T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ELEKTRİK- ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

VITERBI Algoritmasının FPGA ile Gerçeklenmesi

Ahmed Melih ULUSOY
031511564

Nuh NAR
031511556

MÜHENDİSLİK TASARIMI I

BURSA 2021
T.C.

BURSA ULUDAĞ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

VITERBI Algoritmasının FPGA ile Gerçeklenmesi

Ahmed Melih ULUSOY
031511564

Nuh NAR
031511556

Projenin Danışmanı : Prof. Dr. Tuncay ERTAŞ

BURSA 2021

Uludağ Üniversitesi Mühendislik Fakültesi tez yazım kurallarına uygun olarak hazırladığım bu Bitirme Projesi çalışmasında;

- Tez içindeki bütün bilgi ve belgeleri, akademik kurallar çerçevesinde elde ettiğimi
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları, bilimsel ahlak kurallarına uygun olarak sunduğumu
- Başkalarının eserlerinden yararlanılması durumunda, ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Bu tezin herhangi bir bölümünü üniversitemde veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

04.01.2021

Nuh Nar	Ahmed Melih Ulusoy
031511556	031511564

Danışmanlığımda hazırlanan Bitirme Projesi çalışması, tarafımdan kontrol edilmiştir.

04.01.2021

Prof. Dr. Tuncay Ertaş

ÖZET

Bu projede QPSK modülasyon ile elde edilen verinin Viterbi dekoderi ile FPGA ortamında demodüle edilmiş sinyalin Viterbi algoritması ile işlenip modüle edilmiş sinyale en yakın çıkış elde edilmiştir. Bu çalışmada Matlab ortamında rasgele bit üretilip, üretilen bu bitler daha sonra QPSK modülasyonuna tabi tutulup yeni bir veri elde ediliyor. Elde edilen bu veri gürültülü bir kanala aktarılıp veri üzerinde olması muhtemel değişiklikler sanal ortamda oluşturuluyor. Oluşturulmuş bu yeni gürültülü veri QPSK demodülatörüne girdi olarak veriliyor ve demodüle edilen verinin sonuçları kaydediliyor. Bütün bu işlemler sonucunda kanal kodlaması yapılmadan aktarılan veride yaşana kayıp Matlab ortamında girdi ve çıktıların karşılaştırılması ile SNR değeri belirleniyor. Bu işlemden sonra Viterbi dekoderi ile kanal kodlaması yapılmış FPGA modülü devreye sokularak veri tekrar işleme tabi tutuluyor. Bu işlem neticesinde olasılık tabanlı bu algoritmadan geçirilen veri ile oluşan çıkış tekrar kaydedilip sonuçlar kayıt ediliyor. Bu sonuçta giriş verisiyle karşılaştırılıp Viterbi dekoderinin hata oranı belirleniyor. Bu çalışma neticesinde elde edilen iki veri ile kanal kodlaması ile ortam gürültüsü elenip geçirilince hata oranın inanılmaz oranda azaldığı gözlemlenmiştir. Çalışmada FPGA tabanlı yapılması ile işlemin hızlı gerçekleştirilip az güç tüketimi sağlanmıştır. Günümüzde artan veri transferi yoğunluğu ihtiyacında kanal kodlamasının önemini anlamış bulunmaktayız.

ABSTRACT

In this project, the data obtained by QPSK modulation and the signal demodulated in FPGA environment with Viterbi decoder was processed with Viterbi algorithm and the closest output to the modulated signal was obtained. In this study, random bits are generated in Matlab environment and these bits are then subjected to QPSK modulation and a new data is obtained. This data is transferred to a noisy channel and possible changes to the data are created in a virtual environment. This newly created noisy data is input to the QPSK demodulator and the results of the demodulated data are recorded. As a result of all these processes, the loss of the data transferred without channel coding is determined by comparing the inputs and outputs in the Matlab environment, and the SNR value is determined. After this process, the channel coded FPGA module is activated with the Viterbi decoder and the data is processed again. As a result of this process, the output generated by the data passed through this probability-based algorithm is recorded again and the results are recorded. This is compared with the input data and the error rate of the Viterbi decoder is determined. With the two data obtained as a result of this study, it was observed that the error rate was reduced incredibly when the ambient noise was eliminated by channel coding. By making it FPGA-based in the study, the process was performed quickly and less power consumption was achieved. We understand the importance of channel coding in today's increasing data transfer density need.

İÇİNDEKİLER

Sayfa No

ŞEKİLLER DİZİNİ

Sayfa No

1.GİRİŞ

Haberleşme sistemlerinde yaygın olarak kullanılan QPSK (Quadrate Phase Shift Keying: Dördün Faz Kaydırmalı Anahtarlama) modülasyonu ve demodülasyonu, bant genişliği ve gürültü oranı düşünüldüğünde yüksek fayda sağlamakla beraber maliyet olarak 8, 16, 64, …, vb. Aynı tip modülasyonlara göre daha avantajlı olmaktadır. Haberleşme sistemlerinde en önemli konulardan biri güç tüketimidir. QPSK modülasyonunda bant genişliği, gürültü oranı düşük olması yanında bu iki orana istinaden güç tüketimi de azdır. Güç tüketiminin yanı sıra ikilik faz kaydırmalı sisteme göre aynı bant genişliğinde bir yerine iki sinyal biden iletebildiği için hızlıdır. 8, 16, 64 … gibi sistemlerin hızlı olması avantajlı gibi görünse de gürültüden etkilenme olasılıkları da fazladır. Bütün bu bilgiler göz önüne alarak QPSK modülasyonu tercih edilmiştir.

Avantajları ve dezavantajları düşünüldüğünde tercih ettiğimiz QPSK modülasyonu ve demodülasyonu her ne kadar gürültüden diğer modülasyonlar kadar etkilenmese de veri kaybı her zaman yaşanmaktadır. Veri kaybının önüne geçmek için en yaygın kullanılan yöntem ise kanal kodlamasıdır.

Viterbi dekoderi modüle edilmiş sinyalin gürültülü bir ortamdan geçtikten sonra demodüle edilirken gelen verinin algoritmaya uygun olarak işlenip ortam gürültüsünden arındırılmasıyla, veri kaybı minimuma indirilip güvenli bir şekilde veriyi almamızı sağlar.

Uzay iletişimi, Kablosuz haberleşme (Wireless), Mobil iletişim sistemleri (GSM) gibi konularda artan hız, kapasite, … vb. gibi ihtiyaçlar ile oluşan aşırı yoğunlukta veri kaybını önlemek için kanal kodlaması yaygın olarak kullanılmaktadır. Bütün bu yoğunluk ve hız ihtiyacında kullanılan işlemcilerin kapasitesi ihtiyaçları karşılamaktan gün geçtikçe uzaklaşmaktadır ve sorunlar çıkarmaktadır. Yavaş yavaş oluşan bu yoğunluk için FPGA'lar kullanılmaktadır. FPGA'leri tercih etmekteki sebep ise paralel işlem yapabilme kapasitesi ve hız ihtiyaçlarına göre optimize edilebilmesi.

Çalışmamızda FPGA ortamında VHDL dili ile Viterbi dekoderi yapıp Matlab ortamında oluşan farklar gözlemlenip ortam gürültüsünden ne kadar etkilendiğini gözlemleyip, SNR değeri belirlenecek. Bütün bu çalışmalar sonucunda FPGA ile Viterbi dekoderinin avantaj ve dezavantajları gözlemlenecektir.

2.KAYNAK ARAŞTIRMASI

Gupta S., Mehra R. "Reconfigurable Efficient Design of Viterbi Decoder for Wireless Communications Systems" International Journal of Advanced Computer Science and Applications, Vol 2, No 7, 2011 adlı makalede, Viterbi decoder'in evrişim kodlarını çözmek için kablosuz iletişim sistemlerinin ileri düzeltme kodları olarak araştırıldığına değinilmiştir. Bu yazıda düşük güç ve yüksek hızlı viterbi kod çözücü tasarımı üzerine çalışılmıştır. Tasarım Matlab kodlarıyla tasarlanmış Xilinx Synthesis Tool kullanarak sentezlenmiştir. Sonuç olarak kaynak cihazda daha az kaynak tüketerek 62.52 MHz önerilen frekansta cihaz çalışmıştır.

Shaker S. W., Elramly S. H., Shehata K. A. "FPGA implementation of a reconfigurable Viterbi decoder for WiMAX receiver" Proceedings of the International Conference on Microelectronics, ICM (2009) makalesinde, kısıtlama uzunluğu olan K=7, r=1/2 evrişimli kodlayıcı ve Viterbi decoder'in temel blok yapıları gösterilmiştir. Kod çözücü kısmında ACS (Add- Compare-Select) birim yapısı gösterilmiş ve çalışma prensibi açıklanmıştır. Sonuç olarak çip kaynağının %7'sinden daha az kaynak kullanılmış, 47.4 MHz maksimum saat frekansında çalışmıştır.

Kalita S., Gogoi P., Sarma K. K. "Convolutional coding using booth algorithm for application in wireless communication" Department of Electronics and Communication Technology, Gauhati University, Guwahati-781014 bu çalışmada, AWGN kanalında Viterbi kod çözme ile geleneksel evrişimli kodlamanın performansı incelenmiş ve sonuçları incelenmiştir.Sonuç olarak, geleneksel Viterbi kodlayıcı ile karşılaştırıldığında %10 ile %15 arasında bir zaman iyiliştirilmesi gözlenmiş, bit gürültü oranına 1 dB'den 8 dB'e kadar değiştirerek Matlab sonuçları gözlemiş ve grafiğe dökmüştür.

Proakis G. J., Salehi M., 2020, "Sayısal Haberleşme" Palme Yayınevi, ANKARA kitabı ile evrişimlik kodlar ve viterbi decoder ile ilgili temel bilgilere bu kitap baz alınmıştır. Yüksek lisansa yönelik olan bu kitap projemiz ile ilgili sürekli göz önünde bulundurduğumuz bir başucu kitabı görevi görmüştür.

Seke E. 2017, "VHDL Örnekleriyle Sayısal Haberleşmeye Giriş: Kavram-Teori-Uygulama-Sonuç" Seçkin Yayıncılık, ANKARA kitabı ile VHDL'in haberleşme ile olan ilişkisi çerçevesinden proje geliştirilmiş ve bu alan ile ilgili olarak diğer FPGA

kitaplarından ayrışmıştır. Projemiz özellikle bu alana yönelik olduğunda bir diğer faydalı kaynak olmuştur.

Özbay, B. (2017). Viterbi kod çözücünün güç etkin mimari tasarımı ve FPGA gerçeklemesi (Master's thesis, Maltepe Üniversitesi, Fen Bilimleri Enstitüsü). Bu tez çalışması yaptığımız projeye en benzeri olduğu için çalışmamızda temel kaynak oldu.

Bütün bu kaynakların yanında Udemy de yer alan**,** Can Dost Yavuz tarafından hazırlanan **“**Dijital donanım tasarımcısı olma kursu” ile FPGA donanımının VHDL ile kodlaması üzerine kendi eksiklerimiz giderip projeye yönelik farklı yaklaşımlarla yaklaşmamızı sağladı.

3.MATERYAL VE YÖNTEMLER

3.1.HABERLEŞME

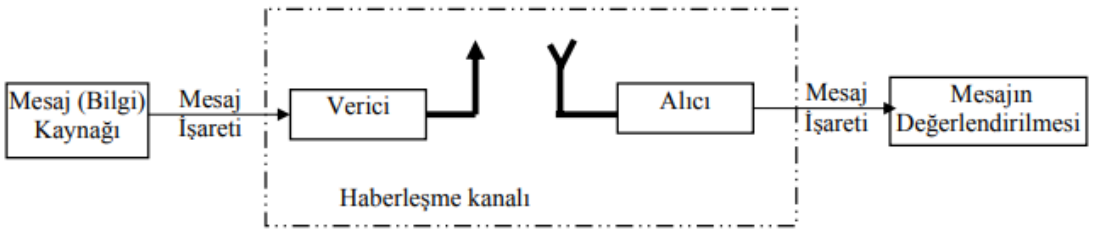
Ses, görüntü, video, veri, telemetrik gibi bilgilerin bir noktadan diğer bir noktaya yüksek verimde, yüksek kalitede ve güvenli bir biçimde iletilmesidir. Haberleşme sistemi; gönderilecek bilginin üretildiği kaynak, gönderici, iletişim ortamı ve alıcı devrelerinden oluşur.

Haberleşmede Kaliteyi Belirleyen Parametreler

- İletişim ortamının kapasitesi.
- Ses iletiminde; anlama, tanıma, hissetme ve gecikme (kulak algılama süresidir 800ms)
- Görüntüde; Resim orijinaline sadık kalma
- Video konferans; Gözün algılamadaki gerekli bir saniyedeki resim çerçeve sayısı
- Veri iletiminde; Bit hata oranı

Haberleşme Bileşenleri

- Veri kaynağı (Mesaj kaynağı)
- Verici-İletici
- Transmisyon Ortamı (kanal)
- Alıcı
- Varış Yeri (mesajın değerlendirilmesi)

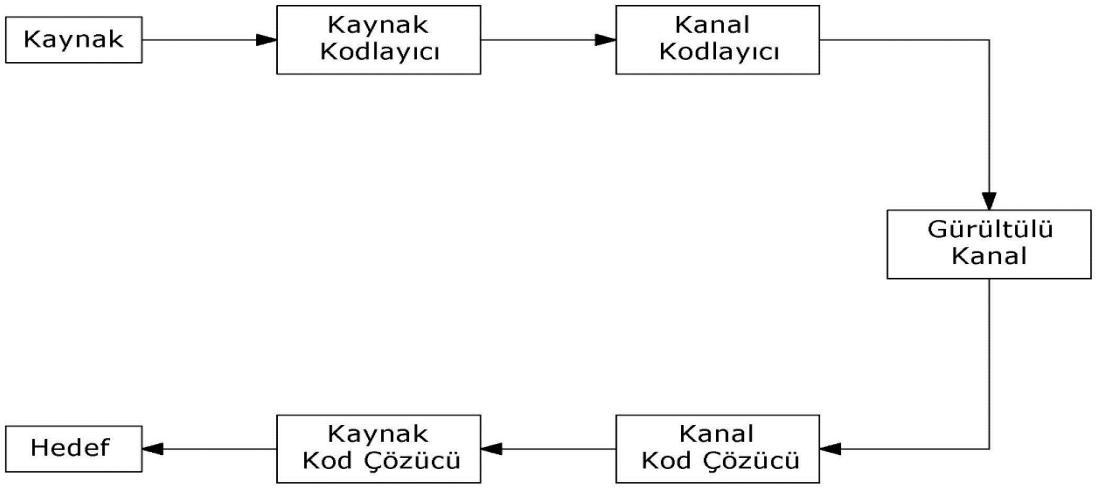


Şekil 1.Haberleşme Blok Yapısı

3.2.İLETİŞİM SİSTEMİ

İletişim sistemlerinin en büyük ihtiyacı iletişim sırasında yaşanan veri kayıpları ve hızdır. Veri, iletim ortamına aktarıldığımda iletim ortamındaki gürültüden ciddi oranda etkilenmekte ve veri kayıpları yaşanmaktadır. Veri aktarımı sırasında veri kaybının önüne

geçmek için çeşitli yollara başvurulmaktadır. Bu yollardan en yaygın olarak kullanıla kanal kodlaması. Kanal kodlamasında veriye ek bitler eklenerek verideki bozulmanın önüne geçmek isteniyor. Ancak bu yol kullanılırken veri yolu tamamladıktan sonra ilk haline getirmek için hata düzeltmesi gerekmektedir ve bu işlem içinde hata düzeltme kodları kullanılmaktadır.

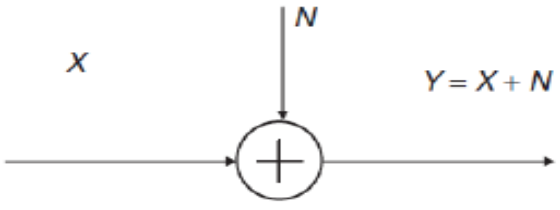


Şekil 2.İletişim Sistemi Yapısı

3.2.1.Kodlama Teorisi

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

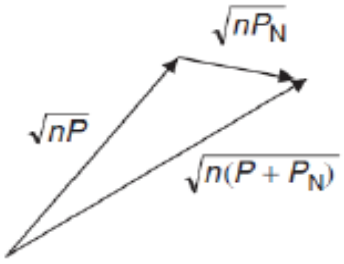
X giriş serisi, N gürültü olmak üzere, çıkış serisi Y'nin oluşumu şekil 3'de gösterilen şekildedir ve denklem 1 ile ifade edilir.



Şekil 3.Gürültülü Yapı

Denklem 1: Y = X + N

Giriş serisi X, giriş serisinin gücü P, giriş serisinin ve gürültünün bit sayısı n, gürültünün gücü *P*C olmak üzere çıkış Y, X'in ve gürültünün vektörel toplamıdır. Şekil 4'de gösterildiği gibi ifade edilmektedir ve denklem 2'deki şekilde tanımlanmaktadır.



Şekil 4 Vektörel Toplam

Denklem 2: $\sqrt{n(P+p_N)} = \sqrt{nP} + \sqrt{nP_n}$

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.

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

3.2.1.1.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.

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.

3.2.1.2.İleri Hata Düzeltme Kodlaması (FEC)

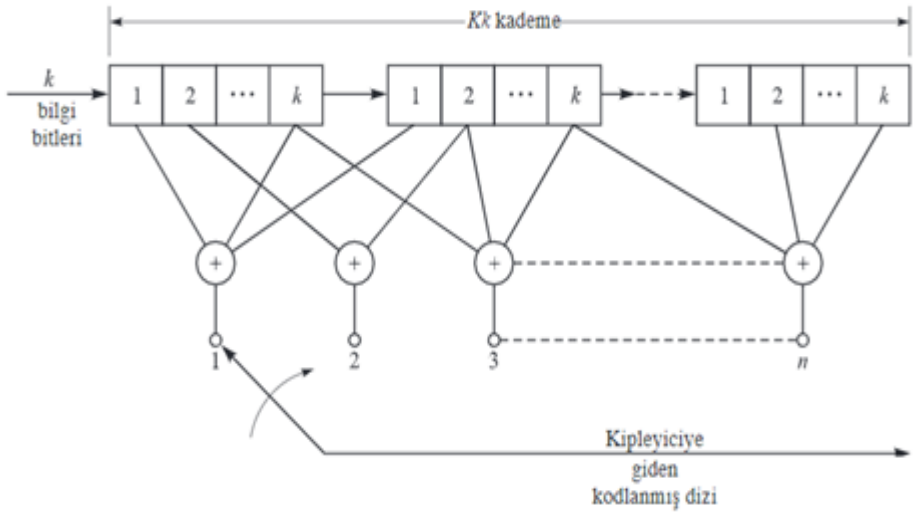
FEC, kodlanacak veriye ek bitler ekleyerek kanalda bu şekilde hareket etmesi esasına dayanmaktadır. Bu tip kodlar hatayı yakalama ve düzeltme kabiliyetine sahiptir. FEC, Blok Kodlar ve Evrişimli Kodlar olmak üzere iki bölümde incelenebilir.

3.2.1.2.1.Blok Kodlama

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. 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.

3.2.1.2.2.Evrişimli Kodlar ve Yapısı

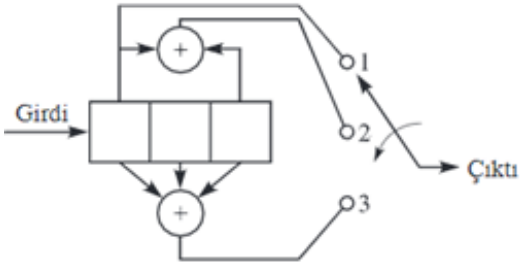
Bir evrişimli kod, iletilecek bilgi dizisini bir doğrusal sonlu durum kaydıran yazmaçtan geçirerek üretilir. Genel olarak, kaydıran yazmaç aşağıdaki şekilde gösterildiği gibi K(k-bit) kademeden ve n doğrusal cebirsel fonksiyon üretecinden oluşur. İkili olduğu varsayılan kodlayıcının girdi verileri, kaydıran yazmaç boyunca bir seferde k bit olarak kaydırılır. Her m bit giriş dizisi için çıkış bitlerinin sayısı n bittir. Sonuç olarak, kod oranı bir kod blok için kod oranının tanımı ile tutarlı olarak R=k/n olarak tanımlanır.



Şekil 5.Evrişimli Kodlayıcı

K parametresi, evrişim kodunun kısıt uzunluğu olarak adlandırılır. Bir evrişimli kodu tanımlamak için bir yöntem, tıpkı blok kodlar için yapıldığı gibi, üreteç matrisini vermektir. Genel olarak, bir evrişimli kod için üreteç matrisi, yarı sonsuzdur, çünkü girdi dizisi uzunluk olarak yarı sonsuzdur. Üreteç matrisini belirtmenin bir alternatifi olarak, n adet modülo-2 toplayıcının her biri için bir vektör olarak, bir n adet vektörün oluşturduğu küme olarak fonksiyonel eşdeğer bir temsili kullanacaktır. Her vektörün Kk boyutu vardır ve kodlayıcının bu modülo-2 toplayıcıya olan bağlantılarını içerir. Vektörün i. Konumunda bir 1, kaydıran yazmaçta karşılık gelen kademede modülo-2 toplayıcıya bağlandığında ve belirli bir konumda 0, bu kademe ve modülo-2 toplayıcı arasında herhangi bir bağlantının olmadığını gösterir.

Daha açıklayıcı olmak gerekirse, aşağıda gösterilen şekilde K=3 kısıt uzunluğunda, k=1 ve n=3 uzunluğuna sahip ikili evrişimli kodlayıcı ele alınsın. Başlangıçta, kaydıran yazmacın tümü-sıfırlar durumunda olduğu varsayılır. İlk giriş bitinin 1 olduğu varsayılsın. Bunu takiben 3 bitlik çıkış biti 111 olur. İkinci bitin 0 olduğu varsayılsın. Bu durumda çıkış dizisi 001 olacaktır. Eğer üçüncü bit 1 ise, çıkış 100 olacaktır ve benzer şekilde devam edecektir. Şimdi her üç bitlik çıkış dizisini 1,2 ve 3, yukarıdan aşağıya doğru üreten fonksiyon üreteçlerinin çıkışlarını numaralandırıldığı varsayılsın. İlk fonksiyon üreteci sadece ilk kademeye bağlı olduğundan (modülo-2 toplayıcıya gerek yoktur), üreteç g1= [100] olarak verilir. İkinci fonksiyon üreteci 1 ve 3'üncü kademelere bağlıdır. Böylelikle, g2= [101] olur.



Şekil 6. K=3, k=1, n=3 evrişimli kodlayıcı

Son olarak da g3= [111] olur. Bu kod için üreteçler daha uygun olarak sekizli biçimde yazılabilir. K=1 olduğunda, kodlayıcıyı belirtmek için her K boyutunda n üretece ihtiyaç olduğu sonucuna varılmaktadır.

g1,g2 ve g3'ün kodlayıcı girişinden üç çıkışa dürtü yanıtları oldukça açıktır. Bu durumda, kodlayıcının girişi u bilgi dizisi ise, üç çıktısı

$$c^{(1)} = u \star g1$$

$$c^{(2)} = u \star g2$$

$$c^{(3)} = u \star g3$$

Olarak verilmektedir. Burada $\star$ evrişim işlemini belirtmektedir. Karşılık gelen c kod dizisi, $c^{(1)}, c^{(2)}\ ve\ c^{(3)}$ çıktılarının

$$c = \left(c_1^{(1)}, c_1^{(2)}, c_1^{(3)}, c_2^{(1)}, c_2^{(2)}, c_2^{(3)}, \dots\right)$$

Biçiminde serpiştirilmesi ile bulunur. Evrişim işlemi, dönüşüm alanındaki çarpma ile eşdeğerdir. U'nun D dönüşümü,

$$u(D) = \sum_{i=0}^{\infty} u_i\, D^i$$

Olarak tanımlanır. Üç dürtü yanıtı g1, g2 ve g3 için aktarım fonksiyonu

$$g_1(D) = 1$$

$$g_2(D) = 1 + D^2$$

$$g_3(D) = 1 + D + D^2$$

Olur. Çıktıların dönüşümleri

$$c^{(1)}(D) = u(D)g1(D)$$

$$c^{(2)}(D) = u(D)g2(D)$$

$$c^{(3)}(D) = u(D)g3(D)$$

İle verilir ve c kodlayıcı çıktısı dönüşümü

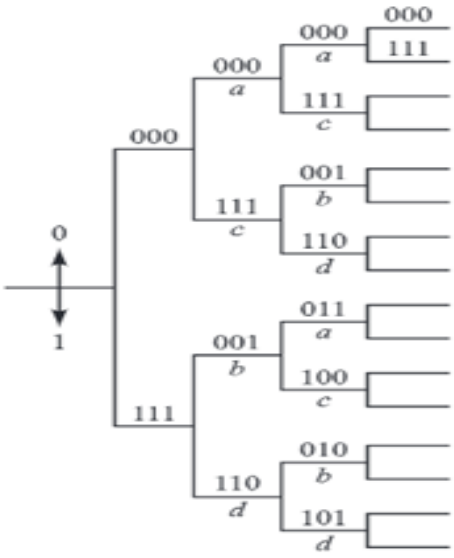
$$c(D) = c^{(1)}(D^3) + Dc^{(2)}(D^3) + D^2c^{(3)}(D^3)$$

Şeklindedir.

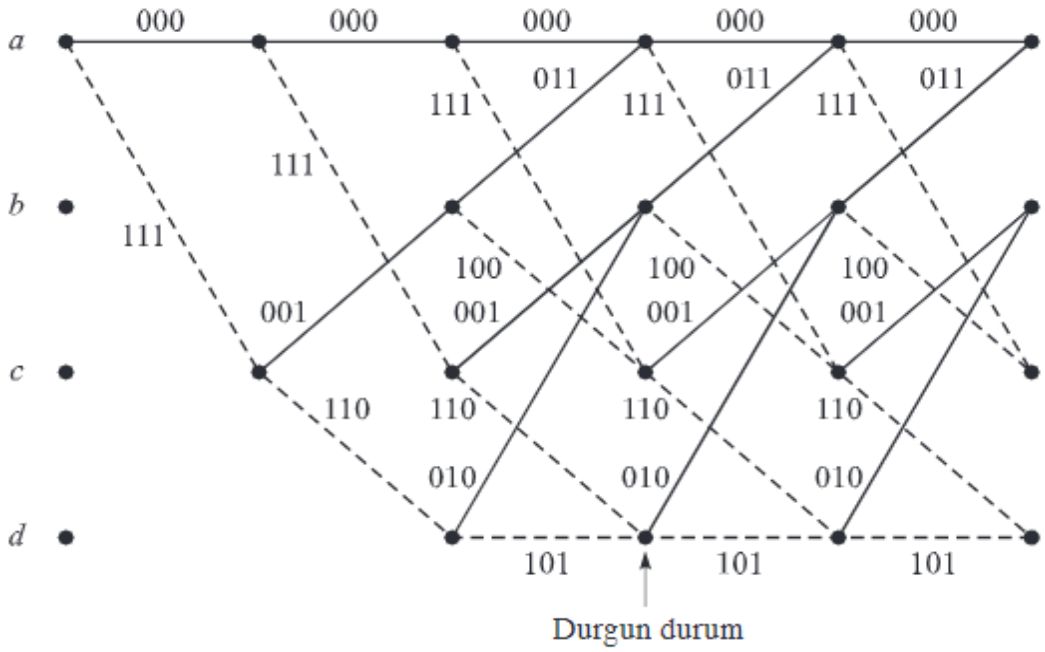
3.2.1.2.2.1.Ağaç, Kafes ve Durum Şemaları

Bir evrişimli kodu tanımlamak için sıklıkla kullanılan üç değişik yöntem vardır. Bunlar ağaç şeması, kafes şeması ve durum şemasıdır. Kodlayıcının başlangıçta tümü sıfırlar durumunda olduğu varsayılırsa, şema, ilk girdi biti 0 ise çıktı dizisinin 000 olduğunu ve ilk bitin 1 olması durumunda çıktı dizisinin 111 olduğunu gösterir. İlk girdi biti 1 ve ikinci bit 0 ise, ikinci üçlük çıktı biti kümesi 001 olur. Ağaç boyunca devam ederken, eğer üçüncü bit 0 ise, o zaman çıktı 011 olur. Üçüncü bit 1 iken, çıktı 100 olur. Belirli bir dizinin bizi ağaçtaki belirli bir düğüme götürdüğü göz önüne alındığında, dallanma kuralı bir sonraki girdi biti 0 ise üst dalı eğer 1 ise alt dalı takip etmektir. Böylece, girdi dizisi tarafından belirlenen ağaçtan belirli bir yol izlenir.

Aşağıda gösterilen evrişimli kodlayıcı tarafından oluşturulan ağacın yakından incelenmesi, yapının üçüncü kademeden sonra kendini tekrar ettiği ortaya koymaktadır. Bu davranış, K=3 kısıt uzunluğu ile uyumludur. Yani, her bir kademedeki 3-bitlik çıktı dizisi, girdi biti ve daha önceki 2 girdi biti, yani, kaydıran yazmacın ilk iki kademesinde bulunan 2 bit ile belirlenir. Kaydıran yazmacın son kademesindeki bit sağa kaydırılır ve çıkışı etkilemez. Böylece, her bir girdi biti için 3 bitlik çıkış dizisi, girdi biti ve kaydıran yazmacın a=00, b=01, c=10, d=11 olarak belirtilen dört olası ile tanımlandığı söylenilebilir.



Şekil 7. 1/3 oranlı, K=3 evrişimli kod için Ağaç Şeması



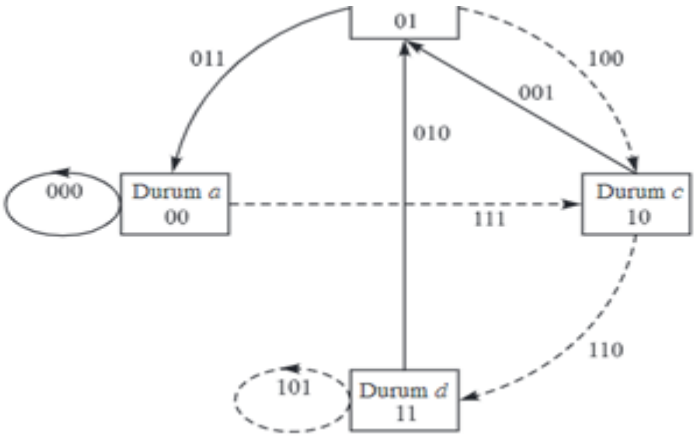
Şekil 8. 1/3 oranlı, K=3 evrişimli kod için kafes şeması

Ağaçtaki her düğümü, kaydıran yazmaçtaki dört olası duruma karşılık gelecek şekilde etiketlersek, üçüncü kademede, a etiketli iki, b etiketli iki, c etiketli iki ve d etiketli iki düğümün olduğu görülür. Şimdi aynı etikete (aynı duruma) sahip olan iki düğümden çıkan tüm dalların, özdeş çıktı dizileri üretmeleri bakımından özdeş olduğu gözlemlenir. Bu, aynı etikete sahip iki düğümün birleştirilebileceği anlamına gelir. Bunu ağaca uygularsak, daha özlü, yani bir kafes olan başka bir şema elde edilir.

Kodlayıcının çıktısı, giriş ve kodlayıcının durumu tarafından belirlendiğinden, durum şeması kafes şemasından daha özlü bir şemadır. Durum şeması, kodlayıcının olası durumlarının ve bir durumdan diğerine olası geçişlerin bir çizgesidir. Bu şema olası geçişlerin

$$a \xrightarrow{0} a\,,a \xrightarrow{1} c\,,b \xrightarrow{0} a\,,b \xrightarrow{1} c\,,c \xrightarrow{0} b\,,c \xrightarrow{1} d\,,d \xrightarrow{0} b\,,d \xrightarrow{1} d$$

Olduğunu göstermektedir. Burada $\alpha \xrightarrow{1} \beta$ girdi 1 olduğunda a durumundayken b durumuna geçişini gösterir. Durum şemasındaki her bir dalın yanına gösterilen 3 bit, çıktı bitlerini temsil eder. Çizgedeki noktalı bir çizgi, giriş bitinin 1 olduğunu, düz çizgi ise giriş bitinin 0 olduğunu gösterir.



Şekil 9. 1/3 oranlı, K=3 evrişimli kod için Durum Şeması

Genellemek için, bir k/n oranı oranlı, K kısıt uzunluklu evrişimli kodunun, ağaç şemasının her bir düğümünden çıkan 2^k adet dal ile tanımlandığı belirtilir. Kafes ve durum şemalarının her biri 2^k(K-1) olası duruma sahiptir. Her bir duruma giren 2^k dal

ve her bir durumdan çıkan 2^k dal vardır (kafes ve ağaçta, ilk baştaki geçici davranıştan sonra geçerlidir). Yukarıda tanımlanan üç tip şema, aynı zamanda, ikili olmayan evrişimli kodları temsil etmek için de kullanılmaktadır. Kod alfabesindeki sembol sayısı q=2^k, k>1 olduğunda, sonuçta ortaya çıkan ikili olmayan kod, eşdeğer bir ikili kod olarak da temsil edilebilir.

3.2.1.2.2.2.Evrişimli Kodların Kod çözümü

Evrişimli kodların kod çözümü için farklı yöntemler vardır. Blok kodlara benzer şekilde, evrişimli kodların kod çözümü de ya yumuşak karar ya da sıfır-bir karar kod çözme ile yapılabilir. Ek olarak, evrişimli kodların en iyi kod çözümü, en büyük olabilirlik veya en büyük sonsal ilkesini kullanabilir. Yüksek kısıt uzunluklu evrişimli kodlar için en iyi kod çözme algoritmaları çok karmaşık hale gelir. Bu gibi durumlarda genellikle en iyiye yakın kod çözme algoritmaları kullanılır.

3.3.Viterbi Algoritması

Bir hafızasız kanal için bir blok kodun kod çözümünde, alınan kod kelimesi ile iletilen 2^k tane olası kod kelimesi arasındaki uzaklıklar (sıfır-bir karar kod çözme için Hamming uzaklığı ve yumuşak karar kod çözme için Öklid uzaklığı) hesaplanmaktadır. Ardından, alınan kod kelimesine en yakın uzaklıkta olan kod kelimesi seçilir. 2^k ölçütün hesaplanmasını gerektiren bu karar kuralı, p<1/3 ve toplanır beyaz Gauss gürültüsü kanal ile ikili simetrik kanal için en küçük hata olasılığına yol açması bakımından en iyidir.

Sabit n uzunluğunda bir blok koddan farklı olarak, bir evrişimli kodlayıcı temel olarak bir sonlu durum makinesidir. Bu nedenle en iyi kod çözücü, hafızalı sinyaller için bir en büyük olabilirlik dizi kestiricisidir. Bu nedenle, bir evrişimli kodun en iyi kod çözümü, en olası dizi için kafesin içinde bir tarama içerir. Kod çözücüden sonra sezicinin sıfır-bir veya yumuşak karar alıp almadığına bağlı olarak, kafes taramasında kullanılan ölçüt, sırasıyla Hamming ölçütü veya Öklid ölçütü olabilir.

Sırasıyla 000 ve 100 iki bilgi dizisi ve 000 000 000 ve 111 001 011 iletilen dizilerine karşılık gelen, kafeste başlangıç durumu a'dan başlayan ve üç durum geçişinden (üç dal) sonra a durumu ile birleşen iki yol düşünülsün. Aktarılan bitler $\{c_{jm}, j = 1,2,3; m = 1,2,3\}$ ile gösterilir.

Burada j indeksi j'inci dalı ve m indeksi bu daldaki m'inci biti gösterir. Buna karşılık, kip çözücünün çıktısı olarak $\{r_{jm}, j = 1,2,3; m = 1,2,3\}$ tanımlanır. Kod çözücü, sıfır-bir karar kod çözme, işlemini gerçekleştirirse, iletilen her bit için sezici çıktısı, 0 ve 1 olur. Diğer taraftan yumuşak karar kod çözme kullanılırsa ve kodlanmış dizi, ikili evre uyumlu PSK tarafından iletilirse, kod çözmenin girdisi,

$$r_{jm} = \sqrt{\varepsilon_c}(2c_{jm} - 1) + n_{jm}$$

Biçimindedir. Burada njm toplanır gürültüyü ve Ec her kod biti için iletilen sinyal enerjisini göstermektedir.

Bir ölçüt kafes boyunca i'inci yolun j'inci dalı için, iletilen dizi $\{c_{jm}^{(i)}, m = 1,2,3\}$ Üzerine koşullandırılmış olan $\{r_{jm}, m = 1,2,3\}$ dizisinin ortak olasılığının logaritması olarak tanımlanır. Yani, ölçüt

$$\mu_j^{(i)} = \log p(r_j | c_j^{(i)}), j = 1,2,3, \ldots$$

Olarak tanımlanır. Dahası, kafes boyunca B adet dal içeren i'inci yol için bir ölçüt

$$PM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)}$$

Olarak tanımlanır.

Kafes boyunca iki yol arasında karar verme kriteri, daha büyük ölçüte sahip olanı seçmektir. Bu kural, doğru bir karar verme olasılığını en büyük yapar veya eşdeğer olarak, bilgi bitleri dizisi için hata olasılığı en küçük yapar. Örneğin, sıfır-bir karar kod çözme işleminin kip çözücü tarafından gerçekleştirildiği ve alınan diziyi {101 000 100} olarak elde ettiği varsayılsın. İ = 0, üç dallı tümü sıfırlar yolunu göstersin ve i=1 başlangıç durumu a'da başlayan ve üç geçişten sonra tümü sıfırlar yolu ile birleşen ikinci üç dallı yolu tanımlasın. Bu iki yol için ölçütler.

$$PM^{(0)} = 6\log(1 - \mathrm{p}) + 3\log p$$

$$PM^{(1)} = 4\log(1 - \mathrm{p}) + 5\log p$$

İle verilir. Burada p bir hata olasılığıdır. P <½ olduğu varsayılırsa PM (0) ölçütünün PM (1) ölçütünden daha büyük olduğu görülür. Bu sonuç, i=1 yolunun alınan yoldan d=5 Hamming uzaklığında iken, tümü sıfırlar yolundan alınan diziden d=3 Hamming uzaklığında olması gözlemiyle tutarlıdır. Böylece Hamming uzaklığı, sıfır-bir karar kod çözme için eşdeğer bir ölçüttür.

Benzer şekilde, yumuşak karar kod çözme işleminin uygulandığı ve kanalın beyaz Gauss gürültüsünü sinyale eklediği varsayılsın. Bu durumda, kip çözücü çıktısı olasılık yoğunluk fonksiyonu istatistiksel olarak

$$p\left(r_{jm}\middle|c_{jm}^{(i)}\right)=\frac{1}{\sqrt{2\pi\sigma^2}}exp\left\{-\frac{\left[r_{jmc}-\sqrt{\varepsilon}(2c_{jm}^{(i)}-1)\right]^2}{2\sigma^2}\right\}$$

İle açıklanır. Tüm dal ölçütleri için ortak olan terimler ihmal edilirse, i'inci yolun d'inci dalı için dal metriği

$$\mu_j^{(i)}=\sum_{m=1}^{n}r_{jm}(2c_{jm}^{(i)}-1)$$

Biçiminde ifade edilebilir. Burada, örnekte n=3'tür. Dolayısıyla, değerlendirilen iki yol için ilinti ölçütleri

$$CM^{(0)}=\sum_{j=1}^{3}\sum_{m=1}^{3}r_{jm}(2c_{jm}^{(0)}-1)$$

$$CM^{(1)}=\sum_{j=1}^{3}\sum_{m=1}^{3}r_{jm}(2c_{jm}^{(1)}-1)$$

Olarak bulunur.

Yukarıdaki tartışmadan, kod çözme işlemi için kafeste,

$c^{(m)}=\max_{c\in\mathrm{T}}\sum_j\log p\left(r_j\middle|c_j\right)$, Genel bir hafızasız kanal için

$c^{(m)}=\min_{c\in\mathrm{T}}\sum_j\left\|r_j-c_j\right\|^2$, Yumuşak karar kod çözme için

$c^{(m)}=\min_{c\in\mathrm{T}}\sum_j d_H(y_j,c_j)$, Sıfır-bir karar kod çözme için

Koşullarını sağlayan bir c kod dizisi aranması gerektiği gözlemlenmektedir. Y'nin sıfır-bir karar kod çözmesinin, kip çözücü çıktısı r üzerinde ikili (sıfır-bir) kararların sonucunu ifade ettiği unutulmamalıdır. Aynı zamanda sıfır-bir karar durumunda, c'nin bileşenleri 0 ve 1 olan ikili kodlanmış diziyi belirtirken, yumuşak karar durumunda c'nin bileşenler Ec^1/2 şeklindedir. Yukarıdan açıkça anlaşılacağı gibi, tüm durumlarda en büyük olasılıkla kod çözme kafes içinde bir eklenir ölçütü en küçük yapan veya en büyük yapan bit yolu bulmayı gerektirir. Bu, tartışıldığı gibi Viterbi algoritması kullanarak yapılır.

Genel olarak, k=1 ve K kısıt uzunlukta bir ikili evrişimli kod Viterbi algoritması ile çözüldüğünde, 2^K-1 durum vardır. Bu nedenle, her kademede 2^K-1 tane sağ kalan yolu ve sağ kalan her yol için bir tane olmak üzere 2^K-1 tane ölçüt vardır. Dahası, bir sefer v bitin K tane (k-bit) kaydıran yazmaç kademesinden oluşan bir kodlayıcıya kaydırıldığı bir evrişimli ikili kod, 2^k*K-1 duruma sahip bir kafes oluşturur. Sonuç olarak, Viterbi algoritması vasıtasıyla böyle bir kodun kod çözümü, 2^k*K-1 sağ kalan yolun ve 2^k*K-1 ölçütün takibini gerektirir. Kafesin her kademesinde, her bir düşümde birleşen 2^k tane yol vardır. Ortak bir düğümde toplanan her yol bir ölçütün hesaplanmasını gerektirdiğinden, her düğüm için hesaplanmış 2^k ölçüt vardır. Her bir düğümde birleşen 2^k yoldan sadece biri kalır ve bu en olası (en küçük uzaklıklı) yoldur. Bu nedenle, her kademede gerçekleştirilen kod çözme işlemindeki hesaplama sayısı, k ve K ile katlanarak artar. Hesaplama yükündeki üstel artış, Viterbi algoritmasının nispeten küçük K ve k değerlerinde kullanımını sınırlar.

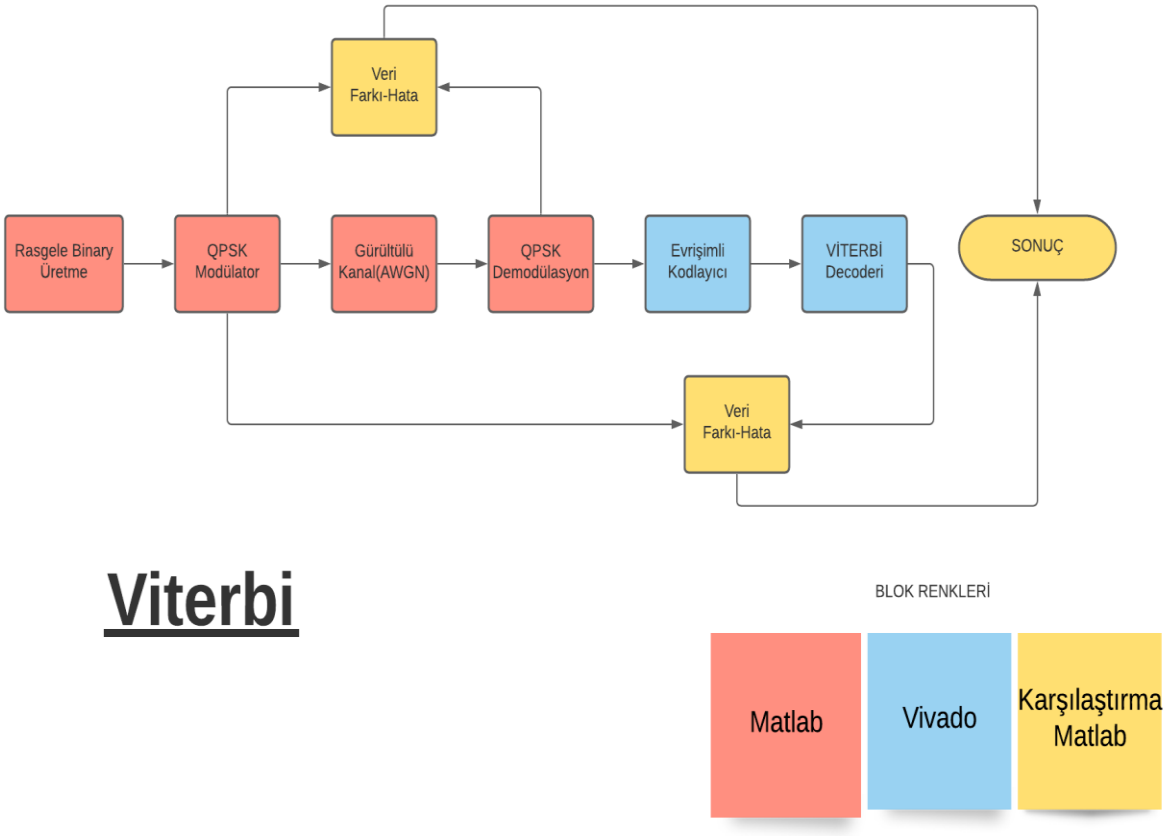
Evrişimli kodlanmış uzun bir bilgi dizisinin kod çözülmesindeki kod çözme gecikmesi çoğu pratik uygulama için genellikle çok uzun olmaktadır. Ayrıca, sağ kalan dizilerin tüm uzunluğunu saklamak için gerekli olan bellek büyük ve pahalıdır. Algoritmanın en iyi başarımını önemli ölçüte etkilemeden sabit bir kod çözme gecikmesi ile sonuçlanacak şekilde Viterbi algoritmasını değiştirmektir. Değişikliğin, herhangi bir t zamanında, sadece sağ kalan dizide en son çözülmüş bilgi bitlerini (sembollerini) tutması gerektiği unutulmamalıdır. Her yeni bilgi biti alındıkça, ölçütleri karşılaştırarak ve en büyük ölçüte sahip dizideki bit lehine karar vererek kafeste geriye doğru alınan bit (sembol) üzerinde son karar verilir

4.ARAŞTIRMA SONUÇLARI

4.1.Tasarım Girişi

Proje kapsamında teorik olarak elde ettiğimiz verileri kullanarak mevcut tasarımları da inceleyerek VHDL dili ile programlamaya başladık. Yapacağımız Viterbi dekoderi iki yapıdan oluşmakta. Bu yapılar dekoder ve dekoder yapısı.

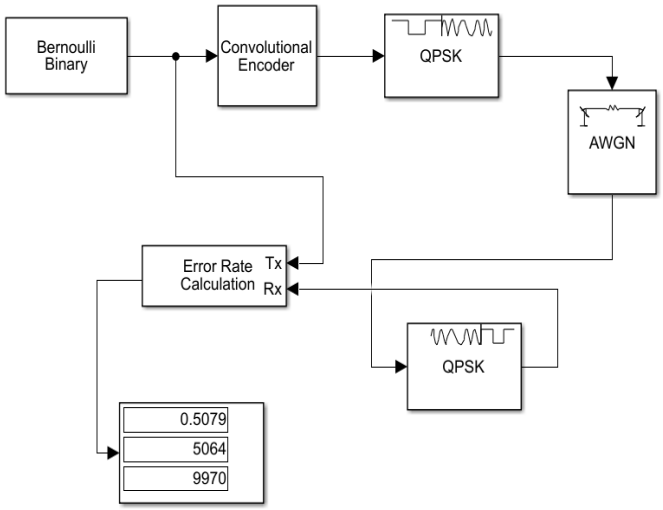
Yaptığımız proje kapsamında Matlab ve Vivado Simulator kullanıldı. Proje toplamda 7 fazdan oluşmakta. Bu fazları aşağıdaki blok şekil x de göstermekteyiz



Şekil 10.Genel Blok Gösterim

4.1.1. Rasgele Binary Üreteci, QPSK Modülasyon ve Demodülasyon

Matlab Simulink aracılığı ile rasgele binary üretmek için Bernoulli Binary üretecinden yararlandık. Bu üreteçle üretilen binary datalar Matlab Simulinkte bulunan hazır QPSK modülasyon, gürültü ve demodülasyon bloğundan geçirildi ve sonuçlar BER karşılaştırıcısıyla karşılaştırıldı.



Şekil 11.QPSK modülasyon ve demodülasyon

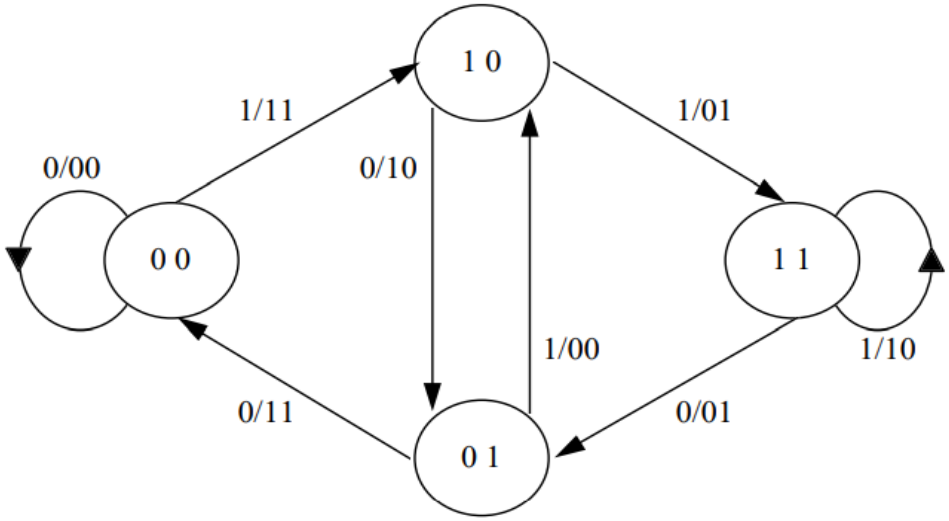
Bu sonuçların ardından asıl yapmak istediğimiz Viterbi dekoderini VHDL ile kodlama safhası başlamıştı.

4.1.2. Viterbi Dekoderi

Viterbi dekoderi iki aşamadan VHDL dili ile kodlamaya başladık ilk kodlamada evrişimli kodlayıcı yani enkoder kodlaması yapıldı ardından enkoder den çıkan verileri Viterbi dekoderine sokarak çıkan sonuçları kaydettik.

4.1.2.1. Evrişimli Kodlayıcı (Convolutional Encoder)

Evrişimli kodların temsil edilmesi işleminde 3 temel yöntem mevcuttur. İlki Generator(üreteç) gösterim, Tree (Ağaç), State (Durum) ve Trellis(kafes) gösterim yöntemleri mevcuttur. Tez çalışmamız State (Durum) gösterim metodu ile evrişimli kodların kodlanması yöntemine dayanır. Durum diyagramı yönteminde iki hafıza (Flip Flop) bloğu kullanarak kaydıran yazmaçlar (Shift Register) ile durumlar güncellenir.



Şekil 12.dekoder durum diyagramı

Input Stream	0	0	1	0	1	1	1
Next State	S0	S0	S1	S2	S1	S3	S3
Output Stream	0 0	0 0	1 1	1 0	0 0	0 1	1 0

Şekil 13 Rasgele veri akış bit diyagramı

Durum diyagramında, kodlayıcının durum bilgisi dairelerde gösterilir. Her biri yeni girdi bilgisi biti, bir durumdan diğerine geçişe neden olur. Yol x/c olarak belirtilen durumlar arasındaki bilgi, girdi bilgisi bit X'in temsil eder ve çoklu çıktı bitler c'yi ifade eder. Evrişimli kodlamanın durumları ve bu durumları bellekte tutan hafıza birimleri başlangıçta sıfırdır. Şekil 12'de iki durumlu bir bellek için durum geçiş şeması gösterilmektedir.

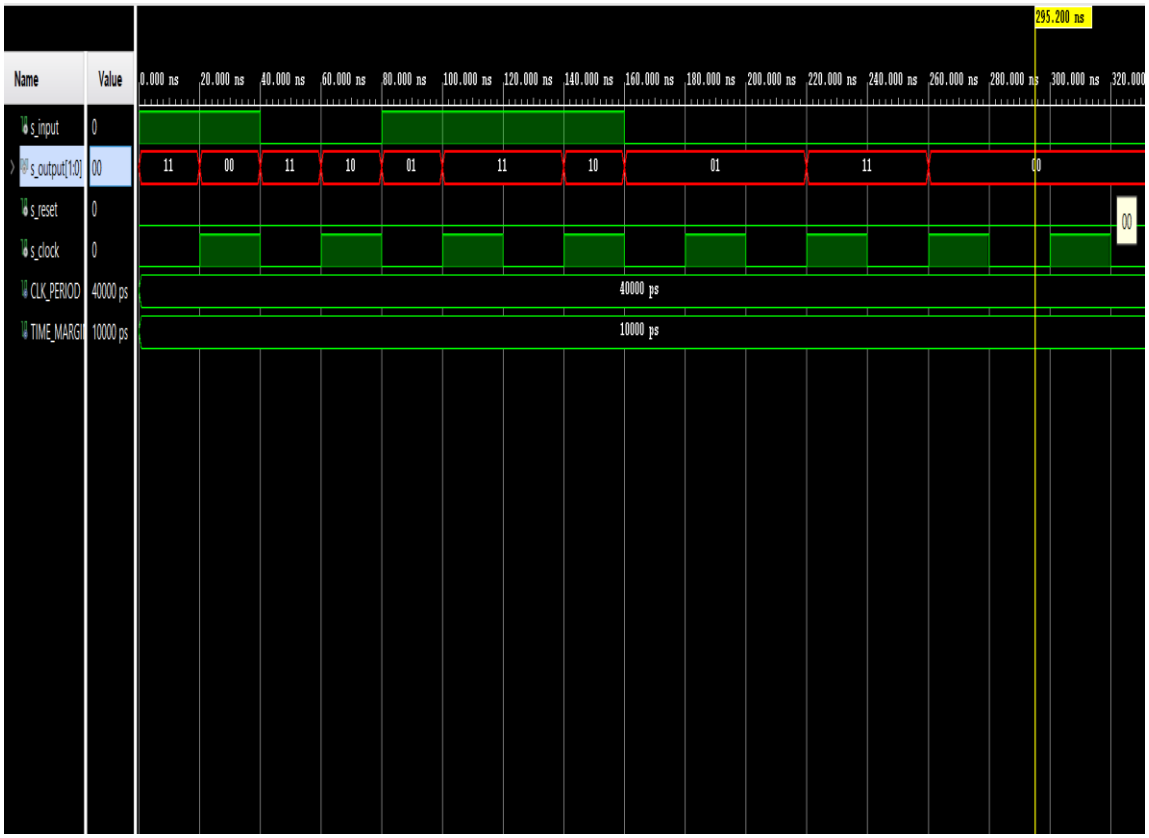
Viterbi kodlayıcı işlevini yerine getiren VHDL kodumuzda 3 bitlik bir hafıza bellekte ayırıyoruz, başlangıçta bu bit "000"dır. Sonra, senkron sıfırlama işlemi yapıyoruz. İlk önce saatin yükselen kenarı tetiklemeli daha sonra sıfırlama(reset) '1' olmadığında yazdıran yazmacı (Shift Register) güncellenecek aksi durumda sıfırlanacaktır. Bu işlemlerimizden sonra iki hafıza birimlerimizin içeriğini belirlemede,

giriş biti ve hafıza içindeki birimler ile xor işlemine tabi tutulur. Çıkışlar bu sayede güncellenmiş olur. Şekil 14'te xor doğruluk tablosu verilmiştir.

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Şekil 14.. Xor doğruluk tablosu

Encoder doğruluk tablosu doğrultusunda VHDL ile kodlanarak "1011000" girişli encoder çıkışı aşağıda gözlemlenmiştir.

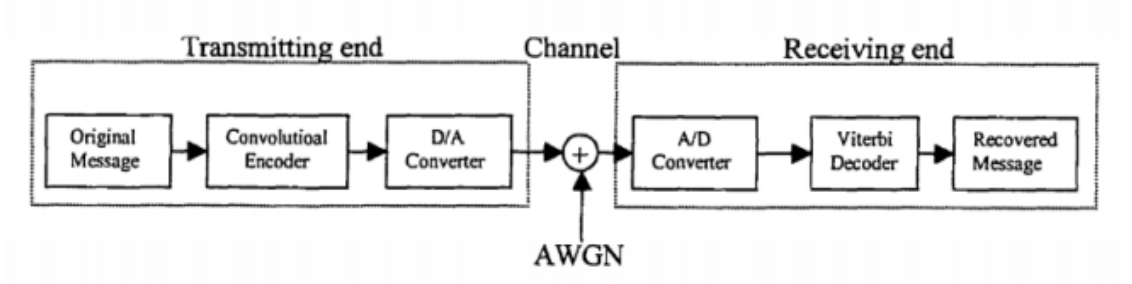


Şekil 15.Encoder Simülasyon Çıktısı

Bu uygulamanın sonucunda elde ettiğimiz çıkışı rtl seviyesinde blok haline getirerek girişi TextIO çıkışını da Viterbi dekoderine bağlayarak bütünleşik yapı elde ettik.

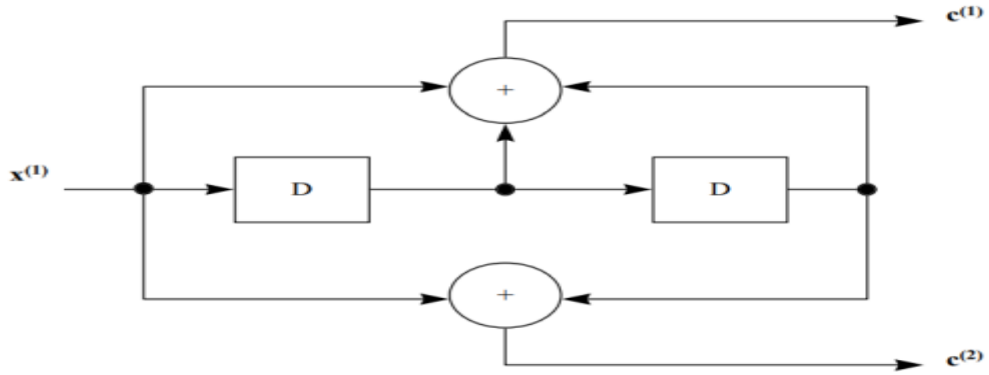
4.1.2.2. Viterbi Dekoderi

Viterbi algoritması, HDM (Hidden Markov Model) modelinde gözlemlenen çıktıların belirli bir dizisine dayalı olarak en olası gizli durum dizisini bulmak için bir maksimum olasılık yöntemidir. Bununla birlikte kod kafesinde özel yapıdan yararlanarak hesaplama yükünü azaltır. Algoritma, t1 zamanında alınan sinyal ile t1 zamanında her duruma giren tüm kafes yolu arasındaki mesafenin ölçüsünün hesaplanmasını içerir. Olası olmayan yolların erken reddedilmesi, kod çözme karmaşıklığını azaltır. Viterbi algoritmasının avantajı, kendi kendini düzeltme özelliğine sahip olmasıdır. Kod, enerji aktarımının en aza indirilmesi ve iletilen yanlış bitleri düzeltmek için çok iyidir. Hata düzeltme, herhangi bir iletişim sisteminin ayrılmaz bir parçasıdır ve bu amaçla evrişimli kodlar (Convolutional Code) ileri hata düzeltme kodları olarak yaygın olarak kullanılmaktadır. Evrişimli kodları çözmek için kullanılan algoritmalar, Viterbi algoritması ve Sıralı algoritmadır. Sıralı kod çözme, uzun kısıtlama uzunluklu(K) evrişimli kodlarla iyi performans gösterir fakat değişken bir kod çözme süresine sahiptir. Viterbi kod çözme, evrişimli kodların kodunu çözmek için en iyi tekniktir, ancak daha küçük kısıt uzunluklarıyla sınırlıdır. (K<10) Sıralı ile karşılaştırıldığında sabit bir kod çözme süresine sahiptir. Çok düşük hata olasılıklarına ulaşmak için daha uzun kısıtlama uzunlukları(K) gerekir ve sıralı kod çözme daha avantajlı hale gelir. Bir kod çözücünün performansı, hatalı kodun; çözülmüş çıktı bitlerin sayısı, Bit Hata Oranı (BER) ile karakterize edilir. Viterbi algoritması, alınan her sembole uygulanan "Hamming Distance" mesafelerine göre bir minimum mesafe yolu belirler. Viterbi algoritması üç ana bölümden oluşur. Branch Metric Unit, Path Metric Unit, Traceback Unit' dir



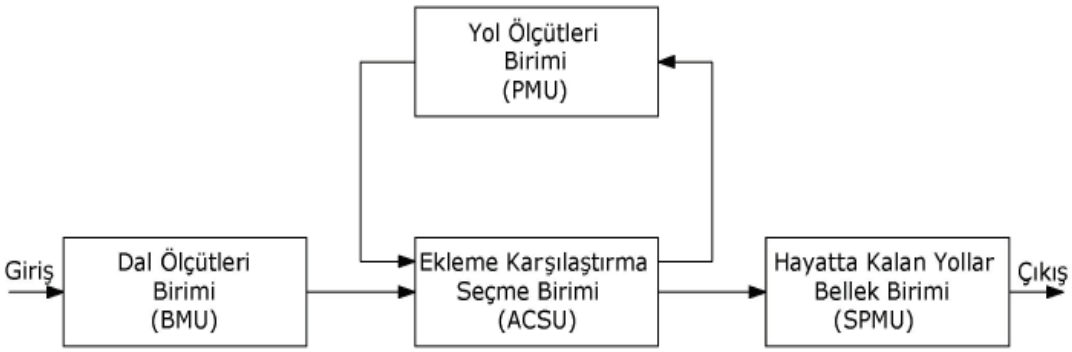
Şekil 16.İletişim Kanal

Yukarıda gösterilen sistem bizim referans aldığımız şematiğimizdir. Bununla birlikte evrişimli kodların g(1,1,1) ve g(1,0,1) jeneratör polinomları ile gösterimi aşağıdaki gibidir.



Şekil 17.Evrişimli kod g (1,1,1) ve g (1,0,1)

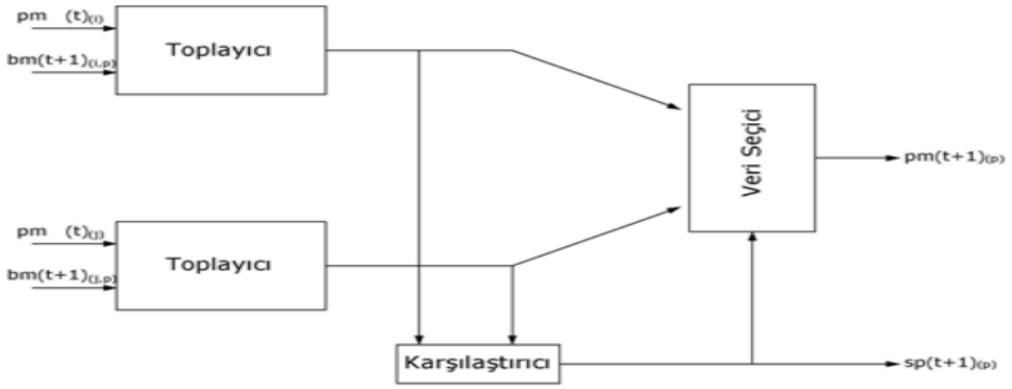
Viterbi bir kod çözme algoritması olduğundan ilk iş Viterbi'nin aşamalarını açıklamak olacaktır. Viterbi üç ana bölümden oluşur: Branch Metric Unit (Dal Ölçü Birimi), Path Metric Unit (Yol Ölçü Birim) ve Traceback Unit (Geriye Dönük İzleme) aşamalardan meydana gelir. Bu aşamaların bloklar ile temsili aşağıdaki gibidir.



Şekil 18. Viterbi Ana Bölümleri

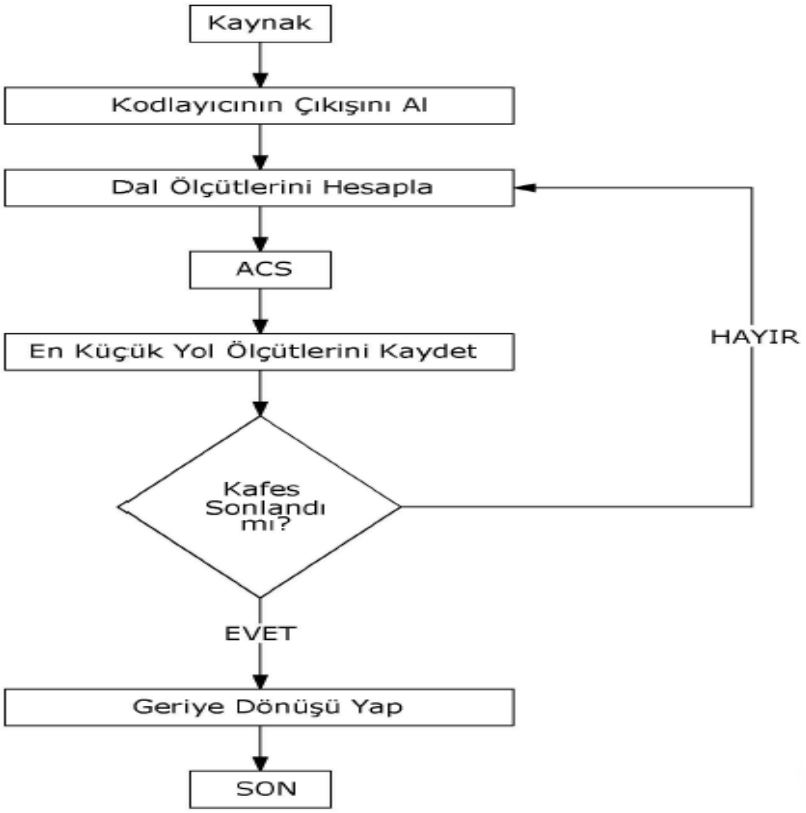
İlk aşama olan dal ölçü biriminde HD (Hamming Distance) değerleri her seferinde anlık, önceki zaman anındaki durumlar arasındaki yollar ve geçerli zaman anındaki durumlar için dal ölçüm değerlerini tutar. Dal ölçü birimi için, HD veya Euclidean (Öklid) Distance kullanılır. İkinci aşama olan yol ölçü biriminde, birikmiş hata ölçüsü 2^(K-1) olası yolları içerir. Örneğin K=3 için 4 farklı yol mevcuttur. Geçerli dal ölçüsü önceki dal ölçü birimine eklenir ve her iki mesafe tüm ACS(Ekleme-Karşılaştırma-Seçme) birimleri ile karşılaştırılır. Hız açısından Viterbi çözücünün performansı esas olarak ACS

birimlerinin sayısı ve hesaplama süreleri ile belirlenir. ACS birimi; iki toplayıcı bloğu, bir karşılaştırıcı ve bir seçici bloğu içerir. Aşağıda bir ACS bir bloğu gösterilmiştir.



Şekil 19. ACS ünitesi

Son birim Traceback Unit (Geriye Dönük İzleme) birimidir. Bu birim geride kalan yolun ve çıktı verilerinin tanımlandığı geri izleme birimidir. Viterbi kod çözme akış şeması aşağıdadır:



Şekil 20. Viterbi Algoritması

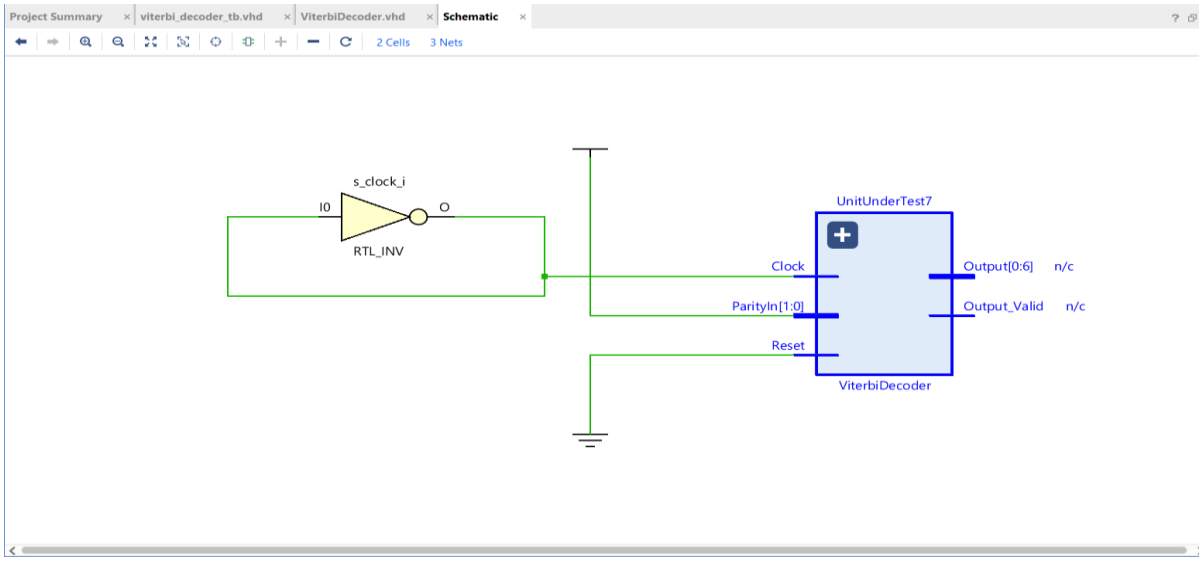
4.1.2.2.1. Viterbi Dekoderi Çalışma Prensibi

Viterbi kod çözücümüzde “generic” VHDL anahtar kelimesiyle koda modülerlik kazandırıyoruz. Bu anahtar kelime tanımlanmış değişkenlerimiz zaman penceremiz, kısıtlama uzunluk değeri(K), toplam yol, hata var mı belirteci ve hata bitinin pozisyonu değerlerini kodumuza giriş olarak veriyoruz. Bu aşamadan sonra 8 bitlik durum geçiş diyagramını ‘type’(tip) anahtar kelimemizle tanımlıyoruz ve bu değerde başlangıç durumu olan boşta(idle) ile ilk duruma atamasını yapıyoruz. Daha sonra Viterbi içindeki tüm yolları kodumuza tanıtan bu tanıttığımız yolları(path) koda bir ROM bellek oluşturarak yüklüyoruz ve bu değerleri ‘constant’ anahtar kelimesiyle tanımlıyoruz, constant anahtar kelimesi ile bu değerler değiştirilemez hale getirilir. Sırada ‘Hamming Distance’ (Hamming mesafesi) hesabını yapmamız gerekmektedir. Bunun için bir fonksiyon tanımlayıp iki bitlik iki veri tanımlanacak ilk iki bit alıcıdan gelen verileri ikinci iki bit, tüm olası yolların gelen bu veriye göre farkını almamıza sağlayan veridir. Burada ise bir döngü içinde koşul yardımıyla x(i) /= y(i) ifadesi ile sayaç değişkeni tanımlayıp farklı olduğunda bu değerimizi artırmamıza olanak tanır. Girişleri 2 bitlik veri olan bu fonksiyonumuz çıkışta tamsayı(integer) tipinde bir değer döndürür. Bu değer bizim Hamming mesafesi dediğimiz değerdir. Sonra en küçük yol birimi fonksiyonu tanımlanır. En küçük yol fonksiyonundan sonra yol metrik hesaplama fonksiyonumda Hamming mesafesi değerimizi çıkış değerimizle toplayıp çıkış sonucumuzu güncelliyoruz. Durum diyagramı olarak tanımladığımız 8 farklı durumun sonlu durum diyagramını (Finite State Machine – FSM) tanımlıyoruz. En sonda ise kodlayıcıyla oluşturduğumuz evrişimli kodların giriş değerlerini kod çözücü ile çıkış değerleri olarak tanımlıyoruz. Örnek kodumuzda 1011000 değerimizin kodlayıcı(encoder) ve decoder (kod çözücü) sonuçları birbiri ile uyumlu olup hata tespit edilmemiştir.

4.1.2.2.2. Viterbi Dekoderinin VHDL çıktısı

Yaptığımız çalışmada her bir üniteyi ayrı ayrı kodlayıp tek bir üst blok yapısında birleştirerek tek bir rtl blok elde ettik. Elde ettiğimiz bu yapı fazlasıyla hafıza birimi içeriyor, bu hafıza da yer kaplasa da yapısal olarak işlem yoğunluğunun azalmasını, yapılan işin hızlanmasını ve güç tasarrufunu sağlamamaktadır. Aşağıda rtl yapı

gözlemlenmektedir.

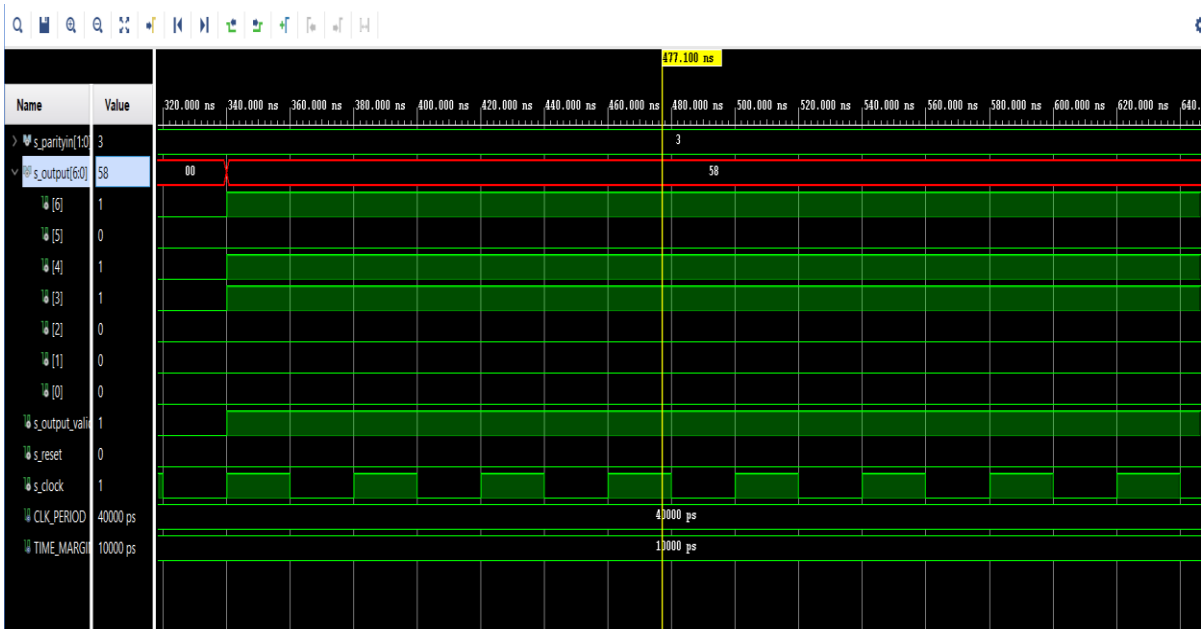


Şekil 21.Viterbi RTL yapı

Bu yapının iç dağılımı ekler bölümündedir.

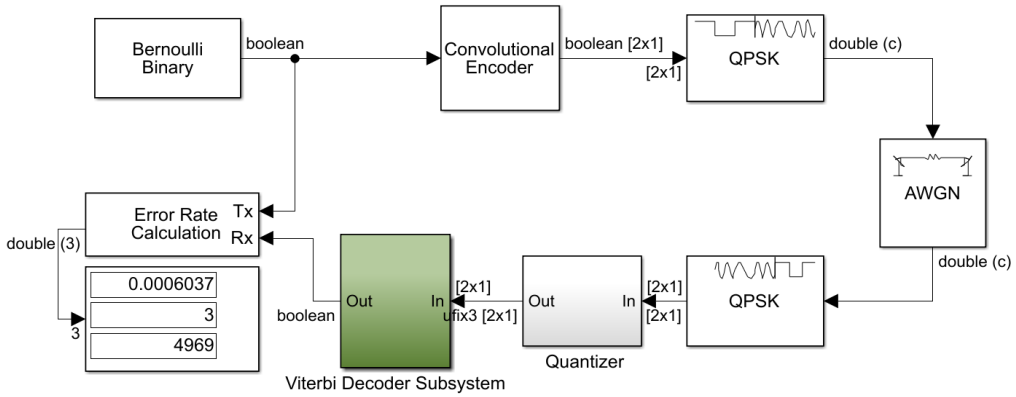
4.1.3. Viterbi Dekoderinin Test Süreci

Viterbi dekoderini, Vivado programında evrişimli kodlayıcı ile birleştirdikten sonra test için VHDL diline özgü testbench kodu yazılarak ilk etapta 10bitlik bir veride sistemin çalışıp çalışmadığına bakıldı. Bu sürecin sonucunda simülasyon çıktısında hata gözlemlenmedi. Gözlem sonuçları aşağıdaki gibidir.



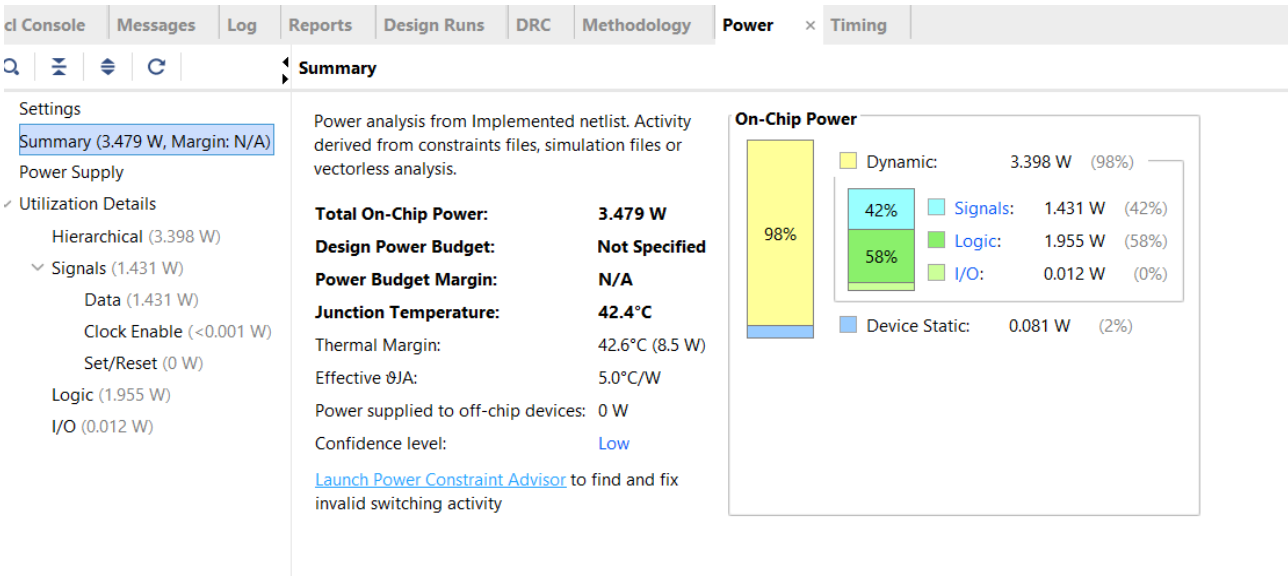
Şekil 22.Dekoder Simülasyon Çıktısı

Yaptığımız çalışmada sonuçları daha kapsamlı gözlemlemek için çoklu data girişi yapılıp 5000 veri üzerindeki hata payı gözlemlendi ve gözlem sonucu Matlab programında BER çıktısı olarak elde edildi.



Şekil 23.Simulink BER sonucu

Vivado da yaptığımız rtl yapıların tamamını derledikten sonra güç tüketim analizi yaptık ve analiz sonuçlarında çok az enerji harcadığımızı ve çok düşük sıcaklıklarda çalıştığımız gözlemledik.



Şekil 24. Güç ve Sıcaklık Raporu

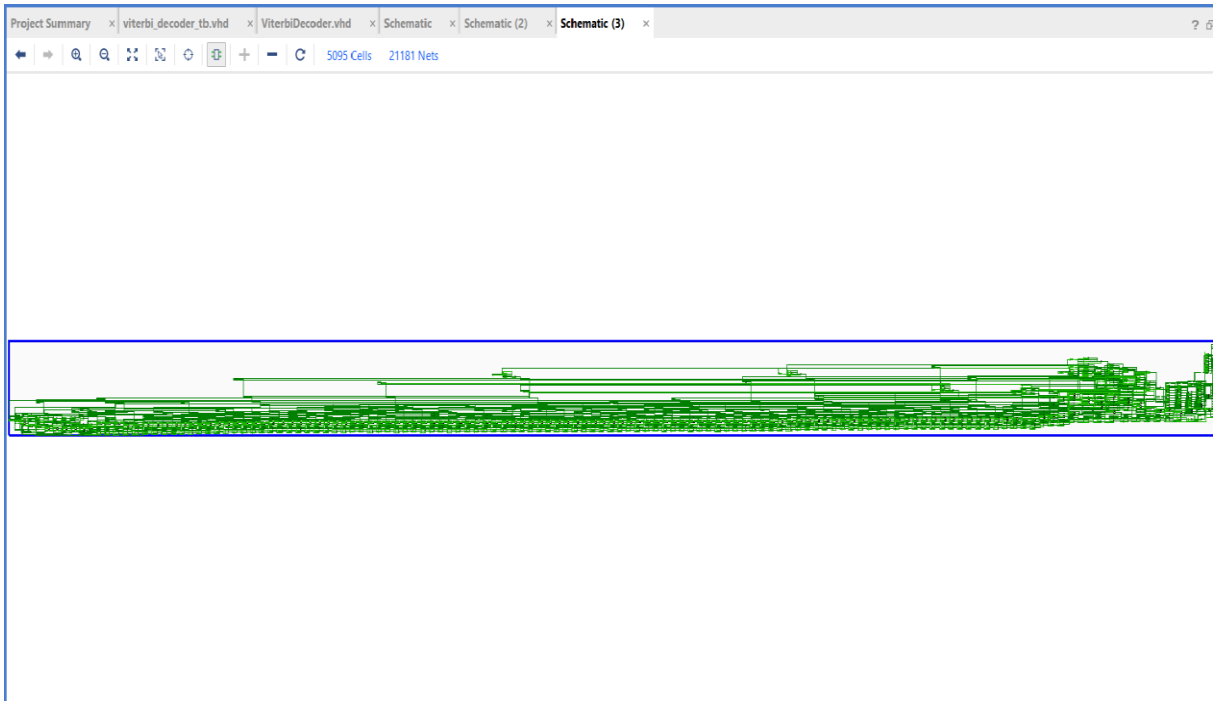
5. TARTIŞMA

Yaptığımız çalışma sonucunda elde ettiğimiz verileri karşılaştırdığımızda kanal kodlaması olmadan yapılan iletişimlerde veri kaybını gürültüyle orantılı olarak fazla olduğu ama kanal kodlaması sayesinde gelen verinin belli metrikler eşliğinde tekrardan işlenip gürültüden arındırılması sayesinde veri kaybının inanılmaz azaldığı gözlemlenmiştir. Yaptığımız çalışmada FPGA tercihinin hız ve güç tüketimi anlamında elimizi rahatlatsa da maliyet anlamında pahalı oluşu bizi zorlamaktadır. Yüksek frekanslı iletişimin ve aşırı yoğunluğun olduğu haberleşme ağlarının olduğu ortamlarda FPGA gerçekten önemli olup maliyeti önemsenmemektir.

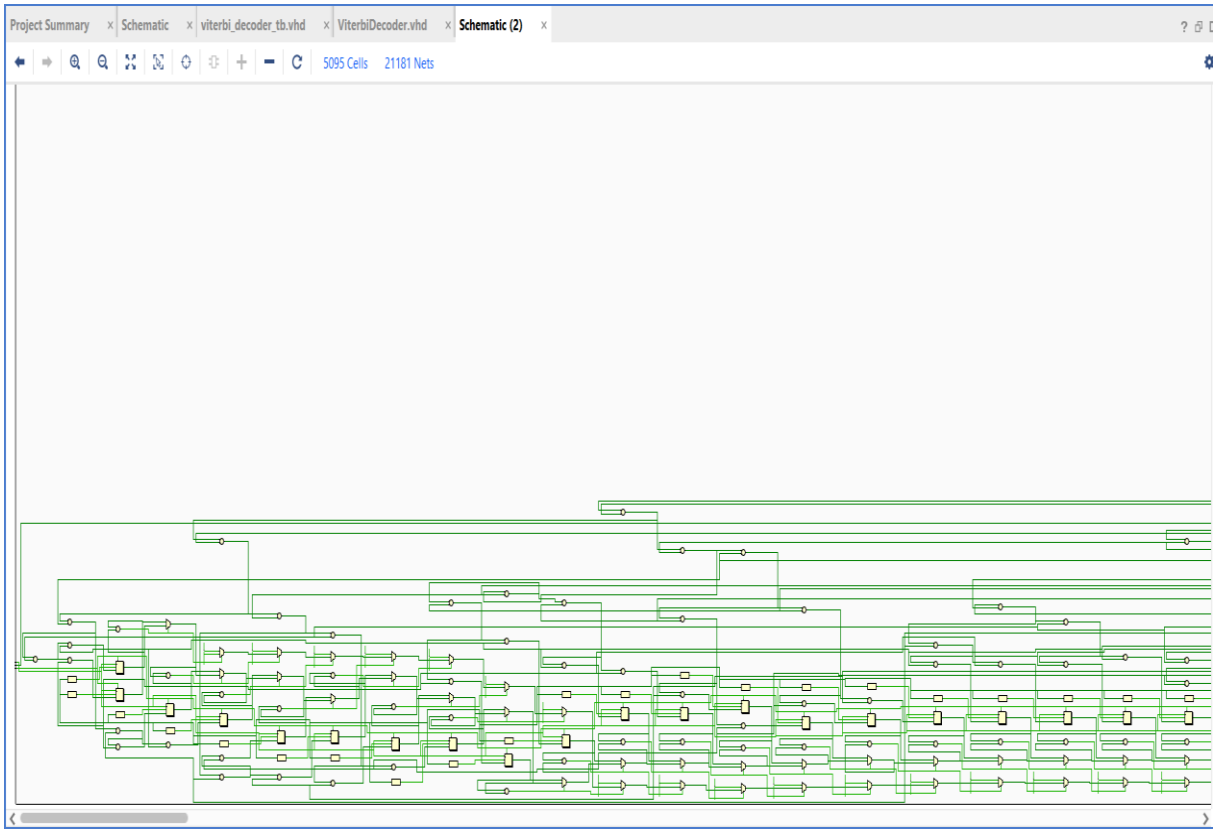
Bu çalışmada yukarıda belirtilen sebepler dışında kazanılan deneyim ve üstüne konularak kompleks kanal tasarımlarına girile bilinir. Günümüzde artık işlemcilerin gigahertz mertebelerine çıkmış olmaları haberleşmede de frekans ölçeğinin yükseltilmesi ve daha dar band genişliklerinin elde edilmesi içinde FPGA kullanımı önem arz etmektedir. Bazı sanayilerde özgün çalışmalar önemli olup bu platformlarda hazır mikroişlemciler yerine FPGA tabanlı entegre sistemlerin kullanımı çok yaygındır. Savunma ve Uzay sanayide FPGA entegrelerin yaygın olarak kullanıldığı bilinmektedir. Bu da yaptığımız çalışmaların ileride bize geri dönüşlerinin iyi olacağı anlamına gelmektedir.

6.EKLER

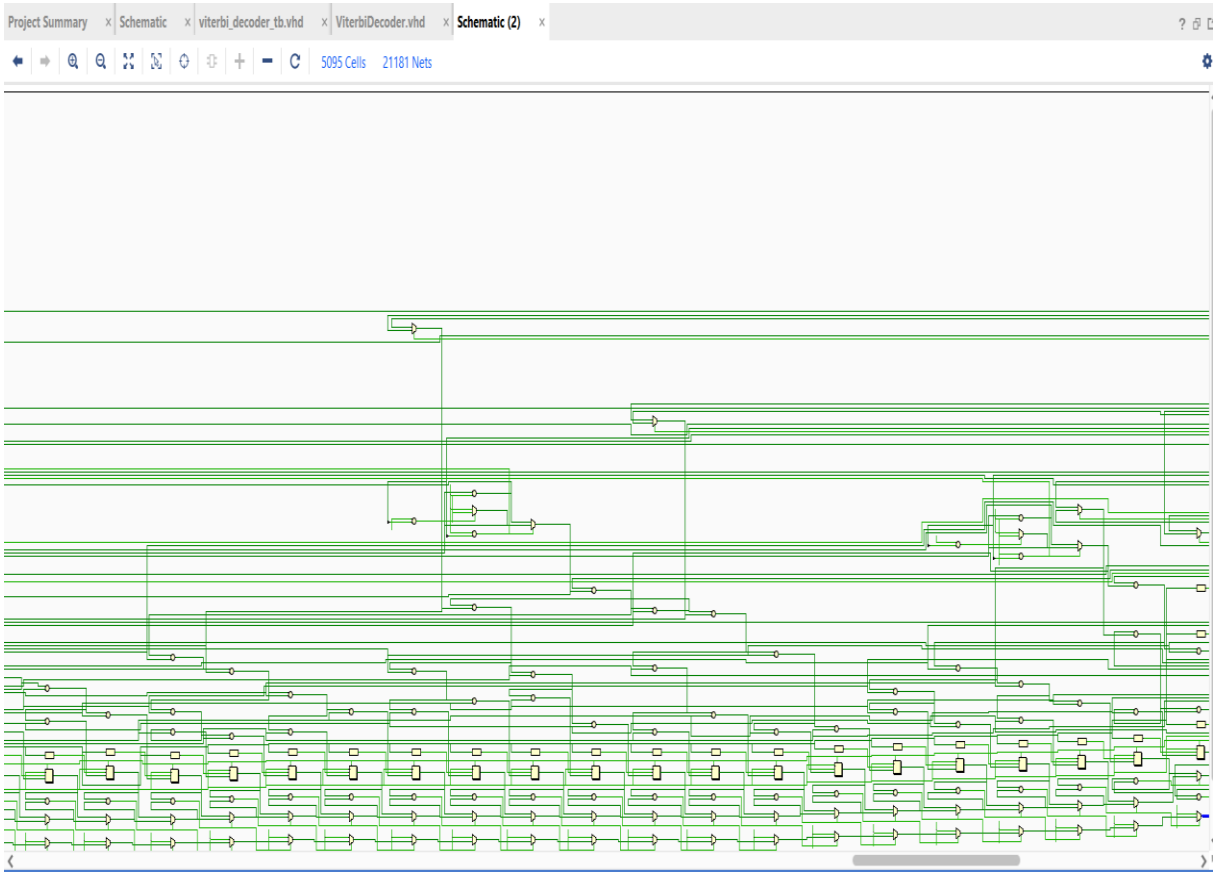
6.1 EK1



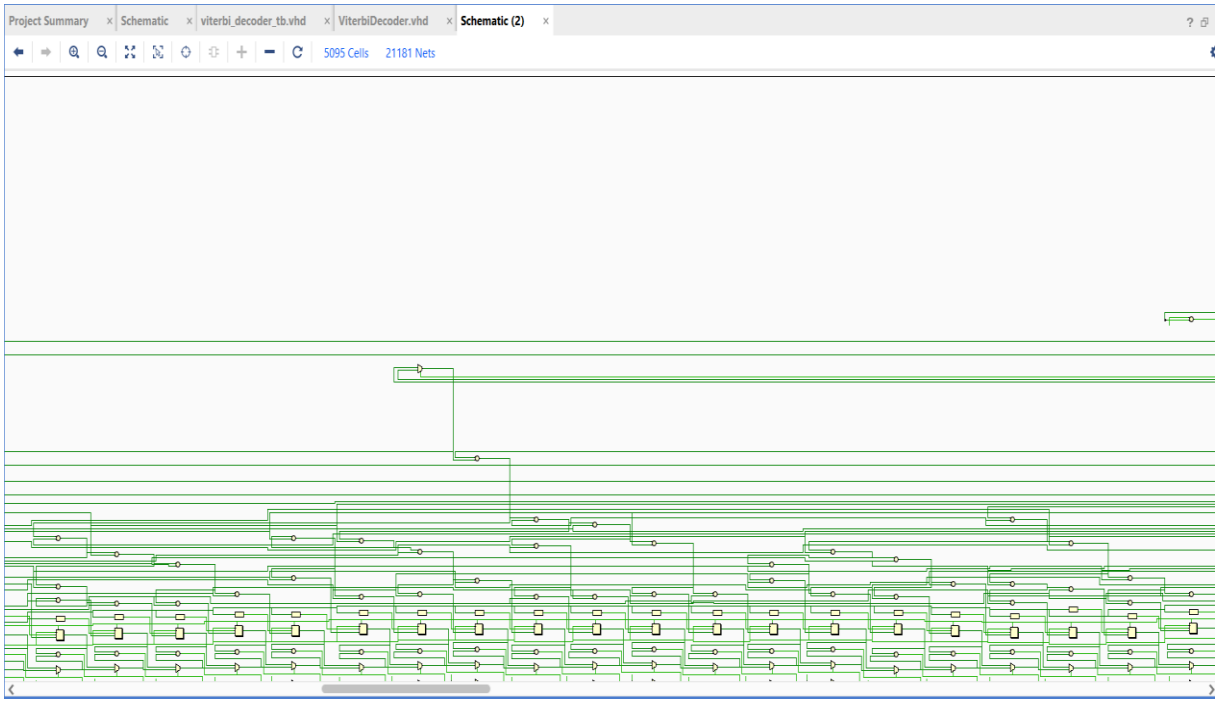
6.1 EK2



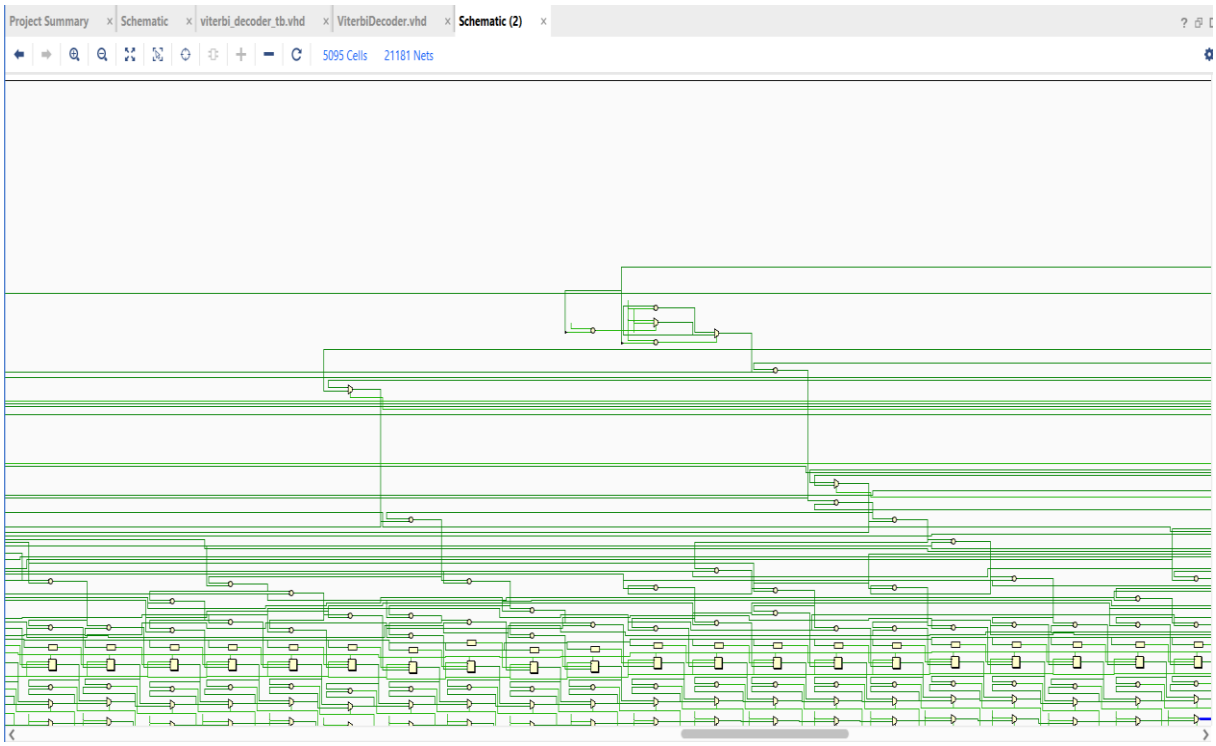
6.1 EK3


Project Summary
Schematic
viterbi_decoder_tb.vhd
ViterbiDecoder.vhd
Schematic (2)
5095 Cells
21181 Nets

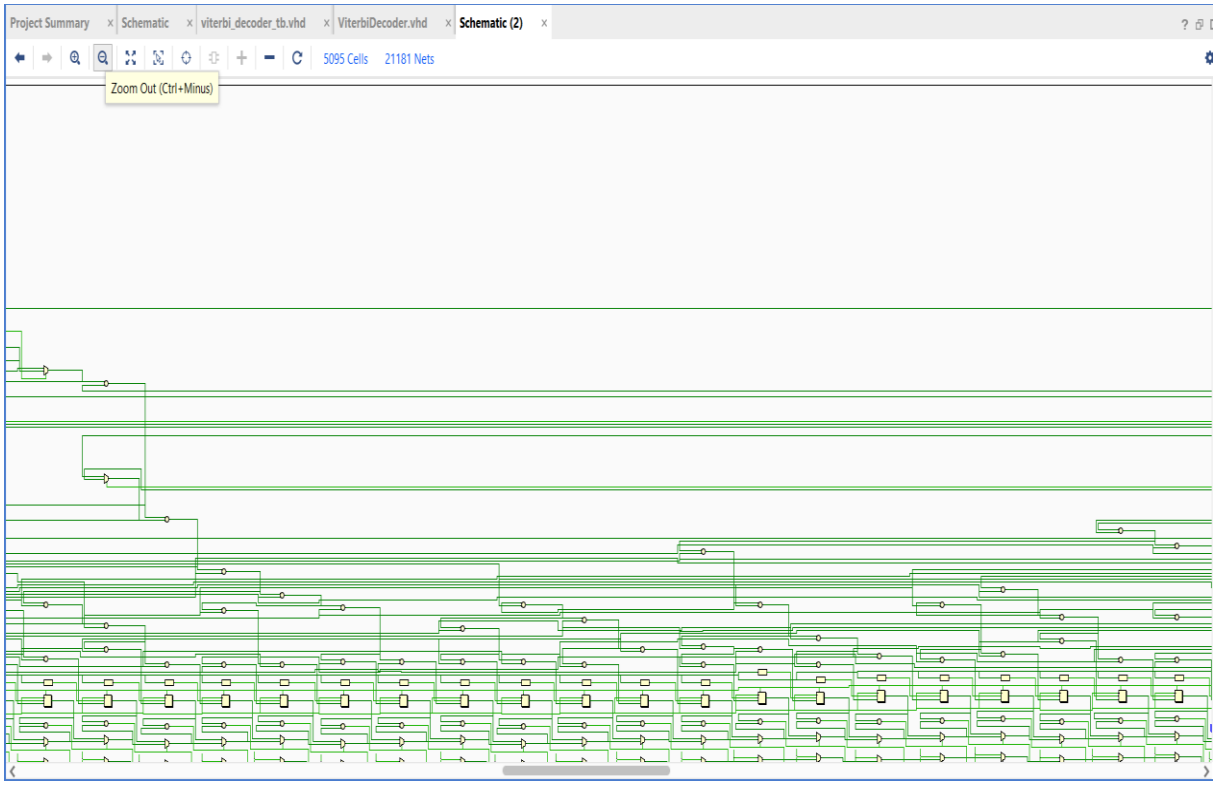
6.1 EK4


Project Summary
Schematic
viterbi_decoder_tb.vhd
ViterbiDecoder.vhd
Schematic (2)
5095 Cells
21181 Nets

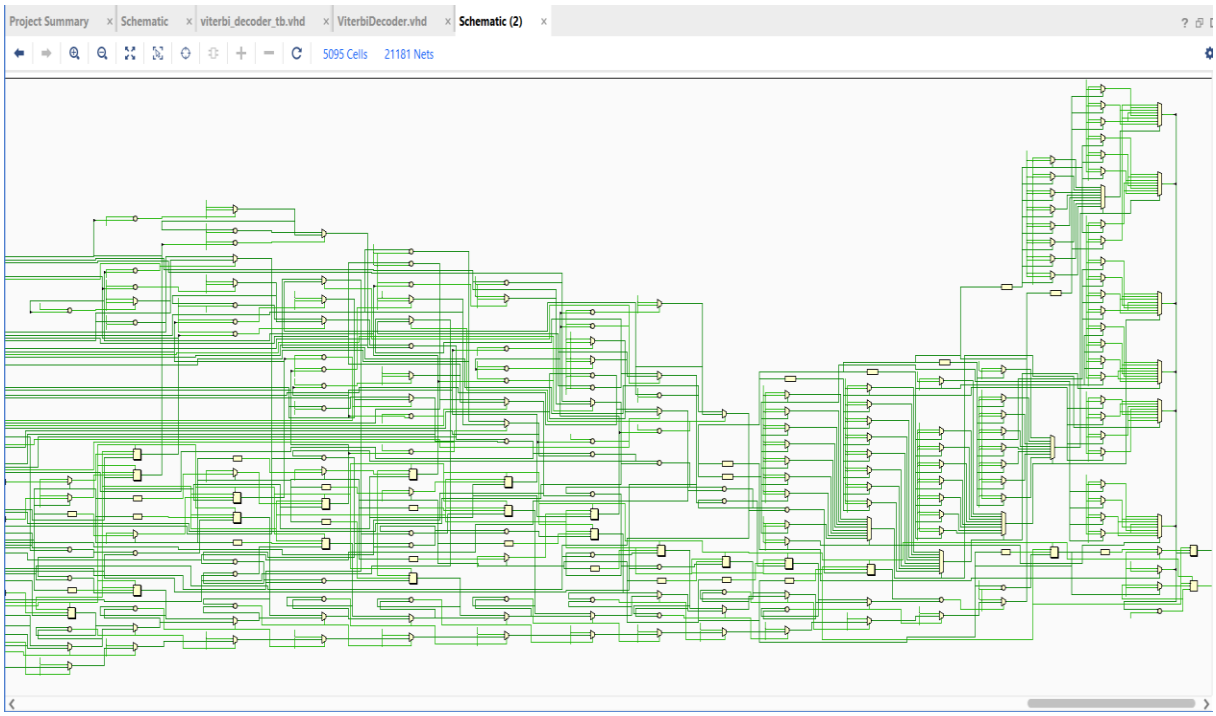
6.1 EK5


Project Summary
Schematic
viterbi_decoder_tb.vhd
ViterbiDecoder.vhd
Schematic (2)
5095 Cells
21181 Nets

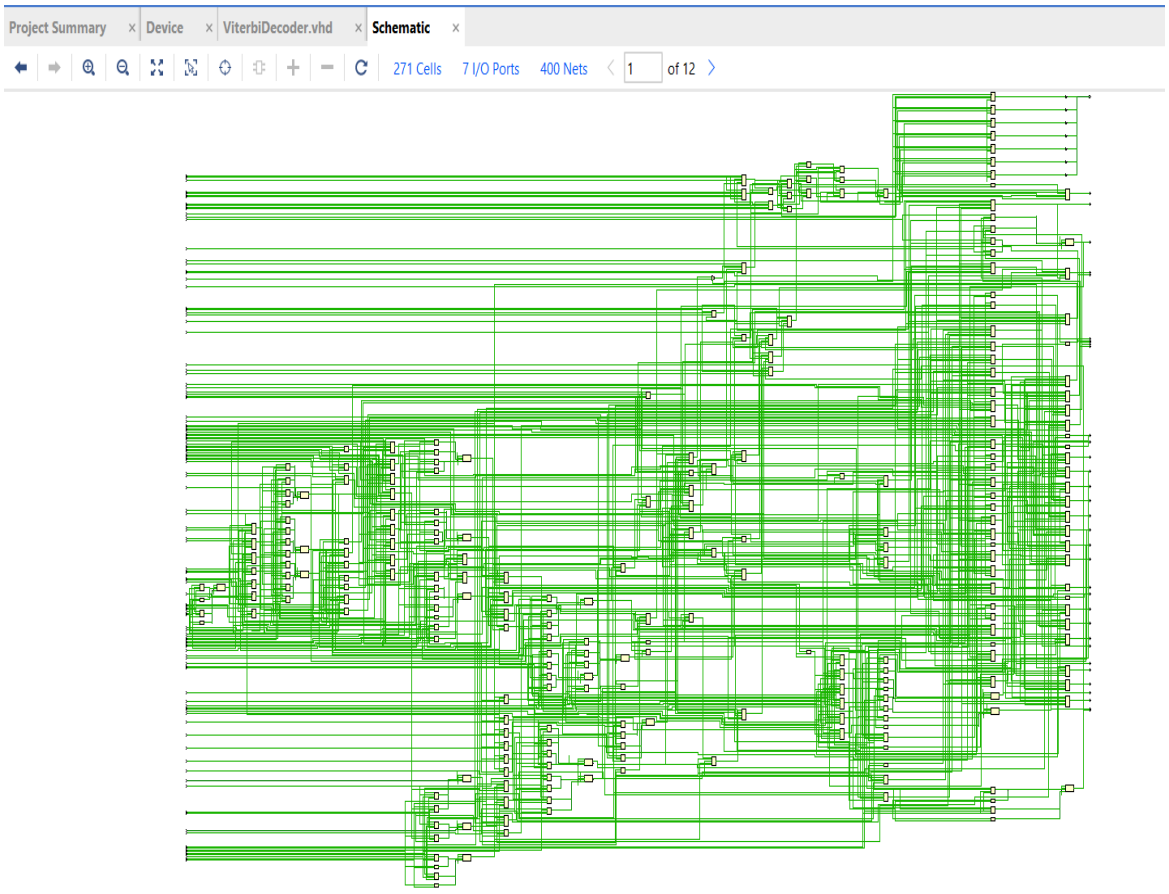
6.1 EK6


Project Summary
Schematic
viterbi_decoder_tb.vhd
ViterbiDecoder.vhd
Schematic (2)
5095 Cells
21181 Nets
Zoom Out (Ctrl+Minus)

6.1 EK7


Project Summary
Schematic
viterbi_decoder_tb.vhd
ViterbiDecoder.vhd
Schematic (2)
5095 Cells
21181 Nets

6.1 EK8


Project Summary
Device
ViterbiDecoder.vhd
Schematic
271 Cells
7 I/O Ports
400 Nets
1
of 12

7.KAYNAKLAR

Gupta S., Mehra R. “Reconfigurable Efficient Design of Viterbi Decoder for Wireless Communications Systems” International Journal of Advanced Computer Science and Applications, Vol 2, No 7, 2011

Shaker S. W., Elramly S. H., Shehata K. A. “FPGA implementation of a reconfigurable Viterbi decoder for WiMAX receiver” Proceedings of the International Conference on Microelectronics, ICM 2009

Kalita S., Gogoi P., Sarma K. K. “Convolutional coding using booth algorithm for application in wireless communication” Department of Electronics and Communication Technology, Gauhati University, Guwahati-781014

Proakis G. J., Salehi M., 2020, “Sayısal Haberleşme” Palme Yayınevi, ANKARA

Seke E. 2017,“VHDL Örnekleriyle Sayısal Haberleşmeye Giriş: Kavram-Teori-Uygulama-Sonuç” Seçkin Yayıncılık, ANKARA

Özbay, B. (2017). Viterbi kod çözücünün güç etkin mimari tasarımı ve FPGA gerçeklemesi (Master's thesis, Maltepe Üniversitesi, Fen Bilimleri Enstitüsü).

Adam O., Shengli Fu, Varanasi M, “Hardware Efficient Encryption Encoder and Decoder Unit”, Military Communication Conference, IEEE, 16-19 Nov. 2008, pp. 1 –6

Yeu-Horng Shiau, Pei-Yin Chen, Hung-Yu Yang, Yi-Ming Lin, Shi-Gi Huang, “An efficient VLSI architecture for convolutional code decoding”, International Symposium on Next Generation Electronics, 2010, pp. 223-226

”FPGA Design and Implementation of a Low-Power Systolic Array-Based Adaptive Viterbi Decoder”,IEEE Transactions on circuits and systemsi: regular papers, vol. 52, no. 2, February 2005.

”A Parallel Viterbi Decoder for Block Cyclic and Convolution Codes”, Department of Electronics and Computer Science, University of Southampton. April 2006

”A Dynamically Reconfigurable Adaptive Viterbi Decoder”, International Symposium on FPGA 2002.

URL 1. https://en.wikipedia.org/wiki/Viterbi_decoder (2020)

URL2 https://www.xilinx.com/products/intellectual-property/viterbi_decoder.html(2020

URL 3. https://www.mathworks.com/help/comm/ref/viterbidecoder.html (2020)

8.TEŞEKKÜR

Bu çalışmada teorik haberleşme kısmında bize yardımcı olan ve projemizde bize desteğini esirgemeyen Prof. Dr. Tuncay ERTAŞ öğretmenimize,

Çalışmamızda bize kitleri öğrenci olduğumuz için indirimli gönderen Digilent firmasına,

Kiti bize hızlıca ulaştıran Robotistan firmasına,

Udemy ’de verdiği kaliteli eğitimden dolayı Can Dost YAVUZ a

Youtube’daki içerikleriyle bize sistem programlama mantığında yardımcı olan basta Serdar DERELİ olmak üzere metin akkın ve kaya ya,

Değerli kitabıyla Doç. Dr. Burak KELLECİ öğretmenimize,

VHDL ile sayısal haberleşmeyi kitaplaştırdığı için Yar. Doç. Dr. Erol SEKE

Çok teşekkür ederiz

9.ÖZGEÇMİŞ

9.1. Ahmed Melih Ulusoy

1995 yılında Malatya'da dünyaya gelmiştir. İlk ve orta öğrenimini Malatya Mehmet Emin Bitlis İlköğretim okulunda tamamlamış. Liseyi Kernek Anadolu Lisesinde okumuştur. 2015 yılında Uludağ Üniversitesi Elektrik Elektronik Mühendisliğini kazanmıştır. Şu an 4. sınıf öğrencisi olarak aktif eğitim hayatına devam etmektedir.

9.2. Nuh Nar

1995 yılında Sarıkamış'ta dünyaya gelmiştir. İlk ve orta öğrenimini Erzurum'da tamamlamıştır. Liseyi Tevfik İleri Anadolu Lisesinde okumuştur.2011 yılında Curlingde milli sporcu olmuştur. Bu alanda Türkiye ve Avrupa'da çeşitli başarılar elde etmiştir.2015 yılında Uludağ Üniversitesi Elektrik Elektronik Mühendisliğini kazanmıştır. Şu an 4. sınıf öğrencisi olarak aktif eğitim hayatına devam etmektedir. Uludağ Üniversite Makina topluluğunda 2016-2017 yılında başkan yardımcılığı yaptı. Alternatif enerjili araç yarışlarında TÜBİTAK'ta teknik tasarım ödülü almıştır. 2017 yılında Shell-Eco Marathon yarışmasına katılarak Avrupa üniversitemizi temsil etmiştir. Bu süreçten sonra sanayide karavan firmasında elektrik kablo tasarımı üzerine 1 yıl çalıştım. Şu an Hezarfen space grupta elektronik donanım tasarımcısı olarak çalışmaktadır.