

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SAYISAL İŞARET İŞLEMCİ İLE V.32 MODEM TASARIMI

**T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**

**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 y. 22.2.2001
Diğer Jüri Üyeleri Doç.Dr. Işıl CELASUN J. Celasun
Doç.Dr. Osman Nuri UÇAN (İ.Ü.)

ŞUBAT 2001

104212

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, çalışmalarımnda 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	1
2. MODEM HABERLEŞME TEMELLERİ	4
2.1. Temel Kavramlar	4
2.1.1. Giriş	4
2.1.2. Haberleşme Yapıları	5
2.1.2.1. Basit Haberleşme	5
2.1.2.2. Yarı Çift Yönlü Haberleşme	6
2.1.2.3. Tam Çift Yönlü Haberleşme	6
2.1.3. Protokol	7
2.1.3.1. Bilginin Kodlanması	7
2.1.3.2. Kontrol	7
2.1.3.3. Hata Algılama ve Düzeltme	8
2.1.3.4. Haberleşme Kanal Verimliliği	9
2.1.3.5. Eşzamanlı Çalışma	9
2.1.3.6. Saydamlık	9
2.2. Modem Donanım Özellikleri	9
2.3. Modem Bağlantı Hatlarının İncelenmesi	12
2.3.1. Ortalama Bant Genişliği	12
2.3.2. Modülasyonlu İşaret Frekans Genlik Değişimi	13
2.3.3. Gürültü	13
2.3.4. Faz Gecikmesi	14
2.4. CCITT V.24 Özellikleri	14
2.5. Eşzamansız ve Eşzamanlı Haberleşme Yapıları	16
2.5.1. Eşzamansız Veri Yapısı	16
2.5.2. Eşzamanlı Veri Yapısı	17
2.5.2.1. LRC	17
2.5.2.2. VRC	18

2.5.2.3. CRC	18
2.6. Kafes Kodlamalı Modülasyon (TCM)	19
2.6.1. Giriş	20
2.6.2. Katlamalı Kod Yapıları	20
2.6.3. Kod Çözme İşlemi ve Özellikleri	21
2.6.4. Kafes Kodlamada İşaret Kümeleri – Katlamalı Kodlar	23
2.6.5. 32 - Nokta Katlamalı Kod	25
2.6.6. Kafes Kodlamalı Sistem Hata Olasılığı ve Viterbi Algoritması	26
3. ITU-T V.32 STANDARDI	27
3.1. Giriş	27
3.2. Genel Özellikler	27
3.3. Hat İşaretleri	28
3.3.1. Taşıyıcı Frekansı ve Modülasyon Hızı	28
3.3.2. Spektrum	28
3.4. Kodlama	28
3.4.1. 9600 bit/saniye Veri Haberleşme Hızında İşaret Kodlaması	28
3.4.1.1. Normal Kodlama	28
3.4.1.2. Kafes Kodlama	30
3.4.2. 4800 bit/saniye Veri Haberleşme Hızında İşaret Kodlaması	32
3.5. Karıştırıcı (Scrambler) ve Çözücü (Descrambler)	33
4. TMS320C5402 SAYISAL İŞARET İŞLEMCİSİ VE YAZILIM GELİŞTİRME KİTİ	35
4.1. Giriş	35
4.2. Sayısal İşaret İşlemciler	36
4.3. TMS320C5402 Sabit Noktalı Sayısal İşaret İşlemcisi	38
4.4. TMS320C5402 Tabanlı DSK Yazılım Geliştirme Kiti	39
4.4.1. DSP Başlangıç Kiti Donanım Kartı	39
4.4.2. "Code Composer Studio" Yazılım Geliştirme Programı	41
5. V.32 MODÜLATÖR TASARIMI	43
5.1. Giriş	43
5.2. V.32 Modülatör Yapısına Genel Bakış	44
5.3. Seri Veri Girişi ve Karıştırıcı (Scrambler)	45
5.3.1. Karıştırıcı Özellikleri	45
5.3.2. Karıştırıcı Tasarımı	47
5.4. Kodlayıcı Katı ve Temel Bant İşareti Oluşturma	48
5.4.1. Kodlayıcı ve Temel Bant İşareti Oluşturma Yapısı	48
5.4.2. Kodlayıcı Tasarımı	51

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 Modülatör 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. Uyarlamalı Dengeleyici ve Açısal Hata Hesaplama	71
6.4.1. Uyarlamalı Dengeleyici Özellikleri	71
6.4.2. Uyarlamalı Dengeleyici Tasarımı	74
6.4.3. Açısal Hata Hesaplama ve Döndürme İşlemi	75
6.5. Karar Verici	77
6.5.1. Katlamalı 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 Dönüştürme 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

KISALTMALAR

PSK	: Phase Shift Keying
FSK	: Frequency Shift Keying
VLSI	: Very Large Scale Integration Systems
POS	: Point of Sale System
SCADA	: Supervisory Control and Data Acquisition System
SER	: Symbol Error Rate
SNR	: Signal Noise Ratio
ITU	: International Telecommunication Union
WTSC	: The World Telecommunication Standardization Conference
ACK	: Acknowledge
NACK	: Not Acknowledge
MAFE	: Modem Analog Front End
DTE	: Data Terminal Equipment
DCE	: Data Communication Equipment
LRC	: Longitudinal Redundancy Check
VRC	: Vertical Redundancy Check
CRC	: Cyclic Redundancy Check
DSK	: DSP Starter Kit

TABLO LİSTESİ

	<u>Sayfa No</u>
Tablo 2.1. CCITT ve Bell Standartları	19
Tablo 3.1. 4800 bit/saniye için farksal kodlama ve 9600 bit/saniye için normal kodlama	29
Tablo 3.2. 9600 bit/saniye hızında kafes kodlaması için farksal kodlayıcı tablosu	30
Tablo 3.3. 9600 bit/saniye hızında iki alternatifli işaret durum tablosu	33
Tablo 5.1. V.32 Karıştırıcı Programı	48
Tablo 5.2. V.32 Kodlayıcı Programı	52
Tablo 5.3. Temel Bant İşaret Oluşturma Programı	53
Tablo 5.4. Örneklenmiş Yükseltilmiş Kosinüs Süzgeç Değerleri.....	59
Tablo 5.5. Örneklenmiş Sinüs ve Kosinüs İşaret Değerleri.....	59
Tablo 5.6. Temel Bant İşaret Süzme Programı	60
Tablo 5.7. Modülasyon Programı	60
Tablo 5.8. Butterworth Süzgeç Programı	61
Tablo 6.1. Hilbert Dönüştürücü Programı	70
Tablo 6.2. Kompleks Demodülatör Programı	71
Tablo 6.3. Uyarlamalı Dengeleyici Programı	74
Tablo 6.4. Uyarlamalı Dengeleyici Katsayı Güncelleme Programı	74
Tablo 6.5. Ters Vektör Döndürme Programı	75
Tablo 6.6. Açısal Hata Hesaplama Programı	76
Tablo 6.7. Vektör Döndürme Programı	76
Tablo 6.8. Gauss Gürültü Üretme Programı	81
Tablo 6.9. Test Amaçlı Kullanılan Sayısal Veri Dizisi.....	82
Tablo 6.10. Ters Farksal Dönüşüm Programı	83
Tablo 6.11. V.32 Çözücü Programı.....	84

ŞEKİL LİSTESİ

	<u>Sayfa No</u>
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 Yapısı.....	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 Yapısı	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 Yapısı.....	14
Şekil 2.14 : DTE – DCE Arası Bağlantı Yapısı.....	15
Şekil 2.15 : V.24 Elektriksel Seviyeleri.....	15
Şekil 2.16 : Eşzamansız Çerçeve Yapısı.....	16
Şekil 2.17 : Eşzamanlı Çerçeve Yapısı.....	17
Şekil 2.18 : LRC Oluşturma Yapısı.....	18
Şekil 2.19 : R=n/m Katlamalı Kodlayıcı Yapısı.....	20
Şekil 2.20 : Kafes Kodlamalı Modülatör.....	23
Şekil 2.21 : 32 – Nokta İşaret Diyagramındaki İşaret Kümelerinin Ayrıştırılması.....	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 ve 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 için Kafes Kodlama ile 32-nokta işaret durum diyagramı ve 4800 bit/saniyede ve öğrenme fazında kullanılan A, B, C ve D durumları.....	32
Şekil 4.1 : TMS320C5402 Sayısal İşaret İşlemci Blok Diyagramı.....	37
Şekil 4.2 : TMS320C5402 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 Şeması.....	46
Şekil 5.3 : Karıştırıcı Katı 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 Dönüşümlü Demodülatör Katı Blok Diyagramı.....	64
Şekil 6.2	: Demodülatör Programı Genel Akış Şeması.....	65
Şekil 6.3	: Hilbert Dönüşümü ve Demodülatör Blok Diyagramı.....	66
Şekil 6.4	: Hilbert Dönüştürücü Darbe, Genlik ve Faz Eğrileri.....	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 Kodlamalı 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 önemini arttırmıştır. Her insana ulaşabilecek kadar yayılmış olan ulusal ve uluslararası telefon şebekesi günümüzde en çok kullanılan iletişim ortamıdır. Sürekli artan hızlı ve güvenilir iletişim ihtiyacı, telefon şebekesi kullanılarak modemler tarafından karşılanır. Modemler, gelişen tümleşik devre ve sayısal işaret işleme teknolojileri sayesinde daha hızlı ve güvenilir veri iletişimi gerçekleştirirler. Modemler; endüstriyel ve askeri sistemler başta olmak üzere bir çok iletişim sisteminde, 2400 bit/saniye ve daha düşük hızlarda kullanılırlar.

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

Bu çalışmada; değişik sayısal işaret işlemcileri araştırılmıştır. TMS320C5402 sayısal işaret işlemcisi ve program geliştirme kiti seçilerek yazılım çalışmaları 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, karıştırıldıktan sonra kafes kodlama teknikleri ve kompleks modülatör ile modüle edilmiştir. Böylece örneklenmiş analog semboller oluşturulmuştur.

Kafes kodlama tekniğinin en önemli özelliği, genişletilmiş işaret kümeleri yardımıyla, kodlanmış işaret dizileri arasındaki en küçük Öklid uzaklığını mümkün olabilecek en büyük uzaklık yapabilecek kodlama ve modülasyon işlemlerinin kullanılmasıdır. Bu şekilde ardışık olarak oluşturulan temel bant işaretleri arasındaki uzaklık artırılarak kodlama kazancı sağlanır.

Demodülasyon işleminde ise örneklenmiş analog semboller Hilbert dönüşümü ve kompleks demodülatör yardımıyla temel banda indirilmiş ve Viterbi algoritması kullanılarak karar verilen sayısal veri grupları çözülerek test dizisi tekrar oluşturulmuştur.

Ayrıca etkin sayısal işaret işlemci programlama teknikleri araştırılmış ve bu teknikler kullanılarak gerekli olan toplam işlemci yükü azaltılmıştır. Kullanılan test dizilerinin boyutları ve içerikleri değiştirilerek gerçekleştirilen 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

Yüzyılımızın son çeyreğinde, elektronik ve bilgisayar teknolojilerinde meydana gelen gelişmeler hızlı ve güvenilir veri iletişim ihtiyacını arttırmıştır. Enformasyon çağına giren dünyamızda yüz milyonlarca bilgisayarın iletişim gereksinimi, tüm insanlara ulaşmış ulusal ve uluslararası telefon şebekesi üzerinden modemlerle çözülebilmektedir. Modemler, ilk olarak faz ve frekans modülasyon (PSK ve FSK) tekniklerinin kullanıldığı düşük hızlarda 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 günlük hayatımıza sokmuştur. Modem iletişiminin, daha hızlı ve güvenilir gerçekleştirilmesi 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 iletimi için, V.32 standardı ile tanımlanmış olan kafes kodlama modülasyon tekniği ve Viterbi algoritması kullanılarak TMS320C5402 sayısal işaret işlemcisi üzerinde modülatör ve demodülatör yapılarının gerçekleştirilmesi amaçlanmıştır. Ayrıca; demodülatör yapısında kullanılan Viterbi kod çözücüsünün hata başarımı (SER), gauss gürültüsü eklenmiş kanal benzetimi yapılarak değişik işaret gürültü oranlarında (SNR) incelenmiştir.

Modemler; sayısal veri iletişiminin mümkün olmadığı 300 – 3500Hz arası bant sınırlı telefon şebekesi kurulduktan sonra ortaya çıkmış cihazlar olarak, sayısal verileri haberleşme kanalının karakteristik özelliklerine uygun analog sembollere dönüştürerek iletirler. Modem haberleşmesinde, bu sınırlı bant içinde veri hızını arttırabilmek 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 düşünüldüğü ve ilk olarak Ungerboeck tarafından ortaya atılan kafes kodlama tekniği, günümüz yüksek hızlı modemlerinde kullanılan bir modülasyon şeklidir [1,2,3,4].

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

Bu modülasyon tekniğinde sonlu durumlu bir kodlayıcı ile sayısal veriler modüle edilmektedir. İletimi haberleşme ortamında gerçekleştirilmiş analog semboller Viterbi algoritmasının [8,9] kullanıldığı dizi kod çözücüsü tarafından demodüle edilir. Bu yapıda, genişletilmiş işaret kümeleri kullanılarak kodlanmış işaret dizileri arasındaki en küçük Öklid uzaklığını mümkün olabilecek en büyük uzaklık yapabilecek kodlama ve modülasyon işlemleri kullanılır [10,11]. Böyle bir modülasyon tekniğinin kullanılması ile ardışık olarak oluşturulan temel bant işaretleri arasındaki uzaklığın belli bir değerin üzerinde olması sağlanır. Kafes kodlamalı modülasyon, katlamalı kodların özel halidir. Modülasyon işlemi gruplara ayrılmış olan işaret kümeleri üzerinde yapılır. İşaret kümelerinde yer alan noktalar, birbirlerine uzaklıkları en büyük olacak şekilde belirlenir.

Üçüncü bölümde; ITU V.32 standardı [12] ile tanımlanmış olan, farksal ve katlamalı kodlama tekniklerini içeren kafes kodlamalı modülasyon yapısı incelenmiş ve özellikleri verilmiştir. ITU V.32 standardı, çift yönlü 9600 bit/saniye veri iletişim hızındaki uygulamalarda tanımlıdır. ITU-T' un 17. çalışma grubu tarafından yeniden gözden geçirilmiş olup Helsinki' de 1-12 Mart 1993 tarihleri arasında yapılan WTSC konferansında onaylanmış ve bugünkü halini almıştır.

Dördüncü bölümde, sayısal işaret işlemciler ile ilgili genel bilgiler verilmiştir. Bununla birlikte, "Texas Instruments" [13] firmasına ait TMS320C5402 kodlu sayısal işaret işlemcisi [14] ve yazılım geliştirme kiti [15] ile ilgili özellikler de bu bölümde açıklanmıştır. Sabit noktalı ve kayan noktalı olmak üzere iki farklı yapıda sayısal işaret işlemciler mevcuttur. Sabit noktalı sayısal işaret işlemcileri, kayan noktalı sayısal işaret işlemcilerine göre genellikle daha hızlı ve daha ucuzdurlar. Sayısal işaret işlemci seçiminde, hız ve fiyat faktörlerinin yanı sıra işlemcinin komut çevrim süresi ile komut kümesinin yapılacak olan uygulamaya uygunluğu da önemlidir. Yazılım geliştirme kitleri, seçilen sayısal işaret işlemci ile çalışmada

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

Beşinci bölümde, tasarlanan modülatör ile ilgili ayrıntılı bilgiler verilmiştir. Buna göre; modülasyon katına sayısal veri, bağlantılı terminalden seri arabirim ile 9600 bit/saniye hızında gelir. Modülasyon katına gelen sayısal veri, V.32 standardında tanımlanmış karıştırıcı çok terimlisine bölünerek karıştırılmış sayısal veri oluşturulur [16,17]. Modemin arayan veya aranan olması durumuna göre her iletişim yönünde farklı karıştırıcı çok terimlisi kullanılır. Karıştırma işlemi ile veri katarında rasgelelik yaratılarak enerjinin iletim bandında düzgün yayılması sağlanır. Enerjinin haberleşme kanalında düzgün yayılması, demodülasyon işlemini kolaylaştırır. Karıştırılmış seri veri katarı, V.32 gereklerine uygun olarak farksal ve katlamalı olarak kodlanır (TCM) ve 32 – nokta işaret diyagramına göre temel bant işareti oluşturulur. Her sembol aralığında bir tam sayı değeri alan temel bant işaretinin, sembol aralık geçişlerinde aldığı tam sayı değerleri nedeniyle süreksizlikler oluşur. Süreksizliklerin varolmasından dolayı; temel bant işareti, alçak geçiren süzgeçten geçirilir [18]. Alçak geçiren süzgeç çıkışındaki temel bant işareti, kompleks modülatör yardımı ile taşıyıcı frekansa ötelenerek modülasyon işlemi tamamlanır.

Altıncı bölümde, tasarlanan demodülatör katı açıklanmıştır. Demodülasyon işleminde; Hilbert dönüşümü [19] kullanılarak, alınan sembole dik ikinci bir sembol oluşturulur ve kompleks demodülatör kullanılarak temel bant işareti elde edilir. Karar verici, Viterbi algoritmasını kullanarak alınan işaretin beş bitlik karşılığını bulur. Karar verilen beş bitlik veri grubuna ters farksal dönüşüm ve çözme işlemi uygulandıktan sonra demodüle edilmiş sayısal veri katarını oluşturulur. Böylece demodülasyon işlemi gerçekleştirilmiş olur.

Bu çalışmada, V.32 standardına uygun modülasyon ve demodülasyon işlemleri sayısal işaret işlemcisi üzerinde gerçekleştirilmiş olup tasarlanan yazılım Ekler bölümünde verilmiştir. Ayrıca; altıncı bölümde demodülatör yapısında kullanılan Viterbi kod çözücüsünün gauss gürültülü kanaldaki hata başarımı değişik işaret gürültü 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 sınıflandırılabilir;

- 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 karşıdaki kişiden tekrarını istemesi ve konuşan kişinin tekrarlamasına karşılık gelir.
- Karşılıklı iki tarafın aynı anda haberleşme isteğinde bulunmaması kuralı, iki nokta arası haberleşmenin ilk kurallarının konulduğu dönemde geçerli olan fakat günümüzdeki teknolojik gelişim içinde işlevini yitiren bir özelliktir. Karşılıklı konuşmada aynı anda sadece bir kişinin konuşması prensibine dayandırılmıştır. Modem 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 çıkılarak geliştirilmiştir. Hata algılama ve düzeltme algoritmaları içerisinde bu özellik desteklenmektedir.

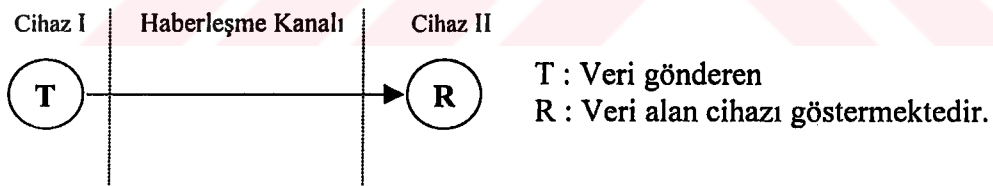
Yukarıda bahsedilen temel kurallar insanlar arasındaki iletişimin tabii kurallarına benzerlik göstermektedir. Geçmişten günümüze kadar geliştirilmiş bulunan tüm haberleşme yapılarında kullanılan algoritmalarda tasarımcılar insanları örnek almışlardır. Dikkatli incelendiğinde; haberleşme içinde bulunan iki cihaz, iki insanın karşılıklı konuşmasına benzer bir şekilde veri iletişiminde bulunurlar.

2.1.2 Haberleşme Yapıları

İki nokta arası haberleşmenin her iki tarafını oluşturan 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. Modem haberleşmesinde; bakır tel haberleşme kanalı, modem ile modeme bağlı olan terminal ise cihaz olarak adlandırılır.

2.1.2.1 Basit Haberleşme

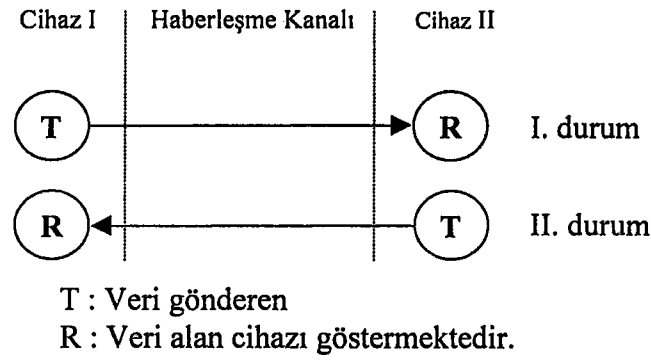
Haberleşen her iki cihazdan birisi sadece veri gönderir. Karşı tarafta bulunan cihaz, gönderilen verileri alır (Şekil 2.1). Diğer yönde veri iletişimi mümkün değildir. Haberleşme sistemi ve haberleşme kanalı buna imkan vermez. Bu tür haberleşme yapısına en iyi örnek televizyon haberleşmesidir. Televizyon gönderilen verileri sadece alabilir. Veri gönderme yeteneğine sahip değildir.



Şekil 2.1 Basit Haberleşme

2.1.2.2 Yarı Çift Yönlü Haberleşme

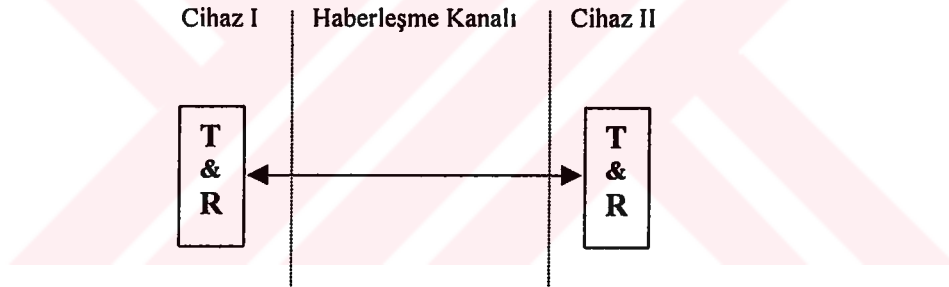
Haberleşen cihazlardan her biri aynı anda ya veri gönderen ya da veri alan konumunda olabilir (Şekil 2.2). Dolayısı ile, haberleşme herhangi bir anda sadece tek yönde olmaktadır. Bu haberleşme biçimine en iyi örnek telsiz haberleşmesidir.



Şekil 2.2 Yarı Çift Yönlü Haberleşme

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

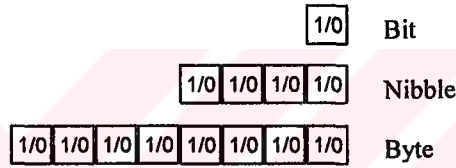
Tam çift yönlü haberleşme yapısının oluşturulmasında çeşitli metotlar kullanılmaktadır. Veri iletişimde tek haberleşme kanalı varsa; her cihaz, haberleşme kanalının değişik frekans bölgelerini veri gönderme ve alma işlemi için kullanır. Diğer bir yöntem ise; iki ayrı haberleşme kanalı kullanmaktır. Tek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

2.1.3 Protokol

Veri haberleşmesinin gerçekleştiği her iki cihaz arasındaki iletişiminin sağlıklı olarak yürüyebilmesi, 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

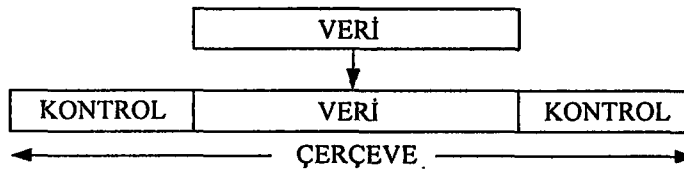
Karşılıklı olarak veri iletişimini gerçekleştiren her iki cihaz veriyi aynı yapıda kodlamalı ve çözmelidir. Cihazlar arasındaki iletişim; bit, 4 bitlik grup (nibble) veya 8 bitlik grup (byte) yapısında olabilmektedir (Şekil 2.4). Bir cihazın veriyi byte yapısında göndermesi karşıdaki cihazın veriyi byte yapısında değerlendirmesini gerektirir. Aksi takdirde verinin doğru iletilmesi mümkün değildir. Eşzamansız haberleşme bit yapısında, eşzamanlı haberleşme ise byte yapısında tanımlanmıştır.



Şekil 2.4 Örnek Veri Yapıları

2.1.3.2 Kontrol

Haberleşmenin sağlıklı yürüyebilmesi 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 Eşzamansız ve Eşzamanlı Haberleşme Yapıları" bölümünde açıklanan eşlik ve CRC gibi bilgileri içerir (Şekil 2.5).



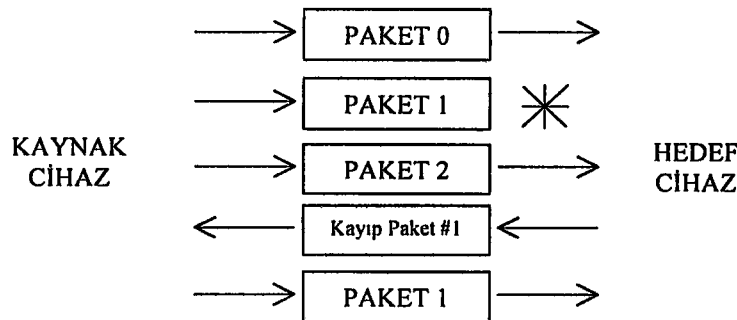
Şekil 2.5 Genel Çerçeve Yapısı

Kontrol niteliğinde kullanılan bilgilerin görevi, faydalı veri iletişiminin sağlıklı olarak gerçekleşmesini sağlamaktır. Veri iletişiminin verimliliği, kullanılan kontrol bilgilerinin faydalı veriye oranına bağlıdır. İletişimin sağlıklı olarak yürütülmesinden taviz vermeden, kontrol bilgilerinin faydalı veriye olan oranını daha da azaltmak iletişimin verimliliğini artırır. Eşzamanlı haberleşmedeki verim, eşzamansız haberleşmeye göre daha yüksektir. Dolayısı ile; eşzamanlı haberleşme hızının 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 bunun üzerine tekrar etmesi kendiliğinden ortaya çıkan doğal bir davranış biçimidir. Aynı şekilde iletişim sırasında oluşabilecek bir hatayı algılayabilmek ve algılanan bu hatayı düzeltebilmek haberleşmenin doğruluğunu ve güvenilirliğini etkiler. Dolayısı ile hata algılama ve düzeltme mekanizmaların haberleşme protokolü içinde oturtulmuş olması gereklidir.

Verici tarafından gönderilen veri, alıcıya haberleşme kanalından geçerek iletilmektedir. Dolayısı ile; hatanın oluşmasında haberleşme ortamının niteliği önem kazanmaktadır. Modern haberleşmesinde kullanılan haberleşme ortamı çift bükülmüş bakır tel kablodur. Bu haberleşme ortamı 300 – 3500Hz bant genişliğine sahiptir. Kullanılan haberleşme sinyalleri, bu bant genişliği içinde bulunan frekanslardır. Sınırlı bant genişliği, iletişimde kullanılan sinyaller ve ortam gürültüsünden dolayı veri iletişiminde hatalar oluşabilmektedir. İletişim sırasında oluşan hataları algılayabilmek kullanılan bir takım kontrol bilgileri ile mümkün olmaktadır. Eşzamansız haberleşmede eşlik kontrol biti, eşzamanlı haberleşmede ise CRC kontrol byte' ı kullanılır. (Bakınız : “2.5 Eşzamansız ve Eşzamanlı Haberleşme Yapıları”) Hatanın algılanmasından sonra, oluşan hatanın nasıl giderileceğinin hata düzeltme algoritmaları ile belirlenmiş olması gerekir. Şekil 2.6’ da görüldüğü gibi paket kaybında hedef cihaz, kaybolan paketi numarası ile kaynak cihazdan istemektedir.



Şekil 2.6 Örnek hata düzeltme algoritması

2.1.3.4 Haberleşme Kanal Verimliliği

Kullanılan haberleşme kanalının verimliliğinin, protokol içinde ele alınmış olması gerekmektedir. Kullanılan haberleşme kanalının özellikleri ve nitelikleri göz önünde tutularak modülasyon teknikleri geliştirilmiştir. Haberleşme ortamının özelliklerine uygun işaretleşme sinyallerinin kullanılması, hata oluşumunu engeller. Bunun yanında; faydalı bilginin sağlıklı olarak iletilmesini sağlamak amacı ile kullanılan kontrol bilgileri, kanal verimliliğinde önemli rol oynar. Haberleşmenin güvenilirliğinden ödün vermeden kontrol bilgilerini azaltabilmek kanal verimliliğini artırır. Ayrıca protokolde oturtulmuş algoritmalarında kanalın verimli kullanılıp kullanılmamasında önemli bir etkisi vardır.

2.1.3.5 Eşzamanlı Çalışma

Veri iletişimi sırasında her iki tarafın, uygun adım içinde kalmaları yani birbirlerini anlamaları garanti altına alınmış olmalıdır. Daha öncede belirtildiği gibi veri iletişimi çerçeve yapısındadır. Çerçeve başı ve çerçeve sonu kontrol bilgileri yardımı ile her iki cihaz karşılıklı olarak birbirlerine ayak uydururlar. Bir başka deyişle eşzamanlı çalışmayı sağlarlar. Eş zamanlamanın kaybolması veri haberleşmesinin bozulmasına sebep olur. Çerçeve başı ve sonu bilinemediği için çerçevenin yorumlanması mümkün olamaz. Çerçeve başı bilgisinin yakalanması eş zamanlamanın tekrar sağlandığı anlamına gelir.

2.1.3.6 Saydamlık

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

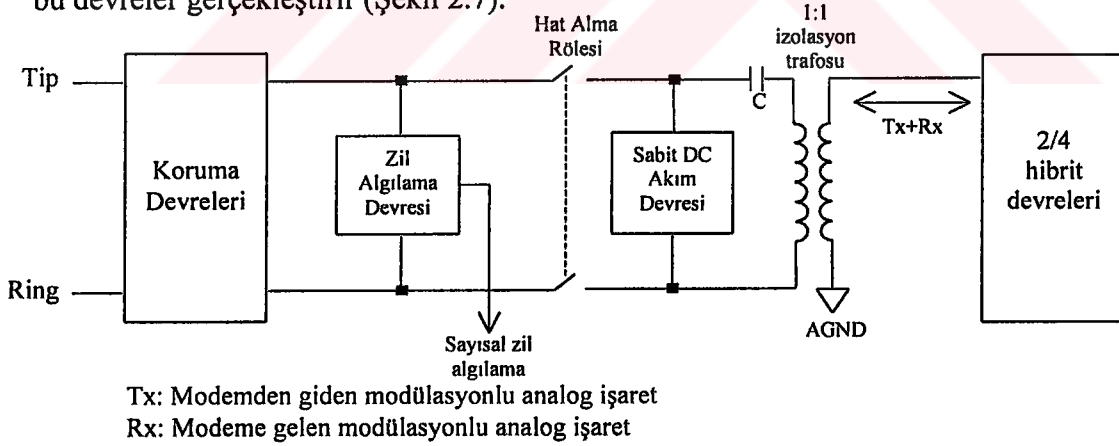
2.2 MODEM DONANIM ÖZELLİKLERİ

Modem, veri haberleşmesinde kullanılan bir haberleşme cihazıdır. Temel prensibi, verileri ikili sayısal yapıdan analog yapıya dönüştürerek yaygın olarak kullanılan telefon hatları üzerinden iletmektir. 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 bağlantıda telefon şebekesi kullanılır. Modem telefon hattını alır ve bağlantı kuracağı modem telefon numarasını arayarak bağlantı kurulur. Arayan modem arayan (Orginate), aranan modem cevaplayan (Answer) konumundadır. Veri iletişimi telefon santralleri üzerinden gerçekleştirilir. Kiralık devre bağlantı şeklinde ise; iki modem uzun süre arada telefon santrali olmadan veri iletişimi yaparlar. 2 veya 4 tel bağlantı olmak üzere iki bağlantı şekli vardır.

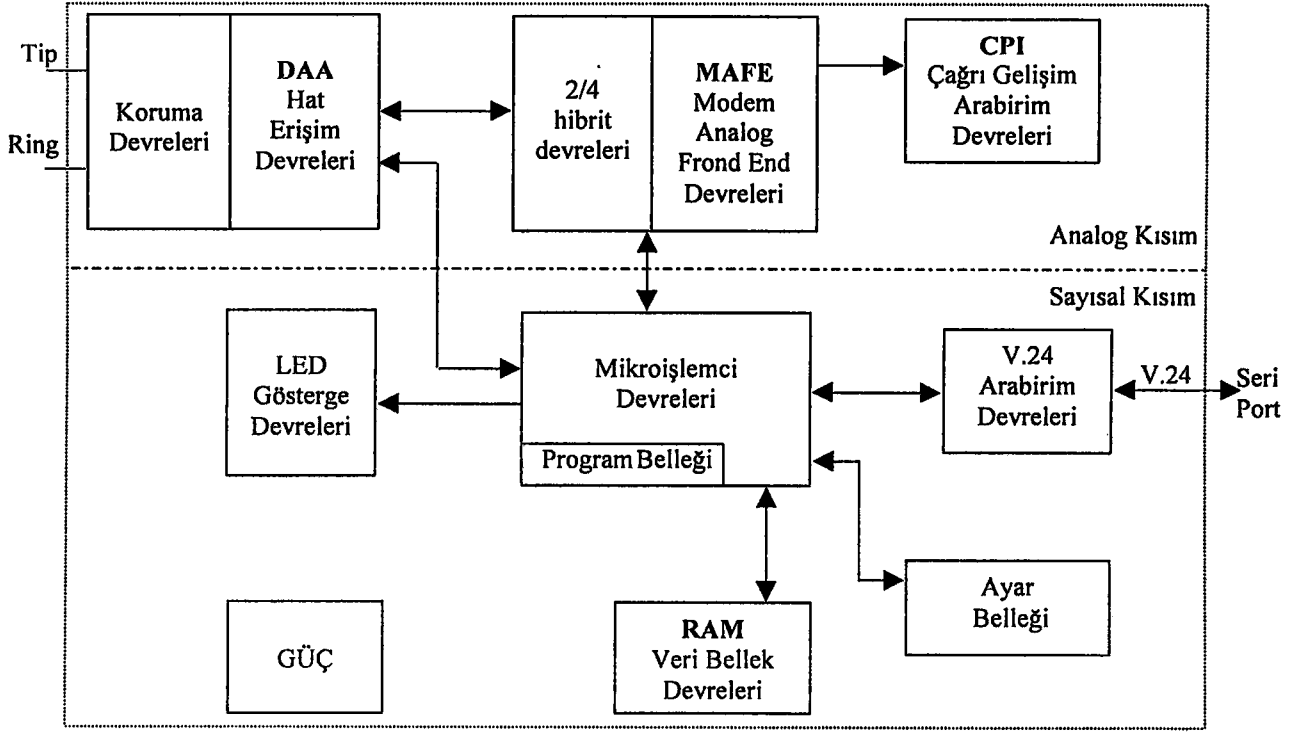
Modemlerde değişik modülasyon teknikleri kullanılarak aynı iletişim ortamında veri hızının artımı sağlanabilmektedir. Modem hızlarının artmasını engelleyen faktörlerin başında frekans bandının sınırlı olması ve iletişim ortamlarının karakteristik özellikleri gelir. Şekil 2.8' de modeme ait genel blok diyagram gösterilmiştir.. Modem sayısal ve analog olarak iki kısımdan oluşur. Sayısal kısım; mikroişlemci devreleri, bellekler, ışıklı gösterge devreleri ve V.24 arabirim devrelerini içerir. Analog kısım ise; koruma devreleri, hat erişim devreleri, hibrit devreleri ile modülasyon ve demodülasyonun yapıldığı MAFE (Modem Analog Frond 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 kullanılırlar. Hat erişim devreleri, modem dış hat arabirimi optik ve galvanik izolasyon sağlayarak oluşturan devrelerdir. Zilin algılanması, bağlı bulunan telefon santraline döngü akımının akıtılması gibi işlemleri bu devreler gerçekleştirir (Şekil 2.7).



Şekil 2.7 Tipik Hat Erişim Devresi (Dial-Up)

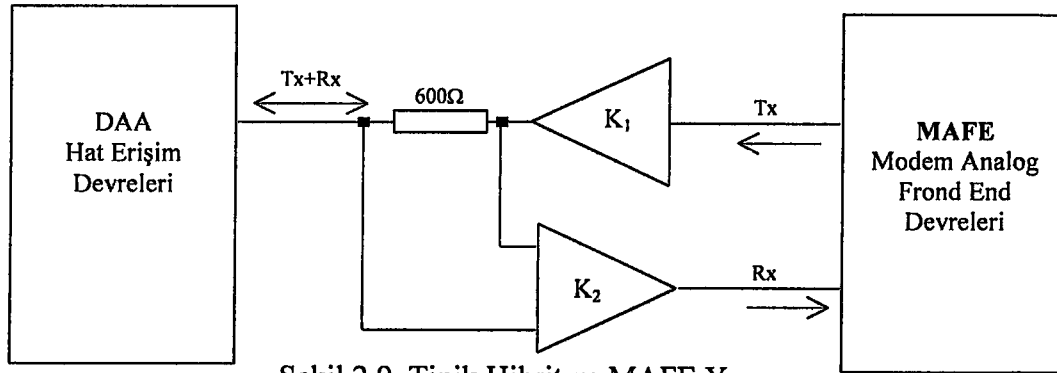
Zil algılama devresi, optik izolasyon sağlayarak elektriksel zil işaretini sayısal işarete dönüştürür. Bağlı bulunan telefon santraline hat alındığında, sabit DC akım devresi ile döngü akımının akıtılması sağlanır. İzolasyon trafosu, galvanik izolasyon gerçekleştirmek amacı ile kullanılır.



Şekil 2.8 Modem Blok Diyagramı

Kiralık devre modem dış hat arabirim devresinde ise, galvanik izolasyonun sağlandığı 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).



Şekil 2.9 Tipik Hibrit ve MAFE Yapısı

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

edilerek mikroişlemci devrelerine gönderilir. Modüle ve demodüle işlemleri sayısal işaret işlemcileri ile gerçekleştirilir.

Modemin çalışmasının kullanıcı tarafından kolay izlenebilir olması gereklidir. Bunun için modemlerde ışıklı göstergeler ve hoparlör gibi elemanlar kullanılır. Işıklı göstergeler yardımı ile modemin çalışması, bağlantıda olması, veri iletişimi ve fiziksel bağlantıların durumu gibi modem haberleşmesine ait değişik durumlar gözlemlenir. Ayrıca modem haberleşmesinde, modülasyonlu ve demodülasyonlu işaretlerin duysal olarak işitilmesi modemin çalışması hakkında fikir verir.

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

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

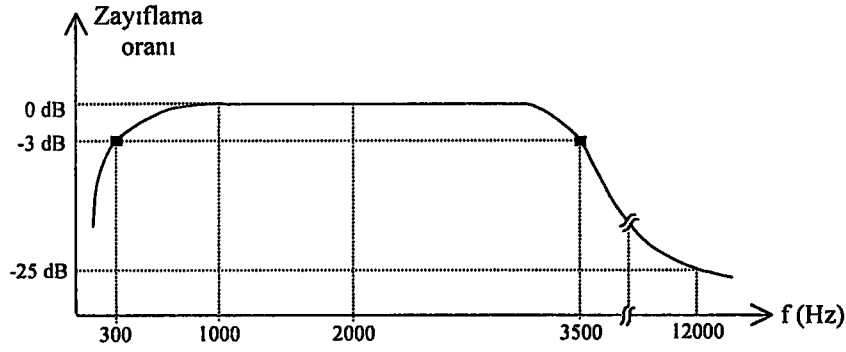
2.3 MODEM BAĞLANTI HATLARININ İNCELENMESİ

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

2.3.1 Ortalama Bant Genişliği

Telefon haberleşmesinin geliştirildiği dönemlerde; kullanılacak kablolar insanların konuşma karakteristiğine göre tasarlanmışlardır. İnsanların kullandıkları frekans bölgesi; 300Hz bölgesinden başlayarak 3500Hz bölgelerine kadar çıkar. İnsanın bulunmuş olduğu ruh hali ve fizyolojik yapısı insanın 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 oldukları için varolan telefon şebekesini kullanabilmeleri amaçlanmıştır. Bunun için, modemler iletişimlerini bu bant içinde yapmak zorundadırlar.

Ortalama Bant genişliği, seçilen referans frekansına göre 3dB zayıflamanın olduğu frekanslar arası tanımlanmıştır. Kuzey Amerika'da referans frekansı 1KHz, Avrupa'da ise 800Hz kullanılır (Şekil 2.10).

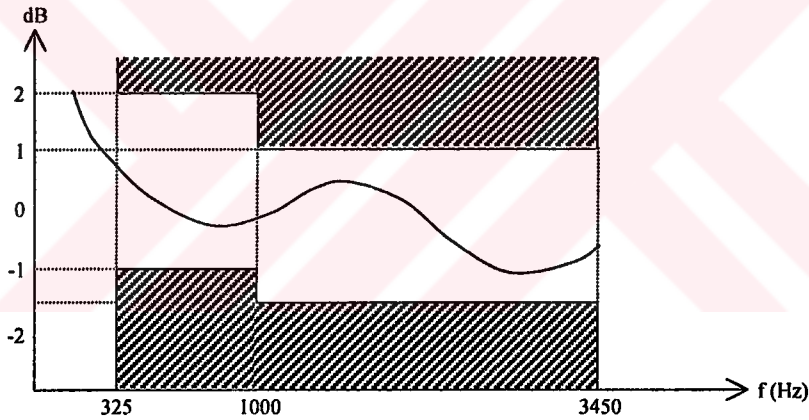


Şekil 2.10 Haberleşme Kanalı Bant Genişliği

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 kalması gerektiğini tanımlar (Şekil 2.11).



Şekil 2.11 G.132 Frekans – Genlik Karakteristiği (Attenuation Distortion)

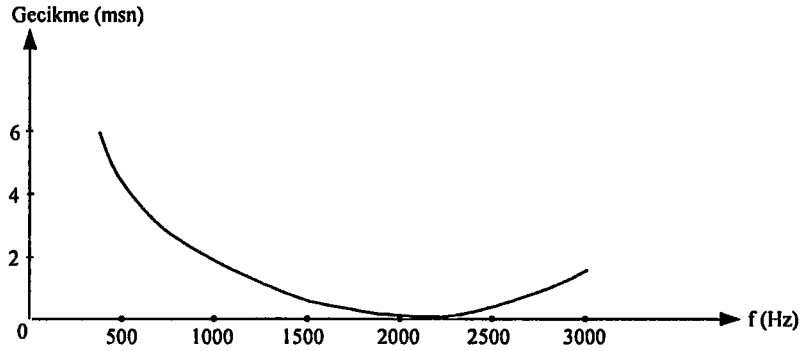
2.3.3 Gürültü

Mutlak sıcaklık ve boltzmann sabiti ile orantılı olan ısısal gürültü ortalama olarak oda sıcaklığında -100dBm seviyesinde oluşan beyaz gürültüdür. Bu gürültü biçiminin engellenmesi mümkün değildir. Pratik çalışmalarda beyaz gürültü sınırı -60dBm olarak kullanılır.

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

2.3.4 Faz Gecikmesi

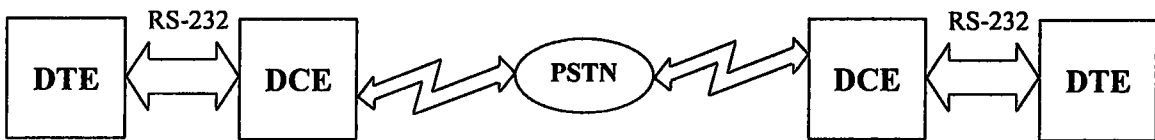
Çeşitli frekans bileşenlerine sahip işaret, iletişim ortamından geçerken frekans bileşenleri farklı gecikmelere uğrar (Şekil 2.12). Bu gecikme bant ortasında az olmakla birlikte bant kenarlarına ve bant dışına çıkmalarda artar. Bu kanal özelliğinden dolayı, hatlarda çıkılabilecek hızlar sınırlı kalmaktadır. Yüksek hızlarda önce gönderilen modülasyonlu işaretin sonraki işarete göre geride kalma problemi ortaya çıkar. Sonuç olarak, iki veri arasındaki bekleme süresi gecikmeden fazla olmalıdır. Bu da, iletişim hızını sınırlamaktadır.



Şekil 2.12 Frekans – Gecikme Karakteristiği

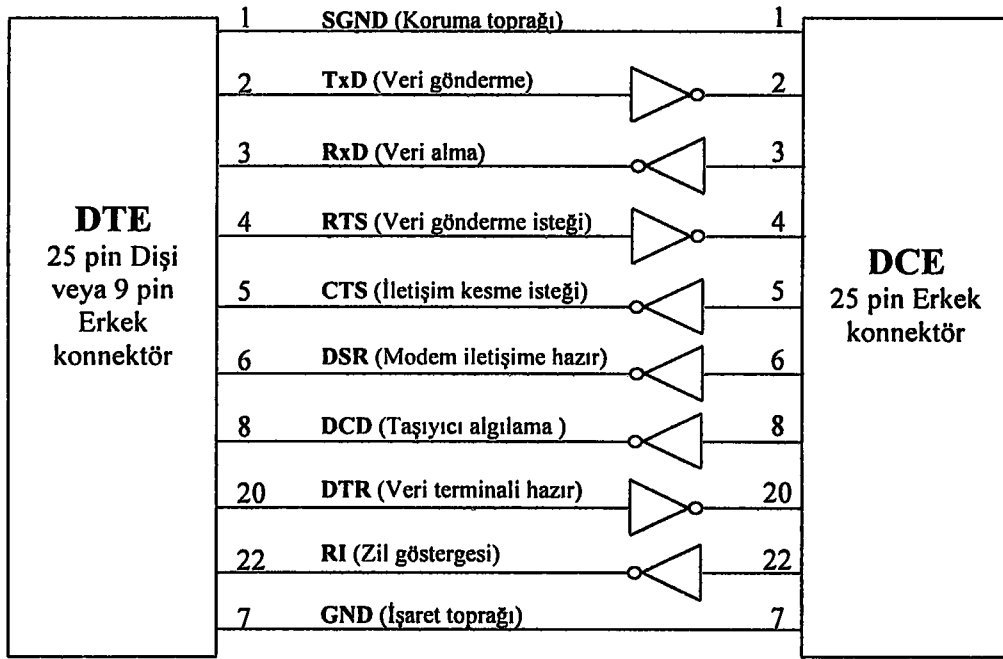
2.4 CCITT V.24 ÖZELLİKLERİ

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.



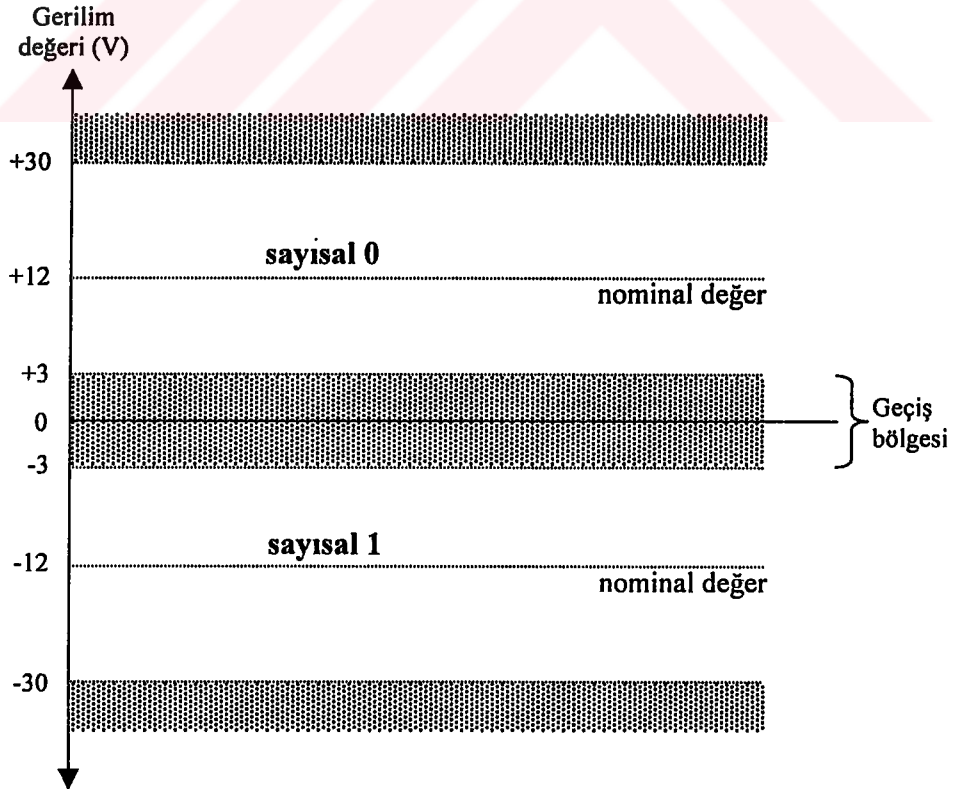
Şekil 2.13 Modem Haberleşme Yapısı

Modem haberleşmesinin başlayabilmesi DTE – DCE arası fiziksel bağlantının kurulmuş olması ile mümkündür. DTR ve DSR bağlantılarının karşılıklı olarak aktif olması bu fiziksel bağlantının kurulduğunu gösterir. Karşılıklı olarak modemlerin bağlantı kurması terminallere DCD bağlantısı aktif edilmesi ile bildirilir. Modem bağlantısının kurulması ile TxD ve RxD bağlantıları veri iletişimini DTE – DCE arasında gerçekleştirir.



Şekil 2.14 DTE – DCE Arası Bağlantı Yapısı

Terminal veri gönderdiği süre boyunca RTS' i aktif etmek zorundadır. Eğer Terminal hızı bağlantı hızından yüksek ise ve terminalden sürekli olarak bilgi akışı varsa modemın tampon belleğinin dolması kaçınılmaz bir sonuçtur. Bu durumda veri akışının kesilmesi gerekmektedir. Buda modemden CTS işaretinin aktif edilmesi ile olur. CTS' in aktif olduğunu gören terminal RTS' i keserek veri akışını durdurmak zorundadır. Bu olaya RTS-CTS el sıkışması denir.



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.

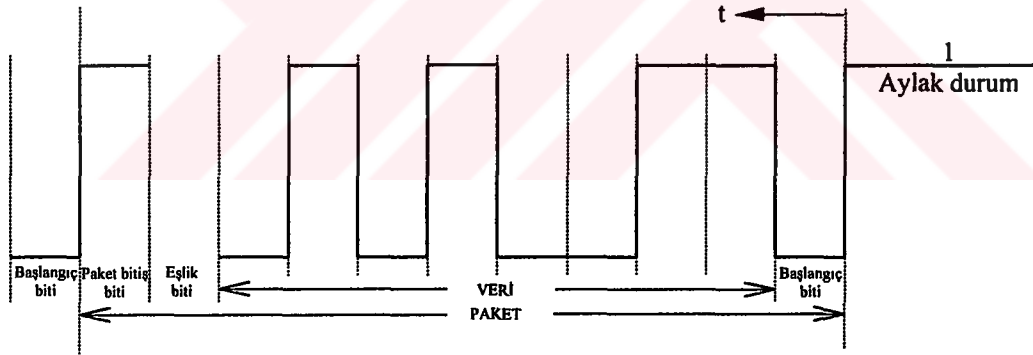
Bu şekilde simetrik kullanılması ile zamanda yapılan iletişimden dolayı hat üzerinde oluşan DC seviye sıfıra yakın tutulur. Bundan dolayı; V.24 bağlantısı uzak mesafelere kolaylıkla taşınabilir.

2.5 EŞZAMANSIZ VE EŞZAMANLI HABERLEŞME YAPILARI

Modemler bağlantı kurduktan sonra veri iletişimini iki farklı yapıda gerçekleştirirler. 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 yapısıdır. Aylak 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 algılamada kullanılan eşlik biti gönderilir. Eşlik biti sonrası çerçeve sonu biti ile bir çerçeve tamamlanmış olunur (Şekil 2.16).



Şekil 2.16 Eşzamansız Çerçeve Yapısı

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

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

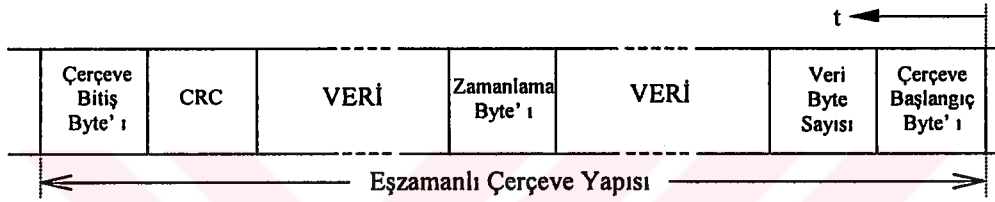
Eşlik biti, hata algılama için kullanılan bir kontrol biti olup çift eşlik, tek eşlik, 1, 0 ve kullanılmama gibi beş değişik seçeneği bulunur. Çerçeve içerisindeki tek sayıda bit değişimlerini algılayabilecek yapıdadır. İletişim sırasında meydana gelecek çift

bit deęişim 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 düşüktür. Fakat hata oluşumu 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' ın dan oluşan bir çerçeve yapısıdır (Şekil 2.17). Verinin uzun olması durumunda senkronizasyonun kaybedilmemesi için zamanlama byte' ları da kullanılır.



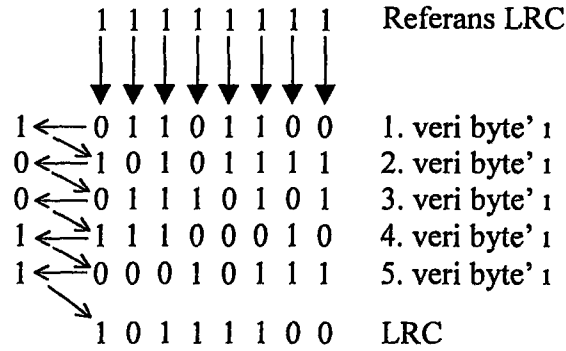
Şekil 2.17 Eşzamanlı Çerçeve Yapısı

Eşzamanlı veri yapısındaki amaç; taşınan fazlalıkları minimize etmek ve hatayı daha iyi algılayabilmek böylece hatasızlığa ulaşabilmektir. Bu iki sebepten dolayı senkron haberleşme tercih edilir. Senkron haberleşmede gönderilen veri mümkün olduğu kadar uzun (maksimum 256 byte) tutulur. Çerçevenin başında, başlangıç byte' ından sonra verinin uzunluğu belirtilir. Bundan dolayı senkron çerçeve uzunluğu değişkendir. Gelen veri içerisinden saat işaretinin çıkarılabilmesi için çerçeve, Biphase veya Manchester teknikleri ile kodlanarak kullanılır. Verinin uzun olması durumunda, saat işaretinin çıkarılamaması gibi sorunlarla karşılaşmamak için zamanlama byte' ları kullanılır. Hata algılama CRC byte' ı ile yapılmaktadır. İletişim, CRC byte' ı kullanıldığı için daha güvenli ve çerçevenin uzun olmasından dolayı da hız daha yüksektir. CRC kontrol byte' ının oluşturulması, LRC ve VRC kontrol byte' ları ile birlikte açıklanabilir.

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' ı oluşturulur ve veri sonrasında gönderilir (Şekil 2.18). Alıcı taraf gönderilen paket içerisindeki veri ve LRC byte' ını, kendi referans LRC' sini kullanarak EXOR işlemine tabi tutar ve her bit sütunu için 0 sonucuna ulaşmaya çalışır. Sonucun 00h olması hatanın olmaması demektir. Bu teknikte byte'

lar arası bitlerin kontrolü yapılır. Her iki tarafında kullandığı referans LRC nin aynı olma zorunluluğu vardır.



Şekil 2.18 LRC Oluşturma Yapısı

2.5.2.2 VRC (Vertical Redundancy Check)

LRC ile aynı mantığa dayanır. Byte içindeki bitlerin EXOR işlemine tabi tutulması ile oluşturulur. LRC ve VRC beraber kullanılarak hata algılama gerçekleştirilir. VRC ile hangi byte' da hata olduğunu, LRC ile hangi bitte hata olduğu gözlenir. Fakat ideal bir yöntem değildir. Çift hatalı byte içinde çift bit hatalarını algılayamazlar.

2.5.2.3 CRC (Cyclic Redundancy Check)

Gönderilecek veri (2.1), bir çok terimliye oturtulur ve bir üreteç çok terimlinin derecesi kadar kaydırılıp üreteç çok terimliye bölünür (2.2). Kalan değer ötelenmiş çok terimliye eklenerek gönderilecek veri oluşturulur (2.3). Hata yakalama olasılığı çok yüksek olan bu sistem tek ve çift bit hataları, hata patlamaları gibi sistematik oluşumları algılayabilir.

$$M(x) = m_{k-1}.x^{k-1} + m_{k-2}.x^{k-2} + \dots + m_1.x + m_0 \quad k : \text{Veri bit sayısı} \quad (2.1)$$

$$\frac{M(x).x^g}{G(x)} = Q(x) \oplus \frac{R(x)}{G(x)} \quad \begin{array}{l} G(x) : \text{Üreteç çok terimlisi} \\ g : \text{Üreteç çok terimli derecesi} \\ Q(x) : \text{Bölüm sonuç çok terimlisi} \\ R(x) : \text{Kalan sonuç çok terimlisi} \end{array} \quad (2.2)$$

$$T(x) = x^g . M(x) \oplus R(x) \quad T(x) : \text{Gönderilecek veri} \quad (2.3)$$

Karşı tarafa gönderilen veri (T(x)), (2.4) denkleminde gösterildiği gibi CRC çok terimlisine bölünerek hata kontrolü yapılır. T(x)' in hatasız bir şekilde iletimi R(x) in olmamasını gerektirir.

$$\frac{T(x)}{G(x)} = \frac{x^g \cdot M(x) \oplus R(x)}{G(x)} = Q(x) \oplus \frac{R(x)}{G(x)} \oplus \frac{R(x)}{G(x)} \quad (2.4)$$

CRC-12, CRC-16, CRC-CCITT ve CRC-32 gibi standart üreteç çok terimliliği 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 iletişimi modülasyonlu işaretler ile yapılır. Sayısal verinin, haberleşme kanalında kullanılan sembollere dönüştürülmesi işlemine modülasyon denir. Bu dönüşüm işleminin tersi demodülasyon olarak adlandırılır. Tablo 2.1’ de CCITT ve Bell tarafından belirlenmiş olan iletişim standartları ve modülasyon yapıları gösterilmiştir.

Tablo 2.1 CCITT ve Bell Standartları

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

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

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

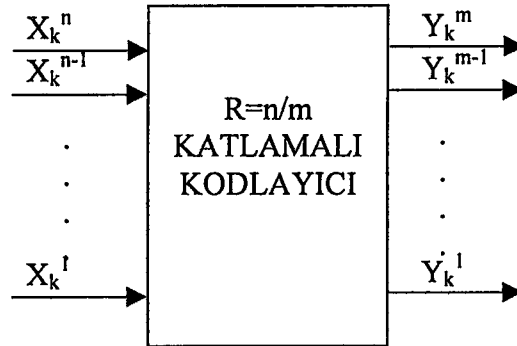
2.6.1 Giriş

Kafes kodlamalı modülasyon (TCM) tekniği, genişletilmiş işaret diyagramları kullanılarak kodlanmış işaret dizileri arasındaki en küçük Öklid uzaklığını artırma ilkesine dayanır. İletim sembollerinin işaret diyagramı üzerinde birbirlerinden uzak seçilmesi, haberleşme ortamındaki gürültülerden kaynaklanan sezme hatalarını azaltmakta ve bant sınırlı haberleşme kanallarındaki hata başarımları artmaktadır.

Sayısal haberleşme sistem tasarımlarında, hata başarımının yüksek tutulması amaçlanmaktadır. Hata başarımı, kullanılan kanal kodlaması teknikleri ile artırılabilir. Kodlanmış sayısal veriler arasındaki Hamming uzaklıkları, karar verme işleminde kullanıldığı için kanal kod tasarımlarının ölçütü olmaktadır. Kodlanmış sayısal veri uzunluğunun artması nedeniyle ortaya çıkan iletim hızı kaybı modülasyon hızının artırılması veya kanal kodlayıcılarının eklediği kontrol verilerine orantılı olarak genişletilmiş işaret diyagramları kullanılarak önlenir. Genişletilmiş işaret diyagramları kullanılarak yapılan tasarımlarda kodlama ve modülasyon işlemlerinin bağımsız olarak düşünülmesi, alıcıda demodülasyon ve kod çözme işlemlerinin ayrı yapılmasını gerektirir. Demodülasyon işleminde yapılan karar verme hatlarından kaynaklanan bilgi kayıpları nedeniyle bağımsız kodlama ve modülasyon işlemleri başarılı olamamaktadır.

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

2.6.2 Katlamalı Kod Yapıları



Şekil 2.19 R=n/m Katlamalı Kodlayıcı Yapısı

k (k=0,1,2...) kodlama adımını göstermek üzere; her k için, n uzunluğundaki x_k giriş dizisine karşılık m uzunluğunda 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.

$$X(D) = [X^1(D), X^2(D), \dots, X^n(D)] \quad (2.5a)$$

$$X^i(D) = X_0^i + X_1^i.D + X_2^i.D^2 + X_3^i.D^3 + \dots \quad 1 < i < n \quad (2.5b)$$

$$Y(D) = [Y^1(D), Y^2(D), \dots, Y^m(D)] \quad (2.6a)$$

$$Y^j(D) = Y_0^j + Y_1^j.D + Y_2^j.D^2 + Y_3^j.D^3 + \dots \quad 1 < j < m \quad (2.6b)$$

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

$$Y(D) = X(D) . G(D) \quad (2.7)$$

$G(D)$, n x m boyutlu bir matristir ve bu matrisin elemanları (2.8)' deki gibi tanımlanır.

$$G_r^j(D) = g_{0r}^j + g_{1r}^j.D + g_{2r}^j.D^2 + g_{3r}^j.D^3 + \dots \quad \begin{matrix} 1 < i < n \\ 1 < j < m \end{matrix} \quad (2.8)$$

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

$$v = \sum_{i=1}^n \max_j [\deg G_i^j(D)] \quad (2.9)$$

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

2.6.3 Kod Çözme İşlemi ve Özellikleri

Klasik haberleşme sistemlerinde modülatör ve hata düzeltmeli kodlayıcı ayrı ayrı tasarlanmaktadır. Bu tip sistemlerde her bir sembol iletim zaman aralığında a adet sayısal veriye karşılık 2^a olası 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 $d_{\min}^H$ olarak tanımlı ise kod çözücü en az $(d_{\min}^H - 1)/2$ kadar olan hataları düzeltebilir. Hata düzeltme yeteneğini arttırabilmek fazlalık kontrol verisini arttırmaya bağlıdır. Kod çözücü dilimlenmiş kanal çıkış örneklerini kullanır. Modülatörden gönderilen ayrık işaretler (a_n) , haberleşme ortamından kaynaklanan ve işarete eklenen sıfır ortalamalı ve σ^2 sapmalı beyaz Gauss gürültüsü örnekleri (w_n) ile birlikte r_n kanal çıkış örneklerini oluşturur (2.10).

$$r_n = a_n + w_n \quad (2.10)$$

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

$$SNR = \frac{E\{|a_n|^2\}}{E\{|w_n|^2\}} \quad (2.11a)$$

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

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

$$|r_n - a_n^I|^2 = \min_{\{a_n^I\} \in C} \sum |r_n - a_n^I|^2 \quad (2.12)$$

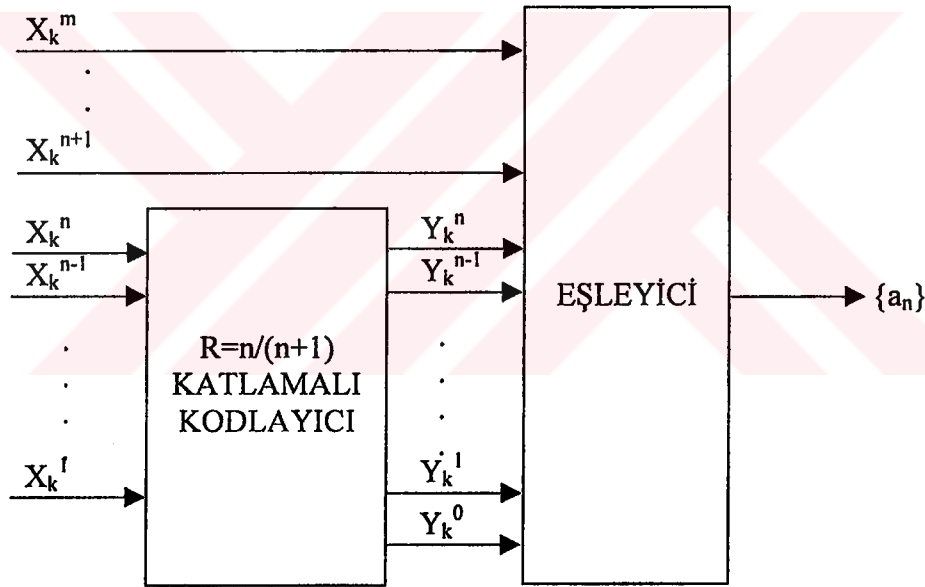
İlk olarak A. J. Viterbi tarafından 1967 yılında katlamalı kodların çözme işlemi için önerilen Viterbi algoritması, alıcıya gelen r_n dizisine en yakın kodlanmış olan a_n^I dizisine karar vermekte kullanılır (2.12). Vericide oluşturulan kodlanmış işaret dizileri (a_n) , katlamalı kodlayıcının belirlediği sonlu durum diyagramını izler. Çözücü yapısı, sert karar verme yerine çözücü girişindeki en olası 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 hatasını oluşturur. Minimum karesel Öklid uzaklığı (2.13)' deki gibi ifade edilir.

$$d_f^2 = \min_{\{a_n^l\} \neq \{a_n\}} \sum |a_n - a_n^l| \quad ; \{a_n\}, \{a_n^l\} \in C \quad (2.13)$$

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 belirlenirler.

Bu kurallar çerçevesinde kafes kodlama kullanılarak temel bant işaretinin üretimi incelenirse; iletilecek m adet sayısal veri biti için n adet bit ($n \leq m$), $n/(n+1)$ oranlı katlamalı kodlayıcıdan geçirilerek $n+1$ 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).



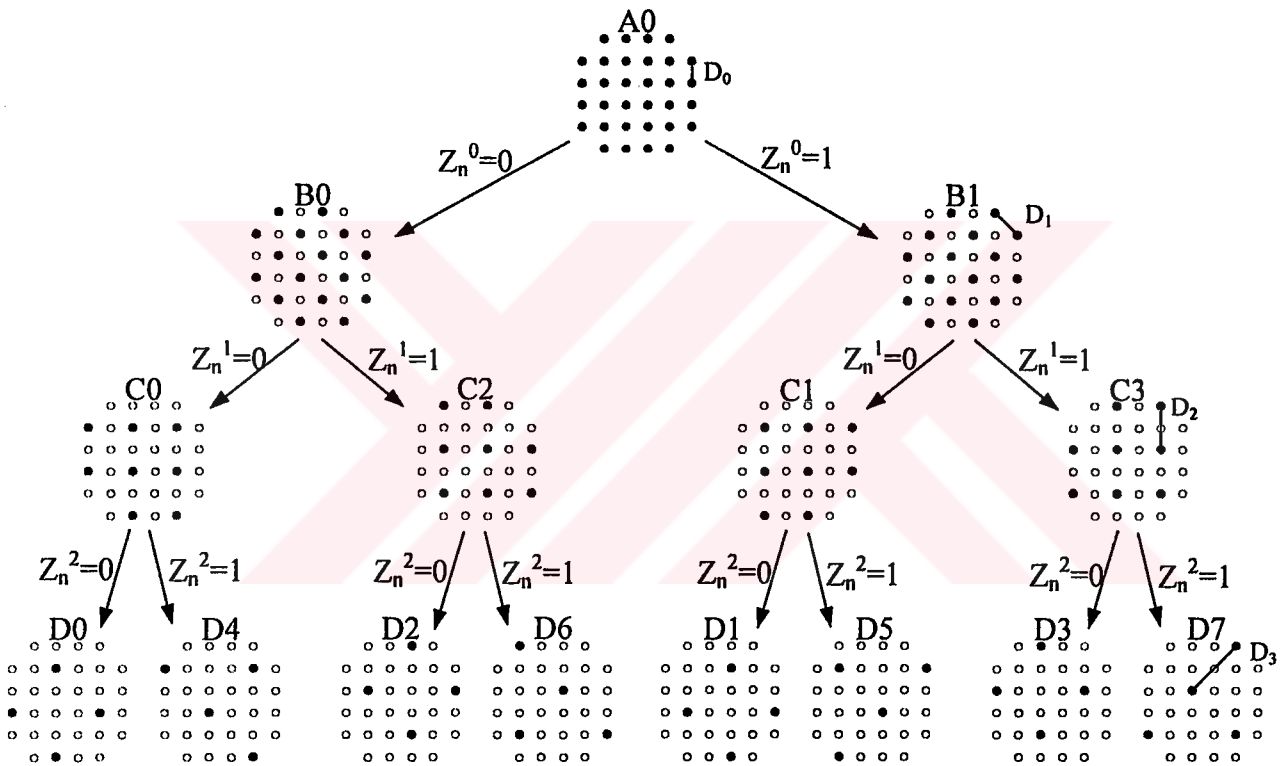
Şekil 2.20 Kafes Kodlamalı Modülatör

Şekil 2.21' de, kafes kodlama modülasyon yapısında kullanılan 32 nokta işaret diyagramındaki işaret kümelerinin ayrıştırılması görülmektedir. Z_k , tam sayı koordinatlı k boyutlu uzayda noktaların kafesini göstermede kullanılır. Bu tip işaret diyagramlarında, kafes noktalarının alt kümeleri sonludur. Bu alt kümeler orijin etrafında merkezlenmiştir. Aralarındaki en küçük uzaklık D_0 ' dır.

İşaret kümelerinin ayrıştırılması, işaret diyagramını sırayla daha küçük kümelere bölme işlemidir. Bu bölme işleminde küme içindeki işaretler arası uzaklıklar artar. Ayrıştırma işlemi, D_{n+1} tasarlanmak istenen kodun istenen işaretler arası uzaklıktan

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, $Z_n = [Z_n^m, \dots, Z_n^0]$ şeklinde göstermek mümkündür. Herhangi iki alt kümenin gösteriminde son q tane bitin uyuşması, bu iki alt kümenin ayrıştırma işleminde q . seviyedeki alt kümenin elemanları olmasını ve aralarındaki uzaklığın D_q olmasını gerektirir. $m-n$ adet kodlanmış veri, seçilen alt küme içinden bir noktaya karşılık düşer. Kafes 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.

$$d_f = \min [D_{n+1}, d_f(n)] \quad (2.14)$$



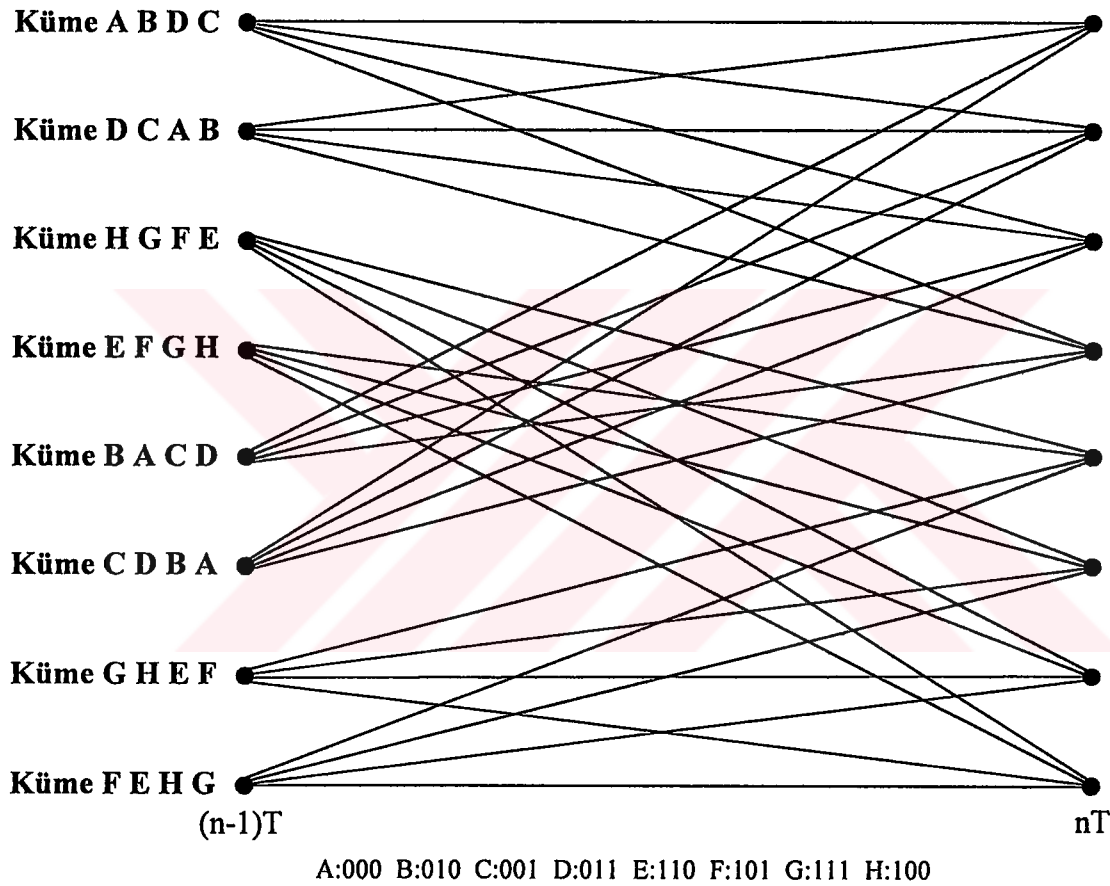
Şekil 2.21 32 – nokta İşaret Diyagramındaki İşaret Kümelerinin Ayrıştırılması

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 kodlayıcıda saklanarak önceden belirlenmiş mantıksal kombinasyonlarla uyumlu olacak şekilde $n+1$ bit grubuna genişletir. Kodlayıcı çıkışını etkileyen kodlayıcı belleklerinin sayısı (S), kodlayıcının varsaydığı 2^S konumlu sonlu durum makinesini belirler. (Bakınız : 5.4.1 Kodlayıcı ve Temel Bant İşareti Oluşturma Yapısı)

Haberleşme kanallarından kaynaklı bozulmalar, en büyük benzerlikli kod çözme algoritması ile giderilir. Alınan işaret, geçerli işaret elemanlarından en yakını olarak kabul edilir. Bu teknikte, geçerli işaret elemanları arasındaki en küçük Öklid uzaklığı arttırılmış olduğundan kodlama kazancı sağlanmaktadır.

2.6.5 32 – Nokta Katlamalı Kod

Katlamalı kodlayıcı durumlarının $S1_n$, $S2_n$, $S3_n$ olduğu ve işaret durum diyagramını oluşturan 32 noktanın 8 ayrı kümeye ayrıldığı kabul edilmektedir. Bu kodlamaya ait durum geçiş diyagramı Şekil 2.22’ de gösterilmiştir.



Şekil 2.22 Katlamalı Kodlama Küme Durum Geçiş Diyagramı

f_i ($S1_n$, $S2_n$, $S3_n$) durumundan f_i ($S1_{n+1}$, $S2_{n+1}$, $S3_{n+1}$) durumuna geçişlerle ilgili işaret eleman kümeleri, $S1_n$, $S2_n$, $S3_n$ durumundan $S1_{n+1}$, $S2_{n+1}$, $S3_{n+1}$ durumuna geçişle 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 kodlamanın 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 farksal kodlanmaları sonucu ortadan kaldırılır.

2.6.6 Kafes Kodlamalı Sistem Hata Olasılığı ve Viterbi Algoritması

Haberleşme kanalının gauss gürültülü olduğu öngörülürse; alıcıda Viterbi kod çözme yapısı kullanılacağından sistem hata olasılığı serbest uzaklıkla alttan sınırlıdır ve (2.15)' deki gibi tanımlanır [1].

$$\Pr(e) > N(df) \cdot Q(df/2\sigma) \quad (2.15)$$

σ^2 : gauss gürültü sapması

$N(df)$: df serbest uzaklıklı ortalama hatalı olay sayısı

$Q(\cdot)$: Gauss hata olasılık fonksiyonu

Viterbi algoritması, katlamalı kodların çözülmesinde kullanılan bir algoritmadır. Bu algoritmada belirli bir dizi uzunluğuna kadar semboller alınmakta ve sonra kod çözme işlemi başlamaktadır. İşaret alındığında karar verme işlemi yapılmamaktadır. Bahsedilen bu dizi uzunluğu kod çözme derinliği olarak adlandırılır. Hatanın azaltılabilmesi kod çözme derinliğinin artırılmasına bağlıdır.

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

$$M_n(\dots a_n) = M_{n-1}(\dots a_{n-1}) + |r_n - a_n|^2 \quad (2.16)$$

Alınan işaret sayısı kod çözme derinliğine ulaştığında tüm durumlara ait değerler karşılaştırılarak en küçük değer bulunur ve bu değerden geriye giderek geçiş elemanları belirlenir. Belirlenen bu dizi, Viterbi algoritmasına göre karar verilen en uygun gönderilen işaret dizisidir.

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.

V.32, ITU-T (International Telecommunication Union – Telecommunication Standardization Sector) 17. çalışma grubu tarafından yeniden gözden geçirilmiş ve Helsinki’ de 1 – 12 Mart 1993 tarihleri arasında yapılan WTSC (The World Telecommunication Standardization Conference) konferansında onaylanmış olup bugünkü halini almıştır.

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 tanımlanabilir;

- 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 iletişimiyle her bir kanal için dik genlik modülasyonu 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 modülasyon yapısı kullanma,
 - 16 işaretli dik genlik modülasyonu
 - 32 işaretli kafes kodlamalı modülasyon

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 iletme.
- g) V.14 veya V.42 standartlarına göre seçimli olarak eşzamansız çalışma.
- h) Karıştırma ve Çözme fonksiyonu.

3.3 HAT İŞARETLERİ

3.3.1 Taşıyıcı Frekansı ve Modülasyon Hızı

Taşıyıcı frekansı $1800 \pm 1\text{Hz}$ olarak tanımlıdır. Kılavuz ton kullanılmaz. Alıcı, alınan frekansı $\pm 7\text{Hz}$ toleransla değerlendirebilecek yapıda olmalıdır. Modülasyon hızı $2400 \text{ baud} \pm 0.01\%$ olarak kullanılır.

3.3.2 Spektrum

Gönderilecek ikili bitler karıştırıcı girişine uygulanır. $600\text{Hz} - 3000\text{Hz}$ arası spektrumda gönderilen enerji yoğunluğu, $600\text{Hz} - 3000\text{Hz}$ arası spektrumdaki maksimum enerji yoğunluğuna göre $4.5 \pm 2.5\text{dB}$ değişmelidir.

3.4 KODLAMA

3.4.1 9600 bit/saniye veri haberleşme hızında işaret kodlaması

9600 bit/saniye hızında iki değişik kodlama yapısı tanımlanmıştır.

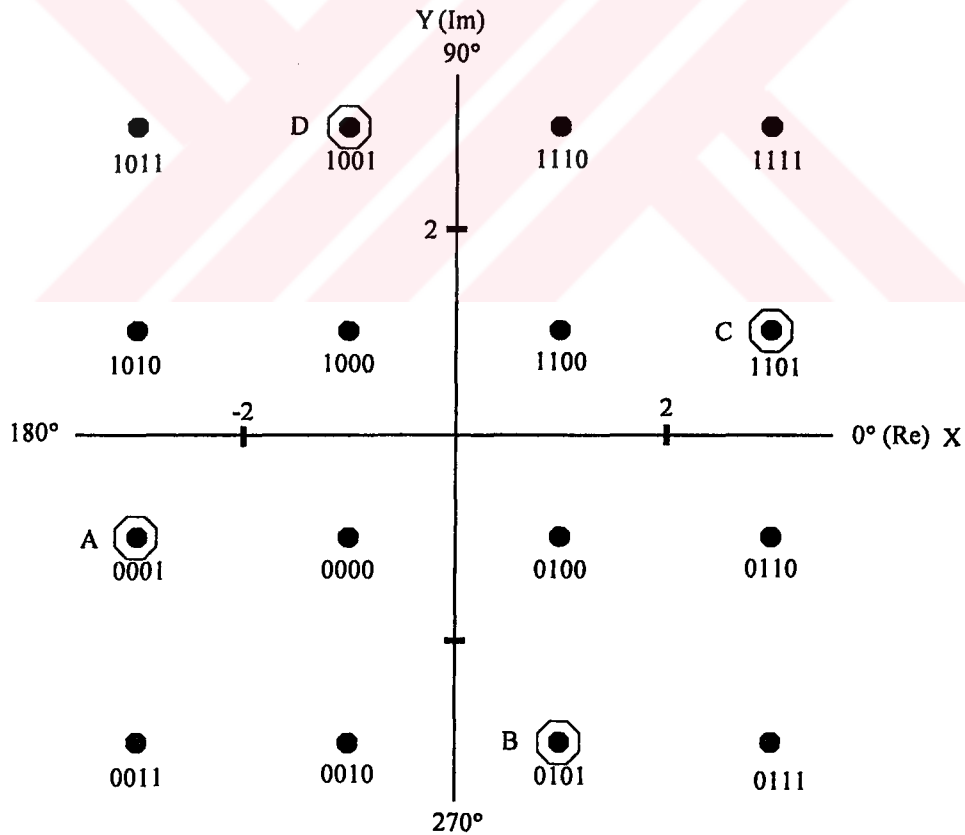
3.4.1.1 Normal Kodlama

Gönderilecek karıştırılmış veri katarı, 4 ardışık veri bitinden oluşan gruplara bölünür. Her grup içinde ilk iki bit olarak bulunan $Q1_n$ ve $Q2_n$, Tablo 3.1' e göre $Y1_n$ ve $Y2_n$ ile farksal kodlamaya tabi tutulurlar. Burada n, 4 ardışık veri bitinden oluşan grup numaralarını ifade etmektedir.

$Y1_n$, $Y2_n$, $Q3_n$ ve $Q4_n$ 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

Girişler		Önceki Çıkışlar		Faz değişimi	Çıkışlar		4800 bit/sn için işaret durumu
$Q1_n$	$Q2_n$	$Y1_{n-1}$	$Y2_{n-1}$		$Y1_n$	$Y2_n$	
0	0	0	0	+90°	0	1	B
0	0	0	1		1	1	C
0	0	1	0		0	0	A
0	0	1	1		1	0	D
0	1	0	0	0°	0	0	A
0	1	0	1		0	1	B
0	1	1	0		1	0	D
0	1	1	1		1	1	C
1	0	0	0	+180°	1	1	C
1	0	0	1		1	0	D
1	0	1	0		0	1	B
1	0	1	1		0	0	A
1	1	0	0	+270°	1	0	D
1	1	0	1		0	0	A
1	1	1	0		1	1	C
1	1	1	1		0	1	B



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

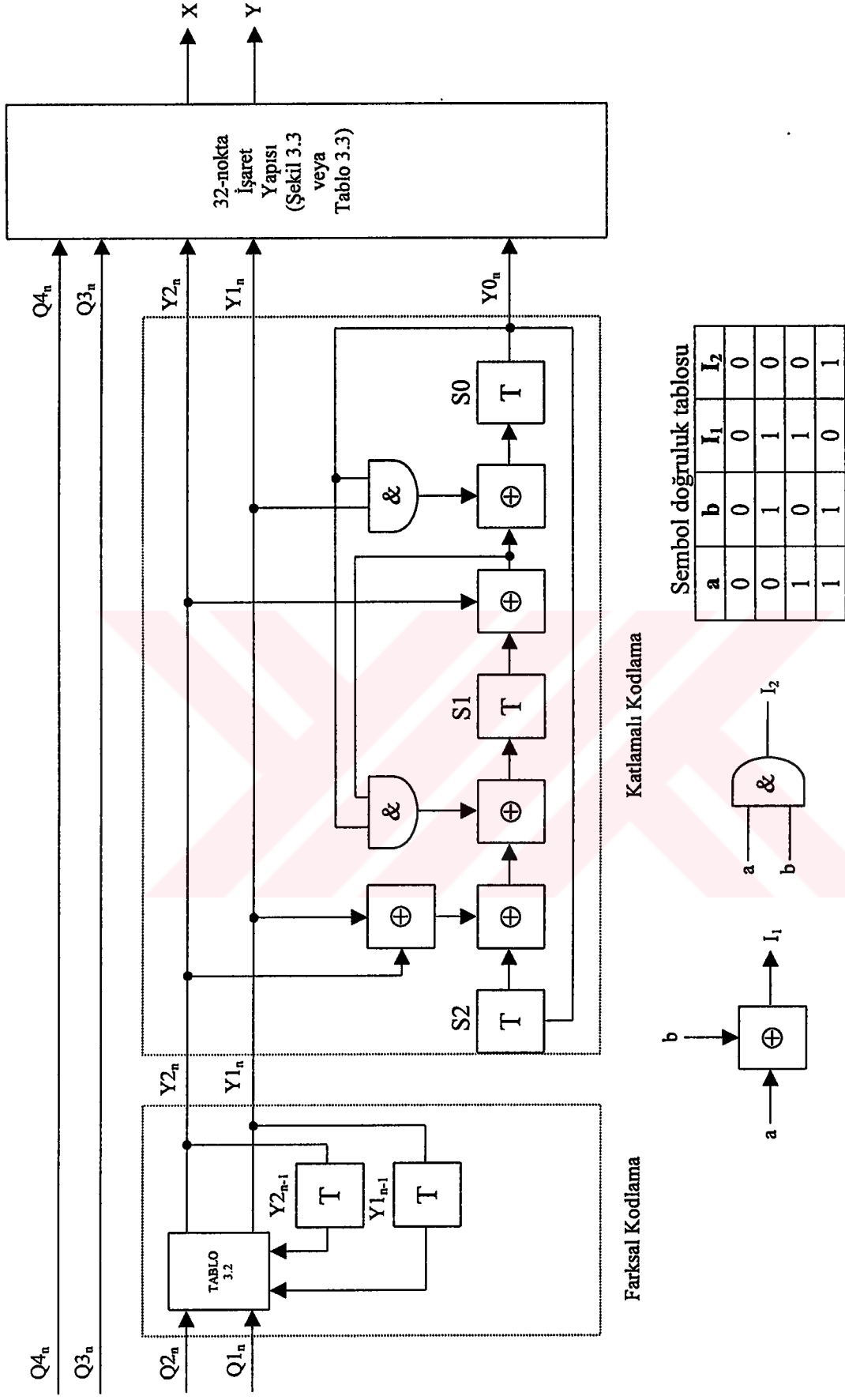
Gönderilecek karıştırılmış veri katarı, 4 ardışık veri bitinden oluşan gruplara bölünür. Şekil 3.2’ de gösterildiği gibi, her grup içinde ilk iki bit olarak bulunan $Q1_n$ ve $Q2_n$, ilk olarak Tablo 3.2’ ye göre $Y1_n$ ve $Y2_n$ ile farksal kodlamaya tabi tutulurlar. Burada n, 4 ardışık veri bitinden oluşan grup numaralarını ifade etmektedir.

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

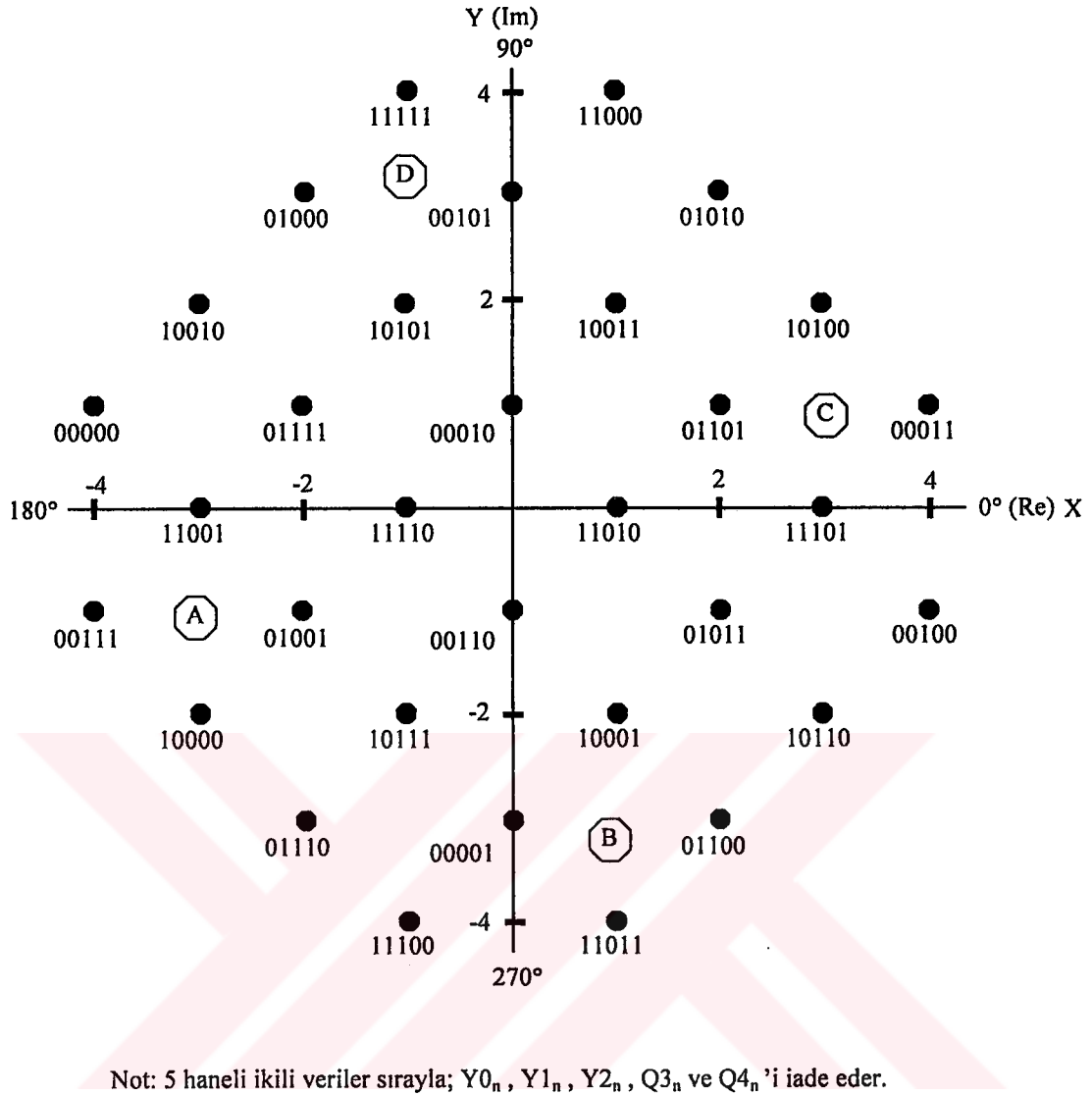
Girişler		Önceki Çıkışlar		Çıkışlar	
$Q1_n$	$Q2_n$	$Y1_{n-1}$	$Y2_{n-1}$	$Y1_n$	$Y2_n$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	0	0
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	1	0	1

Farksal kodlayıcı çıkışları olan $Y1_n$ ve $Y2_n$, ek $Y0_n$ bitinin oluşturulduğu simetrik katlamalı kodlayıcının girişleri olarak kullanılırlar.

Bu ek bit ($Y0_n$) ve bilgi taşıyan $Y1_n$, $Y2_n$, $Q3_n$ ve $Q4_n$ 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



Şekil 3.3 9600 bit/saniye için Kafes Kodlama ile 32-nokta işaret diyagramı ve 4800 bit/saniyede ve öğrenme fazında kullanılan A, B, C ve D durumları.

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 farksal 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

Kodlanmış Girişler					Normal Kodlama		Kafes Kodlama	
(Y0)	Y1	Y2	Q3	Q4	Re(X)	Im(Y)	Re(X)	Im(Y)
0	0	0	0	0	-1	-1	-4	1
0	0	0	0	1	-3	-1	0	-3
0	0	0	1	0	-1	-3	0	1
0	0	0	1	1	-3	-3	4	1
0	0	1	0	0	1	-1	4	-1
0	0	1	0	1	1	-3	0	3
0	0	1	1	0	3	-1	0	-1
0	0	1	1	1	3	-3	-4	-1
0	1	0	0	0	-1	1	-2	3
0	1	0	0	1	-1	3	-2	-1
0	1	0	1	0	-3	1	2	3
0	1	0	1	1	-3	3	2	-1
0	1	1	0	0	1	1	2	-3
0	1	1	0	1	3	1	2	1
0	1	1	1	0	1	3	-2	-3
0	1	1	1	1	3	3	-2	1
1	0	0	0	0	-	-	-3	-2
1	0	0	0	1	-	-	1	-2
1	0	0	1	0	-	-	-3	2
1	0	0	1	1	-	-	1	2
1	0	1	0	0	-	-	3	2
1	0	1	0	1	-	-	-1	2
1	0	1	1	0	-	-	3	-2
1	0	1	1	1	-	-	-1	-2
1	1	0	0	0	-	-	1	4
1	1	0	0	1	-	-	-3	0
1	1	0	1	0	-	-	1	0
1	1	0	1	1	-	-	1	-4
1	1	1	0	0	-	-	-1	-4
1	1	1	0	1	-	-	3	0
1	1	1	1	0	-	-	-1	0
1	1	1	1	1	-	-	-1	4

3.5 KARIŞTIRICI (SCRAMBLER) VE ÇÖZÜCÜ (DESCRAMBLER)

Karıştırma ve çözme işlemleri modem tarafından sağlanmalıdır. Her iletişim yönü farklı karıştırıcı kullanır. İletişimin yönüne göre tanımlanmış ve oluşturulmuş olan çok terimliler (3.1) ve (3.2)' de gösterilmiştir.

$$\text{Arayan modem karıştırıcı çok terimlisi (GPC)} = 1 + x^{-18} + x^{-23} \quad (3.1)$$

$$\text{Aranan modem karıştırıcı çok terimlisi (GPA)} = 1 + x^{-5} + x^{-23} \quad (3.2)$$

Modölatör katında, gönderilecek veri karıştırıcı çok terimlisine bölünür. Bu bölümün bölüm katsayısı, azalan sırayla alınır ve kodlama işlemi için karıştırılmış veri karıştırıcı çıkışında elde edilir. Karıştırma işleminin nedeni, veri dizininde uygun oranda rasgelelik yaratmaktır. Böylece enerji geçirme bandında düzgün yayılır. Enerjinin geçirme bandında düzgün yayılması demodülasyon işlemini kolaylaştırır.

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

Dial-up kullanımda arayan modem (originate); 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 dille gerçekleştirilebilir.

Bu çalışmada; “Texas Instruments” firmasına ait TMS320 sayısal işaret işlemci ailesinden C5000 grubu içerisinde yer alan, C5402 kodlu işlemci kullanılmıştır. V.32 modülatör ve demodülatör tasarımları seçilen sayısal işaret işlemci üzerinde yapılmıştır. Yazılım geliştirme ve tasarım sürecinde; bu işlemci için geliştirilmiş olan çalışma kiti, “DSP/BIOS” ve “Code Composer Studio” yazımları araç olarak kullanılmıştır.

4.2 SAYISAL İŞARET İŞLEMCİLER

Sayısal işaret işlemciler, duyarlılıklarına ve aritmetik işlem tiplerine göre sınıflandırılır. Sabit noktalı sayısal işaret işlemciler, daha hızlı ve daha ucuzdur. Ancak bunların programlanması daha zor olup daha düşük duyarlılık sağlarlar. Bir sayısal işaret işlemciadaki toplam kapı sayısı, kullanılan nümerik gösterimdeki bit sayısının bir fonksiyonudur. Kapı sayısı, kullanılan nümerik gösterimin mevcut aralığı ve işlemcinin duyarlılığı ile doğru orantılı olarak artar. Bu nedenle matematiksel başarımla devre ve sistem karmaşıklığı arasında bir seçim yapılmalıdır.

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

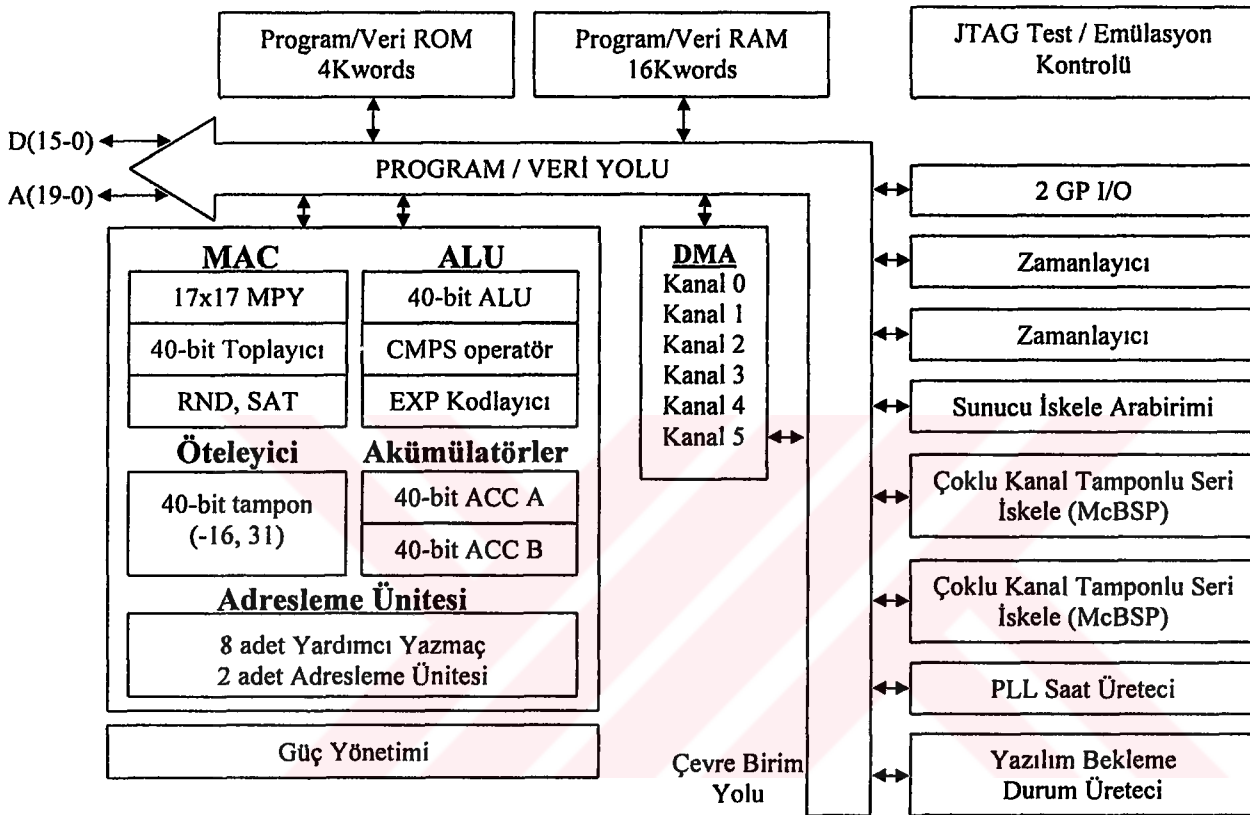
Mantis, işlemci tarafından normalize edildiği için kayan noktalı bir sayının hassasiyeti bir program boyunca sabit kalır. Fakat sabit noktalı verinin doğruluğu, saklanan verinin veya bir işlem sonrasında elde edilen değerin büyüklüğüne göre değişiklik gösterir. Kayan noktalı sayısal işaret işlemcilerde, mantisin 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 gerçekleştirilmesi sorunlarını ortadan kaldırarak kutup veya sıfırların çok daha doğru yerleştirilmesine imkan sağlar.

Kayan noktalı sayısal işaret işlemcilerin çok geniş olan dinamik aralıkları çok küçük veya çok büyük sayıların yüksek hassasiyetle gösterilmesini mümkün kılar. Bu, hızlı fourier dönüşümü (FFT) ve yüksek dereceden öz yinelemeli filtrelerin ara hesaplamalarında özellikle gereklidir. Geniş dinamik aralığın olması, sabit noktalı yapılarda verideki aşağıdan veya yukarıdan taşmayı engellemek için bir ara ölçekleme gereksinimini ortadan kaldırır. Bu yüzden, kayan noktalı bir sayısal işaret işlemcinin kullanımı genellikle yazılımın hacmini ve karmaşıklığını azaltır.

Bahsedilen bu özelliklerin dışında, aşağıda maddeler halinde verilen ve sayısal işaret işlemcilerin karakteristik özelliklerini oluşturan faktörlerde kullanılacak işlemcinin 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 programlanabilen bekleme durumları
- Yonga üzerinde bulunan emülasyon iskeleleri
- Güç yönetimi ve düşük güç harcaması, şeklinde tanımlanabilir.



Şekil 4.1 TMS320C5402 Sayısal İşaret İşlemci Blok Diyagramı

4.3 TMS320C5402 SABİT 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 gerçekleştirildiği 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 işlemcisinin 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 kullanılmak suretiyle bellek kapasitesi 192Kx16bit genişletilebilmektedir.
- Genişletilmiş program belleği desteği sayesinde 8192x16Kbit' e kadar uzatılmış program belleği adresleme imkanı sağlanabilmektedir.
- 40-bit aritmetik mantık birimi (ALU), 2 adet 40 bit toplayıcı (ACC) ve 40-bit toplayıcı ön belleğinin (ACCB) yanı sıra, 16 bit işaretli çarpmaya 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 bekleme durum üretici bulunmaktadır.
- Sağa 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 senkron seri iletişim arabirimi 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) Emülasyon ve test maçı JTAG sınır tarama (IEEE 1149.1)

- (v) Yonga üzerinde tarama bazlı emülasyon 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 işlemci temelli DSP başlangıç kiti (DSP Starter Kit – DSK) tanıtılmaktadır. Çalışma kiti, iki ana parçadan oluşmaktadır.

- Yazılan yazılımların çalıştırıldığı ve üzerinde TMS320C5402 sayısal işaret işlemcisinin bulunduğu donanım kartı.
- Yazılım tasarımının yapıldığı C programlama dili temelli “Code Composer Studio” yazılım geliştirme programı.

4.4.1 DSP Başlangıç Kiti Donanım Kartı

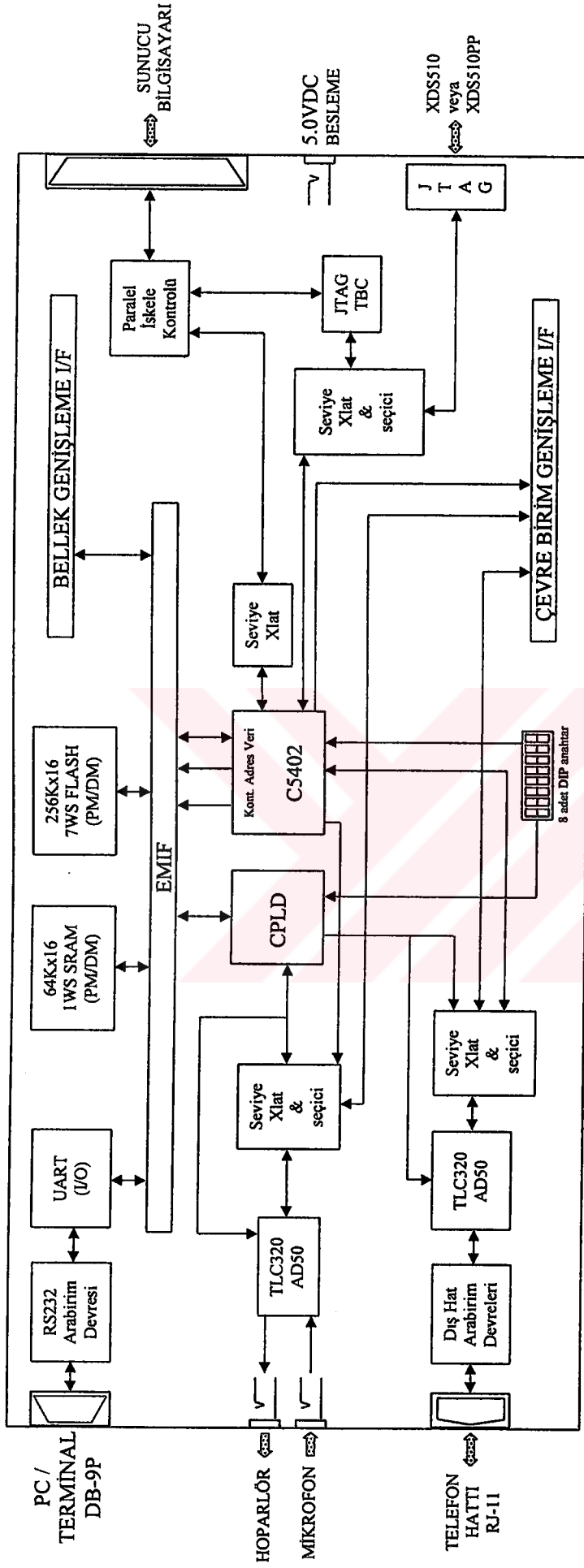
DSP başlangıç kiti donanım kartı, çevirici (C) kodunu gerçek hızında test etmek için üzerinde bir TMS320C5402 işlemcisi barındırır.

Şekil 4.2’ de DSP başlangıç kiti donanım kartının blok diyagramı verilmiştir.

DSP başlangıç kiti donanım kartı, TMS320C5402 sayısal işaret işlemcisinin en basit uygulama devrelerinden birisidir. Devrede bulunan 64x16Kbit RAM bir çok uygulama için yeterli olmaktadır.

Çekirdek program 256x16Kbit' lik FLASH ROM' a yazılmış durumdadır. Devre üzerinde bulunan TLC320AD50 yongası 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 süzme imkanlarına sahiptir.

Yazılan yazılımlar bu donanım sayesinde sınanmaktadır.



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 eşzamansız seri arabirim
- Ek kart takarak sistem genişletme imkanı
- Telefon hat arabirimi
- Paralel iskele ile kontrol

4.4.2 “Code Composer Studio” Yazılım Geliştirme Programı

DSP başlangıç kiti donanım kartı, seri ve paralel iskeleler üzerinden kişisel bir bilgisayara bağlanabilmekte ve kullanılan pencere bazlı hata ayıklama yazılımı sayesinde yazılım geliştirme işlemi oldukça kolaylaşmaktadır. Yüksek seviyeli programlama dili olan C ile yazılan program, çapraz derleyiciden geçirildikten sonra elde edilen kod programcı tarafından minimize edilir. Böylelikle kaynakları daha iyi kullanan, etkin bir kaynak kodu elde edilmiş olur. Bu kaynak kodu, makine koduna dönüştürülür. Bu sayede, hedef donanımda çalıştırılabilecek ve hata ayıklayıcı ile akışı izlenebilecek yürütülebilir bir dosya oluşturulmuş olunur.

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

Kaynakları daha iyi kullanan, etkin bir yazılım elde edebilmek için yapılacak tasarımlarda dikkat edilmesi gereken kurallar aşağıda belirtilmiştir.

- Bir kořula baęlanmamıř komut ieren tekrarlı fonksiyonlar iin mmkn olduęu kadar genel amalı dngler kullanılmalıdır.
- Derleme iřleminden sonra, sayısal iřaret iřlemcinin RAM blgesinde bulunan veri n bellekleri mmkn olan en az sayıda “veri aktarma” komutu gerektirecek řekilde yeniden dzenlenmelidir.
- TMS320C5402 kodunda, beklemeyi en aza indirmek amacıyla mmkn olduęunca “NOP” komutu kullanılmamalıdır.
- Tamsayı verilerin blok halinde bir yerden dięerine aktarılması iin en etkin (en az komut gerektiren) yntem kullanılmalıdır.
- Hesaplama iřlemleri mmkn olduęu kadar program akıřı dıřında yapılmalıdır. Dięer bir anlatımla, iřlem sresinden kazanabilmek iin hesaplama yerine, bellek ierisinde bulunan referans tablolar kullanılmalıdır.
- Program ve veri n bellekleri, birbiriyle uyıřmayan en az sayıda bekletme oluřturulacak řekilde farklı bellek blgelerine yerleřtirilmelidir. Bylelikle her bekletme durumu iin yaklaşık 5 μ s'n' lik ilave bir iřlem sresinin ortaya ıkması engellenebilir.

5. V.32 MODÜLATÖR TASARIMI

5.1 GİRİŞ

V.32 standardına uygun modülatöre sahip modemler; telefon şebekesi veya kiralık devreler üzerinde, 9600 bit/saniye veri iletişim hızında, çift yönlü olarak kullanılırlar. V.32 standardının özellikleri, Bölüm 3' de ayrıntılı olarak incelenmiştir. Seri verinin modülatör katına geliş hızı 9600 bit/saniyedir. Kodlama işlemi, ardışık gelen her dört bit için yapıldığından modülasyon hızı 2400 sembol/saniye (baud) olmaktadır. Taşıyıcı frekansı 1800Hz olarak tanımlıdır ve kılavuz ton bu standart için kullanılmaz. Ayrıca, yankı giderme yapıları bu uygulama için ele alınmamıştır.

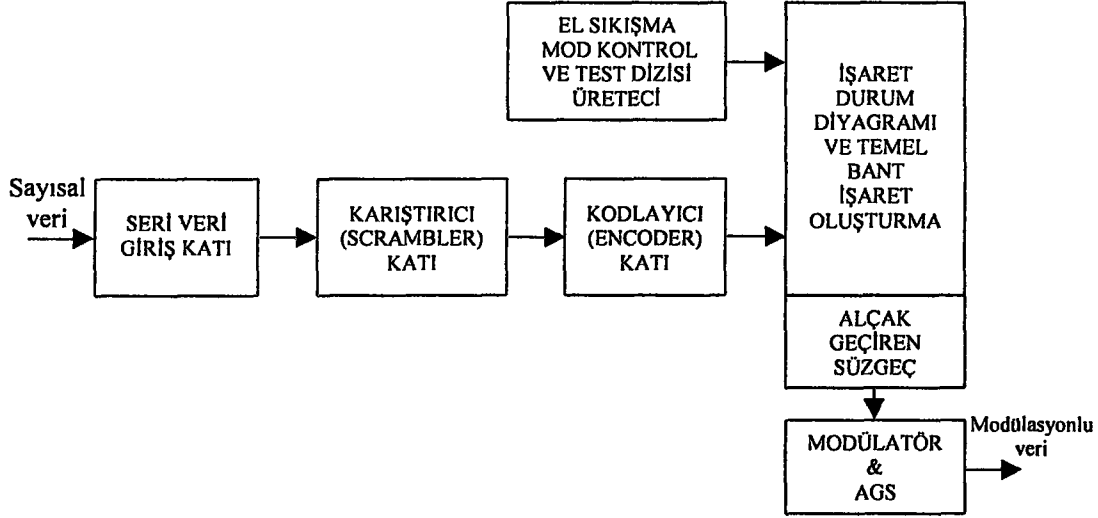
9600 bit/saniye hızında Normal kodlama ve Kafes kodlama olmak üzere iki değişik kodlama yapısı tanımlanmıştır. Normal kodlamada, gönderilecek karıştırılmış veri katarı farksal kodlamaya tabi tutulur.

Kafes kodlamada; gönderilecek karıştırılmış veri katarı, 4 ardışık veri bitinden oluşan gruplara bölünür. Her grup içinde bulunan ilk iki bit, farksal olarak kodlanır. Farksal kodlama çıkışları, katlamalı kodlamanın 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 işareti üretilir. Alçak geçiren süzgeç ve kompleks modülatör ile de modülasyon işlemi gerçekleştirilir.

Tasarım işlemi, bahsedilen V.32 Modülatör özelliklerini gerçekleştirmek için TMS320C5402 sayısal işaret işlemcisi kullanılarak yapılmış olup; kafes kodlama yapısı temel alınmıştır.

5.2 V.32 MODÜLATÖR YAPISINA GENEL BAKIŞ



Şekil 5.1 V.32 Modülatör Blok Diyagramı

Modüle edilerek iletimi gerçekleştirilecek seri veri, modeme bağlı bulunan terminalden V.24 arabirimi aracılığı ile gelir. Ayrıca modem, el sıkışma fazında veya belirlenmiş değişik test işlemlerini gerçekleştirmek için bağlantı fazında çeşitli bit dizileri üretebilir. V.24 arabirimi aracılığı ile bağlı bulunan terminalden gelen seri veri, V.32 standardında tanımlanmış karıştırıcı çok terimlisine bölünür. Bu bölme işlemindeki bölüm katsayısı azalan sırayla alınarak karıştırılmış veri oluşturulur. Arayan ve aranan olmak üzere her iletişim yönü farklı karıştırıcı çok terimlisi kullanır.

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.

Karıştırılmış seri veri katarı, kafes kodlama yapısına (TCM) uygun olarak dört bitlik gruplar halinde sırası ile farksal ve katlamalı 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 kosinüs süzgecinin, x ve y değerleri ile katlama işlemine tabi tutulmasıyla gerçekleştirilmiştir.

İşlem sonucunda ortaya çıkan x(t) ve y(t) temel bant işareti, kompleks modülatör yardımı ile taşıyıcı frekansa ötelenir. Kompleks modülatör kullanılma sebebi işaret diyagramına göre oluşturulmuş temel bant işaretinin kompleks olmasındandır.

Katlama ve modülasyon işlemi, sayısal ortamda örnekleme frekansında (14400Hz) yapıldığı için modüle 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 çalışmaları yazılmış 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 katsayısı karıştırıcı katının çıkışını oluşturur. V.32 standardında, 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 çıkışındaki veri katarı, (5.1) ve (5.2) ifadeleri ile gösterilebilir.

$$\text{Arayan modem ise; } D_S = D_I \oplus D_I \cdot x^{-18} \oplus D_I \cdot x^{-23} \quad (5.1)$$

$$\text{Aranan modem ise; } D_S = D_I \oplus D_I \cdot x^{-5} \oplus D_I \cdot x^{-23} \quad (5.2)$$

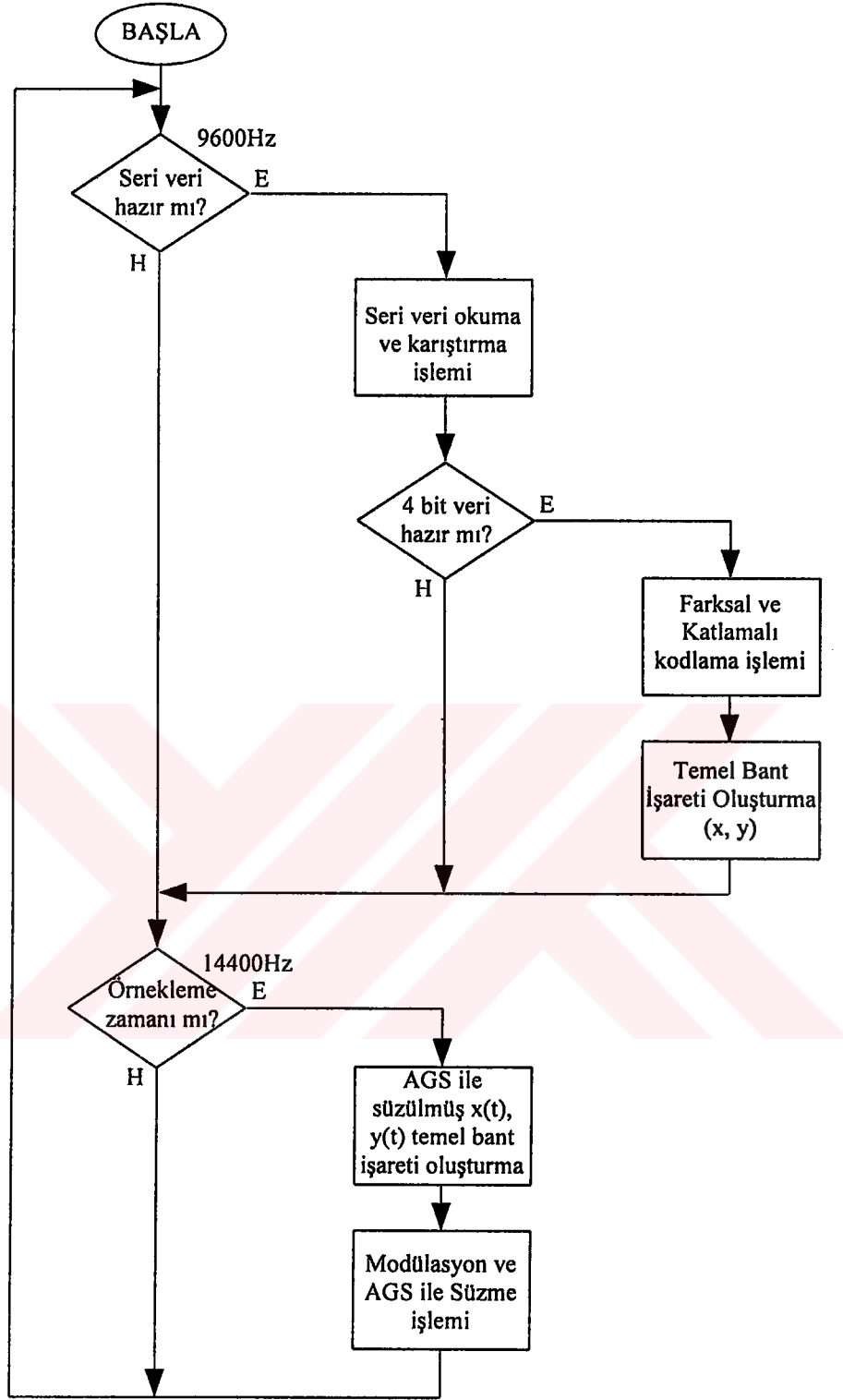
(5.1) ve (5.2) eşitliklerinde;

D_S , karıştırıcı çıkışındaki veri katarı,

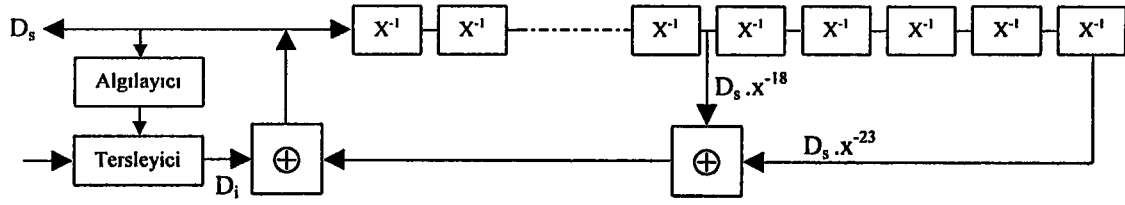
D_I , karıştırıcı girişindeki veri katarı,

$\oplus$, EXOR işlemi,

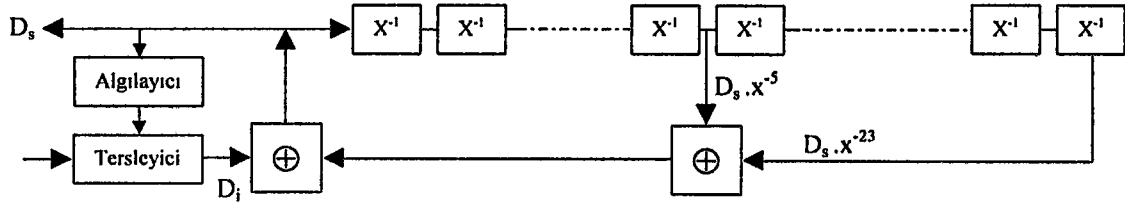
$\cdot$, ikili çarpma işlemini göstermektedir.



Şekil 5.2 Modülatör Programı Genel Akış Şeması



(a) GPC çok terimli kullanılarak karıştırıcı katı blok diyagramı



(b) GPA çok terimli kullanılarak karıştırıcı katı blok diyagramı

Şekil 5.3 Karıştırıcı katı blok diyagramları

Şekil 5.3a ve Şekil 5.3b’ de karıştırma işleminin yapısı görülmektedir. Karıştırıcı çıkışındaki veri katarının (D_s) ardışık olarak 64 bit iletiminden sonra bir sonraki karıştırıcı girişindeki veri (D_i) döndürülür. Bu işlem tüm karıştırıcı uygulamalarında kullanılmalıdır. El sıkışma durumlarında ise karıştırıcı kullanılmaz.

5.3.2 Karıştırıcı Tasarımı

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

Karıştırıcı katı giriş verileri, `transmit_data_table`’ da, çıkış verileri ise `scrambled_data_table`’ da tutulur. Her iki tablo uzunluğu 32 bit olarak belirlenmiştir. Kullanılan çok terimlilerin 23. dereceden olmaları nedeniyle karıştırıcı katında 23 gecikme gözü kullanılır ve gecikme değerleri `scramble_log_table`’ da tutulur. Başlangıçta tüm `scramble_log_table` değerleri “0” olarak tanımlanmıştır. Gönderilecek D_i verisi, karıştırıcı çok terimlisine (Modemin arayan ve aranan olmasına göre kullanılan çok terimli değişmektedir.) göre ilgili gecikme gözlerindeki değerler ile EXOR işlemine tabi tutulur. İşlem sonucu (D_s), `scrambled_data_table`’ a yazılır ve yeni gecikme değerleri `scramble_log_table`’ a yerleştirilir. Oluşturulan karıştırılmış veri (D_s) katarının, ardışık olarak 64 bit iletimi sonrası bir sonraki giriş verisi (D_i) döndürülerek karıştırma işlemine başlanır.

Tablo 5.1 V.32 Karıştırıcı Programı

```

/*****
/* V.32 Karıştırıcı Altprogram Tanımlamaları (Scramble_V32 subrutin defination) */
/*****
#define TRANSMIT_DATA_SIZE          32
#define SCRAMBLE_LOG_SIZE           23 /* V.32 GPC & GPA polinom derecesi..*/
#define SCRAMBLED_DATA_SIZE         32
#define MODEM_SCR_MODE              0
s16 scramble_log_table[SCRAMBLE_LOG_SIZE];
s16 scrambled_data_table[SCRAMBLED_DATA_SIZE];
/*****
/* Scrambler_V32(s16 scr_mode,s16 scr_tr_data) */
/* Gonderilecek veri Org ve Ans durumuna gore scramble edilir. */
/* scr_mode 0 ise GPC 1 ise GPA polinomu kullanılır. */
/* scr_tr_data scramble edilecek veri, geri donus degeri scramble edilmiş veridir. */
/*****
static s16 Scrambler_V32( s16 scr_mode, s16 scr_tr_data )
{
    s16 scr_i,scr_scrmld_data;
    /* V.32 GPC ve GPA polinomu uyarınca karistirilmis veri olusturma*/
    if (scr_mode==0)
        scr_scrmld_data = scr_tr_data ^ ( scramble_log_table[17] ^ scramble_log_table[22] );
    else
        scr_scrmld_data = scr_tr_data ^ ( scramble_log_table[4] ^ scramble_log_table[22] );
    /* scrambler log buffer shift*/
    for (scr_i=22 ; scr_i>=1 ; scr_i--)
        scramble_log_table[scr_i] = scramble_log_table[scr_i-1];
    scramble_log_table[0] = scr_scrmld_data;
    return (scr_scrmld_data);
}
/*****/

```

5.4 KODLAYICI KATI VE TEMEL BANT İŞARETİ OLUŞTURMA

5.4.1 Kodlayıcı ve Temel Bant İşareti Oluşturma Yapısı

Günümüz haberleşme sistemlerinde hata kontrollü kodlama gittikçe önem kazanmaktadır. Hata kontrollü kodlama, veri haberleşmesinde oluşan hataları algılamayı ve düzeltmeyi sağlamak için ek veriler içerir. Bu tip kodlama teknikleri özellikle telefon hatları veya kiralık devreler üzerinden yapılan haberleşmelerde kullanışlıdır. Gönderilen verilerde bulunan ek verilerin ve yazılım tekniklerini kullanarak verilerin alınmasının iletim gücünü arttırmadan bit hata oranının azaltılmasında önemli derecede etkisi vardır.

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

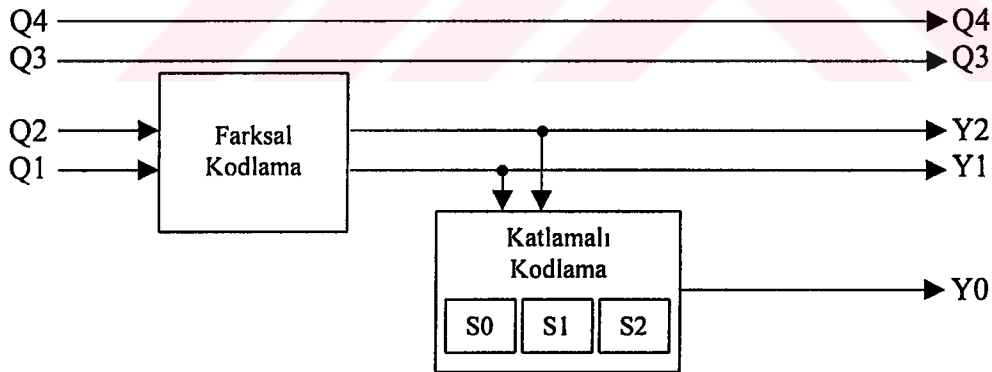
oluşur. Her iki modülasyon tekniğinde de taşıyıcı işaret, birbirinden bağımsız bu iki bilgi kanalı (x ve y) tarafından modüle edilir.

Kafes kodlamada giriş verileri dört bitlik gruplara ayrılır. Her grupta bulunan ilk iki bit, önce farksal kodlamaya sonra bu kodlayıcı çıkışları katlamalı kodlamaya tabi tutularak ek bit oluşturulur. Diğer iki bit, hiç bir kodlamaya tabi tutulmaz. Böylece çıkış veri grubu beş bitten oluşur ve her grup 32 – nokta işaret diyagramında bir noktaya karşılık düşer (Şekil 3.3). Dik genlik modülasyonlu kodlamada ise katlamalı kodlama kullanılmaz. Dolayısı ile, çıkış veri grubu ek bit oluşturulmadığı için dört bitten meydana gelir ve 16 – nokta işaret diyagramında bir noktaya karşılık düşer (Şekil 3.1).

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), farksal ve katlamalı kodlama işlemine tabi tutularak beş 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 içermesidir (Şekil 5.4).



Şekil 5.4 V.32 Kafes Kodlayıcı Blok Diyagramı

V.32 standardı, kodlama işleminde dik genlik modülasyon tekniklerinin kullanılmasını da önermektedir. Bundan dolayı; kodlayıcı katında hata düzeltme bilgisi içermeyen, sadece farksal kodlama yapılan ve Şekil 3.1’ de gösterilen 16 – nokta işaret diyagramı da kullanılır. Eğer katlamalı kodlama kullanılırsa, kodlama çıkışı beş bitlik bilgi içerir ve Şekil 3.3’ de gösterilen 32 – nokta işaret diyagramı kullanılır. Genelde aynı ortalama güç için; 32 – nokta işaret diyagramı kullanan modülasyon şekli, 16 – nokta işaret diyagram yapısına göre daha yüksek bit – hata

oranına (BER) sahiptir. Bunun nedeni; 32 – nokta işaret yapısındaki noktalar arası en küçük uzaklığın diğer yapıya göre az olmasıdır. Bununla birlikte; katlamalı kodlama, giriş verilerinin çıkış sembollerine dönüştürülmesinde bir takım kısıtlamalar getirmektedir (Bakınız : “2.6 Kafes Kodlamalı Modülasyon”). Özellikle, ardışık gelen iki çıkış veri grubunun birbirlerine göre komşu sekizli grup içinde olmalarına izin verilmez. Böylece ardışık gelen iki çıkış veri grubunun tanımladığı noktalar arasındaki uzaklık arttırılmış olur. Böylece, kodlama kazancı dik genlik modülasyonuna göre iki kat artar.

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).

$$Y1_n = Q1_n \oplus Y1_{n-1} \quad (5.3)$$

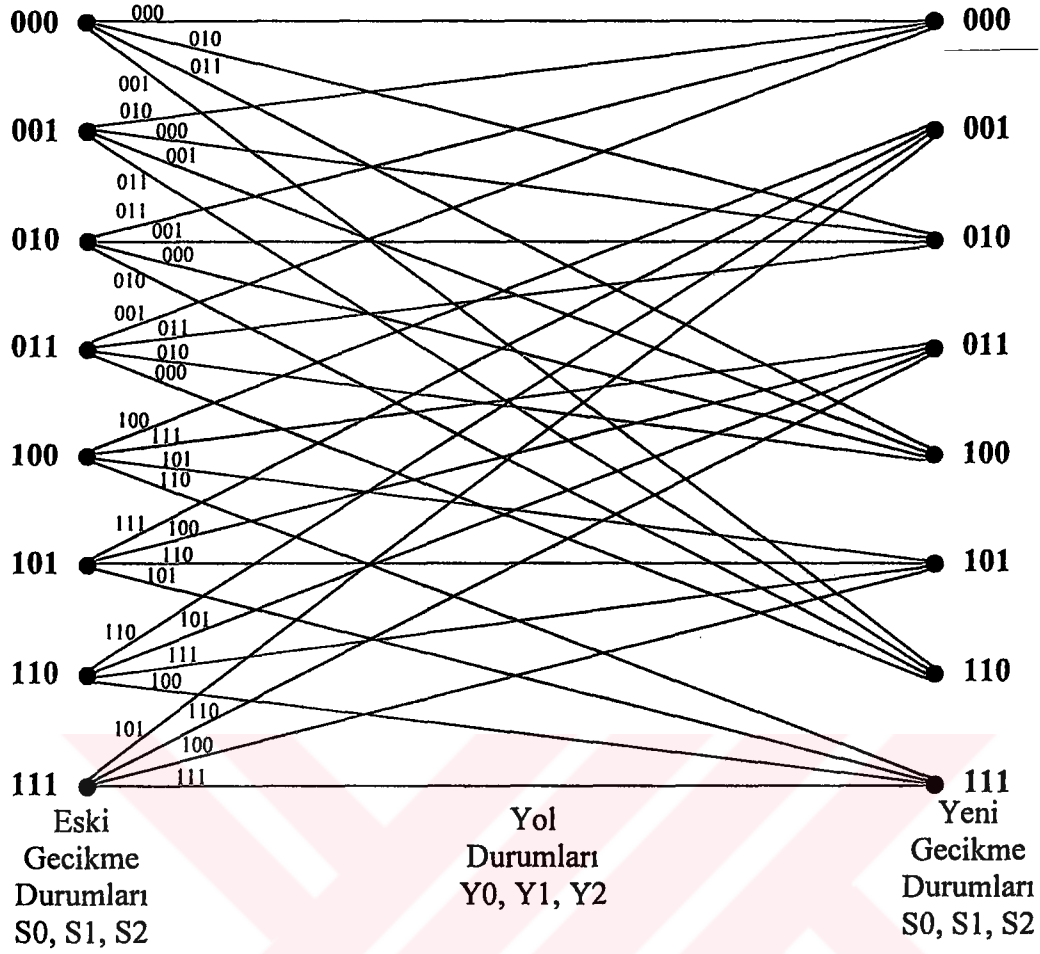
$$Y2_n = (Q1_n \cdot Y1_{n-1}) \oplus Y2_{n-1} \oplus Q2_n \quad (5.4)$$

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

Katlamalı kodlayıcı çıkışında oluşturulan 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 gecikme gözleri bulunmaktadır. Bu gecikme gözleri 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.

Sonlu durum diyagramından da görüleceği gibi katlamalı kodlayıcıda tanımlı olan gecikme gözlerinin sahip olduğu durumdan, olası tüm durumlara geçiş bulunmamaktadır. Kodlayıcı olası sekiz durumdan birinde bulunabilir ve bir sonraki anda bu sekiz durumdan ancak dördüne geçebilir. Bu kısıtlama, katlamalı kodlayıcı tarafından empoze edilmektedir.

Örneğin; (0.0.1) gecikme 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 çıkışındaki 5 bitlik kodlanmış verinin; 32 – nokta işaret diyagramında gösterdiği noktaya ait x ve y temel bant işaret bileşenleri, süzme ve modülasyon işlemlerinin gerçekleştirildiği 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) denklemleri uyarınca farksal kodlama işlemine tabi tutularak y1 ve y2 çıkış verileri oluşturulur. Bu işlem sırasında `diff_encoder_log_table` dizisinde tutulan gecikme değerleri kullanılır.

Katlamalı kodlama işlemi, farksal 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 yazılmış programın sadece 000 durumundan y1 ve y2 verilerine göre diğer durumlara geçişi verilen program parçasında (Tablo 5.2) gösterilmiştir. Bu yazılımda, matris tablo yapısı yerine durum geçiş yapısı kullanılmıştır. Böylece kodlayıcı program çıkış veri grubu beş bittten oluşur ve `signal_element_mapping_input` dizisinde saklanır. Kodlayıcı çıkış verisinin, giriş verisinden daha uzun olmasının sebebi çıkış verisinin hata düzeltme bilgisini içermesidir. Tablo 5.2' de kodlayıcı programının listesi verilmiştir.

Tablo 5.2 V.32 Kodlayıcı Programı

```
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 konvolusyonel kodlama yapılarak */
/* y0, y1, y2, q3 ve q4 cikis degerleri elde edilir ve bu degerler */
/* "signal_element_mapping_input" dizisine sırayla oturtulur. */
/* signal_element_mapping_input[0] = y0, signal_element_mapping_input[1] = y1 */
/* signal_element_mapping_input[2] = y2, signal_element_mapping_input[3] = q3 */
/* signal_element_mapping_input[4] = q4 */
/* conv_encoder_log_s_table[3] ==> s0, s1, s2 */
*****/
static void Encoder_V32( 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 kullaniliyor */
    signal_element_mapping_input[4] = q4; /* q4 direkt kullaniliyor */
    signal_element_mapping_input[1] = q1 ^ diff_encoder_log_table[0]; /* y1 olusturma */
    signal_element_mapping_input[2] = ( q1 & diff_encoder_log_table[0] ) /* y2 olusturma */
    ^ ( 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) )
    {
        conv_encoder_log_s_table_new[2] = 0;
        signal_element_mapping_input[0] = 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; }
        if ( ( signal_element_mapping_input[1] == 1 ) && ( signal_element_mapping_input[2] == 1 ) )
```

```

    { conv_encoder_log_s_table_new[0] = 1;
      conv_encoder_log_s_table_new[1] = 0; }
    if (( signal_element_mapping_input[1] == 0 ) && ( signal_element_mapping_input[2] == 1 ))
    { conv_encoder_log_s_table_new[0] = 1;
      conv_encoder_log_s_table_new[1] = 1; }
  }
  /* y1 ve y2 nin bir sonraki katar icin degerlerinin saklanmasi islemi*/
  diff_encoder_log_table[0] = signal_element_mapping_input[1];
  diff_encoder_log_table[1] = signal_element_mapping_input[2];
  conv_encoder_log_s_table[0] = conv_encoder_log_s_table_new[0];
  conv_encoder_log_s_table[1] = conv_encoder_log_s_table_new[1];
  conv_encoder_log_s_table[2] = conv_encoder_log_s_table_new[2];
}

```

5.4.3 Temel Bant İşaret Oluşturma Programı

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

Tablo 5.3 Temel Bant İşaret Oluşturma Programı

```

const s16 Trellis_X_table[32] = {
    -4,0,0,4,4,0,0,-4,-2,-2,2,2,2,-2,-2,-3,1,-3,1,3,-1,3,-1,1,-3,1,1,-1,3,-1,-1};
const s16 Trellis_Y_table[32] = {
    1,-3,1,1,-1,3,-1,-1,3,-1,3,-1,-3,1,-2,-2,2,2,2,-2,-2,4,0,0,-4,-4,0,0,4};
#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 biter. */
/* Onun yerine bir başka X ve Y gönderilmeye başlanır. */
/*****/
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

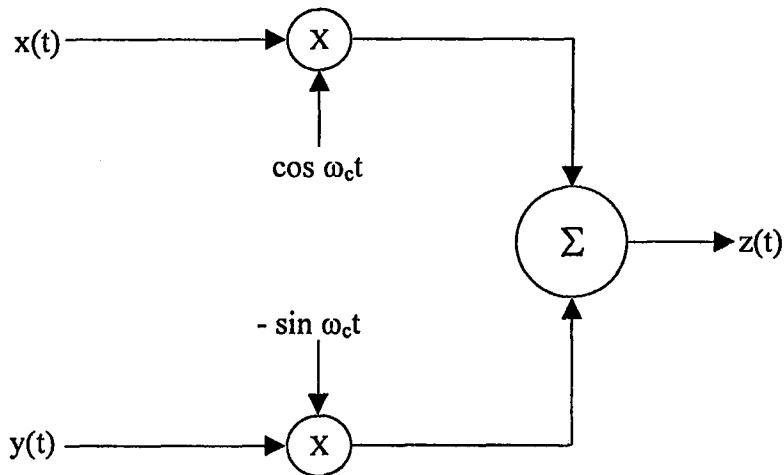
5.5.1 Süzme ve Modülasyon Özellikleri

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

Kodlayıcı çıkışında oluşturulan beş bitlik çıkış verisinin değişimi, x ve y ' nin değişimine sebep olur. Böylece x ve y değerlerinde süreksizlikler oluşmaktadır. Bu da sonsuz bant genişliğine sahip temel bant işaretinin oluşmasına yol açar.

Temel bant işareti, kompleks modülasyon işlemi öncesinde frekans bant genişliğini sınırlamak için alçak geçiren süzgeçten geçirilir. Alçak geçiren süzgeç olarak yükseltilmiş kosinüs süzgeci kullanılacaktır. Bu süzgecin kullanılması ile her sembol iletim aralığındaki işaret değişimlerinde, alçak geçiren süzgeç özelliğine sahip haberleşme kanallarının oluşturduğu bozulma en aza indirilir.

Kompleks modülatör, yükseltilmiş kosinüs süzgeç çıkışından gelen süzölmüş temel bant işaretlerini taşıyıcı frekansına öteler. Kullanılan kompleks modülatör Şekil 5.6' da gösterilmiştir. Modülatörde gösterilen $x(t)$ ve $y(t)$ girişleri alçak geçiren süzgeç çıkışlarıdır. Modülatör, taşıyıcı frekansına (f_c) merkezli toplam ve fark frekansları oluşturarak temel bant işaretini öteler.



Şekil 5.6 V.32 Kompleks Modülatör

Modülasyon işlemi matematiksel olarak incelenir ve $x(t)$ ile $y(t)$ işaretleri tek bir frekans bileşeninden ibaret kabul edilirse (5.5);

$$x(t) = y(t) = \sin \omega_b t \quad (5.5)$$

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

$$z(t) = \sin \omega_b t \cdot \cos \omega_c t - \sin \omega_b t \cdot \sin \omega_c t \quad (5.6)$$

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

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

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

$$\begin{aligned} z(t) &= \frac{1}{2} \cdot [\sin (\omega_c + \omega_b) \cdot t - \sin (\omega_c - \omega_b) \cdot t] - \frac{1}{2} \cdot [\cos (\omega_c - \omega_b) \cdot t - \cos (\omega_c + \omega_b) \cdot t] \\ &= \frac{1}{2} \cdot \sin (\omega_c + \omega_b) \cdot t - \frac{1}{2} \cdot \sin (\omega_c - \omega_b) \cdot t - \frac{1}{2} \cdot \cos (\omega_c - \omega_b) \cdot t + \frac{1}{2} \cdot \cos (\omega_c + \omega_b) \cdot t \\ &= \frac{1}{2} \cdot \sin (\omega_c + \omega_b) \cdot t - \frac{1}{2} \cdot \sin (\omega_c - \omega_b) \cdot t - \frac{1}{2} \cdot \cos (\omega_c - \omega_b) \cdot t + \frac{1}{2} \cdot \cos (\omega_c + \omega_b) \cdot t \end{aligned} \quad (5.8)$$

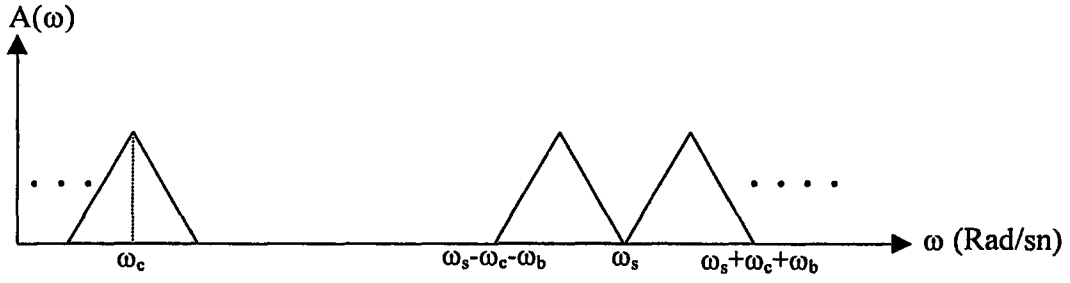
(5.8) ifadesinden de görüleceği gibi $\omega_c + \omega_b$ ve $\omega_c - \omega_b$ fark frekanslarının sinüs fonksiyonlarını içermektedir. Tüm modülasyon işlemleri sayısal olarak yapılacağı için örnekleme frekansı kullanılacaktır. Bundan dolayı, örneklenmiş frekansın etrafında tekrarlanmış spektrumlar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



(a) Temel Bant İşaret Frekans Spektrumu



(b) Taşıyıcı Frekansa Ötellenmiş İşaret Frekans Spektrumu



(c) Örneklenmiş İşaret Frekans Spektrumu

ω_b : temel bant açısai frekansı
 ω_c : taşıyıcı açısai frekansı
 ω_s : örnekleme açısai frekansı

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 gerçekleştirilmiş işaretin frekans spektrumu görülmektedir. Şekil 5.7c' de görülen örnekleme frekansı etrafındaki işaretler ise modülatör çıkışında alçak geçiren süzgeçten geçirilerek atılmalıdır. Şekil 5.7' deki A terimi, genliği ifade etmektedir.

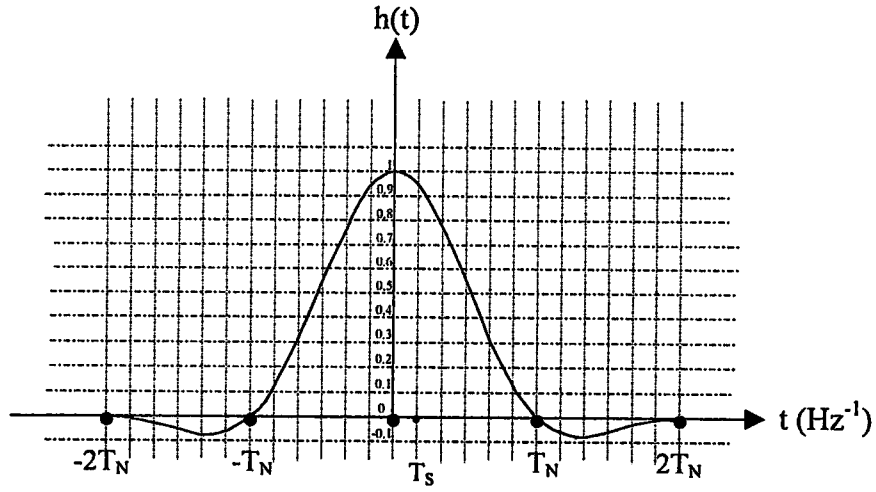
5.5.2 Temel Bant ve Modülasyonlu İşaret Analizi

Modülatör tasarımında, sayısal işaret işlemci kullanılması nedeniyle süzme ve modüle işlemleri sayısal olarak örnekleme frekansında yapılmıştır. V.32 modülatörde iletişim kanalı 600 – 3000Hz' dir. Shannon teoremi uyarınca örnekleme frekansı minimum $2 \cdot 3000 = 6000\text{Hz}$ alınabilir. Kullanılacak örnekleme frekansı 14400Hz olarak seçilmiştir. V.32 modemde, iletilen işaret hızı 2400 baud, taşıyıcı frekansı ise 1800Hz' dir. Örnekleme frekansının bu değerlerin tam katı olarak seçilmesi modülatör tasarımında kolaylık sağlamaktadır.

Temel bant işaretinin bandının sınırlandırılmasında düşüş parametresi 0.75 olan yükseltilmiş kosinüs süzgeci kullanılmaktadır. Yükseltilmiş kosinüs süzgecinin matematiksel ifadesi (5.9) denkleminde görülmektedir.

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

$f_N = 1/T_N$: 2400 işaret/saniye (baud)
 α : 0.75 (düşüş parametresi)



Şekil 5.8 Yükseltilmiş Kosinüs Süzgeç Zaman Cevabı

Şekil 5.8' de, (5.9) denkleminde matematiksel ifadesi verilen yükseltilmiş kosinüs süzgecinin zaman cevabı görülmektedir. T_N modülasyon periyodunu göstermektedir. Modülasyon periyodu, $T_N = 1/2400\text{Hz}^{-1}$ dir. $T_S = 1/14400\text{Hz}^{-1}$ olup, örnekleme periyodunu gösterir.

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

$$m(t) = A(t) \cdot \cos (2.f_C.t + \varphi(t)) \quad (5.10)$$

f_C : 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 süreksizlik göstermektedir. Bundan dolayı $m(t)$ işaretinde süreksizlikler oluşmaktadır. Bunu engellemek için $m(t)$ işareti alçak geçiren süzgeçten geçirilir. Çıkış işareti $z(t)$, (5.11)' de gösterilmiştir.

$$z(t) = h(t) * m(t) \quad (5.11)$$

(5.10) denkleminde verilen $m(t)$ temel bant işaretindeki kosinüs bileşeninin trigonometrik açılımından (5.12) ifadesi elde edilir.

$$\begin{aligned} A \cdot \cos (\omega_C.t + \varphi) &= A \cdot [\cos \omega_C.t \cdot \cos \varphi - \sin \omega_C.t \cdot \sin \varphi] \\ &= A \cdot \cos \varphi \cdot \cos \omega_C.t - A \cdot \sin \varphi \cdot \sin \omega_C.t \\ &= x \cdot \cos \omega_C.t - y \cdot \sin \omega_C.t \end{aligned} \quad (5.12)$$

Böylece çıkış işareti $z(t)$, (5.13) ifadesindeki gibi yazılır. (5.13) denkleminde de görülebileceği gibi, 32 – nokta işaret diyagramındaki x ve y değerleri kompleks modülatörden geçirilerek modülasyonlu işaret oluşturulur.

$$z(t) = x(t) \cdot \cos \omega_c t - y(t) \cdot \sin \omega_c t \quad (5.13)$$

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

$$x(t) = x * h(t) \quad (5.14a)$$

$$y(t) = y * h(t) \quad (5.14b)$$

Katlama işlemi yapılmış ve kompleks modülatörden geçirilmiş modülasyonlu işaret (5.15)' de görülmektedir.

$$z(t) = (x * h(t)) \cdot \cos \omega_c t - (y * h(t)) \cdot \sin \omega_c t \quad (5.15)$$

5.5.3 Süzgeç ve Modülatör Program Tasarımı

5.5.3.1 Program Tasarım Yöntemi

Modülasyonlu işaret denkleminde (5.15) görülen x ve y ; 0, ± 1 , ± 2 , ± 3 ve ± 4 değerlerini alabilir (Şekil 3.3). x ve y değerleri; sembol iletim aralığı boyunca değişmediği ve sabit bir tam sayı olarak kaldığı için, yükseltilmiş kosinüs süzgeci ile x ve y değerleri arasında yapılan katlama işlemi de basit çarpma işlemine dönüşmektedir.

Şekil 5.8' de görülen yükseltilmiş kosinüs süzgecinin zaman cevabı, sonsuz uzunluktadır. Teorik olarak, x ve y değerlerinin sonsuz uzunlukta kuyruklara sahip bu süzgeç ile katlama işlemine tabi tutulması gerekir.

Yükseltilmiş kosinüs süzgeç zaman cevabının karakteristiğinden dolayı sonsuza giderken kuyruklar küçülmemekte ve süzgeç işleminde önemini kaybetmemektedir. Bundan dolayı, modülatör tasarımında yükseltilmiş kosinüs süzgeci $[-2T_N, 2T_N]$ aralığında kullanılmıştır. Bu aralıktaki yükseltilmiş kosinüs süzgecinin T_s periyodu ile örneklenmiş değerleri, tablo halinde program tarafından kullanılır.

Yükseltilmiş kosinüs süzgecinin $[-2T_N, 2T_N]$ aralığında T_s periyodu ile örneklenmiş değerleri Tablo 5.4' de gösterilmiştir. Tablo değerleri, Şekil 5.8' deki örnek değerleri göstermektedir.

Tablo 5.4 Örneklenmiş Yükseltilmiş Kosinüs Süzgeç Değerleri

		Örnek No					
		1	2	3	4	5	6
Sembol İletim Aralığı	$[-2T_N, -T_N]$	0	-0.0878	-0.1660	-0.2128	-0.2070	-0.1347
	$[-T_N, 0]$	0	0.1914	0.4140	0.6367	0.8261	0.9531
	$[0, T_N]$	0.9980	0.9531	0.8261	0.6367	0.4140	0.1914
	$[T_N, 2T_N]$	0	-0.1347	-0.2070	-0.2128	-0.1660	-0.0878

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

- m . x değerinin genliğini modüle ettiği süzgecin $[-2T_N, -T_N]$,
- $m-1$. x değerinin genliğini modüle ettiği süzgecin $[-T_N, 0]$,
- $m-2$. x değerinin genliğini modüle ettiği süzgecin $[0, T_N]$,
- $m-3$. x değerinin genliğini modüle ettiği süzgecin $[T_N, 2T_N]$ aralığındaki değerlerinin toplamından oluşacaktır.

Aynı işlemler $y(t)$ temel bant işareti içinde geçerlidir.

Bu şekilde oluşturulan $x(t)$ ve $y(t)$ değerleri, her periyot için 8 örnek (T_s/T_C) alınarak oluşturulmuş sinüs ve kosinüs tabloları ile modüle işlemine tabi tutulur (5.15). Kullanılan sinüs ve kosinüs tabloları Tablo 5.5' de gösterilmiştir.

Tablo 5.5 Örneklenmiş Sinüs ve Kosinüs İşaret Değerleri

	1	2	3	4	5	6	7	8
Sinüs işareti	0	0.707	1	0.707	0	-0.707	-1	-0.707
Kosinüs işareti	1	0.707	0	-0.707	-1	-0.707	0	0.707

5.5.3.2 Temel Bant İşaret Süzgeç ve Modülasyon Programı

Programlar, örnekleme 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 modülasyon programı tarafından kullanılır.

Tablo 5.6' da Temel Bant İşaret Süzme Programı, Tablo 5.7' de ise Modülasyon programı gösterilmiştir.

Tablo 5.6 Temel Bant İşaret Süzme Programı

```

/*****
/* 14400 Hz hızında çalışan bir altprogramdır. İlgili baud'ta 4 adet XY değerinin- */
/* raised-cos fonk. ile 4 değişik gözdeki çarpım değerlerinin toplamı alınır. */
/* TemelBand_LocateX ve TemelBand_LocateY deki degerler raisecos ile ayırık olarak */
/* konvole edilir. */
/*****/
static void TemelBand_Create_withLPF( )
{
    s16 TBCwL_i, TBCwL_j=0, TBCwL_k=0;

    for ( TBCwL_i = 0 ; TBCwL_i < RaiseCos_KuyrukSayisi ; 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) eşitliğine göre modülasyon işlemine tabi tutulur. Kullanılan sinüs ve kosinüs tabloları Tablo 5.5' de gösterilmiştir. Mod_zt, (5.13) eşitliğindeki z(t)' yi ifade etmektedir.

Tablo 5.7 Modülasyon Programı

```

/*****
/* Hesaplanan x(t) ve y(t) module edilir. */
/* z(t)= Filtered_xt * cos fc.t - Filtered_yt * sin fc.t */
/* Her baud' da 6 örnek 8 örnekli sin ve cos ile module edildiği için bellekli sayacılar kullanılmıstır.*/
/* 8 örnek için TBCwL_Sayac2 sayacı kullanılır. */
/* 3 tam periyod sonrası periyodiklik olustugu için TBCwL_Sayac3 kullanılır. */
/*****/
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_Signal1[Modulation_i+1] = Modulated_Signal1[Modulation_i]; }
    Modulated_Signal[0] = Mod_zt;
    if (TBCwL_Sayac2 == 7)
        TBCwL_Sayac2=0;
    else
        TBCwL_Sayac2++;
}
/*****/

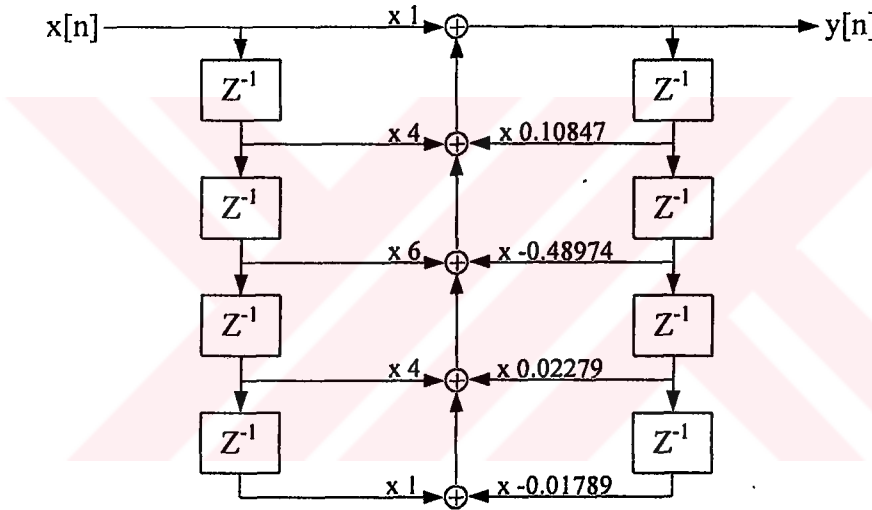
```

5.5.3.3 Modülasyonlu İşaret Süzme Programı

Modülatör tasarımının sayısal ortamda yapılmış olması örnekleme frekansı kullanılmasını gerektirir. Bu nedenden ötürü, modülasyon işlemi sonrası örnekleme frekansın etrafında tekrarlanmış spektrumlar oluşur (Şekil 5.7c).

Modülatör çıkışında semboller alçak geçiren süzgeçten geçirilerek 3500Hz üzerindeki frekans bileşenleri süzülür. Alçak geçiren süzgeç olarak, dördüncü dereceden Butterworth süzgeç kurulmuştur. Alçak geçiren süzgecin çıkış ifadesi (5.16)' da verilmiş ve Şekil 5.9' da gösterilmiştir. Bu ifade kullanılarak tasarlanan süzgece ait alt program Tablo 5.8' de listelenmiştir.

$$y[n] = x[n] + x[n-4] + 4.x[n-3] + 6.x[n-2] + 4.x[n-1] - 0,01789.y[n-4] + 0.02279.y[n-3] - 0.48974.y[n-2] + 0.10847.y[n-1] \quad (5.16)$$



Şekil 5.9 14400Hz örnekleme hızında 4. Dereceden 3500Hz köşe frekanslı alçak geçiren Butterworth süzgeç blok diyagramı

Tablo 5.8 Butterworth Süzgeç Programı

```

/*****
/* modülasyonlu işaretin örnekleme frekansındaki tekrarlarını 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_xv[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] =
( butterworthlpf_xv[0] + butterworthlpf_xv[4] ) + ( 4 * ( butterworthlpf_xv[1] +
butterworthlpf_xv[3] ) ) + ( 6 * butterworthlpf_xv[2] ) +
( -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 bölümde; 9600 bit/saniye veri iletişim hızında, ITU V.32 standardı ile tanımlanan kurallar çerçevesinde çalışan modülatör tasarımı açıklanmıştır.

Yapılan yazılım çalışmalarında, TMS320C5402 sayısal işaret işlemcisine ait program geliştirme kiti 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 örnekleme hızında modülasyonlu sembolleri modulated_sembol_table tablosuna yazar.

6. V.32 DEMODÜLATÖR TASARIMI

6.1 GİRİŞ

İletimi gerçekleştirilecek sayısal veri, Bölüm 5' de açıklanan biçimde modülasyon işlemine tabi tutulur. Modülasyonlu işaret, haberleşme kanalından geçerek alıcı tarafta bulunan demodülatör katına gelir. Demodülatör katının görevi, modülasyonlu işaretten iletimi gerçekleştirilen sayısal veriyi elde etmektir.

Demodülasyon işleminde; taşıyıcı frekansa ötelenmiş giden ve gelen modülasyonlu işaretlerden, alınan modülasyonlu işaret ayrılarak temel bant işareti oluşturulur. Temel bant işaretinin, Viterbi algoritmasına göre 32 – nokta işaret diyagramındaki beş bitlik veri karşılığı belirlenerek ters farksal dönüşüm işlemi yapılır. Oluşturulan dört bitlik veri grupları, çözme işlemine tabi tutulur. Böylece, demodüle edilmiş veri katarı oluşturulmuş olunur.

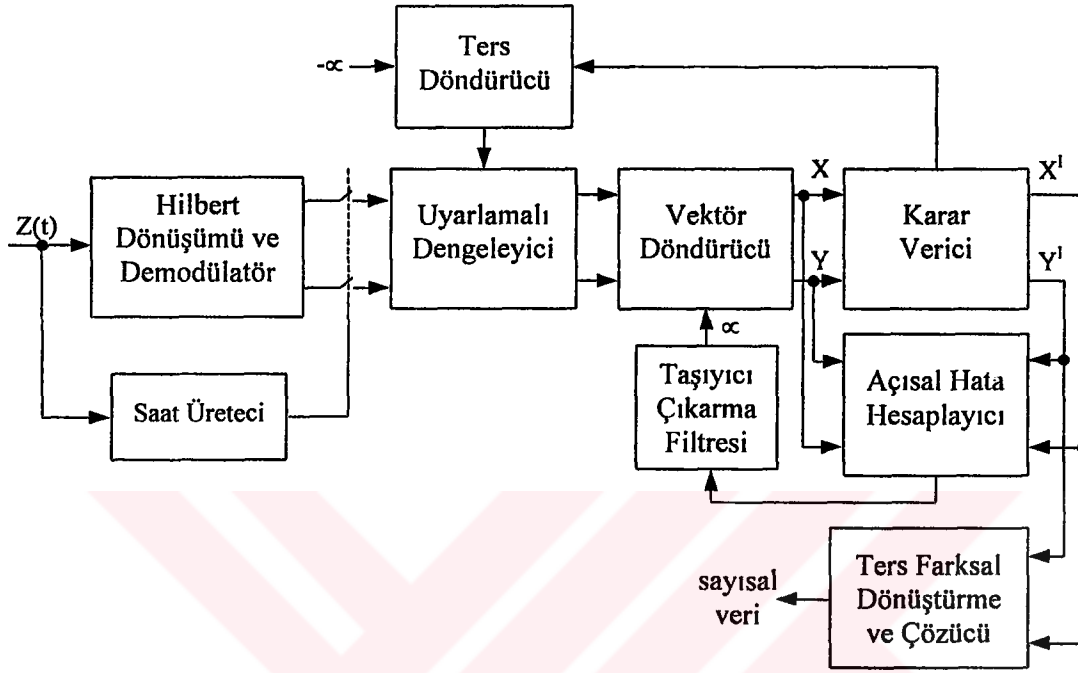
Demodülasyonda, kullanılan yapıların verimli olarak algılama işlemini yapabilmesi için alınan işaretle bulunan gürültülerin alçak geçiren süzgeç ile ayrılması ve küçük genlikli işaretlerin yükseltilmesi gereklidir. Kullanılacak kuvvetlendiricinin kazancı, alınan işaretin ortalama gücüne orantılı olarak değiştirilebilmelidir. Bu şekilde gerçekleştirilen otomatik kazanç kontrolü ile gelen işaretlerin, genlik seviyeleri eşitlenmektedir.

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

6.2 V.32 DEMODÜLATÖR YAPISINA GENEL BAKIŞ

Demodülasyon yapısının şekillenmesindeki ana etken; alınan modülasyonlu işaretin ($Z(t)$), temel banda indirilmesinde kullanılan yöntemdir. Sayısal işaret işleme tekniklerinin kullanılacak olması, tasarımı düşünülen demodülasyon katında Hilbert dönüşümünün kullanılmasını gerektirir. Bu dönüşümün temel prensibi; bir işaretin frekans bileşenlerinin fazını -90° ötelemek ve kendisine dik başka bir işaret

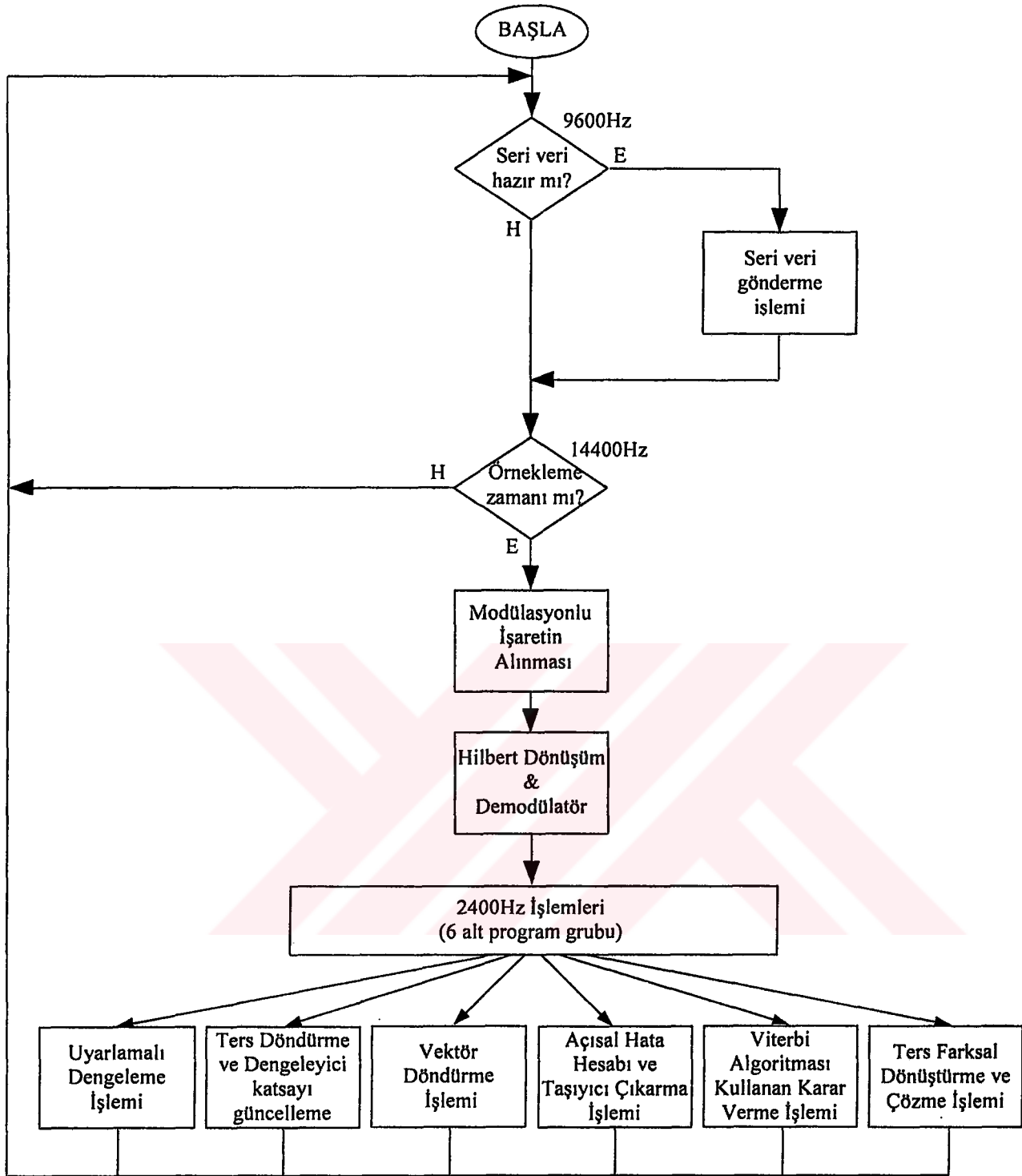
oluşturarak kompleks demodülatör yardımıyla demodülasyonu gerçekleştirmektedir. Bu temel prensipten dolayı, Hilbert dönüşümünün sayısal işaret işleme teknikleri ile gerçekleştirilmesi kolay olmaktadır. Şekil 6.1' de Hilbert dönüşüm tekniğinin kullanıldığı demodülasyon katı blok diyagramı görülmektedir.



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 işareti örneklemede kullanılan frekans ve faz uyumsuzluklarından dolayı oluşan hata, daha önceki vektör hatalarına göre belirlenmiş olan α açısı kadar işareti döndürmekle giderilmeye çalışılır. α , 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 yardımıyla, 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 süzgecine verilir.



Şekil 6.2 Demodülatör Programı Genel Akış Şeması

Alınan vektör ile ideal vektör arasındaki, $|X-X^I|$ ve $|Y-Y^I|$ fark değerleri ise ters dönüştürücü tarafından kullanılır. Ters dönüştürücü, uyarlamalı dengeleyicinin katsayılarını belirlemekte kullanılan parametreleri oluşturur.

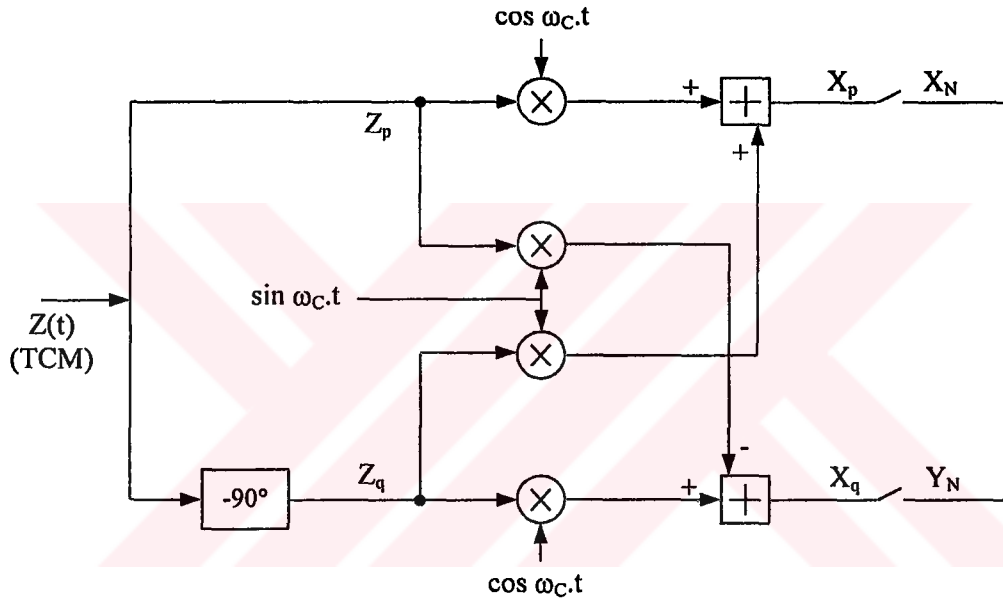
Hilbert dönüşümünün kullanıldığı demodülasyon işlemi sonrası, karar verilen X^I ve Y^I değerlerinin karşılığı olan beş bitlik veriler bulunur. Karar verme işleminde ve dört bitlik faydalı verinin oluşturulmasında Viterbi algoritması kullanılmaktadır.

Karar verici çıkışındaki beş bitlik veri grubu, ters farksal dönüşüme tabi tutularak dört bitlik veri grubu oluşturulur. Bu grup, çözücü çok terimli yardımıyla çözülerek faydalı veri alınmış olunur.

Genel prensipleri yukarıda açıklanan demodülatör katına ait yazılımın akış şeması Şekil 6.2' de verilmiş olup diğer bölümlerde ayrıntılı olarak incelenmiştir.

6.3 HİLBERT DÖNÜŞÜMÜ VE KOMPLEKS DEMODÜLATÖR

6.3.1 Hilbert Dönüşüm Yöntemiyle Demodülasyon



Şekil 6.3 Hilbert Dönüşümü ve Demodülatör Blok Diyagramı

Şekil 6.3'de, Hilbert dönüşümü ile gerçekleştirilen demodülasyon işlemi blok diyagramı görülmektedir.

Modülasyon işlemi ile taşıyıcı frekansa ötelenmiş işaret, bu dönüşüm ile temel banda indirilir. $Z(t)$ işareti, süzölmüş ve otomatik kazanç kontrolünden geçerek alınmış işarettir (6.1). Bu işaretin niteliği, Bölüm 5' de açıklanmıştır.

$$Z(t) = Z_p = x \cdot \cos \omega_c t - y \cdot \sin \omega_c t \quad (6.1)$$

$Z(t)$ işaretinin, -90° faz kaydırılarak Hilbert dönüşümü alınmış ifadesi, (6.2)' de Z_q olarak gösterilmiştir.

$$Z_q = x \cdot \sin \omega_c t + y \cdot \cos \omega_c t \quad (6.2)$$

Alınan işaret taşıyıcısının, demodülasyon işleminde kullanılan yerel taşıyıcı ile faz ve frekansının uyumlu olduğu düşünülürse;

X_p , (6.3)' deki gibi,

$$X_p = (x \cdot \sin \omega_C t + y \cdot \cos \omega_C t) \cdot \sin \omega_C t + (x \cdot \cos \omega_C t - y \cdot \sin \omega_C t) \cdot \cos \omega_C t \quad (6.3a)$$

$$X_p = x \cdot \sin^2 \omega_C t + x \cdot \cos^2 \omega_C t + y \cdot \cos \omega_C t \cdot \sin \omega_C t - y \cdot \cos \omega_C t \cdot \sin \omega_C t \quad (6.3b)$$

$$X_p = x \cdot (\sin^2 \omega_C t + \cos^2 \omega_C t) + y \cdot \cos \omega_C t \cdot \sin \omega_C t - y \cdot \cos \omega_C t \cdot \sin \omega_C t \quad (6.3c)$$

$$X_p = x \quad (6.3d)$$

X_q ise, (6.4)' deki gibi elde edilir.

$$X_q = (x \cdot \sin \omega_C t + y \cdot \cos \omega_C t) \cdot \cos \omega_C t - (x \cdot \cos \omega_C t - y \cdot \sin \omega_C t) \cdot \sin \omega_C t \quad (6.4a)$$

$$X_q = x \cdot \sin \omega_C t \cdot \cos \omega_C t - x \cdot \sin \omega_C t \cdot \cos \omega_C t + y \cdot \sin^2 \omega_C t + y \cdot \cos^2 \omega_C t \quad (6.4b)$$

$$X_q = x \cdot \sin \omega_C t \cdot \cos \omega_C t - x \cdot \sin \omega_C t \cdot \cos \omega_C t + y (\sin^2 \omega_C t + \cos^2 \omega_C t) \quad (6.4c)$$

$$X_q = y \quad (6.4d)$$

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

$$\begin{bmatrix} X_p \\ X_q \end{bmatrix} = \begin{bmatrix} \cos \omega_C t & \sin \omega_C t \\ -\sin \omega_C t & \cos \omega_C t \end{bmatrix} \cdot \begin{bmatrix} Z_p \\ Z_q \end{bmatrix} \quad (6.5)$$

$\beta = \omega_C t$ olarak tanımlanırsa; işaret durum diyagramı üzerindeki açısal vektör öteleme, β açısı kadar olur. Hilbert dönüşümü, zamanda tanımlı bir işlemdir. Demodülasyon işleminin sayısal işaret işleme teknikleri ile yapılacak olması, Hilbert dönüşümünün ayrık yapılmasını gerektirir. Bundan dolayı; Z_p ve Z_q , 14400Hz frekansında örneklenerek kullanılır. Haberleşme kanalında kullanılan işaret hızı 2400 sembol/saniye (baud) olduğu için, her sembolden 6 örnek alınır. Taşıyıcı frekansının 1800Hz ve işaret akış hızının 2400 sembol/saniye olması nedeniyle $\cos(\cdot)$ ve $\sin(\cdot)$

ifadelerinin alınacak her örnek değeri için farklılık gösterir. Bundan dolayı, Hilbert dönüşümlü demodülasyon işlemi için, $t = n.T$ olarak tanımlanırsa (6.6) elde edilir.

$$\omega_C.t = \omega_C.n.T = 2.\pi. f_C.n.T \quad (6.6)$$

$$T = 1/14400 \text{ sn} , f_C = 1800\text{Hz} \text{ ve } n : \text{örnek sayısı}$$

(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 gereklidir. Bu şekilde elde edilen X_p ve X_q değerleri işaret akış hızında örnekleterek, 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 Dönüştürücü

Hilbert dönüştürücüler, giriş işaretlerinin fourier bileşenlerinin fazlarını -90° ötelemekte kullanılırlar. Ayırık 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).

$$x^I(n) = x(n) + j.x^{II}(n) \quad (6.7)$$

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

$$X^I(e^{j\omega}) = X(e^{j\omega}) + j. X^{II}(e^{j\omega}) \quad (6.8)$$

$$X^I(e^{j\omega}) = 2.X(e^{j\omega}) \quad , 0 < \omega < \pi \quad (6.9a)$$

$$X^I(e^{j\omega}) = 0 \quad , \pi < \omega < 2\pi \quad (6.9b)$$

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

$$X^I(e^{j\omega}) = 2.X(e^{j\omega}) \quad , 0 < \omega < \pi \quad (6.10a)$$

$$X(e^{j\omega}) + j. X^{II}(e^{j\omega}) = 0, \pi < \omega < 2\pi \quad (6.10b)$$

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

$$2.X(e^{j\omega}) = X(e^{j\omega}) + j. X^{II}(e^{j\omega}) \quad (6.11a)$$

$$X(e^{j\omega}) = j \cdot X^{II}(e^{j\omega}) \quad (6.11b)$$

$$X^{II}(e^{j\omega}) = -j \cdot X(e^{j\omega}) \quad , 0 < \omega < \pi \quad (6.11c)$$

$X^{II}(e^{j\omega})$ ifadesi (6.10b) denklemi kullanılarak yazılırsa, (6.12) ifadesi elde edilir.

$$j \cdot X^{II}(e^{j\omega}) = -X(e^{j\omega}) \quad (6.12a)$$

$$j^2 \cdot X^{II}(e^{j\omega}) = -j \cdot X(e^{j\omega}) \quad (6.12b)$$

$$X^{II}(e^{j\omega}) = j \cdot X(e^{j\omega}) \quad , \pi < \omega < 2\pi \quad (6.12c)$$

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

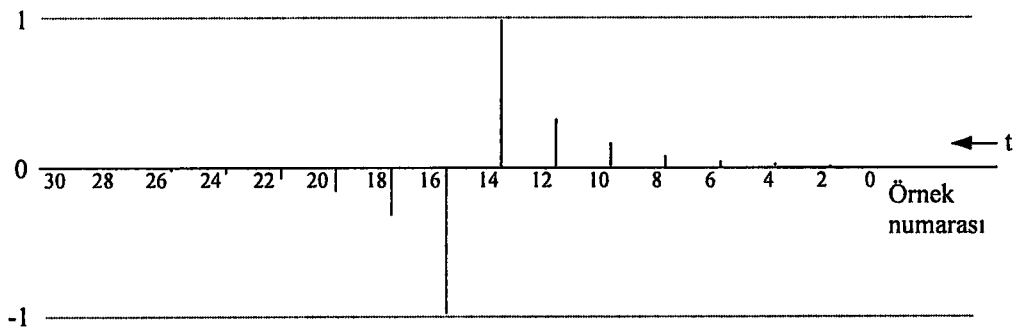
$$X^{II}(e^{j\omega}) = H_d(e^{j\omega}) \cdot X(e^{j\omega}) \quad (6.13a)$$

$$H_d(e^{j\omega}) = -j \quad , 0 < \omega < \pi \quad (6.13b)$$

$$H_d(e^{j\omega}) = j \quad , \pi < \omega < 2\pi \quad (6.13c)$$

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

Hilbert dönüştürücü darbe cevabı, nedensel değildir ve idealde sonsuz uzunluktadır. FIR filtre biçiminde tasarlanarak Hilbert dönüştürücü yapılabilir. Böyle bir dönüştürücünün; filtre uzunluğu, alt kesim frekansı, üst kesim frekansı ve yaklaşma hatası olmak üzere dört ayrı parametresi bulunmaktadır. Filtre cevabının simetrik olabilmesi için, alt ve üst normalize frekans değerlerinin toplamı normalize nyquist frekansına eşit olmalıdır. Ayrıca süzgeç 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 hatası 0.75, alt kesim frekansı 0.04 ve üst kesim frekansı 0.46 olarak kullanılmıştır.



(a) Hilbert Dönüştürücü Darbe Cevabı

Şekil 6.4 Hilbert Dönüştürücü Darbe, Genlik ve Faz Eğrileri

6.3.3 Hilbert Dönüştürücü ve Demodülatör Tsarımı

“6.3.2 Hilbert Dönüştürücü” bölümünde özellikleri verilen 31 uzunluklu Hilbert dönüştürücü yazılımı Tablo 6.1’ de verilmiştir. Şekil 6.4a’ da verilen dönüştürücü darbe büyüklükleri `hilbert_coeffs` tablosunda saklanmıştır. Örneklenmiş dönüştürücü giriş değerleri, `xv` tablosu ile yazılıma girer. Fazı dönmüş işaret Hilbert `Zq` değişkeni ile programdan çıkar.

Tablo 6.1 Hilbert Dönüştürücü Programı

```
/* hilbert transform alt program degerleri */
static float hilbert_coeffs[] = { 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};
#define NZEROS 30
#define KAZANC 1.578125000e+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;

```

```

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

```

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

Tablo 6.2 Kompleks Demodülatör Programı

```

/*****
/* Hilbert donusturucuden gelen Zp ve Zq kompleks demodulasyon islemine tabii tutulur */
/*****
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 Dengeleyici ve Açısal Hata Hesaplama

6.4.1 Uyarlamalı Dengeleyici Özellikleri

Haberleşme kanalının ideal olmaması ve “2.3 Modem Bağlantı Hatlarının İncelenmesi” bölümünde açıklanan özelliklerinden dolayı farklı sembol bileşenleri birbirlerine karışır. Bu bozulmaya, simgeler arası girişim (ISI) adı verilir. Hızın yüksek olduğu iletişimlerde semboller arası girişim, haberleşmeyi bozabilecek düzeyde etkin rol oynar. Kullanılan haberleşme kanalının karakteristik özelliklerinin tam olarak bilinmesi, semboller arası girişimin azaltılmasında kullanılacak dengeleyicilerin belirlenmesinde kolaylık sağlar.

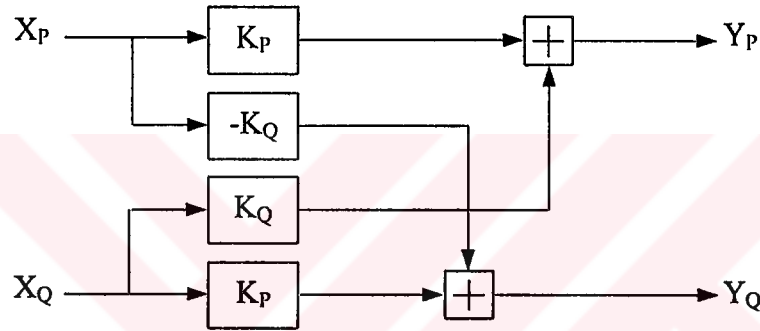
Fakat modem haberleşmesinde kullanılan haberleşme kanalları, rastgelelik gösterir ve özellikleri tam olarak bilinemez. Bundan dolayı, ortalama haberleşme kanal karakteristiklerine göre tasarlanmış dengeleyiciler semboller arası girişimi azaltmakta yeterince etkili olamazlar.

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 iletim yeteneklerinin verimli bir şekilde kullanılabilmesi için uyarlamalı dengeleme yapılması zorunludur.

Uyarlamalı dengeleyiciler, alınan önceki giriş işaretlerindeki bozulmalara göre katsayılarını ayarlayarak dengeleme işlemini gerçekleştirirler. Modemlerin bağlantı kurma sırasında yani el sıkışma aralığında, 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 tekniğinde, modülasyonlu işaret birbirine dik iki taşıyıcı içerdiği için kompleks olarak modellenmektedir. Bundan dolayı kullanılacak dengeleyici de kompleks özellik gösterir.



Şekil 6.5 Kompleks Dengeleyici Blok Diyagramı

K_P ve K_Q kompleks dengeleyici katsayılarıdır. Bu çalışmada; dengeleme işlemi demodülasyon sonrası yapıldığı için kullanılan dengeleyici temel bant dengeleyicisidir ve en küçük karesel ortalama hata yöntemi ile çalışır. Temel olarak dengeleyici (6.14)' deki gibi tanımlanmaktadır.

$$y(n) = \sum_{j=-m}^m K_j(n) \cdot x(n-j) \quad (6.14)$$

Şekil 6.5' de görülen blok diyagram ve (6.14)' de verilen matematiksel tanımdan, kullanılan uyarlamalı dengeleyicinin matematiksel ifadeleri elde edilir (6.15). m dengeleyici derinliğini ifade eder.

$$Y_P = \sum_{i=0}^m X_P^i \cdot K_P^i + X_Q^i \cdot K_Q^i \quad (6.15a)$$

$$Y_Q = \sum_{i=0}^m X_Q^i \cdot K_P^i - X_P^i \cdot K_Q^i \quad (6.15b)$$

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).

$$E_P(n) = f (|X^I(n) - X(n)|) \quad (6.16a)$$

$$E_Q(n) = f (|Y^I(n) - Y(n)|) \quad (6.16b)$$

$$E(n) = E_P(n) + j.E_Q(n) \quad (6.16c)$$

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

$$\begin{bmatrix} E_P \\ E_Q \end{bmatrix} = \begin{bmatrix} \cos(-\alpha) & \sin(-\alpha) \\ \sin(-\alpha) & \cos(-\alpha) \end{bmatrix} \cdot \begin{bmatrix} |X^I - X| \\ |Y^I - Y| \end{bmatrix} \quad (6.17a)$$

$$\alpha = \arctan (X / Y) - \arctan (X^I / Y^I) \quad (6.17b)$$

Dengeleyicinin kullandığı katsayıların yinelenmesi (6.18)'da tanımlanmıştır.

$$K(n+1) = K(n) + k. E(n) . E^*(n) \quad (6.18)$$

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

$$K_P(n+1) = K_P(n) + k. [E_P(n). X_P(n) + E_Q(n). X_Q(n)] \quad (6.19a)$$

$$K_Q(n+1) = K_Q(n) + k. [-E_P(n). X_Q(n) + E_Q(n). X_P(n)] \quad (6.19b)$$

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

$$0 < k < \frac{2}{N \cdot P_{ort}} \quad \begin{array}{l} N : \text{Vurum sayısı} \\ P_{ort} : \text{Ortalama giriş gücü} \end{array} \quad (6.20)$$

$$K_P^n(-1) = K_Q^n(-1) = 0 \quad (6.21a)$$

$$X_P(-n) = X_Q(-n) = 0 \quad (6.21b)$$

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

```
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 hızında dengeleyici çıkışları
/* oluşturulur.Kullanılan dengeleyici kompleks yapıdadır 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 Güncelleme Programı

```
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 döndürülen 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_outp,float tersdon_outq,float adaptin_xp,float adaptin_xq)
{
    int i;
    for (i=6;i>=1;i--) /* p ve q katsayıları birer ötelenir. yeni değerlere 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_outp+adaptin_xq*tersdon_outq);
    Kq_table[0] = Kq_table[1] +
        katsayi_yineleme_parametresi * (adaptin_xp*tersdon_outq+adaptin_xq*tersdon_outp);
}
/*****/
```

Tablo 6.5 Ters Vektör Döndürme Programı

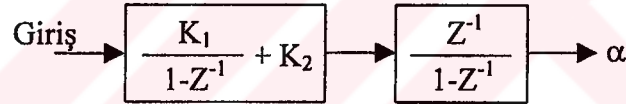
```

/*****
/* ters fark hata vektörü döndürme programı 2400Hz hızında çalışır.
/* ters hata vektör bileşenlerinin reel ve imajinal değerleri istenen açı kadar döndürülür.
/*****/
static void tersvektor_dondurme(float karar_hata_xp,float karar_hata_xq,float donme_acisi1)
{
    float radian_donme_acisi; float pi = 3.1415927;
    radian_donme_acisi = -1 * ( donme_acisi1 / 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 İşlemi

Karar verici girişine gelen vektör ile karar verici çıkışındaki çıkış vektörü arasındaki fark açısal hatanın hesaplanması ve uyarlamalı dengeleyicinin katsayılarının belirlenmesinde kullanılır. Açısal hata hesaplayıcı çıkış değeri, taşıyıcı çıkarma süzgecinden geçirilerek, bir sonraki vektör döndürme açı değeri (α) bulunur (Şekil 6.6). Taşıyıcı çıkarma filtresi, demodülatör katı ile birlikte incelendiğinde PLL çevrimi oluşturur. Hesaplanan açı hatası, bir sonraki vektör döndürme işleminde kullanılması bu çevrimi göstermektedir.



Şekil 6.6 Taşıyıcı Çıkarma Filtresi Blok Diyagramı

Şekil 6.6' da ikinci dereceden bir filtre görülmektedir. K_1 ve K_2 değerleri (6.22) ile bant genişliği değiştirilebilir.

$$K_1 = 1 - \varphi^2 - \sigma^2 \quad (6.22a)$$

$$K_2 = 1 + \varphi^2 - 2\varphi + \sigma^2 \quad (6.22b)$$

$$\varphi = \cos [\omega_n.T.(1-C^2)^{1/2}]. e^{-C.\omega_n.T} \quad (6.22c)$$

$$\sigma = \sin [\omega_n.T.(1-C^2)^{1/2}]. e^{-C.\omega_n.T} \quad (6.22d)$$

ω_n : Doğal frekans, C: Amortisman oranı ve T: Örnekleme periyodu

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

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

Vektör döndürücü çıkışında, (6.23) ifadesi yüksek geçiren filtre tanımladığı için frekans kayması oluşmayacak ve düşük frekansta faz kaymaları oluşmayacaktır. Hesaplanan açı farkı (α) ile, uyarlamalı dengeleyici çıkış vektörü $+\alpha$ ve karar verici fark değer vektörü $-\alpha$ ötelenir. Bu kurallar çerçevesinde yazılmış olan program Tablo 6.6’ da verilmiştir.

Tablo 6.6 Açısıl Hata Hesaplama Programı

```

/*****
/* açısıl hata kesaplayıcı ve PLL çevrimi 2400Hz hızında çalışır.
*****/
static void acisal_hata_hesaplayici_VCO(float parametre1,float parametre2,
float parametre3,float parametre4)
{
/* parametre1 ve 2 - alınan x ve y */ /* parametre3 ve 4 - karar verilen x ve y */
float hata_acisi; float pi = 3.1415927; float PLL_temp1,PLL_temp2,PLL_katsayi1,PLL_katsayi2;
hata_acisi = ( 180 / pi ) * ( atan(parametre2/parametre1) - atan(parametre4/parametre3) );
PLL_temp1 = cos( 2*pi*dogal_fr_deg*(1/orn_fr_deg)*sqrt(1-amortisman_deg*amortisman_deg))*
exp(-1*amortisman_deg*2*pi*dogal_fr_deg/orn_fr_deg);
PLL_temp2 = sin( 2*pi*dogal_fr_deg*(1/orn_fr_deg)*sqrt(1-amortisman_deg*amortisman_deg))*
exp(-1*amortisman_deg*2*pi*dogal_fr_deg/orn_fr_deg);
PLL_katsayi1=1-(PLL_temp1*PLL_temp1)-(PLL_temp2*PLL_temp2);
PLL_katsayi2=1+(PLL_temp1*PLL_temp1)-(2*PLL_temp1)+(PLL_temp2*PLL_temp2);
PLL_out = ( ( PLL_katsayi1/(1-PLL_log) )+PLL_katsayi2 ) * ( PLL_log/(1-PLL_log) ) * ata_acisi;
PLL_log = PLL_out;
}
/*****/

```

Uyarlamalı dengeleyici çıkışları, 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 gerçekleştirir.

$$\begin{vmatrix} Y_P \\ Y_Q \end{vmatrix} = \begin{vmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{vmatrix} \cdot \begin{vmatrix} X \\ Y \end{vmatrix} \quad (6.24)$$

Tablo 6.7 Vektör Döndürme Programı

```

/*****
/* dondürme girişindeki reel ve imajinal vektör bileşenleri istenen açı kadar döndürülür.
*****/
static void vektor_dondurme(float dond_giris_xp,float dond_giris_xq,float donme_acisi)
{ float radian_donme_acisi; float pi = 3.1415927;
radian_donme_acisi = ( donme_acisi / 180 ) * pi;
dond_cikis_xp = ( cos(radian_donme_acisi) * dond_giris_xp) +
(sin(radian_donme_acisi) * dond_giris_xq);
dond_cikis_xq = (-sin(radian_donme_acisi) * dond_giris_xp) +
(cos(radian_donme_acisi) * dond_giris_xq);
}
/*****/

```

6.5 KARAR VERİCİ

6.5.1 Katlamalı Kodlar İçin En Büyük Benzerlikli Kodçözme (Viterbi Algoritması)

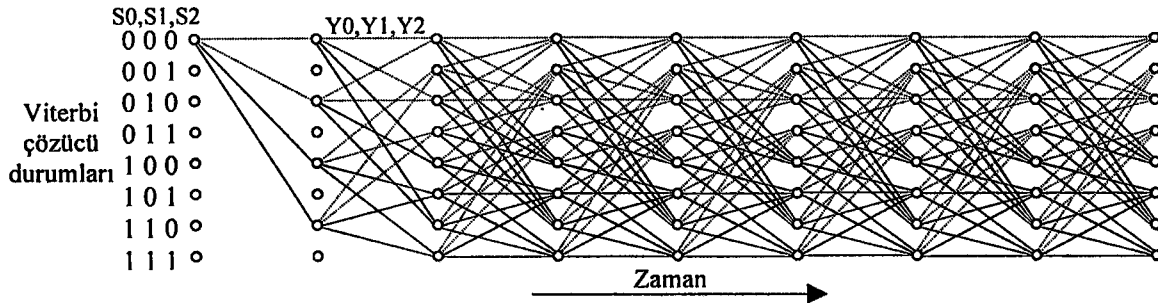
“2.6 Kafes Kodlamalı Modülasyon (TCM)” ve “3. ITU-T V.32 Standardı” bölümlerinde özellikleri incelenen kafes kodlu modülasyon (TCM) tarafından kodlanmış semboller, demodülatör tarafından temel banda indirildikten sonra, Viterbi algoritması kullanılarak çözülür. Viterbi algoritması, yazılım ile karar verilen ve en büyük benzerlik karar kuralı temel alınmış çözücü teknikleri üzerine oturtulmuş olup katlamalı kodların çözülmesinde kullanılan bir yapıdır. 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 çözücünün çıkış olarak karar verdiği nokta, aldığı işarete kullandığı işaret diyagramındaki en yakın noktadır. Başka bir deyişle, alınan işaret ile ideal işaretler arasındaki farkın en az olduğu ideal işaret çıkış olarak kabul edilir.

Yumuşak karar veren çözücü yapılarında, gelen veriyi çözmek için alınan değer ile birlikte geçmiş veri değerleri de kullanılır. Yumuşak karar veren herhangi bir çözücü için uygun fark fonksiyonunun kullanılması gereklidir. Her modülasyon tekniğinde tek bir fark fonksiyonu kullanılır. Yaygın olarak kullanılan iki fark fonksiyonu vardır. Bunlar Hamming ve Öklid fark fonksiyonları olarak adlandırılır.

Hamming fark fonksiyonunun ikili işaretlerde kullanılması uygundur. Faz kaymalı modülasyon veya dik genlik modülasyonlu işaretler için kendi işaret diyagramlarındaki Öklid fark fonksiyonları kullanılır. Beyaz gauss gürültülü haberleşme kanalları için, işaret diyagramındaki herhangi bir noktadan daha uzakta alınan işaret, o noktaya daha az uyar. Böylece; alınan sinyal ile varsayılan nokta arasındaki uzaklık her QAM işareti için bir fark fonksiyonu olur. V.32’ de QAM modülasyonu kullanılmasından bu yana, işaret diyagramı üzerindeki uzaklık tahmininde bu fark fonksiyonu kullanılır.

Şekil 6.7’ de V.32 standardında tanımlı TCM durum geçişleri gösterilmiştir. x ekseninde zaman aralıkları, y ekseninde ise kodlayıcıdaki olası 8 durum gösterilmiştir. Her hangi bir zaman aralığında, kodlayıcı sadece bir duruma erişebilir. Viterbi çözücü, bir yol seçerek karar verene kadar olası tüm durumların yollarını saklamalıdır. Yumuşak karar verme yapısının esası buna dayanır. Yani daha fazla bilgi alınıncaya kadar gerçek karar geciktirilir.



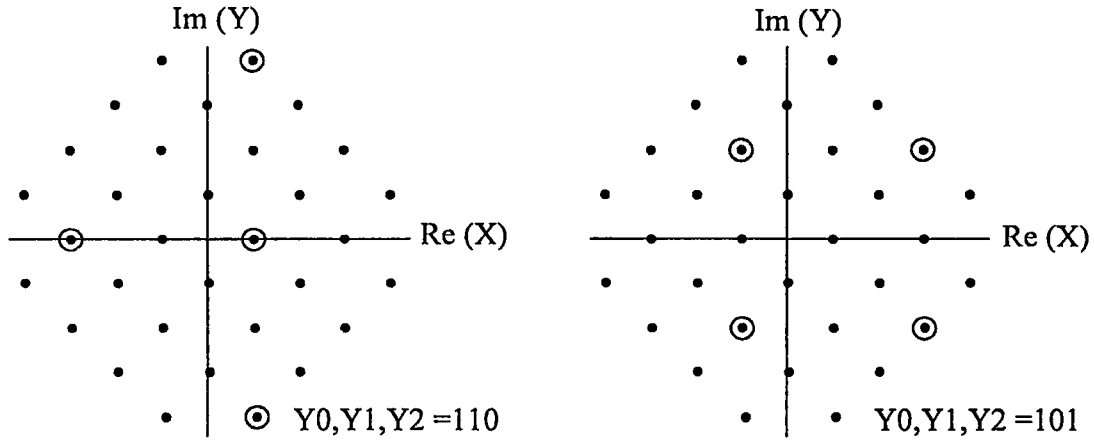
Şekil 6.7 8 Durumlu TCM Durum Geçişleri

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

Bu kısıtlamalardan dolayı pratikte Viterbi algoritmasında, çıkışla ilgili karar vermeden önce sonlu uzunluktaki giriş işaretleri dikkate alınır. Bu nedenle, belirli bir işaret dizi uzunluğuna kadar işaretler alınmakta ve sonra kod çözme işlemi başlamaktadır. Dolayısıyla, işaret alındığında karar verme işlemi yapılmamaktadır. Bahsedilen bu işaret dizi uzunluğu kod çözme derinliği olarak adlandırılır.

Şekil 6.7' de gösterildiği gibi, her zaman aralığında, 8 olası durum mevcuttur. Şimdiki zaman aralığındaki bir durumdan, sonraki zaman aralığındaki 4 farklı duruma geçilebilir. Aynı şekilde; önceki zaman aralığından, sadece 4 durumdan şimdiki zaman aralığındaki tek bir duruma gelinebilir. Bu kodlayıcı özelliğinden dolayı, herhangi bir zaman aralığındaki olası 8 duruma dörder yol ile gelinir. Viterbi çözücü, her olası 8 durum için varolan 4 adet yoldan alınan işarete en küçük karesel Öklid uzaklığı olanı seçer. Böylece; bir önceki zaman aralığından şimdiki zaman aralığındaki olası sekiz duruma geçiş yollarının karesel Öklid fark değerleri bulunmuş olunur. Bu yollara ait fark değerleri ile bu yolların gelmiş olduğu bir önceki zaman aralığındaki durumlar en uygun yol dizisinin karar verilmesine kadar saklanır ve bu karar işleminde kullanılır.

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.

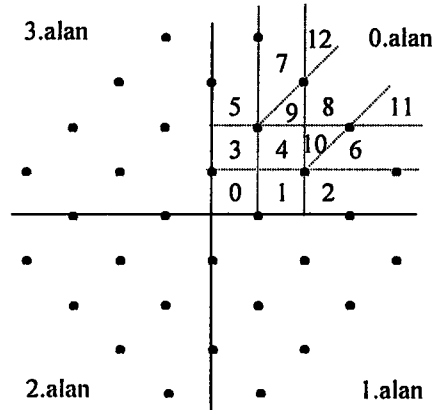


Şekil 6.8 V.32 TCM 32 – nokta İşaret Küme Örnekleri

Şekil 6.8’ de gösterildiği gibi V.32 işaret diyagramındaki her alt küme, simetrik ayarlı ve birbirlerine eşit uzaklıkta duran 4 nokta şeklinde tanımlanmıştır. Ayrıca, küme içindeki noktalar birbirinden olabildiğince uzakta bulunmaktadır. Her örnek aralığının başlangıcında Viterbi çözücü, alınan işareti her küme ve kümenin içinde alınan işaret ile örtüşen nokta ile karşılaştırır. Aslında bu yapı sert çözme yapısıdır. Bu yöntem, çözücü performans kalitesinde etkin değildir. Bunun nedeni her küme içinde bulunan 4 noktanın birbirlerine uzaklıklarının fazla olmasıdır. Dolayısı ile, herhangi bir bozucu gürültü kestirim üzerinde daha az etkili olmaktadır.

6.5.2 Viterbi Çözücü Tasarımı

Çözücü girişine gelen temel bant işaretinin, Şekil 6.9’ da tanımlanmış olan işaret bölgeleri ve alanları belirlenir. Tanımlanmış olan bu bölgelerin, işaret diyagramında bulunan 8 ayrı alt küme içerisindeki en yakın karesel Öklid uzaklığına sahip nokta koordinatları viterbi_bolge_table tablosunda tutulur.



Şekil 6.9 32 – nokta İşaret Bölgeleri

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

Tablodan okunan 8 nokta ile alınan nokta arasındaki karesel Öklid uzaklıkları hesaplanarak DIST_table' da tutulur. Viterbi derinliğine kadar her sembol iletim aralığına ait yol ve gecikme durumları VITERBI_PATH_table ile VITERBI_DLY_ST_table tablolarında saklanarak (6.25) ifadesiyle tanımlanan gecikme durumlarına ait hesaplanmış fark fonksiyonu bulunur.

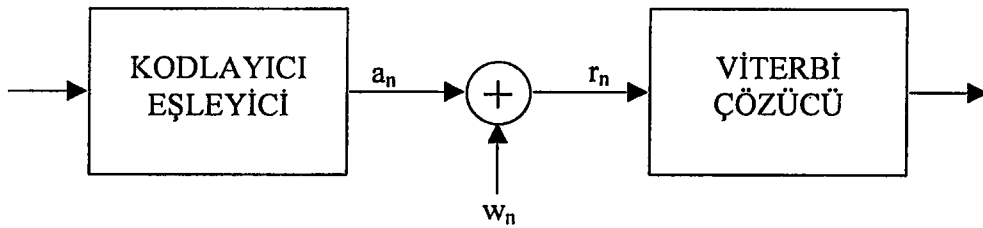
$$\text{fark}_n = 0,9.\text{fark}_{n-1} + 0,1.\text{öklid_uz} \quad (6.25)$$

Bu tanım, alçak geçiren süzgecin IIR olarak kullanılmasıdır. Viterbi derinliğine gelindiğinde, hesaplanmış fark fonksiyon tablosu olan ACCDIST[8] kullanılarak geriye dönük 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 Kodçözme” bölümünde ve bu bölümde 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 başarımının 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 işareti (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

Yapılan tasarım çalışmalarında C programlama dili kullanıldığı için üretilen rasgele sayılar düzgün dağılmaktadır. Bu nedenle düzgün dağılmış rasgele sayılardan gauss dağılım karakteristiğine sahip gürültü bileşenleri elde etmek için Tablo 6.8' deki program parçası yazılmıştır. Bu programda, düzgün dağılan iki rasgele elde edilmiş sayısal değerden AWGN_nx ve AWGN_ny gauss gürültü bileşenleri elde edilir. Programa standart sapma ve ortalama değer parametre olarak atanmaktadır. Beyaz gauss gürültüsü sıfır ortalama olarak kullanılmıştır.

Tablo 6.8 Gauss Gürültü Üretme Programı

```

/*****
static generate_gauss(float gauss_mean, float gauss_sigma)
{
    /* uniform değişken 0 - 32767 arasında olabilir. */
    float uni1, uni2, r; /* uniform and Rayleigh random değişkenler */
    float pi = 3.1415927;
    float rand1, rand2;
    /* 0-1 arasında uniform dağılmış 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 dağılmış 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 işaretin ve gürültünün gücü (6.26) ifadesindeki gibi gösterilir.

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

$$E[a^2(t)] = E_s = \frac{1}{32} \sum_{i=1}^{32} (a_{x_i}^2 + a_{y_i}^2) \quad (6.26b)$$

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

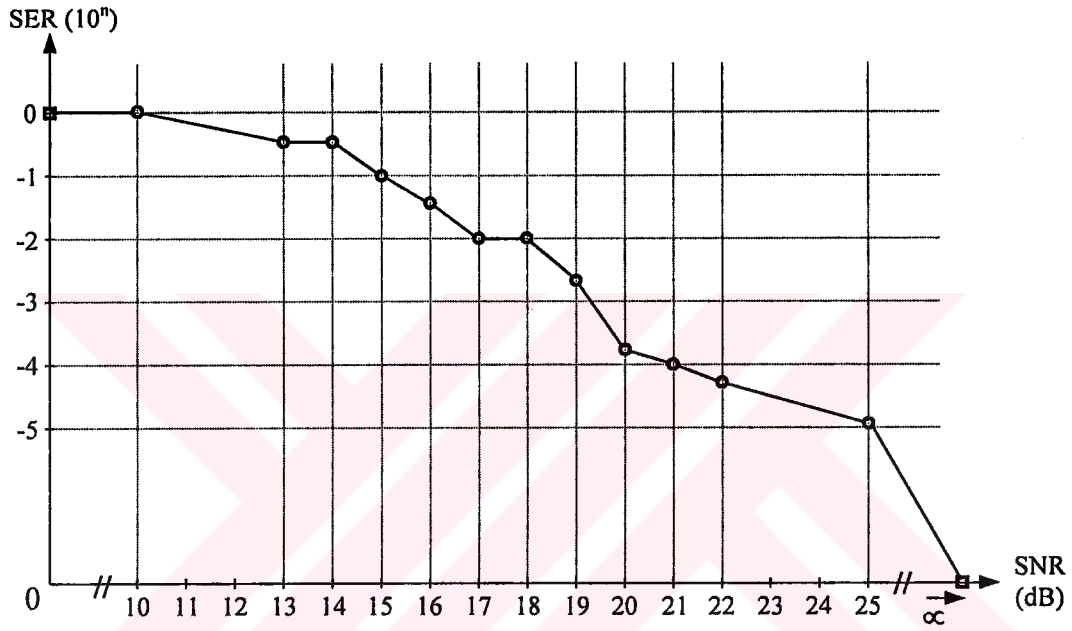
İşaret gürültü oranının değişimi, Viterbi çözücünün hata başarımını etkilemektedir. Sembol hata oranı (6.27)' deki gibi tanımlanmaktadır.

$$\text{SER} = \frac{\text{hatalı alınan sembol sayısı}}{\text{toplam alınan sembol sayısı}} \quad (6.27)$$

Sembol hata oranı, Tablo 6.9' da tanımlı test dizisi kullanılarak 1000 sembol sayısı için değişik SNR oranlarında deneysel olarak elde edilmiştir (Şekil 6.11).

Tablo 6.9 Test Amaçlı Kullanılan Sayısal Veri Dizisi

Test Dizisi Byte No	Test Byte	Hex Kod
1	10010110	0x96
2	00100111	0x27
3	00111101	0x3d
4	01100101	0x65
5	00101111	0x2f
6	00001011	0x0b
7	01111010	0x7a
8	01010110	0x56



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.

$$Q1_n = Y1_n \oplus Y1_{n-1} \quad (6.28)$$

$$Q2_n = (Q1_n \cdot Y1_{n-1}) \oplus Y2_{n-1} \oplus Y2_n \quad (6.29)$$

Ters farksal dönüşümü ile beş bitlik veriden Q1, Q2, Q3 ve Q4 verileri oluşturulur. Bu matematiksel ifadelerin gerçekleştirildiği 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_table1[0];
rec_q_table[1] = ( rec_q_table[0] & diff_encoder_log_table1[0] )
                ^ ( diff_encoder_log_table1[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_table1[0] = signal_element_mapping_output[1];
diff_encoder_log_table1[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, modemın bağlı bulunduğu terminale gönderilir. Çözme işleminde, arayan veya aranan modem olma durumuna göre V.32 standardında tanımlı olan çok terimliler kullanılır. V.32 standardında, arayan modem çözücü çok terimli (GPC), $1 + x^{-5} + x^{-23}$ ve aranan modem çözücü çok terimli (GPA), $1 + x^{-18} + x^{-23}$ olarak tanımlanmıştır. Buna göre karıştırıcı çıkışındaki veri katarı (D_O), (6.30) ve (6.31) ifadeleri ile gösterilebilir.

$$\text{Arayan modem ise; } D_O = D_S \cdot (1 \oplus x^{-5} \oplus x^{-23}) \quad (6.30)$$

$$\text{Aranan modem ise; } D_O = D_S \cdot (1 \oplus x^{-18} \oplus x^{-23}) \quad (6.31)$$

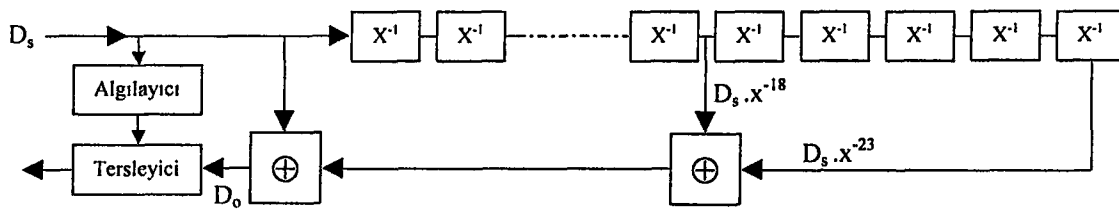
(6.30) ve (6.31) eşitliklerinde;

D_O , çözücü çıkışındaki veri katarı,

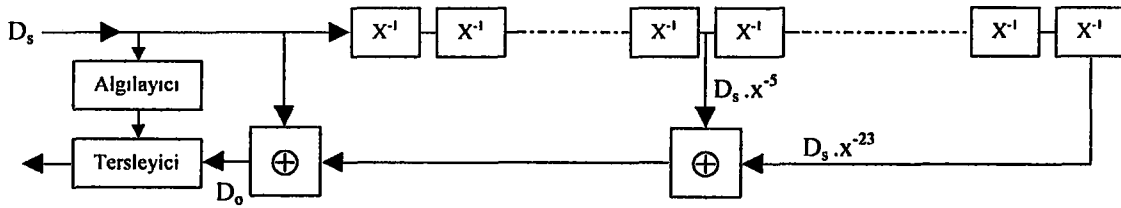
D_S , çözücü girişindeki veri katarı,

$\oplus$, EXOR işlemi,

$\cdot$, binary çarpma işlemini göstermektedir.



(a) GPA çok terimli kullanılarak çözücü katı blok diyagramı



(b) GPC çok terimli kullanılarak oluşturulan çözücü katı blok diyagramı

Şekil 6.12 Çözücü Katı Blok Diyagramları

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

6.6.2.2 Çözücü Tasarımı

Ters farksal 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

```

/*****
/* DeScrambler_V32(s16 descr_mode,s16 descr_rx_data)
/* Alınan 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 edilmiş veridir.
*****/
static s16 DeScrambler_V32( s16 descr_mode, s16 descr_rx_data )
{
    s16 descr_i,descr_scrmld_data;
    /* V.32 GPC ve GPA polinomu uyarınca descrambe veri olusturma*/
    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 oteleme*/
    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 gecikme gözü kullanılır ve gecikme değerleri `descramble_log_table`’ da tutulur. Başlangıçta

tüm descramble_log_table deęerleri “0” olarak tanımlanmıştır. Ters farksal dönüştürücüden gelen D_s verisi, çözücü çok terimlisine (Modemin arayan ve aranan olmasına göre kullanılan çok terimli deęişmektedir.) göre ilgili gecikme gözlerindeki deęerler ile EXOR işlemine tabi tutulur. İşlem sonucu (D_o), oluşturulur ve yeni gecikme deęerleri descramble_log_table’ a yerleştirilir. Alınan veri (D_s) katarının, 64 bit uzunluęuna ulaşması sonrası bir sonraki çıkış verisi (D_o) terslenerek oluşturulur.

6.7 SONUÇ

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

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

Böylece; tasarlanmış modülatör ve demodülatör yapılarının 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 modülasyon teknikleri kullanılır.

Genişletilmiş işaret kümeleri kullanan kafes kodlamalı modülasyon tekniğinde, kodlanmış işaret dizileri arasındaki en küçük Öklid uzaklığı mümkün olabilecek en büyük uzaklık yapılır. Böylece, ardışık oluşturulan temel bant işaretleri arasındaki uzaklığın artması sağlanarak veri iletişiminde oluşabilecek hata olasılıkları azaltılır.

Bu tez çalışmasında; ITU V.32 standardı referans alınarak, 9600 bit/saniye hızında çalışan modülatör ve demodülatör yapıları tasarlanmıştır. Tasarım çalışmaları; TMS320C5402 sayısal işaret işlemcisi ve program geliştirme kiti üzerinde, kafes kodlamalı modülasyon ve Viterbi algoritmaları kullanılarak yapılmıştır. Tasarım çalışmaları ile ilgili bilgiler beşinci ve altıncı bölümlerde verilmiştir. Yazılım çalışmalarında, C programlama dili kullanılmıştır. Program, derlenerek makine diline çevrilmiş ve DSK kartı üzerinde çalıştırılmıştır.

Tasarlanan modülatör yazılımı, 64 bit uzunluğundaki test dizileri ile çalıştırılmıştır. Modülasyon programı, bu test dizilerindeki verileri 9600Hz hızında alarak modüle eder ve 14400Hz örnekleme hızında modülasyonlu sembolleri oluşturur. Örneklenmiş analog semboller, demodülasyon programı ile sayısal veriler haline getirilmiştir. Demodülasyon işlemi sonrası elde edilen sayısal veri dizisinin, ideal kanal kullanıldığında modülasyon işleminde kullanılan test dizisinin aynısı olduğu görülmüştür. Viterbi çözücünün hata başarımı değişik işaret gürültü oranları için incelenmiştir.

Gerçekleştirilen yazılımlar, modemlerin MAFE katında koşulan ve modem hızına göre ITU-T tarafından standartlaştırılmış tekniklerle modülasyon ve demodülasyonu gerçekleştiren yazılımlardır.

Dünya çapında kendini kanıtlamış modem tümleşik devreleri ve yazılımları tasarlayan firmalar, modem üreticilerine toplam çözümler sunmaktadırlar. 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 çalışmasında yapılmış olan yazılımlar daha yüksek hızlarda modemler için uyarlanabilir esnekliktedir. Ayrıca, daha üst seviyede modem kontrol yazılımları da yazılarak değişik uygulamalarda ticari olarak kullanılabilir.



KAYNAKLAR

- [1] Ungerboeck, G., 1982. Channel Coding with Multilevel / Phase Signals, *IEEE Transactions On Information Theory*, Vol.IT-28, 55-67.
- [2] Ungerboeck, G., 1987. Trellis-Coded Modulation with Redundant Signal Sets, Part I : Introduction, *IEEE Communication Magazine*, Vol.25, No.2, 5-11.
- [3] Ungerboeck, G., 1987. Trellis-Coded Modulation with Redundant Signal Sets, Part II : State of the Art, *IEEE Communication Magazine*, Vol.25, No.2, 12-21.
- [4] Wei, L.F., 1984. Rotationally Invariant Convolutional Channel Coding with Expanded Signal Space, Part I and II, *IEEE Journal on Selected Areas in Communications*, Vol.SAC-2, No.5, 659-686.
- [5] Hughes, L., 1992. Data Communications, McGraw-Hill, USA.
- [6] Viterbi, A.J. and Omura, J.K., 1979. Principles of Digital Communication and Coding in *Convolutional Codes*, pp. 227-261, McGraw-Hill, USA.
- [7] Biglieri, E., 1991. Introduction to Trellis-Coded Modulation, With Applications, Macmillan, New York.
- [8] Henrix, H., 1996. Viterbi Decoding Techniques in the TMS320C54x Family Application Report, Texas Instruments, USA.
- [9] Sweeney, P., 1991. Error Control Coding An Introduction, Prentice Hall, UK.
- [10] Chishtie, M.A., 1994. Telecommunications Applications with the TMS320C5x DSPs Application Book, Texas Instruments, USA.
- [11] Johannesson, R. and Zigangirov, K.S., 1999. Fundamentals of Convolutional Coding in *Convolutional Coding*, pp. 233-388, IEEE Press, New York, NY.

- [12] **ITU-V.32**, 1993. A Family of 2-Wire, Duplex Modems Operating at Data Signalling Rates of Up to 9600 bit/s For Use on The General Switched Telephone Network and on Leased Telephone-Type Circuits, *ITU-T*, Helsinki.
- [13] **<http://www.ti.com/>**
- [14] **Anonim**, 1998. DSP User's Guide - TMS32C5x, Texas Instruments, USA.
- [15] **Anonim**, 1999. Code Composer Studio User's Guide, Texas Instruments, USA.
- [16] **ITU-V.22bis**, 1988. 2400 Bits Per Second Duplex Modem Using The Frequency Division Technique Standardized For Use on The General Switched Telephone Network and on Point-to-point 2-Wire Leased Telephone-Type Circuits, *ITU-T*, Melbourne.
- [17] **Bingham, J.**, 1988. The Theory and Practice of Modem Design, John Wiley&Sons, Palo Alto, California.
- [18] **<http://www-users.cs.york.ac.uk/>**
- [19] **Hahn, S.L.**, 1996. Hilbert Transforms in Signal Processing, Artech House, UK.

ÖZGEÇMİŞ

5 Ağustos 1974' de İstanbul' da doğdu. İlköğrenimini Şeyhli İlkokulunda, orta öğrenimini ise Pendik Doğu Ortaokulunda tamamladı. Lise öğrenimine 1988 yılında Tuzla Endüstri Meslek Lisesi Elektronik Bölümünde başladı. 1992 yılında, Tuzla Teknik Lisesi Bilgisayar Bölümünden mezun oldu. Aynı yıl Yıldız Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği bölümünde lisans öğrenimine başladı. 1996 yılında, bölüm üçüncüsü olarak mezun olduktan sonra İstanbul Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği yüksek lisans programına girdi. Halen aynı bölümde öğrenimini sürdürmekte olup tez aşamasına gelmiştir. 1996 yılında Adam Elektronik Ltd. Şti.' de Ar-Ge bölümünde tasarım mühendisi olarak çalışmaya başlamıştır. Halen aynı firmada Ar-Ge Müdürlüğü görevini yürütmektedir.

