Thitimajshima tarafından yapıldı. Turbo kodlar diğer bir deyişle paralel olarak birleştirilmiş evrişimli kodlar tanıtıldı. Bu kodlar tam band genişliğine (full bandwidth) çok yakın oranlarda çok düşük bit hata oranları göstermektedir.Ayrıca çözümleme karmaşıklığı da katlanılabilir miktardadır. Soft-in soft-out kestirim algoritmalarının kullanımı bu başarıdaki anahtar faktörlerinden biridir.

4 PROGRAMLANABİLİR MANTIK

Programlanabilir mantık devreler kapıların ve flip-flopların birbirlerine bağlanmasıyla oluşturulan devreler şeklinde basitçe tanımlanabilir. Bellek hücreleri mantık kapıların gerçekleştirdiği işlevlerin tanımlanmasında, denetiminde ve birbirleriyle olan giriş-çıkış bilgisi ilişkilerinin kayıtlı tutulmasında kullanılır. Bu alanda üretilen çoğu ürünün farklı mimarilerde tasarlanmalarına karşın mantık olarak hepsi aynı temel prensiplerle çalışır.

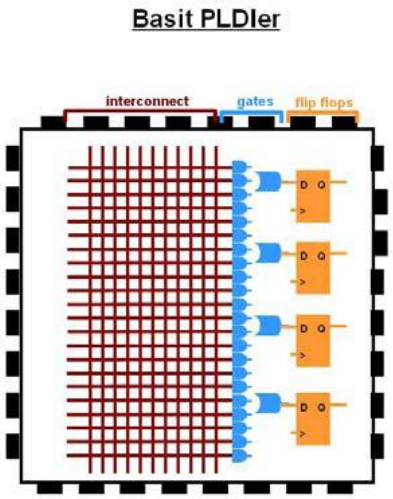
4.1 Programlanabilir Mantık Ürünleri Ve Aralarındaki Farklar

Günümüzde çok çeşitte programlanabilir mantık ürünleri mevcuttur. Bunları üretim tasarımlarına göre değerlendirdiğimizde her tasarımın altında yine tasarımlarına göre farklı ürünler göze çarpmaktadır. Sınıflandırmalar sonucunda temel tipler sayılabilir;

·SPLDs(Simple Programmable Logic Devices)
·CPLDs(Complex Programmable Logic Devices)
·FPGAs(Field Programmable Gate Arrays)

4.1.1 Basit Programlanabilir Mantık Devresi

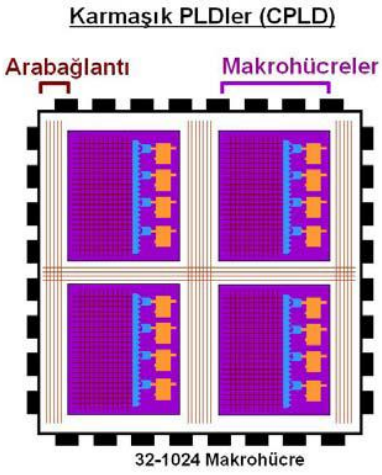
SPLD'ler teknolojilerine göre farklı adlarla da bilinirler. Bunlar,
• PAL (Programlanabilir Dizi Mantık)
• PLA (Programlanabilir Mantık Dizi)
• PLD (Programlanabilir Mantık Aygıt)
SPLD'ler kapasiteleri en düşük, bunun sonucu olarak da en ucuz programlanabilir mantık ünitelerdendir. Bir SPLD ünitesinde 4 ile 22 arası hücre vardır. SPLD'ler bu özelliklerinden ötürü sadece 7400 serisi TTL ürünlerine göre tercih edilirler. SPLD'lerdeki her hücrenin bir diğeri ile direk olarak bağlantısı vardır. SPLD'lerdeki hücrelerin yapımında genellikle sigorta, EPROM, EEPROM veya FLASH gibi değiştirilemeyen bellek hücreleri kullanılır.



Şekil 4.1. Basit PLD Yapısı

4.1.2 Karmaşık Programlanabilir Mantık Devre

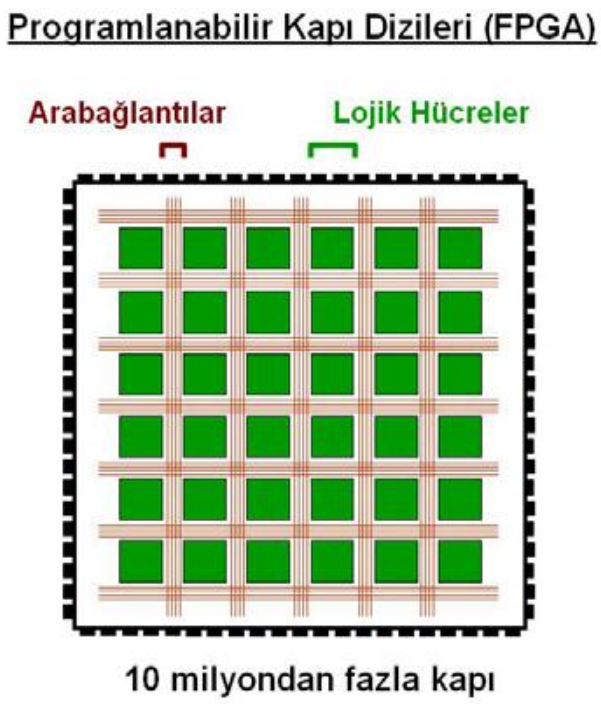
CPLD'ler SPLD'lerle hemen hemen aynıdırlar. CPLD'ler sadece kapasiteleri bakımından SPLD'lerden üstündürler. Bir CPLD in kapasitesi hakkında SPLD'lerin kapasitelerinin 2 ile 64 katı arasındadır denebilir. CPLD'lerde yüzlerce hücre vardır. Bu hücrelerin her CPLD'nin modeline göre 8 ile 16 arasında değişen her bir grubu bir işlev bloğunda toparlanmış ve içerisinde hepsine direk bağlantıları yapılmıştır. Bu işlev blokları arasında da iletişim sağlanmıştır. Fakat CPLD'lerdeki işlev blokları arası bu iletişim türünde kullanılan mantık, dolayısıyla da CPLD'nin hızı üretici firmasına göre değişir.



Şekil 4.2. Karmaşık PLD Yapısı

4.1.3 Programlanabilir Mantık Dizileri

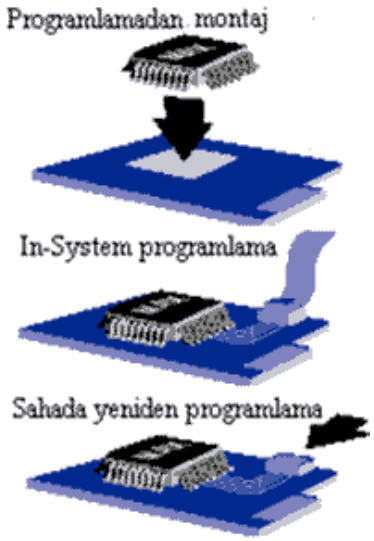
FPGA, programlanabilir mantık blokları ve bu bloklar arasındaki ara bağlantılardan oluşan ve geniş uygulama alanlarına sahip olan sayısal tümleşik devrelerdir. Tasarımcının ihtiyaç duyduğu mantık işlevlerini gerçekleştirme amacına yönelik olarak üretilmiştir. Dolayısıyla her bir mantık bloğunun işlevi kullanıcı tarafından düzenlenebilmektedir. FPGA ile temel mantık kapılarının ve yapısı daha karmaşık olan devre elemanlarının işlevselliği artırılmaktadır. Sahada programlanabilir ismi verilmesinin nedeni, mantık bloklarının ve ara bağlantıların imalat sürecinden sonra programlanabilmesidir.



Şekil 4.3. Programlanabilir Kapı Dizileri

4.1.4 Sahada Yeniden Programlanabilir Yapı

Her ikiside aygıtın devre üzerinde diğer elemanlarla birlikte programlanabileceği özelliğini vurgulamaktadır. Hemen hemen tüm ISP (sahada yeniden programlanabilir) aygıtları SRAM, EEPROM veya FLASH teknolojileri kullanılarak yapılandırılmıştır.



Şekil 4.4. Montaj ve Sahada Programlanmanın Şematik Gösterimi

4.2 FPGA Teknolojisi ve Kullanımı

Programlanabilir mantık ürünleri (Programmable Logic Devices: PLD), sayısal devrelerin tasarlanmasında uzun zamandan beri önemli kolaylıklar sağlamaktadırlar. PLD teknolojisinin günümüzde en önemli uygulaması Sahada Programlanabilir Kapı Dizileri (Field Programmable Gate Array: FPGA)'dir. Bazı üretim firmaları tarafından "do-it-yourself processors" sloganıyla lanse edilen bu yeni ürün sayesinde eğitim ve çalışma dünyasında bir çok insanın kendi işlemcilerini eskisinden daha kolay bir şekilde oluşturma şansları doğmuştur. FPGA'lar söz edilen kullanım alanlarının yanısıra sayısal tasarım ve bilgisayarların donanım mimarilerini öğrenmek isteyenler için kolaylık sağlamaktadır. FPGA'ların her bakımdan başarım gösterimi nedeniyle değişik kullanım alanları doğmuştur. FPGA'lar üzerinde devre tasarlarken, mevcut olan VHDL, Verilog, Şematik tasarım yöntemlerinden devre tasarım ve tanımlama dili olan VHDL tercih edilmiştir. Bu

sayede ileri donanım bilgisine ihtiyaç olmadan ve çok karmaşık tasarımların, belli bir ölçüde daha kolay tasarlanması mümkün olabilmektedir.

VHDL dili, yapı olarak farklı bir dildir. Öğrenilmesi diğer yazılım dillerine göre çok daha zordur. Ama bu dili öğrenebilmek için donanım bilgisine ihtiyaç yoktur. Buda VHDL dilinin, bu tür devre tasarımlarında en çok tercih edilen dil olmasına neden olmaktadır. Bu durum; donanım bilgisi olmayanlarında donanım oluşturup uygulamasına olanak tanır. FPGA teknolojisinin en büyük eksiği, kapı gecikmesi olarak isimlendirilen, işlem yaparken, paralel işlemlerin aynı anda bitmemesi durumunda, sonucun en son işlem bittikten sonra gösterilmek zorunda olmasıdır.

4.2.1 FPGA Mimarisi

Üretim Teknolojileri

FPGA yapısının üretiminde günümüze kadar birçok farklı mimari kullanılmıştır. Bunlardan en son geliştirilen ve günümüzde kullanılan mimari, SRAM tabanlı mimaridir.

SRAM Tabanlı Mimari

FPGA'in üstünlüklerinden en önemlisi SRAM yapılandırma hücreleri kullanmasıdır. Bu hücreler aygıtın tekrar kullanılmasını sağlar. Bu sayede yeni tasarımlar çok kolay bir şekilde hazırlanıp test edilebilir. Ayrıca, sistem içerisinde ana görevinden önce farklı görevleri yerine getirmesi için programlanabilir. Örneğin sistemin ilk açılış anında farklı bir işlem yapması için programlanabilir. Açılış işlemleri bittikten sonra ise asıl görevini gerçekleştirecek şekilde programlanabilir.

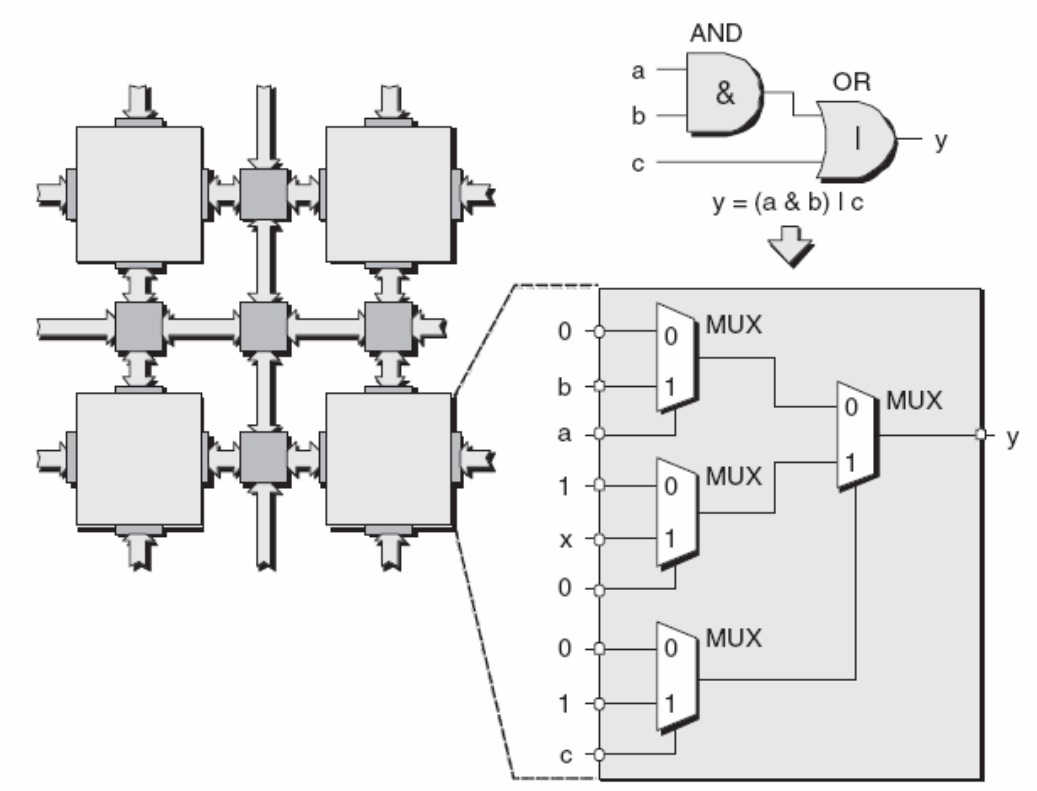
SRAM tabanlı FPGA'lerin diğer önemli bir avantajı ise önü açık bir teknoloji olmasıdır. Birçok bellek üreticisi müşterilerinin istekleri doğrultusunda bu konu üzerinde önemli araştırma ve geliştirme faaliyetlerinde bulunmaktadırlar. Yongada kullanılan diğer birimlerle aynı CMOS teknolojisine sahip olduklarından geliştirilme süreçlerinde ek bir işlem gerektirmezler. Günümüzde boyut, karmaşıklık ve

düzenlilik ölçütleri ele alınarak bu alanda FPGA kullanımı artmıştır. FPGA'in bellek birimlerine karşı diğer bir avantajı ise, bir hata oluşması durumunda FPGA yapısının hata bulma ve düzeltme işlemlerini kısaltmasıdır. SRAM tabanlı FPGA'in anlatılan avantajlarına rağmen bazı dezavantajları da vardır. FPGA her sistem açılışında tekrar yapılandırılmak zorundadır. Bu nedenle sistemde dış bir bellek bulunması gerekmektedir. SRAM tabanlı mimariler için farklı hücre tabanlı mimariler bulunmaktadır. Bunlardan günümüzde kullanılan başlıca ikisi, MUX ve LUT tabanlı hücrelerdir.

4.2.2 Programlanabilir Hücre Mimarileri

MUX Tabanlı Hücre

Üç girişli y = (a & b) | c işlevinin sadece çoğullayıcılardan (multiplexer) oluşan bir blokla nasıl gerçeklenebileceği Şekil 4.5'te verilmiştir.

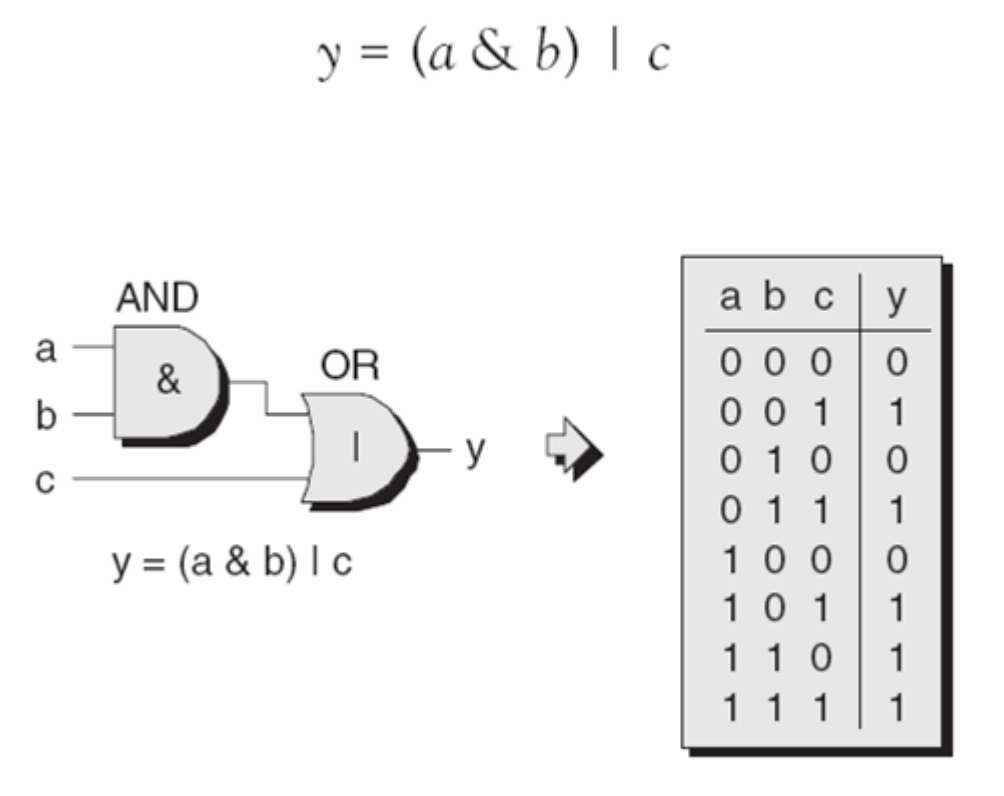


Şekil 4.5. MUX Tabanlı Hücre

Bu blok girişlere verilen mantık 0, mantık 1 ve asıl girişler olan a, b, c ve onların tümleyenlerinin girişe direk verilmesi ile başka bir bloğun çıkışının bağlanması yapılandırılabilir. Bu yöntem her bloğun bir işlevi oluşturması için sayısız yol sağlar.

LUT (Çevirim Tablosu) Tabanlı Hücre

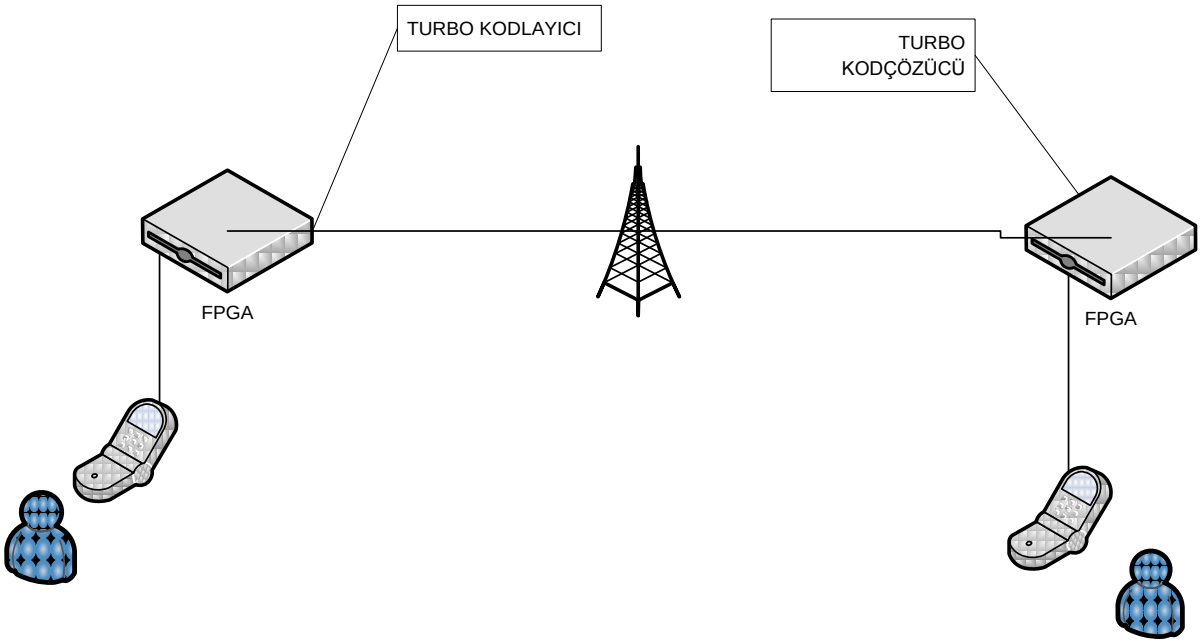
Bu yapıda giriş işaretleri başvuru tablosundan (lookup table) doğru çıkışı bulmak için işaretçi olarak kullanılır. Girişlerin alabileceği her değer için tabloda bir çıkış değeri bulunur. y = (a & b) | c işlevunu bu kez LUT tabanlı mimaride gerçeklemek istersek oluşturmamız gereken başvuru tablosu Şekil 4.6'daki gibi olmalıdır.



Şekil 4.6. LUT Tabanlı Hücre

5 TURBO KODLAMA DENEYSEL UYGULAMASI

FPGA ile Turbo kodlamanın günümüzde kullanımı Şekil 5.1'de görüldüğü şekildedir. Burada, iki adet birbirleri ile bağlantılı FPGA devresi bulunmalıdır. Bunlardan biri kodlama için, diğeri ise kod çözücü için gerekmektedir.

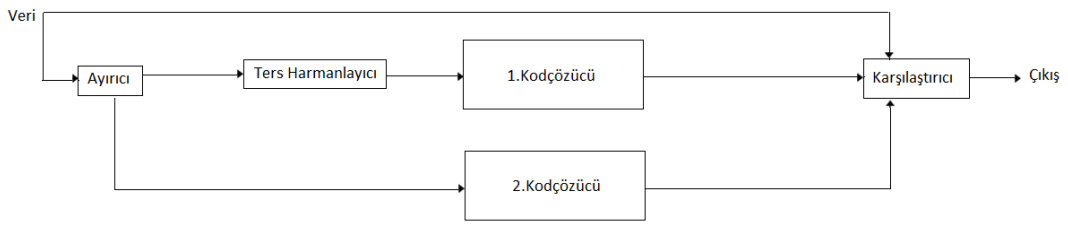


Şekil 5.1. Turbo Kodlama Donanımsal Durumu

5.1 Turbo Kod Çözücünün İşleyişi

Daha öncede açıkladığım gibi iki adet yöntem vardır.
Bunlar;
a)SOVA Kod Çözücü
b)Maksimum Sonsal Olasılık(MAP) yöntemleridir.

Bu bölümde yapılan örnek, MAP algortimasına göre yapılmıştır. Giriş ve çıkış değerleri, FPGA devresi üzerinde aynı ortamda gerçekleştirilmiştir. Gürültü olarakda, gauss gürültüsü rassal olarak bitlere eklenmiştir.



Şekil 5.2 Turbo Kod Çözücü Genel Yapısı

Bu bölümde, K=3 ve r=1/2 bir kodun MAP algoritmasıyla çözülmesi anlatılmıştır. [00] , [10], [01], [11] durumlarına karşılık gelen a, b, c, d sembollerini bu örnekte kullanıldı.

Giriş serisi olarak d=[1 0 0] alınmıştır. Çıktılar u1=d=[1 0 0], u2=v=[1 0 1] olarak bulunur. Yani çıktımız [1 1 0 0 0 1] olarak bulunur. Bu çıktıyı bipolar formda ifade edersek [+1 +1 -1 -1 -1 +1] olarak buluruz. x=u1+n1 (n1 gürültü olmak üzere) ve y=v+n2 olduğunu varsayalım. Bu gürültünün u1 ve u2 çıktılarnı (bipolar formda) x=[1. 4 0. 4 -0. 4] ve y=[0. 8 0.2 1.2] şeklinde etkilediğini varsayalım. Bu değerlerin, yumuşak karar veren algılayıcı tarafından MAP kod çözücüye iletildiğini düşünelim.

Burada n1 ne n2 değerleri Toplamsal beyaz Gauss gürültüsündür (AWGN - Additive White Gaussian Noise). Burada n1 ve n2 değerleri, Gauss dağılımlı kanal değerlerini bulan bir işlev (LUT) kullanılarak bulunmuştur. Burada her girişe ayrı ayrı eklenen n1 ve n2 gauss gürültü değerleri, standart sapma ve beklenen değer içeren gauss formülü kullanılarak, o aralıkta (maksimum ve minimum değer arası), olasılıklara göre seçilen rastgele iki değerdir.

İlk olarak ; $\delta_k^{i,m}$ = 0. 5 exp($x_k u_k^i + y_k v_k^{i,m}$)
Formülünde k=1, k=2 ve k=3 için dallanma ölçülerini (Branch Metric) hesaplarız.

k=1 için;

$\delta_{k=1}^{1,m=a}$ = $\delta_{k=1}^{1,m=b}$ =0.5 exp[$(1.4)(1) + (0.8)(1)$] = 5.0

$\delta_{k=1}^{0,m=a} = \delta_{k=1}^{0,m=b} = 0.5 \exp[(1.4)(-1) + (0.8)(-1)] = 0.05$

$\delta_{k=1}^{1,m=c} = \delta_{k=1}^{1,m=d} = 0.5 \exp[(1.4)(1) + (0.8)(-1)] = 1.0$

$\delta_{k=1}^{0,m=c} = \delta_{k=1}^{0,m=d} = 0.5 \exp[(1.4)(-1) + (0.8)(1)] = 0.25$

k=2 için;

$\delta_{k=2}^{1,m=a} = \delta_{k=2}^{1,m=b} = 0.5 \exp[(0.4)(1) + (0.2)(1)] = 1.0$

$\delta_{k=2}^{0,m=a} = \delta_{k=2}^{0,m=b} = 0.5 \exp[(0.4)(-1) + (0.2)(-1)] = 0.25$

$\delta_{k=2}^{1,m=c} = \delta_{k=2}^{1,m=d} = 0.5 \exp[(0.4)(1) + (0.2)(-1)] = 0.67$

$\delta_{k=2}^{0,m=c} = \delta_{k=2}^{0,m=d} = 0.5 \exp[(0.4)(-1) + (0.2)(1)] = 0.37$

k=3 için;

$\delta_{k=3}^{1,m=a} = \delta_{k=3}^{1,m=b} = 0.5 \exp[(-0.4)(1) + (1.2)(1)] = 0.91$

$\delta_{k=3}^{0,m=a} = \delta_{k=3}^{0,m=b} = 0.5 \exp[(-0.4)(-1) + (1.2)(-1)] = 0.27$

$\delta_{k=3}^{1,m=c} = \delta_{k=3}^{1,m=d} = 0.5 \exp[(-0.4)(1) + (1.2)(-1)] = 0.08$

$\delta_{k=3}^{0,m=c} = \delta_{k=3}^{0,m=d} = 0.5 \exp[(-0.4)(-1) + (1.2)(1)] = 3.0$

olarak buluruz. Bu adımdan sonra aşağıdaki formülle ileri durum (forward) ölçülerini hesaplarız (tipik olarak a=00 konumundan başladığını varsayarak),

$\alpha_{k+1}^{m} = \sum_{j=0}^{1} \delta_k^{j,b(j,m)} \alpha_k^{b(j,m)}$ formülünde kullanabiliriz.

$\alpha_{k+1}^{m=a}$=1. 0 ve $\alpha_{k+1}^{m=b}$=$\alpha_{k+1}^{m=c}$=$\alpha_{k+1}^{m=d}$=0 olarak kabul ederiz. Çünkü, başlangıç durumu a=00 olduğu için, bu durumu bir ve dışında kalan b, c ve d durumlarını sıfır alırız. Bunun sebebi başlangıç durumunun işlemlere başlandığı ilk adım olmasıdır.

k=2 için;

$\alpha_{k=2}^{m=a}$=(0. 05)(1. 0) + (0. 25)(0) = 0. 05

$\alpha_{k=2}^{m=b}$=(5. 0)(1. 0) + (1. 0)(0) = 5. 0

$\alpha_{k=2}^{m=c}$ = $\alpha_{k=2}^{m=d}$ = 0 olarak bulunur.

k=3 için;

$\alpha_{k=3}^{m=a}$= 0. 01

$\alpha_{k=3}^{m=b}$=0. 05

$\alpha_{k=3}^{m=c}$=1. 25

$\alpha_{k=3}^{m=d}$=5. 0

k=4 için;

$\alpha_{k=4}^{m=a}$= 3. 75

$\alpha_{k=4}^{m=b}$=0. 11

$\alpha_{k=4}^{m=c}$=15. 01

$\alpha_{k=4}^{m=d}$=0. 45

şeklinde bulunur.

Aynı şekilde geri durum(backward) ölçüleri;

$\beta_k^m = \sum_{j=0}^{1} \delta_k^{j,m} \beta_{k+1}^{f(j,m)}$ formülü ile hesaplanır.

$\beta_{k=4}^{m=a}$=1. 0 ve $\beta_{k=4}^{m=b} = \beta_{k=4}^{m=c} = \beta_{k=4}^{m=d} = 0$

Burada da en son durumun tipik olarak yine a=00 da biter, böylece son durumda sadece $\beta_{k=4}^{m=a}$ değeri bir olur ama diğer durumlar sıfır alır. Burada durumların başlangıcına ve bitişine bir verilip diğer durumlara sıfır atanmasının sebebi, formülde son durum olan 4. durumu bulmak için k=4 olarak aldığımızda k+1 de 5 olacağı için ve böyle bir durum olmadığından dolayı (aynı şekilde α içinde 0. durum olmadığı için) bu kabullenmeler yapılır.

k=3 için;

$\beta_{k=3}^{m=a}$= (0. 27)(1. 0) + (0. 91)(0) = 0. 27

$\beta_{k=3}^{m=b}$=$\beta_{k=3}^{m=d}$ = 0

$\beta_{k=3}^{m=c}$= (3. 0)(1. 0) + (0. 08)(0) = 3. 0

k=2 için;

$\beta_{k=2}^{m=a}$= 0. 07

$\beta_{k=2}^{m=b}$= 0. 75

$\beta_{k=2}^{m=c}$= 0. 1

$\beta_{k=2}^{m=d}$= 1. 11

k=1 için;

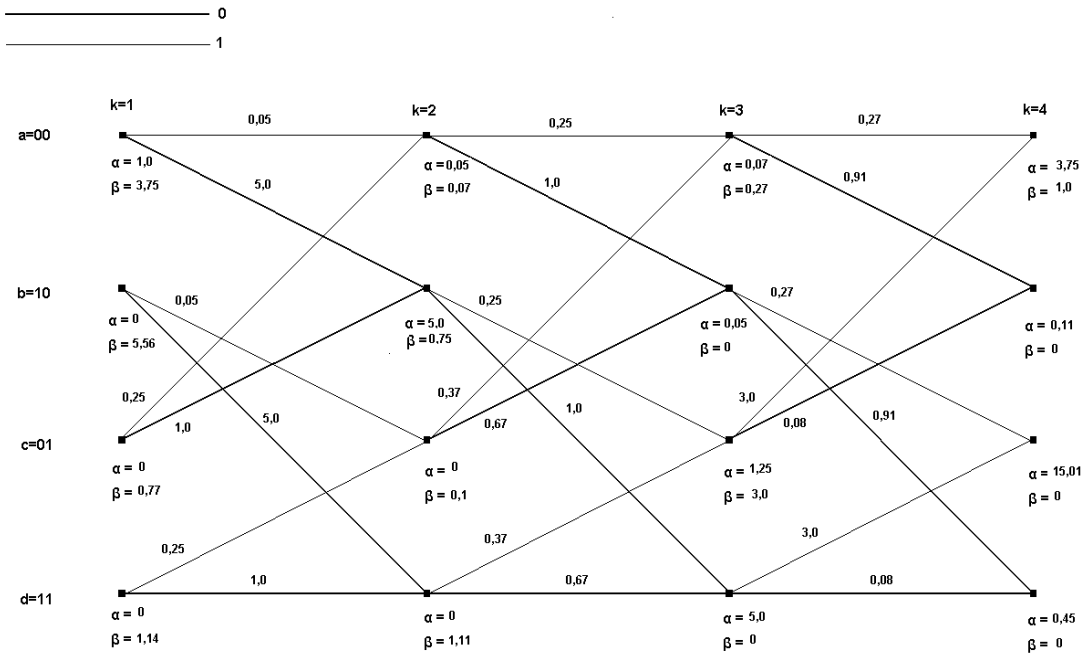
$\beta_{k=1}^{m=a}$= 3. 75

$\beta_{k=1}^{m=b}$= 5. 56

$\beta_{k=1}^{m=c}$= 0. 77

$\beta_{k=1}^{m=d}$= 1. 14 olarak buluruz.

Bu verileri aşağıdaki şekilde görülmektedir.



Şekil 5.3 Turbo Kod Çözücü Trellis Şeması

Bu verileriden yararlanılarak logaritmik olabilirlik oranı (Log-Likelihood Ratio) hesaplanır.

L(d_k) >0 ise d_k = 1 ve L(d_k) < 0 ise d_k = 0

Bu şekilde k=1 için ;

$$L(d_1) = \log\left(\frac{1.0 * 5.0 * 0.75}{1.0 * 0.05 * 0.07}\right) = 3.03$$

k=2 için;

$$L(d_2) = \log\left(\frac{(0.05*1.0*0) + (5.0*1.0*0)}{(0.05*0.25*0.27) + (5.0*0.25*3.0)}\right) = -\infty$$

k=3 için;

$$L(d_3) = \log\left(\frac{(0.01*0.91*0) + (0.05*0.91*0) + (1.25*0.08*0) + (5.0*0.08*0)}{(0.01*0.27*1.0) + (0.05*0.27*0) + (1.25*3.0*1.0) + (5.0*3.0*0)}\right) = -\infty$$

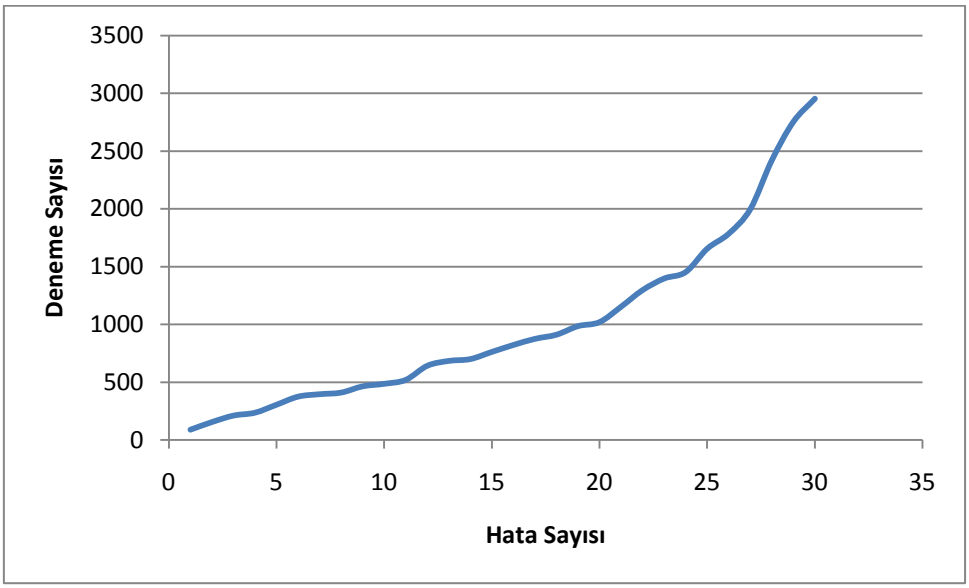
bulunur. Sonuçta giriş dizimiz doğru olarak d=[1 0 0] olarak buluruz.

5. 2 Harmanlayıcının Değişimine Göre Hata ve BER Durumları

2 db İçin Hata Durumları

Burada, A harmanlayıcısı, B harmanlayıcısından daha başarılı olarak ayarlanmıştır. A harmanlayıcısının hamming uzaklığı (6), B harmanlayıcısının hamming uzaklığından (4) daha fazladır.

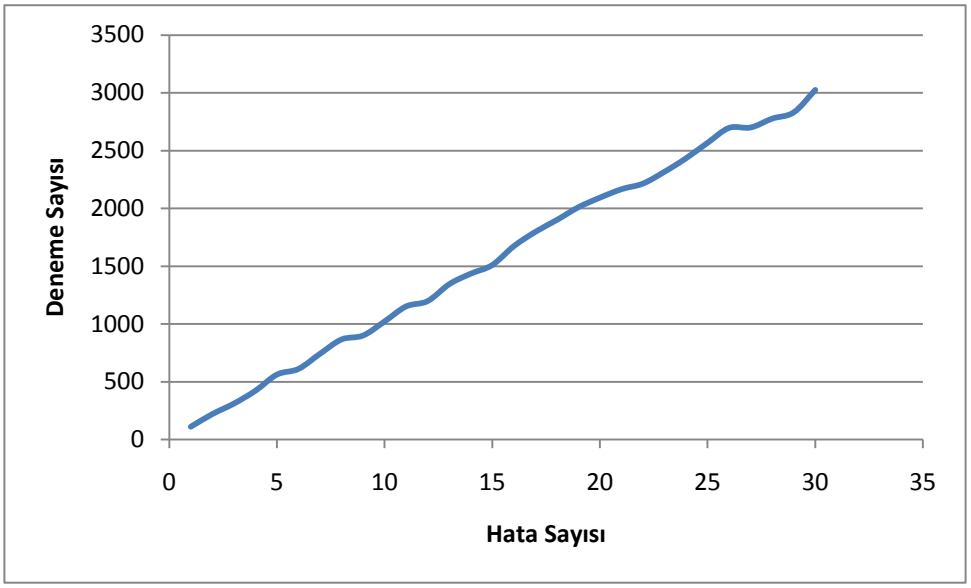
Harmanlayıcı A Durumunda;



Şekil 5.4. A Harmanlayıcısında 2dB için Hata Grafiği

2 dB de hata-deneme sayıları grafiği yukarıdaki gibidir. Burada görüldüğü gibi, yanlış sayısının 30'u bulması yaklaşık 3000 kodlama denemesi kadar sürmüştür.

Harmanlayıcı B Durumunda;

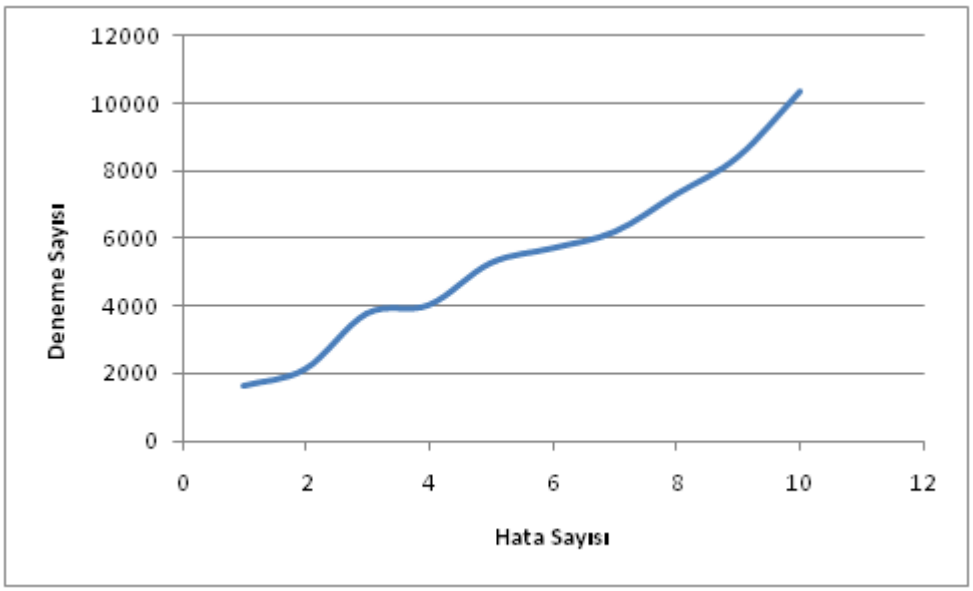


Şekil 5.5. B Harmanlayıcısında 2dB için Hata Grafiği

B harmanlayıcısında ise, yanlış sayısının 30'u bulması yaklaşık 3000 kodlama denemesi kadar sürmüştür.

5 db İçin Hata Durumları

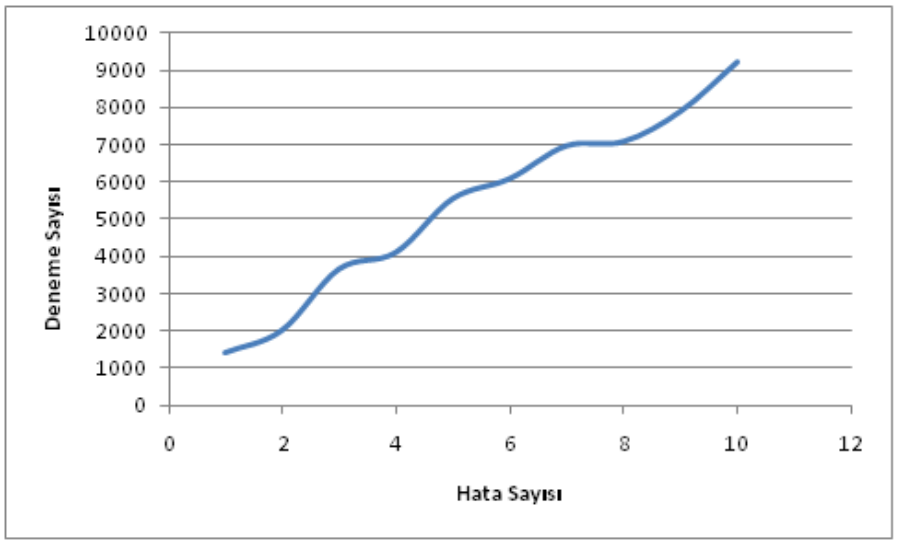
Harmanlayıcı A Durumunda;



Şekil 5.6. A Harmanlayıcısında 5dB için Hata Grafiği

A harmanlayıcısında, yanlış sayısının 10'u bulması 10500 kodlama denemesi kadar sürmüştür.

Harmanlayıcı B Durumunda;

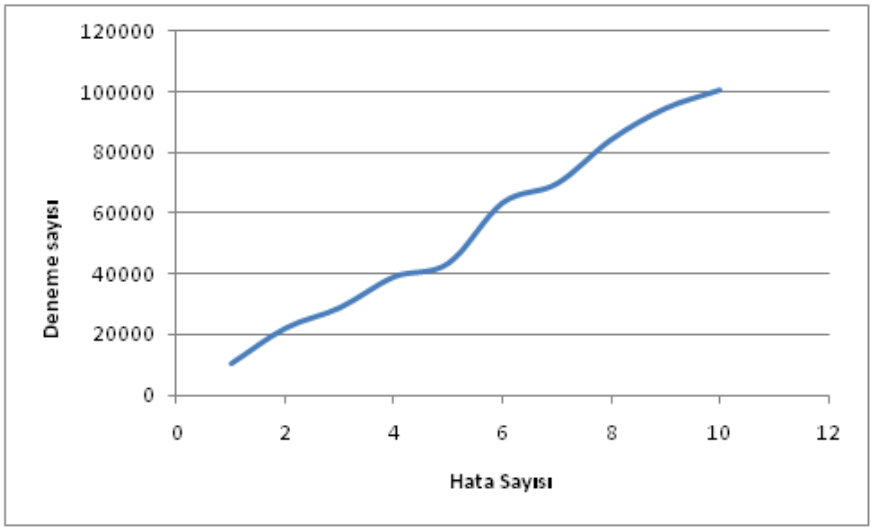


Şekil 5.7. B Harmanlayıcısında 5dB için Hata Grafiği

5 dB de hata-deneme sayıları grafiği yukarıdaki gibidir. Burada görüldüğü gibi, yanlış sayısının 10'u bulması yaklaşık 9500 kodlama denemesi kadar sürmüştür.

10 db İçin Hata Durumları

Harmanlayıcı A Durumunda;



Şekil 5.8. A Harmanlayıcısında 10dB için Hata Grafiği

A harmanlayıcısında, yanlış sayısının 10'u bulması yaklaşık 105000 kodlama denemesi kadar sürmüştür.

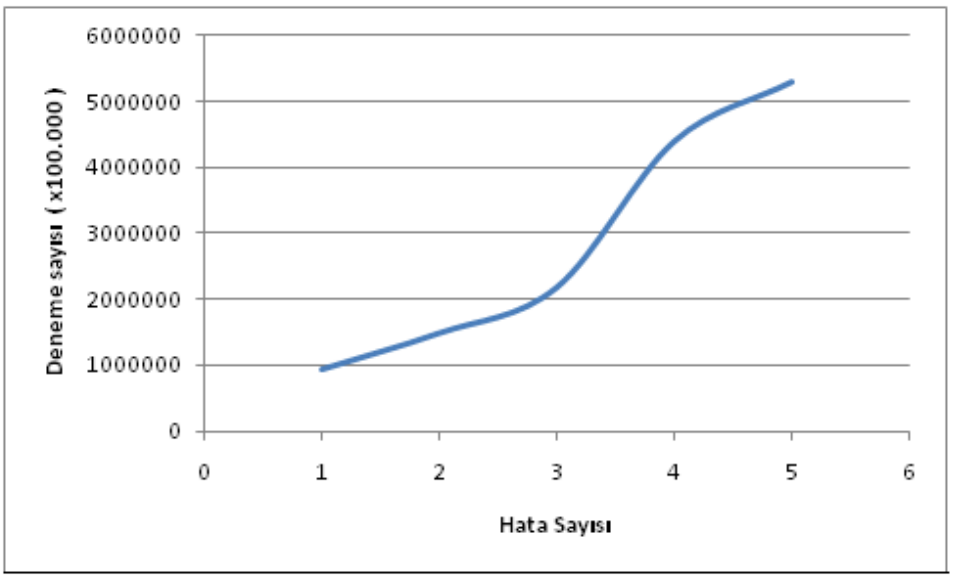
Harmanlayıcı B Durumunda;

Şekil 5.9. B Harmanlayıcısında 10dB için Hata Grafiği

10 dB de hata-deneme sayıları grafiği yukarıdaki gibidir. Burada görüldüğü gibi, yanlış sayısının 10'u bulması yaklaşık 98000 kodlama denemesi kadar sürmüştür.

15 db İçin hata Durumları

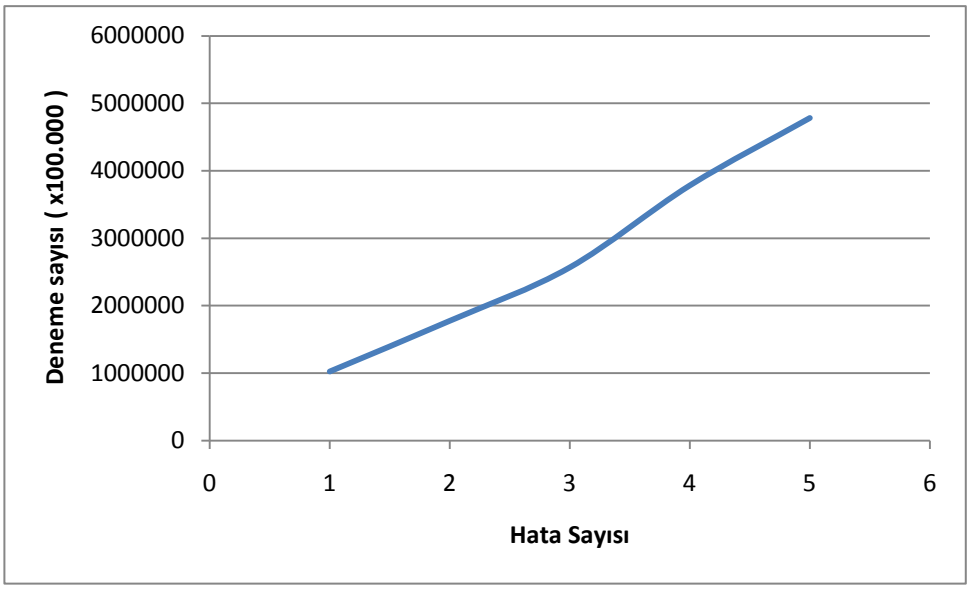
Harmanlayıcı A Durumunda;



Şekil 5.10. A Harmanlayıcısında 15dB için Hata Grafiği

15 dB de hata-deneme sayıları grafiği yukarıdaki gibidir. Burada görüldüğü gibi, yanlış sayısının 5'i bulması yaklaşık 5.2 milyon kodlama denemesi kadar sürmüştür.

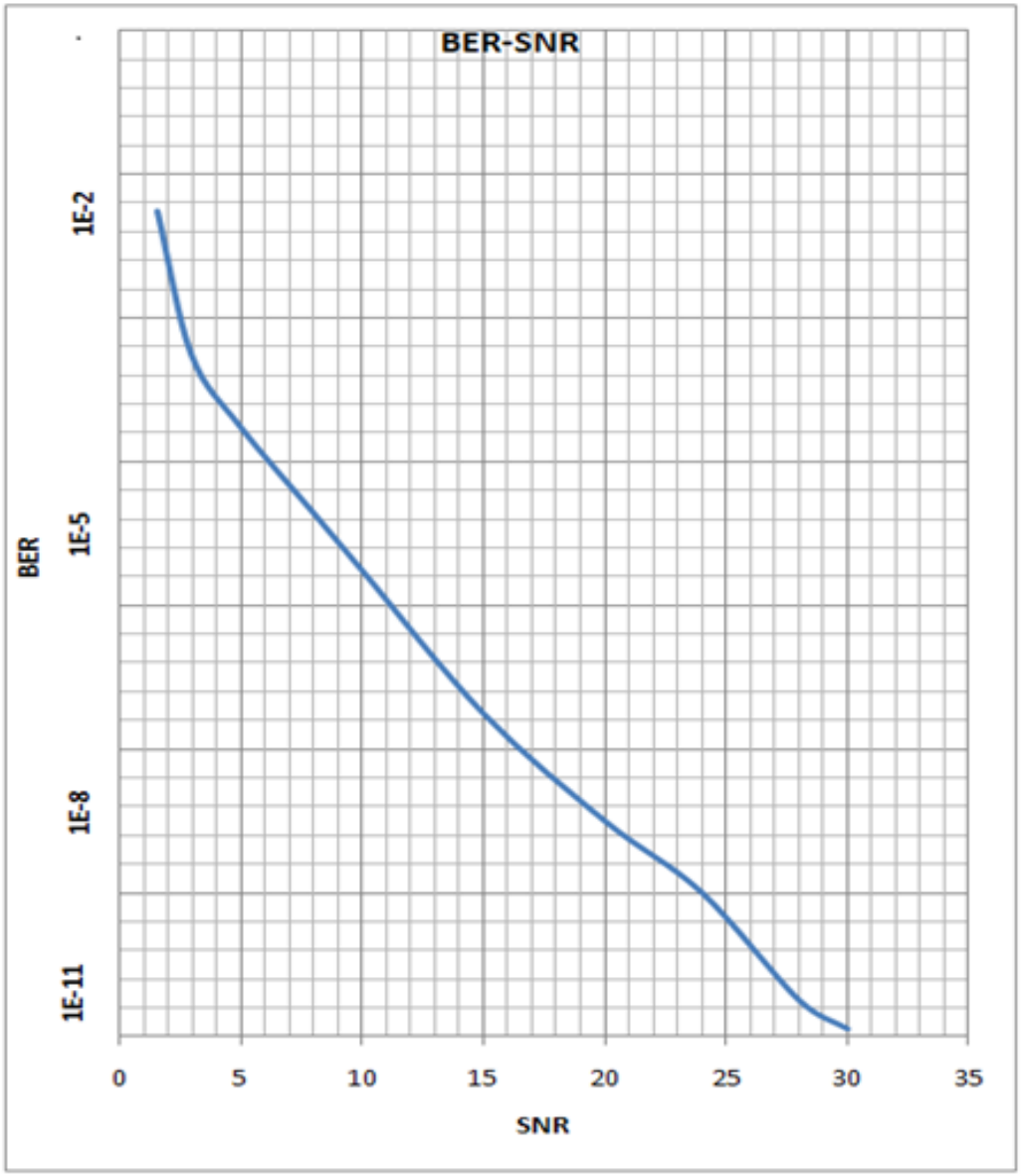
Harmanlayıcı B Durumunda;



Şekil 5.11. B Harmanlayıcısında 15dB için Hata Grafiği

B harmanlayıcısında ise, yanlış sayısının 5'i bulması yaklaşık 4.8 milyon kodlama denemesi kadar sürmüştür. Burada, grafiklerimizin her harmanlayıcıya göre farklılık göstermesinn sebebi, harmanlayıcının karmaşıklığı arttıkça, hata sayısının azalma eğiliminde olamasıdır. Sinyal Gürültü Oranı, SNR olarak adlandırılır. İletişimde en önemli sorunlardan biri gürültüdür. İletilmek istenen bilgi sinyalinin gücü ile bu sinyalin üzerindeki gürültünün gücü arasındaki orana sinyal gürültü oranı (SNR) denir. Bu oran mümkün olduğu kadar düşük tutulursa, kanal kapasitesi kullanım verimliliği (etkin band genişliği) o kadar artar. Harmanlayıcının A ve B durumlarının deneme - hata sayıları grafiklerinden sonra, A ve B harmanlayıcıları için, BER(Bit Eror Rate) grafikleride elde edilmiştir. BER değeri, hatalı kodlama sayısı ile bu hatalı kodlamaların yapıldığı toplam kodlama sayısının bölümünden elde edilir. Böylelikle, her bir hatalı kodlamanın, toplamda kaç deneme başına elde edildiğinin oransal değerini bulabiliriz.

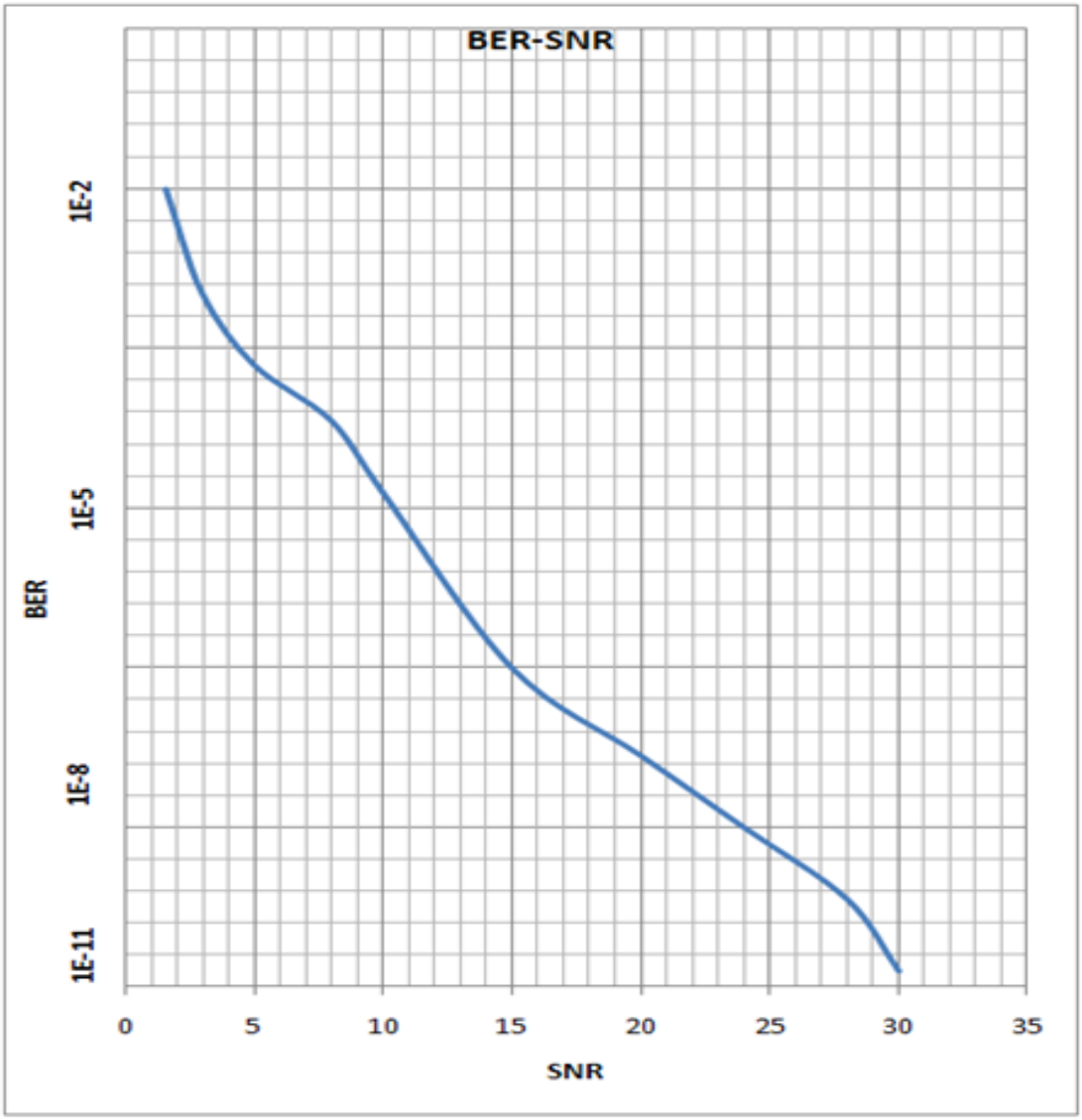
Harmanlayıcı A Durumuna Göre BER/SNR



Şekil 5.12. Harmanlayıcı A Durumuna Göre BER/SNR Grafiği

Sekil 5.12' de görüldügü gibi harmanlayıcının A durumundaki grafiğimizde, turbo kod başarımı oldukça yüksektir. Bu aralıklarda sinyal gürültü oranı artmakta, hata oranı düşmektedir.

Harmanlayıcı B Durumuna Göre BER/SNR



Şekil 5.13. Harmanlayıcı B Durumuna Göre BER/SNR Grafiği

Sekil 5.13' de görüldügü gibi harmanlayıcının B durumunda, turbo kod başarımı oldukça yüksektir. Bu aralıklarda sinyal gürültü oranı artmakta, hata oranı düşmektedir. Yanlız, A durumuna göre, B durumunda düşüş esnasında, bazı dalgalanmalar (oscillations) görülmektedir. Bunun nedeni, giriş verisinin, b harmanlayıcısına göre, A harmanlayıcısında daha yüksek oranda karıştırılmasıdır. Bunun sonucu olarak ta, A harmanlayıcısının BER değerleri, B harmanlayıcısına göre daha başarılıdır.

6. SONUÇ

Haberleşmenin önemi günümüzde sürekli artmakta ve bu artışa bağlı olarak teknik özellikleri sürekli gelişmektedir. Haberleşmede asıl amaç, bilginin sağlıklı bir şekilde iletişiminin gerçekleşmesidir. Kodlama teknikleri, hata analizleri ve hataları düzeltme işlemleri ile haberleşmenin daha sağlıklı ve hızlı bir şekilde yapılmasını sağlamak mümkündür. Bu çalışmayla sayısal haberleşme sistemlerinde kullanılan paralel yapıdaki turbo kodlar ile ilgili araştırmalar, incelemeler ve geliştirmeler yapılmıştır. Turbo kodlamanın, en yüksek başarı oranına sahip algoritması olan MAP algoritması ile ilişkisi ve uyumu incelenip, uygulaması yapılmıştır.

Bu inceleme yapılırken temelden karmaşığa doğru bir yöntem benimsenmiş ve öncelik olarak basit kodlama yöntemleri anlatılmıştır. Ayrıca iletişimin kalitesinin ve güvenilirliğini arttırılması için gereken şartlar ve hata kontrol yöntemleri araştırılmıştır. FPGA devresi üzerinde VHDL kodlama dili ile yazılan program sayesinde, aynı devre üzerinde hem kodlama hem de kod çözücü sistemler çalıştırılmıştır. Günümüz haberleşme ortamına en yakın ortamın sağlanması için kodlanmış verilere, kod çözücüye ulaşmasından hemen önce gauss gürültüleri eklenmiştir. Bu sayede gerçeğine en yakın haberleşme ortamı hazırlanmış ve turbo kodların başarım incelemeleri yapılmıştır.

Bu çalışmada bulunan BER değerleri, FPGA devresi üzerinde8 bitlik iletimde, MAP algoritması kullanılarak Turbo kodlama ve kod çözümünde sırasıyla 2, 5,10 ve 15 dB SNR oranları için gerçekleştirilmiştir. Bunlar 2 dB için 10^{-2} , 5 dB için 10^{-3} , 10 dB için 10^{-5} ve 15dB için de 10^{-11}'dir. Bu değerler, günümüzde FPGA devresi ile 8 bitlik iletimlerde, SOVA algoritması kullanılarak yapılan çalışmalarda bulunan BER değerlerinden daha düşüktür. MAP algoritması ile elde edilen değerler, SOVA algoritması ile elde edilen değerlerden ortalama %30 daha başarılıdır. Bu da MAP algoritmasının, SOVA algoritmasından daha iyi sonuçlar verdiğini göstermektedir.

Ayrıca bu çalışmada, Turbo kodlamada ve kod çözücüde standart olarak kullanılan harmanlayıcının, haberleşmenin doğruluğuyla ve güvenilirliğiyle olan bağlantısıda incelenmiş ve farklı harmanlayıcıların kullanılması durumunda, kodlama başarımlarının karşılaştırılması yapılmıştır. Bu karşılaştırmada, harmanlayıcının bitleri karıştırma oranının, haberleşme başarımına doğrudan etki ettiği görülmüştür. Sekiz bitin altısı karıştırılmış olan harmanlayıcı (A) ile sekiz bitin dördünün karıştırıldığı harmanlayıcının (B), kodlama ve kod çözücüye etkileri ayrı ayrı incelenmiştir. Burada, harmanlayıcının karıştırma oranının kodlamanın başarımını olumlu etkilediği görülmüştür. A harmanlayıcısının BER değeri 0,95x 10^{-3} iken, B harmanlayıcısı kullanıldığında 1,05x 10^{-3} değeri elde edilmiştir.

Bu çalışma, Altera DE1 FPGA devresi (Ek 1) ile gerçekleştirilmiştir. Bu uygulamalar Altera'nın FPGA tasarım programı QUARTUS II ile tasarlanmış ve uygulanmıştır. Gelecekte, Altera ve Xilinx firmalarının yeni nesil FPGA'larını kullanmak, tasarım konusunda daha da esnek bir çalışma sağlayacaktır. Çok daha farklı senaryolara aynı anda yanıt verebilecek hale gelecektir. Böylece elde edilen yüksek kapasiteli tasarım için, düşük kapasiteli FPGA'lerde yaşanan kapı gecikmesi sorunu giderilmiş olur. Buna karşın hem Turbo kod algoritmalarının, hem de FPGA donanımının paralel yapıda olmasından dolayı, verimlilik artacaktır.

MAP algoritması kullanan Turbo kodlamanın geleceğinde,kodlamanın içerdiği büyük boyuttaki hesapsal karmaşanın azaltılması ve daha basit yapıda bir algoritma olması için çalışmalar yapılmalıdır. Bu çalışmalar başarıya ulaştığında, ileride çok daha kolay kodlanabilen yapılar elde edilebilinir. Buda yüksek verimliliğe sahip olan Turbo kodlamanın, hemen hemen bütün haberleşme sistemlerinde kullanılabilir hale getirilmesinde önemli bir adım olur.

Sonuç olarak; MAP algoritması kullanan Turbo kodlamada yukarıda açıklandığı üzere, bir diğer kod çözücü algoritması olan SOVA algoritmasına göre daha verimli sonuçlar vermiştir.

KAYNAKLAR LİSTESİ

[1] Theodore S. Rappaport, Wireless Communications, Principles & Practice, Prentice Hall, 1999

[2] Bahl, L. R. , Cocke, J. , Jelinek, F. and Raviv, J. , Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate, IEEE Trans. Information Theory, p.284-287,1974

[3] C. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit errorcorrecting coding and decoding: Turbo-codes, International Conference on Communications, ICC'93, 1064-1070 , 1993

[4] Sazlı M. , A Review Turbo Coding And Decoding, Commun. Fac. Sci. Univ. Ank.Series A2-A3 49 : 1-17 , 2005

[5] LB Communications, Method for Hardware Implementation of a Convolutional Turbo Code Interleaver and a Sub-block Interleaver , 2004

[6] Fossorier, M. P. C. , Burkert, F. , Lin, S. and Hagenauer, J. , On the equivalence between SOVA and Max-Log-MAP decodings, IEEE Comm. Lett. , vol. 2, no. 5, pp. 137-139 , 1998

[7] A Novel BER Model for Turbo Coding over Rayleigh Channel Department of Electronics and Information Engineering, Huazhong University of Science and Technology Wuhan, China , 2005

[8] Shuai Shi, Development of System for Teaching Turbo Code Forward Error Correction Techniques , Electronics letters, Vol. 148 No:67, 2007

[9] Nyambane Rogers, Hardware Implementation of a Reprogrammable Turbo Decoder , International Symposium on Turbo Codes & Related Topics, Brest, France, pp. 115-198 , 2005

[10] M. C. Valenti and J. Sun, The UMTS Turbo Code an Efficient Decoder Implementation Suitable for Software Defined Radios, International Journal of Wireless Information Networks, 2001

[11] J. F. Cheng and T. Ottosson, Linearly Aprroximated log-MAP Algorithms for Turbo Coding, Proc. Of IEEE VTC, 2000

[12] J. Vogt, A. Finger, Improving the Max-Log-MAP Turbo Decoder, Electronics letters, Vol. 36 No:23, 2000

[13] A. J. Viterbi, An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes, IEEE. J. Sel Areas Commun. Vol. 16, no:2, pp. 260-264 , 1998

[14] Bernard Sklar, A primer on Turbo Code Concepts, IEEE Communication Magazine, 0163-6804/97/ : 94-102 , 1997

[15] Claude Berrou, Alain Glavieux and Punya Thitimajshima. Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes, Proc. ICC'93, pp. 1064-1070 , 1993

[16] Maxfield C. , Design Warriors Guide to FPGA, Mentor Graphics Corporation and Xilinx, Inc. , 2004

[17] C. Douillard, M. Jezequel and C. Berrou, The Turbo Code Standard for DVB-RSC, 2nd International Symposium on Turbo Codes & Related Topics, Brest, France, pp. 535-538 , 2000

[18] Burr, A. , Turbo-codes: the Ultimate Error Control Codes, Electronics and Communication Engineering Journal. , 2001

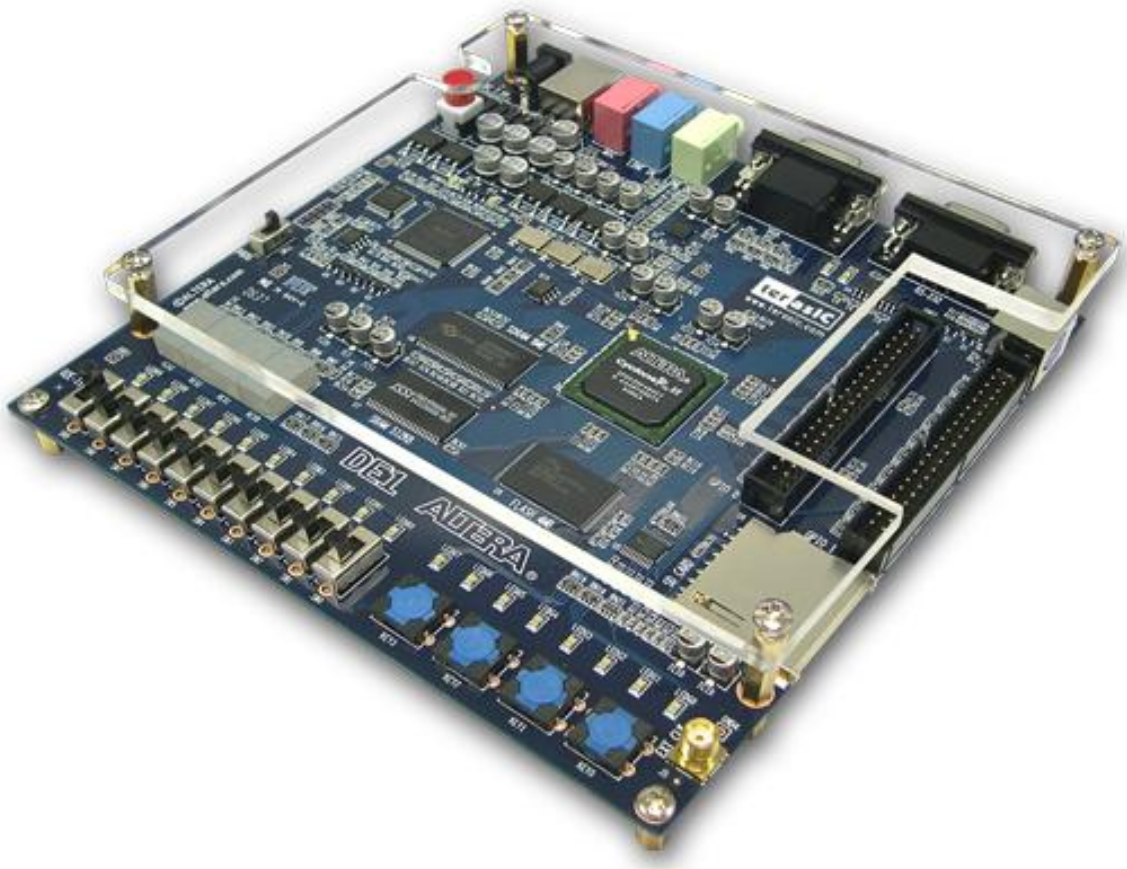
[19] Douillard, C. , Jesequel, M. , Berrou, C. , Picart, A. , Didier, P. , and Glavieux, A., Iterative Correction of Intersymbol Interference: Turbo Equalization, European Trans. Telecom. , vol. 6, pp. 507-511 , 1995

[20] D. Giancristofaro, A. Bartolazzi, Novel DVB-RSC Standard Turbo Code Details and Performances of a Decoding Algorithm, ESA conference, 2001

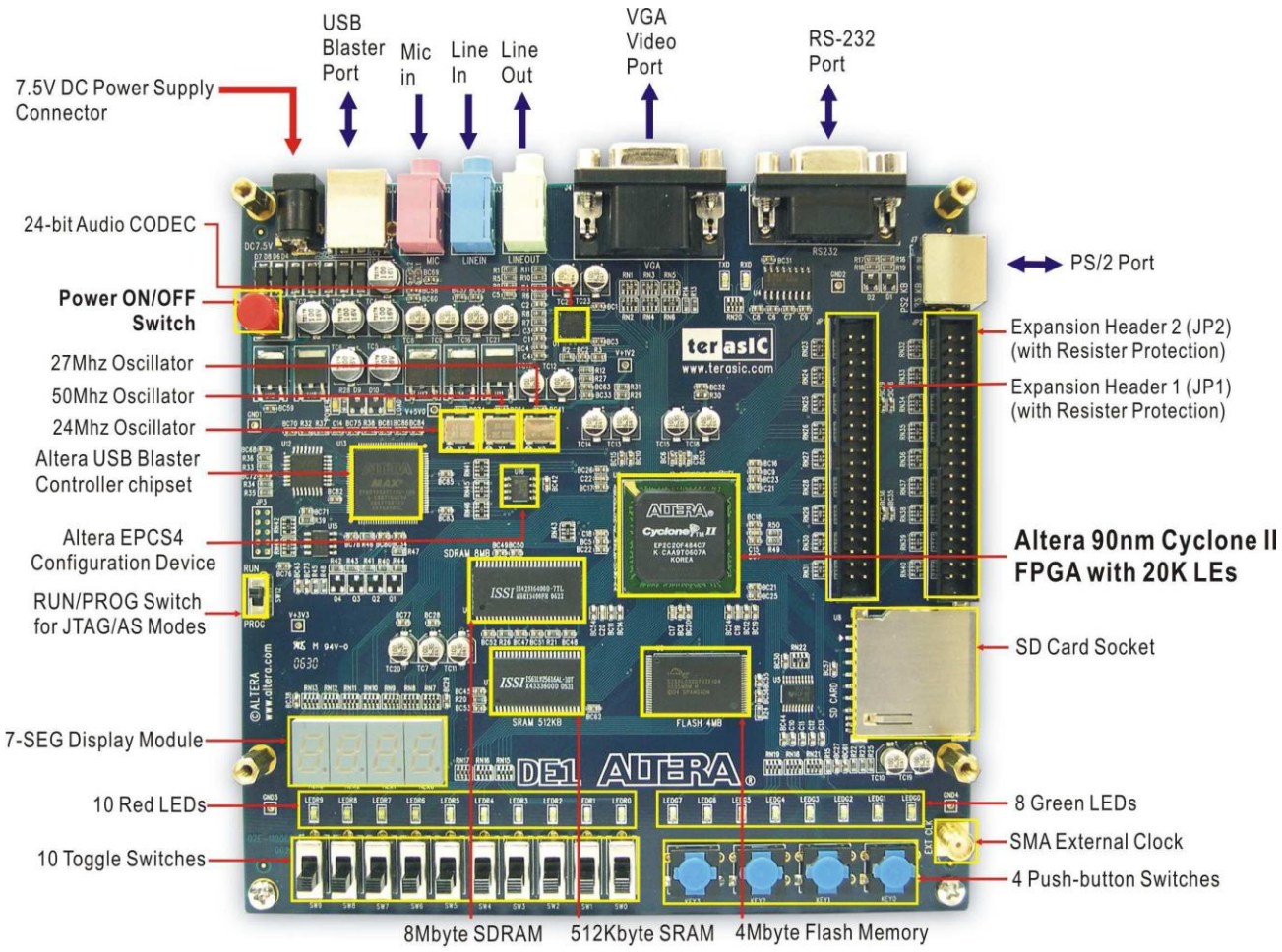
[21] Hanzo, L. , Liew, T. H. , Yeap, B. L. , Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels, Wiley-IEEE Press, 2002

[22] Robertson, P. , Villebrun E. and Hoeher, P. , 1995, A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain, Proc. ICC, vol. 2, p. 1009-1013, 2005

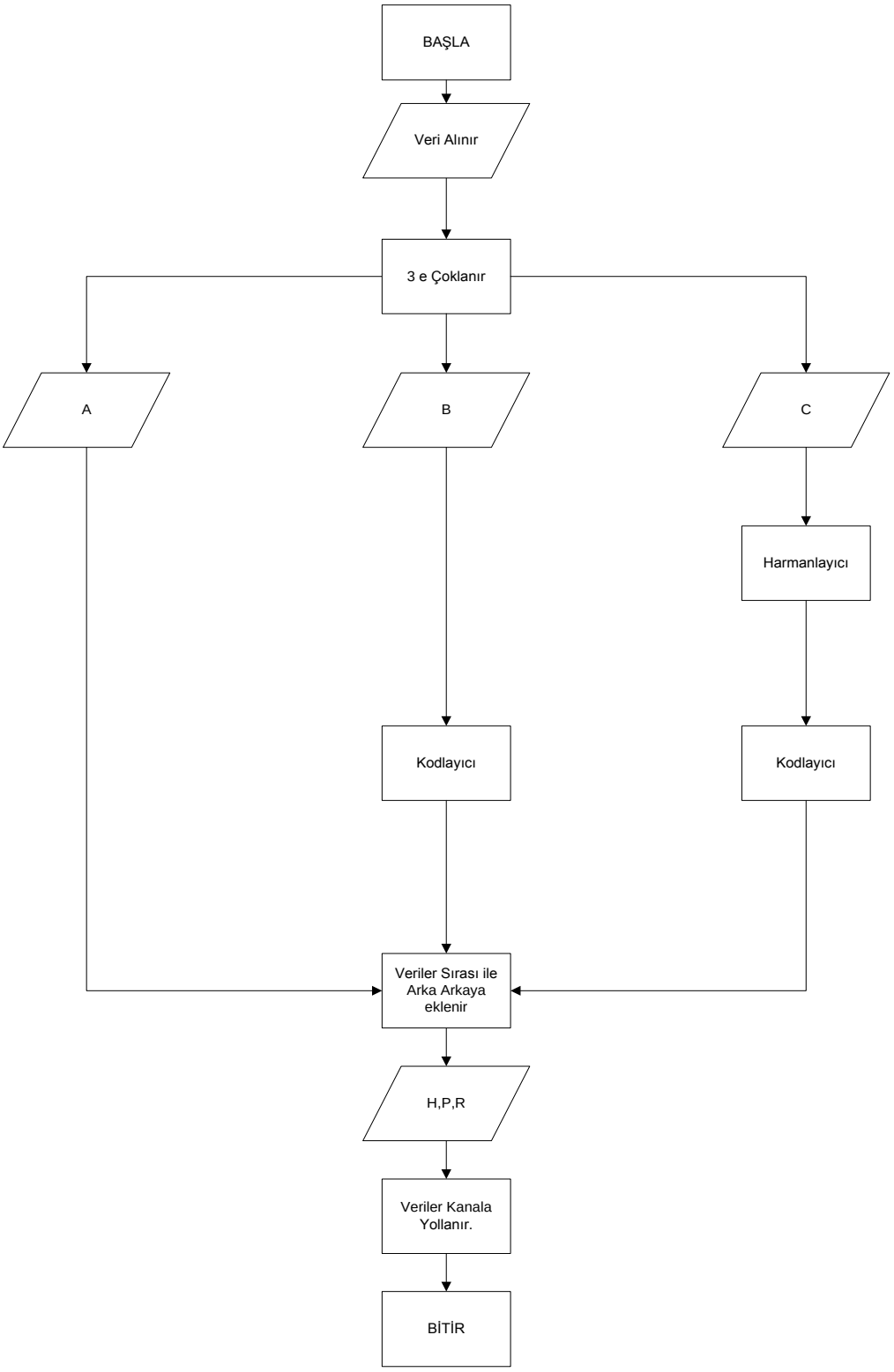
EK-1 Altera DE1 FPGA Devre Şekli



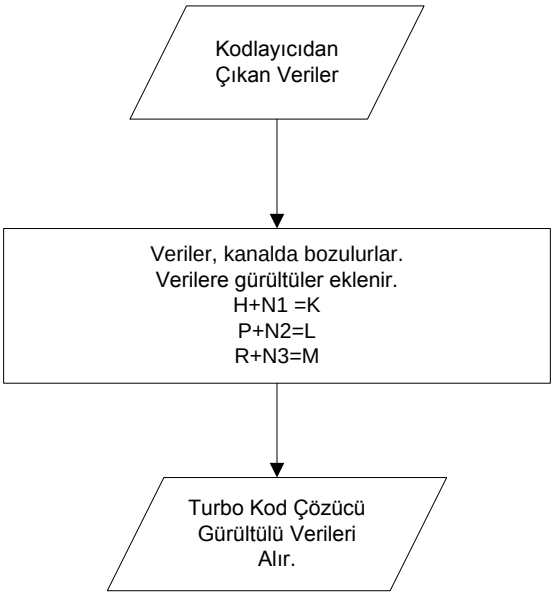
EK-2 Altera DE1 FPGA Devresinin Girişlerinin ve Çıkışlarının Gösterimi



EK-3 Turbo Kodlayıcı Akış Şeması



EK-4 Verilerin Kanala Verilmesi



EK-5 Turbo Kod Çözücünün Akış Şeması