T.C.

MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

VITERBI KOD ÇÖZÜCÜ'NÜN GÜÇ ETKİN MİMARİ TASARIMI VE FPGA GERÇEKLEMESİ

BURCU ÖZBAY

Yüksek Lisans Tezi

Tez Danışmanı

Yrd. Doç. Dr. Serap Çekli

İSTANBUL 2017

T.C.

MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

VITERBI KOD ÇÖZÜCÜ'NÜN GÜÇ ETKİN MİMARİ TASARIMI VE FPGA GERÇEKLEMESİ

YÜKSEK LİSANS TEZİ

Burcu Özbay

Tez Danışmanı

Yrd. Doç. Dr. Serap Çekli

İSTANBUL – 2017

Bu tez çalışması, Maltepe Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... / ..... / ..... tarih ve ..... / ..... sayılı kararıyla oluşturulan jüri tarafından ……. Yüksek Lisansı Tezi olarak kabul edilmiştir.

JÜRİ

Danışman

Yrd. Doç. Dr. Serap Çekli

Üye

Yrd. Doç. Dr. Ali Akman

Üye

Yrd. Doç. Dr. Selçuk Sevgen

maltepe üniversitesi istanbul www.maltepe.edu.tr Marmara Eğitim Vakfı 1997 İstanbul maltepe university istanbul www.maltepe.edu.tr

YEMİN METNİ

…../…../20….

Yüksek Lisans tezi olarak sunduğum "Viterbi Kod Çözücü'nün Güç Etkin Mimari Tasarımı ve FPGA Gerçeklemesi" adlı çalışmanın, proje safhasından sonuçlanmasına kadar olan bütün süreçlerinde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın tarafımca yazıldığını ve yararlandığım bütün eserlerin "Kaynakça"da gösterilenlerden oluştuğunu, "Kaynakça"da yer alan bu eserlerden metin içinde atıf yaparak yararlanmış olduğumu belirtir ve onurumla doğrularım.

121402106

Burcu Özbay

İmza

ÖZET

Viterbi Kod Çözücü'nün Güç Etkin Mimari Tasarımı ve FPGA Gerçeklemesi

Bu tez çalışmasında Viterbi Kod Çözücü ve güç etkin Viterbi Kod Çözücü'nün mimari tasarımı ve FPGA üzerinde gerçeklemesi hedeflenmiştir. Bir Viterbi Kod Çözme Sistemi, Katlamalı kodlayıcı ve Viterbi Kod Çözücü'den oluşmaktadır. Katlamalı kodlayıcının giriş serisinden ürettiği kod kelimeleri gürültülü bir iletişim kanalından geçerek kod çözücüye ulaşmaktadır. Ancak kanaldaki bu gürültüden dolayı kodda bozulmalar meydana gelebilir. Viterbi Kod Çözücü, en büyük olabilirlik temelli çalışan Viterbi Algoritması'nı kullanarak, gelen bozulmuş veriden, asıl mesajı en iyi şekilde arındırmakta ve giriş mesajını çözmektedir.

Bir Viterbi Kod Çözücü, Dal Ölçütleri Birimi (BMU), Ekleme Karşılaştırma Seçme Birimi (ACSU) Yol Ölçütleri Birimi (PMU) ve Hayatta Kalan Yol Bellek Birimi (SPMU) olmak üzere dört temel birimden oluşmaktadır. Bu birimlerin içerisinde her saat darbesinde yenilenen karmaşık hesaplamalar yapılmaktadır.

Bu tez çalışmasında bu karmaşıklığın azaltılması amaçlanarak, güç ve alan kullanımı bakımından verimli bir kod çözücü mimari tasarımı hedeflenmiştir. Öncelikle klasik bir sert karar Viterbi Kod Çözücü'nün büyük ölçekli tümdevre (VLSI) gerçeklemesi için bir mimari tasarım sunulmuştur. Bu mimari üzerinden daha az karmaşıklıkla bir tasarım yapılarak kod çözücü güç verimli olacak şekilde iyileştirilmeye çalışılmıştır. Sonuç olarak, bu çalışma kapsamında biri diğerinin güç etkin hali olmak üzere iki tane Viterbi Kod Çözücü mimari tasarımı ve benzetimi, Xilinx ISE Design Suit olarak bilinen tasarım ve benzetim ortamı ve Verilog donanım tanımlama dili kullanılarak yapılmıştır.

Anahtar Kelimeler: Viterbi Kod Çözücü, Viterbi Kod Çözücü tasarımı, FPGA, en büyük olabilirlik, ileri hata düzeltme

ABSTRACT

Power Efficient Viterbi Decoder Architectural Design and FPGA Imlementation

In this thesis, it is aimed to architectural design and implementation of Viterbi Decoder and power effective Viterbi Decoder on FPGA. A simple Viterbi decoding system consists of a convolutional encoder and a Viterbi Decoder. The convolutional encoder generates code words from the input message. Then, this code passes through a noisy communication channel to the decoder, however the corruption occurs in the code word due to noise in the channel. The Viterbi Decoder by Viterbi Algorithm based on the maximum likelihood, extracts the original message from the corrupted message and estimates the input messages.

The Branch Metric Unit (BMU), Add-Compare-Select Unit (ACSU) Path Metric Unit (PMU) and Survivor Path Memory Unit (SPMU) are the four main units of the Viterbi decoder. In these units, complex calculations are repeated at each every clock cycle.

In this thesis, it is focused on design an efficient decoder architecture in terms of power and utilization amount with the aim of reducing this complexity. Therefore, initially VLSI implementation of a classical hard decision Viterbi Decoder's architectural design is presented. Based on this initial design, a design with less complexity has been improved so that the decoder is power efficient. As a result, two Viterbi Decoder architectural designs and simulations, one of which is power efficient reconfigured are made using the design and simulation environment known as the Xilinx ISE Design Suite and the Verilog hardware description language (Verilog HDL).

Key Words: Viterbi Decoder, Viterbi Decoder architecture, FPGA, maximum likelihood, forward error correction

ÖNSÖZ

Bilgi aktarımı günümüz dünyasının gerekliliklerindendir. Doğru bilgi aktarımı ise bilgi aktarımı esnasında en önemsenen şeydir. Bir verici, alıcıya iletişim sistemleri üzerinden bilgi aktarımı yapmaktadır. Ancak bazen iletişim kanalında oluşan gürültüden dolayı alıcı bilgiyi gönderildiği şeklinden farklı algılamaktadır. Bu duruma çözüm olarak kodlama yöntemleri geliştirilmiştir. Bilginin kodlanarak iletilmesi, kodlanan bilginin çözülmesi ihtiyacını doğurur ve bu bir iletişim sisteminde kod çözücüler tarafından gerçekleştirilir.

Viterbi Kod Çözücü'ler, ileri hata düzeltme kodlarından olan katlamalı kodların çözülmesi için geliştirilen kod çözücülerdir. Bu tez çalışmasında Viterbi Kod Çözücü için bir mimari tasarım sunulmuştur ve bu tasarım güç etkin olacak şekilde geliştirilmiştir.

Tez konumu belirlememden bitirene kadar olan süreçte hep destekleyen, çalışmam boyunca içinden çıkamayacağımı düşündüğüm her noktada en doğru çözüm yolunu gösteren ve içten yardımlarını esirgemeyen değerli tez danışmanım Yrd. Doç. Dr. Serap Çekli'ye, hep yanımda olan ailem ve arkadaşlarıma teşekkürlerimle…

Mayıs 2017 Burcu Özbay

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

KISALTMALAR

ACK	Acknowlegement	Alındı Bildirimi
ACSU	Add Compare Select Unit	Ekleme Karşılaştırma Seçme Birimi
ARQ	Automatic Repeat Request	Otomatik Tekrar İsteği
AWGN	Additive White Gaussian Noise	Toplamsal Beyaz Gauss Gürültüsü
BEC	Backward Error Corection	Geri Hata Düzeltme
BER	Bit Error Rate	Bit Hata Oranı
BM	Branch Metric	Dal Ölçütleri
BMU	Branch Metric Unit	Dal Ölçütleri Birimi
ED	Euclid Distance	Öklid Uzaklığı
FEC	Forward Error Corection	İleri Hata Düzeltme
FPGA	Field Programmable Gate Array	Alanda Programlanabilir Kapı Dizileri
HD	Hamming Distance	Hamming Uzunluğu
HDL	Hardware Description Language	Donanım Tanımlama Dili
HMM	Hidden Markov Model	Saklı Markov Modeli
ISE	Integrated Software Environment	Tümdevre Yazılım Ortamı
LUT	Look Up Table	Arama Tablosu
MLA	Maximum Likelihood Algorithm	En Büyük Olabilirlik Algoritması
NACK	Negative Acknowlegement	Negatif Alındı Bildirimi
PM	Path Metric	Yol Ölçütleri

PMU	Path Metric Unit	Yol Ölçütleri Birimi
RE	Register Exchange	Kaydedici Değiştirme
RTL	Register Transfer Level	Kaydedici Devir Seviyesi
SNR	Signal to Noise Ratio	Sinyal Gürültü Oranı
SP	Survivor Path	Hayatta Kalan Yol
SPMU	Survivor Path Memory Unit	Hayatta Kalan Yol Bellek Birimi
SRAM	Static Random Acces Memory	Sabit Rasgele Erişimli Bellek
TB	Traceback	Geri İzleme
TCM	Trellis Code Modulation	Kafes Kod Modulasyonu
VA	Viterbi Algorithm	Viterbi Algoritması
VD	Viterbi Decoder	Viterbi Kod Çözücü
VHDL	WHSIC Hardware Description Language	WHSIC Donanım Tanımlama Dili
VLSI	Very Large Scale Integration	Çok Büyük Ölçekli Tümdevre

1. GİRİŞ

İletim kanalında oluşan gürültü, bir iletişim sisteminin güvenli veri aktarımı üzerindeki en büyük tehlikedir. Bu tehlikeden kurtulmak için çeşitli hata düzeltmeli kodlama teknikleri verinin büyüklüğüne, etkinlik hedeflenen alana ve alıcının önceliğine göre kullanılmaktadır. Kodlanan veri kod çözücüler tarafından çözülerek alıcıya iletilmektedir. Bir kod çözücü için, gürültülü bir kanalda aktarılan verinin bozulmamış haline ulaşabilmesi ve bunu etkin başarım ile yapması hedeflenmektedir. Viterbi Kod Çözücü (VD)'ler bu temelde çalışan kod çözücülerdir ve kablosuz (Wireless) iletişim sistemleri, uydu ve uzay iletişim sistemleri, Mobil İletişim Sistemleri (GSM), Kod Bölümlemeli Çoklu Erişim (CDMA) sistemleri gibi alanlarda çokça kullanılmaktadır.

Bir VD sistemi, ileri hata düzeltme kodlarından olan katlamalı kodları üreten katlamalı kodlayıcı ve VD'den oluşur. Katlamalı kodlar, ilk defa 1955 yılında Peter Ellias tarafından, kod çözücünün karmaşıklığını azaltmak amacıyla blok kodlara alternatif olarak sunulmuştur. Bir katlamalı kodlayıcı sonlu durum makinesi temeline dayanarak çalışmaktadır ve kodlayıcının durumlarını ve durumlar arası geçişleri sağlayan giriş ileti bitleri, kafes gösterimiyle ifade edilebilmektedir.

VD, Viterbi Algoritması (VA)'nı kullanarak çalışmaktadır. A.J. Viterbi, VA'yı kod çözmede hata olasılığını azaltmak için 1967 yılında önermiştir. Algoritma, katlamalı kodlayıcı ile kodlanmış giriş sinyalinin en büyük olabilirlik (ML) ilkesine göre çözümlenmesi esasına dayanmaktadır [1, 2].

Günümüze dek VD'nin sayısal sistem tasarımı için birçok yaklaşım yapılmıştır. Bu yaklaşımlar genellikle kod çözmedeki hızı arttırmaya [7, 9, 10], donanımsal karmaşıklığı azaltarak güç tasarruflu donanım tasarımına [4, 5, 12-14, 16, 17, 19-22], kısıtlama uzunluğunu değiştirmeye [8-10] ve radix4 tabanlı sayısal mimarilere [3, 11] odaklanmıştır.

VD'nin hızını arttırmaya yönelik tasarımlar genel olarak, kafesteki durum sayısını arttırarak geleneksel bir ACSU'nun seri olan mimarisini paralele dönüştürerek yapılmaktadır. Bu çeşit bir mimari sayesinde kod çözücünün hızının % 33 oranında artması sağlanmıştır [7]. Bu tip mimarilerde hız artışı sağlanmaktadır. Ancak donanımsal olarak karmaşıklık arttığı için güç tüketimi de artmış olur. Diğer bir kod çözünün hızını arttırma yaklaşımı, iş hattı mimarisinin kullanılmasıdır. VD'nin, kısıtlama uzunluğu yüksek tasarlanmış bir kodlayıcıdan gelen kodu çözmesi zaman almaktadır. ACSU'da iş hattı mimarisinin kullanılması kod çözücünün hızını arttırmaktadır. Ancak iş hattı mimarisinin derinliği arttıkça kod çözücünün donanımsal olarak karmaşıklığı da artacaktır. Bu yüzden mimarinin derinliği ve kodlayıcının kısıtlama uzunluğu için seçilen en iyi değerlerle güç tüketimi en az olacak şekilde kod çözücünün hızı arttırılmaktadır. Bu mimaride en iyi sonuç, derinlik 5, kısıtlama uzunluğu 9 olduğunda alınmıştır [9, 10].

VD'yi güç etkin hale getirmek için yapılan çalışmalar ise, temelinde donanım karmaşıklığını düşürerek güç tasarrufu sağlamaya odaklanmaktadır. Donanım karmaşıklığını düşürmek için ise genel olarak kodu çözmek için geriye dönülürken, güç tüketimini azaltan bir algoritma kullanılmasına [12, 16, 17, 20], SPMU'nun veya ACSU'nun mimarisinin değiştirilmesine yönelik çalışmalar yapılmıştır [5, 7, 12, 13, 19, 21, 22].

Bir VD için kaydedici değiştirme (RE) veya geri izleme (TB) algoritmaları kullanılarak seçilen yol üzerinden geriye dönüş ile giriş mesajı çözülmektedir. TB'de belleğin güncelleme hızı, RE'ye göre daha az olduğu için, güç tüketimi önemli ölçüde azaltılmaktadır. Eğer güç verimli çalışan bir tasarım yapılacaksa geriye dönüş, TB algoritması ile gerçekleştirilmelidir. Klasik TB, üç büyük operasyonu içermektedir. Bunlar karar biti yazma, geri izleme okuma ve kodu çözmedir. Her döngüde kararların bir sütunu belleğe yazılmakta ve m sütunu, geriye dönme ve kodu çözme için okunmaktadır. İşaretçiler sütun okumasını takip etmek için kullanılmaktadır. m tane sütunu okumak için m tane işaretçi kullanılması çoklu bellek gerektirmektedir. Tasarlanan bir ön TB tekniği sayesinde, m işaretçinin her aşamada okuma yapması yerine tek bir işaretçi, m aşamayı önceden izlemekte ve bir defa karar olarak yazmaktadır. Böylece bellek alanı ve erişim frekansı

azaltılmaktadır [17]. Böyle bir tasarımda TB herhangi bir andan başlatılmakta ve seçilen yolun 2^{K-1} yol değeriyle karşılaştırılması ihtiyacı ortadan kaldırılmaktadır [20].

Kablosuz iletişim uygulamalarını hedefleyen bir VD için karar biti yazma ve geri izleme okuma işlemi arasındaki doğal paralellikten yararlanarak tasarlanan bir TB öncesi mimarisi ile SP'lerin bellekten okunma işlemlerini azaltmaktadır. Bellek erişim işlemlerinin azaltılması sonucunda belleğin boyutu ve kod çözme gecikmesi % 25 oranında azaltılmıştır. TB öncesi yaklaşımının, geleneksel TB mimarisi ile karşılaştırıldığında %11'e kadar enerji verimliliği ve %21'lik alan tasarrufu sağladığını gösterilmiştir [12]. Farklı bir ön TB mimarisi çözülmüş verileri bloklar halinde elde etmek için önerilmiştir [16]. Bu mimaride, hayatta kalan yolun (SP) numarası kaydedildiğinden, TB ve çözme işlemi sırasında karar bitinin kaydedilmesine gerek kalmaz. Bu nedenle, SPMU'nun güç tüketimi ve alanı mevcut TB yaklaşımlarınınkinden daha küçüktür.

VD'nin ACSU için seri aritmetiğini temel alarak yapılan tasarımlar alan ve güç verimli kod çözücüyü mümkün kılmaktadır. TB yapılacak bellek birimini ikiye bölerek yapılan bir tasarım ile kod çözücünün güç verimli çalışması sağlanmıştır [5]. Bellek kullanımını azaltmak amacıyla ara sonuçları hafızaya almadan sadece hesaplamalara katarak yapılan tasarım sayesinde bilinen kod çözücüye göre daha az eleman kullanıldığı için güç tasarrufu yapılmıştır [7].

Güç etkin kod çözücü için kullanılan bir başka yaklaşım Kafes Kod Modülasyonu (TCM)'dur. Sayısal iletişimde kullanılan hata düzeltici kodlamayı ve modülasyonu birleştiren bir tekniktir. Amacı sinyal bant genişliğini veya iletilen gücü arttırmadan iletişim kanalında koda, gürültüye karşı bağışıklık kazandırmaktır. Süreç, sinyal haritalama yöntemini kullanmaktadır [14].

VD'nin güç verimli tasarımı için en başarılı yaklaşım ise ACSU'ya tanımlanan bir eşik değer ile bu değerin üzerinde yol değerine sahip yolların hesaplamaya katılmamasını sağlayarak kod çözücün hesap yükünü azaltmak temeline dayanmaktadır. Bu yaklaşıma T-Algoritması denilmektedir ve çoklukla bu algoritmayı kullanan tasarımlar yapılmıştır [19, 21, 22]. Bu şekilde çalışan kod

çözücü tasarımları sayesinde, PMU'da %25-47 oranında depolama alanından tasarruf sağlanmıştır [12]. Ayrıca VD'nin güç tüketimi %69 oranında azaltılmıştır [13].

Bu tez çalışmasında giriş mesajı, kısıtlama uzunluğu 3, kod oranı $^1/_2$ olarak tasarlanan katlamalı kodlayıcıda kodlanmış böylece ileri hata düzeltme kodlarından olan katlamalı kod elde edilmiştir. Bu kod, gürültülü bir iletim kanalından geçirildikten sonra alıcıya ulaşmadan, kodlanmadan önceki haline döndürülmesi için, sert karar yöntemine göre çalışan VD tasarlanmıştır. Tasarlanan kod çözücü üzerinden yapılan yeni bir tasarım ile kod çözücü güç etkin hale getirilmiştir.

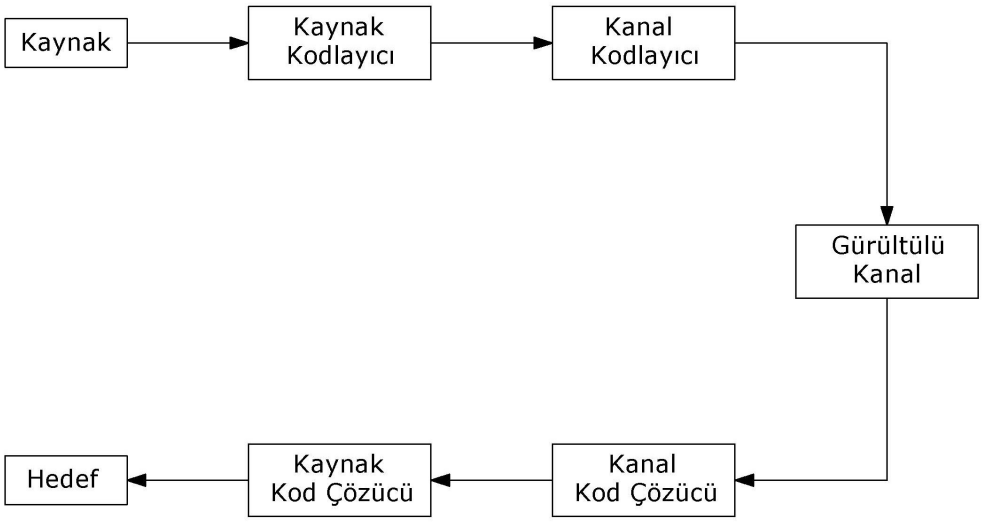
Tez çalışmasının ayrıntılı tanıtımı ve literatürdeki çalışmalar Bölüm 1'de açıklanmıştır. Bölüm 2; kodlama teorisinin, kodlayıcının yapısının, durum diyagramının ve katlamalı kodlayıcının çalışmasını, Bölüm 3 ise; VA'nın ve kafes (Trellis) gösteriminin ayrıntılı teorik arka planını tanımlamaktadır. Bölüm 4, bu çalışma için tasarlanan VD'nin ve güç etkin VD'nin mimari tasarımını içermektedir. Bölüm 5, tasarlanan iki kod çözücünün benzetim sonuçlarının, hata düzeltme kabiliyetlerinin ve güç performans sonuçlarının karşılaştırıldığı, tez çalışmasından çıkan sonuç ile ilgilenen son bölümdür. Sınamanın yanı sıra çalışmayı daha ileri götürmek için mümkün olan gelecekteki çalışmaları içermektedir.

Bu çalışmada benzetimler ve sentezleme için Xilinx ISE ortamı, donanım tanımlama için Verilog HDL kullanılmıştır.

2. İLETİŞİM SİSTEMİ

Dünyada sürekli gelişmekte olan iletişim sistemlerinin en büyük ihtiyacı iletişim esnasındaki güvenilirlik ve hızdır. Bazen iletişim kanalındaki gürültüden dolayı alıcı veriyi gönderildiği şekliyle algılayamamakta ve veri güvenilirliği tehlikeye girmektedir. Bu yüzden iletişim sisteminde bilgi aktarımındaki güvenilirlik, farklı kodlama teknikleriyle veriye ek bitler eklenerek verideki bozulma olasılığını azaltarak sağlanmaktadır. Şekil 2.1'de gösterildiği gibi veri iletişim kanalında kodlanmış şekilde iletilmektedir. Ancak bu işlem verinin, kanaldaki hareketini tamamladıktan sonra kodlanmadan önceki haline döndürülerek alıcıya iletilmesi ihtiyacını doğurmaktadır. Bu da bir hata düzeltmeli kod çözücüyle sağlanmaktadır.

İletişim sistemlerinde hata tespiti ve düzeltilmesi büyük önem taşımaktadır. Bu işlemi gerçekleştiren kod çözücülere hata düzeltmeli kod çözücüler denilmekte ve belirtilen sebeplerden dolayı iletişim sistemlerinde çokça kullanılmaktadırlar.



Şekil 2.1 İletişim sistemi [15]

Bir iletişim sistemi için, x_i, mesaj serisinin i anındaki elemanı olmak üzere; $P(x_i) = P_i$ bu anda mesajda hata oluşma olasılığıdır [15]. Bu mesajdan oluşan bilgi denklem (1)'deki gibi tanımlanmaktadır.

$$I_i = -\log_b P_i = \log_b \left(\frac{1}{P_i}\right) \tag{1}$$

Bilgi, mesajın kendisi değil, hatasız oluşma olasılığıdır.

$$0 \leq P_i \leq 1 \qquad I_i \geq 0 \tag{2}$$

$$P_i \to 1 \qquad I_i \to 0 \tag{3}$$

$$P_i < P_j \qquad I_i > I_j \tag{4}$$

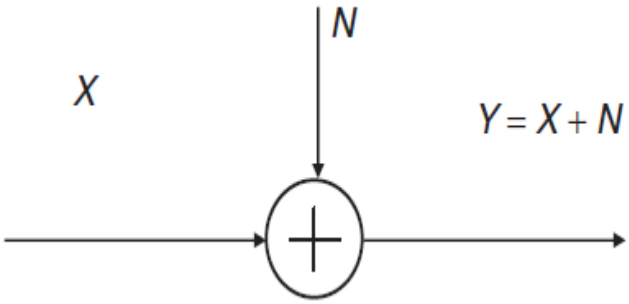
Bu denklemlere sırasıyla bakılırsa; denklem (2), mesajda hata oluşma olasılığı 0 ve 1 arasındaysa bilgi bir şekilde oluşur, denklem (3), mesajda hata oluşma olasılığı arttıkça ve 1'e yaklaştıkça bilgi 0'a yani oluşmama seviyesine yaklaşır, denklem (4), i anında hata olasılığı, j anında hata olasılığından küçükse; i anında bilgi, j anına göre daha yüksek oranda ortaya çıkar anlamına gelmektedir.

2.1. Kodlama Teorisi

Her gerçek iletişim sisteminde, algılanan ve gerçek bilgi arasında farklılıklar oluşabilir. Bu sinyal üzerine binen gürültüden kaynaklanmaktadır.

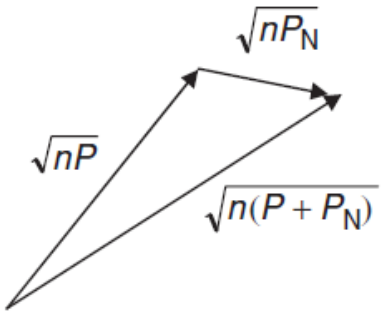
X giriş serisi, N gürültü olmak üzere, çıkış serisi Y'nin oluşumu Şekil 2.2'de gösterilen şekildedir ve denklem (5) ile ifade edilmektedir.

$$Y = X + N \tag{5}$$



Şekil 2.2 Gürültüden etkilenen çıkış serisi

Giriş serisi X, giriş serisinin gücü P, giriş serisinin ve gürültünün bit sayısı n, gürültünün gücü P_N olmak üzere çıkış Y, X'in ve gürültünün vektörel toplamıdır. Şekil 2.3'de gösterildiği gibi ifade edilmektedir ve denklem (6)'daki şekilde tanımlanmaktadır.



Şekil 2.3 Çıkış serisinin vektörel toplamı [15]

$$\sqrt{n(P + P_N)}=\sqrt{nP}+\sqrt{nP_N} \quad (6)$$

Sayısal iletişim sistemlerinde gürültünün gücü yüksekse iletilen bilginin doğruluğu tehlikeye girmektedir. Sistemde gürültü kaynaklı hata fazlaysa iletim kanalı kullanılamaz bir kanal haline gelmektedir. Bundan dolayı bazı tekniklerle verilen sinyal gürültü oranı (SNR) için hata olasılığı azaltılmaktadır. 1948'de Claude

Shannon güvenli olmayan bir kanal üzerinden hatasız bir iletim yapılabileceğini teorik olarak kanıtlamıştır. Bunun anlamı şudur; uygun kodlama yöntemleriyle iletilen bilginin büyüklüğü kanal kapasitesine eşit ya da kapasiteyi aşmayacak hale getirilebilir. Kodlama teorisinin hedefi kanal kapasitesinin kullanımındaki etkinlikle veri aktarımı arasında bir en iyileme sağlamaktır.

Kanal kapasitesi bir kanalın üzerinden iletilen en büyük bilgi büyüklüğü olarak tanımlanabilir ve denklem (7) ile ifade edilmektedir [15].

$$C_s = I_i(x_i)\ \text{bit/sembol} \tag{7}$$

Saniyede maksimum sembol oranı s ise, birim zaman başına kanal kapasitesi denklem (8)'deki gibidir.

$$C = sC_s \tag{8}$$

Kanal bant genişliği B olan bir kanaldaki sinyalin gücü S iken, bu sinyale etkiyen Toplamsal Beyaz Gauss Gürültüsü (AWGN) N ise, bu kanalın kapasitesi C, denklem (9) ile gösterilebilir;

$$C = B\log_2\left(1 + \frac{S}{N}\right) \tag{9}$$

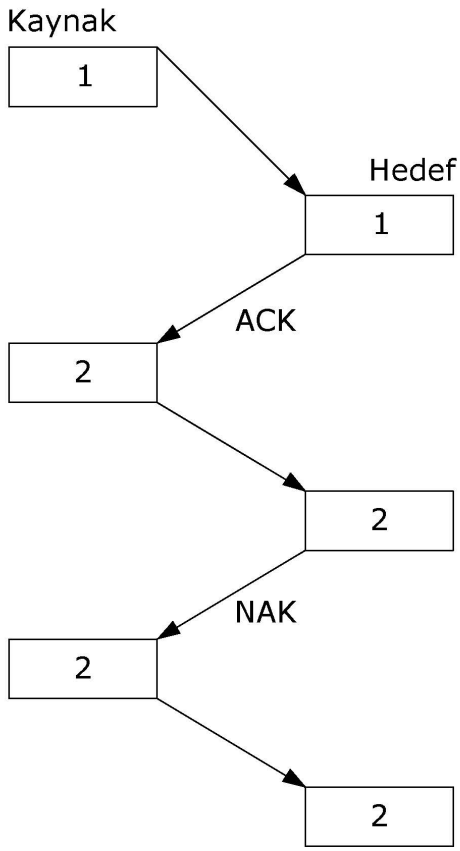
Bir iletişim sistemi için bu bir limittir. Bu kanalda bu limit değerinin üstünde bir iletimde veride bozulma olacaktır [15].

Bu problem geriye doğru hata düzeltmeli kodlama ve ileri hata düzeltmeli kodlama olmak üzere iki tür kodlama yöntemiyle çözülebilir.

2.1.1. Geriye Hata Düzeltme Kodlaması (BEC)

Bazı iletişim sistemlerinde koddaki hata düzeltilemez sadece algılanabilir. Bu tip kodlar genellikle iletilen veri yok olduğunda veya anlaşılamayacak derecede bozulduğunda kullanılmaktadır. Alıcıdan verinin bir daha gönderilmesine ilişkin bir

istekte bulunmaktadır. Buna otomatik tekrar isteği (ARQ) denir ve bu yöntem Şekil 2.4'de görüldüğü gibi çalışmaktadır. Bir sağlama biti hatanın yakalanması için veri çerçevesine eklenmektedir. Eğer alıcı hiç hata algılamazsa göndericiye bir alındı bildirimi (ACK) göndermektedir. Hata algılarsa bir negatif ACK sinyali yani NAK göndermekte ya da hiçbir sinyal göndermeden göndericinin veriyi bir kez daha göndermesi için beklemede kalmaktadır. ARQ tekniği etkili bir teknik olmasına rağmen veri iletimi tekrarından dolayı sistemde iki kat kapasite gerektirmektedir. Ağ iletişim sisteminde çok hata oluşmadığı veya küçük veri grupları için tercih edilebilir olmasına rağmen kablosuz iletişim sistemlerinde gürültülü kanallardan kaynaklanan yüksek hata oranından dolayı kullanışsız olmaktadır. Bu sebeple gürültülü kanallar için ileri hata düzeltme kodları (FEC) sıklıkla kullanılan bir yöntem haline gelmiştir.



Şekil 2.4 BEC türü kodlama [15]

2.1.2. İleri Hata Düzeltme Kodlaması (FEC)

FEC, kodlanacak veriye ek bitler ekleyerek kanalda bu şekilde hareket etmesi esasına dayanmaktadır. Bu tip kodlar hatayı yakalama ve düzeltme kabiliyetine sahiptir. FEC, Blok Kodlar ve Katlamalı Kodlar olmak üzere iki bölümde incelenebilir.

2.1.2.1. Blok Kodlar

k bit uzunluğundaki mesajın, n bit uzunluğundaki kod kelimesi halinde kodlanmasıdır. (n,k) parametrelerine sahip bir blok kod için, kod oranı R, denklem (10) ile verilmektedir [15];

$$R = \frac{k}{n} \tag{10}$$

Blok kodlar, k uzunluğundaki mesaj bitine n-k uzunluğundaki eşlik biti eklenerek oluşturulmaktadır. Kod denklem (11)'de gösterildiği gibi bir X matrisiyle ifade edilmektedir.

$$X = [m\ p] \tag{11}$$

Mesaj ve eşlik bitleri denklem (12) gibi gösterilmektedir.

$$m = [m_1, m_2, \dots m_k] \text{ ve } p = [p_1, p_2, \dots p_{n-k}] \tag{12}$$

X matrisi, kodlayıcının üretim matrisi G ile mesajın çarpılması sonucunda oluşmaktadır.

$$X = mG \tag{13}$$

G üretim matrisi, k x k uzunluğunda birim matris I ve k x (n-k) uzunluğunda eşlik matrisi Z'nin birleşimidir ve denklem (14)'deki gibi ifade edilmektedir [15].

$$G = [I\ Z] \tag{14}$$

Hamming Uzunluğu (HD), herhangi kod çiftinin birbirine benzerliği ile ilgili bir parametredir. Bütün kod çiftleri için hesaplanabilir. En küçük HD d_{min}, mümkün olan bütün kodlar arasındaki en küçük HD'yi ifade etmektedir. Kod tarafından düzeltilebilir bit hatalarının sayısı, t, denklem (15) ile bulunmaktadır [15].

$$t = \left[\frac{d_{min}-1}{2}\right] \tag{15}$$

Bundan dolayı, eğer büyük bir d_{min}'e sahip bir koddan bahsediliyorsa, kodun hata düzeltme gücü daha yüksek olacaktır. Ancak yüksek bir d_{min} , yüksek bir n demektir ve bu kodun etkinliğini düşürmektedir. n'in büyümesi kodlayıcı ve kod çözücünün karmaşıklığını arttırmaktadır. Bu sebeple kod çözücünün başarımı ve kod kelimesinin uzunluğu arasında bir sınır vardır.

Blok kodlama için kodlayıcı, k bitlik mesajın, n bitlik kod kelimesine nasıl dönüştürüleceğini içeren bir arama tablosu kullanmaktadır. Böyle bir arama tablosunda n bit kod kelimesi için 2k tane girdiye ihtiyaç duyulmaktadır [15]. Bu büyük mesajlar için kodlamada ve özellikle kod çözmede büyük karmaşıklığa sebep olmaktadır. Blok kodların bir alt kümesi olan katlamalı kodlar bu duruma çözüm olmuştur. Kaydırmalı kaydedicilerin kullanılmasıyla, eşlik bitleri, kodlayıcıya gelen mesaj bitleri tarafından üretilmektedir. Bu uygulama sayesinde blok kodlarda oluşan kodlayıcı ve kod çözücünün karmaşıklığı büyük oranda azaltılmaktadır.

2.1.2.2. Katlamalı Kodlar

Katlamalı kodlama, iletim kanalında gürültü kaynaklı oluşan hatalardan dolayı değişen verinin alıcıya doğru iletimi için kullanılan kodlama yöntemlerinden bir tanesidir. Veri bir katlamalı kodlayıcı ile kodlanarak iletim kanalında bu şekilde hareket etmektedir.

Bir katlamalı kodlayıcı sonlu durum makinesi temeline dayanmaktadır. Kodlanmış dizi şu anki ve önceki mesajlara bağlı olarak giriş bilgi serisinden üretilmektedir. Kodlayıcı Şekil 2.5'de görüldüğü gibi bir veya daha fazla D flip flop ve xor (özel veya…) kapılarından oluşmaktadır. Burada görülen flip flopların üzerindeki bilgi, kodlayıcının, şu anki durumunu göstermektedir ve giriş mesajına göre bir sonraki duruma geçmesini sağlamaktadır.

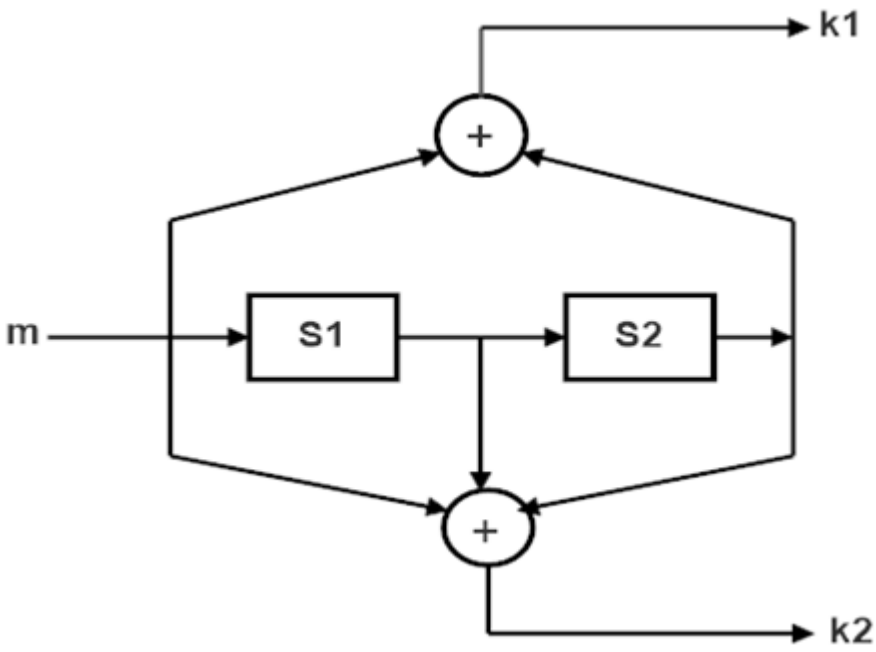
Bir katlamalı kod, Cconv (n, k, K) şeklinde ifade edilmektedir. Burada k, mesaj biti uzunluğu; n, kod biti uzunluğu; K, kısıtlama uzunluğu, kodlayıcının hafızasıdır, kodun derinliği de denilebilir. Her giriş bitinin kaç defa çıkış biti üretimi üzerinde etkisi olduğunu göstermektedir. Hafızanın daha yüksek oranda olması katlamalı kod çözücünün karmaşıklığının artacağı anlamına gelmektedir.

k bit mesaj, K-1 tane hafıza elemanına sahip kaydırmalı kaydediciden geçerek, modulo-2 toplama işlemine göre toplanmakta ve n bitlik çıkış kodu üretilmektedir. Katlamalı kodlayıcı, durum tablosu ve kafes diyagramıyla tanımlanabilir. Durumlar kodlayıcının kaydırmalı kaydedicisinin bir üretimidir. Çıkış kodu ise, kodlanılacak mesaj ve o anki durumun bir fonksiyonudur.

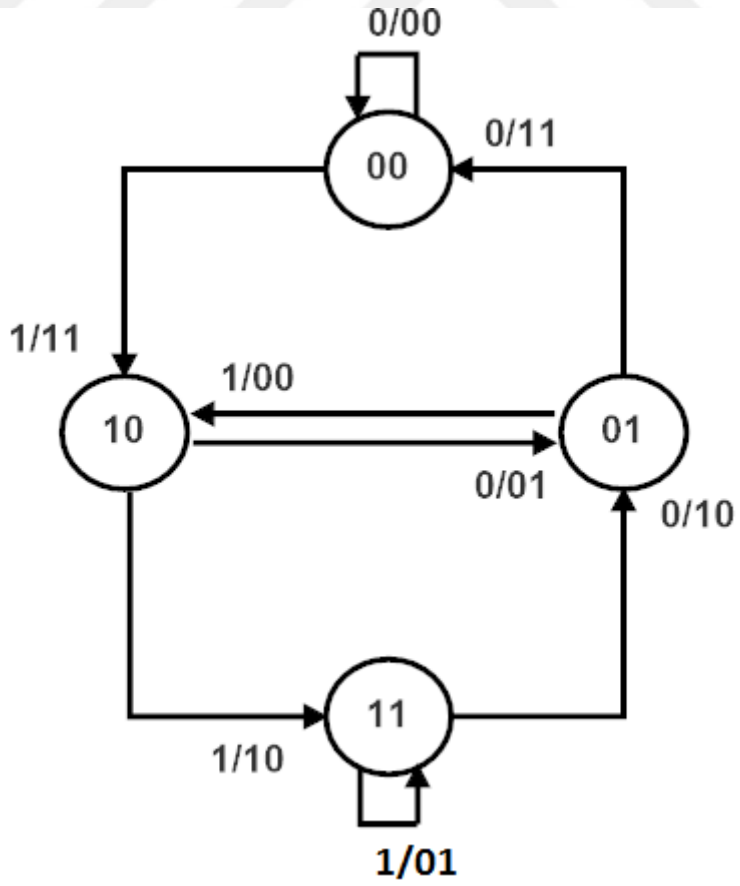
Durum diyagramı, bir giriş mesajının kodlandığı zamandan diğer giriş mesajının kodlandığı zamana geçişi göstermektedir. Kafes diyagramı ise durum diyagramının bir tanımıdır.

Şekil 2.5'de gösterilen kodlayıcı, Cconv (2, 1, 3) şeklinde ifade edilmektedir. Bu ifadeye göre bir bitlik mesaj, iki bit olarak kodlanmaktadır. Her mesaj kaydırmalı kaydedicinin ilk hafıza elemanına gelmekte ve hafıza elemanlarında tutulan bitler bir defa sağa kaymaktadır. Böylelikle kodlayıcı bir sonraki duruma geçmektedir.

Şekil 2.5'de gösterilen kodlayıcı için oluşturulan durum diyagramı Şekil 2.6'da, bu durum diyagramından oluşan durum tablosu da Tablo 2.1'de verilmiştir. Bu tabloda gelen mesaj bitine göre kodlayıcının hangi durumdan hangi bir duruma geçtiği ve çıkış kodunun o anda ne olduğu gösterilmektedir.



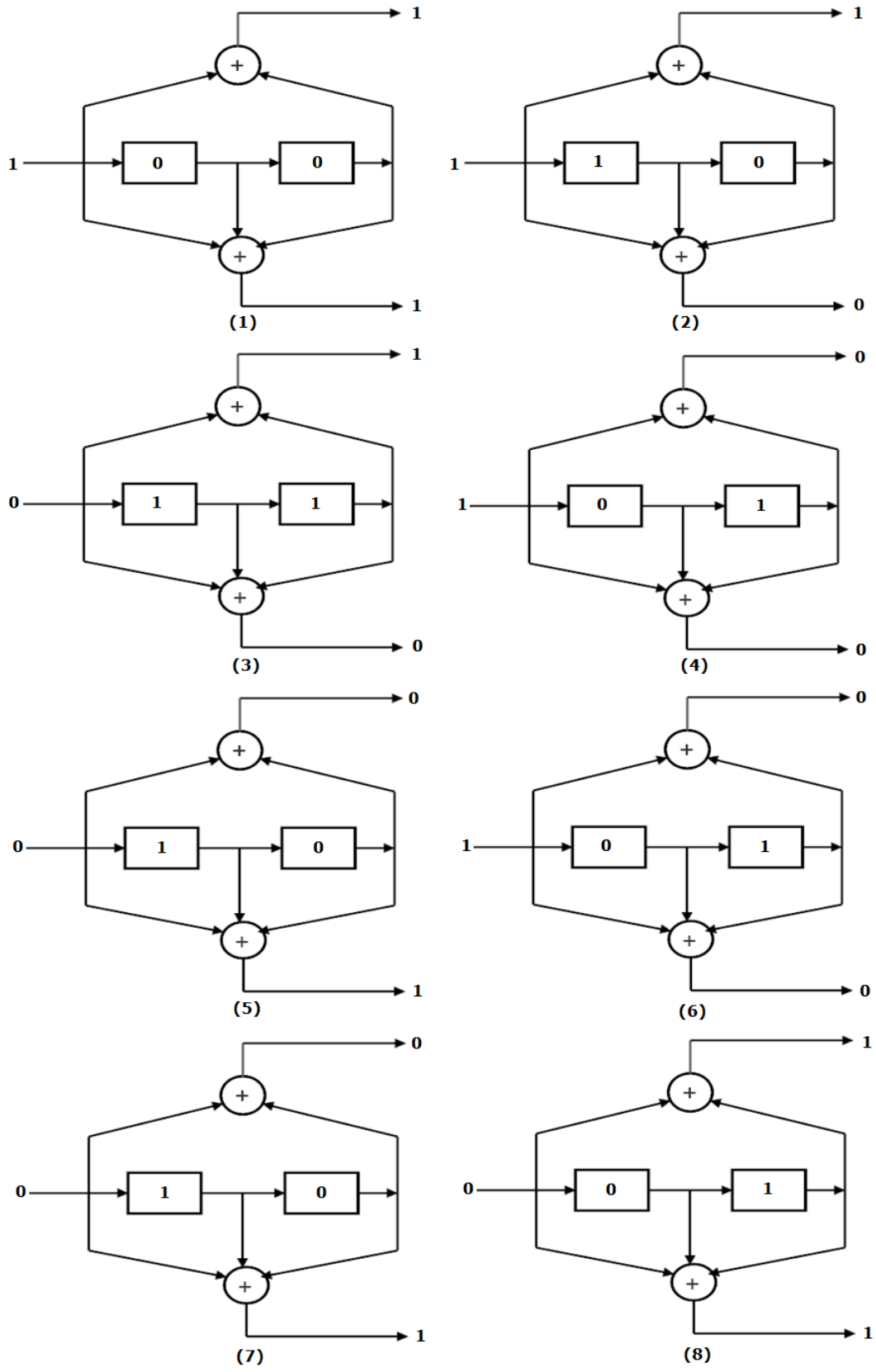
Şekil 2.5 Cconv (2, 1, 3) Katlamalı kodlayıcı [15]



Şekil 2.6 Şekil 2.5'de verilen kodlayıcının durum diyagramı

Giriş Mesajı m	Durum (t) S1 S2	Durum (t+1) S1 S2	Kod k1	Kod k2
-	00	00	-	-
0	00	00	0	0
1	00	10	1	1
0	01	00	1	1
1	01	10	0	0
0	10	01	0	1
1	10	11	1	0
0	11	01	1	0
1	11	11	0	1

Tablo 2.1 Verilen kodlayıcının durum tablosu



Şekil 2.7 Giriş mesajı (11010100) için kodlayıcı durumları ve çıkış kodları

Şekil 2.7'de giriş mesajı (11010100) için kodlayıcının durumları ve çıkış kodları gösterilmiştir. Çıkış koduna üretim polinomları veya üreteç matrisleri kullanılarak da ulaşılabilir.

Şekil 2.5'deki kodlayıcının üretim dizisi denklem (16) ve denklem (17) ile verilmiştir:

$$g^{(1)} = (101) \tag{16}$$

$$g^{(2)} = (111) \tag{17}$$

Bu gösterimde 1'ler kaydediciye bağlantıyı temsil ederken, 0'lar bağlantının olmamasını ifade etmektedir.

Giriş dizisi (11010100) ise;

Mesaj polinomu denklem (18):

$$M(D) = 1 + D + D^3 + D^5 \tag{18}$$

Üretim polinomu denklem (19):

$$G(D) = 1 + D^2 \;\; 1 + D + D^2 \tag{19}$$

Modulo-2 toplama işlemine göre kod polinomu denklem (21)'de verildiği gibidir.

$$K(D) = \big(k1(D)k2(D)\big) \tag{20}$$

$$= [1 + D + D^3 + D^5][1 + D^2 \;\; 1 + D + D^2] \tag{21}$$

$$= 1 + D + D^2 + D^7 \;\; 1 + D^4 + D^6 \; + D^7 \tag{22}$$

Çıkış dizisi denklem (23)'de verildiği gibi olacaktır:

$$k = (11, 10, 10, 00, 01, 00, 01, 11) \tag{23}$$

Başka bir şekilde çıkış dizisine denklem (24) kullanılarak da ulaşılabilir.

$$(\text{Çıkış Dizisi})_{1xn} = \text{Mod2}\left(\begin{pmatrix}\text{Giriş}\\ \text{Dizisi}\end{pmatrix}_{1xK+1} x \begin{pmatrix}\text{Üretim}\\ \text{Matrisi}\end{pmatrix}_{Kxn}\right) \quad (24)$$

Şekil 2.5'de gösterilen kodlayıcı için G, üretim matrisi denklem (25)'da gösterilmiştir;

$$G = \begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix} \quad (25)$$

Böylece Çıkış dizisi denklem (26) ile hesaplanmaktadır.

$$(k^{(1)}\ k^{(2)})_{1x2} = \text{Mod2}\left((\text{Giriş } 0\ 0)_{1x3} x \begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}_{3x2}\right) \quad (26)$$

Giriş mesajı (11010100) olmak üzere;

Giriş 1 için;

$$\text{Mod2}\left((1\ 0\ 0)_{1x3} x \begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}_{3x2}\right) = (1\ 1) \quad (27)$$

Giriş 1 için;

$$\text{Mod2}\left((1\ 1\ 0)_{1x3} x \begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}_{3x2}\right) = (1\ 0) \quad (28)$$

Giriş 0 için;

$$\text{Mod2}\left((0\ 1\ 1)_{1x3} x \begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}_{3x2}\right) = (1\ 0) \quad (29)$$

Giriş 1 için;

$$\text{Mod2}\left((1\ 0\ 1)_{1x3} x \begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}_{3x2}\right) = (0\ 0) \quad (30)$$

Giriş 0 için;

$$\mathrm{Mod2}\left((0\ 1\ 0)_{1x3}\mathrm{x}\begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}_{3x2}\right) = (0\ 1) \tag{31}$$

Giriş 1 için;

$$\mathrm{Mod2}\left((1\ 0\ 1)_{1x3}\mathrm{x}\begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}_{3x2}\right) = (0\ 0) \tag{32}$$

Giriş 0 için;

$$\mathrm{Mod2}\left((0\ 1\ 0\ )_{1x3}\mathrm{x}\begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}_{3x2}\right) \tag{33}$$

Giriş 0 için;

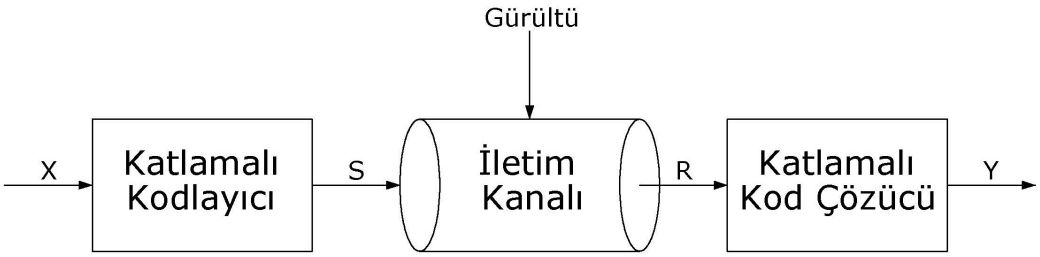
$$\mathrm{Mod2}\left((0\ 0\ 1)_{1x3}\mathrm{x}\begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{pmatrix}_{3x2}\right) = (1\ 1) \tag{34}$$

Böylece çıkış dizisi denklem (35)'de verildiği şekilde oluşmuştur ve görüldüğü gibi denklem (23)'ün aynısıdır. Şekil 2.7 ile uyumludur.

$$\mathrm{k} = (11, 10, 10, 00, 01, 00, 01, 11) \tag{35}$$

3. VITERBI ALGORİTMASI

3.1. Viterbi Algoritması'nın Temelleri



Şekil 3.1 İletim kanalında mesaj iletimi

VA, ileri hata düzeltme kodlarından olan katlamalı kodların çözümlenmesinde ML algoritmasına dayanarak çalışmaktadır. ML yaklaşımı katlamalı kodlayıcı için oluşturulmuş kafesteki koda en benzer yolu bulmaktadır. VA, Gizli Markov Modeli (HMM) bağlamında, gözlemlenen olayların bir dizilimiyle ortaya çıkan ve Viterbi yolu adı verilen, gizli durumların en olası sırasını içeren yolun bulunması olarak tanımlanmaktadır.

Şekil 3.1'de görüldüğü gibi bir veri kümesi X, kodlayıcı ile bir katlamalı kod kelimesi olan S'ye dönüştürülmektedir. S, gürültülü bir iletişim kanalında hareket ettikten sonra VD, bunu bir R vektörü olarak algılamakta ve gönderilen kod kelimesine en benzer olacak şekilde bir Y vektörü olarak çözümlemektedir.

Algoritma en benzer kodu çözmede, olasılık $P(R|X^{(m)})$ karşılaştırma esasına dayanmaktadır. Burada R algılanan seri ve $X^{(m)}$, gönderilen X serisinde en yüksek benzerlik olasılıklı olandır. Dal benzerlik fonksiyonu $P(R_i|X_i^{(m)})$'dur. Burada R_i algılanan kod serisinin i.dalı, $X_i^{(m)}$, gönderilen kod serisi $X^{(m)'}$in i. dalıdır. Yol

benzerlik fonksiyonu $P(R|X^{(m)})$, dal benzerlik fonksiyonu $P(R_i|X_i{}^{(m)})$'ndan oluşmaktadır [18].

$$P\left(R\middle|X^{(m)}\right) = \prod_{i=0}^{\infty} P\left(R_i\middle|X_i{}^{(m)}\right) \tag{36}$$

$$\log P\left(R\middle|X^{(m)}\right) = \sum_{i=1}^{\infty} \log P\left(R_i\middle|X_i{}^{(m)}\right) \tag{37}$$

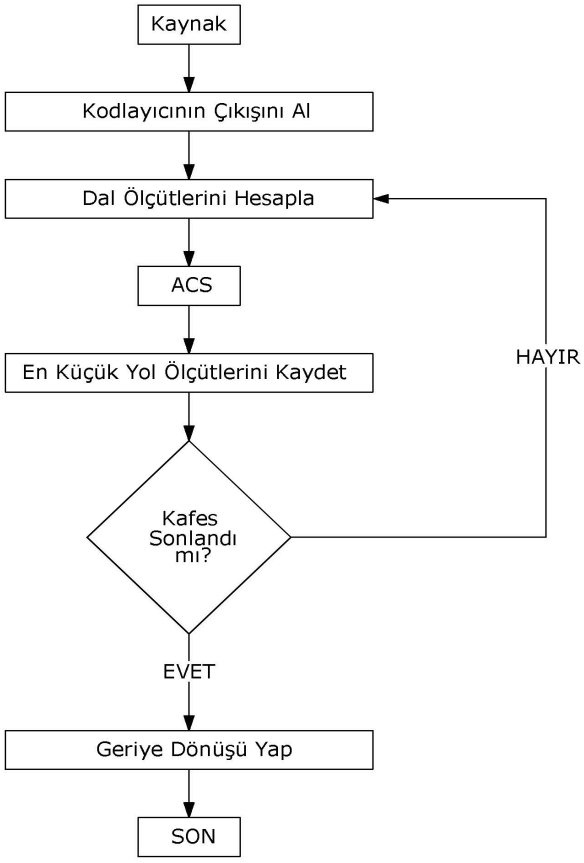
Denklem (36) ile anlatılmak istenen şudur; en benzer yol, en benzer dalların seçilerek toplanmasından oluşan bir yoldur.

d, algılanan ve gönderilen diziler arasındaki Hamming Uzunluğu; n, gönderilen bit sayısı; p hata olasılığı olmak üzere olasılık denklem (38)'deki gibi hesaplanmaktadır [18].

$$P(R|X) = (1-p)^{n-d} \text{ x } p^{d} \tag{38}$$

VA, hata olasılığını azaltmasındaki başarısından dolayı en uygun algoritma olarak tanımlanabilir. Algoritma aşağıdaki adımları izleyerek çalışmaktadır.

1. Kafes yollarının ölçütleri, Euclid Uzaklığı (EU) veya Hamming Uzaklığı (HD)'na dayanılarak hesaplanır.
2. Veri iletimi bitene kadar her kodlayıcı çıkışındaki kod için ACSU'da, en kısa yani en benzer yol seçilir. Bu yol belleğe alınır. Seçilen bu en kısa yollara, hayatta kalan yollar (SP) denir.
3. Uygun bir geriye dönüş algoritması kullanılarak her bir durumdan diğerine geçerken seçilen SP'nin her dalına karşılık kodlanan mesaj bulunur.

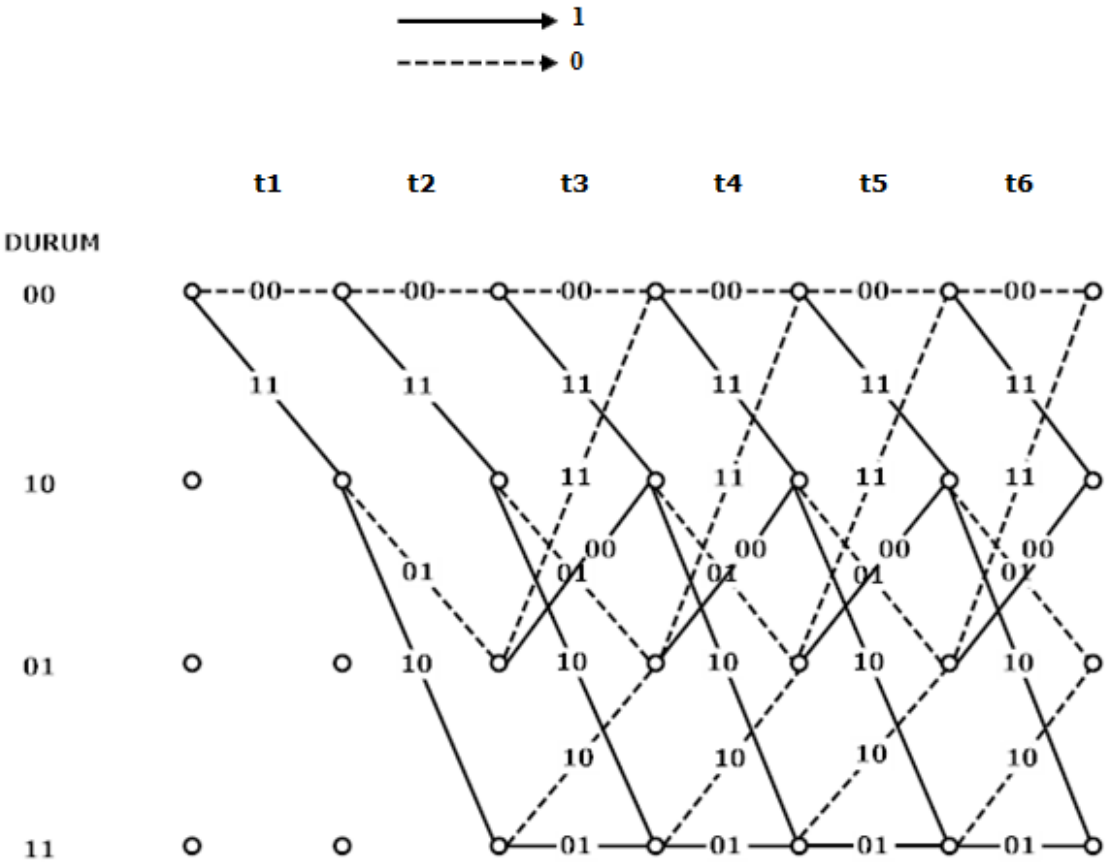


Şekil 3.2 Viterbi Kod Çözme Algoritması'nın akış şeması

3.2. Kafes Gösterimi

Kafes gösterimi, durum diyagramlarından üretilen bir gösterimdir. Kafes diyagramı sayesinde kodlayıcı bir durumdan diğerine geçerken hangi çıkış kodunu üretmektedir ve kodlayıcının girişine hangi mesaj gelmektedir kolaylıkla görülmektedir. Ayrıca VA için büyük önem taşıyan SP'ler bu diyagramdan takip edilmektedir.

Kodlayıcının kısıtlama uzunluğu K ise, kafes gösterimindeki durum sayısı 2^{K-1}'dır [13].



Şekil 3.3 Şekil 2.5'de gösterilen, Cconv (2, 1, 3) kodlayıcının, Şekil 2.6'de gösterilen durum diyagramından üretilen kafes gösterimi

Başlangıç anında kodlayıcı 00 durumundadır. Şekil 2.6'daki durum diyagramından da görüldüğü gibi kodlayıcının girişine 0 gelmesi durumunda, kodlayıcı 00 durumunda kalmaya devam etmektedir ve çıkış kodu olarak 00 üretmektedir. Girişe 1 gelmesi durumunda ise kodlayıcı 10 durumuna geçmektedir ve kod olarak 11 üretmektedir. Girişe gelen mesaj bitine göre bütün durumlar arası geçişler ve çıkış kodları temel alınarak Şekil 3.3'de gösterilen kafes diyagramı oluşmuştur.

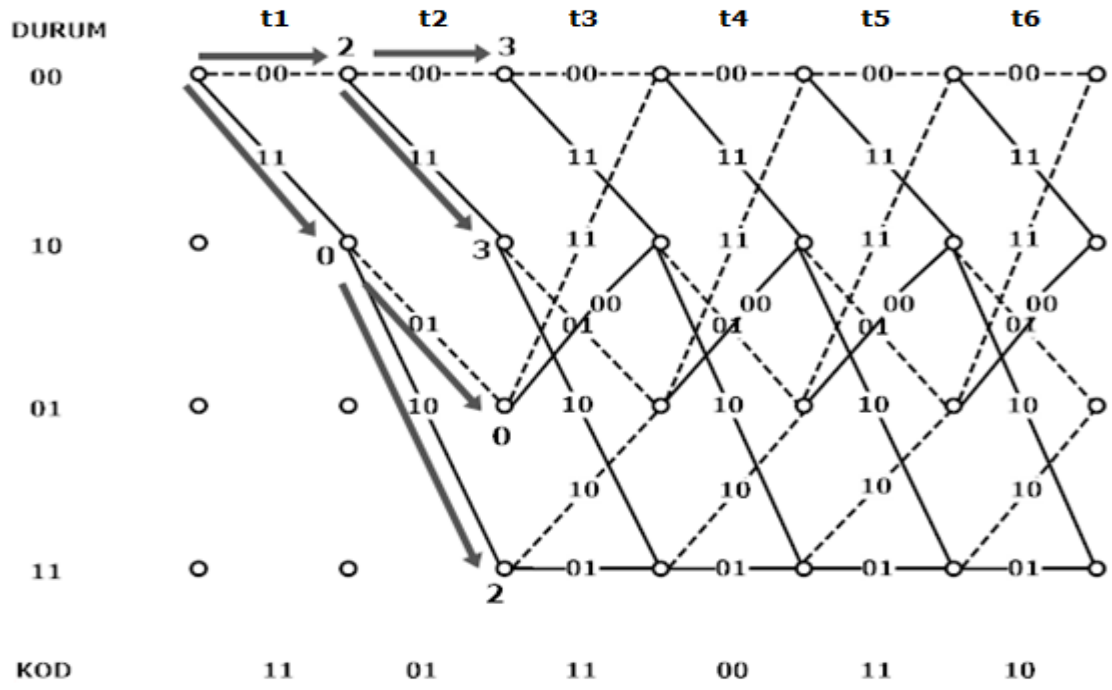
Her dal, o anda kodlayıcının, çıkış koduna ne kadar benzediğini gösteren ölçütler taşımaktadır. Kafesin sonuna kadar dal ölçütleri (BM) toplanarak devam etmektedir yani şu anki koda olan benzerlik ölçütünü taşıyan her yolun ölçütü,

kendisine gelinene kadarki geçilen dalların BM'lerinin ve kendi BM'sinin toplamıdır.

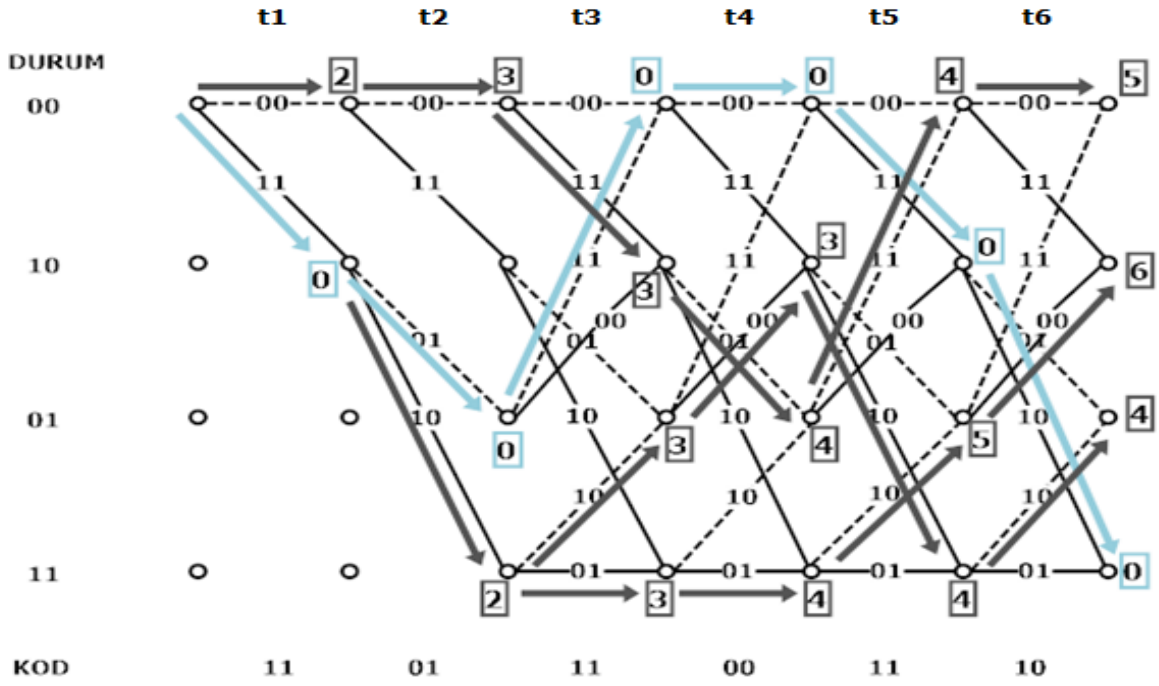
Şekil 2.5'de gösterilen kodlayıcı için bir bitlik mesajdan iki bitlik bir kod elde edilmektedir. Algılanan kod ile kodlayıcının çıkışının verdiği hesaplanan kod arasında karşılaştırma yapılarak dalların BM'leri hesaplanmaktadır. Bu iki kod birbirine ne kadar benziyorsa dalın BM'si o kadar düşük olmaktadır. Eğer kodlayıcıdan çıkıp, kod çözücüye gelen kodun geçtiği iletim kanalı gürültüsüzse algılanan kodda değişiklik olmamakta ve o kodu temsil eden BM, 0 olmaktadır. Kod hiç bozulmamışsa kafes diyagramının sonunda bu yolun toplam BM'si yani yol ölçütü (PM) 0 olmaktadır ve PM'si en düşük olan yani kodun doğru çözülmesini sağlayacak olan yoldur. VD tarafından en benzer yol yani SP olarak seçilmektedir.

Şekil 3.4'de, Şekil 2.5'deki kodlayıcı ile kodlanmış, 100011 mesaj dizisinin kafes diyagramı gösterilmiştir. Bu mesaj dizisi ile kodlayıcı çıkışından 11 01 11 00 11 10 kod dizisi elde edilmektedir. Bu diyagram şöyle çalışmaktadır; başlangıç koşulunda kodlayıcı 00 durumunda olduğu için kafes de 00 durumundan başlamaktadır. İlk saat darbesi için kod çözücünün girişine 11 kodu gelmiştir. Bundan dolayı bir sonraki saat darbesine kadar en benzer dal 11 dalıdır ve HD'ye göre BM'si 0'dır. 00 dalının ise iki biti de 11 kodundan farklıdır. Böylece bu dalın BM'si 2'dir. Bir sonraki, yani t2 zaman aralığında ise 01 kodu gelmektedir. t1 zaman aralığında 0 ölçütüne sahip dal üzerinden, 01 dalı seçilirse iki dalın toplam BM'si 0 olmakta ve bu iki zaman aralığı için, bu yol üzerinden gidilirse, kodun hatasız algılandığı anlamına gelmektedir. t1 zamanında BM'si 2 olan dalın, bir sonraki zaman için sahip olduğu olası yolların ikisi de 01 kodundan bir bit farklıdır. Dolayısıyla bu yolların hangisinden gidilirse gidilsin t2 zamanının sonunda, ilk iki zaman aralığındaki toplam BM, 3 olacaktır. Böylece kod ilk iki zaman aralığı için 3 bit yanlış algılanmış olacaktır. Kodlayıcının girişine gelen mesaj dizisi bitene kadar bu işlem adımları BM'ler toplanarak devam etmektedir. Böylece kodlama işleminin bittiği ana kadar gelen bütün olası yolların PM'leri hesaplanmaktadır. Şekil 3.5'de kafes tamamlandığında SP olarak ortaya çıkabilecek yollardan dördü ve bunların PM'leri verilmiştir. VD, prensip olarak PM'si düşük olan yolu seçmektedir. Burada şekil 3.6'da gösterilen, ölçütü 0 olan yol seçilecek ve geriye dönme algoritması o yol

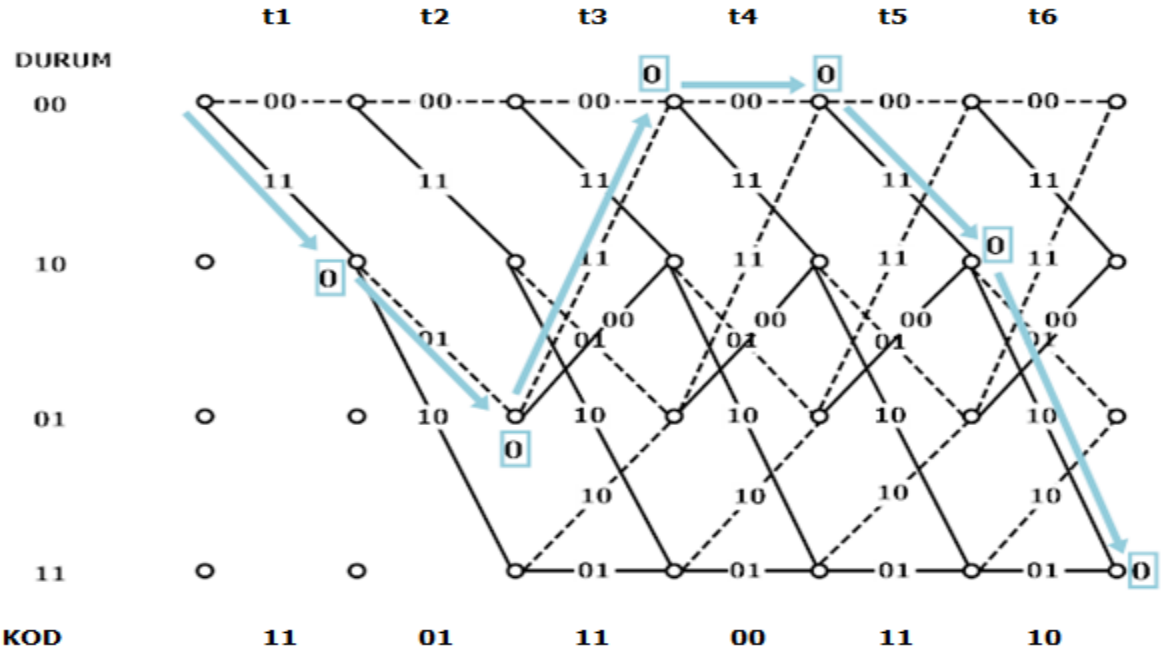
üzerinden uygulanacaktır. Böylelikle girişe gelen koda en yüksek oranda yaklaşılmaktadır.



Şekil 3.4 Şekil 2.5'deki kodlayıcı ile kodlanmış, 100011 mesaj dizisinin kafes diyagramı

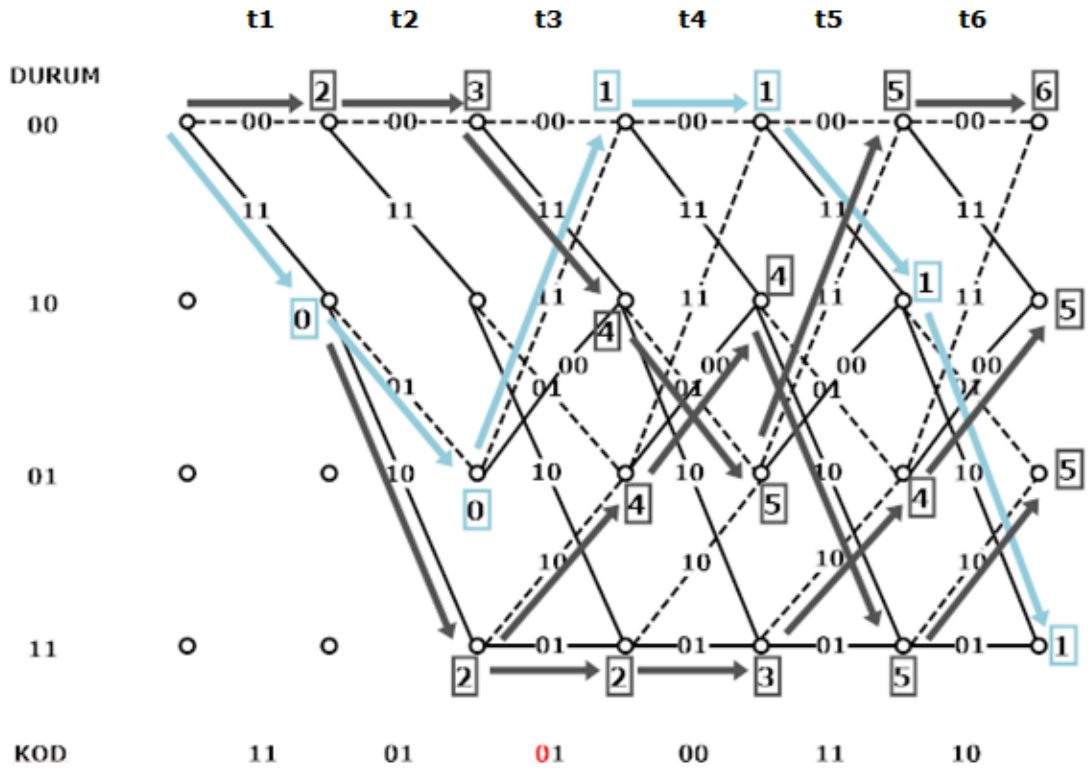


Şekil 3.5 Şekil 2.5' deki kodlayıcı ile kodlanmış, 100011 mesaj dizisi için olası dört SP'nin, PM' leri gösterimi



Şekil 3.6 Şekil 2.5'deki kodlayıcı ile kodlanmış, 100011 mesaj dizisi için en düşük PM'ye sahip yolun gösterimi

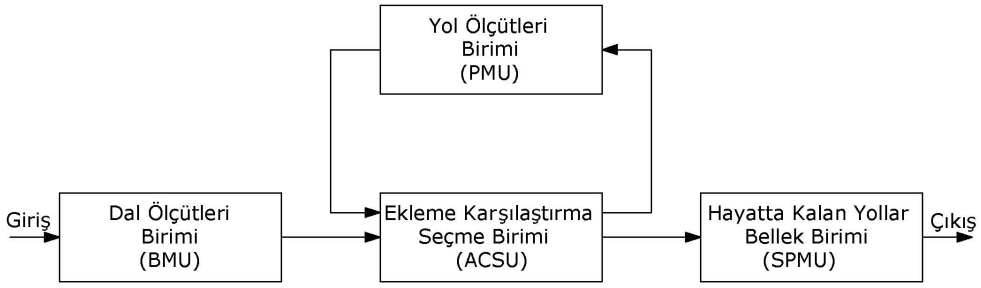
Kodlayıcı çıkışından 11 01 11 00 11 10 olarak gelen kod dizisinin kanaldaki gürültüden dolayı kodlayıcı tarafından 11 01 01 00 11 10 olarak algılandığını düşünelim. Yani beşinci bit yanlış algılanmış olsun. Şekil 3.7'de gösterildiği gibi 5. bit hatalı olduğu için ilk iki zaman aralığında en küçük toplam BM, 0 olmasına rağmen t3 zaman aralığında BM, o bitdeki hatadan dolayı artık 1 olmaktadır. Kafes tamamlandığında başka hatalı bit olmadığı için bu yolun, PM'si 1 olacaktır. Diğer SP'lerle kıyaslandığında en küçük PM'ye sahip olan yol yine bu yol olduğu için kod çözücü bu yol üzerinden geriye dönecek ve kodu 100011 olarak çözecektir. Yani kod, kod çözücüye kanaldaki gürültüden dolayı yanlış gelmesine rağmen VD sayesinde kodlayıcıya geldiği şekilde, doğru çözülmektedir.



Şekil 3.7 Şekil 2.5'deki kodlayıcı ile kodlanmış, 100011 mesaj dizisi için yanlış kodlama halinde oluşan seçilmiş dört SP için PM'ler

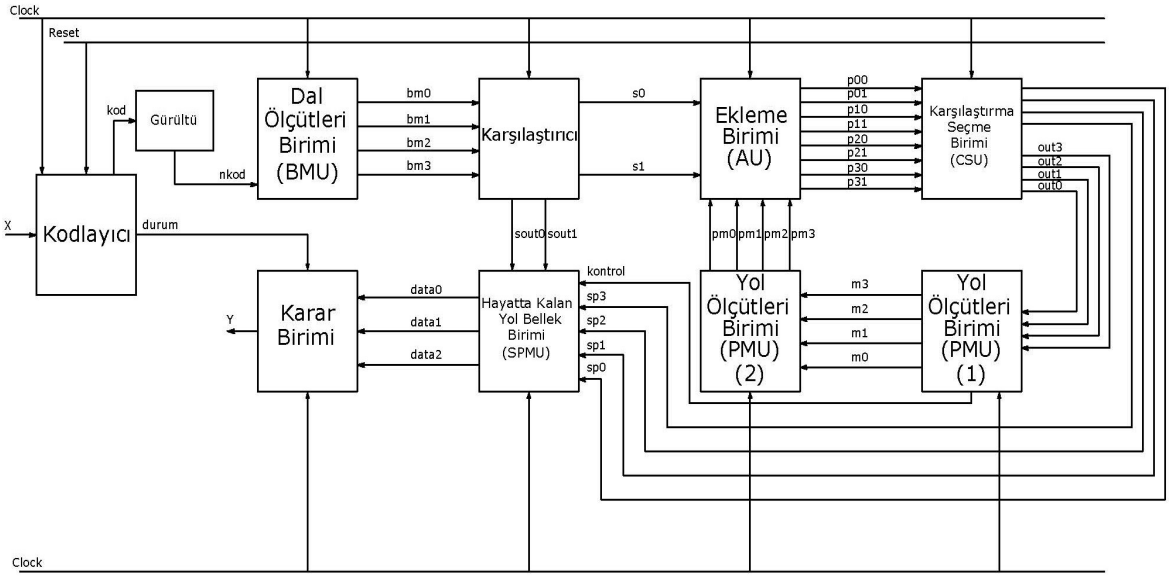
4. VITERBI KOD ÇÖZÜCÜ TASARIMI

Bir Viterbi Kod Çözücü, Şekil 4.1'de gösterildiği gibi Dal Ölçütleri, Ekleme Karşılaştırma Seçme, Yol Ölçütleri ve Hayatta Kalan Yol Hafıza Birimi olmak üzere dört temel birimden oluşmaktadır.



Şekil 4.1 VD'nin temel birimleri

Bu birimlerin dışındaki birimler, bu dört temel birimin bölünmesiyle oluşturulup tasarımdan tasarıma farklılık gösterebilir. Ancak temelinde bu dört birimin yaptıklarını gerçekleştirmektedirler. Bu çalışmada tasarlanan kodlayıcı, gürültü birimi, VD'nin birimleri ve birimler arası bağlantılar Şekil 4.2'de verilmiştir. Bu tasarım için kodlayıcı olarak Şekil 2.5'de gösterilen katlamalı kodlayıcı kullanılmıştır. X giriş serisi, kodlayıcı ile kodlandıktan sonra, bu kodun gürültü biriminden geçtikten sonra üzerine gürültü binmiş hali 'nkod' olarak çıkmaktadır.



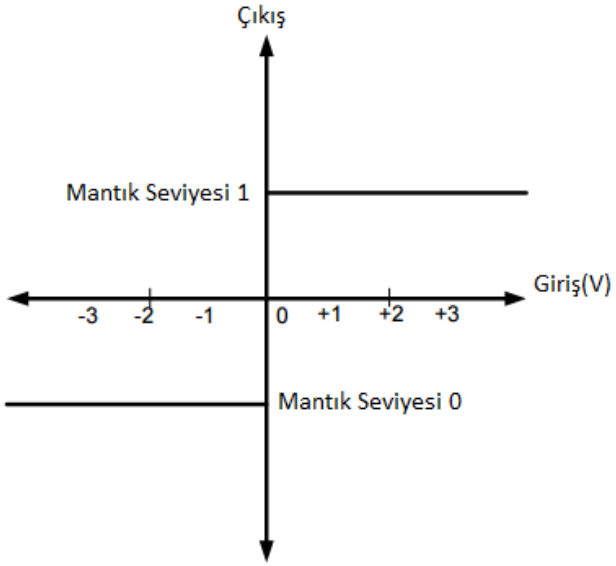
Şekil 4.2 Tasarlanan VD'nin yapısı

4.1. Dal Ölçütleri Birimi (BMU)

İlk birim Dal Ölçütleri Birimi olarak adlandırılır. Bu birimde algılanan kod ile kodlayıcının çıkışı arasındaki fark BM olarak atanmaktadır. Şekil 4.2'de gösterildiği gibi kod, katlamalı kodlayıcıdan çıkıp gürültü biriminde olduğundan farklı olarak algılanma ihtimali ile BMU'ya gelmektedir. Burada her saat darbesinde algılanan kod, olması gereken kod değerleri ile karşılaştırılmaktadır. Böylece 'bm0, bm1, bm2, bm3' BM'ler hesaplanmakta ve bu BM'ler, hayatta kalan olarak seçilmek üzere ACSU'ya gönderilmektedir. BM'lerin hesaplanması için sert veya yumuşak karar yöntemine göre Hamming Uzunluğu (HD) ya da Euclid Uzunluğu (ED) kullanılabilir.

Algılanan analog sinyal, farklı kuantalanmış enerji seviyelere çevrilebilir. Algılanan sinyal, Şekil 4.3'de gösterildiği gibi 0 ve 1 olmak üzere iki enerji seviyesine dönüştürülürse buna sert karar kodlama yöntemi, eğer giriş sinyali iki enerji seviyesinden daha fazla seviyeye kuantalanırsa buna yumuşak karar kodlama

denilmektedir. Yumuşak karar yöntemi, sert karar yöntemine göre giriş sinyali ile ilgili daha çok bilgi vermektedir. Ancak daha yüksek bir karmaşıklığa sahiptir. Sert karar, HD'ler; yumuşak karar, ED'ler hesaplanarak yapılmaktadır.

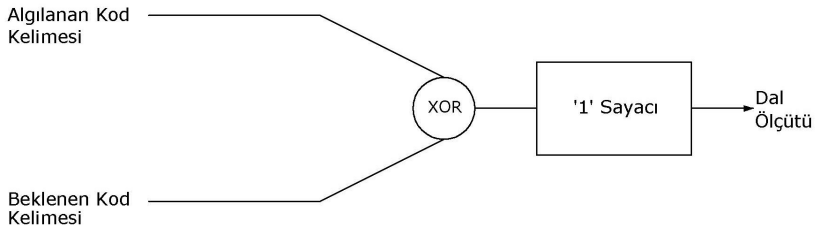


Şekil 4.3 Sert karar yöntemi

ED, iki nokta arasındaki lineer uzaklıktır. Bir nokta p1, (x_1, y_1) kordinatlarına ve diğer bir nokta p2, (x_2, y_2) koordinatlarına sahip ise, bu iki nokta arasındaki ED denklem (39)'da verildiği gibidir.

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{39}$$

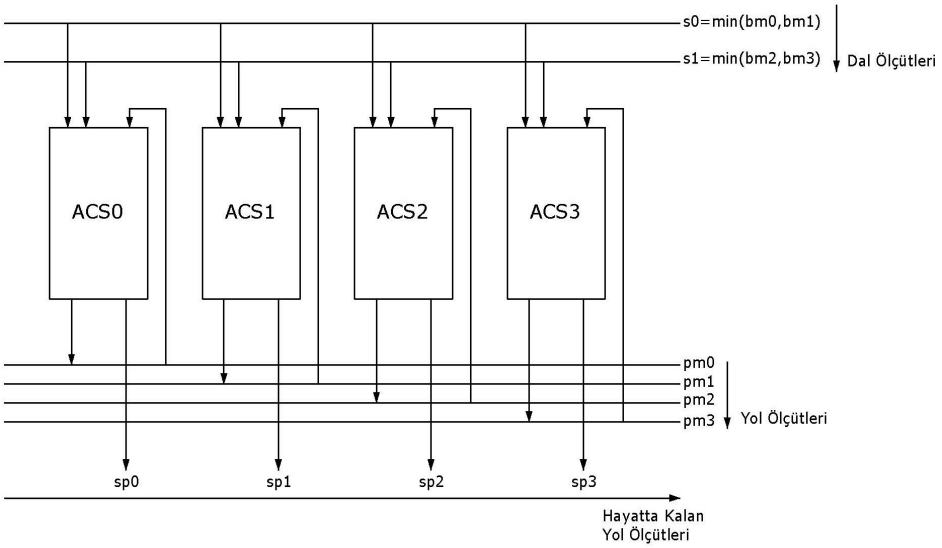
Bu çalışma için tasarlanan kod çözücüler, güç verimli tasarım hedeflendiği için sert karar yöntemini kullanarak çalışmaktadır. Şekil 4.4'de sert karar yöntemini yani HD'yi hesaplayarak çalışan bir BMU'nun iç yapısı gösterilmiştir. Algılanan kod ile kodlayıcının çıkışını bit bit karşılaştırmakta, her ikisi arasında kaç bit farklılık var ise HD'ye bu değeri atamaktadır.



Şekil 4.4 Hamming Uzaklığı'na göre çalışan BMU [6]

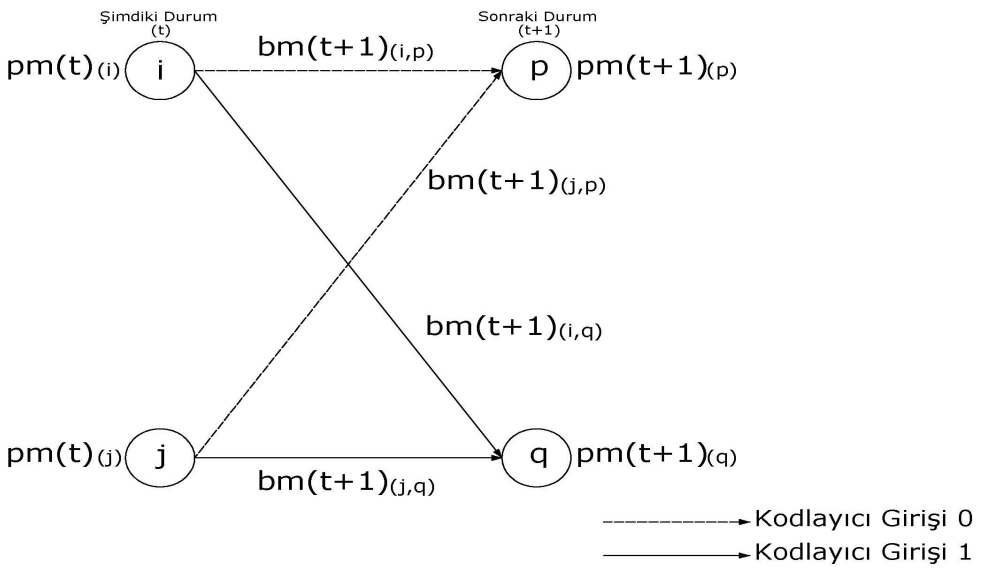
Gerçekleştirilen ve Şekil 4.2'de gösterilen tasarımda bu birimin çıkışına bir karşılaştırıcı birim eklenerek BM'ler, ACSU'dan önce bir defa daha kıyaslanmıştır ve BM'leri yüksek olan bazı dallar, ACSU'ya gitmeden önce elenerek kod çözücünün hesaplama yükü azaltılmıştır. Bu karşılaştırıcıdan karşılaştırma sonrasında seçilen dallar 's0 ve s1' olarak atanmıştır.

4.2. Ekleme Karşılaştırma Seçme Birimi (ACSU)



Şekil 4.5 ACSU'nun girişleri ve çıkışları

ACSU, mevcut PM'ler ile güncel BM'lerin toplandığı, karşılaştırıldığı ve en küçük PM'lere sahip yolun seçildiği birimdir. Tasarlanan kod çözücünün, ACS Birimi'nin nasıl çalıştığı Şekil 4.5'de gösterilmiştir. BM'ler, Şekil 4.2'de gösterilen karşılaştırıcıdan ACSU'ya gelmektedir. Bu ölçütler, bu birimde bir önceki zaman aralığından gelen PM'ler ile toplanmaktadır. Bu işlemden sonra oluşan yeni PM'ler karşılaştırılmakta ve en küçük PM'ye sahip olan yollar, SP'ler olarak seçilerek SPMU'ya gönderilmektedir. Ayrıca bu zaman aralığını da kapsayan PM'ler hesaplamaların yapılması için PMU'ya gitmektedir.



Şekil 4.6 ACSU'da ekleme işleminin kelebek yapısı [18]

Şekil 4.6'da ACSU'da gerçekleşen ekleme işlemi gösterilmiştir. Kod çözücü i ve j durumunda iken kodlayıcının girişine gelen mesaja göre p veya q durumuna geçmektedir. p ve q durumlarındaki PM'ler pm_{t+1}^{p} ve pm_{t+1}^{q}; i ve j durumunda sahip oldukları PM'ler pm_{t}^{i} ve pm_{t}^{j} 'ye, p ve q durumuna geçmek için gerekli BM'lerin eklenmesi ile bulunmaktadır.

$bm_{t+1}^{i,p}$, $bm_{t+1}^{j,p}$; i durumunu ve j durumunu, p durumuna geçiren BM'ler ve $bm_{t+1}^{i,q}$, $bm_{t+1}^{j,q}$; i durumunu ve j durumunu, q durumuna geçiren BM'ler olmak

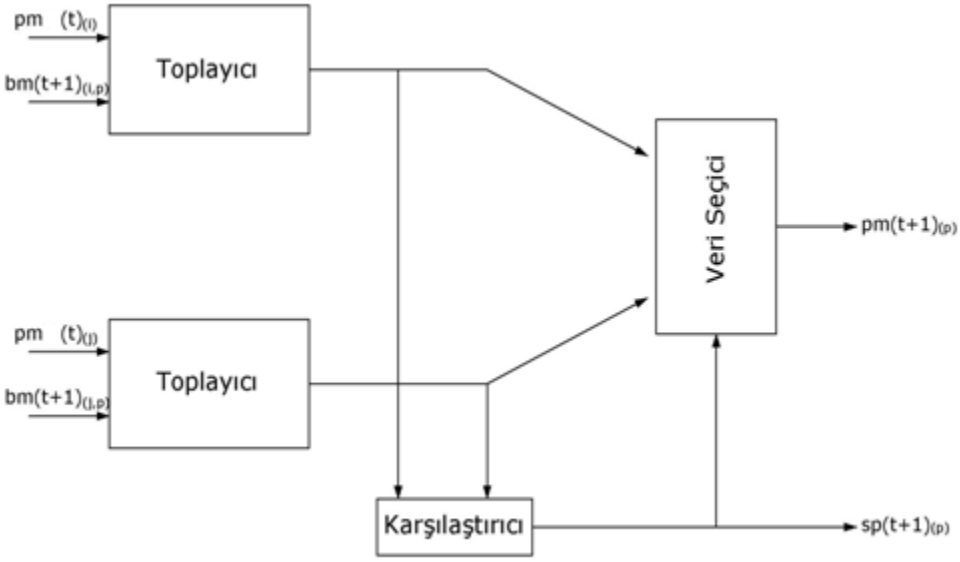
üzere, Şekil 4.6'daki kelebek diyagramı denklem (39), (40), (41), (42) ile ifade edilmektedir.

$$pm_{t+1}^{p} = pm_{t}^{i} + bm_{t+1}^{i,p} \quad (39)$$

$$pm_{t+1}^{p} = pm_{t}^{j} + bm_{t+1}^{j,p} \quad (40)$$

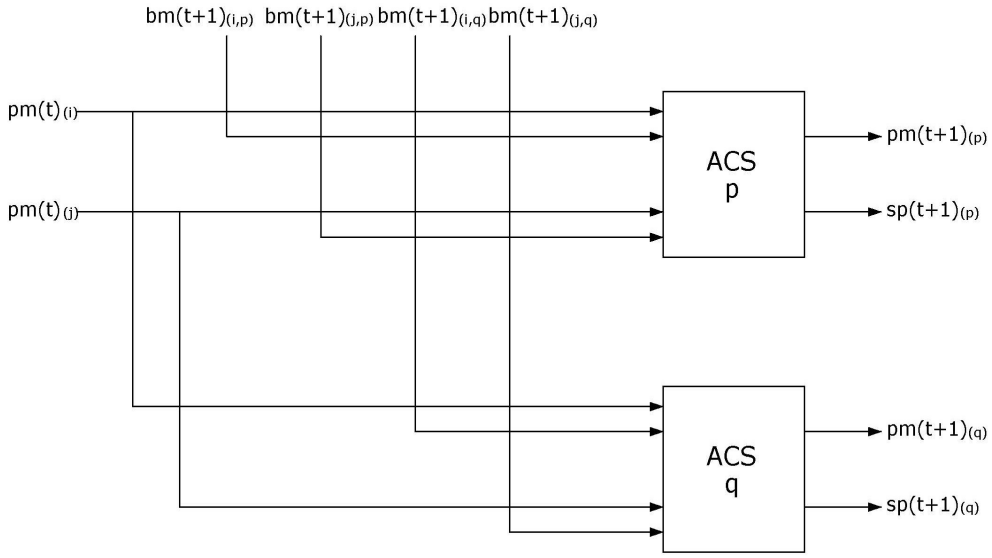
$$pm_{t+1}^{q} = pm_{t}^{i} + bm_{t+1}^{i,q} \quad (41)$$

$$pm_{t+1}^{q} = pm_{t}^{j} + bm_{t+1}^{j,q} \quad (42)$$



Şekil 4.7 p durumu için ACSU [18]

Şekil 4.7'deki karşılaştırıcı p durumuna giden iki yolun, BM'lerini karşılaştırarak yollardan bir tanesini seçmektedir. Seçtiği yolu 1 veya 0 olarak kodlamaktadır. Bu değer veri seçicinin seçme ucuna gitmekte ve SP'nin PM'lerinin olduğu kanalı aktif hale getirerek, seçicinin çıkışına bu yolun ölçütlerinin gitmesini sağlamaktadır.



Şekil 4.8 p ve q için ACSU

VD'yi i ve j durumundan p ve q durumuna geçiren ACSU Şekil 4.8'de gösterilmiştir. Bu birimde p ve q durumları için (43) ve (44)'de verilen denklemler gerçekleşmektedir [18].

$$pm_{t+1}^{p} = \min[(pm_{t}^{i} + bm_{t+1}^{i,p}), (pm_{t}^{j} + bm_{t+1}^{j,p})] \quad (43)$$

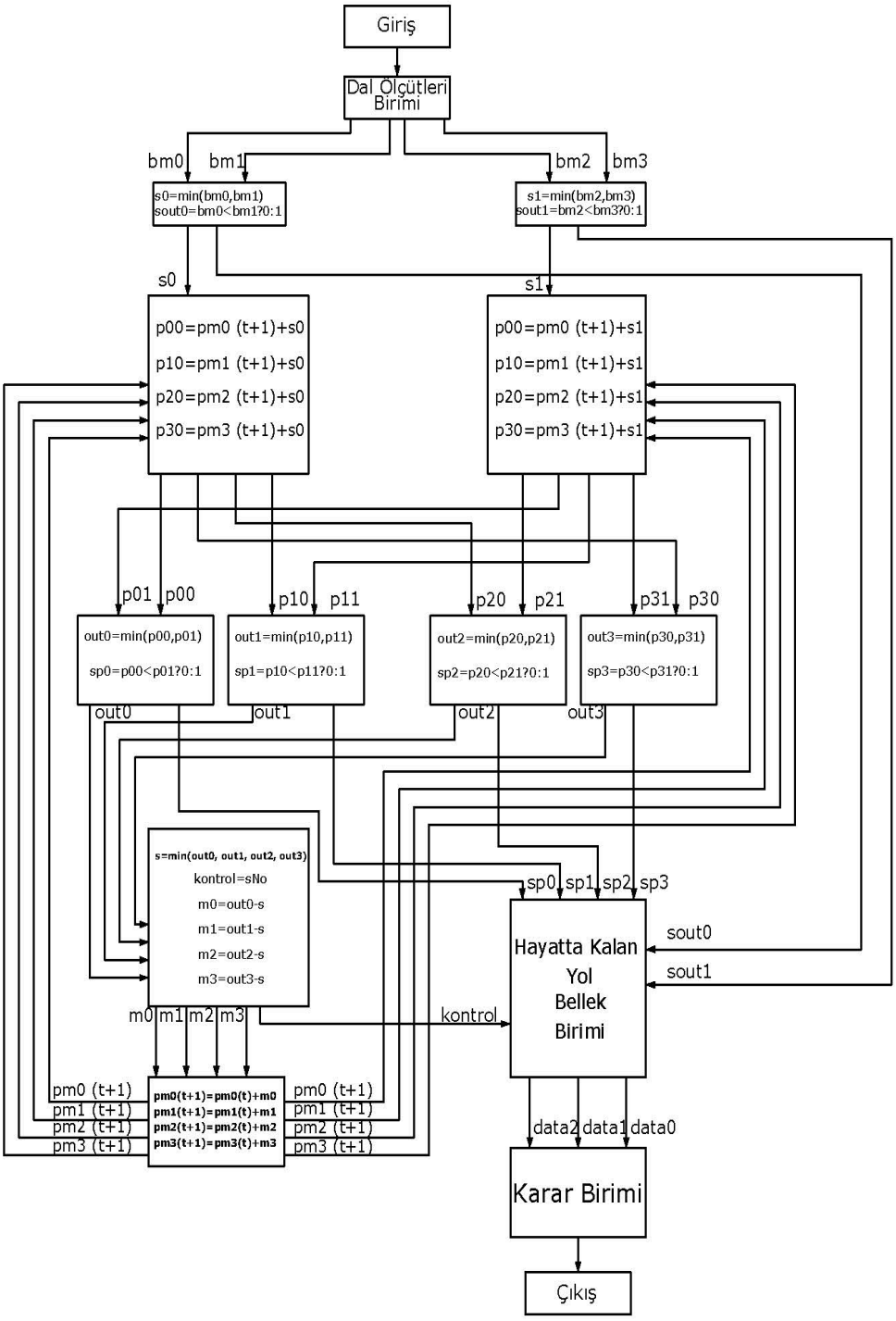
$$pm_{t+1}^{q} = \min[(pm_{t}^{i} + bm_{t+1}^{i,q}), (pm_{t}^{j} + bm_{t+1}^{j,q})] \quad (44)$$

Şekil 4.2'de gösterildiği gibi bu tasarımda ACSU, Ekleme (AU) ve Karşılaştırma Seçme Birimi (CSU) olmak üzere iki birim halinde tasarlanmıştır. BMU'dan gelen BM'ler, bir önceki zaman aralığından gelen PM'ler ile Ekleme Birimi'nde toplanmaktadır. Elde edilen yeni PM'ler, 'p00, p01, p10, p11, p20, p21, p30, p31' olmak üzere Karşılaştırma Seçme Birimi'ne gelmektedir. Gelen PM'ler birbirleri ile kıyaslanarak, en küçük ölçütlere sahip yollar, SP'ler olarak seçilmektedir. SP'ler, 'sp0, sp1, sp2, sp3', kaydedilmek ve geriye dönüşte kullanılmak üzere SPMU'ya gönderilmektedir. SP'lerin PM'leri, 'out0, out1, out2, out3', kod çözücünün diğer birimlerinde gerekli hesaplamalar yapıldıktan sonra bir sonraki duruma geçişte oluşacak olan yeni BM'ler ile toplanılmak üzere Ekleme Birimi'ne dönmektedir.

4.3. Yol Ölçütleri Birimi (PMU)

Bu birim SP'lerin, istenilen andaki BM'lerinin bulunduğu ve o andaki PM'lerinin hesaplandığı birimdir. Bu çalışmada PMU Şekil 4.2'de gösterildiği gibi iki birim halinde tasarlanmıştır. Karşılaştırma Seçme Birimi'nde, SP'ler olarak seçilen yolların, PM'leri, PMU (1)'e gelmektedir. Bu birimde, gelen SP'ler arasında karşılaştırma yapılmakta ve en küçük PM'ye sahip yol, en küçük yol olarak belirlenmektedir. Bu yolun PM'si, SP'lerin PM'lerinden çıkartılarak, her bir SP'nin, kendi PM'sine katkısı bu zaman aralığının sonunda bulunmaktadır ve bu değerler, 'm0, m1, m2, m3' olmak üzere, PMU (2)'ye giderek bir önceki zamandan kalma PM'ler ile toplanmaktadır. Böylece bu zaman aralığından gelen PM'ler 'pm0, pm1, pm2 ve pm3', kod için hesaplanan yeni BM'lerle toplanmak üzere Ekle Birimi'ne gitmektedir.

PMU'dan itibaren anlatılan işlemler karmaşık ve iç içe geçmiş hesaplamalar içermektedir. Şekil 4.2.'de gösterilen tasarlanan VD'nin, ACSU ve PMU'yu kapsayan bütün birimlerinin içerisinde yapılan hesaplamalar Şekil 4.9'da gösterilmiştir.



Şekil 4.9 Tasarlanan VD'nin ACSU'su ve PMU'su içerisinde gerçekleşen işler

4.4. Hayatta Kalan Yol Bellek Birimi (SPMU)

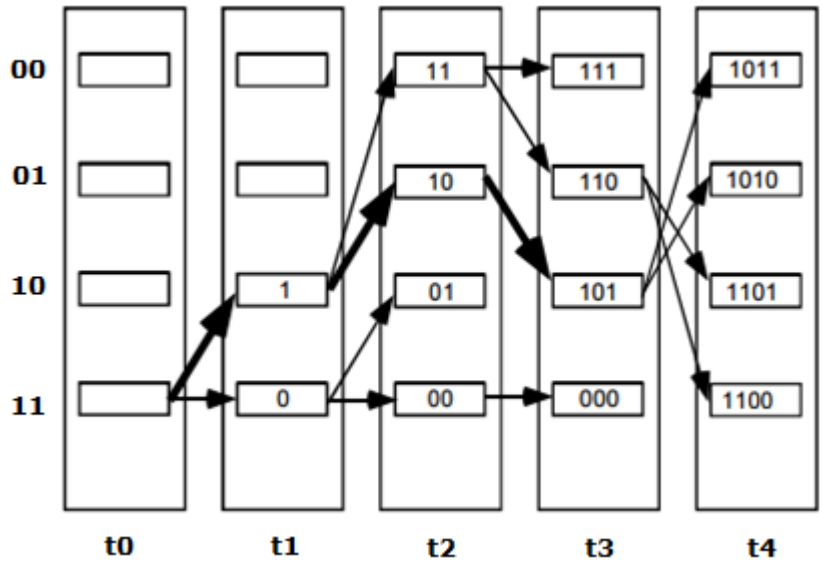
SPMU, kodu çözmek için takip edilecek SP'lerin saklandığı birimdir. VD, gelen mesajın ne olduğunu çözmek için geri izleme (TB) ve kaydedici değiştirme (RE) olmak üzere iki yöntem kullanmaktadır. TB, daha az donanım ihtiyacına sahip olmasına rağmen daha uzun zamanda çözümleme yapmaktadır. RE'nin ise donanımsal olarak gerekliliği dolayısıyla güç tüketimi fazladır.

Bu çalışmada TB yöntemiyle geriye dönülerek kod çözülmüştür. Bu yöntem için bir SPMU'ya ihtiyaç vardır. SP'ler burada hafızaya alınmakta ve geriye dönüşte dal başına kodlanan bit çözülmektedir. TB algoritması, PM'si en düşük yol üzerine uygulanmaktadır. İlk kaydedilen PM'den itibaren durumlar arası geçişlerde kodlanan bitler çözülmekte, kodlayıcıya gelen son bite kadar bu işlem devam etmektedir.

4.4.1. Kaydedici Değiştirme (RE)

RE, yalın bir yöntemdir. Ancak yüksek miktarda güç tüketimi ve alan kaplamasından dolayı kısıtlama uzunluğu büyük VD'lar için, RE tercih edilmemektedir. Şekil 4.10'da gösterildiği gibi SP'ler için başlangıç durumundan şimdiki duruma kadar her durumu içeren bir kaydedici tanımlanmaktadır. Bu kaydedici kodlanan bilgi yolunu tutmaktadır.

Son durumun kaydedicisi, çıkış bilgi serisini içerdiği için RE yöntemi geriye dönme ihtiyacını ortadan kaldırmaktadır. Bu sebeple TB'ye göre daha hızlı çalışmaktadır. SP bilgisi için her kaydedicinin en küçük değerlikli bitine başvurulmakta ve her durumda bir sonraki biti bulmak için sola kaydırma işlemi yapılmaktadır. Bundan dolayı her kaydedici en az değerli bitten en değerli bite doğru o ana kadarki SP değerlerini tutmaktadır.



Şekil 4.10 Kaydedici değiştirme algoritması [6]

4.4.2. Geri İzleme (TB)

Bu çalışmada güç verimli çalışan kod çözücüler tasarlamak amaçlandığı için geriye dönüş TB yöntemiyle yapılmıştır. Tasarımda SPMU'ya; karşılaştırıcıdan, BM'ler karşılaştırıldıktan sonra seçilen BM'lere ait değerler, 'sout0 ve sout1'; ACSU'dan, seçilen SP ölçütlerine ait değerler, 'sp0, sp1, sp2, sp3'; PMU'dan SP'ler kıyaslanarak seçilen, en küçük yol değerine sahip yolun numarasının atandığı bir 'kontrol' değeri gelmekte ve belleğe alınmaktadır. Bu değerler her saat darbesinde Karar Birimine gönderilmektedir. Karar Birimi'nde uygulanan uygun bir TB algoritması sayesinde, en küçük PM'ye sahip olan yolun hangi dallar üzerinden geldiği bulunmakta ve bu yol takip edilerek kodlayıcının girişine gelen mesaj çözülmektedir.

SPMU'deki değerler, Karar Birimi'ne gönderilirken aşağıda verilen atamalar yapılmıştır.

data0[0] ← *sout0*
data0[1] ← *sout1*

data1[0] ← sp0

data1[1] ← sp1

data1[2] ← sp2

data1[3] ← sp3

data2 ← kontrol

Bu ‘data0, data1, data2’ sinyallerini alan Karar Birimi’nde uygulanan TB algoritmasının sözde kodu aşağıda verilen şekildedir.

Her saat darbesininin yükselen kenarında
eğer
data2 = 2’b00 ve data1[0] = 1’b0 ve data0[0] = 1’b0 ise
kod çözücü çıkışı ← 2’b00;
değilse
data2 = 2’b00 ve data1[0] = 1’b0 ve data0[0] = 1’b1 ise
kod çözücü çıkışı ← 2’b01;
değilse
data2 = 2’b00 ve data1[0] = 1’b1 ve data0[1] = 1’b0 ise
kod çözücü çıkışı ← 2’b10;
değilse
data2 = 2’b00 ve data1[0] = 1’b1 ve data0[1] = 1’b1 ise
kod çözücü çıkışı ← 2’b11;
değilse
data2 = 2’b01 ve data1[1] = 1’b0 ve data0[0] = 1’b0 ise
kod çözücü çıkışı ← 2’b00;
değilse
data2 = 2’b01 ve data1[1] = 1’b0 ve data0[0] = 1’b1 ise
kod çözücü çıkışı ← 2’b01;
değilse
data2 = 2’b01 ve data1[1] = 1’b1 ve data0[1] = 1’b0 ise
kod çözücü çıkışı ← 2’b10;
değilse
data2 = 2’b01 ve data1[1] = 1’b1 ve data0[1] = 1’b1 ise
kod çözücü çıkışı ← 2’b11;
değilse
data2=2’b10 ve data1[2]=1’b0 ve data0[0]=1’b0 ise

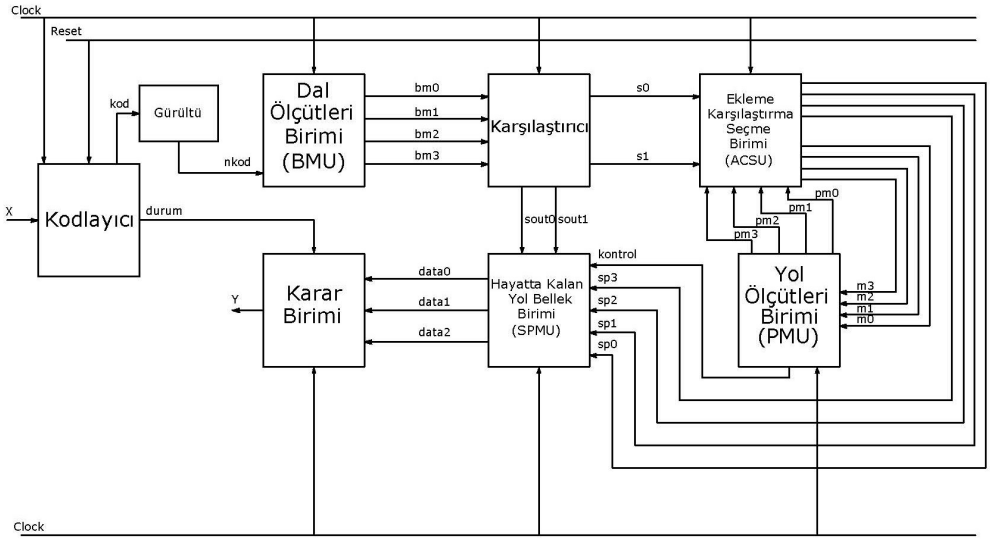
kod çözücü çıkışı ← 2'b00;
değilse
data2 = 2'b10 ve data1[2] = 1'b0 ve data0[0] = 1'b1 ise
kod çözücü çıkışı ← 2'b01;
değilse
data2 = 2'b10 ve data1[2] = 1'b1 ve data0[1] = 1'b0 ise
kod çözücü çıkışı ← 2'b10;
değilse
data2 = 2'b10 ve data1[2] = 1'b1 ve data0[1] = 1'b1 ise
kod çözücü çıkışı ← 2'b11;
değilse
data2 = 2'b11 ve data1[3] = 1'b0 ve data0[0] = 1'b0 ise
kod çözücü çıkışı ← 2'b00;
değilse
data2 = 2'b11 ve data1[3] = 1'b0 ve data0[0] = 1'b1 ise
kod çözücü çıkışı ← 2'b01;
değilse
data2 = 2'b11 ve data1[3] = 1'b1 ve data0[1] = 1'b0 ise
kod çözücü çıkışı ← 2'b10;
değilse
data2 = 2'b11 ve data1[3] = 1'b1 ve data0[1] = 1'b1 ise
kod çözücü çıkışı ← 2'b11;

4.5. Güç Etkin Viterbi Kod Çözücü Tasarımı

Güç etkin bir VD için en çok kullanılan yaklaşım, hesaplama yükünü dolayısıyla donanımsal karmaşıklığı azaltmak ve kod çözücünün güç verimli çalışmasını sağlamaktır.

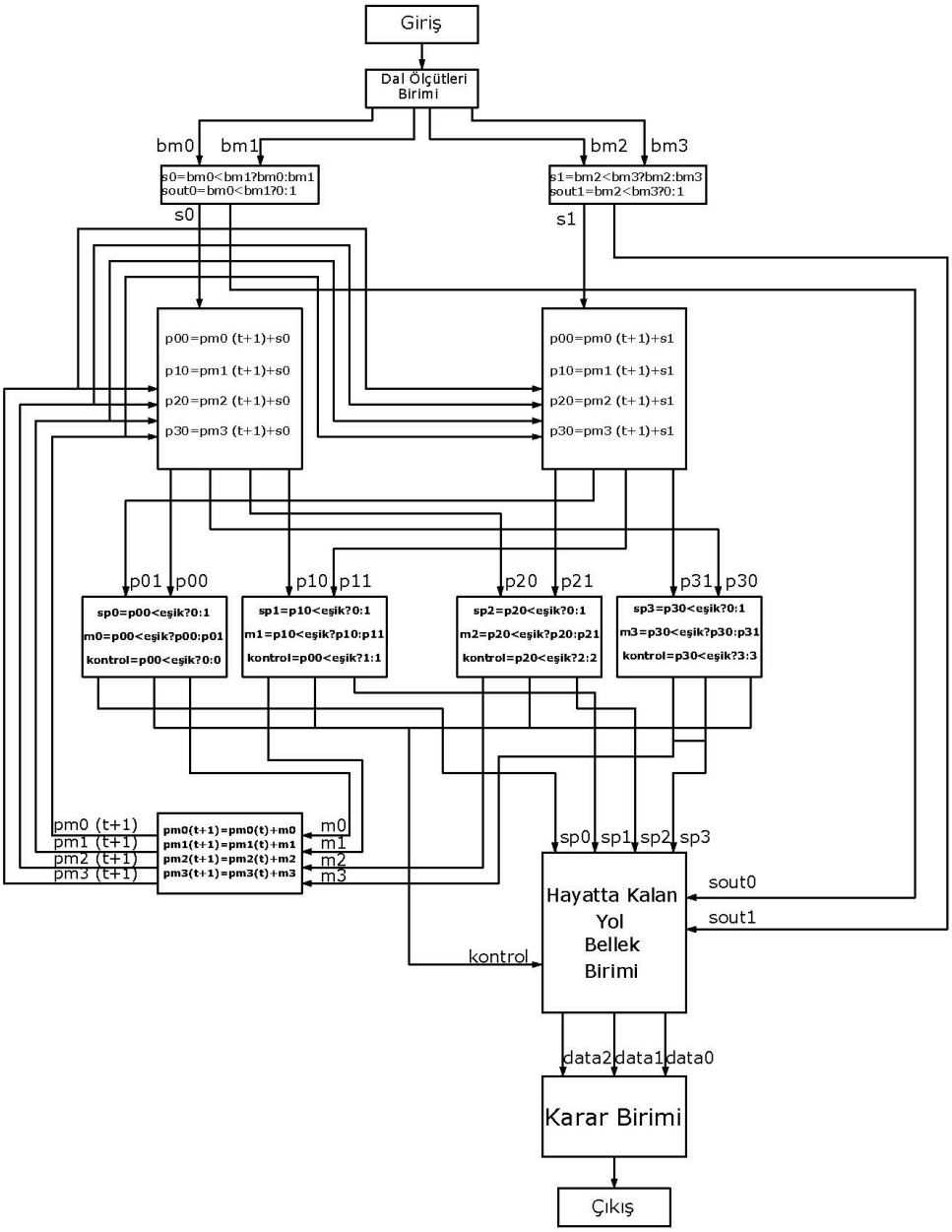
Bu tez çalışmasında ilk tasarlanan kod çözücüde BMU'nun çıkışına eklenen karşılaştırıcı birim sayesinde BM'leri büyük olan dalların bazıları elenmiş ve hesaplamaya katılmamıştır. Bundan dolayı aslında tasarlanan ilk kod çözücü de klasik bir VD'ye göre güç etkin çalışmaktadır. İlk tasarlanan kod çözücüyü daha güç verimli çalışır hale getirmek amacıyla Şekil 4.11'de gösterilen kod çözücü

tasarlanmıştır. Bu güç verimli VD'nin, PM'leri için bir eşik değer tanımlanmıştır ve bu değer üzerinde kalan yollar birbiriyle karşılaştırılmadan elenerek kod çözücünün hesap yükü azaltılmıştır. Burada eşik değerinin seçimi önemlidir. Çünkü düşük seçilen bir eşik değer kod çözücünün hesap karmaşıklığını yüksek oranda azaltmasına rağmen SP olarak seçilecek yani takip edilerek kodun çözülmesini sağlayacak yolun elenmesine sebep olabilir. Bu da kod çözücünün en temel hedefinden uzaklaşması yani kodu doğru çözememesi anlamına gelmektedir. Bu nedenle seçilen eşik değer kod çözücünün doğru çözümleme yapmasına engel olmadan, en güç etkin şekilde çalışmasını sağlayacak en iyi değer olmalıdır.



Şekil 4.11 Güç etkin tasarlanan VD'nin yapısı

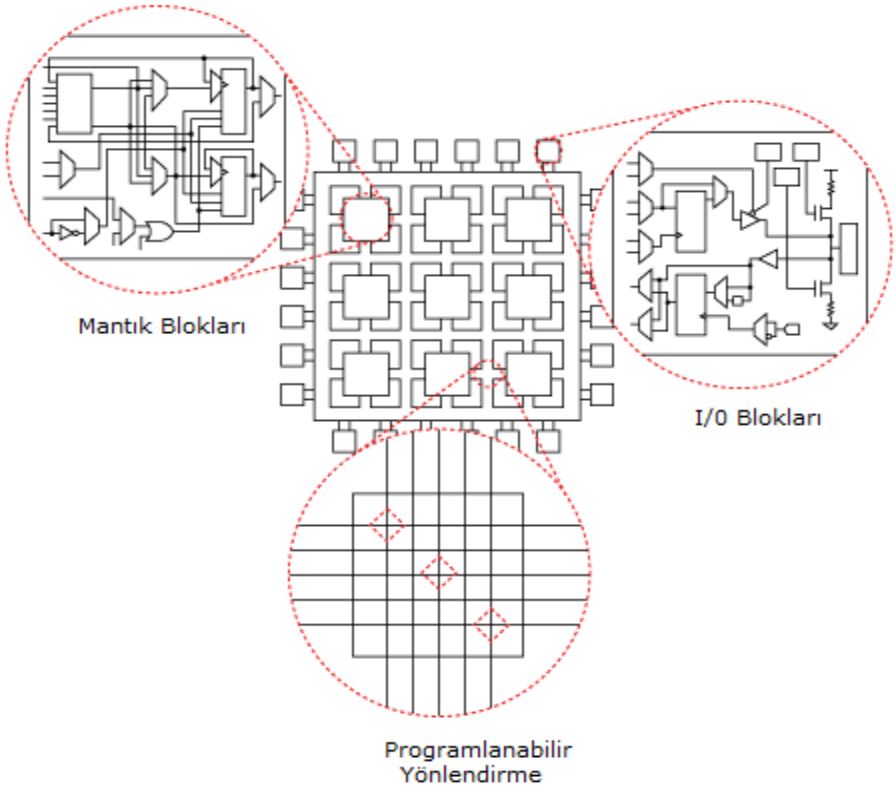
Şekil 4.11'de görüldüğü gibi güç verimli VD, ACSU ve PMU'nun tek birim olarak tasarlanması haricinde Şekil 4.2'de gösterilen ilk tasarlanan kod çözücü ile aynı birimlere sahiptir. ACSU ve PMU'yu kapsayan bütün birimlerinin içerisinde yapılan hesaplamalar Şekil 4.12'de gösterilmiştir ve burada görüldüğü gibi bahsedilen eşik değer 'eşik' olarak ACSU içinde tanımlanarak hesap yükünün azaltılmasına buradan itibaren başlanmıştır.



Şekil 4.12 Güç etkin tasarlanan VD'nin ACSU'su ve PMU'su içerisinde gerçekleşen işler

4.6. FPGA (Alanda Programlanabilir Kapı Dizileri)

FPGA'ler, kullanıcı gereksinimine göre herhangi bir sayısal sisteme programlanabilen tümleşik yarı iletken devrelerdir. Tasarım esnekliği ve az sayıda cihaz kullanımı nedeniyle FPGA birçok sinyal işleme görevini gerçekleştirmek için çok uygun bir seçenektir. FPGA'ler herhangi bir uygulama için kolayca programlanabilir, aynı FPGA içeriği değiştirilip tekrar programlanarak bir başka uygulama için de kullanılabilir. Bir FPGA'in mantık hücrelerinden oluşan iç mimarisi Şekil 4.13'de verilmiştir. Verilog, FPGA için en çok kullanılan tasarım dilidir. FPGA'deki programlanabilir mantık bileşenlerine, mantıksal bloklar denilmektedir. Karmaşık ve kombinasyonel işlevler bu mantık bloklarında gerçekleştirilmektedir. Basit flip floplar veya bütünüyle bellek blokları olabilecek bellek öğeleri, bu mantıksal bloklara dahildir.



Şekil 4.13 FPGA'in iç mimarisi

FPGA üzerinde gerçeklenecek bir mimari tasarımda güç verimliliği için şu noktalara dikkat edilmelidir:

- Tasarıma göre yapılandırılabilir mantık bloklarının sayısı,
- Sabit işlevli mantık bloklarının sayısı,
- Bellek kaynaklarının boyutu,

Yapılandırılabilir mantık blokları, kaydediciler, veri seçiciler ve arama tablolardan (LUT) oluşmaktadır. Kaydediciler, her biri tek bir biti temsil eden bir dizi flip flop devresidir ve bir sonraki saat darbesine kadar üzerindeki veriyi muhafaza etmekle sorumludur. LUT'lar ise, her giriş kombinasyonu için önceden tanımlanmış bir çıkış tablosunu oluşturan ve saklayan mantıksal statik bellek (SRAM) bloklardır. Böylece tasarımda gerçekleşen mantıksal işlemlerin sonucunun hızlı bir şekilde oluşmasını sağlamaktadırlar. Bir mimari tasarımda kaydedici ve LUT kullanımı sayısındaki büyüklük bu mimarinin donanımsal karmaşıklığının yüksekliğini ve alan kullanımının genişliğini göstermektedir.

Tümleşik Yazılım Ortamı (ISE), FPGA'ları programlamak için kullanılmak üzere Xilinx firmasının geliştirdiği bir ara yüz yazılımıdır. ISE ortamı, donanım tanımlama dilleri ya da şematik çizimler sayesinde FPGA'de çalıştırılmak üzere sistemler tasarlamaya olanak sağlamaktadır. ISE ortamında Verilog, VHDL gibi donanım tanımlama dilleriyle oluşturulan tasarım, sentezleme ve gerçekleme aşamalarından geçirilerek FPGA'e yüklenmektedir. Ayrıca tasarım çalıştırılmadan hata ayıklamak ve hatanın nereden kaynaklandığını görmek üzere test yapmaya da olanak sağlamaktadır. Bunlara ek olarak ISE kullanıcıya tasarladığı sistemlerin ne kadar gecikmeye sahip olduğunu, donanım üzerinde ne kadar yer kapladığını ve yapılan tasarımın FPGA'ın hangi bölgesine yerleştirileceğine kadar detaylı bilgiler sunmaktadır.

5. BENZETİM ve SONUÇLAR

5.1. Benzetim Çalışmaları

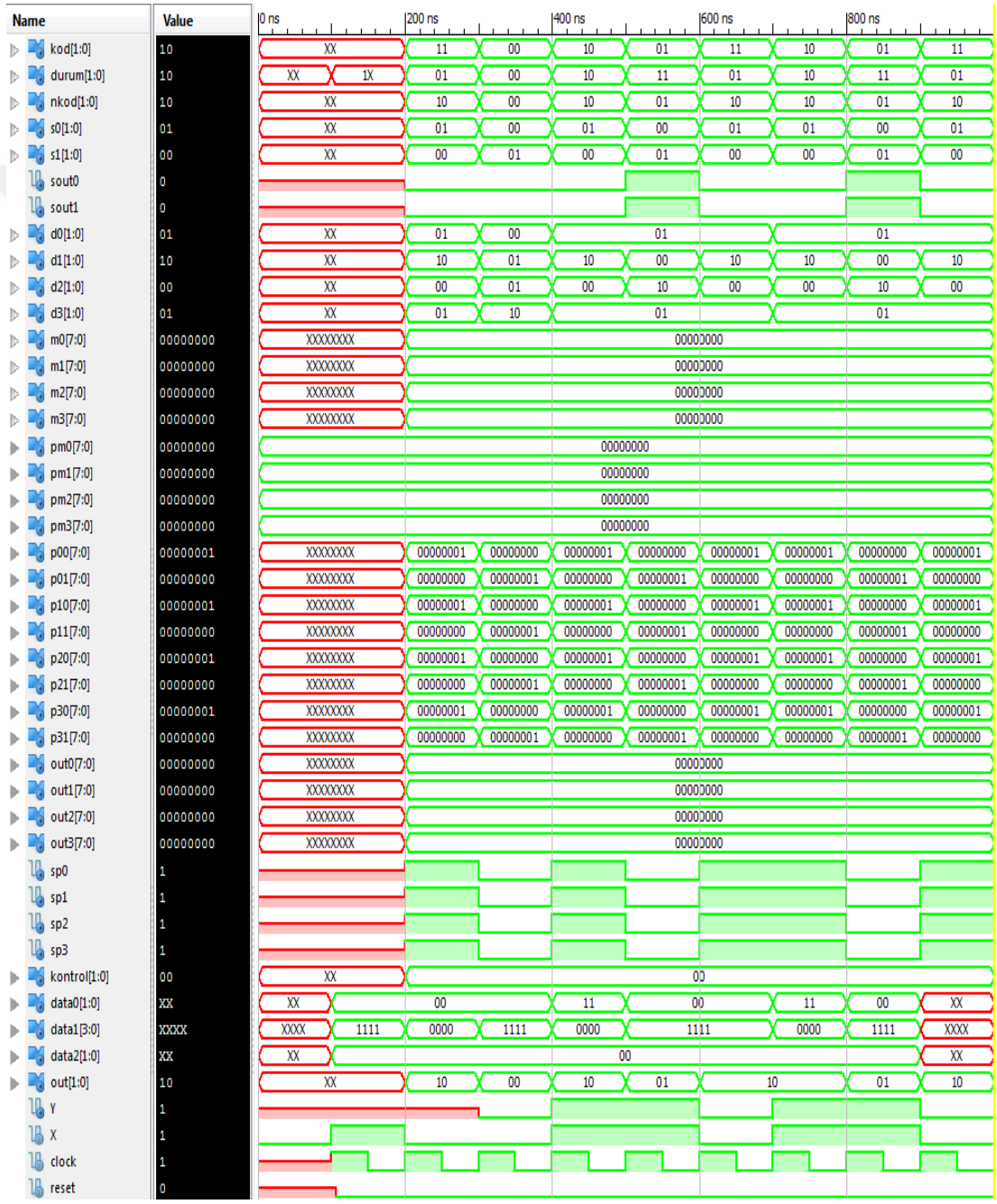
Şekil 5.1'de, Şekil 4.2'de gösterilen VD'nin birimlerinin giriş ve çıkış sinyallerinin benzetimi verilmiştir. Burada görüldüğü gibi 'X', tasarlanan katlamalı kodlayıcı girişine uygulanan mesaj serisi olmak üzere, bu kodlayıcının durumları 'durum', çıkışı ise 'kod' olarak belirtilmiştir. Bu kod kelimesi gürültü biriminden geçtikten sonra 'nkod' olarak çıkmış ve bazı bitleri değişmiştir. Buna rağmen tasarlanan VD sayesinde, kod çözücünün çıkışı Y, Şekil 5.1'de görüldüğü gibi X ile uyumludur, diğer bir deyişle doğru çözümlenmiştir. nkod temel alınarak, her saat darbesi için HD'ye göre 'd0, d1, d2, d3' BM'leri hesaplanmıştır. BM'lerin ikili kıyaslanması sonucunda küçük iki BM, 's0 ve s1' olarak atanmış ve bir önceki zaman aralığından gelen o ana kadarki toplam küçük PM'ler, 'pm0, pm1, pm2, pm3' ile toplanarak yeni PM'ler 'p00, p01, p10, p11, p20, p21, p30, p31' oluşturulmuştur. İkili kıyaslamalar sonucunda küçük ölçütler 'out0, out1, out2, out3' olarak atanmıştır. Buradan 'm0, m1, m2, m3' o andaki BM'ler bulunarak ve bu değerler kullanılarak o ana kadarki toplam yeni PM'ler 'pm0, pm1, pm2, pm3' elde edilmiştir. Bu aşamalar her saat darbesinde tekrarlanmaktadır.

Şekil 5.2'de, başlangıç anında üzerinde PM'ler bulunduran bir kod çözücünün benzetimi gösterilmektedir. Kod çözücünün, kodda büyük değişim olduğundaki davranışını ifade etmektedir. Şekil 5.1'e göre PM'lerdeki artış, kodun kod çözücüye yüksek hata oranıyla gelmesinden kaynaklanmaktadır. Ancak bu hata oranının yüksekliğine rağmen çıkış sinyali Y, yine doğru çözülmüştür.

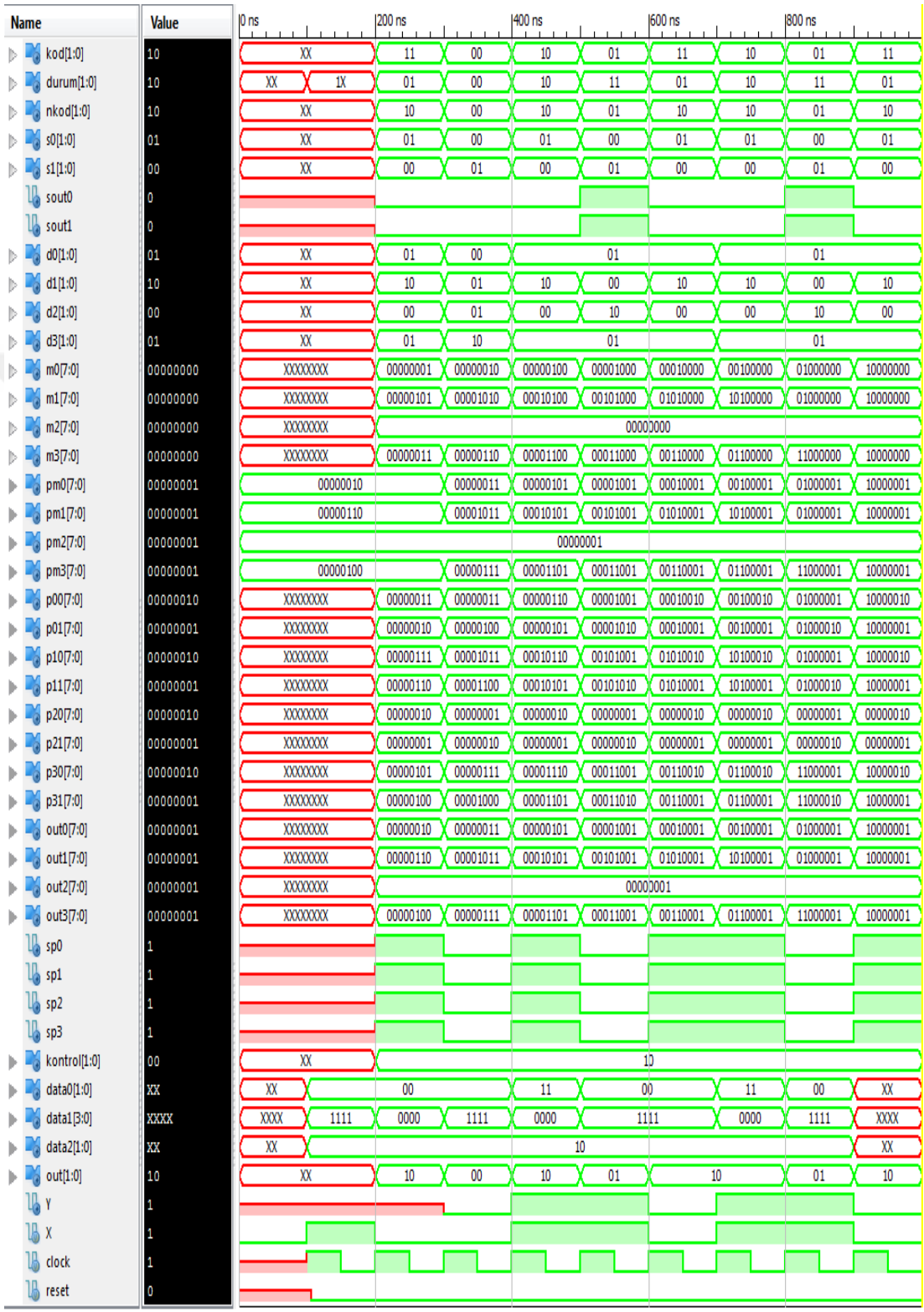
Güç etkin kod çözücü hesaplama yükünden kurtulmak için belli bir eşik değer üzerindeki PM'leri eleme temeline dayanarak tasarlanmıştır. Şekil 5.3, kod çözücünün bu yaklaşımla tasarlanan güç etkin halinin benzetimidir. Bu kod çözücü için söz konusu olan eşik değer 0 alınmıştır. Bu değerin üzerinde PM'ye sahip olan yollar karşılaştırmalara katılmamıştır. Bu sayede birçok yol elendiği için kod

çözücünün hesap yükü dolayısıyla karmaşıklığı büyük oranda azaltılmış buna rağmen yine çıkış sinyali Y, giriş sinyali X'in aynısıdır, yani doğru çözülmüştür.

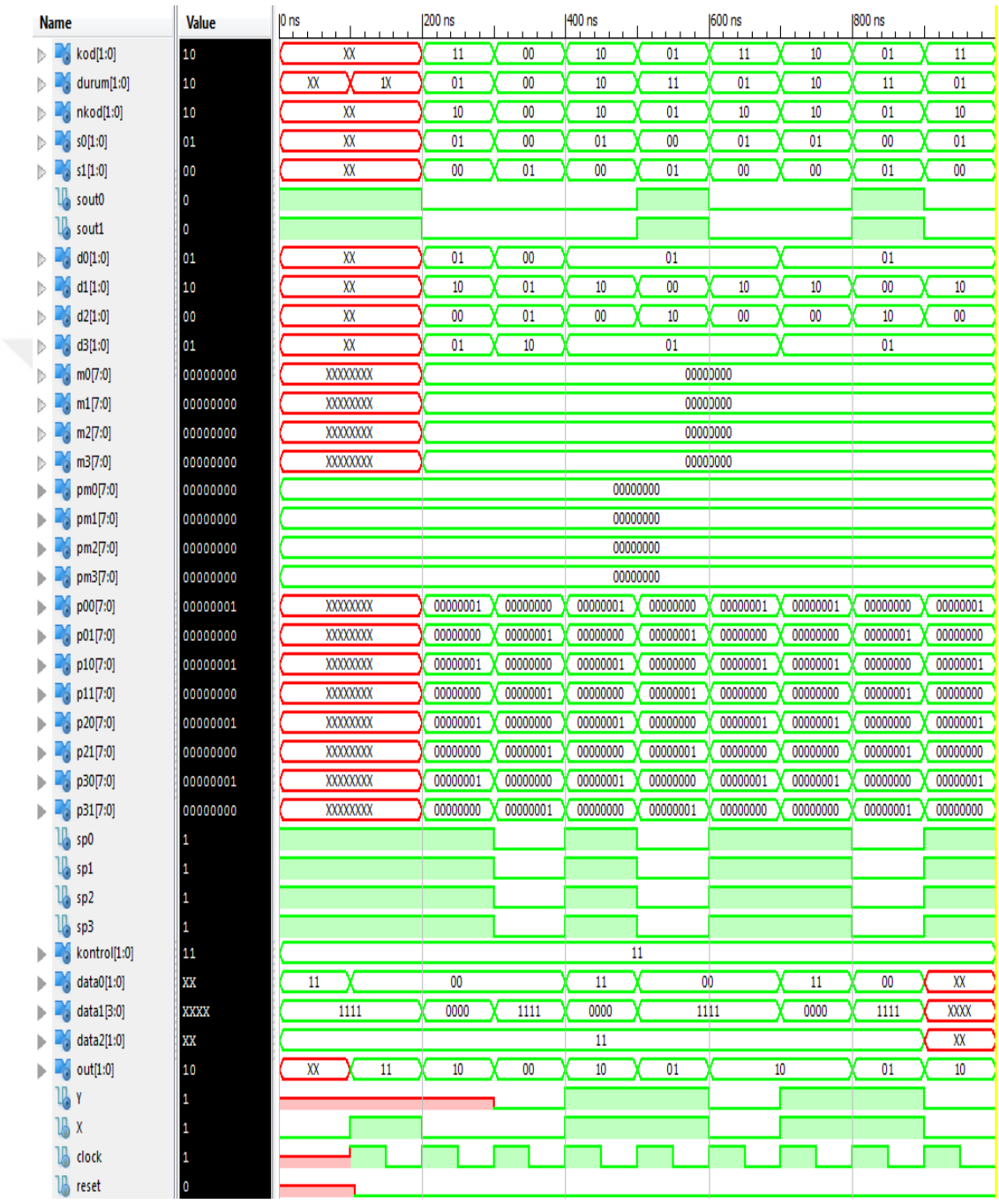
Tasarlanan VD ve güç etkin VD'nin, Xilinx ISE Design Suit gerçeklemesi sonucunda oluşun RTL şemaları Şekil 5.4 ve Şekil 5.5'de verilmiştir.



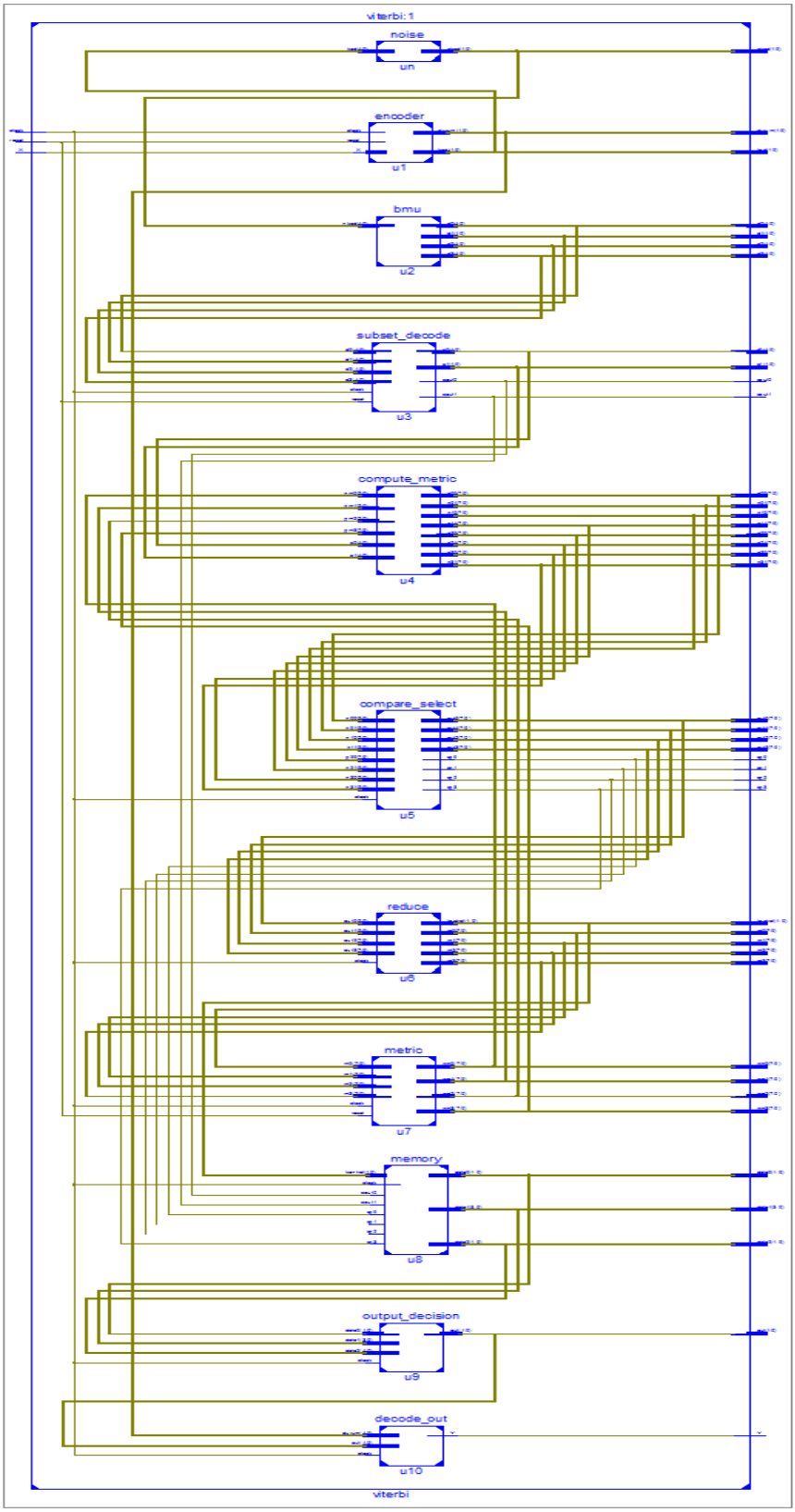
Şekil 5.1 Tasarlanan VD'nin verilen bir X giriş serisine göre Xilinx ISE Design Suit benzetimi



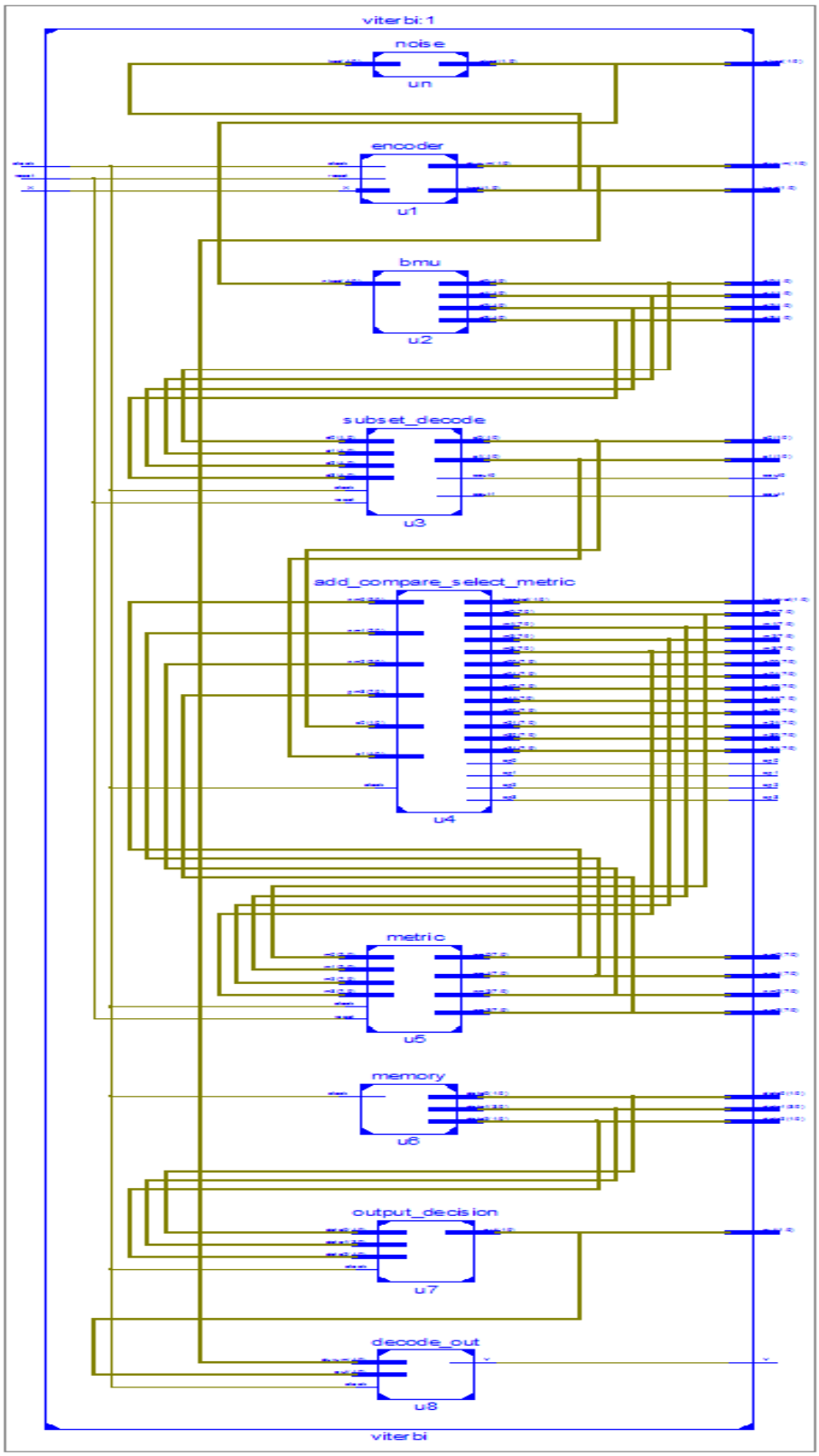
Şekil 5.2 VD'nin geçmişten gelen yol değerleri ile tasarlanmış halinin benzetimi



Şekil 5.3 Güç etkin VD'nin eşik= 0 değeriyle benzetimi



Şekil 5.4 Şekil 4.2'deki VD'nin Xilinx ISE Design Suit gerçeklemesi sonucunda oluşun RTL şeması



Şekil 5.5 Şekil 4.11'de tasarımı verilen güç etkin VD'nin Xilinx ISE Design Suit gerçeklemesi sonucunda oluşun RTL şeması

5.2. Sonuçlar

FPGA ortamında gerçeklenen bir tasarımda, kullanılan kaydedici ve arama tablosu sayısı, tasarımın donanımsal karmaşıklığını ve alan kullanımını açık şekilde ortaya koymaktadır. Şekil 5.6'da, bu çalışma kapsamında ilk tasarlanan kod çözücünün donanım kullanımı özeti, Şekil 5.7'de ise, güç etkin tasarlanan kod çözücünün donanım kullanımı özeti gösterilmektedir. Bir tasarımda ne kadar fazla kaydedici ve arama tablosu varsa, o tasarımda o kadar fazla mantıksal işlem gerçekleşiyor ve mantıksal blok kullanımı fazladır demektir. Eğer güç verimli bir mimari hedefleniyorsa tasarımın donanım karmaşası dolayısıyla kaydedeci ve arama tablosu sayısı azaltılmalıdır. Bu amaçla Şekil 4.2'de tasarımı gösterilen VD'ye bir eşik değer tanımlayarak güç etkin VD tasarımı yapılmıştır. PM'ler birbirleriyle kıyaslanmadan bu değerle kıyaslanmıştır ve eşik değer üzerindeki yollar elenmiştir. Böylece PM karşılaştırma işlemi sayısı ve bu karşılaştırmaların sonuçlarının bellekte tuttuğu yer azaltıldığı için, güç verimli tasarımda, Tablo 5.1'de görüldüğü gibi kaydedici sayısında %80, arama tablosu sayısında ise %57 oranında azalma sağlanmıştır. Bu sonuç ikinci tasarımda, ilk tasarıma göre daha az mantıksal blok kullanıldığı dolayısıyla mantıksal blok kaynaklı güç tüketiminin daha az olduğu anlamına gelmektedir.

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	30	18,224	1%	
Number used as Flip Flops	6			
Number used as Latches	24			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	253	9,112	2%	
Number used as logic	252	9,112	2%	
Number using O6 output only	199			
Number using O5 output only	2			
Number using O5 and O6	51			
Number used as ROM	0			
Number used as Memory	0	2,176	0%	
Number used exclusively as route-thrus	1			
Number with same-slice register load	0			
Number with same-slice carry load	1			
Number with other load	0			
Number of occupied Slices	93	2,278	4%	
Number of MUXCYs used	68	4,556	1%	
Number of LUT Flip Flop pairs used	256			
Number with an unused Flip Flop	226	256	88%	
Number with an unused LUT	3	256	1%	
Number of fully used LUT-FF pairs	27	256	10%	
Number of unique control sets	5			
Number of slice register sites lost to control set restrictions	10	18,224	1%	

Şekil 5.6 Şekil 4.2'de tasarımı verilen VD'nin donanım kullanımını gösteren Xilinx ISE Design Suit ekran görüntüsü

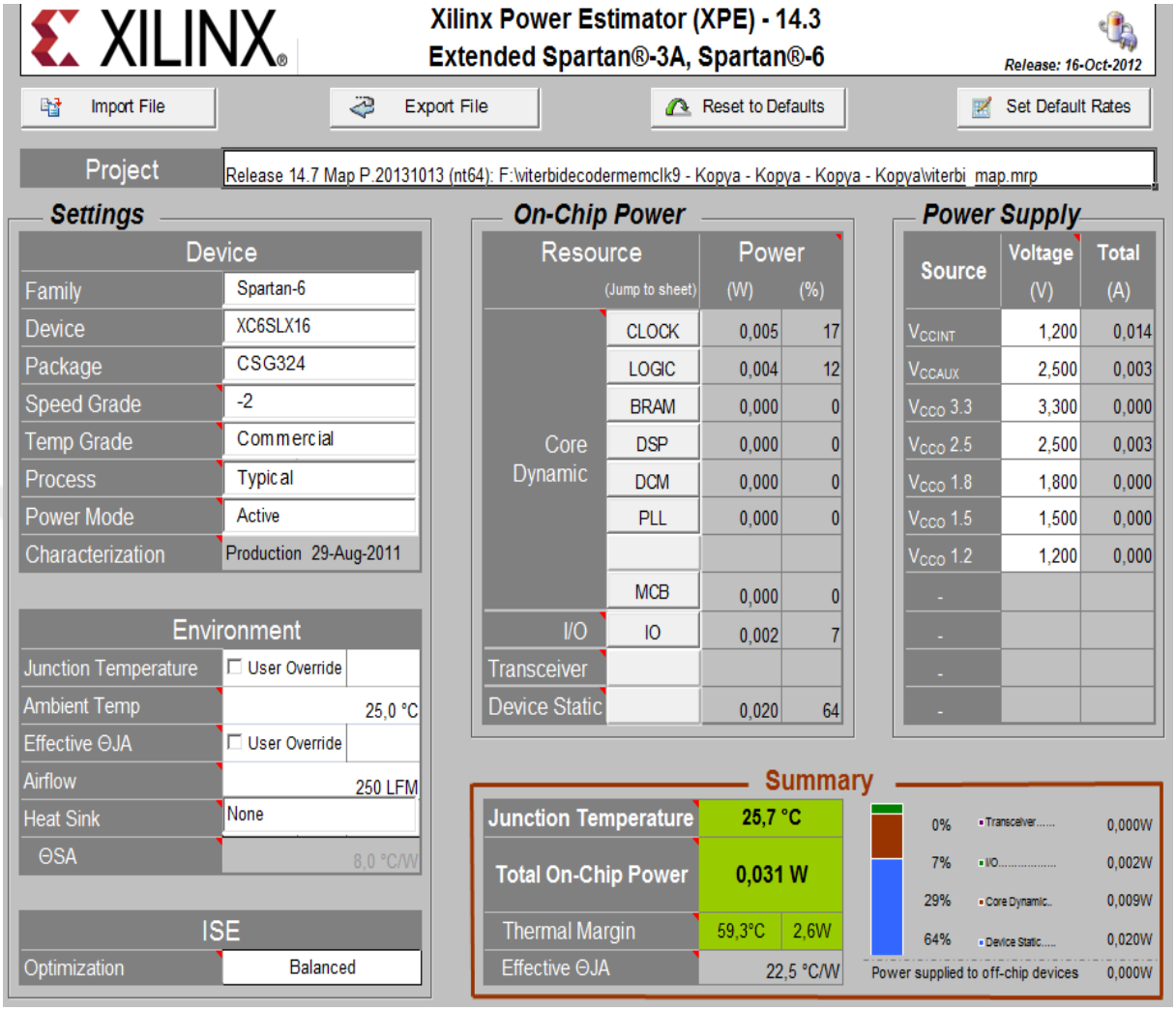
Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	6	18,224	1%	
Number used as Flip Flops	6			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	110	9,112	1%	
Number used as logic	108	9,112	1%	
Number using O6 output only	56			
Number using O5 output only	2			
Number using O5 and O6	50			
Number used as ROM	0			
Number used as Memory	0	2,176	0%	
Number used exclusively as route-thrus	2			
Number with same-slice register load	1			
Number with same-slice carry load	1			
Number with other load	0			
Number of occupied Slices	36	2,278	1%	
Number of MUXCYs used	36	4,556	1%	
Number of LUT Flip Flop pairs used	110			
Number with an unused Flip Flop	105	110	95%	
Number with an unused LUT	0	110	0%	
Number of fully used LUT-FF pairs	5	110	4%	
Number of unique control sets	2			
Number of slice register sites lost to control set restrictions	10	18,224	1%	

Şekil 5.7 Şekil 4.11'de tasarımı verilen güç etkin tasarlanan VD'nin donanım kullanımını gösteren Xilinx ISE Design Suit ekran görüntüsü

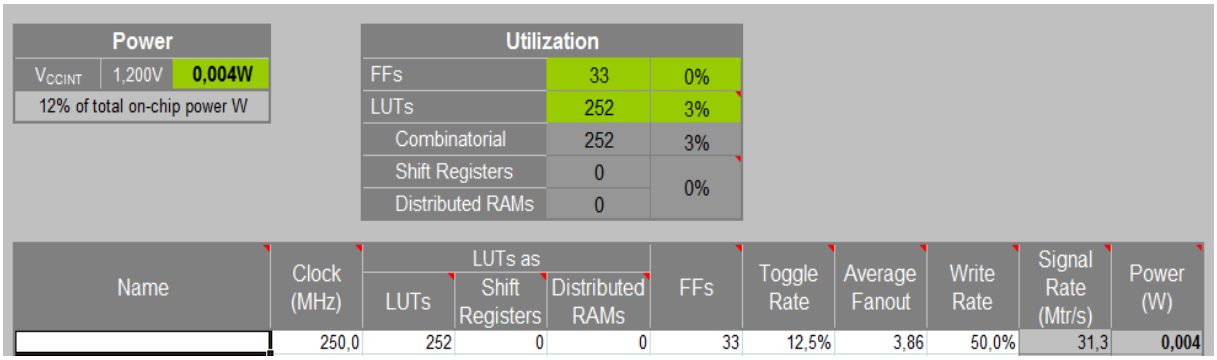
Mimari Tasarım	Viterbi Kod Çözücü	Güç Etkin Viterbi Kod Çözücü	Donanımsal Küçülme
Kaydedici Sayısı	30	6	%80
Arama Tablosu Sayısı	253	110	%57
GCLK Sayısı	1	1	-
Giriş Çıkış Portları Sayısı	4	4	-

Tablo 5.1 Viterbi Kod Çözücü ve Güç Etki Viterbi Kod Çözücü donanım kullanımı tablosu

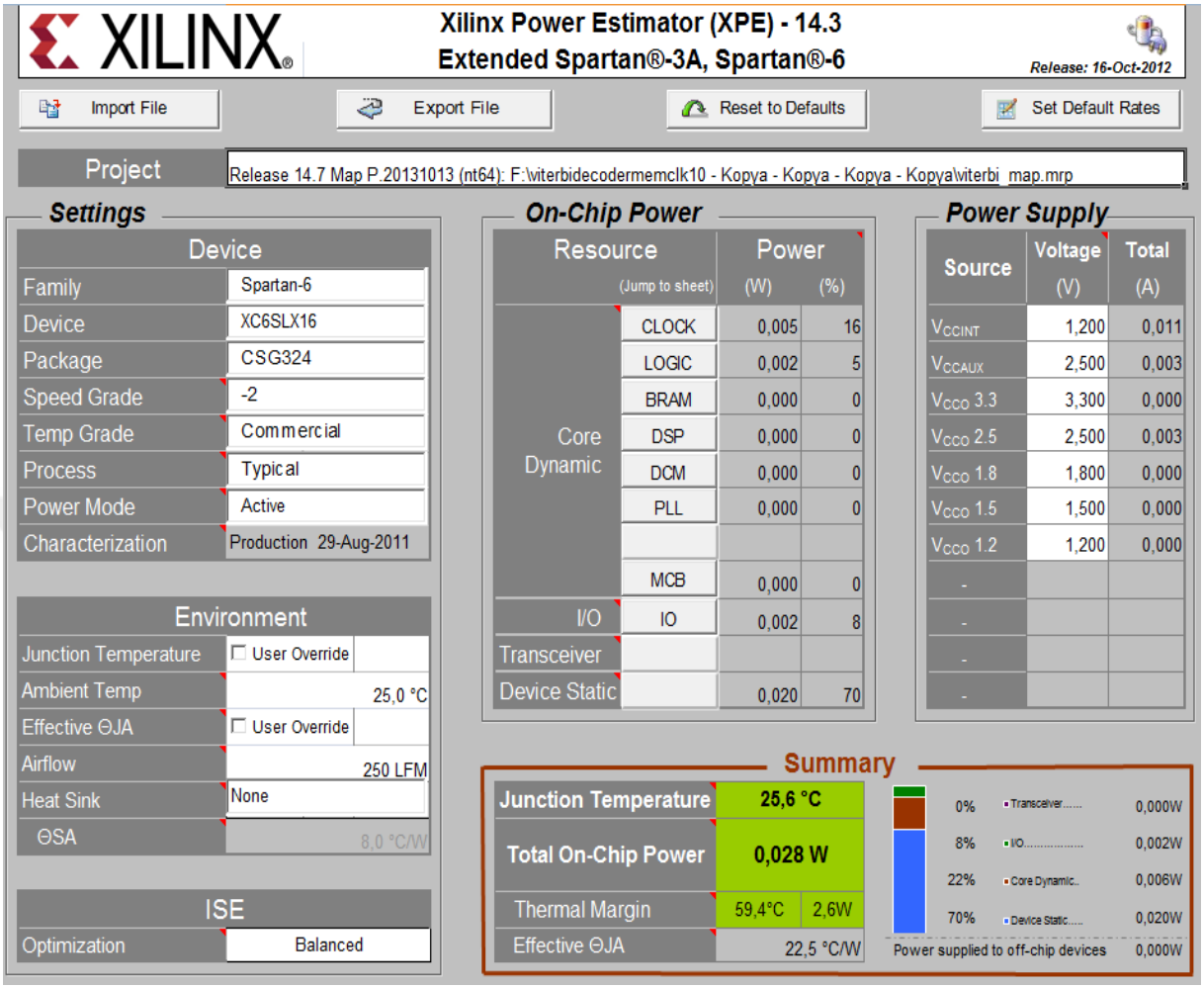
Şekil 5.8'de ve 5.10'da görüldüğü üzere tasarlanan iki kod çözücünün 'map' dosyası kullanılarak Xilinx Power Estimator ile güç analizi yapılmıştır. Şekil 4.2'de gösterilen tasarımın FPGA'de gerçeklenmesi sonucunda toplam güç tüketimi 0.031 W iken Şekil 4.11'de gösterilen güç etkin tasarımın toplam güç tüketimi 0.028 W olmuştur. Her iki tasarım için bu değerlerin 0.020 W'ı, kullanılan FPGA için sabit harcanan güç değeri; 0.005 W'ı, saat tetiklemesinden dolayı kaybedilen güç, 0.002 W'ı; giriş çıkış portları üzerinde kaybedilen güç değerleridir. Güç tüketimi kaynaklarında farklılık sadece mantık bloklarının güç tüketiminden gelmektedir. Tasarımlar arasında sadece mantık bloklarının kullanımı üzerine değişiklik yapıldığı için bu zaten beklenen bir sonuçtur. Şekil 5.9'da görüldüğü üzere ilk tasarlanan VD'nin donanım kullanımından kaynaklanan güç tüketimi 0.004 W iken, Şekil 5.11'de görüldüğü gibi güç etkin tasarlanan VD'nin donanım kullanımından kaynaklanan güç tüketimi 0.002 W olmuştur. Dolayısıyla bu çalışmada güç etkin tasarımda mantık bloklarının güç kullanımında toplam olarak %50 oranında azalma sağlanmıştır.



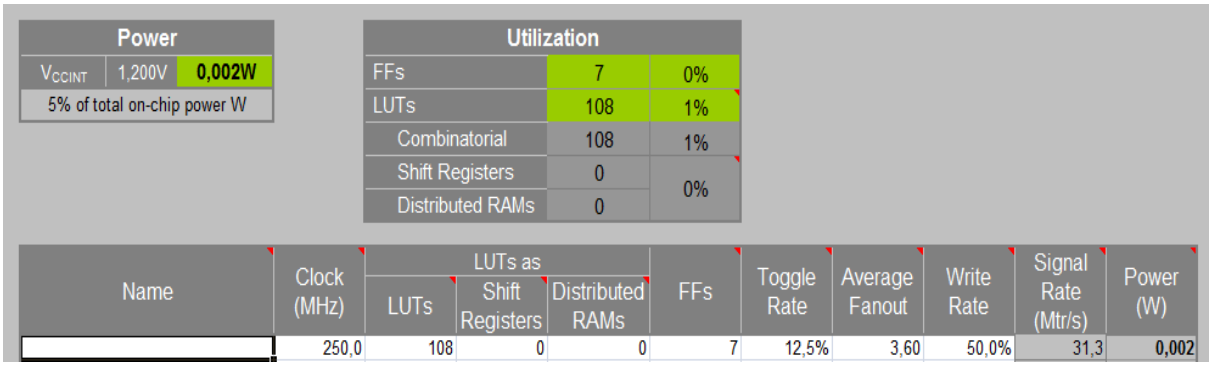
Şekil 5.8 Şekil 4.2'de tasarımı verilen VD için Xilinx Power Estimator ile yapılan güç kullanımı analizinin ekran görüntüsü



Şekil 5.9 Şekil 4.2'de tasarımı verilen VD için Xilinx Power Estimator ile yapılan mantık bloklarının güç kullanımı analizinin ekran görüntüsü



Şekil 5.10 Şekil 4.11'de tasarımı verilen güç etkin VD için Xilinx Power Estimator ile yapılan güç kullanımı analizinin ekran görüntüsü



Şekil 5.11 Şekil 4.11'de tasarımı verilen güç etkin VD için Xilinx Power Estimator ile yapılan mantık bloklarının güç kullanımı analizinin ekran görüntüsü

5.3. Tartışma ve Öneriler

Bu tez çalışmasında tasarlanan güç etkin VD için, klasik tasarlanan VD'ye göre kaydedici sayısında %80, arama tablosu sayısında ise %57 oranında azalma sağlanmıştır. Böylece donanımsal güç kullanımında %50 oranında verim elde edilmiştir. Dolayısıyla tez çalışması amacına ulaşmıştır. Gelecekte yapılacak çalışmalar güç verimli kod çözücünün, kod çözmedeki hızını arttırmaya yönelik olacaktır. Kodlayıcının kısıtlama uzunluğu arttırılarak, sistemde kodu çözmek için donanımsal olarak daha karmaşık bir VD ihtiyacı yaratma ve bu kod çözücüye tanımlanan bir eşik değer ile yine bu karmaşıklık azaltılacaktır. Ayrıca VA, Gizli Markov Modeli'ne göre incelenerek güç verimli hale getirilmeye çalışılacaktır.

6. KAYNAKLAR

[1] Viterbi, A.J., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", IEEE Trans. Information Theory, vol. IT-13, pp. 260-269, 1967.

[2] Forney, G.D., "The Viterbi Algorithm", Proc. IEEE, vol. 61, pp. 268-278, March 1973.

[3] Black, P. J., Meng, T.H., "A 140-Mb/s, 32-state radix-4 Viterbi Decoder", IEEE J. Solid-State Circuits, vol. 27, pp. 1877–1885, Dec. 1992.

[4] Page, K., Chau, P.M., "Improved Architectures for the Add–Compare–Select Operation in Long Constraint Length Viterbi Decoding", IEEE J. Solid-State Circuits, Vol. 33, pp. 151–155, no. 1, January 1998

[5] Kang, I., Willson, A., "Low-Power Viterbi Decoder for CDMA Mobile Terminals", IEEE J. of Solid-State Circ., vol.33, no. 3, pp. 473-482, Mar. 1998.

[6] Ranpara, S., "On a Viterbi Decoder Design for Low Power Dissipation", Faculty of the Virginia Polytechnic Institute and State University, MSc. Thesis, April, 1999

[7] Lee, I., Sonntag, J.L., "A New Architecture for the Fast Viterbi Algorithm", IEEE Global Telecommunications Conference, San Francisco, vol. 3, pp. 1664 –1668, Nov. 2000.

[8] Chadha, K., Cavallaro, J.R., "A Reconfigurable Viterbi Decoder Architecture", Conference Record of 35th Asilomar Conference on Signals, Systems and Computers, vol.1, pp. 66-71, 2001.

[9] Benaissa, M., Zhu, Y., "A Novel High-Speed Configurable Viterbi Decoder for Broadband Access", EURASIP Journal on Applied Signal Processing, pp.1317–1327, 2003

[10] Zhu, Y., Benaissa, M., "Reconfigurable Viterbi Decoding Using a New ACS pipelining technique," Proceedings of the 2003 IEEE international conference on Application-Specific Systems, Architectures, and Processors (ASAP2003), The Hague, The Netherlands, pp. 360-368., 24-26 June 2003

[11] Bruels, N., Sicheneder, E., Loew, M., Schackow, A., Gliese, J., Sauer, C., "A 2.8 Gb/s, 32-State, Radix-4 Viterbi Decoder Add-Compare-Select Unit", Symposium on VLSI Circuits Digest of Technical Papers, pp.170-173, June 2004

[12] Gang, Y., Arslan, T., Erdogan, A.T., "An Efficient Pre-Traceback Approach for Viterbi Decoding in Wireless Communication", ISCAS 2005, IEEE International Symposium on Circuits and Systems, Vol. 6, pp. 5441- 5444, May. 2005

[13] Tessier, R., Swaminathan, S., Ramaswamy, R., Goeckel, D., Burleson, W., "A Reconfigurable, Power-Efficient Adaptive Viterbi Decoder", IEEE Transactions On

Very Large Scale Integration (VLSI) Systems, vol. 13, no. 4, pp. 484 – 488, April 2005

[14] Dinhand, A., Xiao, H., "A Hardware-Efficient Technique to Implement a Trellis Code Modulation Decoder", IEEE Transactions On Very Large Scale Integration (VLSI) Systems, vol. 13, no. 6, pp. 745-750, June 2005

[15] Moreira, J.K., Farrel, P.G., "Essential of Error Control Coding", John Wiley & Sons Ltd, pp. 1-77, pp. 157-205,2006.

[16] Tang, Y.C., Hu, D.C., Wei, W., Lin, W.C., Lin, H., "A Memory-Efficient Architecture for Low Latency Viterbi Decoders", 2009 International Symposium on VLSI Design, Automation and Test, 28-30 April 2009

[17] Cholan, K., "Design and Implementation of Low Power High Speed Viterbi Decoder", International Conference on Communication Technology and System Design, vol.30, pp.61-68, 2011

[18] Yılmaz, İ.C., "Design and Simulation of Soft Decision Viterbi Decoder", Department of Electrical and Electronics Engineering, Çukurova University Institute of Natural and Aplied Sciences, MSc. Thesis, 2011

[19] Latha, S.L., Kumari, D.L., "Low-Power Adaptive Viterbi Decoder for TCM Using T-Algorithm", International Journal of Scientific and Research Publications, vol. 3, no. 8, August 2013

[20] Bhatt, N., Shah, M., Asodoriya, B., "FPGA Implementation Of Power Efficient Low Latency Viterbi Decoder", Indianternational Journal of Engineering Research & Technology (IJERT) vol. 2, no 5, May 2013.

[21] Thakre, R.I. "Design of T-Algorithm Based High-Speed Low-Power Viterbi Decoder for TCM Decoders", International Journal of Innovative Research in Electronics and Communications (IJIREC), vol 1, no 1, pp 33-38, April 2014

[22] Kumar, V.G., Sudhir, A.C., "Implementation of Viterbi Decoder using T-algorithm for TCM Decoders" International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, vol. 3, no 5, May 2015

ÖZGEÇMİŞ

Burcu Özbay 1990 yılında Besni'de doğmuştur. Liseyi Özel Marmara Fen Lisesi'nde okuduktan sonra 2007 yılında başladığı Hacettepe Üniversitesi Fizik Mühendisliği Bölümü'nden 2012 yılında mezun olmuştur. 2013 yılından itibaren Maltepe Üniversitesi Mühendislik ve Doğa Bilimleri Fakültesi Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak çalışmaktadır.