SAYISAL İŞARET İŞLEMCİ İLE V.32 MODEM TASARIMI

YÜKSEK LİSANS TEZİ
Müh. Mustafa ISNIK
504961405

Tezin Enstitüye Verildiği Tarih : 22 Ocak 2001
Tezin Savunulduğu Tarih : 5 Şubat 2001

Tez Danışmanı : Doç. Dr. M. Ertuğrul ÇELEBİ
Diğer Jüri Üyeleri : Doç. Dr. Işıl CELASUN
Doç. Dr. Osman Nuri UÇAN (İ.Ü.)

ŞUBAT 2001
ÖNSÖZ

Bana bu konu üzerinde çalışma fırsatı tanıyan ve engin tecrübesinden yararlandığım tez danışmanım sayın Doç. Dr. Ertuğrul ÇELEBİ’ ye, Prof. Dr. Ümit AYGÖLÜ’ ye, her zaman olduğu gibi tez çalışması esnasında da bana destek olan aileme ve dedeme, çalışmalarımında teknik destek aldığım Adam Elektronik Ltd. Şti.’ ne, görüş ve düşüncelerine ihtiyaç duyduğumda fikirleriyle beni yönlendiren Prof. Dr. Macit GÜNEŞ’ e sonsuz teşekkürlerimi sunarım.

Şubat 2001

Mustafa ISNIK
**İÇİNDEKİLER**

**KISALTMALAR**  vi  
**TABLO LİSTESİ**  vii  
**ŞEKİL LİSTESİ**  viii  
**ÖZET**  x  
**SUMMARY**  xi  

1. **GİRİŞ**  

2. **MODEM HABERLEŞME TEMELLERİ**  
   2.1. Temel Kavramlar  
      2.1.1. Giriş  
      2.1.2. Haberleşme Yapıları  
         2.1.2.1. Basit Haberleşme  
         2.1.2.2. Yarı Çift Yönlü Haberleşme  
         2.1.2.3. Tam Çift Yönlü Haberleşme  
      2.1.3. Protokol  
         2.1.3.1. Bilginin Kodlanması  
         2.1.3.2. Kontrol  
         2.1.3.3. Hata Algılama ve Düzeltilme  
         2.1.3.4. Haberleşme Kanal Verimliliği  
         2.1.3.5. Eşzamanlı Çalışma  
         2.1.3.6. Saydamlık  
      2.2. Modem Donanım Özellikleri  
      2.3. Modem Bağlantı Hatlarının İncelemesi  
         2.3.1. Ortalama Bant Genişliği  
         2.3.2. Modülasyonlu İşaret Frekans Genlik Değişimi  
         2.3.3. Gürültü  
         2.3.4. Faz Gecikmesi  
      2.4. CCITT V.24 Özellikleri  
      2.5. Eşzamanlısız ve Eşzamanlı Haberleşme Yapıları  
         2.5.1. Eşzamanlısız Veri Yapısı  
         2.5.2. Eşzamanlı Veri Yapısı  
         2.5.2.1. LRC  
         2.5.2.2. VRC
5.4.3. Temel Bant İşareti Oluşturma Programı 53
5.5. Süzme ve Modülasyon İşlemi 54
  5.5.1. Süzme ve Modülasyon Özellikleri 54
  5.5.2. Temel Bant ve Modülasyonlu İşaret Analizi 56
  5.5.3. Süzgeç ve Modulatör Program Tasarımı 58
    5.5.3.1. Program Tasarım Yöntemi 58
    5.5.3.2. Temel Bant İşaret Süzgeç ve Modülasyon Programı 59
    5.5.3.3. Modülasyonlu İşaret Süzme Programı 61
5.6. Sonuç 62

6. V.32 DEMODÜLATÖR TASARIMI 63
  6.1. Giriş 63
  6.2. V.32 Demodülatör Yapısına Genel Bakış 64
  6.3. Hilbert Dönüşümü ve Kompleks Demodülatör 66
    6.3.1. Hilbert Dönüşüm Yöntemiyle Demodülasyon 66
    6.3.2. Hilbert Dönüştürücü 68
    6.3.3. Hilbert Dönüştürücü ve Demodülatör Tasarımı 70
  6.4. Uyarılamalı Dengeleyici ve Açısal Hata Hesaplama 71
    6.4.1. Uyarılamalı Dengeleyici Özellikleri 71
    6.4.2. Uyarılamalı Dengeleyici Tasarımı 74
    6.4.3. Açısal Hata Hesaplama ve Döndürme İşlemi 75
  6.5. Karar Verici 77
    6.5.1. Katılamalı Kodlar İçin En Büyük Benzerlikli Kod Çözme (Viterbi Algoritması) 77
    6.5.2. Viterbi Çözücü Tasarımı 79
    6.5.3. Viterbi Çözücü Performansı ve Hata Başarımı 80
  6.6. Ters Farksal Dönüşürme ve Çözme 82
    6.6.1. Ters Farksal Dönüşüm 82
    6.6.2. Çözücü (Descrambler) 83
      6.6.2.1. Çözücü Özellikleri 83
      6.6.2.2. Çözücü Tasarımı 84
  6.7. Sonuç 85

7. SONUÇLAR VE ÖNERİLER 86

KAYNAKLAR 88

ÖZGEÇMİŞ 90
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSK</td>
<td>Phase Shift Keying</td>
</tr>
<tr>
<td>FSK</td>
<td>Frequency Shift Keying</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration Systems</td>
</tr>
<tr>
<td>POS</td>
<td>Point of Sale System</td>
</tr>
<tr>
<td>SCADA</td>
<td>Supervisory Control and Data Acquisition System</td>
</tr>
<tr>
<td>SER</td>
<td>Symbol Error Rate</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal Noise Ratio</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>WTSC</td>
<td>The World Telecommunication Standardization Conference</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledge</td>
</tr>
<tr>
<td>NACK</td>
<td>Not Acknowledge</td>
</tr>
<tr>
<td>MAFE</td>
<td>Modem Analog Front End</td>
</tr>
<tr>
<td>DTE</td>
<td>Data Terminal Equipment</td>
</tr>
<tr>
<td>DCE</td>
<td>Data Communication Equipment</td>
</tr>
<tr>
<td>LRC</td>
<td>Longitudinal Redundancy Check</td>
</tr>
<tr>
<td>VRC</td>
<td>Vertical Redundancy Check</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>DSK</td>
<td>DSP Starter Kit</td>
</tr>
<tr>
<td>Sayfa No</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td></td>
</tr>
<tr>
<td>Tablo 2.1. CCITT ve Bell Standartları</td>
<td>19</td>
</tr>
<tr>
<td>Tablo 3.1. 4800 bit/saniye için farklı kodlama ve 9600 bit/saniye için normal kodlama</td>
<td>29</td>
</tr>
<tr>
<td>Tablo 3.2. 9600 bit/saniye hızında kafes kodlaması için farklı kodlayıcı tablosu</td>
<td>30</td>
</tr>
<tr>
<td>Tablo 3.3. 9600 bit/saniye hızında iki alternatifli işaret durum tablosu</td>
<td>33</td>
</tr>
<tr>
<td>Tablo 5.1. V.32 Karıştırıcı Programı</td>
<td>48</td>
</tr>
<tr>
<td>Tablo 5.2. V.32 Kodlayıcı Programı</td>
<td>52</td>
</tr>
<tr>
<td>Tablo 5.3. Temel Bant İşaret Oluşturma Programı</td>
<td>53</td>
</tr>
<tr>
<td>Tablo 5.4. Örneklenmiş Yükseltilmiş Kosinüs Sütçe Değerleri</td>
<td>59</td>
</tr>
<tr>
<td>Tablo 5.5. Örneklenmiş Sinüs ve Kosinüs İşaret Değerleri</td>
<td>59</td>
</tr>
<tr>
<td>Tablo 5.6. Temel Bant İşaret Süzme Programı</td>
<td>60</td>
</tr>
<tr>
<td>Tablo 5.7. Modülasyon Programı</td>
<td>60</td>
</tr>
<tr>
<td>Tablo 5.8. Butterworth Sütçe Programı</td>
<td>61</td>
</tr>
<tr>
<td>Tablo 6.1. Hilbert Dönüştürücü Programı</td>
<td>70</td>
</tr>
<tr>
<td>Tablo 6.2. Kompleks Demodülatör Programı</td>
<td>71</td>
</tr>
<tr>
<td>Tablo 6.3. Uyarılamalı Dengeleyici Programı</td>
<td>74</td>
</tr>
<tr>
<td>Tablo 6.4. Uyarılamalı Dengeleyici Katsayı Güncellemesi Programı</td>
<td>74</td>
</tr>
<tr>
<td>Tablo 6.5. Ters Vektör Dönüştürme Programı</td>
<td>75</td>
</tr>
<tr>
<td>Tablo 6.6. Açısal Hata Hesaplama Programı</td>
<td>76</td>
</tr>
<tr>
<td>Tablo 6.7. Vektör Dönüştürme Programı</td>
<td>76</td>
</tr>
<tr>
<td>Tablo 6.8. Gauss Gürültülü Ürete Programı</td>
<td>81</td>
</tr>
<tr>
<td>Tablo 6.9. Test Amacı Kullanılan Sayısal Veri Dizisi</td>
<td>82</td>
</tr>
<tr>
<td>Tablo 6.10. Ters Farksal Dönüşüm Programı</td>
<td>83</td>
</tr>
<tr>
<td>Tablo 6.11. V.32 Çözücü Programı</td>
<td>84</td>
</tr>
</tbody>
</table>
ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1 : Basit Haberleşme ................................................................. 5
Şekil 2.2 : Yarı Çift Yönlü Haberleşme .................................................. 6
Şekil 2.3 : Tam Çift Yönlü Haberleşme .................................................. 6
Şekil 2.4 : Örnek Veri Yapıları .............................................................. 7
Şekil 2.5 : Genel Çerçeve Yapıları .......................................................... 7
Şekil 2.6 : Örnek Hata Düzeltme Algoritması ......................................... 8
Şekil 2.7 : Tipik Hat Erişim Devresi (Dial-Up) ....................................... 10
Şekil 2.8 : Modem Blok Diyagramı ......................................................... 11
Şekil 2.9 : Tipik Hibrit ve MAFe Yapıtı ................................................ 11
Şekil 2.10 : Haberleşme Kanalı Bant Genişliği ...................................... 13
Şekil 2.11 : G.132 Frekans – Genlik Karakteristiği (Attenuation Distortion) 13
Şekil 2.12 : Frekans – Gecikme Karakteristiği ...................................... 14
Şekil 2.13 : Modem Haberleşme Yapıtı .................................................. 14
Şekil 2.14 : DTE – DCE Arası Bağlantı Yapıtı .................................... 15
Şekil 2.15 : V.24 Elektriksel Seviyeler .................................................. 15
Şekil 2.16 : Eşzamanlı Çerçeve Yapıtı .................................................. 16
Şekil 2.17 : Eşzamanlı Çerçeve Yapıtı .................................................. 17
Şekil 2.18 : LRC Oluşturma Yapıtı ....................................................... 18
Şekil 2.19 : R=n/m Katlamalı Kodlama Yıktısı .................................... 20
Şekil 2.20 : Katlamalı Kodlama Modülatör .......................................... 23
Şekil 2.21 : 32 – Nokta İşaret Diyagramındaki İşaret Kümelerinin Ayrıntılıması ............................................................ 24
Şekil 2.22 : Katlamalı Kodlama Küme Durum Geçiş Diyagramı .............. 25
Şekil 3.1 : 9600 bit/saniye için normal kodlama 16-nokta işaret durum diyagramı ve 4800 bit/saniyede öğrenme fazında kullanılan A, B, C ve D durumları................................................................. 29
Şekil 3.2 : 9600 bit/saniye Hızında Kafes Kodlama .................................. 31
Şekil 3.3 : 9600 bit/saniye Hızında Kafes Kodlama ile 32-nokta işaret durum diyagramı ve 4800 bit/saniyede öğrenme fazında kullanılan A, B, C ve D durumları................................................................. 32
Şekil 4.1 : TMS320C5420 Sayısal İşaret İşlemci Blok Diyagramı .......... 37
Şekil 4.2 : TMS320C5420 Tabanlı DSK Yazılım Geliştirme Kiti Donanım Blok Diyagramı ................................................................. 40
Şekil 5.1 : V.32 Modülatör Blok Diyagramı .......................................... 44
Şekil 5.2 : Modülatör Programı Genel Akış Şemasi ................................ 46
Şekil 5.3 : Karşıştırıcı Kat Blok Diyagramları ...................................... 47
Şekil 5.4 : V.32 Kafes Kodlayıcı Blok Diyagramı .................................. 49
Şekil 5.5 : V.32 Katlamalı Kodlayıcı Sonlu Durum Diyagramı .............. 51
Şekil 5.6 : V.32 Kompleks Modülatör ...................................................... 54
Şekil 5.7 : V.32 Modülatör İşaret Frekans Spektrumları ...................... 56
Şekil 5.8 : Yükseltilmiş Kosinüs Süzgeç Zaman Cevabı........................................ 57
Şekil 5.9 : 14400Hz Örnekleme Hızında 4. Dereceden 3500Hz Köşe
Frekanslı Alçak Geçiren Butterworth Süzgeç Blok Diyagramı........ 61
Şekil 6.1 : Hilbert Dönüşümü Demodülatör Katı Blok Diyagramı................. 64
Şekil 6.2 : Demodülatör Programı Genel Açı Şeması.................................... 65
Şekil 6.3 : Hilbert Dönüşümü ve Demodülatör Blok Diyagramı....................... 66
Şekil 6.4 : Hilbert Dönüştürücü Darbe, Genlik ve Faz Eşikleri..................... 70
Şekil 6.5 : Kompleks Dengeleyici Blok Diyagramı........................................ 72
Şekil 6.6 : Taşıyıcı Çıkarma Filtresi Blok Diyagramı.................................... 75
Şekil 6.7 : 8 Durumlu TCM Durum Geçişleri............................................. 78
Şekil 6.8 : V.32 TCM 32 – nokta İşaret Küme Örnekleri............................... 79
Şekil 6.9 : 32 – nokta İşaret Bölgeleri........................................................ 79
Şekil 6.10 : Hat Kodlamaı Temel Bant İletişim Sistemi............................... 80
Şekil 6.11 : Gauss Gürültülü Kanalda Viterbi Performansı.......................... 82
Şekil 6.12 : Çözücü Katı Blok Diyagramı.................................................... 84
SAYISAL İŞARET İŞLEMCİ İLE V.32 MODEM TASARIMI

ÖZET

Elektronik ve bilgisayar teknolojilerinin yakın dönemdeki hızlı gelişimi, veri iletişiminin öneminin artmıştır. Her insana ulaşabilecek kadar yaygın olan ulusal ve uluslararası telefon şebekesi günümüzde en çok kullanılan iletişim ortamıdır. Sürekli artan hız ve güvenilir iletişim ihtiyacı, telefon şebekesi kullanılarak modemler tarafından karşılanır. Modemler, gelişen tümleşik devre ve sayisal işaret işleme teknolojileri sayesinde daha hızlı ve güvenilir veri iletişimini gerçekleştirmirler. Modemler; endüstriyel ve askeri sistemler başta olmak üzere bir çok iletişim sisteminde, 2400 bit/saniye ve daha düşük hizlarda kullanılarlar.

Bu tez çalışmasında; 9600 bit/saniye hızındaki sayisal verilerin iletişim için modülatör ve demodülatör yapının, kafes kodlama modülasyon tekniği, Hilbert dönüşümü ve Viterbi algoritması ile gerçekleştirilmesi amaçlanmıştır.

Bu çalışmada; değişik sayisal işaret işlemcileri araştırılmıştır. TMS320C5402 sayisal işaret işlemcisi ve program geliştirme kitleri seçilerek yazılım çalışmalarını bu işlemci için program geliştirme kitinde gerçekleştirilmiştir. Yazılım çalışmalarında; telefon şebekesi ve kiralık devreler üzerinde, çift yönlü 9600 bit/saniye veri iletişim hızında tanımlı ITU V.32 standardı referans olarak kullanılmıştır.

Modülasyon işlemi, test dizisi kullanılarak gerçekleştirilmiştir. Test dizisi, karştırılduktan sonra kafes kodlama teknikleri ve kompleks modülatör ile modüle edilmiştir. Böylece örneklemlenmiş analog semboller oluşturulmuştur.

Kafes kodlama tekniğin en önemli özelliği, genişletilmiş işaret kümeleri yardımcı, kodlanmış işaret dizileri arasındaki küçük Ölçül uzaklığı mümkün olabilecek en büyük uzaklık yapabilecek kodlama ve modülasyon işleyenlerinin kullanılmasıdır. Bu şekilde ardışık olarak oluşturuluran temel bant işaretleri arasındaki uzaklık arttırularak kodlama kazancı sağlanır.

Demodülasyon işleminde ise örneklemlenmiş analog semboller Hilbert dönüşümü ve kompleks demodülatör yardımcı bandı indirilmiş ve Viterbi algoritması kullanılarak karar verilen sayisal veri grupları çözülerek test dizisi tekrar oluşturulmuştur.

Ayrıca etkin sayisal işaret işlemci programlama teknikleri araştırılmış ve bu teknikler kullanılarak gerekli olan toplam işlemcı yükü azaltılmıştır. Kullanılan test dizilerinin boyutları ve içerikleri degerşirilerek gerçeklesten modülasyon ve demodülasyon yazılımları test edilmiştir.
DESIGNING V.32 MODEM WITH DIGITAL SIGNAL PROCESSOR

SUMMARY

Recent developments in electronics and computer technologies increased the importance of data communication. Today, still the most widely used telecommunication environment is public switched telephone network that is within reach of almost every person. Increasing fast and reliable communication requirement is fulfilled by modems using telephone network. With the help of VLSI and digital signal processing technologies improvements, modems have become more faster and reliable data communication tools. Modems are used in most of communication systems mainly in industrial and military systems with in usually 2400 bit per second or lower speed.

In this thesis; its aimed to design modulator and demodulator structures for data communication in 9600 bit per second using the techniques of trellis coding modulation, Hilbert transformation and Viterbi algorithm along the ITU-T V.32 recommendation lines. Furthermore, various digital signal processors were investigated. TMS320C5402 digital signal processor and its programming set were chosen and codes were developed for programming this digital signal processor. ITU V.32 standards that are defined on duplex operation with 9600 bit per second data communication speed on telephone network and leased line, are used in developing those.

Modulating signal is a test pattern. The scrambled test pattern is modulated with trellis coding techniques and complex modulator. Thus, sampled analogue symbols are created.

Trellis coding technique’ one of the most important feature includes coding and modulation operations so as to achieve Euclidean distance, that exist between the coded signal patterns, with the help of expanded symbol sets. In this way, coding gain is increased by making the distance higher which is between the base band signal that is created consecutively.

In demodulation operation; sampled analogue symbols are converted to the base band with the help of Hilbert transformation and complex demodulator. Test pattern is created again by descrambling digital data groups which are specified using Viterbi Algorithm.

In addition, effective digital signal processor programming techniques are investigated. Thus, total processor load is reduced by using these techniques. Developed programs about modulation and demodulation are tested by modifying dimensions and contents of used test patterns.
1. GİRİŞ

Yüzyıllımızın son çeyreğinde, elektronik ve bilgisayar teknolojilerinde meydana gelen gelişmeler hızlı ve güvenilir veri iletişim ihtiyacını artırmıştır. Enformasyon çağına giren dünyamızda yüz milyonlarca bilgisayarın iletişim gereksinimi, tüm insanlara ulaşmış ulusal ve uluslararası telefon şebekesi üzerinden modemlerle çözülebilmektedir. Modepler, ilk olarak faz ve frekans modülasyon (PSK ve FSK) tekniklerinin kullanıldığı düşük hizlarda çalışan iletişim cihazları olarak ortaya çıkmışlardır. Günümüze kadar uzanan süreç içerisinde artan hızlı ve güvenilir iletişim ihtiyaçları, geliştirilen VLSI mikro denetleyicileri ve sayısal işaret işleme teknolojilerinin kullanıldığı tümleşik devreler ile yapılan akıllı modem tasarımlarını güvenlik hayatımıza sokmuştur. Modem iletişiminin, daha hızlı ve güvenilir gerçekleştibilmesi amacıyla değişik modülasyon teknikleri geliştirilmekte ve standartlaştırılmaktadır.

Günümüzde; mevcut telefon alt yapısı üzerinde, askeri ve sivil amaçlı her yer ve koşulda, güvenilir ve tercih edilen modem iletişim hızları 9600 bit/saniye’ nin altında kalmaktadır. Örneğin; füze ve radar iletişim sistemleri, POS, trafik sinyalizasyon ve SCADA sistemleri ile endüstriyel uygulamalar 2400 bit/saniye ve altında modemlerle işletilmektedir.

Bu çalışmada; 9600 bit/saniye hızındaki sayısal verilerin iletişimi için, V.32 standardı ile tanımlanmış olan kafes kodlama modülasyon tekniği ve Viterbi algoritması kullanılarak TMS320C5402 sayısal işaret işlemcisi üzerinde modülatör ve demodülatör yaplarının gerçekleştirilmesi amaçlanmıştır. Ayrıca; demodülatör yapısında kullanılan Viterbi kod çözucüsünün hata başarımı (SER), gauss gürlültüsü eklenmiş kanal benzetimi yaparak değişik işaret gürlültü oranlarında (SNR) incelenmiştir.

Modemler; sayısal veri iletişiminin mümkün olamadığı 300 – 3500Hz arası bant sınırlı telefon şebekesi kurulduktan sonra ortaya çıkmış cihazlar olarak, sayısal verileri haberleşme kanalının karakteristik özelliklerine uygun analog sembollere dönüştürerek iletilir. Modem haberleşmesinde, bu sınırlı bant içinde veri hızını artırmak amacıyla değişik modülasyon teknikleri uygulanır. Bu teknikler, dünya çapında veri iletişimini mümkün kılmak amacıyla uluslararası standartlara
oturtulmuştur. Modülasyon ve demodülasyon işlemleri bu standartlara göre yapılır. Hata düzeltmenin modülasyon işlemi ile birlikte düşünülmemiştir ve ilk olarak Ungerboeck tarafından ortaya atılan kafes kodlama teknigi, günümüz yüksek hızlı modemlerinde kullanılan bir modülasyon seklindir [1,2,3,4].

Veri iletişiminin yapılması, doğal olarak oluşmuş karşılıklı konusma kurallarına göre geliştirilmiştir. Karşılıklı konusma sırasında uyan bu kurallar veri iletişim protokolünün temellerini oluşturumaktadır. Veri iletişiminin bu temel özellikleri [5] ikinci bölümde incelenmiştir. Ayrıca; bu bölümde, yüksek hızlı modemlerde kullanılan kafes kodlamalı modülasyonu (TCM) incelenmiştir [6,7].


kolaylıklar sağlamaktadır. Böylece, yazılım geliştirme işlemleri makine dili yerine üst seviye bir dille gerçekleştirilir.


Bu çalışmada, V.32 standardına uygun modülasyon ve demodülasyon işlemlerle sayısal işaret işlemecisi üzerinde gerçeklenmiş olup tasarlanan yazılım Ekler bölümünde verilmiştir. Ayrıca; altıncı bölümde demodülatör yapısında kullanılan Viterbi kod çözücüünün gauss güdültülü kanaldaki hata başarımı değişik işaret güdültülü oranlarında incelenmiştir.
2. MODEM HABERLEŞME TEMELLERİ

2.1 TEMEL KAVRAMLAR

2.1.1 Giriş

İki nokta arası haberleşmenin yapısı ve temel özellikleri, insanlar arasında doğal olarak oluşmuş bulunan karşılıklı konuşma kuralları göz önünde tutularak benzer kurallar üzerine oturtulmuştur. İki insanın karşılıklı olarak konuşması esnasında kendiliğinden ortaya çıkan bu kurallar iki nokta arasında gelişen haberleşmenin temellerini oluşturmaktadır.

İki nokta arası haberleşmenin dayandığı temel kurallar aşağıdaki gibi sıralanabilir:

- İki nokta arası haberleşmenin her iki tarafını oluşturan birimlerin, herhangi bir nedenden dolayı oluşan veri kayıplarını düzeltebilecek yapıda olması gereklidir. Karşılıklı konuşma sırasında dinleyen kişinin anlamadığı zaman karşındaki kişiinden tekrarını istemesi ve konuşan kişinin tekrarlamamasına karşılık gelir.

- Karşılıklı iki tarafın aynı anda haberleşmeiste bulunmaması kuralı, iki nokta arası haberleşmenin ilk kurallarının konulduğu dönemde geçerli olan fakat günümüzdeki teknolojik gelişim içinde işlerliğini yitiren bir özelliktir. Karşılıklı konuşmada aynı anda sadece bir kişinin konuşması prensibine dayandırılmıştır. Mount haberleşmesinde bu özellik, yarı çift yönlü ve tam çift yönlü uygulama olarak karşımıza çıkar. Modülasyon standardına göre uygulama biçimi de değişiklik göstermektedir.

- Haberleşen her iki tarafın, haberleşme sırasında karşı tarafın veriyi doğru algılayıp algılamadığını sorması temelde oluşturulmuş bir kuraldır. Konuşan kişinin belirli bir süre sonunda susup dinleyen kişiye anlayıp anlamadığını sorması özelliğinden çıkararak geliştirilmiştir. Hata algılama ve düzeltme algoritmaları içerisinde bu özellik desteklenmektedir.
Yukarıda bahsedilen temel kurallar insanlar arasındaki iletişimin tabii kurallarına benzerlik göstermektedir. Geçmişten günümüze kadar geliştirilmiş bulunan tüm haberleşme yapılarında kullanılan algoritmaları tasarmcılar insanları örnek almışlardır. Dikkatli incelendiğinde; haberleşme içinde bulunan iki cihaz, iki insanın karşılıklı konuşmasına benzer bir şekilde veri iletişiminde bulunurlar.

2.1.2 Haberleşme Yapıları

İki nokta arasında haberleşmenin her iki tarafını oluşturulan birimlere cihaz adı verilir. Veri iletişimi, iki cihaz arasında değişik haberleşme kanalları üzerinden yapılabilir (hava, fiber optik, bakır tel v.b.). Her cihaz; verici (sadece veri gönderen), alıcı (sadece veri alan) veya hem verici hem alıcı (aynı anda veri gönderip alan) pozisyonlarında olabilir. Bundan dolayı, üç farklı haberleşme yapısı tanımlıdır. Modern haberleşmesinde; bakır tel haberleşme kanalı, modem ile modeme bağlı olan terminal ise cihaz olarak adlandırılır.

2.1.2.1 Basit Haberleşme


![Şekil 2.1 Basit Haberleşme](image)

2.1.2.2 Yarı Çift Yönlü Haberleşme

Haberleşen cihazlardan her biri aynı anda ya veri gönderen ya da veri alan konumunda olabilir (Şekil 2.2). Dolayısı ile, haberleşme herhangi bir anda sadece tek yönde olmaktadır. Bu haberleşme biçimine en iyi örnek telsiz haberleşmesidir.
2.1.2.3 Tam Çift Yönlü Haberleşme

Haberleşen cihazlardan her biri, aynı anda hem veri gönderen hem de veri alan durumundadır (Şekil 2.3). Dolayısı ile, haberleşmenin herhangi bir anında gönderme ve alma işlemi yapılabilir. Bu haberleşme biçimine en iyi örnek telefon haberleşmesidir.

Şekil 2.3 Tam Çift Yönlü Haberleşme

Tam çift yönlü haberleşme yapısının oluşturulmasında çeşitli metotlar kullanılmaktadır. Veri iletişiminde tek haberleşme kanalı varsa; her cihaz, haberleşme kanalının değişik frekans bölgebelerini veri gönderme ve alma işlemi için kullanır. Diğer bir yöntem ise; iki ayrı haberleşme kanalı kullanmaktır. Tek haberleşme kanalındaki frekans paylaşımı tam çift yönlü haberleşme yapısına örnek olarak dial-up ve 2 tel-kiralık devre modem haberleşmesi, iki ayrı haberleşme kanalı tam çift yönlü haberleşme yapısına ise örnek olarak 4 tel-kiralık devre modem haberleşmesi verilebilir.
2.1.3 Protokol

Veri haberleşmesinin gerçekleştirilği her iki cihaz arasındaki iletişiminin sağlıklı olarak yürütebilmesi, varolan çeşitli kurallara bağlıdır. Bu kuralların her iki cihaz tarafından bilinmesi ve yürütülmesi gereklidir. Bu kurallara genel olarak protokol denir. Protokol ana başlığı altında değişik kavramlar ve kurallar tanımlanmıştır.

2.1.3.1 Bilginin Kodlanması

Karsılıklı olarak veri iletişimini gerçekleştiren her iki cihaz veriyi aynı yapida kodlamalı ve çözmelidir. Cihazlar arasındaki iletişim; bit, 4 bitlik grup (nibble) veya 8 bitlik grup (byte) yapısında olabilmektedir (Şekil 2.4). Bir cihazın veriyi byte yapısında göndermesi karşılık cihazın veriyi byte yapısında değerlendirilmesini gerektirir. Aksi taktirde verinin doğru iletilmesi mümkün değildir. Eşzamanlı haberleşme bit yapısında, eşzamanlı haberleşme ise byte yapısında tanımlanmıştır.

![Bit, Nibble, Byte](image)

Şekil 2.4 Örnek Veri Yapıları

2.1.3.2 Kontrol

Haberleşmenin sağlıklı yürütebilmesi için iki nokta arasında çeşitli kurallar oluşturulmuş ve standartlaşmıştır. Hangi tür veri haberleşmesi olursa olsun veriler çerçeve yapısında iletilir. Çerçeve içerisinde kontrol niteliği taşıyan bit veya byte'lar temel olarak; kaynak adresi, alıcı adresi, mesaj tipi, mesaj uzunluğu, ACK, NACK, çerçeve başı, çerçeve sonu ve "2.5 Eşzamanız ve Eşzamanlı Haberleşme Yapıları" bölümünde açıklanan eşlik ve CRC gibi bilgileri içerir (Şekil 2.5).

![Çerçeve Yapıları](image)

Şekil 2.5 Genel Çerçeve Yapısı
Kontrol niteliğinde kullanılan bilgilerin görevi, faydali veri iletişiminin sağlıgı olarak gerçekleşmesini sağlamaktır. Veri iletişiminin verimliliği, kullanılan kontrol bilgilerinin faydali veriye oranına bağlıdır. İletişimin sağlıgı olarak yürütülmesinden taviz vermeden, kontrol bilgilerinin faydali veriye olan oranını daha da azaltmak iletişimin verimliliğini arttırmır. Eşzamanlı haberleşmedeki verim, eşzamanlı haberleşme göre daha yüksektr. Dolayısı ile; eşzamanlı haberleşme hizin önemli olduğu yerlerde özellikle tercih edilir.

2.1.3.3 Hata Algılama ve Düzeltme

İki insan karşılıklı konuşurken; dinleyen tarafın anlamaması durumunda anlatan kişiden tekrarını istemesi, anlatan kişinin de bunu üzerine tekrar etmesi kendiliğinden ortaya çıkan doğal bir davranış biçimidir. Aynı şekilde iletişim sırasında oluşabilecek bir hatayı algılayabilecek ve algılanan bu hatayı düzeltmekteki haberleşme menin doğruluğunu ve güvenilirliğini etkiler. Dolayısı ile hata algılama ve düzeltme mekanizmaların haberleşme protokolü içinde oturtulmuş olması gereklidir.


\[\text{Şekil 2.6 Örnek hata düzeltme algoritması}\]

```
PKT 0
PKT 1
PKT 2
PKT 1
PKT 1
PKT 1
```

KAYNAK CIHAZ

HEDEF CIHAZ

Kayıp Paket #1

Şekil 2.6 Örnek hata düzeltme algoritması
2.1.3.4 Haberleşme Kanal Verimliliği


2.1.3.5 Eşzamanlı Çalışma

Veri iletişimini sırasında her iki tarafın, uygun adım içinde kalmaları yani birbirlerini anlamaları garantı altında alınmış olmalıdır. Daha öncede belirtildiği gibi veri iletişimini çerçeve yapısıdır. Çerçeve başı ve çerçeve sonu kontrol bilgileri yardımı ile her iki cihaz karşılıklı olarak birbirlerine ayak uydururlar. Bir başka deyişle eşzamanlı çalışmaya sağlarlar. Es zamanlanmanın kaybolması veri haberleşmesinin bozulmasına sebep olur. Çerçeve başı ve sonu bilinemediği için çerçeveyin yorumlanması mümkün olamaz. Çerçeve başı bilgisinin yakalanması eş zamanlanmanın tekrar sağladığı anlamına gelir.

2.1.3.6 Saydamlık

Veri iletişimini destekleyen mekanizmanın taraflardan saklanabilmesi sağlanmalıdır. Haberleşmedeki fiziksel yapı, iletişim protokolleri ve kuralları haberleşmeyi gerçekleştiren tarafları ilgilendirmemelidir. Bu tabaksal bir mimarinin kurulmuş olması ile mümkündür.

2.2 MODEM DONANIM ÖZELLİKLERİ

Modem, veri haberleşmesinde kullanılan bir haberleşme cihazdır. Temel prensibi, verileri ikili sayısal yapıdan analog yapıya dönüştürerek yaygın olarak kullanılan telefon hatları üzerinden iletmeektir. Telefon hatları, insan sesinin frekans spektrumu olan 300 – 3500Hz’ lik bant ile sınırlı olduğu için modemlerde bu frekans bandını kullanmak zorundadır. Modemleri, aralarındaki bağlantıların fiziksel yapısına ve iletişimde kullanılan paket biçimine göre sınıflandırmak mümkün olmaktadır.
Dial-up bağlantında telefon şebekesi kullanılır. Modem telefon hattını alır ve bağlantılı kuracağı modemin telefon numarasını arayarak bağlantısı kurulur. Arayan modem arayan (Orginate), aranan modem cevaplayan (Answer) konumundadır. Veri iletişimi telefon santralleri üzerinden gerçekleştilir. Kiralık devre bağlantısı şeklinde ise; iki modem uzun süre arada telefon santral olmadan veri iletişimi yaparlar. 2 veya 4 tel bağlantı olmak üzere iki bağlantılı şekli vardır.

Modemlerde değişik modülasyon teknikleri kullanılarak aynı iletişim ortamında veri hızının artırmı sağlanabilmektedir. Modem hızlarının artmasını engelleyen faktörlerin başında frekans bandının sınırlı olması ve iletişim ortamının karakteristik özellikleri gelir. Şekil 2.8’ de modeme ait genel blok diyagram gösterilmiştir. Modem sayısal ve analog olarak iki kısımdan oluşur. Sayısal kısm; mikroöçlemci devreleri, bellekler, işıklı gösterge devreleri ve V.24 arabirim devrelerini içerir. Analog kısm ise; koruma devreleri, hat erişim devreleri, hibrit devreleri ile modülasyon ve demodülasyonun yapıldığı MAFE (Modem Analog Fron End) devrelerinden oluşur.

Koruma devreleri, telefon hatlarından gelebilecek sürekli 220V değmesi veya yıldırım düşmesi gibi zararlı etkilerden hat erişim devrelerinden itibaren modem iç devrelerini korumak amacı ile kullanılarlar. Hat erişim devreleri, modemin dış hat arabirimi optik ve galvanik izolasyon sağlayarak oluşturan devrelerdir. Zilin algılanması, bağlı bulunan telefon santraline döngü akımın aktılması gibi işlemleri bu devreler gerçekleştirmir (Şekil 2.7).

![Diagram](image)

**Şekil 2.7 Tipik Hat Erişim Devresi (Dial-Up)**

Zil algılama devresi, optik izolasyon sağlayarak elektriksel zil işaretini sayısal işaretle dönüştürür. Bağlı bulunan telefon santraline hat alındığında, sabit DC akım devresi ile döngü akımının aktılması sağlanır. İzolasyon trafosu, galvanik izolasyon gerçekleştirmek amacı ile kullanılır.
Kiralık devre modem dış hat arabirim devresinde ise, galvanik izolasyonun sağladığı izolasyon trafosu bulunur.

Telefon haberleşmesi; gönderilen ve alınan işaretlerin Tip ve Ring üzerinde aynı anda bulunması ile Tip ve Ring uçlarının birbirlerine göre dengeli olması özelliğine sahiptir. Bundan dolayı; modem dış hat arabirim devresinde bulunan izolasyon trafosu çıkışında tek tel üzerinde giden ve gelen modülasyonlu işaretler bulunur (2 tel dengesiz iletişim). 2/4 tel hibrit devreleri yardımı ile gönderilen ve alınan modülasyonlu işaretler birbirinden ayrılır (Şekil 2.9).

MAFE devreleri modülasyon ve demodülasyon işlemini gerçekleştği kısımdır. Mikroislemci devrelerinden gelen sayısal işaret modüle edilerek 2/4 tel hibrit devrelerine gönderilir. Aynı şekilde dış hattan gelen modülasyonlu işaret demodüle

---

11
edilerek mikroişlemci devrelerine gönderilir. Modüle ve demodüle işlemlerleri sayısal işaret işlemlerleri ile gerçekleştirilir.

Modemin çalışmasının kullanıcı tarafından kolay izlenebilir olması gereklidir. Bunun için modemlerde ışıklı göstergeler ve hoparlör gibi elemanlar kullanılır. ışıklı göstergeler yardımcı ile modemin çalışması, bağlantıda olması, veri iletişimini ve fiziksel bağlantıların durumunu modemin haberleşmesine ait değişik durumlar gözlemlemir. Ayrıca modem haberleşmesinde, modülsyonlu ve demodülsyonlu işaretlerin duysal olarak işitilmesi modemin çalışması hakkında fikir verir.

Modemin tüm işlemleri, mikroişlemci devreleri tarafından program belleğinden yürütülen modern programı ile gerçekleştirilir. Modemin ayar profili genellikle EEPROM üzerinde saklanır.

V.24 arabirim devrelerinde ise, terminalden modemde gelen elektriksel işaretler sayısal işaretlere ve modemden terminale gönderilmek istenen sayısal işaretler V.24 standardına uygun elektriksel işaretlere dönüştürülür.

2.3 MODEM BAĞLANTI HATLARININ İNCELENMESİ

Modemler; ikili bükmüş bakır teller kullanılarak, telefon santralleri üzerinden dial-up veya direkt olarak (kiralık devre) birlirlerine bağlanırlar. Telefon ve modem haberleşme ortamı olarak kullanılan bakır tel özelliklerii, genel olarak 4 başlık altında toplanabilir;

2.3.1 Ortalama Bant Genişliği

Telefon haberleşmesinin geliştirildiği dönemlerde; kullanılacak kablolar insanların konuşma karakteristiğine göre tasarlanmıştır. İnsanların kullandıkları frekans bölgesi; 300Hz bölgesinde başlayarak 3500Hz bölgelerine kadar çıkar. İnsanın bulunmuş olduğu ruh halı ve fizyolojik yapısı insanların kişisel konuşma frekans bandını belirler. Geniş bir frekans bölgesine sahip hattın maliyeti de fazla olmaktadır. Dolayısı ile telefon hatları sadece insan konuşmalarının iletimi için tasarlanmıştır. Modemler daha sonra geliştirilmiş cihazlar olduklarını için varolan telefon şebekesini kullanabilmeleri amaçlanmıştır. Bunun için, modemler iletişimlerini bu bant içinde yapmak zorundadırlar.

Ortalama Bant genişliği, seçilen referans frekansına göre 3dB zayıflamının olduğu frekanslar arası tanımlanmıştır. Kuzey Amerika'da referans frekansı 1KHz, Avrupa'da ise 800Hz kullanılır (Şekil 2.10).
2.3.2 Modülasyonlu İşaret Frekans Genlik Değişimi

İletişim kanalının frekans bandında (300-3500Hz); veri haberleşmesinde kullanılan modülasyonlu işaretlerin, frekans bileşenlerinin genlikleri CCITT G.132 olarak bilinen standart ile karakterize edilmiştir.

G.132 standardı; 1KHz referans frekansının genlik seviyesine göre, 300 – 3500Hz arasındaki frekans bileşenlerinin genlik seviyeleri arasındaki farklılıkların hangi sınırlar arasında kalmastı gerektiğini tanımlar (Şekil 2.11).

2.3.3 Gürtültü

Mutlak sıcaklık ve boltzmann sabiti ile orantılı olan isısal gürtültü ortalama olarak oda sıcaklığında -100dBm seviyesinde oluşan beyaz gürtültüdür. Bu gürtültü biçiminin engellenmesi mümkün değildir. Pratik çalışmalarında beyaz gürtültü sınıri -60dBm olarak kullanılır.

Kilometreler boyunca yan yana giden farklı hatlar arasındaki kapasitif etki sonucu sembol karışmaları (crosstalk) ortaya çıkar. Bu etkiye azaltmak amacı ile aynı hattı oluşturan kablolar birbirlerine dolanır ve dengeli iletişim kullanılır.
2.3.4 Faz Gecikmesi


Şekil 2.12 Frekans – Gecikme Karakteristiği

2.4 CCITT V.24 ÖZELLİKLERİ

CCITT V.24 veya RS-232C olarak bilinen standart, veri terminal cihazı (DTE) ve veri haberleşme cihazı (DCE) arasındaki bağlantı ve haberleşme özelliklerini belirler. Şekil 2.13’ de iki nokta arası modem haberleşmesinin genel yapısı görülmektedir. Şekil 2.14’ de ise DTE ve DCE arasında tanımlanmış olan CCITT V.24 bağlantı yapısı verilmiştir.

Şekil 2.13 Modem Haberleşme Yapısı


**Şekil 2.14 DTE – DCE Arası Bağlantı Yapısı**


**Şekil 2.15 V.24 Elektriksel Seviyeleri**
Şekil 2.15’ de V.24 standardında 1 ve 0 bölgeleri görülmektedir. 1 ve 0 işaretinin elektriksel değeri nominal +12V ve -12V tur.

Bu şekilde simetrik kullanılması ile zamanda yapılan iletişimden dolayı hat türünde oluşan DC seviye sıfıra yakın tutulur. Bundan dolayı; V.24 bağlantısı uzak mesafelere kolaylıkla taşınabilir.

2.5 EŞZAMANSIZ VE EŞZAMANLI HABERLEŞME YAPILARI

Modemler bağlantı kurduktan sonra veri iletişimini iki farklı yapıda gerçekleştirebilirler. Bunlar eşzamansız ve eşzamanlı veri yapılarıdır.

2.5.1 Eşzamansız Veri Yapısı

Çerçeve başlangıç biti, veri bitleri, eşlik biti ve çerçeve sonu bitinden oluşan bir çerçeve yapıtır. Aylık olarak 1 pozisyonunda bulunan veri hattı bir bit süre ile 0 konumuna çekilerek çerçeve başlangıç biti verilmiş olunur. Çerçeve başlangıç biti sonrası 7 veya 8 bit veri ve ardından hata algılama kullanlan eşlik biti gönderilir. Eşlik biti sonrası çerçeve sonu biti ile bir çerçeve tamamlanmış olunur (Şekil 2.16).

![Şekil 2.16 Eşzamansız Çerçeve Yapısı](image)

Eşzamansız çerçeve yapısı, donanım tarafından oluşturulur ve çözülür. Çerçeve başlangıç ve bitiş bitleri ile eşlik biti eşzamansız çerçevede kullanılan kontrol bitleridir.

Çerçeve bitleri birbirinden ayırmada ve senkronizasyonda çerçeve başlangıç ve bitiş bitleri kullanılır. Çerçeve bitiş biti, 1, 1.5 ve 2 bit olmak üzere seçeneğidir.

Eşlik biti, hata algılama için kullanılan bir kontrol biti olup çift eşlik, tek eşlik, 1, 0 ve kullanılmama gibi beş değişik seçeneği bulunmaktadır. Çerçeve içerisindeki tek sayıda bit değişimlerini algelayabilecek yapıdadır. İletişim sırasında meydana gelecek çift
bit değişim hatalarını algılayamaz. Ayrıca, iletişimi gerçekleştiren her iki tarafında aynı yapıda eşlik bitini oluşturması gereklidir.

Eşzamanlı çerçeve yapısında, faydalı veri dışındaki kontrol bitleri fazladır. Bu yüzden veri iletim verimi senkron yapıya nazaran daha düştüktür. Fakat hata oluşum durumunda senkron yapıya göre paket yapısı kısa olduğu için bunun telafisi çabuk olabilmektedir.

### 2.5.2 Eşzamanlı Veri Yapısı

Çerçeve başlangıç, veri uzunluk, veri, CRC ve çerçeve bitiş byte' in dan oluşan bir çerçeve yapısıdır (Şekil 2.17). Verinin uzun olması durumunda senkronizasyonun kaybedilmemesi için zamanlama byte' ları da kullanılır.

![Şekil 2.17 Eşzamanlı Çerçeve Yapısı](image)


### 2.5.2.1 LRC (Longitudinal Redundancy Check)

Kullanılan referans LRC byte' ı nın her biti verinin aynı numaralı bitleri ile EXOR işlemine tabi tutularak LRC byte' i oluşturulur ve veri sonrasında gönderilir (Şekil 2.18). Alıcı taraf gönderilen paket içerisindeki veri ve LRC byte' ini, kendi referans LRC'ı sini kullanarak EXOR işlemine tabi tutar ve her bit sütunu için 0 sonucuna ulaşmaya çalışır. Sonucun 00h olması hatanın olmaması demektir. Bu teknikle byte'
lar arası bitlerin kontrolü yapılır. Her iki tarafında kullandığı referans LRC nin aynı olma zorunluluğu vardır.

\[
\begin{array}{c}
1 1 1 1 1 1 1 1 1 1 \\
\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\
1 \leftrightarrow 0 1 1 0 1 1 0 0 \quad 1. \text{veri byte'ı} \\
0 \leftrightarrow 1 0 1 0 1 1 1 1 \quad 2. \text{veri byte'ı} \\
0 \leftrightarrow 0 1 1 1 0 1 0 1 \quad 3. \text{veri byte'ı} \\
1 \leftrightarrow 1 1 1 0 0 0 1 0 \quad 4. \text{veri byte'ı} \\
1 \leftrightarrow 0 0 0 1 0 1 1 1 \quad 5. \text{veri byte'ı} \\
1 0 1 1 1 1 0 0 \quad \text{LRC}
\end{array}
\]

Şekil 2.18 LRC Oluşturma Yapısı

2.5.2.2 VRC (Vertical Redundancy Check)


2.5.2.3 CRC (Cyclic Redundancy Check)

Gönderilecek veri (2.1), bir çok terimliye otorultur ve bir üretç çok terimlinin derecesi kadar katılırıp üretç çok terimliye bölünür (2.2). Kalan değer oteldenmiş çok terimliye eklenecek gönderilecek veri oluşturulur (2.3). Hata yakalama olasılığı çok yüksek olan bu sistem tek ve çift bit hataları, hata patlamları gibi sistematik oluşumları algılayabilir.

\[
M(x) = m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \ldots + m_1x + m_0 \quad k: \text{Veri bit sayısı} \tag{2.1}
\]

\[
\frac{M(x)x^q}{G(x)} = Q(x) \oplus \frac{R(x)}{G(x)} \\
\]

G(x) : Üretç çok terimli 
\(g\) : Üretç çok terimli derecesi 
Q(x) : Bölüm sonuç çok terimli 
R(x) : Kalan sonuç çok terimli 
\tag{2.2}

\[
T(x) = x^q . M(x) \oplus R(x) \quad T(x) : \text{Gönderilecek veri} \tag{2.3}
\]

Karşılık gönderilen veri (T(x)), (2.4) denkleminde gösterildiği gibi CRC çok terimlisine bölünerek hata kontrolü yapılır. T(x)' in hatası bir şekilde ileti R(x) in olmamasını gerektirir.
\[
\frac{T(x)}{G(x)} = \frac{x^8 \cdot M(x) \oplus R(x)}{G(x)} = \frac{Q(x)}{G(x)} \oplus \frac{R(x)}{G(x)}
\] (2.4)

CRC-12, CRC-16, CRC-CCITT ve CRC-32 gibi standart üreteç çok terimlileri çeşitli uygulamalarda kullanılarak hata algılama işlemi gerçekleştirilir.

Algılanan hata, değişik hata düzeltme protokolleri ile karşılıklı DTE’ler arasında giderilmeye çalışılır. Günümüzde kullanılan değişik tipte hata düzeltme protokolleri mevcuttur.

2.6 KAFES KODLAMALI MODÜLASYON (TCM)

“2.3 Modem Bağlantı Hatlarının İncelenmesi” bölümünde özellikleri açıklanan ve modem haberleşmesinde kullanılan haberleşme kanalı, 0.3KHz ile 3.5KHz arasında iletişim imkanı sağlar. Bundan dolayı veri iletişimini modülasyonlu işaretler ile yapılır. Sayısal verinin, haberleşme kanalında kullanılan sembollere dönüştürülmesi işleme modülasyon denir. Bu dönüştürme işleminin terci demodülasyon olarak adlandırılır. Tablo 2.1’de CCITT ve Bell tarafından belirlenmiş olan iletişim standartları ve modülasyon yapıları gösterilmiştir.

<table>
<thead>
<tr>
<th>Standart</th>
<th>Hız</th>
<th>Operasyon Şekli</th>
<th>Modülasyon Şekli</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bell 103</td>
<td>0-300bps</td>
<td>Tam Çift Yön</td>
<td>FSK</td>
</tr>
<tr>
<td>Bell 20x1</td>
<td>1200bps</td>
<td>Yarı Çift Yön</td>
<td>PSK</td>
</tr>
<tr>
<td>Bell 202</td>
<td>1200bps</td>
<td>Yarı Çift Yön</td>
<td>FSK</td>
</tr>
<tr>
<td>Bell 212A</td>
<td>0-1200bps</td>
<td>Tam Çift Yön</td>
<td>PSK</td>
</tr>
<tr>
<td>CCITT V.21</td>
<td>0-300bps</td>
<td>Tam Çift Yön</td>
<td>FSK</td>
</tr>
<tr>
<td>CCITT V.22</td>
<td>1200bps</td>
<td>Tam Çift Yön</td>
<td>PSK</td>
</tr>
<tr>
<td>CCITT V.22bis</td>
<td>2400bps</td>
<td>Tam Çift Yön</td>
<td>QAM</td>
</tr>
<tr>
<td>CCITT V.23</td>
<td>1200/75bps</td>
<td>Yarı Çift Yön</td>
<td>FSK</td>
</tr>
<tr>
<td>CCITT V.26</td>
<td>2400bps</td>
<td>Yarı Çift Yön</td>
<td>PSK</td>
</tr>
<tr>
<td>CCITT V.27</td>
<td>4800bps</td>
<td>Yarı Çift Yön</td>
<td>PSK</td>
</tr>
<tr>
<td>CCITT V.29</td>
<td>9600bps</td>
<td>Yarı Çift Yön</td>
<td>QAM</td>
</tr>
<tr>
<td>CCITT V.32</td>
<td>9600bps</td>
<td>Tam Çift Yön</td>
<td>TCM</td>
</tr>
<tr>
<td>CCITT V.32bis</td>
<td>14400bps</td>
<td>Tam Çift Yön</td>
<td>TCM</td>
</tr>
<tr>
<td>CCITT V.34</td>
<td>28800bps</td>
<td>Tam Çift Yön</td>
<td>TCM</td>
</tr>
<tr>
<td>CCITT V.34+</td>
<td>33600bps</td>
<td>Tam Çift Yön</td>
<td>TCM</td>
</tr>
</tbody>
</table>

Görüldüğü gibi Bell standardı sadece 1200bps kadar hızları destekler. Modem haberleşmesinde kullanılan haberleşme kanalının karakteristik özelliklerinden dolayı, kanal üzerindeki sembol hızı artamamaktadır. Bu nedenle; değişik modülasyon teknikleri kullanılarak modemlerin gönderdikleri sayısal veri hızı arttırılır.

Bu bölümde, yüksek hızlı modemlerde kullanılan modülasyon tekniği olan kafes kodlamalı modülasyon (TCM) incelenmiştir.
2.6.1 Giriş

Kafes kodlamalı modülasyon (TCM) tekniği, genişletilmiş işaret diyaframları kullanılarak kodlanmış işaret dizileri arasındaki en küçük Öklid uzaklığını artırma ilkesine dayanır. İletim sembollерinin işaret diyaframı üzerinde birbirlerinden uzak seçilmesi, haberleşme ortamındaki gürültülerden kaynaklanan hataları azaltmakta ve bandı sınırlı haberleşme kanallarındaki hata başarımı artmaktadır.


Kafes kodlamalı modülasyon, modülasyon ve kanal kodlama işlemlerinin birlikte tasarlandığı bir tekniktir. Bu nedenle; kodlayıcı, ikili sayısal veriler arasındaki Hamming uzaklığı yerine modülasyonu kanal işaretleri arasındaki Öklid uzaklığını artıracak biçimde tasarlanmıştır. Aynı şekilde alınan kanal işaretleri; demodüle edilerek ikili sayısal verilere dönüştürüldükten sonra çözülerek yerine, kod çözücü direkt olarak alınan kanal işaretlerini çözceek şekilde tasarlanır [6].

2.6.2 Katlamalı Kod Yapıları

![Sekil 2.19 R=n/m Katlamalı Kodlayıcı Yapısı](image-url)
k (k=0,1,2,...) kodlama adımı göstermek üzere; her k için, n uzunlukundaki x_k giriş dizisine karşılık m uzunlukta y_k çıkış dizisi oluşturan yapıya R=n/m oranlı katlamalı kodlayıcı denir (Şekil 2.19). (2.5)' de giriş, (2.6)' da çıkış dizisi görülmektedir.

\[ X(D) = [ X^1(D), X^2(D), \ldots, X^n(D) ] \] 
\[ X^i(D) = x_0^i + x_1^i.D + x_2^i.D^2 + x_3^i.D^3+ \ldots \quad 1 < i < n \] 
\[ Y(D) = [ Y^1(D), Y^2(D), \ldots, Y^n(D) ] \] 
\[ Y^j(D) = y_0^j + y_1^j.D + y_2^j.D^2 + y_3^j.D^3+ \ldots \quad 1 < j < m \]

Böylece katlamalı kodlama işlemi, (2.7)' de olduğu gibi tanımlanabilir.

\[ Y(D) = X(D) \cdot G(D) \] 
\[ G(D), n \times m \] boyutlu bir matristir ve bu matrisin elemanları (2.8)' deki gibi tanımlanır.

\[ G^i_j(D) = g_0^i_j + g_1^i_j.D + g_2^i_j.D^2 + g_3^i_j.D^3+ \ldots \quad 1 < j < n \]

Katlamalı kodlayıcının kısıtlanmış uzunluğu (v), gerekli olan bellek sayısını belirler ve (2.9)' da ki gibi tanımlıdır.

\[ v = \sum_{i=1}^{n} \max_j \{ \deg G^i_j(D) \} \] 

v uzunlığında bir katlamalı kodlayıcı, v katlı ötelemeli bir kaydedicidir. V.32 standardında kullanılan katlamalı kodlayıcı v=3 uzunluklu 2/3 oranıdır. Dolayısı ile 3 adet bellek gözü kullanılır ve çıkış dizisi bu bellek gözlerindeki durumlara bağlıdır.

2.6.3 Kod Çözme İşleimi ve Özellikleri

Klasik haberleşme sistemlerinde modülatör ve hata düzeltmeli kodlayıcı ayrı ayrı tasarlanmaktadır. Bu tipe sistemlerde her bir sembol iletim zaman aralığındında a adet sayısal veriye karşılık 2^a olan analog işaret karşılık gelir. Demodülatör katında ise a adet sayısal veri tekrar oluşturulur. Kafes kodlamalı modülasyonda ise; hata düzeltme, modülasyon işlemi ile birlikte düşünülmektedir. a adet faydalı sayısal
veriye, n-a adet fazlalık kontrol verisi eklenerek modülasyon işlemi yapılır. Klasik haberleşmede, kod dizileri arasındaki farklı simge sayısı, Hamming uzaklığı olarak isimlendirilir. En küçük uzaklık \( d_{\text{min}}^{H} \) olarak tanımlı ise kod çözücü en az \( (d_{\text{min}}^{H} - 1)/2 \) kadar olan hataları düzeltabilir. Hata düzeltme yeteneğini artırmak için fazlalık kontrol verisini artırma bağdır. Kod çözücü dilimlenmiş kanal çıkış örneklerini kullanır. Modülatörden gönderilen ayrı işaretler \( (a_n) \), haberleşme ortamından kaynaklanan ve işaretе eklenen sıfır ortalamalı ve \( \sigma^2 \) sapmalı beyaz Gauss gürültüsü örnekleri \( (w_n) \) ile birlikte \( r_n \) kanal çıkış örneklerini oluşturur (2.10).

\[
r_n = a_n + w_n
\]  
(2.10)

Böyle bir modelleme için işaret gürültü oranı (SNR), (2.11)’deki gibi tanımlanabilir.

\[
\text{SNR} = \frac{E\{|a_n|^2\}}{E\{|w_n|^2\}}
\]  
(2.11a)

\[
\text{SNR} = \begin{cases} 
E\{|a_n|^2\} / \sigma^2 , & \text{tek boylulu modülasyon} \\
E\{|a_n|^2\} / 2\sigma^2 , & \text{çift boylulu modülasyon}
\end{cases}
\]  
(2.11b)

En uygun çözme karar kuralı ile alıcı girişinde elde edilen \( r_n \) dizisine en küçük karesel Öklid uzaklıkları bir \( a_n^l \) dizisi kodlanmış işaretler arasında belirlenir. Bu \( a_n^l \) dizisi (2.12) eşitliğini sağlamalıdır [2].

\[
|r_n - a_n|^2 = \min_{\{a_n^l\} \in C} \sum |r_n - a_n|^2
\]  
(2.12)

İlk olarak A. J. Viterbi tarafından 1967 yılında katılamalı kodların çözme işlemi için önerilen Viterbi algoritması, alıcıya gelen \( r_n \) dizisine en yakın kodlanmış olan \( a_n^l \) dizisine karar vermekte kullanılır (2.12). Vericide olusturulan kodlanmış işaret dizileri \( (a_n) \), katılamalı kodlayıcının belirlediği sonlu durum diyagramını izler. Çözücü yapısı, sert karar verme yerine çözücü girişindeki en olması kodlanmış ardışık işaret dizisine karar verir. Gönderilen kodlu işaret dizileri ve yumuşak olarak Viterbi algoritması tarafından karar verilen kodlu işaret dizileri arasındaki fark olası karar verme hatalını oluşturur. Minimum karesel Öklid uzaklığı (2.13)'deki gibi ifade edilir.
\[ d^2 = \text{Min} \sum_{\{a_n\} \neq \{a_n\}} |a_n - a_n'| ; \{a_n\}, \{a_n'\} \in \mathcal{C} \] (2.13)

2.6.4 Kafes Kodlamada İşaret Kümeleri – Katlamalı Kodlar

Kafes kodlamalı modülasyon katlamalı kodların özel halidir. Modülasyon işlemi gruplara ayrılmış olan işaret kümeleri üzerine yapılır. İşaret kümelerinde yer alan noktalar, birbirlerine uzaklıkları en büyük olacak şekilde belirlenir.

Bu çerçevesinde kafes kodlama kullanılan temel bant işaretinin üretimi incelenecektir; iletişmek m adet sayısal veri biti için n adet bit (n \leq m), n/(n+1) oranlı katlamalı kodlayıcılıdanne geçrilerek n+1 kodlanmış veri biti elde edilir. Genişletilmiş 2^{m+1} lik işaret durum diyagramı içindeki 2^{n+1} adet işaret kümelerinden birini seçmekte kullanılır. m-n adet kodlanmayan veri ise, bu işaret kümesi içinde yer alan 2^{m-n} işaretten hangisi olduğunu belirler (Şekil 2.20).

![Diagram](image)

Şekil 2.20 Kafes Kodlamalı Modülatör


İşaret kümelerinin ayrıştırılması, işaret diyagramını sırayla daha küçük kümelere bölme işlemidir. Bu bölme işleminde kümé içindeki işaretler arası uzaklıklar artar. Ayrıştırma işlemi, D_{n+1} tasarlanmak istenen kodun istenen işaretler arası uzaklktan
daha büyük veya eşit olana kadar n+1 kez tekrarlanır. Sonuç olarak Şekil 2.21’de görülen alt kümeler elde edilir. Her bir alt kümeyi, \( Z_n = [Z_n^m, ..., Z_n^0] \) şeklinde göstermek mümkündür. Herhangi iki alt kümenin gösteriminde son q tane bitin uyuşması, bu iki alt kümenin ayrıştırma işleminde q. seviyedeki alt kümenin elemanları olmasına ve aralarındaki uzaklığın \( D_q \) olmasını gerektirir. m-n adet kodlanmış veri, seçilen alt küme içinden bir noktaya karşılık düşer. Kaşes kodlamada alt küme noktaları \( 2^m-n \) paralel geçişle birleştirilir. Bu modülasyonda serbest Öklid uzaklığı, (2.14)'deki gibi tanımlanır. \( D_{n+1} \) alt küme içindeki minimum paralel geçiş uzaklığını, \( d_f(n) \) ise kümeler arası minimum uzaklığı gösterir.

\[
d_f = \min [D_{n+1}, d_f(n)]
\]  

Şekil 2.21 32 – noka İşaret DiyagramaNDaki İşaret Kümelerinin Ayrıştırılması

Her sembol gönderme anında, Şekil (2.20)'de gösterilmiş n/n+1 oranlı katlamalı kodlayıcı n adet giriş bilgisine karşılık n+1 kodlanmış bilgi üretir. Genel olarak, bu kodlama tekniği n adet veri biti gruplarının sonlu durumlu bir makineye uygulanmasını içerir. Bu sonlu durum makinesi; her bir giriş grubunu, bazı belirli bitler kodlayıcida saklanarak önceden belirlenmiş mantıksal kombinasyonlarla uyumlu olacak şekilde n+1 bit grubuna genişletir. Kodlayıcı çıkışı etkileyen kodlayıcı belleklerinin sayısı (S), kodlayıcının varsayıdığı \( 2^S \) konumlu sonlu durum makinesini belirler. (Bakınız : 5.4.1 Kodlayıcı ve Temel Bant İşareti Oluşturma Yapısı)

24

2.6.5 32 – Nokta Katlamalı Kod

Katlamalı kodlayıcı durumlarının \( S_1, S_2, S_3 \) olduğu ve işaret durum diyagramını oluşturan 32 noktasının 8 ayrı kümeye ayrıldığı kabul edilmektedir. Bu kodlamaya ait durum geçiş diyagramı Şekil 2.22’de gösterilmiştir.

\[ \text{Küme A B D C} \]
\[ \text{Küme D C A B} \]
\[ \text{Küme H G F E} \]
\[ \text{Küme E F G H} \]
\[ \text{Küme B A C D} \]
\[ \text{Küme C D B A} \]
\[ \text{Küme G H E F} \]
\[ \text{Küme F E H G} \]

\[ (n-1)T \]
\[ nT \]

Şekil 2.22 Katlamalı Kodlama Küme Durum Geçiş Diyagramı

\( f_i \) (\( S_1, S_2, S_3 \)) durumundan, \( f_i \) (\( S_{1+n}, S_{2+n}, S_{3+n} \)) durumuna geçişlerle ilgili işaret eleman kümeleri, \( S_1, S_2, S_3 \) durumundan \( S_{1+n}, S_{2+n}, S_{3+n} \) durumuna geçişe ilgili işaret eleman kümesinden tüm işaret elemanlarının sırasıyla -90°, -180° ve -270° döndürülmesiyle elde edilir. (i = 1,2,3)

Şekil 2.22’de durum geçiş diyagramı verilen kafes kodlamasının kullanıldığı V.32 modülatörün yapısı Şekil 3.2’de görülmektedir. 32 – nokta işaret diyagramının 90°,
180° ve 270° faz belirsizlikleri katlamalı kodlayıcı girişleri olarak kullanılan verilerin farklı kodlanmaları sonucu ortadan kaldırılır.

2.6.6 Kafes Katlamalı Sistem Hata Olasılığı ve Viterbi Algoritmasy

Haberleşme kanalının Gauss gürültülü olduğu öngörülürse; ayrıca Viterbi kod çözme yapışı kullanılabileceği sistem hata olasılığı serbest uzaklıkla alttan sınırlıdır ve (2.15)'deki gibi tanımlanır [1].

\[ P_r(e) > N(df) \cdot Q(df/(2\sigma)) \]  \hspace{1cm} (2.15)

\[ \sigma^2: \text{Gauss gürültü sapması} \]
\[ N(df): \text{df serbest uzaklığı ortalama hatalı sayışı} \]
\[ Q(.): \text{Gauss hata olasılık fonksiyonu} \]


Her işaret alındığında hangi paralel geçiş kümesi içerisindeki hangi elemana yakın olduğu belirlenir. Bu eleman için Öklik uzaklığı hesaplanır ve eski değerlerin en küçük olanı ile toplanarak yeni değerler oluşturulur. n.T ani için yeni değerler (2.16)'deki gibi tanımlanır.

\[ M_n (\ldots a_n) = M_{n-1} (\ldots a_{n-1}) + |r_n - a_n|^2 \]  \hspace{1cm} (2.16)

3. ITU-T V.32 STANDARDI

3.1 GİRİŞ

V.32 standardı, ulusal telefon şebekesi (PSTN – Public Switching Telephone Network) ve kiralık devreler üzerinde kullanılan 9600 bit/saniye veri işaretleşme hızındaki çift yönlü modem uygulamalarında tanımlıdır.


3.2 GENEL ÖZELLİKLER

V.32 standardı, modemlerin ulusal telefon şebekesi ve noktadan noktaya kiralık devreler üzerinden 9600 bit/saniye veri sinyalleşme hızında çift yönlü bağlanabilmesi amacı ile oluşturulmuştur. Genel karakteristik özellikleri aşağıdaki gibi tanımlanalabilir;

a) Ulusal telefon şebekesi ve kiralık devreler üzerinde tam çift yönlü çalışma.

b) Yankı giderme teknikleri ile kanal ayırma.

c) 2400 baud hızında eşzamanlı hat iletişimiyile her bir kanal için dik genlik modülsiyonu kullanma.

d) 9600 bit/saniye, 4800 bit/saniye ve 2400 bit/saniye eşzamanlı veri iletişim hızlarını kullanma.

e) 9600 bit/saniye veri haberleşme hızında farklı iki modülsiyon yapışı kullanma,
   • 16 işaretli dik genlik modülsiyonu
   • 32 işaretli kafes kodlamalı modülsiyon
İki alternatif modülasyon şekli olmasına rağmen 9600 bit/saniye veri iletişim hızında modemler 16 işaretli modülasyon yapısını desteklemelidir.

f) Veri hızı, kodlama gibi özellikleri el sıkışma fazında diğer modeme ileme.

g) V.14 veya V.42 standartlarına göre seçilmiş olarak eşzamansız çalışma.

h) Karşıtarma ve Çözme fonksiyonu.

3.3 HAT İŞARETLERİ

3.3.1 Taşıyıcı Frekansı ve Modülasyon Hızı

Taşıyıcı frekansı 1800 ±1Hz olarak tanımlıdır. Kılavuz ton kullanılmaz. Alıcı, alınan frekansı ±7Hz toleransla değerlendirilebilecek yapida olmalıdır. Modülasyon hızı 2400 baud ± 0.01% olarak kullanılır.

3.3.2 Spektrum

Gönderilecek ikili bitler karıştırıcı girişine uygulanır. 600Hz – 3000Hz arası spektrumda gönderilen enerji yoğunluğu, 600Hz – 3000Hz arası spektrumdaki maksimum enerji yoğunluğuna göre 4.5 ±2.5dB değişmelidir.

3.4 KODLAMA

3.4.1 9600 bit/saniye veri haberleşme hızında işaret kodlaması

9600 bit/saniye hızında iki değişik kodlama yapısı tanımlanmıştır.

3.4.1.1 Normal Kodlama

Gönderilecek karıştırılmış veri katarı, 4 ardışık veri bitinden oluşan gruplara bölünür. Her grup içinde ilk iki bit olarak bulunan Q1n ve Q2n, Tablo 3.1’ e göre Y1n ve Y2n ile farksal kodlamaya tabi tuturlar. Burada n, 4 ardışık veri bitinden oluşan grup numaralarını ifade etmekteidir.

Y1n , Y2n , Q3 ve Q4n bitleri Şekil 3.1’ de gösterilen ve Tablo 3.3’ de listelenen işaret diyagramına göre modüle edilirler.
Tablo 3.1 4800 bit/saniye için farksal kadran kodlama ve 9600 bit/saniye için normal kodlama

<table>
<thead>
<tr>
<th>Girişler</th>
<th>Önceki Çıkıslar</th>
<th>Faz değişimi</th>
<th>Çıkıslar</th>
<th>4800 bit/sn için işaret durumu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1_n</td>
<td>Q2_n</td>
<td>Y1_n-1</td>
<td>Y2_n-1</td>
<td>360°</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

| 0       | 1              | 0            | 0        | 0                            |
| 0       | 1              | 0            | 1        | 1                            |
| 0       | 1              | 1            | 0        | 0                            |
| 0       | 1              | 1            | 1        | 1                            |

| 1       | 0              | 0            | 0        | 1                            |
| 1       | 0              | 1            | 1        | 0                            |
| 1       | 0              | 1            | 0        | 0                            |
| 1       | 0              | 1            | 1        | 1                            |

| 1       | 1              | 0            | 0        | 1                            |
| 1       | 1              | 1            | 1        | 0                            |
| 1       | 1              | 1            | 0        | 0                            |
| 1       | 1              | 1            | 1        | 1                            |

Not: 4 haneli ikili veriler sırayla; Y1_n, Y2_n, Q3_n ve Q4_n’i iade eder.

Şekil 3.1 9600 bit/saniye için normal kodlama 16-nokta işaret diyagramı ve 4800 bit/saniyede ve öğrenme fazında kullanılan A, B, C ve D durumları.
3.4.1.2 Kafes Kodlama

Gönderilecek karışırılmış veri katan, 4 arıtırık veri bitinden oluşan gruplara bölünür. Şekil 3.2’ de gösterildiği gibi, her grup içinde ilk iki bit olarak bulunan Q1n ve Q2n , ilk olarak Tablo 3.2’ ye göre Y1n ve Y2n ile farksal kodlamaya tabi tutulurlar. Burada n, 4 arıtırık veri bitinden oluşan grup numaralarını ifade etmektedir.

Tablo 3.2: 9600 bit/saniye hızında kafes kodlaması için farksal kodlayıcı tablosu

<table>
<thead>
<tr>
<th>Girişler</th>
<th>Önceki Çıkıslar</th>
<th>Çıkıslar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1n</td>
<td>Q2n</td>
<td>Y1n-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Farksal kodlayıcı çıkışları olan Y1n ve Y2n , ek Y0n bitinin oluşturduğu simetrik katlamalı kodlayıcının girişleri olarak kullanılar.

Bu ek bit (Y0n) ve bilgi taşıyan Y1n , Y2n , Q3n ve Q4n bitleri Şekil 3.3’ de gösterilen ve Tablo 3.3’ de listelenen işaret diyagramına göre modüle edilirler.
Şekil 3.2  9600 bit/saniye hızında Kafes Kodlama

Sembol doğruluk tablosu

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>I₁</th>
<th>I₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

3.4.2 4800 bit/sn veri haberleşme hızında işaret kodlaması

Gönderilecek karıştırılmış veri katarı, 2 ardışık veri bitinden oluşan gruplara bölünür. Q1_n ve Q2_n olarak tanımlanan bu bitler, Tablo 3.1' e göre Y1_n , Y2_n ile farklı kodlamaya tabi tutulurlar. Burada n, 2 ardışık veri bitinden oluşan grup numaralarını ifade etmektedir. 4800bps veri iletişim hızında kullanılan A, B, C ve D işaret durumları Şekil 3.1' de gösterilmiştir.
Tablo 3.3 9600 bit/saniye hızında iki alternatifli işaret diyagram tablosu

<table>
<thead>
<tr>
<th>Kodlanmış Girişler</th>
<th>Normal Kodlama</th>
<th>Kafes Kodlama</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Y0) Y1 Y2 Y3 Y4</td>
<td>Re(X) Im(Y)</td>
<td>Re(X) Im(Y)</td>
</tr>
<tr>
<td>0 0 0 0 0</td>
<td>-1 -1</td>
<td>-4 1</td>
</tr>
<tr>
<td>0 0 0 0 1</td>
<td>-3 -1</td>
<td>0 -3</td>
</tr>
<tr>
<td>0 0 0 1 0</td>
<td>-1 -3</td>
<td>0 1</td>
</tr>
<tr>
<td>0 0 0 1 1</td>
<td>-3 -3</td>
<td>4 1</td>
</tr>
<tr>
<td>0 0 1 0 0</td>
<td>1 -1</td>
<td>4 -1</td>
</tr>
<tr>
<td>0 0 1 0 1</td>
<td>1 -3</td>
<td>0 3</td>
</tr>
<tr>
<td>0 0 1 1 0</td>
<td>3 -1</td>
<td>0 -1</td>
</tr>
<tr>
<td>0 0 1 1 1</td>
<td>3 -3</td>
<td>4 -1</td>
</tr>
<tr>
<td>0 1 0 0 0</td>
<td>-1 1</td>
<td>-2 3</td>
</tr>
<tr>
<td>0 1 0 0 1</td>
<td>-1 3</td>
<td>-2 -1</td>
</tr>
<tr>
<td>0 1 0 1 0</td>
<td>-3 1</td>
<td>2 3</td>
</tr>
<tr>
<td>0 1 0 1 1</td>
<td>-3 3</td>
<td>2 -1</td>
</tr>
<tr>
<td>0 1 1 0 0</td>
<td>1 1</td>
<td>2 -3</td>
</tr>
<tr>
<td>0 1 1 0 1</td>
<td>3 1</td>
<td>2 1</td>
</tr>
<tr>
<td>0 1 1 1 0</td>
<td>1 3</td>
<td>-2 -3</td>
</tr>
<tr>
<td>0 1 1 1 1</td>
<td>3 3</td>
<td>-2 1</td>
</tr>
<tr>
<td>1 0 0 0 0</td>
<td>- -</td>
<td>-3 -2</td>
</tr>
<tr>
<td>1 0 0 0 1</td>
<td>- -</td>
<td>1 -2</td>
</tr>
<tr>
<td>1 0 0 1 0</td>
<td>- -</td>
<td>-3 2</td>
</tr>
<tr>
<td>1 0 0 1 1</td>
<td>- -</td>
<td>1 2</td>
</tr>
<tr>
<td>1 0 1 0 0</td>
<td>- -</td>
<td>3 2</td>
</tr>
<tr>
<td>1 0 1 0 1</td>
<td>- -</td>
<td>-1 2</td>
</tr>
<tr>
<td>1 0 1 1 0</td>
<td>- -</td>
<td>3 -2</td>
</tr>
<tr>
<td>1 0 1 1 1</td>
<td>- -</td>
<td>-1 -2</td>
</tr>
<tr>
<td>1 1 0 0 0</td>
<td>- -</td>
<td>1 4</td>
</tr>
<tr>
<td>1 1 0 0 1</td>
<td>- -</td>
<td>-3 0</td>
</tr>
<tr>
<td>1 1 0 1 0</td>
<td>- -</td>
<td>1 0</td>
</tr>
<tr>
<td>1 1 0 1 1</td>
<td>- -</td>
<td>1 -4</td>
</tr>
<tr>
<td>1 1 1 0 0</td>
<td>- -</td>
<td>-1 -4</td>
</tr>
<tr>
<td>1 1 1 0 1</td>
<td>- -</td>
<td>3 0</td>
</tr>
<tr>
<td>1 1 1 1 0</td>
<td>- -</td>
<td>-1 0</td>
</tr>
<tr>
<td>1 1 1 1 1</td>
<td>- -</td>
<td>-1 4</td>
</tr>
</tbody>
</table>

3.5 KARIŞTIRICI (SCRAMBLER) VE ÇÖZÜCÜ (DESCRAMBLER)

Kariştirma ve çözme işlemleri modem tarafından sağlanmalıdır. Her iletişim yönü farklı kariştirici kullanılar. İletişimin yönüne göre tanımlanmış ve oluşturulmuş olan çok terimler (3.1) ve (3.2) de gösterilmiştir.

Arayan modem kariştirici çok terimlisi (GPC) = 1 + \( x^{-18} + x^{-23} \)  

(3.1)

Aranan modem kariştirici çok terimlisi (GPA) = 1 + \( x^5 + x^{23} \)  

(3.2)

Demodülatör katında ise alınan karıştırılmış veri, karıştırıcı çok terimlisi ile çarpılarak çözülür.

Dial-up kullanımda arayan modem (originat); karıştırıcı çok terimlisi olarak GPC’yi, çözücü çok terimlisi olarak GPA’yi ve aranan modem (answer); karıştırıcı çok terimlisi olarak GPA’yi, çözücü çok terimlisi olarak GPC’yi kullanır.

Kiralık devre kullanımlarında ise, karıştırıcı ve çözücü çok terimlileri el sıkışma fazında karşılıklı anlaşma ile belirlenir.
4. TMS320C5402 SAYISAL İŞARET İŞLEMCİSİ VE YAZILIM GELİŞTİRME KİTİ

4.1 GİRİŞ

Sayısal işaret işleme teknolojisi, son yıllarda güçlü sabit noktalı (fixed point) ve kayan noktalı (floating point) sayısal işaret işlemcilerin (DSP – Digital Signal Processor) ortaya çıkışı ile önemli gelişmeler kaydetmiştir.

Sabit noktalı sayısal işaret işlemcileri, kayan noktalı sayısal işaret işlemcilerine göre genellikle daha hızlı ve daha ucuz çözümler olarak ortaya çıkar. Sayısal işaret işlemci seçiminde, hız ve fiyat faktörlerinin yanı sıra işlemcinin komut çevrim süresi ile komut kümesinin yapılacak olan uygulamaya uygunluğu da önemli bir faktördür.

Yazılım geliştirme kitleri, seçilen sayısal işaret işlemci ile çalışmada kolaylıklar sağlamaktadır. Böylece, yazılım geliştirme işlemleri makine dili yerine üst düzey bir diyle gerçekleștirilebilir.


4.2 SAYISAL İŞARET İŞLEMCİLER

Sistem işlevselliği için gerekli olmayan bir başarım düzeyi sağlayan numerik gösterim kullanmak, kaynakların verimsiz olarak kullanılması anlamına gelir. Kayan nokta uygulamasının, sabit nokta uygulamasına karşı üstünükleri oldukça fazladır. Bu üstünükler, genellikle bir ürün veya program için yapılan geliştirme çalışmalarında büyük tasarruf sağlar.

Mantis, işlemci tarafından normalize edildiği için kayan noktalı bir sayının hassasiyeti bir program boyunca sabit kalır. Fakat sabit noktalı verinin doğruluğu, saklanan verinin veya bir işlem sonrasında elde edilen değerin büyüklüğine göre değişiklik gösterir. Kayan noktalı sayısal işaret işlemcilerde, mantısın otomatik olarak normalize edilmesi nedeniyle sayıların yuvarlanması, sabit noktalı uygulamalara göre çok daha küçük hatalara yol açar. Çok büyük veya çok küçük sayıların gösterilmesi yeteneğiyle de birleşen bu sabit hassasiyet, filtrelerin gerçeklemesi sorunlarını ortadan kaldırarak kutup veya sıfırların çok daha doğru yerleştirilmesine imkan sağlar.


Bahsedilen bu özelliklerin dışında, aşağıda maddeler halinde verilen ve sayısal işaret işleminin karakteristik özelliklerini oluşturan faktörlerde kullanlacak işleminin seçiminde önemli rol oynarlar. Bunlar;

- Yonga üzerinde bulunan veri ve program bellek miktarları
- Program ve veri için kullanılan veri yolu yapısı ve adresleme kapasitesi
- Yonga üzerinde bulunan ön bellek miktarı
- Yonga üzerinde bulunan çevre birim donanımları
- Sabit ve kayan noktalı aritmetik işlem kapasitesi
- Adresleme modelleri
- Yonga üzerinde bulunan doğrudan bellek erişim (DMA) imkanı
- Harici ve dahili kesme imkanı
- Donanım ve yazılım ile програмланabilen bekleme durumları
- Yonga üzerinde bulunan emülsasyon iskeleleri
- Güç yönetimi ve düşük güç harcaması, şeklinde tanımlanabilir.

Şekil 4.1 TMS320C5402 Sayısal İşaret İşlemci Blok Diyagramı

4.3 TMS320C5402 SABIT NOKTALI SAYISAL İŞARET İŞLEMCİSİ

TMS320C5402 sayısal işaret işlemcisi, TMS320 ailesi içerisinde C5000 grubundan bir işlemcidir. C5000 (C54x) grubunun asıl özelliği, güç tüketiminin verimli bir şekilde sağlanmış olmasıdır.

TMS320C5402, 1.8V çekirdek besleme gerilimi ve aktif durumda 60mW, aylak durumda nW mertebesindeki güç tüketimi ile bu grup içerisinde göze çarpan bir işlemcidir.
Ayrıca TMS320C5402, yüksek performansı, oldukça esnek komut kümesi, gelişkin paralel mimarisi ve fiyat avantajı ile önemli iletişim uygulamalarında rahatlıkla kullanılabilen bir yongadır.

16 bitlik sabit noktalı işlemleri destekleyen işlemci, CMOS teknolojisi ile gerçekleştği için düşük enerji tüketimiyle yüksek fonksiyonel yoğunluk sağlamaktadır. 10 ns' lik tek çevrimli komut kümesi ile saniyede yaklaşık 100 milyon komutu yürütebilmektedir (100 MIPS). Şekil 4.1'de blok diyagramı gösterilmiş olan TMS320C5402 sayısal işaret işlemcinin temel özellikleri aşağıda verilmiştir.

- 4Kx16bit ön yükleme ROM belleği ve 16Kx16bit çift kapılı statik RAM belleği yonga üzerine yerleştirilmiştir. Yonga dışında 64Kx16bit program belleği, 64Kx16bit veri belleği ve 64Kx16bit giriş/çıkış belleği kullanımak suretiyle bellek kapasitesi 192Kx16bit genişletilebilimmaktadır.

- Genişletilmiş program belleği desteği sayesinde 8192x16Kbit'e kadar uzatılmış program belleği adresleme imkânı sağlanabilmektedir.

- 40-bit aritmetik mantık birimi (ALU), 2 adet 40 bit toplayıcı (ACC) ve 40-bit toplayıcı ön belleğin (ACCB) yanı sıra, 16 bit işaretli çarpma izin verebilen 17 bit paralel çarpıcı bulunmaktadır. Tüm çarpma ve toplama işlemleri tek çevrimde yerine getirilmektedir.

- Daha yavaş çalışan hafıza veya giriş çıkış elemanları için yazılım ile 14 makine çevrimine kadar ayarlanabilen belekle durum üretici bulunmaktadır.

- Sağ 0' dan 16' ya kadar ve sola 0' dan 31' e kadar kaydırma kapasiteli 40-bit tampon bellek bulunmaktadır.

- Seri haberleşmeyi destekleyen çevre birimlerle haberleşmek için, tam çift yönlü, tampon bellekli, çok iskeleli sendron seri iletişim arabirimini bulunmaktadır.

- 3.3V (1.8V çekirdek) besleme geriliminde düşük güç tüketimi sağlanmaktadır.

- 144 pin TQFP ve 144 microStar BGA kılıf ile 12x12mm lik entegre boyutundadır.

- TM5320C5402 aynı zamanda aşağıdaki fonksiyonları desteklemektedir,
  (i) 64K paralel giriş/çıkış iskelesi
  (ii) Saat üretimi
  (iii) 16 bit zamanlayıcı
  (iv) Emülsasyon ve test mağlú JTAG sınırlı tarama (IEEE 1149.1)
(v) Yonga üzerinde tarama bazı emülsasyon imkanı
(vi) Genel amaçlı giriş çıkış uçları

4.4 TMS320C5402 TABANLI DSK YAZILIM GELİŞTİRME KİTİ

Bu bölümde, üzerinde V.32 modülatör ve demodülatör tasarım çalışmaları yapılmış olan TMS320C5402 sayısal işaret işlemeci temelli DSP başlangıç kitii (DSP Starter Kit – DSK) tanıtılmaktadır. Çalışma kitii, iki ana parçadan oluşmaktadır.

- Yazılan yazılımların çalıştırıldığı ve üzerinde TMS320C5402 sayısal işaret işlemecisinin bulunduğu donanım kartı.

- Yazılım tasarımının yapıldığı C programlama dili temelli “Code Composer Studio” yazılım geliştirme programı.

4.4.1 DSP Bağlangıç Kitii Donanım Kartı

DSP bağlangıç kitii donanım kartii, çevirici (C) kodunu gerçek hızında test etmek için üzerinde bir TMS320C5402 işlemci barındırır.

Şekil 4.2’de DSP bağlangıç kitii donanım kartının blok diyagramı verilmiştir.

DSP bağlangıç kitii donanım kartı, TMS320C5402 sayısal işaret işlemecisinin en basit uygulama devrelerinden birisidir. Devrede bulunan 64x16Kbit RAM bir çok uygulama için yeterli olmaktadır.

Çekirdek program 256x16Kbit’lik FLASH ROM’a yazılımış durumdadır. Devre üzerinde bulunan TLC320AD50 yonga ses kalitesinde çift kanal giriş/çıkış analog arabirim sağlamaktadır. Tek yonga ile; A/D, D/A dönüştürme işlemlerini 14 bitlik bir dinamik ile sağlamaktadır. Ayrıca; D/A, A/D dönüştürme işlemleri için programlanabilir örnekleme hızına ve sızme imkanlarına sahiptir.

Yazılan yazılımlar bu donanım sayesinde sınırılmaktadır.
Şekil 4.2 TMS320C5402 Tabanlı DSK Yazılım Geliştirme Kiti Donanım Blok Diyagramı
Bu donanım kartının temel özellikleri;

- TMS320C5402 sabit noktalı sayısal işaret işlemci
- 64Kx16 bit SRAM
- 256Kx16 bit FLASH Bellek
- Ses kalitesinde analog veri girişi için arabirim (TLC320AD50)
- Doğrudan mikrofon ve hoparlör bağlanabilme özelliği
- RS232 eşamansız seri arabirim
- Ek kart takarak sistem genişletme imkanı
- Telefon hat arabirimi
- Paralel iskele ile kontrol

4.4.2 “Code Composer Studio” Yazılım Geliştirme Programı


C programlama dilinin kullanıldığı yazılım geliştirme programı, DSP/BIOS adı verilen ve alt düzeydeki DSP yazılımları hazırlar olarak barındıran bir işletim sistemine sahiptir. Böylece; yazılım geliştirme işlemleri, üst düzeyde yapılabilimektedir. Donanım kartı, yazılım geliştirme programı tarafından paralel iskele aracılığı ile kontrol altında tutulur.

Kaynakları daha iyi kullanılan, etkin bir yazılım elde edebilmek için yapılacak tasarımarda dikkat edilmesi gereken kurallar aşağıda belirtilmiştir.
• Bir koşula bağlanmamış komut içeren tekrarlı fonksiyonlar için mümkün olduğu kadar genel amacı döngüler kullanılamalıdır.

• Derleme işleminde sonra, sayısal işaret işlecinin RAM bölgesinde bulunan veri ön bellekleri mümkün olan en az sayıda “veri aktarma” komutu gerektirecek şekilde yeniden düzenlenmelidir.

• TMS320C5402 kodunda, beklemeyi en aza indirmek amacıyla mümkün olduğunca “NOP” komutu kullanılmamalıdır.

• Tamsayı verilerin blok halinde bir yerden diğerine aktarılması için en etkin (en az komut gerektiren) yöntem kullanılamalıdır.

• Hesaplama işlemlerini mümkün olduğu kadar program aksi dışında yapılmalıdır. Diğer bir anlamla, işlem süresinden kazanılabilmek için hesaplama yerine, bellek içerisinde bulunan referans tablolar kullanılamalıdır.

• Program ve veri ön bellekleri, birbiriyile uyumayan en az sayıda bekletme oluşturulacak şekilde farklı bellek bölgelerine yerleştirilmelidir. Böylelikle her bekletme durumu için yaklaşık 5 μsn'lik ilave bir işlem süresinin ortaya çıkması engellenebilir.
5. **V.32 MODÜLATÖR TASARIMI**

5.1 **GİRİŞ**


9600 bit/saniye hızında Normal kodlama ve Kafes kodlama olmak üzere iki değişik kodlama yapısı tanımlanmıştır. Normal kodlamada, gönderilecek karıştırılmış veri katarı farklı kodlamaya tabi tutulur.

Kafes kodlamada; gönderilecek karıştırılmış veri katarı, 4 ardiçk veri bitinden oluşan gruplara bölünür. Her grup içinde bulunan ilk iki bit, farklı olarak kodlanır. Farksal kodlama çıkışları, katlama kodlanmanın girişleri olarak kullanılır ve hata kontrolü için ek bit oluşturulur. Oluşturulan bu ek bit, farksal kodlama çıkışları ve her grup içinde bulunan son iki bit modülasyonun işaret diyagramını oluşturur.

İşaret diyagramlarına göre temel bant işaret üretilir. Alçak geçiren süzgeç ve kompleks modülatör ile de modülasyon işlemi gerçekleştirilir.

Tasarım işlemi, bahsedilen V.32 Modülatör özelliklerini gerçekleştirmek için TMS320C5402 sayisal işaret işlemcisi kullanılarak yapılmış olup; kafes kodlama yapısı temel alınmıştır.
5.2 V.32 MODÜLATÖR YAPISINA GENEL BAKIŞ

![V.32 Modülatör Blok Diyagramı](image)

*Şekil 5.1 V.32 Modülatör Blok Diyagramı*


Karıştırma işleminin amacı veri dizisinde rasgelelik yaratmak, böylece sürekli "1" veya "0" veri dizilerinin oluşumunu önleyerek enerjinin iletim bandında düzgün yayılmasını sağlamaktır. Enerjinin iletim bandında düzgün yayılması demodülasyon işlemini kolaylaştırır.

Kariştirilmiş seri veri katari, kafes kodlama phépsına (TCM) uygun olarak dört bitlik gruplar halinde sırası ile farksal ve katısal kodlama işlemine tabi tutulur. Kafes kodlama işlemi sonucunda ortaya çıkan beş bitlik kodlanmış veri grupları, 32 – nokta işaret diyagramındaki noktalara karşılık gelir. Bu noktalara ait x ve y değerleri modülasyon işleminde kullanılır.

İşaret diyagramının karakteristiğinden dolayı temel bant işaretinde her işaret değişiminde süreksizlikler oluşur. Süreksizliklerin oluşması temel bant işaret bant genişliğinin sonsuz olmasına yol açar.
Bundan dolayı, temel bant işaretinin bant genişliğini sınırlamak amacı ile alçak geçiren süzgeç kullanılmalıdır (Şekil 5.1). Bu işlem; alçaltılmış kosinus süzgecinin, x ve y değerleri ile katlama işlemine tabi tutulmasıyla gerçekleştirilmiştir.

İşlem sonucunda ortaya çıkan x(t) ve y(t) temel bant işaret, kompleks modülatör yardımı ile taşıcı frekansa ötelenir. Kompleks modülatör kullanıma sebebi işaret diyagramına göre oluşturulmuş temel bant işaretinin kompleks olmasıdır.

Katlama ve modülasyon işlemi, sayısal ortamda örnekleme frekansında (14400Hz) yapıldığı için modülde edilmiş işaret alçak geçiren süzgeçten geçirilerek örnekleme frekansı etrafındaki tekrarlı işaretler yok edilir (Şekil 5.7).

Bu bölümde; modülasyon işleminde kullanılan teknikler verilmiş ve tasarım çalışmalar yazılılmış program parçaları ile birlikte ayrıntılı olarak incelenmiştir. Tasarımı gerçekleştirilen modülatör programının akış şeması Şekil 5.2' de görülmektedir.

5.3 SERİ VERİ GİRİŞİ VE KARIŞTIRICI (SCRAMBLER)

5.3.1 Karıştırıcı Özellikleri

Modemin bağlı bulunduğu terminalden gelen seri veriye, modülasyon işlemi uygulanmadan önce karıştırma işlemi yapılmalıdır. Karıştırıcı girişine gelen seri veri, arayan veya aranan modem olma durumuna göre ilgili çok terimliye bölünmelidir. Bu işlemde bölüm katsayısi karıştırıcı katının çiğnımı oluşturur. V.32 standartında, arayan modem karıştırıcı çok terimlisi (GPC), \(1 + x^{-18} + x^{23}\) ve aranan modem karıştırıcı çok terimlisi (GPA), \(1 + x^{-5} + x^{23}\) olarak tanımlanmıştır. Buna göre karıştırıcı çiğnindaki veri katarı, (5.1) ve (5.2) ifadeleri ile gösterilebilir.

Arayan modem ise; \(D_S = D_I \oplus D_I \cdot x^{-18} \oplus D_I \cdot x^{23}\) \hspace{1cm} (5.1)

Aranan modem ise; \(D_S = D_I \oplus D_I \cdot x^{-5} \oplus D_I \cdot x^{23}\) \hspace{1cm} (5.2)

(5.1) ve (5.2) eşitliklerinde;

\(D_S\), karıştırıcı çiğnindaki veri katarı,
\(D_I\), karıştırıcı girişindeki veri katarı,
\(\oplus\), EXOR işlemi,
\(\cdot\), ikili çarpma işlemini göstermektedir.
Şekil 5.2 Modülatör Programı Genel Akış Şeması
Şekil 5.3 Kariştırıcı Katı Blok Diyagramları

Şekil 5.3a ve Şekil 5.3b de kariştırma işleminin yapısı görülmektedir. Karşıtircı çıkışındaki veri katarın (D₅) ardışık olarak 64 bit iletiminden sonra bir sonraki kariştırıcı girişindeki veri (D₇) döndürülür. Bu işlem tüm kariştırıcı uygulamalarında kullanımlıdır. El sıkışma durumlarında ise kariştırıcı kullanılmaz.

5.3.2 Kariştırıcı Tasarımı

Gelen seri veri, “5.3.1 Kariştırıcı Özellikleri” bölümünde açıklanan kurallara göre modülasyon işlemi öncesi kariştırılır. Kariştırıcı programı (Tablo 5.1), Şekil 5.3’de gösterilen kariştırıcı katı blok diyagramları göz önde tutularak yazılmıştır.

5.4 KODLAYICI KATI VE TEMEL BANT İŞARETİ OLUŞTURMA

5.4.1 Kodlayıcı ve Temel Bant İşareti Oluşturma Yapısı


V.32 standardında, yukarıda bahsedilen hata kontrollü kodlama tekniklerini içeren 32 - nokta işaret diyagramının kullanıldığı Kafes kodlama modülasyonu (TCM) veya 16 - nokta işaret diyagramının kullanıldığı Dik genlik modülasyonu (QAM) tanımlıdır. Kodlayıcı girişine gelen dörtlü bit grupları, kullanılan işaret diyagramı üzerinde bir noktaya karşılık düşer. Her nokta, gerçek (x) ve sanal (y) olmak üzere iki değerden
oluştur. Her iki modülasyon tekniğinde de taşıyıcı işaret, birbirinden bağımsız bu iki bilgi kanalı (x ve y) tarafından modüle edilir.


V.32 Kafes kodlayıcı iki ayrı fonksiyonel blok olarak tanımlanabilir;

- Farksal kodlayıcı
- Katlamalı kodlayıcı

Dört bitlik gruplar halinde kodlayıcı girişine gelen veriler (Q1, Q2, Q3, Q4), farklı ve katlamalı kodlama işleminde tabi tutularak başına bit uzunluğunda çıkış verileri (Y0, Y1, Y2, Q3, Q4) elde edilir. Kodlayıcı çıkış verisi, kodlayıcı giriş verisinden daha uzundur. Bunun nedeni kodlayıcı çıkış verilerinin hata düzeltme bilgisini içermemesidir (Şekil 5.4).

![Şekil 5.4 V.32 Kafes Kodlayıcı Blok Diyagramı](image-url)


Farksal kodlama, haberleşme kanalındaki 90°, 180° ve 270° faz belirsizliklerinin etkilerini gidermek amacıyla katlamalı kodlama girişlerine uygulanır. Farksal kodlama işlemi, (5.3) ve (5.4) matematiksel ifadeleri ile tanımlanır (Tablo 3.2).

\[ Y_{1n} = Q_{1n} \oplus Y_{1n-1} \]  
(5.3)

\[ Y_{2n} = (Q_{1n} \cdot Y_{1n-1}) \oplus Y_{2n-1} \oplus Q_{2n} \]  
(5.4)

Şekil 5.4’de görüldüğü gibi, katlamalı kodlayıcının girişleri, farksal olarak kodlanmış giriş verileridir. Farksal kodlamının nedeni, haberleşme kanalındaki faz dönmemelerinin neden olduğu hataların çoğalmasına izin vermektedir. Bilgi dizisi, faz dönmemelerinin oluşturduğu noktalardaki hatalar dışında alıcı tarafından tekrar oluşturulur.

Katlamalı kodlayıcı çıkışında oluşan Y0 verisi sadece hata düzeltme bilgisi taşımaktadır. Katlamalı kodlayıcının yapısı Şekil 3.2’ de ayrıntılı olarak görülmektedir. Bu yapıda, AND ve EXOR sayısal ifadeleri ile birbirlerine ilişkilendirilmiş geçime gözleri bulunmaktadır. Bu geçirmeye gözler S0, S1 ve S2 olarak tanımlanmıştır (Şekil 5.4). Şekil 3.2’ de verilen katlamalı kodlama yapısından, Şekil 5.5’de görülen sonlu durum diyagramı oluşturulur.


Örneğin; (0.0.1) geçikme durumlarından sadece (0.0.0), (0.1.0), (1.0.0) ve (1.1.0) durumlarına geçilebilmektedir.
Şekil 5.5 V.32 Katlamalı Kodlayıcı Sonlu Durum Diyagramı

Bu kodlama, kafes yapısı olarak adlandırılır. Bu kısıtlama sayesinde, kafes kodlamada bit - hata oranı azdır. Kodlayıcı çiğnındaki 5 bitlik kodlanmış verinin; 32 – nokta işaret diyagramında gösterdiği noktaya ait x ve y temel bant işaret bileşenleri, süzme ve modülsyon işlemlerinin gerçekleştirilğini fonksiyonlara verilir.

5.4.2 Kodlayıcı Tasarımı

Kodlayıcı tasarımı, “5.4.1 Kodlayıcı ve Temel Bant İşareti Oluşturma Yapısı” bölümünde özellikleri verilen Kafes kodlama yapısı (TCM)' na göre yapılmıştır. Kodlayıcı programı, standart V.32 kodlayıcı işlemlerini yerine getirerek dört bitlik gruplar halinde gelen veriyi beş bitlik çıkış verisi (y0, y1, y2, q3 ve q4) halinde kodlar. Dört bitlik giriş verisi (q1, q2, q3 ve q4), scrambled_data_table' dan alınarak programa parametre olarak atanır. Kodlayıcı çıkış verileri ise signal_element_mapping_input dizisine yazılır. Farksal ve katlamalı kodlama işlemlerinde kullanılan gecikme elemanları, diff_encoder_log_table ve conv_encoder_log_s_table dizilerinde tutulur. Kullanılan bu gecikme
elemanlarının başlangıç değerleri "0" olarak alınmıştır. q3 ve q4 giriş verileri hiçbir işleme tabi tutulmadan çıkış verisi olarak kullanılır. q1 ve q2 giriş verileri ise (5.3) ve (5.4) denklemeleri uyarınca farklı kodlama işleminde tabi tutularak y1 ve y2 çıkış verileri oluşturulur. Bu işlem sırasında diff_encoder_log_table dizisinde tutulan geçikme değerleri kullanılır.

Katlamalı kodlama işlemi, farklı kodlama işleminde oluşturulan y1 ve y2 çıkış verileri ile conv_encoder_log_s_table dizi değerlerine göre Şekil 5.5' de gösterilmiş sonlu durum diyagramında gösterilen şekilde y0 çıkış verisini oluşturur.
Sonlu durum diyagramına göre yazılımış programın sadece 000 durumundan y1 ve y2 verilerine göre diğer durumlara geçiş verilen program parçasında (Tablo 5.2) gösterilmiştir. Bu yazılmda, matris tablo yapısı yerine durum geçiş yapısı kullanılmıştır. Böylece kodlayıcı program çıkış veri grubu beş bitten oluşur ve signal_element_mapping_input dizininde saklanır. Kodlayıcı çıkış verisini, giriş verisinden daha uzun olmasını sebebi çıkış verisinin hata düzeltme bilgisini içermesidir. Tablo 5.2' de kodlayıcı programının listesi verilmiştir.

Tablo 5.2 V.32 Kodlayıcı Programı

```c
s16 signal_element_mapping_input[5];
s16 diff_encoder_log_table[2];
s16 conv_encoder_log_s_table[3];

/*==================================================================*/
/* q1, q2, q3 ve q4 giriş değerlerine, difransiyel kodlama ve konvolusyonal kodlama yapılırak */
/* y0, y1, y2, q3 ve q4 çıkış değerleri elde edilir ve bu değerler */
/* "signal_element_mapping_input" dizisine sırayla oturulur. */
/*==================================================================*/
/* signal_element_mapping_input[0] = y0, signal_element_mapping_input[1] = y1 */
/* signal_element_mapping_input[4] = q4 */
/* conv_encoder_log_s_table[3] = s0, s1, s2 */
/*==================================================================*/

static void Encoder_532(s16 q1, s16 q2, s16 q3, s16 q4 )
{
    s16 conv_encoder_log_s_table_new[3] = {0,0,0};
    signal_element_mapping_input[3] = q3;  /* q3 direkt kullanılıyor */
    signal_element_mapping_input[4] = q4;  /* q4 direkt kullanılıyor */
    signal_element_mapping_input[1] = q1 ^ diff_encoder_log_table[0];  /* y1 oluşturulma */
    signal_element_mapping_input[2] = (q1 & diff_encoder_log_table[0]) ^ (diff_encoder_log_table[1] ^ q2);
    if((conv_encoder_log_s_table[0] == 0 ) && (conv_encoder_log_s_table[1] == 0 ))
    {
        conv_encoder_log_s_table[2] = 0;
        if((signal_element_mapping_input[1] == 0 ) && ( signal_element_mapping_input[2] == 0 ))
        {
            conv_encoder_log_s_table_new[0] = 0;
            conv_encoder_log_s_table_new[1] = 0;
        }
        if((signal_element_mapping_input[1] == 1 ) && ( signal_element_mapping_input[2] == 0 ))
        {
            conv_encoder_log_s_table_new[0] = 0;
            conv_encoder_log_s_table_new[1] = 1;
        }
        {
            conv_encoder_log_s_table_new[0] = 0;
            conv_encoder_log_s_table_new[1] = 1;
        }
    }
}
```
5.4.3 Temel Bant İşaret Oluşturma Programı

signal_element_mapping_input dizisinde saklanan kodlayıcı program çıkış veri grubu, temel bant işaretinin oluşturulmasında kullanılar. 5 bitlik kodlanmış verinin; 32 – nokta işaret diyaframında gösterdiği noktaya ait x ve y temel bant işaret bileşenleri (Tablo 3.3), Trellis_X_table ve Trellis_Y_table’da sabit olarak kullanılmıştır. Sızılmış temel bant işaretinin 4 sembol iletim aralığı boyunca gönderilmesi, oluşturulun temel bant işaretlerinin bellekli olarak saklanmasını gerektirmektedir. Bunun için; TemelBand_LocateX ve TemelBand_LocateY tabloları kullanılır.

Tablo 5.3 Temel Bant İşaret Oluşturma Programı

```c
const s16 Trellis_X_table[32] = {
    -4,0,0,4,0,0,-4,2,2,2,2,-2,-2,2,2,2,-2,-2,2,2,2,-2,0,0,0,0,0,0,
};
const s16 Trellis_Y_table[32] = {
    1,3,1,1,-1,3,1,1,-1,3,1,1,-1,3,1,1,-1,3,1,1,-1,3,1,1,-1,3,1,1,-1,3,1,1,-1,
};
#define RaiseCos_KuyrukSayisi 4
s16 TemelBand_LocateX[RaiseCos_KuyrukSayisi] = {0,0,0,0};
s16 TemelBand_LocateY[RaiseCos_KuyrukSayisi] = {0,0,0,0};
s16 TemelBandX=0, TemelBandY=0;
/* Aynı anda saklanması gerekli 4 adet X ve Y vardır. Her baudda biri birer. */
/* Onun yerine bir başka X ve Y gönderilmeye bıraklan. */

/*********************************************************************************
static void TemelBand_Create() {
    s16 TemelBand_i,TemelBand_j;
    TemelBand_i = (signal_element_mapping_input[0]*16) + (signal_element_mapping_input[1]*8) +
                   (signal_element_mapping_input[2]*4) + (signal_element_mapping_input[3]*2) +
                   signal_element_mapping_input[4];
    TemelBandX = Trellis_X_table[TemelBand_i];
    TemelBandY = Trellis_Y_table[TemelBand_i];

    for (TemelBand_j= RaiseCos_KuyrukSayisi-2 ; TemelBand_j>=0 ; TemelBand_j-- ) {
        TemelBand_LocateX[TemelBand_j+1] = TemelBand_LocateX[TemelBand_j];
        TemelBand_LocateY[TemelBand_j+1] = TemelBand_LocateY[TemelBand_j];
    }
    TemelBand_LocateX[0] = TemelBandX;
    TemelBand_LocateY[0] = TemelBandY;
} /**********************************************************************************/
```
5.5 SÜZME VE MODÜLASYON İŞLEMİ

5.5.1 Süzme ve Modülasyon Özellikleri

Kodlayıcı çıkışında oluşturulan beş bitlik veri (Y₀, Y₁, Y₂, Q₃ ve Q₄), 32 – nokta işaret diyalogramında bir noktaya karşılık gelir. Her nokta x ve y olmak üzere iki sayı veya x+iy gibi bir kompleks sayı ile ifade edilebilir. Dolayısıyla taşıyıcı işaret modüle edecek sayların kompleks yapıda olması modülatörün kompleks olmasını gerektirmektedir.


Kompleks modülatör, yükseltilmiş koşnüs sızgelen çıktından gelen süzülmüş temel bant işaretlerini taşıyıcı frekansına öteler. Kullanılan kompleks modülatör Şekil 5.6’da gösterilmiştir. Modülatörde gösterilen x(t) ve y(t) girişleri alçak geçiren sızgelen çıkışlardır. Modülatör, taşıyıcı frekansına (fᵣ) merkezli toplam ve fark frekansları oluşturarak temel bant işaretini öteler.

![Şekil 5.6 V.32 Kompleks Modülatör](image-url)
Modülasyon işlemi matematiksel olarak incelenir ve \( x(t) \) ile \( y(t) \) işaretleri tek bir frekans bileşeninden ibaret kabul edilirse (5.5);

\[
x(t) = y(t) = \sin \omega_b t
\]  
(5.5)

(5.6)'da gösterilen modülatör çıkış ifadesi elde edilir.

\[
z(t) = \sin \omega_b t \cdot \cos \omega_c t - \sin \omega_b t \cdot \sin \omega_c t
\]  
(5.6)

\[
\sin \omega_b t \cdot \cos \omega_c t = \frac{1}{2} \cdot [\sin (\omega_c+\omega_b)t - \sin (\omega_c-\omega_b)t]
\]  
(5.7a)

\[
\sin \omega_b t \cdot \sin \omega_c t = \frac{1}{2} \cdot [\cos (\omega_c-\omega_b)t - \cos (\omega_c+\omega_b)t]
\]  
(5.7b)

Ayrıca (5.7a) ve (5.7b) trigonometrik özelliklerinden yararlanarak \( z(t) \) tekrar oluşturulur (5.8).

\[
z(t) = \frac{1}{2} \cdot [\sin (\omega_c+\omega_b)t - \sin (\omega_c-\omega_b)t] - \frac{1}{2} \cdot [\cos (\omega_c-\omega_b)t - \cos (\omega_c+\omega_b)t]
\]

\[
= \frac{1}{2} \cdot \sin (\omega_c+\omega_b)t - \frac{1}{2} \cdot \sin (\omega_c-\omega_b)t - \frac{1}{2} \cdot \cos (\omega_c-\omega_b)t + \frac{1}{2} \cdot \cos (\omega_c+\omega_b)t
\]

\[
= \frac{1}{2} \cdot \sin (\omega_c+\omega_b)t - \frac{1}{2} \cdot \sin (\omega_c-\omega_b)t - \frac{1}{2} \cdot \cos (\omega_c-\omega_b)t + \frac{1}{2} \cdot \cos (\omega_c+\omega_b)t
\]  
(5.8)

(5.8) ifadesinden de görüleceği gibi \( \omega_c+\omega_b \) ve \( \omega_c-\omega_b \) fark frekanslarının sinüs fonksiyonlarını içermektedir. Tüm modülasyon işlemleri sayısal olarak yapılacağı için örneklemle frekansı kullanılabaktır. Bundan dolayı, örneklenmiş frekansın etrafında tekrarlanmış spektrumlar oluşur (Şekil 5.7c).

![Image](a) Temel Bant İşaret Frekans Spektrumu

![Image](b) Taşıyıcı Frekansa Ötelenmiş İşaret Frekans Spektrumu
Şekil 5.7 V.32 Modülatör İşaret Frekans Spektrumları

Şekil 5.7a’ da (5.5) eşitliğinde tanımlanan temel bant işaretinin frekans spektrumu görülmektedir. Şekil 5.7b’ de (5.6) ve (5.8) eşitliklerinde tanımlanan modülasyonu gerçekleştirmiş işaretin frekans spektrumu görülmektedir. Şekil 5.7c’ de görülen örnekleme frekansı etrafındaki işaretler ise modülatör çıktısında alçak geçiren süzgeçten geçirilerek atılmıştır. Şekil 5.7’ deki A terimi, genliği ifade etmektedir.

5.5.2 Temel Bant ve Modülasyonlu İşaret Analizi

Modülatör tasarımında, sayısal işaret işleme nedeniyle süzme ve modüle işlemlerini sayısal olarak örnekleme frekansında yapılmıştır. V.32 modülatörde iletişim kanalı 600 – 3000Hz’ dir. Shannon teoremi uyarınca örnekleme frekansı minimum 2 . 3000 = 6000Hz alınabilir. Kullanılacak örnekleme frekansı 14400Hz olarak seçilmiştir. V.32 modemde, iletilen işaret hızı 2400 baud, taşıyıcı frekans ise 1800Hz’ dir. Örnekleme frekansının bu değerlerin tam katı olarak seçilmesi modülatör tasarımında kolaylık sağlamaktadır.

Temel bant işaretinin bandını sınırlandırılmasına düşüş parametresi 0.75 olan yüksektilmiş kosinus süzgeci kullanılmaktadır. Yüksektilmiş kosinus süzgecinin matematiksel ifadesi (5.9) denkleminden görülmektedir.

\[
h(t) = \frac{\sin \pi.f_N.t}{\pi.f_N.t} \cdot \frac{\cos \pi.\alpha.f_N.t}{1 - 4. f_N^2.\alpha^2.t^2} \quad (5.9)
\]

\( f_N = \frac{1}{T_N} \): 2400 işaret/saniye (baud)
\( \alpha : 0.75 \) (düşüş parametresi)
Şekil 5.8' de, (5.9) denkleminde matematiksel ifadesi verilen yükseltilmiş cosinus sütütnin zaman cevabı görünmektedir. Tₙ modülasyon periyodunu göstermektedir. Modülasyon periyodu, Tₙ = 1/2400Hz⁻¹ dir. Tₛ = 1/14400Hz⁻¹ olup, örnekleme periyodunu gösterir.

Kodlayıcı çıkışındaki beş bitlik verinin gösterdiği vektörün taşıyıcı frekansına ötelenmesi ile oluşan m(t) işaret (5.10) denkleminde gösterilmiştir.

\[ m(t) = A(t) \cdot \cos (2fₚt + φ(t)) \]  \hspace{1cm} (5.10)

\( fₚ \): 1800Hz taşıyıcı frekansı
\( A \): 32 – nokta işaret durum diyagramına göre vektör genliği
\( φ \): 32 – nokta işaret durum diyagramına göre vektör açısı

m(t) işaretinde bulunan A ve φ parametreleri kodlayıcı çıkışındaki veri gruplarına göre değişmekte ve süreklilik göstermektedir. Bundan dolayı m(t) işaretinde süreklilikler oluşmaktadır. Bunu engellemek için m(t) işaret alçak geçiren sütütnen geçirilir. Çıkış işaret z(t), (5.11)'de gösterilmiştir.

\[ z(t) = h(t) * m(t) \]  \hspace{1cm} (5.11)

(5.10) denkleminde verilen m(t) temel bant işaretindeki cosinus bileşeninin trigonometrik açılıından (5.12) ifadesi elde edilir.

\[ A \cdot \cos (\omegaₚt + φ) = A \cdot [\cos \omegaₚt \cdot \cos φ - \sin \omegaₚt \cdot \sin φ] \]
\[ = A \cdot \cos φ \cdot \cos \omegaₚt - A \cdot \sin φ \cdot \sin \omegaₚt \]
\[ = x \cdot \cos \omegaₚt - y \cdot \sin \omegaₚt \]  \hspace{1cm} (5.12)
Böylece çıkış işaretleri \( z(t) \), (5.13) ifadesindeki gibi yazılır. (5.13) denklemindeki de
görulebileceği gibi, 32 - nokta işaret diyagramındaki \( x \) ve \( y \) değerleri kompleks
modülatörden geçirilerek modülsyonoynlu işaret oluşturulur.

\[
x(t) = x \cdot \cos \omega_c \cdot t - y \cdot \sin \omega_c \cdot t
\]

\[
y(t) = y \cdot \cos \omega_c \cdot t - x \cdot \sin \omega_c \cdot t
\]

(5.13)

\( x \) ve \( y \) değerleri, kompleks modülatörden geçirilmeden önce yükseltilmiş kosinus
süzgecinden geçirilir. Bu işlem, \( x \) ve \( y \) değerlerinin \( h(t) \) ile katlama işlemine tabi
tutulmasıdır (5.14).

\[
x(t) = x \cdot h(t)
\]

(5.14a)

\[
y(t) = y \cdot h(t)
\]

(5.14b)

Katlama işlemi yapılmış ve kompleks modülatörden geçirilmiş modülsyonoynlu işaret
(5.15)’de görülmektedir.

\[
z(t) = (x \cdot h(t)) \cdot \cos \omega_c \cdot t - (y \cdot h(t)) \cdot \sin \omega_c \cdot t
\]

(5.15)

5.5.3 Süzgeç ve Modülatör Program Tasarımı

5.5.3.1 Program Tasarımı Yöntemi

Modülsyonoynlu işaret denkleminde (5.15) görülen \( x \) ve \( y \); \( 0, \pm 1, \pm 2, \pm 3 \) ve \( \pm 4 \)
değerlerini alabilir (Şekil 3.3). \( x \) ve \( y \) değerleri; sembol iletim aralığı boyunca
değişmediği ve sabit bir tam sayı olarak kaldıği için, yükseltilmiş kosinus sözgeci ile
\( x \) ve \( y \) değerleri arasında yapılan katlama işlemi de basit çarpma işleminin
dönüşmüştür.

Şekil 5.8’de görülen yükseltilmiş kosinus sözgecinin zaman cevabı, sonsuz
uzunluktadır. Teorik olarak, \( x \) ve \( y \) değerlerinin sonsuz uzunluğa kuyrukla sahip
bu süzgeç ile katlama işlemini tabi tutulması gerekir.

Yükseltilmiş kosinus süzgeç zaman cevabının karakteristiğinden dolayı sonsuza
giderken kuyruklar kütülmeke ve süzgeç işleminde önemi kaybetmektedir.
Bundan dolayı, modülatör tasarımında yükseltilmiş kosinus süzgeci \([-2T_s, 2T_s]\)
aralığında kullanılmıştır. Bu aralıktaki yükseltilmiş kosinus sözgecinin \( T_s \) periyodu
ile örneklenmiş değerleri, tablo halinde program tarafından kullanılır.

Yükseltilmiş kosinus sözgecinin \([-2T_s, 2T_s]\) aralığında \( T_s \) periyodu ile örneklenmiş
degerleri Tablo 5.4’de gösterilmiştir. Tablo değerleri, Şekil 5.8’deki örnek değerleri
göstermektedir.
Tablo 5.4 Örneklenmiş Yükseltilmiş Kosinüs Süzgeç Değerleri

<table>
<thead>
<tr>
<th>Örnek No</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>[-2TN, -TN]</td>
<td>0</td>
<td>-0.0878</td>
<td>-0.1660</td>
<td>-0.2128</td>
<td>-0.2070</td>
<td>-0.1347</td>
</tr>
<tr>
<td>[-TN, 0]</td>
<td>0</td>
<td>0.1914</td>
<td>0.4140</td>
<td>0.6367</td>
<td>0.8261</td>
<td>0.9531</td>
</tr>
<tr>
<td>[0, TN]</td>
<td>0.9980</td>
<td>0.9531</td>
<td>0.8261</td>
<td>0.6367</td>
<td>0.4140</td>
<td>0.1914</td>
</tr>
<tr>
<td>[TN, 2TN]</td>
<td>0</td>
<td>-0.1347</td>
<td>-0.2070</td>
<td>-0.2128</td>
<td>-0.1660</td>
<td>-0.0878</td>
</tr>
</tbody>
</table>

Her sembol iletim aralığında belirlenen x ve y değerleri, örneklenmiş süzgeç değerleri ile çarpma işlemine tabi tutulur. Yükseltilmiş kosinüs süzgecinin [-2TN, 2TN] aralığında kullanılması nedeniyle çarpma işlemi 4 sembol iletim aralığı boyunca yapılır. Bundan dolayı, modülsasyon işlemine tabi tutulacak herhangi bir m. sembol iletim aralığındaki x(t) temel bant işaretleri,

m. x değerinin genliğini modüle ettiği süzgecin [-2TN, -TN],

m-1. x değerinin genliğini modüle ettiği süzgecin [-TN, 0],

m-2. x değerinin genliğini modüle ettiği süzgecin [0, TN],

m-3. x değerinin genliğini modüle ettiği süzgecin [TN, 2TN] aralığındaki değerlerinin toplamından oluşacaktır.

Aynı işlemler y(t) temel bant işaretleri içinde geçerlidir.

Bu şekilde oluşturulunan x(t) ve y(t) değerleri, her periyot için 8 örnek (T_S/T_C) alınarak oluşturulmuş sinüs ve kosinüs tabloları ile modüle işleme tabi tutulur (5.15). Kullanılan sinüs ve kosinüs tabloları Tablo 5.5’de gösterilmiştir.

Tablo 5.5 Örneklenmiş Sinüs ve Kosinüs İşaret Değerleri

<table>
<thead>
<tr>
<th>Sinüs işaretleri</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.707</td>
<td>1</td>
<td>0.707</td>
<td>0</td>
<td>-0.707</td>
<td>-1</td>
<td>-0.707</td>
<td>0</td>
</tr>
</tbody>
</table>

Kosinüs işaretleri

<table>
<thead>
<tr>
<th>Kosinüs işaretleri</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.707</td>
<td>1</td>
<td>0</td>
<td>-0.707</td>
<td>-1</td>
<td>-0.707</td>
<td>0</td>
<td>0.707</td>
<td></td>
</tr>
</tbody>
</table>

5.5.3.2 Temel Bant İşaret Süzgeç ve Modülsasyon Programı

Programlar, örneklemle frekansı olan 14400Hz frekansında çalışır ve “5.5.3.1 Program Tasarım Yöntemi” bölümündeki yöntemler kullanılarak tasarlanmıştır. Temel Bant İşaret Süzme Programı sonucunda elde edilen Filtered_xt ve Filtered_yt değerleri modülsasyon programı tarafından kullanılır.

Tablo 5.6’ da Temel Bant İşaret Süzme Programı, Tablo 5.7’ de ise Modülsasyon programı gösterilmiştir.
Tablo 5.6 Temel Bant İşaret Süzme Programı

```
static void TemelBand_Create_withLPF()
{
    s16 TBCwl_i, TBCwl_j=0, TBCwl_k=0;

    for ( TBCwl_i = 0 ; TBCwl_i < RaiseCos_KuyrukSayısı ; TBCwl_i++ )
    {
        TBCwl_j = TBCwl_j + ( TemelBand_LocateX[TBCwl_i] * 
                           RaiseCos_SampleTable[(TBCwl_i * Baud_SampleNumber) + TBCwl_Sayac1] );

        TBCwl_k = TBCwl_k + ( TemelBand_LocateY[TBCwl_i] * 
                           RaiseCos_SampleTable[(TBCwl_i * Baud_SampleNumber) + TBCwl_Sayac1] );
    }

    if (TBCwl_Sayac1 < 5)
    {
        TBCwl_Sayac1++;
    }
    else
    {
        TBCwl_Sayac1=0;
        Filtered_xt = TBCwl_j;  /* x(t) = TBCwl_j */
        Filtered_yt = TBCwl_k;  /* y(t) = TBCwl_k */
    }
}
```

Filtered_xt ve Filtered_yt, (5.13) eşitligine göre modülasyon işlemine tabi tutulur. Kullanılan sinüs ve kosinus tabloları Tablo 5.5' de gösterilmiştir. Mod_zt, (5.13) eşitliğindeki z(t)' yi ifade etmektedir.

Tablo 5.7 Modülasyon Programı

```
static void Modulation()
{
    s16 Modulation_i;

    Mod_zt = ( Filtered_xt * TekPerCos_SampleTable[TBCwl_Sayac2] ) - 
              ( Filtered_yt * TekPerSin_SampleTable[TBCwl_Sayac2] );

    Mod_zt1 = ( Filtered_xt * TekPerSin_SampleTable[TBCwl_Sayac2] ) + 
              ( Filtered_yt * TekPerCos_SampleTable[TBCwl_Sayac2] );

    for ( Modulation_i=Modulated_Signal_TamponUz-2 ; Modulation_i>=0 ; Modulation_i-- )
    {
        Modulated_Signal[Modulation_i+1] = Modulated_Signal[Modulation_i];
        Modulated_Signal[Modulation_i+1] = Modulated_Signal1[Modulation_i];
    }

    Modulated_Signal[0] = Mod_zt;

    if (TBCwl_Sayac2 == 7)
    {
        TBCwl_Sayac2=0;
    }
    else
    {
        TBCwl_Sayac2++;
    }
}
```
5.5.3.3 Modülasyonlu İşaret Süzme Programı

Modülatör tasarımıının sayısal ortamda yapımış olması örnekleme frekansını kullanılamasını gerektirir. Bu nedenden ötürü, modülasyon işlemi sonrası örnekleme frekansı etrafında tekrarlanmış spektrumlar oluşur (Şekil 5.7c).

Modülatör çıkışında semboller alıcı geçiren sızgıcten geçirilerek 3500Hz üzerindeki frekans bileşenleri sızlandırılır. Alıcı geçiren sızgıc olarak, dördüncü dereceden Butterworth sızgıc kurulmuştur. Alıcı geçiren sızgıcın çıkış ifadesi (5.16)'da verilmiş ve Şekil 5.9’da gösterilmiştir. Bu ifade kullanılarak tasarlanan sızgıc e ait alt program Tablo 5.8’de listelenmiştir.

\[
\]  
(5.16)

\[\text{Şekil 5.9} \quad 14400Hz örnekleme hızında 4. Dereceden 3500Hz köşe frekanslı alıcı geçiren Butterworth sızgıc blok diyagramı\]

**Tablo 5.8 Butterworth Sızgıc Programı**

/*****************************/
/* modülasyonlu işaretin örnekleme frekansındaki tekrarları yok etmek için */
/* 3500Hz LPF kullanılmıştır. */
/*****************************/
static void butterworth_lpf(float butterworth_invalue)
{
    int i;

    for (i=0 ; i<butterworthlpf_sifir_sa ; i++)
        butterworthlpf_xv[i] = butterworthlpf_xv[i+1];

    butterworthlpf_xx[butterworthlpf_sifir_sa] = butterworth_invalue / butterworthlpf_kazanc;

    for (i=0 ; i<butterworthlpf_kutup_sa ; i++)
        butterworthlpf_yv[i] = butterworthlpf_yv[i+1];
butterworthlpf_yv[butterworthlpf_kutup_sa] =
    ( -0.0178922931 * butterworthlpf_yv[0] ) + ( 0.0227985409 * butterworthlpf_yv[1] ) +
    ( -0.4897483812 * butterworthlpf_yv[2] ) + ( 0.1084764138 * butterworthlpf_yv[3] );

    butterworthlpf_outvalue = butterworthlpf_yv[butterworthlpf_kutup_sa];
}/*********************************************************************************/

5.6 SONUÇ

Beşinci bölümdde; 9600 bit/saniye veri iletişim hızında, ITU V.32 standardı ile tanımlanan kurallar çerçevesinde çalışan modülatör tasarımını açıklanmıştır.

Yapılan yazılım çalışmalarında, TMS320C5402 sayısall işaret işlemcisine ait program geliştirme kitabı kullanılmıştır. Modülatör yazılımı, " C " programlama dilinde tasarlanmış olup, makine diline çevrilerek DSK kartı üzerinde çalıştırılmıştır.

Tasarlanan modülatör yazılımı, 64 bit uzunluğunda olan ve transmit_data_table tablosunda tutulan değişik test dizileri ile çalıştırılır. Program, tablodaki sayısal verileri 9600Hz hızında alarak modüle eder ve 14400Hz örneklememe hızında modülasyonlu sembollerı modulated_sembol_table tablosuna yazar.
6. **V.32 DEMODÜLATÖR TASARIMI**

6.1 **GİRİŞ**

İletimi gerçekleştirecek sayısal veri, Bölüm 5’ de açıklanan biçimde modülsyon işlemine tabi tutulur. Modülsyonlu işaret, haberleşme kanalından geçerek alcı tarafından bulunan demodülatör katına gelir. Demodülatör katının görevi, modülsyonlu işaretten ilerlemi gerçekleştirdiğin sayısal veriyi elde etmektedir.


Demodülatör tasarımında sayısal işaret işleme teknikleri kullanılmıştır. Bununla birlikte; alınan işaret taşıyıcısına faz kilitli yerel taşıyıcının üretildiği tutağı demodülsyon tekniklerinden olan Hilbert dönüşümü, demodülsyon katında uygulanmıştır.

6.2 **V.32 DEMODÜLATÖR YAPISINA GENEL BAKIŞ**

Demodülsyon yapısının şekillenmesindeki ana etken; alınan modülsyonulu işaretin \(Z(t)\), temel banda indirilmesinde kullanılan yöntemdir. Sayısal işaret işleme tekniklerinin kullanılacağı olması, tasarım düşünülen demodülsyon katında Hilbert dönüşümünün kullanılmasını gerektirir. Bu dönüşümün temel prensibi; bir işaretin frekans bileşenlerinin fazını \(-90^\circ\) ötelemek ve kendisine dik başka bir işaret
oluşturarak kompleks demodülatör yardımyla demodülasyonu gerçekleştirmektir. Bu temel prensipten dolayı, Hilbert dönüşümünün sayısal işaret işleme teknikleri ile gerçekleştirilmesi kolay olmaktadır. Şekil 6.1' de Hilbert dönüşüm teknliğinin kullanıldığı demodülasyon katı blok diyagramı görülmektedir.

Şekil 6.1 Hilbert Dönüşümlü Demodülatör Katı Blok Diyagramı

Süzme ve otomatik kazanç kontrolü yapılan işaret (Z(t)); Hilbert dönüşümü uygulanarak, oluşturulan yerel taşıyıcı frekans ile demodüle edilir. Dönüşüm ve demodüle işlemi sonrasında, uyarlamalı dengeleyici ile kullanılan haberleşme kanalı ve süzgeçlerden kaynaklanan bozulmalar düzeltilir. Gelen işaretli örneklemede kullanılan frekans ve faz uyumsuzluklarından dolayı oluşan hata, daha önceki vektör hatalarına göre belirlenmiş olan $\alpha$ açısı kadar işaretli döndürmekle giderilmeye çalışılır. $\alpha$, taşıyıcı çıkarma süzgeci çıkışındaki hata açısıdır. İşaret döndürme işlemi, daha önceki açı hatalarına göre yapıldığı için, hata tam olarak ortadan kaldırılamayacaktır.

Karar verici; algılanan X ve Y vektör izdüşüm değerleri ve daha önce karar verilmiş olan X' ve Y' değerleri yardımyla, yeni X' ve Y' değerlerine karar verir. Algılanan X ve Y izdüşüm değerli vektör ile karar verilen ideal vektör arasındaki hata açısı, bir sonraki vektör döndürme işleminde kullanılmak üzere taşıyıcı çıkarma süzgeceine verilir.
**Şekil 6.2 Demodülatör Programı Genel Akış Şeması**

Alınan vektör ile ideal vektör arasındaki, \(|X-X'|\) ve \(|Y-Y'|\) fark değerleri ise ters dönüştürücü tarafından kullanılır. Ters dönüştürücü, uyarlamalı dengeleyicinin katsayılarnını belirlemekte kullanılan parametreleri oluşturur.

Hilbert dönüştümünün kullanıldığı demodülasyon işlemi sonrası, karar verilen \(X'\) ve \(Y'\) değerlerinin karşılığı olan beş bitlik veriler bulunur. Karar verme işleminde ve dört bitlik faydalı verinin oluşturulmasında Viterbi algoritması kullanılmaktadır.
Karar verici çıktındaki beş bitlik veri grubu, ters farksal dönüştüme tabi tutularak dört bitlik veri grubu oluşturulur. Bu grup, çözücü çok terimlisi yardımıyla çözülecek faydali veri alınmış olunur.

Genel prensipleri yukarıda açıklanan demodülatör kavramına ait yazının akış şeması Şekil 6.2’ de verilmiş olup diğer bölümlerde ayrıntılı olarak incelenmiştir.

6.3 HILBERT DÖNÜŞÜMÜ VE KOMPLEKS DEMODÜLATÖR

6.3.1 Hilbert Dönüşüm Yöntemiyle Demodülyasyon

![Hilbert Dönüşüm Şeması](image)

Şekil 6.3 Hilbert Dönüşümü ve Demodülatör Blok Diyagramı

Şekil 6.3’de, Hilbert dönüştümü ile gerçekleştirilen demodülyasyon işlemi blok diyagramı görülmektedir.

Modülyasyon işlemi ile taşıyıcı frekansa ötelenmiş işaret, bu dönüşüm ile temel banda indirilir. Z(t) işaret, süzülmiş ve otomatik kazanç kontrolünden geçerek alınmış işaretdir (6.1). Bu işaretin niteliği, Bölüm 5’ de açıklanmıştır.

\[ Z(t) = Z_p = x \cdot \cos \omega_C t - y \cdot \sin \omega_C t \]  

(6.1)

Z(t) işaretinin, -90° faz kaydırlaraka Hilbert dönüşümü alınmış ifadesi, (6.2)’ de \( Z_q \) olarak gösterilmiştir.

\[ Z_q = x \cdot \sin \omega_C t + y \cdot \cos \omega_C t \]  

(6.2)
Alınan işaret taşıyıcısının, demodülasyon işleminde kullanılan yerel taşıyıcı ile faz ve freksansın uyumlu olduğu düşünülürse;

\[ X_p, (6.3)' \text{ deki gibi,} \]

\[ X_p = (x \sin \omega_c t + y \cos \omega_c t) \sin \omega_c t + (x \cos \omega_c t - y \sin \omega_c t) \cos \omega_c t \tag{6.3a} \]

\[ X_p = x \sin^2 \omega_c t + x \cos^2 \omega_c t + y \cos \omega_c t \sin \omega_c t - y \cos \omega_c t \sin \omega_c t \tag{6.3b} \]

\[ X_q = x (\sin^2 \omega_c t + \cos^2 \omega_c t) + y \cos \omega_c t \sin \omega_c t - y \cos \omega_c t \sin \omega_c t \tag{6.3c} \]

\[ X_p = x \tag{6.3d} \]

\[ X_q \text{ ise, (6.4)' \text{ deki gibi elde edilir.} \]

\[ X_q = (x \sin \omega_c t + y \cos \omega_c t) \cos \omega_c t - (x \cos \omega_c t - y \sin \omega_c t) \sin \omega_c t \tag{6.4a} \]

\[ X_q = x \sin \omega_c t \cos \omega_c t - x \sin \omega_c t \cos \omega_c t + y \sin^2 \omega_c t + y \cos^2 \omega_c t \tag{6.4b} \]

\[ X_q = x \sin \omega_c t \cos \omega_c t - x \sin \omega_c t \cos \omega_c t + y (\sin^2 \omega_c t + \cos^2 \omega_c t) \tag{6.4c} \]

\[ X_q = y \tag{6.4d} \]

Şekil 6.3' de blok diyagramı gösterilen ve (6.3) ile (6.4) ifadelerinde matematiksel tanımı verilmiş olan Hilbert dönüştürümlü demodülasyon işlemi, iki boyutlu uzayda tanımlı olan işaret durum diyagramı üzerinde açısal vektör öteleme işlemini tanımlamaktadır. Hilbert dönüştürümlü demodülasyon işlemi, (6.3) ve (6.4) matematiksel ifadeleri yerine (6.5)' de gösterilen matris çarpımı ile de tanımlanabilir.

\[
\begin{bmatrix}
X_p \\
X_q
\end{bmatrix} =
\begin{bmatrix}
\cos \omega_c t & \sin \omega_c t \\
-\sin \omega_c t & \cos \omega_c t
\end{bmatrix}
\begin{bmatrix}
Z_p \\
Z_q
\end{bmatrix}
\tag{6.5}
\]

\(\beta = \omega_c t\) olarak tanımlanrsa; işaret durum diyagramı üzerindeki açısal vektör öteleme, \(\beta\) açısı kadar olur. Hilbert dönüştümü, zamanda tanımlı bir işlemidir. Demodülasyon işleminin sayısal işaretleme teknikleri ile yapılacağ olması, Hilbert dönüştümünün ayrık yapılmasını gerektirir. Bundan dolayı; \(Z_p\) ve \(Z_q\), 14400Hz frekansında örneklenmek kullanılır. Haberleşme kanalında kullanılan işaret hızı 2400 sembol/saniye (baud) olduğu için, her sembolden 6 örnek alınır. Taşıyıcı frekansın 1800Hz ve işaret akış hızının 2400 sembol/saniye olması nedeniyle \(\cos(.)\) ve \(\sin(.)\)
ifadelerinin alınacak her örnek değeri için farklılık gösterir. Bundan dolayı, Hilbert dönüşümü sayıca demodülasyon işlemi için, \( t = n.T \) olarak tanımlanrsa (6.6) elde edilir.

\[
\omega_c.t = \omega_c.n.T = 2.\pi. f_c.n.T
\]

\[
T = 1/14400 \text{ sn}, \quad f_c = 1800\text{Hz ve } n : \text{örnek sayısı}
\]

(6.6) ifadesinden, \( \cos (\pi/4.n) \) ve \( \sin (\pi/4.n) \) elde edilir. Böylece 6 örnek için bu değerlerin bir tabloda tutulması ve Hilbert dönüşümlü demodülasyon işleminde kullanılması gerekilidir. Bu şekilde elde edilen \( X_p \) ve \( X_q \) değerleri işaret akış hızında örneklencerek, \( X_N \) ve \( Y_N \) değerleri elde edilir. İşaret akış hızı, gelen işaretin zarfından çıkarılarak kullanılır. Yapılan tüm işlemlerde kullanılan yerel taşıyıcı frekansı ile alınan taşıyıcı frekansının faz ve frekans uyumsuzlukları vektör döndürücü ile giderilmeye çalışılır.

### 6.3.2 Hilbert Dönüşürtücüleri

Hilbert dönüşürtürcüler, giriş işaretlerinin fourier bileşenlerinin fazlarını -90° ötelemekte kullanılırlar. Ayrıntı işaretlerin kompleks olarak gösterimi, zamanda tanımlı olan işaretlerin gösterimine benzer (6.7). \( x(n) \) dizisinin fourier dönüşümü, \( X(e^{j\omega}) \) olarak tanımlanabilir ve kompleks bir dizi olarak gösterilebilir (6.8).

\[
x^I(n) = x(n) + j.x^II(n)
\]

\[(6.7)\]

\((6.7)'\) de gösterilen \( x^I(n) \) dizisinin fourier dönüşümü alınmış ifadesi, \((6.8)'\) de verilmiştir.

\[
X^I(e^{j\omega}) = X(e^{j\omega}) + j.X^II(e^{j\omega})
\]

\[(6.8)\]

\[
X^I(e^{j\omega}) = 2.X(e^{j\omega}), \quad 0<\omega<\pi
\]

\[(6.9a)\]

\[
X^I(e^{j\omega}) = 0, \quad \pi<\omega<2\pi
\]

\[(6.9b)\]

(6.9) denklemi göz önüne alınarak, (6.8) denklemi sayesinde \( 0<\omega<2\pi \) aralığında geçerli olan (6.10) eşitlikleri yazılabilir.

\[
X^I(e^{j\omega}) = 2.X(e^{j\omega}), \quad 0<\omega<\pi
\]

\[(6.10a)\]

\[
X(e^{j\omega}) + j.X^II(e^{j\omega}) = 0, \quad \pi<\omega<2\pi
\]

\[(6.10b)\]

(6.8) eşitliği, (6.10.a) denklemi yardımıyla tekrar yazılarak, \( X^II(e^{j\omega}) \) ifadesi (6.11) elde edilir.

\[
2.X(e^{j\omega}) = X(e^{j\omega}) + j.X^II(e^{j\omega})
\]

\[(6.11a)\]
\[ X(e^{j\omega}) = j \cdot X^{\prime}(e^{j\omega}) \quad (6.11b) \]
\[ X^{\prime}(e^{j\omega}) = -j \cdot X(e^{j\omega}) \quad , \quad 0 < \omega < \pi \quad (6.11c) \]

\[ X^{\prime}(e^{j\omega}) \text{ ifadesi (6.10b) denklemi kullanılarak yazılırsa, (6.12) ifadesi elde edilir.} \]

\[ j \cdot X^{\prime}(e^{j\omega}) = -X(e^{j\omega}) \quad (6.12a) \]
\[ j^2 \cdot X^{\prime}(e^{j\omega}) = -j \cdot X(e^{j\omega}) \quad (6.12b) \]
\[ X^{\prime}(e^{j\omega}) = j \cdot X(e^{j\omega}) \quad , \quad \pi < \omega < 2\pi \quad (6.12c) \]

Böylece, \( X^{\prime}(e^{j\omega}) \) ve \( H_d(e^{j\omega}) \) (6.13)'deki gibi tanımlanabilir.

\[ X^{\prime}(e^{j\omega}) = H_d(e^{j\omega}) \cdot X(e^{j\omega}) \quad (6.13a) \]
\[ H_d(e^{j\omega}) = -j \quad , \quad 0 < \omega < \pi \quad (6.13b) \]
\[ H_d(e^{j\omega}) = j \quad , \quad \pi < \omega < 2\pi \quad (6.13c) \]

\( x(n) \) dizisi, \( H_d(e^{j\omega}) \) frekans cevabına sahip Hilbert dönüştürücüden geçirilerek \( x^{\prime}(n) \) dizisi elde edilir (Şekil 6.4b).

Hilbert dönüştürücü darbe cevabı, nedensel değildir ve idealde sonsuz uzunluktaadır. FIR filtre biçiminde tasarlanarak Hilbert dönüştürücü yapılabılır. Böyle bir dönüştürücünün; filtre uzunluğu, alt kesim frekansı, üst kesim frekansı ve yaklaşma hattası olmak üzere dört ayrı parametresi bulunmaktadır. Filtre cevabının simetrik olabilmesi için, alt ve üst normalize frekans değerlerinin toplamı normalize nyquist frekansına eşit olmalıdır. Ayrıca sütçe uzunluğunun artması, geçirme bandındaki dalgalanmayı azaltmaktadır (Şekil 6.4a). Şekil 6.4' de darbe cevabı ile genlik ve faz eğrileri gösterilen Hilbert dönüştürücünün uzunluğu 31, yaklaşma hattası 0.75, alt kesim frekansı 0.04 ve üst kesim frekansı 0.46 olarak kullanılmıştır.

---

(a) Hilbert Dönüştürücü Darbe Cevabı
(b) Hilbert Dönüşçü Güc ve Fazı Cevabı

Şekil 6.4 Hilbert Dönüşçü Darbe, Güc ve Fazı Eğrileri

6.3.3 Hilbert Dönüşçü ve Demodülatör Tasarımı

"6.3.2 Hilbert Dönüşçü" bölümünde verilen 31 uzunlukta Hilbert dönüştürücü yazılımı Tablo 6.1'de verilmiştir. Şekil 6.4a'da verilen dönüştürücü darbe büyükleri `hilbert_coeffs` tablosunda saklanmıştır. Örneklenmiş dönüştürücü giriş değerleri, `xv` tablosu ile yazılıma girer. Fazi dönümsüz işaret `hilbert_zp` deşıkeni ile programdan çıkar.

**Tablo 6.1 Hilbert Dönüşçü Programı**

```c
/* hilbert transformer alt program degerleri */
static float hilbert_coeffs[] = {0,0.0058593750, 0, 0.0097656250, 0, 0.0214843750, 0,
0.0449218750, 0, 0.0839843750, 0, 0.1542968750, 0,
0.3066406250, 0, 0.9980468750, 0,-0.9980468750, 0,
-0.3066406250, 0,-0.1542968750, 0,-0.0839843750, 0,
-0.0449218750, 0,-0.0214843750, 0,-0.0097656250, 0,
-0.0058593750};

static float xv[NZEROS+1] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
#define NZEROS 30
#define KAZANC 1.5781250000e+00

/* hilbert dönüştürücü programı */

static void hilbert_transformer()
{
    float sum; int i;
    for (i = 0; i < NZEROS; i++)
        xv[i] = xv[i+1];
    xv[NZEROS] = Hilbert_Zp / KAZANC;
    xv[NZEROS] = Hilbert_Zp;
    sum = 0.0;
}
```

70
for (i = 0; i <= NZEROS; i++)
    sum += (hilbert_coeffs[i] * xv[i]);
Hilbert_Zq = sum;

Şekil 6.3’de görülen kompleks demodülasyon işleminin gerçekleştiriltiği program, Tablo 6.2’de verilmiştir.

Tablo 6.2 Kompleks Demodülatör Programı

for (i = 0; i <= NZEROS; i++)
    sum += (hilbert_coeffs[i] * xv[i]);
Hilbert_Zq = sum;

static void Demodulation( )
{
    Demod_Xp = (Hilbert_Zq * TekPerSin_SampleTable[TBCwl_Sayac4] +
                (Hilbert_Zp * TekPerCos_SampleTable[TBCwl_Sayac4] ));
    Demod_Xq = (Hilbert_Zq * TekPerCos_SampleTable[TBCwl_Sayac4] -
                (Hilbert_Zp * TekPerSin_SampleTable[TBCwl_Sayac4] ));
    if (TBCwl_Sayac4 == 7) TBCwl_Sayac4 = 0; else TBCwl_Sayac4++;
}

6.4 UYARLAMALI DENGLEYİCİ VE AÇISAL HATA HESAPLAMA

6.4.1 Uyarlamalı Dengeleyici Özellikleri


Fakat modern haberleşme mesininde kullanılan haberleşme kanalları, rastgelelik gösterir ve özellikleri tam olarak bilinemez. Bundan dolayı, ortalama haberleşme kanal karakteristiklerine göre tasarlanmış dengeleyiciler semboller arası girişimi azaltmakta yeterince etkili olamazlar.

Tipik bir haberleşme kanalı, 300 – 3500Hz arasında bant geçiren bir filtre gibi davranır. Bu bant içerisinde kullanılan herhangi bir frekans bileşenine karşı haberleşme kanalı farklı zayıflatma ve gecikme uygular. Bu zayıflatma ve gecikme özelliklerinin varolması nedeniyle, oluşabilecek hataların önüne geçilebilmesi ve
haberleşme kanalının tüm iletişim yeteneklerinin verimli bir şekilde kullanılabilmesi için uyarlamalı dengeleme yapılması zorunludur.

Uyarlamalı dengeleyiciler, alınan önceki giriş işaretlerindeki bozulmalara göre katsayılarını ayarlayarak dengeleme işlemini gerçekleştirmirler. Modellerin bağlantı kurma sırasında yani el sıkışma aralığından, kullanılan veri iletişim standardında tanımlı olan veri dizileri kullanılarak her iki modemin uyarlamalı dengeleyicileri eğitilir ve başlangıç katsayı değerleri belirlenir. Bu katsayı değerleri, veri iletişimi sırasında sürekli yenilenerek demodülatör katı haberleşme kanalına uyumlu tutulur.

Kafes kodlamalı modülasyon teknikinde, modülasyonlu işaret birbirine dik iki taşıyıcı içerdığı için kompleks olarak modellenmektedir. Bundan dolayı kullanılabilecek dengeleyici de kompleks özellik gösterir.

\[ y(n) = \sum_{j=-m}^{m} k_j(n).x(n-j) \]  \hspace{1cm} (6.14)

Şekil 6.5 Kompleks Dengeleyici Blok Diyagramı

\[ y_p = \sum_{i=0}^{m} x_p^i \cdot k_p^i + x_Q^i \cdot k_Q^i \]  \hspace{1cm} (6.15a)

\[ y_Q = \sum_{i=0}^{m} x_Q^i \cdot k_P^i - x_P^i \cdot k_Q^i \]  \hspace{1cm} (6.15b)
Dengeleyici katsayıları, karesel ortalama hatayı en küçük yapacak şekilde yinelenir. \( E_P \) ve \( E_Q \), karar verme işleminde oluşan hatanın bir fonksiyonu olarak tanımlanabilir (6.16).

\[
\begin{align*}
E_P(n) &= f\left( |X'(n) - X(n)| \right) \\
E_Q(n) &= f\left( |Y'(n) - Y(n)| \right) \\
E(n) &= E_P(n) + jE_Q(n)
\end{align*}
\]  

(6.16a) (6.16b) (6.16c)

Katsayıların güncellenmesi işleminde kullanılan \( E_P \) ve \( E_Q \) hata değerleri ters döndürme işlemi sonuçlarıdır (6.17).

\[
\begin{align*}
\begin{bmatrix} E_P \\ E_Q \end{bmatrix} &= \begin{bmatrix} \cos(-\alpha) & \sin(-\alpha) \\ \sin(-\alpha) & \cos(-\alpha) \end{bmatrix} \cdot \begin{bmatrix} |X' - X| \\ |Y' - Y| \end{bmatrix} \\
\alpha &= \arctan\left( \frac{X}{Y} \right) - \arctan\left( \frac{X'}{Y'} \right)
\end{align*}
\]  

(6.17a) (6.17b)

Dengeleyicinin kullandığı katsayıların yinelenmesi (6.18)'da tanımlanmıştır.

\[
K(n+1) = K(n) + k \cdot E(n) \cdot E^*(n)
\]  

(6.18)

Kompleks dengeleyicinin katsayıları olan \( K_P \) ve \( K_Q \), (6.19)' de tanımlı olan matematiksel ifadelerle her sembol aralığında yenilenir.

\[
\begin{align*}
K_P(n+1) &= K_P(n) + k \cdot [E_P(n) \cdot X_P(n) + E_Q(n) \cdot X_Q(n)] \\
K_Q(n+1) &= K_Q(n) + k \cdot [-E_P(n) \cdot X_Q(n) + E_Q(n) \cdot X_P(n)]
\end{align*}
\]  

(6.19a) (6.19b)

(6.18) ve (6.19) denklemlerinde kullanılan adaptasyon katsayısı \( k \), (6.20)' deki eşitsizliğe uyuyor ve \( n = 0, 1, 2, \ldots, N \) için ilk değerler (6.21)' deki gibi kullanılır.

\[
0 < k < \frac{2}{N \cdot P_{\text{ort}}} \quad N : \text{Vurum sayısı} \quad P_{\text{ort}} : \text{Ortalama giriş gücü}
\]  

\[
K_P^n(-1) = K_Q^n(-1) = 0
\]  

(6.20a) (6.20b)

\[
X_P(-n) = X_Q(-n) = 0
\]  

(6.21a) (6.21b)
6.4.2 Uyarlamalı Dengeleyici Tasarımı

Uyarlamalı dengeleyici tasarımı, "6.4.1 Uyarlamalı Dengeleyici Özellikleri" bölümünde anlatılan kurallar çerçevesinde yapılmıştır. KatSayı güncelleme ve ters döndürme işlemleri dengeleyici tasarımı ile birlikte ele alınmıştır. Dengeleyici derinliği 7 olarak seçilmiştir.

Tablo 6.3 Uyarlamalı Dengeleyici Programı

```c
float adaptif_giris_xp_table[7] = {0,0,0,0,0,0,0};
float adaptif_giris_xq_table[7] = {0,0,0,0,0,0,0};
/**
 */
/* katsayı yinelemede oluşturulan katsayılar ile dengeleyiciye gelen */
/* giriş katsayıları yardımıyla 2400Hz hzında dengeleyici çıkışları */
/* oluşturulur. Kullanılan dengeleyici komplex yapidadır ve 7 derinliklidir. */
/***************************************************************************/

static void adaptif_dengeleyici(float adaptif_giris_xp,float adaptif_giris_xq)
{
  int i;
  adaptif_cikis_xp=0; adaptif_cikis_xq=0;
  /* dengeleyicinin kullandığı xp ve xq tabloları 7 derinliğe kadar tutuluyor. */
  for (i=6;i>=1;i--)
  {
    adaptif_giris_xp_table[i] = adaptif_giris_xp_table[i-1];
    adaptif_giris_xq_table[i] = adaptif_giris_xq_table[i-1];
  }
  adaptif_giris_xp_table[0] = adaptif_giris_xp;
  adaptif_giris_xq_table[0] = adaptif_giris_xq;

  /* adaptif dengeleme sonuçları 7 derinliğe kadar bulunuyor */
  for (i=0;i<=>6;i++)
  {
    adaptif_cikis_xp = adaptif_cikis_xp + ((adaptif_giris_xp_table[i] * Kp_table[i]) +
      (adaptif_giris_xq_table[i] * Kq_table[i] ));
    adaptif_cikis_xq = adaptif_cikis_xq + ((adaptif_giris_xq_table[i] * Kp_table[i]) -
      (adaptif_giris_xp_table[i] * Kq_table[i] ));
  }
}
/***************************************************************************/
```

Tablo 6.4 Uyarlamalı Dengeleyici Katsayı Güncellemesi Programı

```c
float Kp_table[7] = {0,0,0,0,0,0,0};
float Kq_table[7] = {0,0,0,0,0,0,0};
/***************************************************************************/
/* alfa açısı kadar döndürtilen fark bileşenleri ile demodülatör çıkışı */
/* oluşan değerler yardımıyla adaptif dengeleyicinin kullanacağı birbirine */
/* dik bileşenler için 7 adet katsayı belirlenir. */
/***************************************************************************/

static void katsayi_yineleme(float tersdon_oupt,float tersdon_outq,float adaptin_xp,float adaptin_xq)
{
  int i;
  for (i=6;i>=1;i--)
  {
    /* p ve q katsayıları birer ötelenir. yeni değerlerle yer açılır. */
    Kp_table[i] = Kp_table[i-1]; Kq_table[i] = Kq_table[i-1];
    Kp_table[0] = Kp_table[1] +
      katsayi_yineleme_parametresi * (adaptin_xp*tersdon_oupt+adaptin_xq*tersdon_outq);
    Kq_table[0] = Kq_table[1] +
      katsayi_yineleme_parametresi * (adaptin_xp*tersdon_oupt+adaptin_xq*tersdon_outq);
  }
}
/***************************************************************************/
```

74
Tablo 6.5 Ters Vektör Döndürme Programı

```c
static void tversktor_dondurme(float karar_hata_xp, float karar_hata_xq, float donme_acisi)
{
    float radian_donme_acisi; float pi = 3.1415927;
    radian_donme_acisi = -1 * (donme_acisi / 180) * pi;
    ters_dond_cikis_xp = (cos(radian_donme_acisi) * karar_hata_xp) +
                        (sin(radian_donme_acisi) * karar_hata_xq);
    ters_dond_cikis_xq = (-sin(radian_donme_acisi) * karar_hata_xp) +
                         (cos(radian_donme_acisi) * karar_hata_xq);
}
```

6.4.3 Açısal Hata Hesaplama ve Döndürme İşlemini

Kara verici girişine gelen vektör ile kara verici çıkışındaki çıkış vektörü arasındaki farklı açısal hatanın hesaplanması ve uyarlamalı dengeleyicinin katsaylarının belirlenmesinde kullanılır. Açısal hata hesaplayıcı çıkış değeri, taşıyıcı çıkarma süzgecinden geçirilerek, bir sonraki vektör döndürme açı değeri (α) bulunur (Şekil 6.6). Taşıyıcı çıkarmafiltresi, demodülatör katı ile birlikte incelendiğinde PLL çevrими oluşturulur. Hesaplanan açı hatası, bir sonraki vektör döndürme işleminde kullanılmış bu çevrimi göstermektedir.

![Şekil 6.6 Taşıyıcı Çıkarma Filtresi Blok Diyagramı](image)

Şekil 6.6’ da ikinci dereceden bir filtre görülmektedir. K₁ ve K₂ değerleri (6.22) ile bant genişliği değiştirilebilir.

\[
K_1 = 1 - \varphi^2 - \sigma^2 \\
K_2 = 1 + \varphi^2 - 2\varphi + \sigma^2
\]  
(6.22a)  
(6.22b)

\[
\varphi = \cos \left[ \omega_n T \cdot (1-C^2)^{1/2} \right] \cdot e^{-C \cdot \omega_n T} \\
\sigma = \sin \left[ \omega_n T \cdot (1-C^2)^{1/2} \right] \cdot e^{-C \cdot \omega_n T}
\]  
(6.22c)  
(6.22d)

\( \omega_n \): Doğal frekans, C: Amortisman oranı ve T: Örnekleme periyodu

Demodülatörde kullanılan bu çevrminin, vektör döndürücü çıkış ve girişine göre frekans cevabı (6.23)' deki gibidir.

\[
\text{Çıkış / Giriş} = 1 - \frac{(1-Z^{-1})^2}{1+Z^{-1}(K_1+K_2-2) + Z^{-2}(1-K_1)}
\]  
(6.23)

Tablo 6.6 Açışal Hata Hesaplama Programı

```c
void aчисal_hata_hesaplayici_VCO(float parametro1, float parametro2,
                             float parametro3, float parametro4)
{
    float hata_acisi, pi = 3.1415927;
    float PLL_temp1, PLL_temp2, PLL_katsayi1, PLL_katsayi2;
    hata_acisi = (180 / pi) * atan(parametro2 / parametro1) - atan(parametro4 / parametro3);
    PLL_temp1 = cos(2*pi*dogal_fr_deg*(1/ornek_fr_deg)*sqrt((1-amortisman_deg*amortisman_deg)))*
                exp(-1*amortisman_deg*2*pi*dogal_fr_deg/ornek_fr_deg);
    PLL_temp2 = sin(2*pi*dogal_fr_deg*(1/ornek_fr_deg)*sqrt((1-amortisman_deg*amortisman_deg)))*
                exp(-1*amortisman_deg*2*pi*dogal_fr_deg/ornek_fr_deg);
    PLL_katsayi1 = PLL_temp1 - PLL_temp2;
    PLL_katsayi2 = PLL_temp1 + PLL_temp2;
    PLL_out = ((PLL_katsayi1/(1-PLL_log)) + PLL_katsayi2) * (PLL_log/(1-PLL_log)) * ATA_acisi;
    PLL_log = PLL_out;
}
```

Uyarlamalı dengeleyici çıkısları, hesaplanan hata açısı kadar döndürülerek karar verme işlemine başlanır. Döndürme işlemi; matematiksel olarak (6.24)’ de tanımlanmış olup, Tablo 6.7’ de gösterilen program parçası ile gerceklendirilir.

\[
\begin{array}{c|cc|c}
Y_P & \cos(\alpha) & \sin(\alpha) & X \\
Y_Q & -\sin(\alpha) & \cos(\alpha) & Y \\
\end{array}
\]  

(6.24)

Tablo 6.7 Vektör Döndürme Programı

```c
void vektor_dondurme(float dondur_giris_x, float dondur_giris_y, float donme_acisi)
{
    float radian_donne_acisi = donme_acisi / 180;
    dondur_giris_x = cos(radian_donne_acisi) * dondur_giris_x +
                     sin(radian_donne_acisi) * dondur_giris_y;
    dondur_giris_y = -sin(radian_donne_acisi) * dondur_giris_x +
                     cos(radian_donne_acisi) * dondur_giris_y;
}
```
6.5 KARAR VERİCİ

6.5.1 Katlamah Kodlar İçin En Büyük Benzerlikli Kod çözme (Viterbi Algoritması)

“2.6 Kafes Kodlamah Modülasyon (TCM)” ve “3. ITU-T V.32 Standardı” bölümlerinde özellikleri incelenen kafes kodlu modülasyon (TCM) tarafından kodlanmış semboller, demodülatör tarafından temel banda indirildikten sonra, Viterbi algoritması kullanılarak çözülür. Viterbi algoritması, yazılım ile karar verilen ve en büyük benzerlik karar kuralı temel alınmış çözücü teknikleri üzerine oturtulmuş olup katlamalı kodların çözülmesinde kullanılan bir yapılıt. Herhangi bir çözücü yapısının temel fonksiyonu, en muhtemel olan çıkışı seçmektir. Çözücü yapıları, sert ve yumuşak karar veren iki ayrı grupta incelenebilir.

Sert karar veren basit bir çözünün çıkışı olarak karar verdiği nokta, aldığı işarette kullanılan işaret diyagramındaki en yakın noktasıdır. Başka bir deyişle, alınan işaret ile ideal işaretler arasındaki farkın en az olduğu ideal işaret çıkışı olarak kabul edilir.


İdealde, en büyük benzerlik metodunda çıkışla ilgili herhangi bir karar verilmeden önce tüm girişlere bakılır. Ideal olarak tanımlanmış bu yaklaşım, iki ayrı etkenden dolayı gerçek zaman uygulamalarında yapılabılır değildir. Bu etkenler, fazla bellek gereksinimi ve çözümü çıkış oluşturmadan önceki doğal zaman gecikmeleri olarak tanımlanabilir.


Her yol gösterilen yol durumları ile tanımlanmıştır. Yol durumları 5 bitlik veri grubu içindeki 3 biti içerir. Böylece her yol durumu, işaret diyagramı üzerindeki sekiz ayrı alt kümeden bir tanesini tanımlar.
Şekil 6.8 V.32 TCM 32 – nokta İşaret Küme Örnekleri


6.5.2 Viterbi Çözücü Tasarımı


Şekil 6.9 32 – nokta İşaret Bölmleri

Böylece, alınan işaret ile işaret diyagramındaki 32 nokta arasındaki karesel Öklid uzaklıklarının hesabı yerine, alınan işaretin diyagram üzerindeki yeri belirlenir.

\[ f_{n} = 0.9f_{n-1} + 0.1\text{öklid}_{uz} \] (6.25)

Bu tanım, alçak geçiren süzgecin IIR olarak kullanılmasıdır. Viterbi derinliğine gelindiğinde, hesaplanmış fark fonksiyon tablosu olan ACCDIST[8] kullanılarak geriye döndük olarak gecikme durumları ve geçiş yolları belirlenir. Gecikme durumlarına ait hesaplanmış fark fonksiyonunun başlangıç değerleri 0.5, 0, 0, 0, 0, 0, 0, 0 olarak kullanılır.

“6.5.1 Katlamalı Kodlar İçin En Büyük Benzerlikli Kod çözme” bölümlünde ve bu bölümdde açıklanan özellikler çerçevesinde tasarımı yapılan Viterbi çözücü programı Ekler bölümünde verilmiştir.

6.5.3 Viterbi Çözücü Performansı ve Hata Başarımı

Viterbi çözücü performansı ve hata başarınının incelenmesinde kodlayıcı ve kod çözücüden oluşan temel bant iletişim sistemi kullanılmıştır (Şekil 6.10). Kod çözücü ve eşleyici çıkışındaki temel bant işaretine \(a_n\) gauss gürültüsü eklenerek Viterbi çözücü giriş işaretine \(r_n\) belirlenmiştir (2.10). Eklenen gauss gürültüsünün \(w_n\) standart sapması değiştirilerek farklı işaret gürültü oranlarında (SNR), hatalı karar verilen sembol sayısı (SER) bulunmuştur.

![Şekil 6.10 Hat Kodlamalı Temel Bant İletişim Sistemi](image)

Yapılan tasarım çalışmalarda C programlama dili kullanıldığı için üretilen rasgele sayılar düzgün dağılmaktadır. Bu nedenle düzgün dağılmış rasgele sayılar arasında gauss dağılım karakteristiğine sahip gürültü bileşenleri elde etmek için Tablo 6.8' deki program parçası yazılmıştır. Bu programda, düzgün dağılan iki rasgele elde edilmiş sayısalden değerden \(\text{AWGN}_nx\) ve \(\text{AWGN}_ny\) gauss gürültü bileşenleri elde edilir. Programa standart sapma ve ortalama değer parametre olarak atanmaktadır. Beyaz gauss gürültüsü sıfır ortalamalı olarak kullanılmıştır.
Tablo 6.8 Gauss Gürültü Üretme Programı

```c
/* *******************************************************************************
static generate_gauss(float gauss_mean, float gauss_sigma)
{
    /* uniform değişken 0 - 32767 arasında olabılır. */
    float uni1,uni2,r; /* uniform and Rayleigh random değişkenler */
    float pi = 3.1415927;
    float rand1,rand2;
    /* 0-1 arasında uniform dağılımı random sayı üretme */
    rand1 = rand();
    srand(rand1);
    uni1 = rand1 / Generate_Gauss_Max;
    if (uni1 == 1) uni1 = 0.999999999;
    /* Rayleigh-distributed random sayı üretme */
    r = gauss_sigma * sqrt(2.0 * log(1/(1 - uni1))) ;
    /* 0-1 arasında uniform dağılımı random sayı tekrar üretme */
    rand2 = rand();
    srand(rand2);
    uni2 = rand2 / Generate_Gauss_Max;
    if (uni2 == 1) uni2 = 0.999999999;
    /* Gaussian-distributed random sayı üretme */
    AWGN_nx = gauss_mean + r * cos(2 * pi * uni2);
    AWGN_ny = gauss_mean + r * sin(2 * pi * uni2);
}
/***********************************************************************
```

İşaret gürültü oranı, gönderilen işareti ve gürültünün gücü (6.26) ifadesindeki gibi gösterilir.

\[
\text{SNR (dB)} = 10.\log_{10} \frac{E[a^2(t)]}{E[w^2(t)]} \quad (6.26a)
\]

\[
E[a^2(t)] = E_t = \frac{1}{32} \sum_{i=1}^{32} (a_{xi}^2 + a_{yi}^2) \quad (6.26b)
\]

\[
E[w^2(t)] = N_0 = 2.\sigma^2 \quad (6.26c)
\]

İşaret gürültü oranının değişimi, Viterbi çözücüünün hata başarımını etkilemektedir. Sembol hata oranı (6.27)' deki gibi tanımlanmaktadır.

\[
\text{SER} = \frac{\text{hatalı alınan sembol sayısı}}{\text{toplamar alınan sembol sayısı}} \quad (6.27)
\]

Sembol hata oranı, Tablo 6.9' da tanımlı test dizisi kullanılarak 1000 sembol sayısı için değişik SNR oranlarında deneyssel olarak elde edilmiştir (Şekil 6.11).
Tablo 6.9 Test Amaçlı Kullanılan Sayısal Veri Dizisi

<table>
<thead>
<tr>
<th>Test Dizisi</th>
<th>Test Byte</th>
<th>Hex Kod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10010110</td>
<td>0x96</td>
</tr>
<tr>
<td>2</td>
<td>00100111</td>
<td>0x27</td>
</tr>
<tr>
<td>3</td>
<td>00111101</td>
<td>0x3d</td>
</tr>
<tr>
<td>4</td>
<td>01100101</td>
<td>0x65</td>
</tr>
<tr>
<td>5</td>
<td>00101111</td>
<td>0x2f</td>
</tr>
<tr>
<td>6</td>
<td>00001011</td>
<td>0x0b</td>
</tr>
<tr>
<td>7</td>
<td>01111010</td>
<td>0x7a</td>
</tr>
<tr>
<td>8</td>
<td>01010110</td>
<td>0x56</td>
</tr>
</tbody>
</table>

Şekil 6.11 Gauss Gürültülü Kanalda Viterbi Performansı

6.6 TERS FARKSAL DÖNÜŞTÜRME VE ÇÖZME

6.6.1 Ters Farksal Dönüşüm

Karar verme işlemi sonrası alınan sembole karşılık oluşturulan 5 bitlik veri grubu (Y0, Y1, Y2, Q3 ve Q4) yardımıyla ters farksal dönüşüm yapılır. Bu dönüşümün matematiksel ifadeleri (6.28) ve (6.29) de tanımlanmıştır. Y1 ve Y2 verileri ile Q1 ve Q2 oluşturulur. Y0, ek bit olduğu için kullanılmayacaktır.

\[
Q_{1n} = Y_{1n} \oplus Y_{1n-1} \quad (6.28)
\]

\[
Q_{2n} = (Q_{1n} \cdot Y_{1n-1}) \oplus Y_{2n-1} \oplus Y_{2n} \quad (6.29)
\]

Ters farksal dönüşümü ile beş bitlik veriden Q1, Q2, Q3 ve Q4 verileri oluşturulur. Bu matematiksel ifadelerin gerçekleştiği program parçası Tablo 6.10' da verilmiştir.
Tablo 6.10 Ters Farksal Dönüşüm Programı

/* Ters diff.dönüşüm */
rec_q_table[0] = signal_element_mapping_output[1] ^ diff_encoder_log_table[0];
rec_q_table[1] = (rec_q_table[0] & diff_encoder_log_table[0])
^ (diff_encoder_log_table[1] ^ signal_element_mapping_output[2]);
rec_q_table[2] = signal_element_mapping_output[3];
rec_q_table[3] = signal_element_mapping_output[4];
diff_encoder_log_table[0] = signal_element_mapping_output[1];
diff_encoder_log_table[1] = signal_element_mapping_output[2];

/* ********************************************* */

6.6.2 Çözücü (Descrambler)

6.6.2.1 Çözücü Özellikleri

Ters farksal dönüşüm ile oluşturulan Q1, Q2, Q3 ve Q4, çözme işlemine tabi tutulur. Çözme işlemi yapılan veriler, modemin bağlı bulunduðu terminale gönderilir. Çözme işleminde, aranan veya aranan modem olma durumuna göre V.32 standardında tanımlı olan çok terimliler kullanılır. V.32 standardında, arayan modem çözücü çok terimlisi (GPC), \(1 + x^{-5} + x^{-23}\) ve aranan modem çözücü çok terimlisi (GPA), \(1 + x^{-18} + x^{-23}\) olarak tanımlanmıştır. Buna göre karşıtancı kişileri aranan veri katarı (\(D_o\)), (6.30) ve (6.31) ifadeleri ile gösterilebilir.

Arayan modem ise;
\[ D_o = D_s \cdot (1 \oplus x^{-5} \oplus x^{-23}) \]  \(\text{(6.30)}\)

Aranan modem ise;
\[ D_o = D_s \cdot (1 \oplus x^{-18} \oplus x^{-23}) \]  \(\text{(6.31)}\)

(6.30) ve (6.31) eşitliklerinde;
\(D_o\), çözücü kişileri aranan veri katarı,
\(D_s\), çözücü girişindeki veri katarı,
\(\oplus\), EXOR işlemi,
\(\cdot\), binary çarpma işlemi göstermektedir.

\(\text{(a) GPA çok terimlisi kullanılan çözücü katı blok diyagramı}\)
(b) GPC çok terimlisi kullanılan çözücü katı blok diyagramı

Şekil 6.12 Çözücü Katı Blok Diyagramları

Şekil 6.12a ve Şekil 6.12b’ de çözme işleminin yapısı görülmektedir. Kariştırıcı girişindeki veri katarı (Dₐ)’nin ardışık olarak 64 bit iletiminden sonra bir sonraki kariştırıcı çıkışındaki veri (D₀) terslenir. Bu işlem tüm çözücü uygulamalarında kullanılmalıdır. El sıkışma durumlarında ise çözücü kullanılmaz.

6.6.2.2 Çözücü Tasarımı

Ters farklı dönüşüm işlemi sonrası oluşturulan seri veri grubu, “6.6.2.1 Çözücü Özellikleri” bölümünde açıklanan kurallara göre çözülür. Yazılan çözücü programı (Tablo 6.11), Şekil 6.12’ de verilmiş çözücü katı blok diyagramları göz önünde tutularak yazılmıştır.

Tablo 6.11 V.32 Çözücü Programı

```c
/***************************************************************************/
/* DeScrambler_V32(s16 descr_mode,s16 descr_rx_data) */
/* Alinan veri Org ve Ans durumuna göre descramble edilir. descr_mode 0 ise GPA, */
/* 1 ise GPC polinomu kullanılır. descr_rx_data descramble edilecek veri */
/* geri donus degeri descramble edilmis veridir. */
/***************************************************************************/
static s16 DeScrambler_V32(s16 descr_mode, s16 descr_rx_data )
{
    s16 descr_i,descr_scrmld_data;
    /* V.32 GPC ve GPA polinomu uyarinca descramble veri oluşturma*/
    if(descr_mode==1)
        descr_scrmld_data = descr_rx_data ^( descramble_log_table[17] ^ descramble_log_table[22]);
    else
        descr_scrmld_data = descr_rx_data ^( descramble_log_table[4] ^ descramble_log_table[22]);
    /* descrambler log buffer otelseme*/
    for (descr_i=22; descr_i>=1; descr_i--)
        descramble_log_table[descr_i] = descramble_log_table[descr_i-1];
    descramble_log_table[0] = descr_rx_data;
    return (descr_scrmld_data);
}
/***************************************************************************/
```

Çözücü katı çıkış verileri, received_data_table’ da tutulur. Kullanılan çok terimlilerin 23. dereceden olmaları nedeniyle çözücü katında 23 geçikme gözü kullanılır ve geçikme değerleri descramble_log_table’ da tutulur. Başlangıçta
tüm descramble_log_table değerleri "0" olarak tanımlanmıştır. Ters farksal dönüştürücüden gelen D₅ verisi, çözücü çok terimlisine (Modemin arayan ve aranan olmasına göre kullanılan çok terimli değişmektedir.) göre ilgili gecikme gözlerindeki değerler ile EXOR işlemine tabi tutulur. İşlem sonucu (D₀), oluşturulur ve yeni gecikme değerleri descramble_log_table’ a yerleşirilir. Alınan veri (D₅) katarının, 64 bit uzunluğuna ulaşması sonrası bir sonraki çıkış verisi (D₀) terslenerek oluşturulur.

6.7 SONUÇ

Altıncı bölümde; 9600 bit/saniye veri iletişim hızında çalışan demodülatör tasarımını açıklanmıştır.

modulated_sembol_table tablosunda bulunan 14400Hz frekansında örneklenmiş analog semboller, bu bölümde açıklanan tekniklerle demodu ve edilmştir. Demodülasyon işlemi sonrası elde edilen sayısal veri dizisinin, transmit_data_table tablosuna uygunluğu test edilmiştir. 64 bitlik test dizisi dairesel olarak modülatör girişine verilmektedir. Her dairesel dönme başlangıcında bir bitlik ötelemeyele periyodiklik engellenmiştir.

Böylece; tasarlanmış modülatör ve demodülatör yapının hata başarımı, sürekli sayısal veri iletim koşulu altında gerçek zamanda programın koşulmasıyla deneySEL olarak incelenmiştir.
7. SONUÇLAR VE ÖNERİLER

300 – 3500Hz arası bant sınırlı mevcut telefon şebeke alt yapısı üzerinden veri iletişimini daha hızlı ve güvenilir biçimde gerçekleştirmek için modemlerde değişik modülsasyon teknikleri kullanılır.

Genişletilmiş işaret kümeleri kullanan kafes kodlama modülsasyon tekniğinde, kodlanmış işaret dizileri arasındaki en küçük Öklid uzaklığını mümkün olabilecek en büyük uzaklık yapılar. Böylece, arııık oluşturulan temel bant işaretleri arasındaki uzaklığın artması sağlanarak veri iletişiminde olası hata olasılıklarını azaltılmiştir.

Bu tez çalışmasında; ITU V.32 standardı referans alınarak, 9600 bit/saniye hızında çalışan modülatör ve demodülatör yapıları tasarlanmıştır. Tasarım çalışmalarını; TMS320C5420 sayısal işaret işlemcisi ve program geliştirme kitleri üzerinden, kafes kodlama modülsasyon ve Viterbi algoritmaları kullanılarak yapılmıştır. Tasarım çalışmalarını ile ilgili bilgiler beşinci ve altıncı bölümlerde verilmiştir. Yazılım çalışmalarında, C programlama dili kullanılmıştır. Program, derlenerek makine diline çevrilmiş ve DSK kartı üzerinden çalıştırılmıştır.

Tasarlanan modülatör yazımı, 64 bit uzunluğundaki test dizileri ile çalıştırılmıştır. Modülsasyon programını, bu test dizilerindeki verileri 9600Hz hızında alarak modüle eder ve 14400Hz örnekleme hızında modülsasyonlu sembollerı oluşturur. Örneklemiş analog semboller, demodülsasyon programı ile sayısal veriler haline getirilmiştir. Demodülsasyon işlemi sonrası edilen sayısal veri dizisinin, ideal kanal kullanıldığında modülsasyon işlemine kullanılan test dizisinin aynı olduğu görülmuştur. Viterbi çözücünün hata başarımı değişik işaret gürültü oranları için incelenmiştir.

Gerçekleştirenilen yazılımlar, modemlerin MAFE katında koşulan ve modem hızına göre ITU-T tarafından standartlaştırılmış tekniklerle modülsasyon ve demodülsasyonu gerçekleştiren yazılımlardır.

Dünya çapında kendini kanıtlaması modern tümleşik devreleri ve yazılımları tasarlayan firmalar, modem üreticilerine toplam çözümler sunmaktadır. Bu çözümlerin maliyetleri, büyük oranda satış fiyatlarına yansıtılmaktadır. Gelişen DSP
teknolojileri ile modem yazılımlarının kolayca yapılabilir olması, daha ucuz modem çözümlerinin ortaya çıkmasına sebep olmuştur.

Bu tez çalışmaları yapılmış olan yazılımlar daha yüksek hızlarda modemler için uyarlanabilir esnekliktedir. Ayrıca, daha üst seviyede modem kontrol yazılımları da yazılıarak değişik uygulamalarda ticari olarak kullanılabilir.
KAYNAKLAR


[18] [http://www-users.cs.york.ac.uk/](http://www-users.cs.york.ac.uk/)

ÖZGEÇMİŞ