

1550 P

**DESIGN OF REED-SOLOMON AND CONVOLUTIONAL CODED OFDM  
SYSTEM AND PERFORMANCE ANALYSIS**

A Thesis

Presented to the Institute of Science and Engineering  
of IŞIK University

In Partial Fulfillment of the Requirements for the Degree of  
Master of Science

in

The Department of Electronics Engineering

by

Ufuk Bal

2003

Approval of the Institute of Science and Engineering

Prof. Dr. Endal Panayirci

(Title and Name)  
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.



Prof. Dr. Endal Panayirci  
(Title and Name)  
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

\_\_\_\_\_  
(Title and Name)  
Co-Supervisor

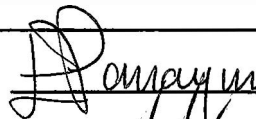
Prof. Dr. Endal Panayirci  
(Title and Name)  
Supervisor

Examining Committee of Members

Prof. Dr. Endal Panayirci (IŞIK)

Prof. Dr. Ahmet Akşen (IŞIK)

Prof. Dr. Ümit Aygözü (İTÜ)





## **ABSTRACT**

### **DESIGN OF REED-SOLOMON AND CONVOLUTIONAL CODED OFDM SYSTEM AND PERFORMANCE ANALYSIS**

**Bal, Ufuk**

In this study, OFDM and error-correcting codes, which have an important place in digital communication systems, have been studied. Literature research has been made on OFDM and error-correcting codes like Reed-Solomon and convolutional codes. Reed-Solomon and convolutional coded OFDM system has been designed. Furthermore, system's performance analysis has been made.

OFDM is a special case of multicarrier transmission. The basic principle of OFDM is to split a high-rate datastream into a number of lower rate streams that are transmitted simultaneously over a number of subcarriers. One of the main reasons to use OFDM is to increase the robustness against frequency selective fading or narrowband interference. In a single carrier system, a single fade or interferer can cause the entire link to fail, but in a multicarrier system, only a small percentage of the subcarriers will be affected. Error correcting codes can then be used to correct the errors.

In error correcting, combining convolutional and block codes in a concatenated code is particularly powerful technique. In this study Reed-Solomon codes are used as an outer code and convolutional codes are used as an inner code.

## ÖZET

### REED-SOLOMON VE KONVOLÜSYONEL KODLU DİKGEN FREKANS BÖLMELİ ÇOĞULLAMA(OFDN) SİSTEMİ'NİN TASARIMI VE PERFORMANS ANALİZİ

**Bal, Ufuk**

Bu çalışmada sayısal haberleşme sistemlerinde önemli yeri olan OFDM ve hata düzelten kodlar incelenmiştir.OFDN ve hata düzelten kodlama çeşitlerinden Reed-Solomon ve Konvolüsyonel kodlara yönelik bir literatür araştırması ile Reed-Solomon ve Konvolüsyonel kodlu OFDM sistem tasarımı yapılmıştır.Ayrıca bu sistemin performans analizi yapılmıştır.

OFDM çok-taşıyıcılı iletimin özel bir durumudur. OFDM'in temel prensibi yüksek orandaki bilgi dizisini alttaşıyıcılar üzerinden aynı anda iletilen daha düşük oranda bilgi dizilerine ayırmaktır. OFDM kullanmanın önemli nedenlerinden biri frekans seçmeli bozulmaya veya darband girişimine karşı dayanıklılığı arttırmasıdır.Tek taşıyıcılı bir sistemde, tek bir bozulma veya girişim bütün hattı etkileyebilir, fakat çok taşıyıcılı bir sistemde, sadece küçük bir oranı etkilenir.Daha sonrada hata düzeltim kodlaması, bu hataları düzeltmek için kullanılabilir.

Hata düzeltmede Konvolüsyonel ve blok kodların birlikte kullanımı çok etkili bir yöntemdir.Bu çalışmada dış kod olarak Reed-solomon ve iç kod olarak Konvolüsyonel kodları kullanılmıştır.



## TABLE OF CONTENTS

	PAGE
ABSTRACT .....	iii
OZET .....	iv
TABLE OF CONTENTS .....	v
LIST OF TABLES .....	viii
CHAPTER	
1. INTRODUCTION .....	1
2. OFDM .....	2
2.1. Evolution of OFDM .....	2
2.2. OFDM Basics .....	5
2.3. Guard Time and Cyclic Extension .....	12
2.4. Windowing .....	13
2.5. Choice of OFDM Parameters .....	17
2.6. Need For Coding .....	18
2.6.1. Block Coding in OFDM .....	18
2.6.2. Convolutional Coding .....	19
2.6.3. Concatenated Coding .....	19
2.7 Synchronization .....	20
2.7.1 Symbol synchronization .....	20
2.7.1.1 Timing errors .....	20
2.7.1.2 Carrier phase noise .....	21
2.7.2 Sampling-frequency synchronization .....	21
2.7.3 Carrier Frequency Synchronization .....	22

2.7.3.1	Frequency Errors .....	22
2.7.3.2	Frequency Estimators .....	22
2.8	Detection .....	23
2.8.1	Coherent Detection .....	23
2.9	Applications of OFDM .....	25
2.9.1	Digital Audio Broadcasting .....	25
2.9.2	Terrestrial Digital Video Broadcasting .....	27
3.	CODING .....	30
3.1.	Introduction .....	31
3.2.	Types of Errors .....	33
3.3.	Error Control Strategies .....	34
3.4.	Types of Codes .....	34
4.	LINEAR BLOCK CODES .....	36
4.1.	Hamming Codes .....	41
4.2.	BCH Codes .....	42
4.2.1.	Decoding of BCH Codes .....	45
4.2.1.1.	Peterson's Algorithm .....	47
4.2.1.2.	Berlekamp's Algorithm .....	49
5.	REED SOLOMON CODES .....	52
5.1.	Historical Overview .....	52
5.2.	Definition .....	54
5.3.	Encoding .....	55
5.4.	Decoding .....	56
5.4.1.	Peterson's Algorithm .....	58

5.4.2. Berlekamp's Algorithm .....	63
5.4.3. Euclid's Algorithm .....	66
6. CONVOLUTIONAL CODING .....	70
6.1. Structural Properties of Convolutional Codes.....	74
6.2. Viterbi Algorithm .....	76
6.3. Punctured Convolutional Codes .....	77
7. SIMULATION .....	79
8.CONCLUSION .....	85
REFERENCES .....	86

## LIST OF TABLES

TABLE	TITLE	PAGE
2.1	Transmission modes of DAB.....	26
4.1	Linear Block Code with $k = 4$ , $n=7$ .....	38
4.2	Minimal polynomials of the elements in $GF(2^4)$ generated by $p(x)=1+x+x^4$ .....	45
4.3	Three representations for the elements of $GF(2^4)$ generated by $p(x)=1+x+x^4$ .....	46
4.4	Berlekamp's iterative algorithm .....	51
5.1	Application of Berlekamp's algorithm .....	67
5.2	Application of Euclid's algorithm for integers .....	70
5.3	Application of Euclid's algorithm .....	72
7.1	Parameter's of the channel .....	86

# 1 INTRODUCTION

Multi-carrier modulation, in particular Orthogonal Frequency Division Multiplexing (OFDM), has been successfully applied to a wide variety of digital communications applications over the past several years.

In almost all applications of multi-carrier modulation, satisfactory performance cannot be achieved without the addition of some form of coding. In wireless systems subjected to fading, extremely high signal-to-noise ratios are required to achieve reasonable error probability. In addition, interference from other wireless channels is frequently severe. On wireline systems, large constellation sizes are commonly employed to achieve high bit rates. Coding in this case is essential for achieving the highest possible rates in the presence of crosstalk and impulsive and other interference.

So, here first the basics of OFDM are presented. In addition to basics of OFDM, the subsystems of an OFDM implementation are described, such as synchronization and coding. Then error correcting codes, specially Reed-Solomon codes, are described in detail. At last, RS and Convolutional coded OFDM system's simulation results are given. The system's details are given in the 7th chapter.

## 2 OFDM

### 2.1 Evolution of OFDM

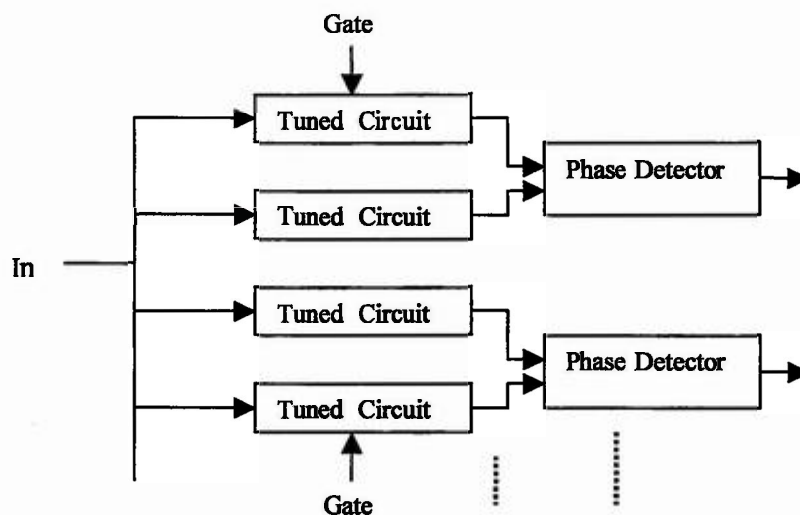
The use of Frequency Division Multiplexing (FDM) goes back over a century, where more than one low rate signal, such as telegraph, was carried over a relatively wide bandwidth channel using a separate carrier frequency for each signal. To facilitate separation of the signals at the receiver, the carrier frequencies were spaced sufficiently far apart so that the signal spectra did not overlap. Empty spectral regions between the signals assured that they could be separated with readily realizable filters. The resulting spectral efficiency was therefore quite low.

Instead of carrying separate messages, the different frequency carriers can carry different bits of a single higher rate message. The source may be in such a parallel format, or a serial source can be presented to a serial-to-parallel converter whose output is fed to the multiple carriers. Such a parallel transmission scheme can be compared with a single higher rate serial scheme using the same channel. The parallel system, if built straightforwardly as several transmitters and receivers, will certainly be more costly to implement. Each of the parallel sub-channels can carry a low signalling rate, proportional to its bandwidth. The sum of these signalling rates is less than can be carried by a single serial channel of that combined bandwidth because of the unused guard space between the parallel subcarriers. On the other hand, the single channel will be far more susceptible to inter-symbol interference. This is because of the short duration of its signal elements and the higher distortion produced by its wider frequency band, as compared with the long duration signal elements and narrow bandwidth in sub-channels in the parallel system. Before the development of equalization, the parallel technique was the preferred means of achieving high rates over a dispersive channel, in spite of its high cost and relative bandwidth inefficiency. An added benefit of the parallel technique is reduced susceptibility to most forms of impulse noise.



The first solution of the bandwidth efficiency problem of multi-tone transmission (not the complexity problem) was probably the “Kineplex” system. The Kineplex system was developed by Collins Radio Co. For data transmission over an H.F. radio channel subject to severe multi-path fading. In that system, each of 20 tones is modulated by differential 4-PSK without filtering. The spectra therefore of the  $\sin(kf)/f$  shape and strongly overlap. However, similar to modern OFDM, the tones are spaced at frequency intervals almost equal to the signalling rate and are capable of separation at the receiver.

The reception technique is shown in Figure 2.1. Each tone is detected by a pair of tuned circuits. Alternate symbols are gated to one of the tuned circuits, whose signal is held for the duration of the next symbol. The signals in the two tuned circuits are then processed to determine their phase difference, and therefore the transmitted information. The older of the two signals is then quenched to allow input of the next symbol. The key to the success of the technique is that the time response of each tuned circuit to all tones. Other than the one to which it is tuned, goes through zero at the end of the gating interval, at which point that interval is equal to the reciprocal of the frequency separation between tones. The gating time is made somewhat shorter than the symbol period to reduce inter-symbol interference, but efficiency of 70% percent of the Nyquist rate is achieved. High performance over actual long H.F. channels was obtained, although at a implementation cost. Although fully transistorized, the system required two large bays of equipment.



**Figure 2.1** The Collins Kineplex receiver

A subsequent multi-tone system was proposed using 9-point QAM constellations on each carrier, with correlation detection employed in the receiver. Carrier spacing equal to the symbol rate provides optimum spectral efficiency. Simple coding in the frequency domain is another feature of this scheme. The above techniques do provide the orthogonality needed to separate multi-tone signals spaced by the symbol rate. However the  $\text{sinc}(kf)/f$  spectrum of each component has some undesirable properties. Mutual overlap of a large number of subchannel-spectra is pronounced. Also, spectrum for the entire system must allow space above and below the extreme tone frequencies to accommodate the slow decay of the sub-channel spectra. For these reasons, it is desirable for each of the signal components to be bandlimited so as to overlap only the immediately adjacent sub-carriers, while remaining orthogonal to them.

The major contribution to the OFDM complexity problem was the application of the Fast Fourier Transform (FFT) to the modulation and demodulation processes. Fortunately, this occurred at the same time digital signal processing techniques were being introduced into design of modems. The technique involved assembling the input information into blocks of  $N$  complex numbers, one for each sub-channel. An inverse FFT is performed on each block, and the resultant transmitted serially. At the receiver, the information is recovered by performing an FFT on the received block of signal samples. This form of OFDM is often referred to as Discrete Multi-Tone (DMT). The spectrum of the signal on the line is identical to that of  $N$  separate QAM signals, at  $N$  frequencies separated by the signalling rate. Each such QAM signal carries one of the original input complex numbers. The spectrum of each QAM signal is of the form  $\text{sinc}(kf)/f$ , with nulls at the center of the other sub-carriers, as in the earlier OFDM systems.

Overlap of consecutive transmitted blocks is a problem, we can solve by using cyclic prefix. Another issue is how to transmit the sequence of complex numbers from the output of the inverse FFT over the channel.

The process is straightforward if the signal is to be further modulated by a modulator with I and Q inputs.



Otherwise, it is necessary to transmit real quantities. This can be accomplished by first appending the complex conjugate to the original input block. A  $2N$ -point inverse FFT now yields  $2N$  real numbers to be transmitted per block, which is equivalent to  $N$  complex numbers.

The most significant advantage of this DMT approach is the efficiency of the FFT algorithm. An  $N$ -point FFT requires only on the order of  $N \log N$  multiplications, rather than  $N^2$  as in a straightforward computation. The efficiency is particularly good when  $N$  is a power of 2, although that is not generally necessary. Because of the use of the FFT, a DMT system typically requires fewer computations per unit time than an equivalent single channel system with equalization. An overall cost comparison between the two systems is not as clear, but the costs should be approximately equal in most cases.

Over the last 20 years or so, OFDM techniques and, in particular, the DMT implementation, has been used in a wide variety of applications. Several OFDM voiceband modems have been adopted as the standard for the Asymmetric Digital Subscriber Line (ADSL), which provides digital communication at several Mb/s from a telephone company central office to a subscriber, and a lower rate in the reverse direction, over a normal twisted pair of wires in the loop plant.

OFDM has been particularly successful in numerous wireless applications, where its superior performance in multi-path environments is desirable. Wireless receivers detect signals distorted by time and frequency selective fading. OFDM in conjunction with proper coding and interleaving is a powerful technique for combating the wireless channel impairments that a typical OFDM system might face.

## **2.2 OFDM Basics**

The basic principle of OFDM is to split a high-rate datastream into a number of lower rate streams that are transmitted simultaneously over a number of subcarriers. Because the symbol duration increases for the lower rate parallel subcarriers, the relative amount of dispersion in time caused by multipath delay spread is decreased.

An OFDM signal consists of a sum of subcarriers that are modulated by using phase shift keying (PSK) or quadrature amplitude modulation (QAM). If  $d_i$  are the complex QAM symbols  $N_s$  is the number of subcarriers,  $T$  the symbol duration, and  $f_c$  the carrier frequency, then one OFDM symbol starting at  $t=t_s$  can be written as

$$s(t) = \text{Re} \left\{ \sum_{i=-N_s/2}^{i=N_s/2-1} d_{i+N_s/2} \exp(j2\pi(f_c - (i+0.5)/T)(t-t_s)) \right\}, t \leq t_s \leq t_s+T$$

$$s(t) = 0, t < t_s \wedge t > t_s+T$$

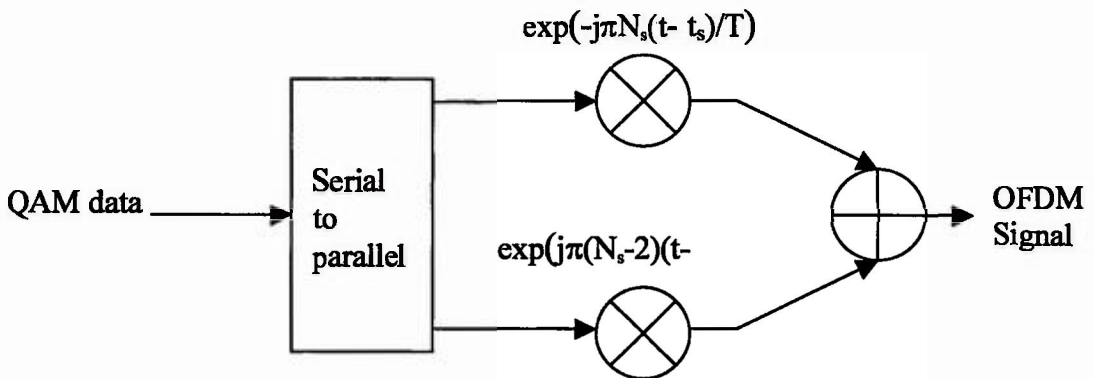
(2.1)

In the literature, often the complex baseband notation is used. The real and imaginary parts correspond to the in-phase and quadrature parts of the OFDM signal which have to be multiplied by a cosine and sine of the desired carrier frequency to produce the final OFDM signal.

$$s(t) = \sum_{i=-N_s/2}^{i=N_s/2-1} d_{i+N_s/2} \exp(j2\pi(i/T)(t-t_s)), t \leq t_s \leq t_s+T$$

$$s(t) = 0, t < t_s \wedge t > t_s+T$$

(2.2)

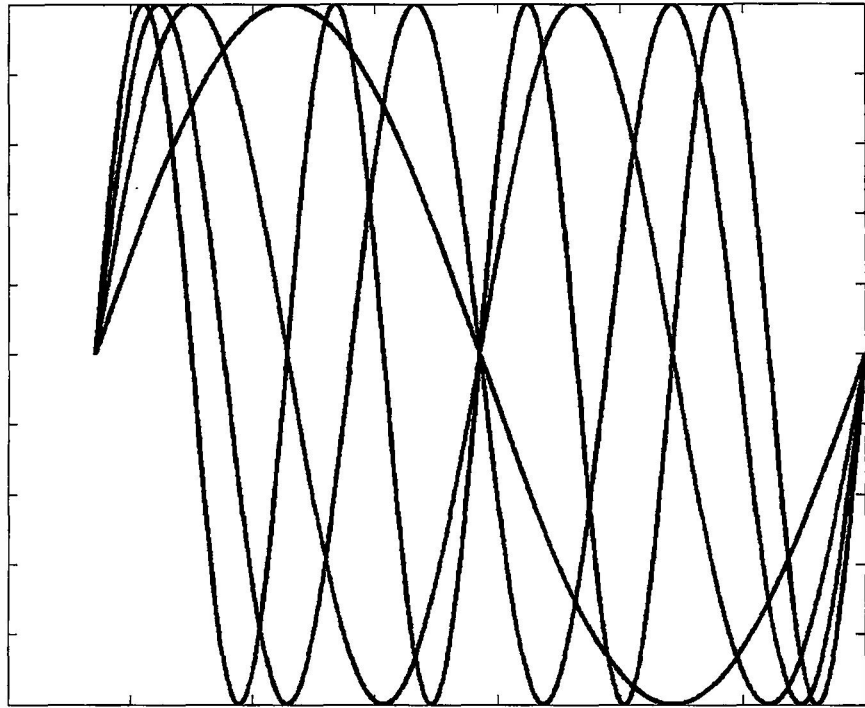


**Figure 2.2 OFDM Modulator**

As an example, Figure 2.3 shows four subcarriers from one OFDM signal. In this example, all subcarriers have the same phase and amplitude, but in practice the amplitudes and phases may be modulated differently for each subcarrier. Note that each subcarrier has exactly an integer number of cycles in the interval  $T$ , and the number of cycles between adjacent subcarriers differs by exactly one. This property accounts for the orthogonality between the subcarriers. For instance, if the  $j$ th subcarrier from (2.2) is demodulated by downconverting the signal with a frequency of  $j/T$  and then integrating the signal over  $T$  seconds, the result is as written in (2.3). By looking at the intermediate result, it can be seen that a complex carrier is integrated over  $T$  seconds. For the demodulated subcarrier  $j$ , this integration gives the desired output  $d_{j+N/2}$  (multiplied by a constant factor  $T$ ), which is the QAM value for that particular subcarrier. For all other subcarriers, the integration is zero, because the frequency difference  $(i-j)/T$  produces an integer number of cycles within the integration interval  $T$ , such that the integration result is always zero.

$$\begin{aligned}
 & \int_{t_s}^{t_s+T} \exp(j2\pi j/T)(t-t_s) \sum_{i=(N/2)}^{(N/2)-1} d_{i+N/2} \exp(j2\pi i/T)(t-t_s) dt \\
 &= \sum_{i=(N/2)}^{(N/2)-1} d_{i+N/2} \int_{t_s}^{t_s+T} \exp(j2\pi(i-j)/T)(t-t_s) dt = d_{j+N/2} T
 \end{aligned}$$

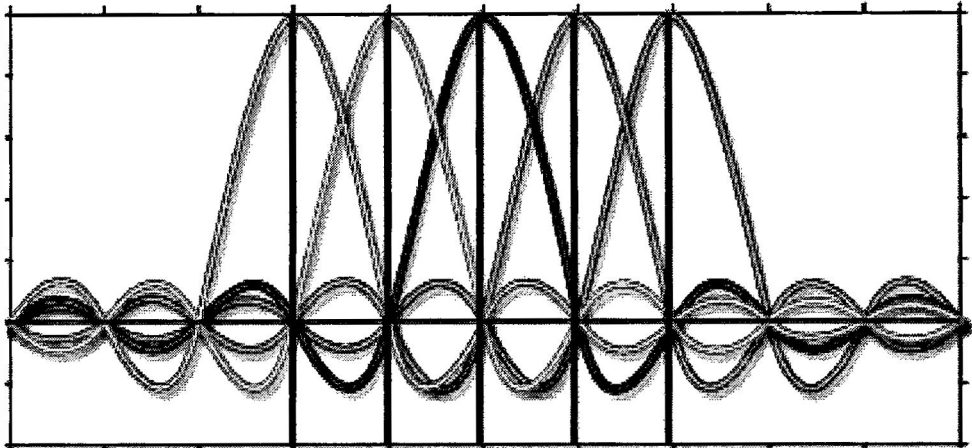
(2.3)



**Figure 2.3** Example of four subcarriers in one OFDM symbol

The orthogonality of the different OFDM subcarriers can also be demonstrated in another way. According to (2.1), each OFDM symbol contains subcarriers that are nonzero over a  $T$ -second interval. Hence, the spectrum of a single symbol is a convolution of a group Dirac pulses located at the subcarrier frequencies with the spectrum of a square pulse that is one for a  $T$ -second period and zero otherwise. The amplitude spectrum of the square pulse is equal to  $\text{sinc}(\pi fT)$ , which has zeros for all frequencies  $f$  that are an integer multiple of  $1/T$ . This effect is shown in Figure 2.4 which shows the overlapping sine spectra of individual subcarriers. At the maximum of each subcarrier spectrum, all other subcarrier spectra are zero. Because an OFDM receiver essentially calculates the spectrum values at those points that correspond to the maxima of individual subcarriers, it can demodulate each subcarrier free from any interference from the other subcarriers. Basically, Figure 2.4 shows that the OFDM spectrum fulfills Nyquist's criterion for an intersymbol interference free pulse shape. Notice that the pulse shape is present in the frequency domain and not in the time domain, for which the Nyquist criterion usually applied. Therefore, instead of intersymbol interference (ISI), it is intercarrier

interference (ICI) that is avoided by having the maximum of one subcarrier spectrum correspond to zero crossings of all the others.



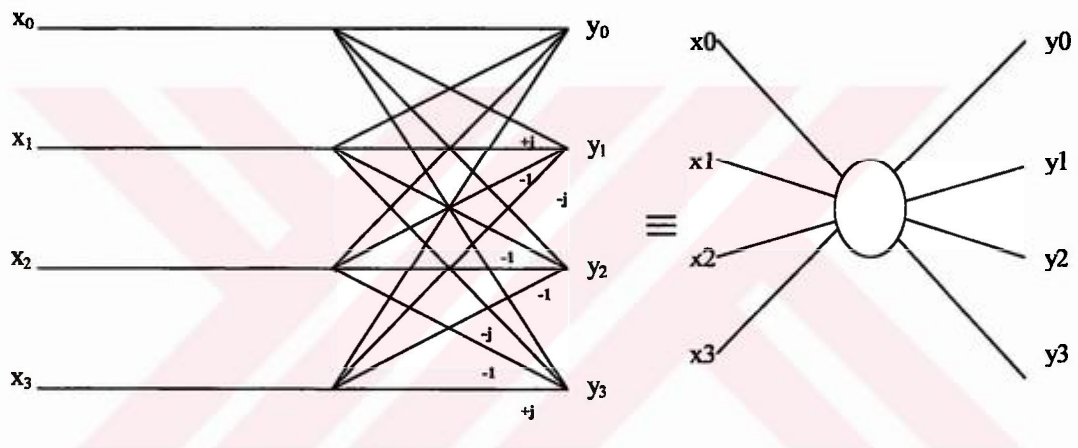
**Figure 2.4** Spectra of the individual subcarriers

The complex baseband OFDM signal as defined by (2.2) is in fact nothing more than the inverse Fourier transform of  $N_s$  QAM input symbols. The time discrete equivalent is the inverse discrete Fourier transform (IDFT), which is given by (4), where the time  $t$  is replaced by a sample number  $n$ . In practice, this transform can be implemented very efficiently by the inverse fast Fourier transform (IFFT). An  $N$  point IDFT requires a total of  $N^2$  complex multiplications—which are actually only phase rotations. Of course, there are also additions necessary to do an IDFT, but since the hardware complexity of an adder is significantly lower than that of a multiplier or phase rotator, only the multiplications are used here for comparison. The IFFT drastically reduces the amount of calculations by exploiting the regularity of the operations in the IFFT. Using the radix-2 algorithm, an  $N$ -point IFFT requires only  $(N/2) \cdot \log_2(N)$  complex multiplications. For a 16-point transform, for instance, the difference is 256 multiplications for the IDFT versus 32 for the IFFT—a reduction by a factor of 8!. This difference grows for larger numbers of subcarriers, as the IDFT complexity grows quadratically with  $N$ , while the IFFT complexity only grows slightly faster than linear.

$$s(n) = \sum_{i=0}^{N_s-1} d_i \exp(j2\pi(in/N)) \quad (2.4)$$



The number of multiplications in the IFFT can be reduced even further by using a radix-4 algorithm. This technique makes use of the fact that in a four-point IFFT, there are only multiplications by  $(1, -1, j, -j)$ , which actually do not need to be implemented by a full multiplier, but rather by a simple add or subtract and a switch of real and imaginary parts in the case of multiplications by  $j$  or  $-j$ . In the radix-4 algorithm, the transform is split into a number of these trivial four-point transforms, and non trivial multiplications only have to be performed between stages of these four point transforms. In this way, an  $N$ -point FFT using the radix-4 algorithm requires only  $(3/8)N(\log_2 N - 2)$  complex multiplications or phase rotations and  $N \log_2 N$  complex additions. For a 64-point FFT, for example, this means 96 rotations and 384 additions or 2.5 and 6 rotations and additions per sample, respectively.



**Figure 2.5** The radix-4 butterfly

Figure 2.5 shows the four-point IFFT, which is known as the radix-4 butterfly that forms the basis for constructing larger IFFT sizes. Four input values  $x_0$  to  $x_3$  are transformed into output values  $y_0$  to  $y_3$  by simple additions and trivial phase rotations. For instance,  $y_1$  is given by  $x_0 + jx_1 - x_2 - jx_3$ , which can be calculated by doing four additions plus a few additional I/Q swappings and inversions to account for the multiplications by  $j$  and  $-1$ .

The radix-4 butterfly can be used to efficiently build an IFFT with a larger size. For instance, a 16-point IFFT is depicted in figure 2.5. The 16-point IFFT contains two stages with four radix-4 butterflies, separated by an intermediate stage where the 16 intermediate results are phase rotated by the twiddle factor  $\omega^i$ , which

defined as  $\exp(j2\pi i/N)$ . Notice that for  $N=16$ , rotation by the twiddle factor  $\omega^i$  reduces to a trivial operation for  $i=0,4,8$  and  $12$ , where  $\omega^i$  is  $1, j, -1, -j$ , respectively. Taking this into account, the 16-point IFFT actually contains only eight non-trivial phase rotations, which is a factor of 32 smaller than the amount of phase rotations for the IDFT. These non-trivial phase rotations largely determine the implementation complexity, because the complexity of a phase rotation or complex multiplication is an order of magnitude larger than the complexity of an addition.

As an example of how to generate an OFDM symbol, let us assume that we want to transmit eight binary values  $\{1 \ 1 \ 1 \ -1 \ 1 \ 1 \ -1 \ 1\}$  on eight sub carriers. The IDFT or IFFT then has to calculate:

$$\frac{1}{8} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & (1/2)\sqrt{2}(1+j) & j & (1/2)\sqrt{2}(-1+j) & -1 & (1/2)\sqrt{2}(-1-j) & -j & (1/2)\sqrt{2}(1-j) \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & (1/2)\sqrt{2}(-1+j) & -j & (1/2)\sqrt{2}(1+j) & -1 & (1/2)\sqrt{2}(1-j) & j & (1/2)\sqrt{2}(-1-j) \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & (1/2)\sqrt{2}(-1-j) & j & (1/2)\sqrt{2}(1-j) & -1 & (1/2)\sqrt{2}(1+j) & -j & (1/2)\sqrt{2}(-1+j) \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & (1/2)\sqrt{2}(1-j) & -j & (1/2)\sqrt{2}(-1-j) & -1 & (1/2)\sqrt{2}(-1+j) & j & (1/2)\sqrt{2}(1+j) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

$$\equiv \frac{1}{8} \begin{bmatrix} 4 \\ \sqrt{2}(1+j(\sqrt{2}-1)) \\ 2+2j \\ -\sqrt{2}(1+j(\sqrt{2}+1)) \\ 0 \\ -\sqrt{2}(1-j(\sqrt{2}+1)) \\ 2-2j \\ \sqrt{2}(1-j(\sqrt{2}-1)) \end{bmatrix} \tag{2.5}$$

The left-hand side of (2.5) contains the IDFT matrix, where every column corresponds to a complex subcarrier with a normalized frequency ranging from  $-4$  to  $+3$ . The right-hand side of (2.5) gives the eight IFFT output samples that form one OFDM symbol. In practice, however, these samples are not enough to make a real OFDM signal. The reason is that there is no oversampling present, which would introduce intolerable aliasing if one would pass these samples through a digital-to-analog converter. To introduce oversampling, a number of zeros can be added to the input data. For instance, eight zeros could be added to the eight input samples of the previous example, after which a 16-point IFFT can be performed to get 16 output samples of a twice-oversampled OFDM signal. Notice that in the complex IFFT as in (2.5), the first half of the rows correspond to positive frequencies while the last part correspond to negative frequencies. Hence, if oversampling is used, the zeros should be added in the middle of the data vector rather than appending them at the end. This ensures the zero data values are mapped onto frequencies close to plus and minus half the sampling rate, while the nonzero data values are mapped onto the subcarriers around 0 Hz. For the data of the previous example, the oversampled input vector would become  $\{1 \ 1 \ 1 \ -1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ -1 \ 1\}$ .

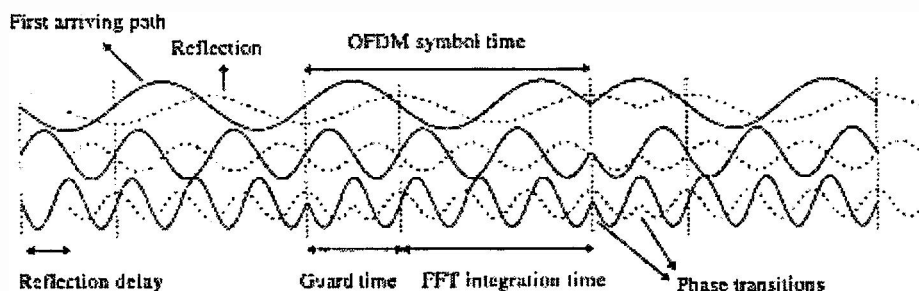
### 2.3 Guard Time and Cyclic Extension

One of the most important reasons to do OFDM is the efficient way it deals with multipath delay spread. By dividing the input datastream in  $N_s$  subcarriers, the symbol duration is made  $N_s$  times smaller, which also reduces the relative multipath delay spread, relative to the symbol time, by the same factor. To eliminate the intersymbol interference almost completely, a guard time is introduced for each OFDM symbol. The guard time is chosen larger than the expected delay spread, such that multipath components from one symbol cannot interfere with the next symbol. The guard time could consist of no signal at all. In that case, however, the problem of intercarrier interference (ICI) would arise. ICI is crosstalk between different subcarriers, which means they are no longer orthogonal.

To eliminate ICI, the OFDM symbol is cyclically extended in the guard time. This ensures that delayed replicas of the OFDM symbol always have an integer



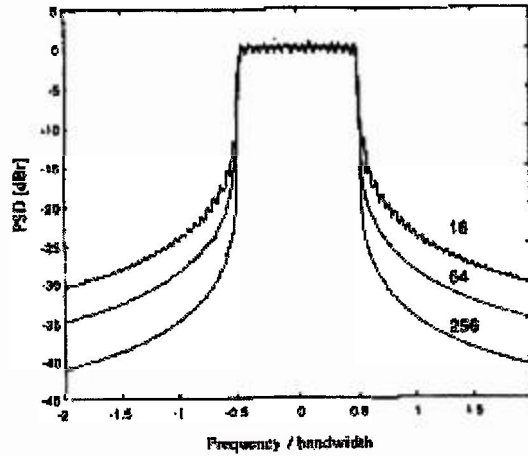
number of cycles within the FFT interval, as long as the delay is smaller than the guard time. As a result, multipath signals with delays smaller than the guard time cannot cause ICI.



**Figure 2.6** Example of an OFDM symbol with three subcarriers in a two ray multipath channel.

## 2.4 Windowing

Looking at an example OFDM signal like in figure 2.6, sharp phase transitions caused by the modulation can be seen at the symbol boundaries. Essentially, an OFDM signal like the one depicted in Figure 2.6 consists of a number of unfiltered QAM subcarriers. As a result, the out-of-band spectrum decreases rather slowly, according to a sine function. As an example of this, the spectra for 16, 64, and 256 subcarriers are plotted in Figure 2.7. For larger number of subcarriers, the spectrum goes down more rapidly in the beginning, which is caused by fact that the sidelobes are closer together. However, even the spectrum for 256 subcarriers has a relatively large -40dB bandwidth that is almost four times the -3dB bandwidth.



**Figure 2.7** Power spectral density without windowing for 16, 64, and 256 subcarriers.

To make the spectrum go down more rapidly, windowing can be applied to the individual OFDM symbols. Windowing an OFDM symbol makes the amplitude go smoothly to the zero at the symbol boundaries. A commonly used window type is the raised cosine window, which is defined as

$$w(t) = \begin{cases} 0.5 + 0.5 \cos(\pi + t\pi/(\beta T_s)) & 0 \leq t \leq \beta T_s \\ 1.0 & \beta T_s \leq t \leq T_s \\ 0.5 + 0.5 \cos((t - T_s)\pi/(\beta T_s)) & T_s \leq t \leq (1 + \beta)T_s \end{cases} \quad (2.6)$$

Here,  $T_s$  is the symbol interval, which is shorter than the total symbol duration because we allow adjacent symbols to partially overlap in the roll-off region. The time structure of the OFDM signal now looks like Figure 2.8.

In equation form, an OFDM symbol starting at time  $t=t_s=kT_s$  is defined as

$$\begin{aligned}
 s_k(t) &= \text{Re} \left\{ w(t-T_s) \sum_{i=-(N_s/2)}^{(N_s/2)-1} d_{i+N_s(k+1/2)} \exp(j2\pi(fc-(i+0.5)/T)(t-t_s-T_{\text{prefix}})) \right\} \\
 &, t_s \leq t \leq t_s + T_s(1+\beta) \\
 s_k(t) &= 0 \\
 &, t < t_s \quad \wedge \quad t > t_s + T_s(1+\beta)
 \end{aligned} \tag{2.7}$$

In practice, the OFDM signal is generated as follows: first,  $N_c$  input QAM values are padded with zeros to get  $N$  input samples that are used to calculate an IFFT. Then, the last  $T_{\text{prefix}}$  samples of the IFFT output are inserted at the start of the OFDM symbol, and the first  $T_{\text{postfix}}$  samples are appended at the end. The OFDM symbol is then multiplied by a raised cosine window  $w(t)$  to more quickly reduce the power of out-of-band subcarriers. The OFDM symbol with a delay of  $T_s$ , such there is an overlap region of  $\beta T_s$ , where  $\beta$  is the rolloff factor of the raised cosine window.

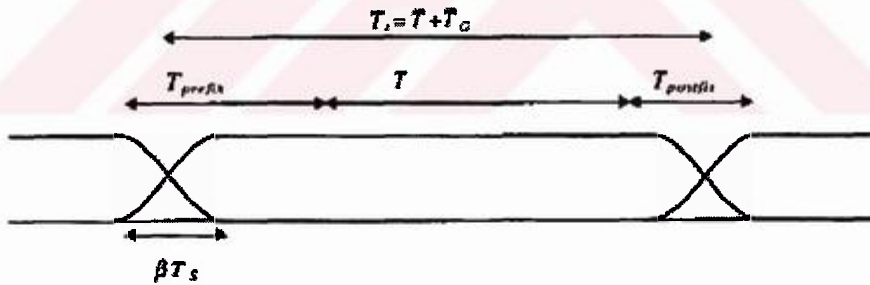
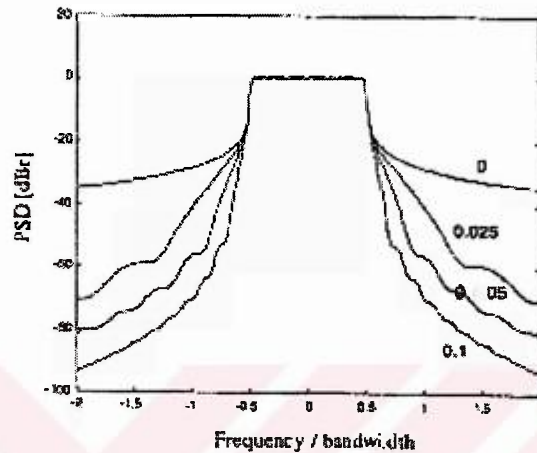


Figure 2.8 OFDM cyclic extension and windowing.

Figure 2.9 shows spectra for 64 subcarriers and different values of the rolloff factor  $\beta$ . It can be seen that a rolloff factor of 0.0025—so the rolloff region is only 2.5% of the spectrum interval—already makes a large improvement in the out-of-band spectrum. For instance, the -40dB bandwidth is more than halved to about twice the -3dB bandwidth. Larger rolloff factors improve the spectrum further, at the cost, however, of a decreased delay spread tolerance. The latter effect is demonstrated in Figure 2.9, which shows the signal structure of an OFDM signal for a two-ray multipath channel. The receiver demodulates the subcarriers by taking an FFT over

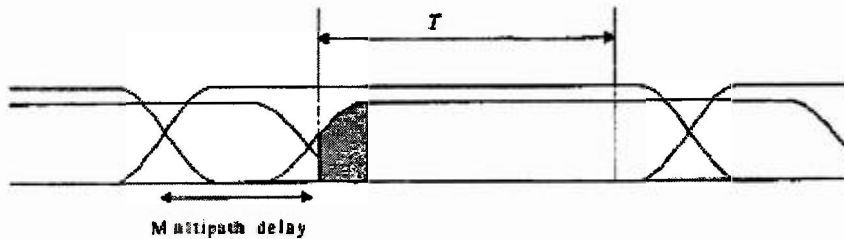
the  $T$ -second interval between dotted lines. Although the relative delay between the two multipath signals is smaller than the guard time, ICI and ISI are introduced because of the amplitude modulation in the gray part of the delayed OFDM symbol. The orthogonality between subcarriers as proved by (2.3) only holds when amplitude and phase of the subcarriers are constant during the entire  $T$ -second interval. Hence, a rolloff factor of  $\beta$  reduces the effective guard time by  $\beta T_s$ .



**Figure 2.9** Spectra for raised cosine windowing with rolloff factors of 0, 0.025, 0.05, and 0.1.

Instead of windowing, it is also possible to use conventional filtering techniques to reduce the out-of-band spectrum. Windowing and filtering are dual techniques; multiplying an OFDM symbol by a window means the spectrum is going to be a convolution of the spectrum of the window function with a set of impulses at the subcarrier frequencies. When filtering is applied, a convolution is done in the time domain and the OFDM spectrum is multiplied by the frequency response of the filter. When using filters, care has to be taken not to introduce rippling effects on the envelope of the OFDM symbols over a timespan that is larger than the rolloff region of the windowing approach. Too much rippling means the undistorted part of the OFDM envelope is smaller, and this directly translates into less delay spread tolerance. Notice that digital filtering techniques are complex to implement than windowing. A digital filter requires at least a few multiplications per sample, while windowing only requires a few multiplications per symbol, for those samples which fall into the rolloff region. Hence, because only a few percent of the samples are in

the rolloff region, windowing is an order of magnitude less complex than digital filtering.



**Figure 2.10** OFDM symbol windows for a two ray multipath channel, showing ICI and ISI because in the gray part, the amplitude of the delayed subcarriers is not constant.

## 2.5 Choice of OFDM Parameters

The choice of various OFDM parameters is a tradeoff between various, often conflicting requirements. Usually, there are three main requirements to start with: bandwidth, bit rate, and delay spread. The delay spread directly dictates the guard time. As a rule, the guard time should be about two to four times the root-mean-squared delay spread. This value depends on the type of coding and QAM modulation. Higher order QAM (like 64-QAM) is more sensitive to ICI and ISI than QPSK; while heavier coding obviously reduces the sensitivity to such interference.

Now that the guard time has been set, the symbol duration can be fixed. To minimize the signal-to-noise ratio (SNR) loss caused by the guard time, it is desirable to have the symbol duration much larger than the guard time. It cannot be arbitrarily large, however, because a larger symbol duration means more subcarriers with a smaller subcarrier spacing, a larger implementation complexity, and more sensitivity to phase noise and frequency offset, as well as an increased peak-to-average power ratio. Hence, a practical design choice is to make the symbol duration at least five times the guard time, which implies a 1dB SNR loss because of the guard time.

After the symbol duration and guard time are fixed, the number of subcarriers follows directly as the required -3dB bandwidth divided by the subcarrier spacing, which is the inverse of the symbol duration less the guard time. Alternatively, the

number of subcarriers may be determined by the required bit rate divided by the bit rate per subcarrier. The bit rate per subcarrier is defined by the modulation type, coding rate, and symbol rate.

## **2.6 Need For Coding**

In almost all applications of multi-carrier modulation, satisfactory performance cannot be achieved without the addition of some form of coding. In wireless systems subjected to fading, extremely high signal-to-noise ratios are required to achieve reasonable error probability. In addition, interference from other wireless channels is frequently severe. On wireline systems, large constellation sizes are commonly employed to achieve high bit rates. Coding in this case is essential for achieving the highest possible rates in the presence of crosstalk and impulsive and other interference.

Proper coding design is extremely important for a digital communication link. A designer should take several design factors into account. Those include required coding gain for intended link budget, channel characteristics, source coding requirements, modulation, etc. Coding in OFDM systems has an additional dimension. It can be implemented in time and frequency domain such that both dimensions are utilised to achieve better immunity against frequency and time selective fading.

### **2.6.1 Block Coding in OFDM**

In classical block coding, input data are blocked into groups of  $k$  bits, and each block is mapped into an output block of  $n$  bits, where  $n > k$ . In the canonical form,  $n-k$  parity checks are computed among the input bits according to some algebraic procedure, and then appended to the original block. This requires an increase in bandwidth by a factor of  $n/k$ . The reciprocal of this factor is the efficiency, or rate, of the code.



Only  $2^k$  of the possible  $2^n$  output blocks are legitimate code words. The code is chosen such that the minimum “Hamming distance”, which is the number of bits in which code words differ, is maximized. The code is described by the set of numbers  $[n,k,d]$ , where  $d$  is the minimum Hamming distance.

At the receiver, the  $n$ -bit block is recovered, possibly with errors, by demodulation and framing. The decoder finds the permissible code word that is closest Hamming distance to this received block. The  $n-k$  check bits may then be deleted and the result output as a replica of the original input. If  $d = 2t+1$ , then any set of  $t$  or fewer errors in the block can be corrected.

### 2.6.2 Convolutional Coding

Another very important form of coding is convolutional. Instead of operating on symbols arranged as blocks, the coding operates continuously on streams of symbols. At the encoder, the input is fed continually through a shift register of length  $m$ . The memory of the code is the “constraint length  $(m+1)$ , the number of output symbols affected by an input symbol. Each time a bit is read into the register, several modulo-2 sums of the present and past bits are formed. The choice of which bits are operated on is designated as a polynomial  $P(z)$  with binary coefficients.  $n$  such modulo-2 sums are formed and multiplexed to form the output of the “mother code”. Since  $n$  bits are generated for each input bit, the rate of the code is  $1/n$ .

### 2.6.3 Concatenated Coding

Combining convolutional and block codes in a concatenated code is particularly powerful technique. The block code is the outer code, that is it is applied first at the transmitter and last at the receiver. The inner convolutional code is very effective at reducing the error probability. However when a convolutional code does make an error, it appears as a large burst. This occurs when the Viterbi algorithm chooses a wrong sequence. The outer block code, especially an interleaved Reed-Solomon code is very effective in correcting that burst error.

## **2.7 Synchronization**

One of the arguments against OFDM is that it is highly sensitive to synchronization errors, in particular, to frequency errors.

### **2.7.1 Symbol synchronization**

#### **2.7.1.1 Timing errors**

A great deal of attention is given to symbol synchronization in OFDM systems. However, by using a cyclic prefix, the timing requirements are relaxed somewhat. The objective is to know when the symbol starts. A timing offset gives rise to a phase rotation of subcarriers. This phase rotation is the largest on the edges of the frequency band. If a timing error is small enough to keep the channel impulse response within the cyclic prefix, the orthogonality is maintained. In this case a symbol timing delay can be viewed as a phase shift introduced by the channel, and the phase rotations can be estimated by a channel estimator. If a time shift is larger than the cyclic prefix, ISI will occur.

There are two main methods for timing synchronizations: based on pilots or on the cyclic prefix. An algorithm of the former kind was suggested by Warner and Leung. They use a scheme where the OFDM signal is transmitted by frequency modulation(FM). The transmitter encodes a number of reserved subchannels with known phases and amplitudes. The synchronization technique, with modifications, is applicable to OFDM signals transmitted by amplitude modulation. Their algorithm consists of 3 phases: power detection, coarse synchronization and fine synchronization.

The first phase(power detection) detects whether or not an OFDM signal is present by measuring the received power and compare it to a threshold. The second phase(coarse synchronization) is used to acquire synchronization alignment to within  $\pm 0.5$  samples. This performance is not acceptable, but this phase serves to simplify the tracking algorithm(which can assume that the timing error is small). The coarse synchronization is done by correlating the received signal to a copy of the transmitted



synchronization signal. To find the peak of this correlation with enough accuracy, a digital filter is used to provide interpolated data values at four times the original data rate. In the last phase (fine synchronization) of the synchronization, the subchannels with pilots are equalized with the estimated channel obtained from pilots. Since the coarse synchronization guarantees that the timing error is less than  $\pm 0.5$ , the channel impulse response is within the cyclic prefix. The remaining phase errors on the pilot subchannels are due to timing error and can be estimated by linear regression.

### 2.7.1.2 Carrier phase noise

Carrier phase noise is caused by imperfections in the transmitter and receiver oscillators. For a frequency-selective channel, no distinction can be made between the phase rotation introduced by a timing error and a carrier phase offset. Carrier phase noise can be modeled as a Wiener process  $\theta(t)$  with  $E\{\theta(t)\}=0$  and  $E\{(\theta(t_0 + t) - \theta(t_0))^2\} = 4\pi\beta(t)$ , where  $\beta$  (in Hz) denotes the one-sided 3dB linewidth of the Lorentzian power density spectrum of the free-running carrier generator. The degradation in SNR, i.e., the increase in SNR need to be compensate for the error, can be approximated by

$$D(\text{dB}) \cong (11/6 \ln 10)(4\pi N\beta/W)(E_s/N_0),$$

(2.8)

where  $W$  is the bandwidth and  $E_s/N_0$  is the per-symbol SNR. Note that the degradation increases with the number of subcarriers. Due to the rapid variations of the phase noise, it may cause large problems.

### 2.7.2 Sampling-frequency synchronization

The received continuous-time signal is sampled at instants determined by the receiver clock. There two types of methods of dealing with the mismatch in sampling frequency. In synchronized-sampling systems a timing algorithm controls a voltage controlled crystal oscillator in order to allign the receiver clock with the transmitter

clock. The other method is non-synchronized sampling where the sampling rate remains fixed, which requires post-processing in the digital domain. The effect of a clock frequency offset is twofold: the useful signal component is rotated and attenuated and, in addition, ICI is introduced. Non-synchronized sampling systems are much more sensitive to a frequency offset, compared with a synchronized sampling system.

### 2.7.3 Carrier Frequency Synchronization

#### 2.7.3.1 Frequency Errors

Frequency offsets are created by differences in oscillators in transmitter and receiver, Doppler shifts, or phase noise introduced by non-linear channels. There are two destructive effects caused by a carrier frequency offset in OFDM systems. One is the reduction of signal amplitude (the sine functions are shifted and no longer sampled at the peak) and the other is the introduction of ICI from the other carriers. The latter is caused by the loss of orthogonality between subchannels. Denote the relative frequency offset, normalized by the subcarrier spacing, by  $\Delta f = \Delta F / (W/N)$  where  $\Delta F$  is the frequency offset and  $N$  the number of subcarriers. The degradation  $D$  in SNR (in dB) can then be approximated by

$$D(\text{dB}) \cong (10/3 \ln 10) (\pi \Delta f)^2 (E_s/N_0) = (10/3 \ln 10) (\pi N \Delta F / W)^2 (E_s/N_0). \quad (2.9)$$

Note that the degradation (in dB) increases with the square of the number of subcarriers, if  $\Delta F$  and  $W$  are fixed.

#### 2.7.3.2 Frequency Estimators

Several carrier synchronization schemes have been suggested in literature. As with symbol synchronization, they can be divided into two categories: based on pilots or cyclic prefix.

In pilot-aided algorithms some subcarriers are used for the transmission of pilots (usually a pseudo-noise (PN) sequence). Using these known symbols, the phase rotations caused by the frequency offset can be estimated. Under the assumption that the frequency offset is less than half the subcarrier spacing, there is a one-to-one correspondence between the phase rotations and the frequency offset. To assure this, an acquisition algorithm must be applied.

A related technique is to use cyclic prefix. The redundancy of the cyclic prefix can be used in several ways: e.g., by creating a function that peaks at zero offset and finding its maximizing value or by doing maximum likelihood estimation.

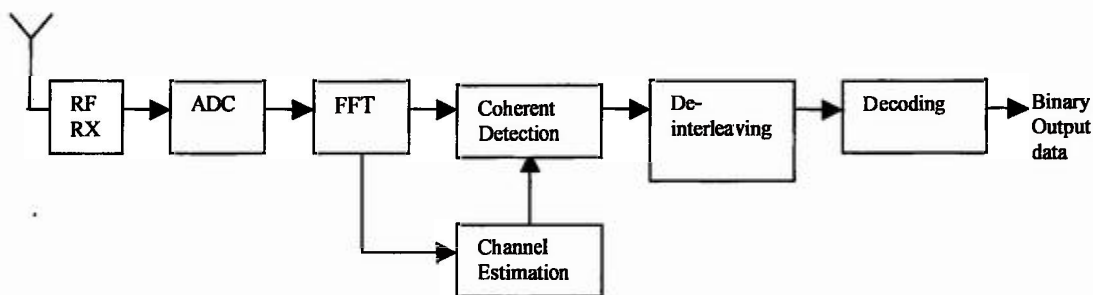
## **2.8 Detection**

In an OFDM link, the data bits are modulated on the subcarriers by some form of phase shift keying (PSK) or quadrature amplitude modulation (QAM). To estimate the bits at the receiver, knowledge is required about the reference phase and amplitude of the constellation on each subcarrier. In general, the constellation of each subcarrier shows a random phase shift and amplitude change, caused by carrier frequency offset, timing offset, and frequency selective fading. To cope with these unknown phase and amplitude variations, two different approaches exist. The first one is coherent detection, which uses estimates of the reference amplitudes and phases to determine the best possible decision boundaries for the constellation of each subcarrier. The main issue with coherent detection is how to find the reference values without introducing too much training overhead. The second approach is differential detection, which does not use absolute reference values, but only looks at the phase and/or amplitude differences between two consecutive symbols. Differential detection can be done both in the time domain or in the frequency domain.

### **2.8.1 Coherent Detection**

Figure 2.11 shows a block diagram of a coherent OFDM receiver. After downconversion and analog-to-digital conversion, the FFT is used to demodulate the

N subcarriers of the OFDM signal. For each symbol, the FFT output contains N values. However, these values contain random phase shifts and amplitude variations caused by the channel response, local oscillator drift, and timing offset. It is the task of the channel estimation block to learn the reference phases and amplitudes for all subcarriers.



**Figure 2.11** Block diagram of an OFDM receiver with coherent detection

In general, radio channels are fading both in time and in frequency. Hence a channel estimator has to estimate time-varying amplitudes and phases of all subcarriers. One way to do this is to use a two-dimensional channel estimator that estimates the reference values based on a few known pilot values. Based on these pilots, all other reference values can be estimated by performing a two dimensional interpolation.

To be able to interpolate the channel estimates both in time and in frequency from the available pilots, the pilot spacing has to fulfill the Nyquist sampling theorem, which states that the sampling interval must be smaller than the inverse of the double-sided bandwidth of the sampled signal. For the case of OFDM, this means that there exist both a minimum subcarrier spacing and a minimum symbol spacing between pilots. By choosing the pilot spacing much smaller than these minimum requirements, a good channel estimation can be made with a relatively easy algorithm. The more pilots are used, however, the smaller the effective SNR becomes that is available for data symbols. Hence, the pilot density is a tradeoff between channel estimation performance and SNR loss.

## 2.9 Applications of OFDM

### 2.9.1 Digital Audio Broadcasting

DAB is the successor of current analog audio broadcasting based on AM and FM. DAB offers improved sound quality, comparable to that of a compact disc, new data services, and a higher spectrum efficiency. DAB was standardized in 1995 by the European Telecommunications Standards Institute (ETSI) as the first standard to use OFDM. The basis for this standard was the specification developed by the European Eureka 147 DAB project, which started in 1988.

DAB has four transmission modes using different sets of OFDM parameters which are listed in Table. The parameters for modes I to III are optimized for use in specific frequency bands, while mode IV was introduced to provide a better coverage range at the cost of an increased vulnerability to Doppler shift.

**Table 2.1** DAB OFDM Parameters

	Mode I	Mode II	Mode III	Mode IV
Number of subcarriers	1,536	384	192	768
Subcarrier spacing	1 kHz	4 kHz	8 kHz	2 kHz
Symbol time	1.246 ms	311.5 $\mu$ s	155.8 $\mu$ s	623 $\mu$ s
Guard time	246 $\mu$ s	61.5 $\mu$ s	30.8 $\mu$ s	123 $\mu$ s
Carrier frequency	<375 MHz	<1.5 GHz	<3 GHz	<1.5 GHz
Transmitter separation	<96 km	<24 km	<12 km	<48 km

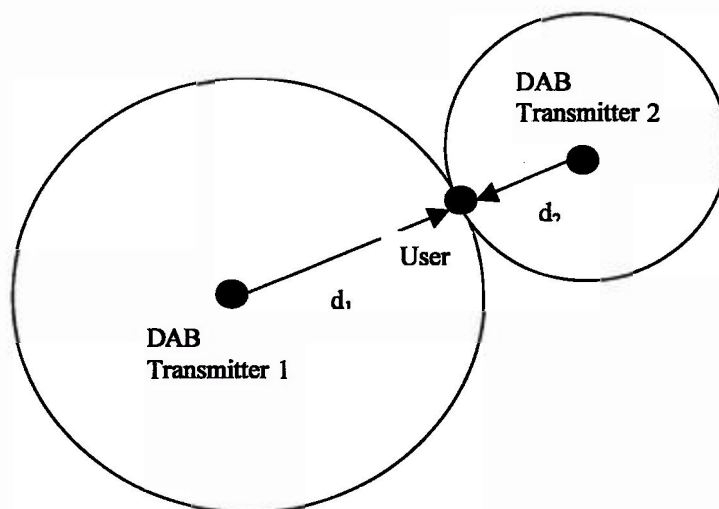
One important reason to use OFDM for DAB is the possibility to use a single frequency network, which greatly enhances the spectrum efficiency. In a single frequency network, a user receives the same signal from the transmitters simultaneously. Because of the propagation differences among transmitters, there is some delay between the arrival of the signals. This is illustrated in Figure 2.12, where two DAB signals arrive at the user with a delay difference that is equal to the distance difference ( $d_1 - d_2$ ) divided by the speed of light. Basically, to the user this situation is equivalent to a two-ray multipath channel. Hence as long as the propagation differences between the two signals are smaller than the guard time of



the OFDM symbols, no ISI or ICI will occur. The addition of the two time-shifted signals creates a diversity advantage for the user; the probability that the sum of both signals has an unacceptably low power because of shadowing or flat fading is much lower than the probability that one of the individual signals is too weak.

The DAB transmitted data consists of a number of audio signals, sampled at 48 kHz with an input resolution up to 22 bits. The digital audio signal is compressed to a rate in the range of 32 to 384 kbps, depending on the desired quality. The signal is divided into frames of 24 ms. The start of a frame is indicated by a null symbol, which is a silence period that is slightly larger than the duration of a normal OFDM symbol. Then, a reference OFDM symbol is sent which serves as the starting point for the differential decoding of the QPSK-modulated subcarriers. Differential encoding is applied in the time domain, so in the receiver, the phase of each subcarrier is compared with the phase of the same subcarriers from the previous OFDM symbol.

The digital input data is encoded by a rate  $\frac{1}{4}$  convolutional code with constraint length 7 to provide protection against fading subcarriers. The coding rate can be increased up to  $\frac{8}{9}$  puncturing. This gives a maximum total data rate of  $1,536 \times 2 \times \frac{8}{9} \times 1/1.246 \times 10^3$ , which is approximately 2.2 Mbps. The coded data are interleaved to separate coded bits in the frequency domain as much as possible, which avoids large error bursts in case of a deep fade affecting a group of subcarriers.

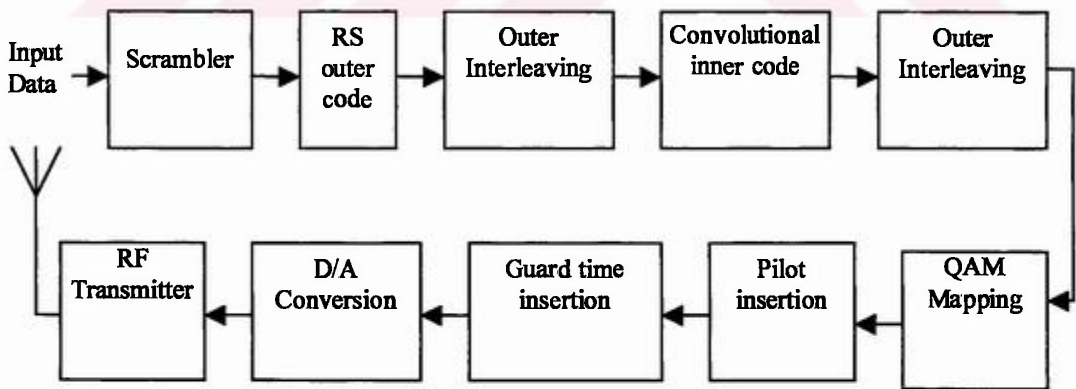


**Figure 2.12** User receiving two DAB transmitters

## 2.9.2 Terrestrial Digital Video Broadcasting

Research on a digital system for television broadcasting has been carried out since the late 1980s. In 1993, a pan-broadcasting-industry group started the Digital Video Broadcasting (DVB) project. Within this project, a set of specifications was developed for the delivery of digital television over satellites, cable, and through terrestrial transmitters.

Terrestrial DVB uses OFDM with two possible modes, using 1,705 and 6,817 subcarriers, respectively. These modes are referred to as 2k and 8k modes, respectively, as these are the sizes of the FFT/IFFT needed to generate and demodulate all subcarriers. The main reason to have two modes were doubts about the implementability of the 8k subcarrier system. Basically, the 2k system is a simplified version which requires an FFT/IFFT that is only a quarter of the size that is needed for the 8k system. Because the guard time is also four times smaller, the 2k system can handle less delay spread and less propagation delay differences among transmitters within a single-frequency network. The FFT interval duration for the 8k system is 896  $\mu$ s, while the guard time can have four different values from 28 to 224  $\mu$ s. The corresponding values for the 2k system are four times smaller.



**Figure 2.13** Block diagram of a DVB-T transmitter

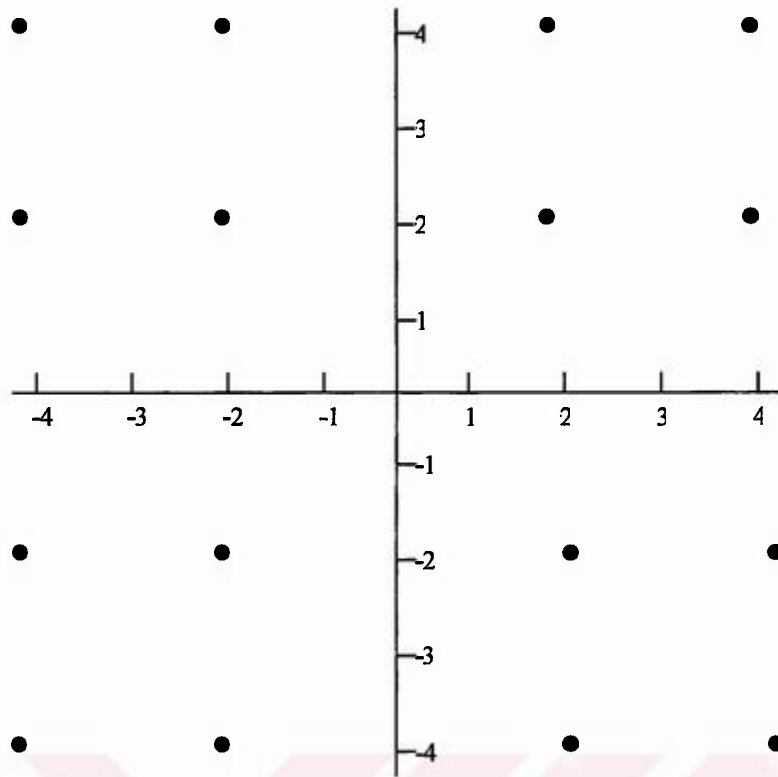
Figure 2.13 shows a block diagram of a DVB-T transmitter. The input data are divided into groups of 188 bytes, which are scrambled and coded by an outer shortened Reed-Solomon code (204,188,t=8). This can correct up to eight erroneous bytes in a frame of 204 bytes. The coded bits are interleaved by a convolutional interleaver that

interleaves byte-wise with a depth of 12 bytes and then again coded by a rate  $\frac{1}{2}$  constraint length 7 convolutional code with generator polynomials (171,133 octal). The rate of this latter code can be increased by puncturing to  $\frac{2}{3}$ ,  $\frac{3}{4}$ ,  $\frac{5}{6}$ , or  $\frac{7}{8}$ . The convolutionally encoded bits are interleaved by an inner interleaver and then mapped onto QPSK, 16-QAM, or 64-QAM symbols.

For 16-QAM and 64-QAM, optional hierarchical coding can be applied. In this case, the constellation points are moved farther away from the origin so the quadrants can be detected more reliably than the position within each quadrant. This is illustrated in Figure 2.14 for a hierarchical 16-QAM constellation. For this constellation, the minimum distance between points from different quadrants is double the distance of a normal 16-QAM constellation, where the constellation points would have values of 1 and 3 instead of 2 and 4. Corrected for the increased power of the constellation, the detection of the quadrants has a 3-dB SNR advantage over the detection of points in a normal 16-QAM constellation. The advantage of this hierarchical coding is that users for which the SNR is just too low to decode all bits can at least decode the two most significant bits that determine the quadrant. These bits give them the same video signal, but a lower resolution.

To obtain reference amplitudes and phases to perform coherent QAM demodulation, pilot subcarriers are transmitted. For the 8k mode, in each symbol there are 768 pilots, so 6,048 subcarriers remain for data. The 2k mode has 192 pilots and 1,512 data subcarriers. The position of the pilot varies from symbol to symbol with a pattern that repeats after four OFDM symbols. The pilots allow a receiver to estimate the channel both in frequency as well as in time, which is important as for mobile receivers there can be significant channel changes within a few OFDM symbols.





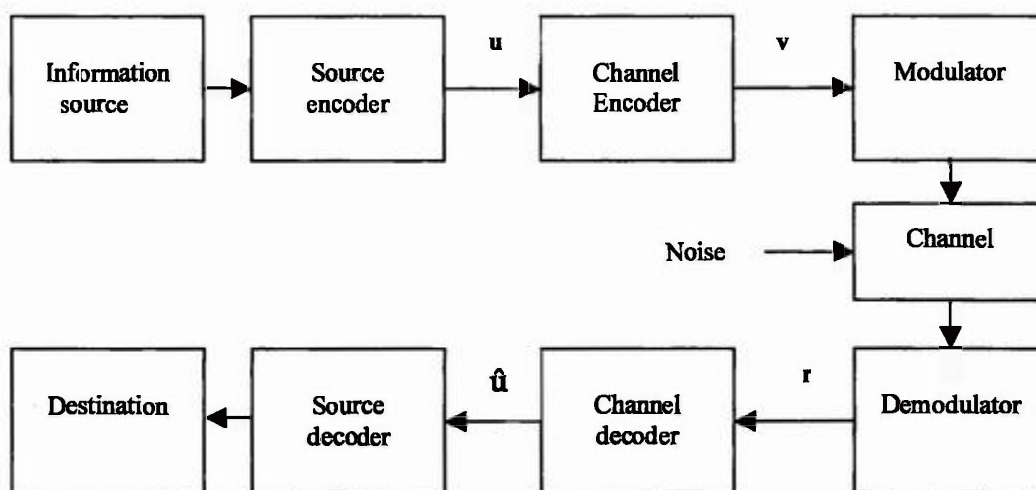
**Figure 2.14** Hierarchical 16-QAM

## 3 CODING

### 3.1 Introduction

In recent years, there has been an increasing demand for efficient and reliable digital data transmission and storage systems. This demand has been accelerated by the emergence of large-scale, high speed data networks for the exchange, processing, and storage of digital information in the military, governmental, and private spheres. A merging of communications and computer technology is required in the design of these systems. A major concern of the designer is the control of errors so that the reliable reproduction of data can be obtained.

In 1948, Shannon demonstrated in a landmark paper that, by proper encoding of the information, errors induced by a noisy channel or storage medium can be reduced to any desired level without sacrificing the rate of information transmission or storage.



**Figure 3.1** A typical transmission system

A typical transmission system may be represented by the block diagram shown in Figure 3.1. The information source can be either a person or a machine. The source output, which is communicated to the destination, can be either a continuous waveform or a sequence of discrete symbols.

The source encoder transforms the source output into a sequence of binary digits called the information sequence  $\mathbf{u}$ .

The channel encoder transforms the information sequence  $\mathbf{u}$  into a discrete encoded sequence  $\mathbf{v}$  called a code word.

Discrete symbols are not suitable for transmission over a physical channel or recording on a digital storage medium. The modulator transforms each output symbol of the channel encoder into a waveform of duration  $T$  seconds which is suitable for transmission. This waveform enters the channel and is corrupted by noise. Typical transmission channels include telephone lines, high-frequency radio links, telemetry links, microwave links, satellite links and so on. The demodulator processes each received waveform of duration  $T$  and produces an output that may be discrete(quantized) or continuous(unquantized).

The channel decoder transforms the received sequence  $\mathbf{r}$  into a binary sequence  $\hat{\mathbf{u}}$  called the estimated sequence. The decoding strategy is based on the rules of channel encoding and the noise characteristics of the channel. Ideally,  $\hat{\mathbf{u}}$  will be a replica of the information sequence  $\mathbf{u}$ , although the noise may cause some decoding errors.

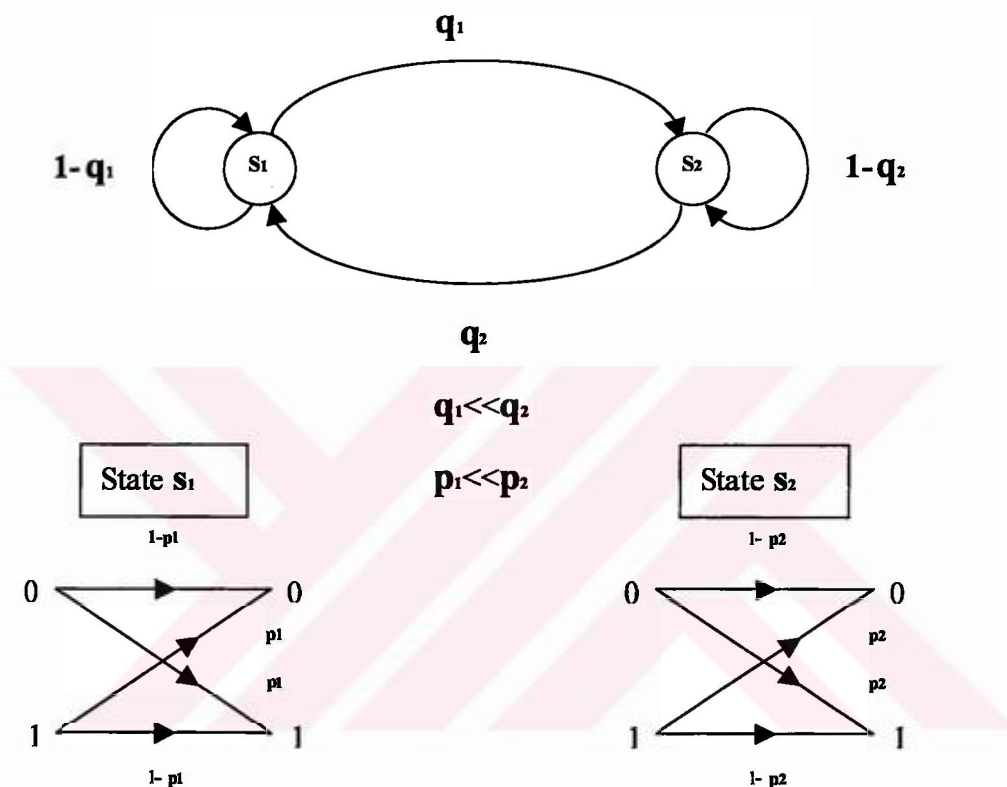
The source decoder transforms the estimated sequence  $\hat{\mathbf{u}}$  into an estimate of the source output and delivers this estimate to the destination.

### 3.2 Types of Errors

On memoryless channels, the noise affects each transmitted symbol independently. Hence transmission errors occur randomly in the received sequence,

and memoryless channels are called random-error channels. Good examples of random-error channels are the deep-space channel and many satellite channels. The codes devised for correcting random errors are called random-error-correcting codes.

On channels with memory, the noise not independent from transmission to transmission.



**Figure 3.2** Simplified model of a channel with memory

The above model contains two states, a “good state“ in which transmission errors occur infrequently, and a “bad state“ in which transmission errors are highly probable. The channel is good state most of the time, but on occasion shifts to the bad state due to a change in the transmission characteristics of the channel. As a consequence, transmission errors occur in clusters or bursts because of the high transition probability in the bad state, and channels with memory are called burst-error channels. The codes devised for correcting burst errors are called burst-error-correcting codes.

Finally, some channels contain a combination of both random and burst errors. These are called compound channels, and codes devised for correcting errors on these channels are called burst-and-random-error-correcting codes.

### 3.3 Error Control Strategies

If the transmission is strictly in one direction, from transmitter to receiver, we call this one-way system. Error control for a one way system must be accomplished using forward error correction (FEC), that is, by employing error correcting codes that automatically correct errors detected at the receiver.

In some cases, like telephone channels and some satellite communication systems, a transmission system can be two way. Error control for a two-way system can be accomplished using error detection and retransmission, called automatic repeat request (ARQ). In an ARQ system, when errors are detected at the receiver, a request is sent for the transmitter to repeat the message, and this continues until the message is received correctly.

There are two types of ARQ systems: stop-and-wait ARQ and continuous ARQ. With stop-and-wait ARQ, the transmitter sends a code word to the receiver and waits for a positive (ACK) or negative (NAK) acknowledgement from the receiver. If ACK is received (no errors are detected), the transmitter sends the next code word. If NAK is received (errors detected), it resends the preceding code word. When the noise is persistent, the same code word may be retransmitted several times before it is correctly received and acknowledged.

With continuous ARQ, the transmitter sends code words to the receiver continuously and receives acknowledgements continuously. When a NAK is received, the transmitter begins a retransmission. It may back up to the code word in error and resend that word plus the words follow it. This is called go-back-N ARQ. Alternatively, the transmitter may simply resend only those code words that are acknowledged negatively. This is known as selective-repeat ARQ. Selective-repeat ARQ is more efficient than go-back-N ARQ, but requires more logic and buffering.

Continuous ARQ is more efficient than stop-and-wait ARQ, but it is also more expensive. Stop-and-wait ARQ is designed for use on half-duplex channels, whereas continuous ARQ is designed for use on full-duplex channels.

The major advantage of ARQ over FEC is that error detection requires much simpler decoding equipment than does error correction. Also, ARQ is adaptive in the sense that information is retransmitted only when error occur. On the other hand, when the channel rate is high, retransmissions must be sent too frequently, and the system throughput, the rate at which newly generated messages are correctly received, is lowered by ARQ. In this situation, a combination of FEC for the most frequent error patterns, together with error detection and retransmission for the less likely error patterns, is more efficient than ARQ alone.

### 3.4 Types of Codes

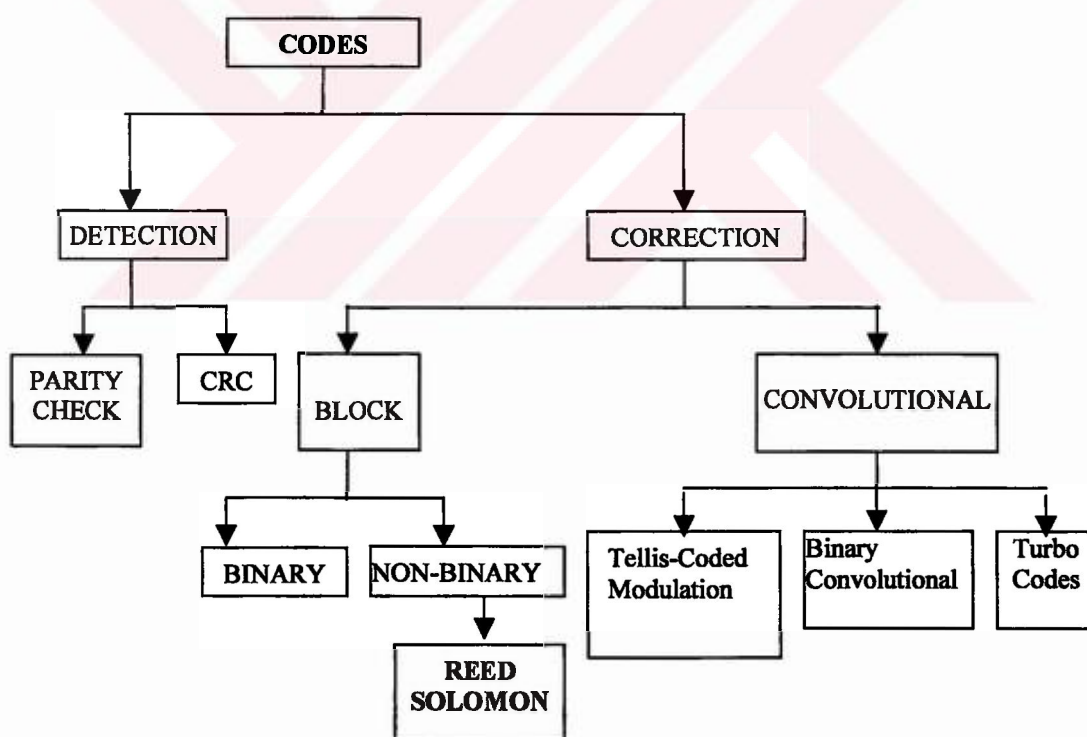


Figure 3.3 Taxonomy of some code classes

There are two different types of codes in common use today, block codes and convolutional codes.



Block codes are the most popular type of error correction codes, and a rich selection falls under this classification. The codes get their name because the encoder takes in a message block of finite length, adds redundancy, and sends out a code word that is also a block of longer length than the message. The most widely used type of block codes is Linear Block codes.

Convolutional codes use a completely different approach to coding. Instead of breaking the message into blocks, the entire message stream is converted into a single code word. Convolutional codes get their name because the encoding process can be viewed as convolving the message stream with the impulse response of the code.



## 4 LINEAR BLOCK CODES

We assume that the output of an information source is a sequence of binary digits “0” or “1”. In block coding, this binary information sequence is segmented into message blocks of fixed length; each message block, denoted by  $\mathbf{u}$ , consists of  $k$  information digits. There are a total of  $2^k$  distinct messages. The encoder, according to certain rules, transforms each input message  $\mathbf{u}$  into a binary  $n$ -tuple  $\mathbf{v}$  with  $n > k$ . This binary  $n$ -tuple  $\mathbf{v}$  is referred to as the code word of the message  $\mathbf{u}$ . Therefore, corresponding to the  $2^k$  possible messages, there are  $2^k$  code words. This set of  $2^k$  code words is called a block code.

**Definition** A block code of length  $n$  and  $2^k$  code words is called a linear  $(n, k)$  code if and if only its  $2^k$  code words form a  $k$ -dimensional subspace of the vector space of all the  $n$ -tuples over the field  $GF(2)$ .

In fact, a binary block code is linear if and only if the modulo-2 sum of two code word is also a code word. Since an  $(n, k)$  linear code  $C$  is a  $k$ -dimensional subspace of the vector space  $V_n$  of all the binary  $n$ -tuples, it is possible to find  $k$  linearly independent code words,  $\mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_{k-1}$  in  $C$  such that every code word  $\mathbf{v}$  in  $C$  is a linear combination of these  $k$  code words.

$$\mathbf{v} = u_0 \mathbf{g}_0 + u_1 \mathbf{g}_1 + \dots + u_{k-1} \mathbf{g}_{k-1} \quad u_i = 0 \text{ or } 1, \text{ for } 0 \leq i \leq k-1.$$

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G} = (u_0, u_1, \dots, u_{k-1}) \cdot \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix}$$

$$= u_0 \mathbf{g}_0 + u_1 \mathbf{g}_1 + \dots + u_{k-1} \mathbf{g}_{k-1}. \quad (4.1)$$

The rows of  $G$  generate the  $(n,k)$  linear code  $C$ . For this reason the matrix  $G$  is called a generator matrix for  $C$ .

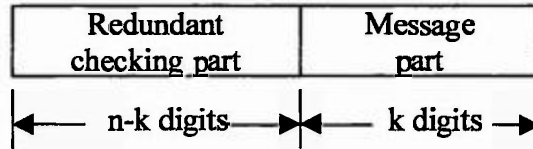
**TABLE 4.1:** Linear Block Code with  $k = 4, n=7$

Messages	Codewords
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

As an example the generator matrix of the block code given in Table 4.1 is as follows:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \mathbf{g}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (4.2)$$

A desirable property for a linear block code possesses is the systematic structure of the code words.



Systematic format of a code word.

The message part consists of k unaltered information (or message) digits and the redundant checking part consists of n-k parity-check digits, which are linear sums of the information digits. A linear block code with this structure is referred to as a linear systematic block code.

A linear systematic (n,k) code is completely specified by a k x n matrix G of the following form:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & \dots & p_{0,n-k-1} & | & 1 & 0 & 0 & 0 & \dots & 0 \\ p_{10} & p_{11} & \dots & p_{1,n-k-1} & | & 0 & 1 & 0 & 0 & \dots & 0 \\ p_{20} & p_{21} & \dots & p_{2,n-k-1} & | & 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & | & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \dots & p_{k-1,n-k-1} & | & 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \quad (4.3)$$

← P matrix →
← k x k identity matrix →

$$\begin{aligned}
 \mathbf{v} &= (v_0, v_1, \dots, v_{n-1}) \\
 &= (u_0, u_1, \dots, u_{k-1}) \cdot \mathbf{G}
 \end{aligned} \quad (4.4)$$

From the above equations:

$$\begin{aligned}
 v_{n-k+i} &= u_i && \text{for } 0 \leq i < k \\
 v_j &= u_0 p_{0j} + u_1 p_{1j} + \dots + u_{k-1} p_{k-1,j} && \text{for } 0 \leq j < n-k
 \end{aligned} \quad (4.5)$$

The n-k equations given above are called parity-check equations of the code.

For any  $k \times n$  matrix  $G$  with  $k$  linearly independent rows, there exists an  $(n-k) \times n$  matrix  $H$  with  $n-k$  linearly independent rows such that any vector in the row space of  $G$  is orthogonal to the rows of  $H$  and any vector that is orthogonal to the rows of  $H$  is in the row space of  $G$ . Hence, we can describe the  $(n,k)$  linear code generated by  $G$  in an alternate way as follows: An  $n$ -tuple  $\mathbf{v}$  is a code word in the code generated by  $G$  if and only if  $\mathbf{v} \cdot \mathbf{H}^T = \mathbf{0}$ . This matrix  $H$  is called a parity-check matrix of the code.

$$\mathbf{H} = [\mathbf{I}_{n-k} \mathbf{P}^T] = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & p_{00} & p_{01} & \dots & p_{k-1,0} \\ 0 & 1 & 0 & \dots & 0 & p_{01} & p_{11} & \dots & p_{k-1,1} \\ 0 & 0 & 1 & \dots & 0 & p_{02} & p_{21} & \dots & p_{k-1,2} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & p_{0, n-k-1} & p_{1, n-k-1} & \dots & p_{k-1, n-k-1} \end{bmatrix} \quad (4.6)$$

Let  $\mathbf{u} = (u_0, u_1, \dots, u_{k-1})$  be the message to be encoded. In systematic form the corresponding code word would be:  $\mathbf{v} = (v_0, v_1, \dots, v_{n-k-1}, u_0, u_1, \dots, u_{k-1})$ . Using the fact that  $\mathbf{v} \cdot \mathbf{H}^T = \mathbf{0}$ , we obtain

$$v_j + u_0 p_{0j} + u_1 p_{1j} + \dots + u_{k-1} p_{k-1,j} = 0 \quad \text{for } 0 \leq j < n-k \quad (4.7)$$

Rearranging the equations we obtain the same parity-check equations.

Consider an  $(n,k)$  linear code with generator matrix  $G$  and parity-check matrix  $H$ . Let  $\mathbf{v} = (v_0, v_1, \dots, v_{n-1})$  be a code word that was transmitted over a noisy channel. Let  $\mathbf{r} = (r_0, r_1, \dots, r_{n-1})$  be the received vector at the output of the channel. Because of the channel noise,  $\mathbf{r}$  may be different from  $\mathbf{v}$ . The vector sum

$$\mathbf{e} = \mathbf{r} - \mathbf{v} = (e_0, e_1, \dots, e_{n-1}) \quad (4.8)$$

This n-tuple is called the error vector. It follows from the above equation that the received vector  $\mathbf{r}$  is the vector sum of the transmitted code word and the error and the error vector, that is,

$$\mathbf{r} = \mathbf{v} + \mathbf{e} \quad (4.9)$$

When  $\mathbf{r}$  is received, the decoder computes the following (n-k)-tuple:

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = (s_0, s_1, \dots, s_{n-k-1}) \quad (4.10)$$

which is called the syndrome of  $\mathbf{r}$ . Then  $\mathbf{s} = \mathbf{0}$  if and only if  $\mathbf{r}$  is a code word.

The syndrome  $\mathbf{s}$  depends only on the error pattern  $\mathbf{e}$ :

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = (\mathbf{v} + \mathbf{e}) \cdot \mathbf{H}^T = \mathbf{v} \cdot \mathbf{H}^T + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{0} + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T \quad (4.11)$$

The relationship between the syndrome digits and the error digits is like below:

$$\begin{aligned} s_0 &= e_0 + e_{n-k} p_{00} + e_{n-k+1} p_{10} + \dots + e_{n-1} p_{k-1,0} \\ s_1 &= e_1 + e_{n-k} p_{01} + e_{n-k+1} p_{11} + \dots + e_{n-1} p_{k-1,1} \\ &\vdots \\ &\vdots \\ &\vdots \\ s_{n-k-1} &= e_{n-k-1} + e_{n-k} p_{0,n-k-1} + e_{n-k+1} p_{1,n-k-1} + \dots + e_{n-1} p_{k-1,n-k-1} \end{aligned} \quad (4.12)$$

The syndrome digits are simply linear combinations of the error digits. The above equations have  $2^k$  solutions. In other words, there are  $2^k$  error patterns that result in the same syndrome, and the true error pattern  $\mathbf{e}$  is just one of them. To minimize the probability of a decoding error, the most probable error pattern that satisfies the equations is chosen as the true error vector. If the channel is BSC, the most probable error pattern is the one that has the smallest number of nonzero digits. In other words which has the minimum Hamming weight.



The Hamming weight of  $\mathbf{v}$  denoted by  $w(\mathbf{v})$ , is defined as the number of nonzero components of  $\mathbf{v}$ . The Hamming distance between  $\mathbf{v}$  and  $\mathbf{w}$ , denoted by  $d(\mathbf{v}, \mathbf{w})$ , is defined as the number of places where they differ. The Hamming distance is a metric function that satisfies the triangle inequality. Let  $\mathbf{v}$ ,  $\mathbf{w}$ , and  $\mathbf{x}$  be three  $n$ -tuples. Then

$$d(\mathbf{v}, \mathbf{w}) + d(\mathbf{w}, \mathbf{x}) \geq d(\mathbf{v}, \mathbf{x}) \quad (4.13)$$

It follows from the definition of Hamming distance and the definition of modulo-2 addition that the Hamming distance between two  $n$ -tuples,  $\mathbf{v}$  and  $\mathbf{w}$ , is equal to the sum of  $\mathbf{v}$  and  $\mathbf{w}$ , that is,

$$d(\mathbf{v}, \mathbf{w}) = w(\mathbf{v} + \mathbf{w}) \quad (4.14)$$

Given a block code  $C$ , one can compute the Hamming distance between any two distinct code words. The minimum distance of  $C$ , denoted by  $d_{\min}$  is defined as

$$d_{\min} = \min\{d(\mathbf{v}, \mathbf{w}) : \mathbf{v}, \mathbf{w} \in C, \mathbf{v} \neq \mathbf{w}\} \quad (4.15)$$

**Theorem** The minimum distance of a linear block code is equal to the minimum weight of its nonzero code words.

The random-error-detecting capability of a block code with minimum distance  $d_{\min}$  is  $d_{\min}-1$ . There are  $2^k-1$  undetectable,  $2^n-2^k$  detectable error patterns. The parameter  $t = \lfloor (d_{\min} - 1)/2 \rfloor$  is called the random-error-correcting capability of the code where  $\lfloor (d_{\min} - 1)/2 \rfloor$  denotes the largest integer no greater than  $(d_{\min} - 1)/2$ . The code is referred to as  $t$ -error-correcting code.

#### 4.1 Hamming Codes

Hamming codes are the first class of linear codes devised for error correction. These codes and their variations have been widely used for error control in digital communication and data storage systems.

For any positive integer  $m \geq 3$ , there exists a Hamming code with the following parameters:

Code length:	$n = 2^m - 1$
Number of information symbols:	$k = 2^m - m - 1$
Number of parity-check symbols:	$n - k = m$
Error-correcting capability:	$t = 1 (d_{\min} = 3)$

## 4.2 BCH Codes

First we must give the definition of cyclic code: An  $(n, k)$  linear code  $C$  is called cyclic code if every shift of a code vector in  $C$  is also a code vector in  $C$ .

The Bose, Chaudhuri, and Hocquenghem (BCH) codes form a large class of powerful random error-correcting cyclic codes. For any positive integers  $m \geq 3$  and  $t (t < 2^m - 1)$  there exists a binary BCH code with the following parameters:

Block length:	$n = 2^m - 1$
Number of parity-check digits:	$n - k \leq mt$
Minimum distance:	$d_{\min} \geq 2t + 1$

This code is capable of correcting any combination of  $t$  or fewer errors. The generator polynomials of this code is specified in terms of its roots from the Galois field  $GF(2^m)$ .

The generator polynomial  $g(x)$  of the  $t$ -error-correcting BCH code of length  $2^m - 1$  is the lowest-degree polynomial over  $GF(2)$  which has  $\alpha, \alpha^2, \alpha^3, \dots, \alpha^{2t}$  as its roots [ $g(\alpha^i) = 0$  for  $1 \leq i \leq 2t$ ]. Let  $\phi_i(x)$  be the minimal polynomial of  $\alpha^i$ . Then  $g(x)$  must be the least common multiple of  $\phi_1(x), \phi_2(x), \dots, \phi_{2t}(x)$ , that is,

$$g(x) = \text{LCM}\{\phi_1(x), \phi_2(x), \dots, \phi_{2t}(x)\}$$

If  $i$  is an even integer, it can be expressed as a product of the following form:  $i=i'2^l$  ( $i'$  is an odd number and  $l \geq 1$ ). Then  $\alpha^i = (\alpha^{i'})^{2^l}$  is a conjugate of  $\alpha^{i'}$  and therefore  $\alpha^i$  and  $\alpha^{i'}$  have the same minimal polynomial, that is:  $\phi_i(x) = \phi_{i'}(x)$ . As a result  $g(x)$  can be reduced to:

$$g(x) = \text{LCM}\{\phi_1(x), \phi_3(x), \dots, \phi_{2^{l-1}}(x)\}. \quad (4.16)$$

Example 1: Let  $\alpha$  be a primitive element of the  $\text{GF}(2^4)$  given by Table 2 such that  $1+\alpha+\alpha^4=0$ . From Table 2 the minimal polynomials of  $\alpha, \alpha^3, \alpha^5$  are:

$$\phi_1(x) = 1+X+X^4$$

$$\phi_3(x) = 1+X+X^2+X^3+X^4$$

$$\phi_5(x) = 1+X+X^2$$

The double-error-correcting BCH code of length  $n = 2^4 - 1 = 15$  is generated by  $g(x) = \text{LCM}\{\phi_1(x), \phi_3(x)\}$ .

$$\begin{aligned} g(x) &= \phi_1(x)\phi_3(x) \\ &= (1+X+X^4)(1+X+X^2+X^3+X^4) \\ &= 1+X^4+X^6+X^7+X^8 \end{aligned}$$

For the triple-error-correcting BCH code:

$$\begin{aligned} g(x) &= \text{LCM}\{\phi_1(x), \phi_3(x), \phi_5(x)\} \\ &= (1+X+X^4)(1+X+X^2+X^3+X^4)(1+X+X^2) \\ &= 1+X+X^2+X^4+X^5+X^8+X^{10} \end{aligned}$$

**TABLE 4.2:** Minimal polynomials of the elements in  $GF(2^4)$  generated by  $p(x)=1+x+x^4$

Conjugate roots	Minimal polynomials
0	X
1	X+1
$\alpha, \alpha^2, \alpha^4, \alpha^8$	$X^4+X+1$
$\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$	$X^4+X^3+X^2+X+1$
$\alpha^5, \alpha^{10}$	$X^2+X+1$
$\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$	$X^4+X^3+X+1$

**TABLE 4.3:** Three representations for the elements of  $GF(2^4)$  generated by

Power representation	Polynomial representation	4-Tuple representation
0	0	(0 0 0 0)
1	1	(1 0 0 0)
$\alpha$	$\alpha$	(0 1 0 0)
$\alpha^2$	$\alpha^2$	(1 1 0 0)
$\alpha^3$	$\alpha^3$	(0 0 1 0)
$\alpha^4$	$1+\alpha$	(1 0 1 0)
$\alpha^5$	$\alpha + \alpha^2$	(0 1 1 0)
$\alpha^6$	$\alpha^2 + \alpha^3$	(1 1 1 0)
$\alpha^7$	$1+\alpha + \alpha^3$	(0 0 0 1)
$\alpha^8$	$1 + \alpha^2$	(1 0 0 1)
$\alpha^9$	$\alpha + \alpha^3$	(0 1 0 1)
$\alpha^{10}$	$1+\alpha + \alpha^2$	(1 1 0 1)
$\alpha^{11}$	$\alpha + \alpha^2 + \alpha^3$	(0 0 1 1)
$\alpha^{12}$	$1+\alpha + \alpha^2 + \alpha^3$	(1 0 1 1)
$\alpha^{13}$	$1+ \alpha^2 + \alpha^3$	(0 1 1 1)
$\alpha^{14}$	$1+ \alpha^3$	(1 1 1 1)

### 4.2.1 Decoding of BCH Codes

For decoding at-error-correcting primitive BCH code, the syndrome is  $2t$ -tuple,

$$\mathbf{S} = (S_1, S_2, \dots, S_{2t}) = \mathbf{r} \cdot \mathbf{H}^T$$

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1} \\ 1 & (\alpha^3) & (\alpha^3)^2 & (\alpha^3)^3 & \dots & (\alpha^3)^{n-1} \\ 1 & (\alpha^5) & (\alpha^5)^2 & (\alpha^5)^3 & \dots & (\alpha^5)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & (\alpha^{2t-1}) & (\alpha^{2t-1})^2 & (\alpha^{2t-1})^3 & \dots & (\alpha^{2t-1})^{n-1} \end{bmatrix}$$

$$S_i = \mathbf{r}(\alpha^i) = r_0 + r_1\alpha^i + r_2\alpha^{2i} + \dots + r_{n-1}\alpha^{(n-1)i} \quad \text{for } 1 \leq i \leq 2t$$

Dividing  $\mathbf{r}(X)$  by the minimal polynomial  $\phi_i(x)$  of  $\alpha^i$  we obtain:

$$\mathbf{r}(X) = \mathbf{a}_i(x)\phi_i(x) + \mathbf{b}_i(x)$$

Since  $\phi_i(\alpha^i) = 0$ , we have :

$$S_i = \mathbf{r}(\alpha^i) = \mathbf{b}_i(\alpha^i).$$

**Example2:** Consider the double-error-correcting (15,7) BCH code given in Example1. Suppose that the vector  $\mathbf{r} = (1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0)$  is received. The corresponding polynomial is:

$$\mathbf{r}(X) = 1 + X^8$$

The syndrome consists of 4 components,

$$\mathbf{S} = (S_1, S_2, S_3, S_4)$$

The minimal polynomials of  $\alpha$ ,  $\alpha^2$  and  $\alpha^4$  are identical and

$$\varphi_1(x) = \varphi_2(x) = \varphi_4(x) = 1+X+X^4$$

The minimal of polynomial of  $\alpha^3$  is

$$\varphi_3(x) = 1+X + X^2+X^3 +X^4$$

Dividing  $r(X)$  by  $\varphi_1(x)$  the remainder is:  $b_1(X) = X^2$

Dividing  $r(X)$  by  $\varphi_3(x)$  the remainder is:  $b_3(X) = 1 + X^3$

Substituting  $\alpha$ ,  $\alpha^2$  and  $\alpha^4$  into  $b_1(X)$  we obtain:

$$S_1 = \alpha^2, S_2 = \alpha^4, S_3 = \alpha^8$$

Substituting  $\alpha^3$  into  $b_3(X)$  we obtain:

$$S_4 = 1 + \alpha^9 = 1 + \alpha + \alpha^3 = \alpha^7$$

Thus:

$$\mathbf{S} = (\alpha^2, \alpha^4, \alpha^7, \alpha^8).$$

Since  $\alpha^1, \alpha^2, \dots, \alpha^{2t}$  are roots of each code polynomial,  $v(\alpha^i) = 0$  for  $1 \leq i \leq 2t$ .

We obtain the following relationship between the syndrome components and the error pattern:

$$S_i = r(\alpha^i) = v(\alpha^i) + e(\alpha^i) = 0 + e(\alpha^i) = e(\alpha^i) \quad \text{for } 1 \leq i \leq 2t \quad (4.17)$$



Suppose that the error pattern  $e(X)$  has  $\nu$  errors at locations  $X^{J_1}, X^{J_2}, \dots, X^{J_\nu}$ , that is:

$$e(X) = X^{J_1} + X^{J_2} + \dots + X^{J_\nu}. \quad (4.18)$$

From (4.17) and (4.18) we obtain the following set of equations:

$$\begin{aligned} S_1 &= \alpha^{J_1} + \alpha^{J_2} + \dots + \alpha^{J_\nu} \\ S_2 &= (\alpha^{J_1})^2 + (\alpha^{J_2})^2 + \dots + (\alpha^{J_\nu})^2 \\ S_3 &= (\alpha^{J_1})^3 + (\alpha^{J_2})^3 + \dots + (\alpha^{J_\nu})^3 \\ &\vdots \\ &\vdots \\ &\vdots \\ S_{2t} &= (\alpha^{J_1})^{2t} + (\alpha^{J_2})^{2t} + \dots + (\alpha^{J_\nu})^{2t} \end{aligned} \quad (4.19)$$

Any method for solving these equations is a decoding algorithm for the BCH codes. Once  $\alpha^{J_1}, \alpha^{J_2}, \dots, \alpha^{J_\nu}$  have been found  $J_1, J_2, \dots, J_\nu$  tell us the error locations in  $e(X)$ . In general, the equations have many possible solutions. The solution that yields an error pattern with the smallest number of errors is the right solution. For large  $t$ , solving equations directly is difficult and ineffective. There are methods to avoid this difficulty. Two of them are: *Peterson's algorithm* and *Berlekamp's iterative algorithm*.

#### 4.2.1.1 PETERSON'S ALGORITHM

Substituting  $\beta_1 = \alpha^{J_1}$  (4.19), we obtain:

$$\begin{aligned} S_1 &= \beta_1 + \beta_2 + \dots + \beta_\nu \\ S_2 &= \beta_1^2 + \beta_2^2 + \dots + \beta_\nu^2 \\ S_3 &= \beta_1^3 + \beta_2^3 + \dots + \beta_\nu^3 \\ &\vdots \\ &\vdots \\ &\vdots \\ S_{2t} &= \beta_1^{2t} + \beta_2^{2t} + \dots + \beta_\nu^{2t}. \end{aligned} \quad (4.20)$$

We define the following polynomial:

$$\sigma(X) = (1+\beta_1X) + (1+\beta_2X) + \dots + (1+\beta_vX) \quad (4.21)$$

$$= \sigma_0 + \sigma_1X + \sigma_2X^2 + \dots + \sigma_vX^v$$

The roots of  $\sigma(X)$  are  $\beta_1^{-1} + \beta_2^{-1} + \dots + \beta_v^{-1}$ , which are the inverses of the error location numbers. For this reason,  $\sigma(X)$  is called error-location polynomial.

$$\sigma_0 = 1$$

$$\sigma_1 = \beta_1 + \beta_2 + \dots + \beta_v$$

$$\sigma_2 = \beta_1\beta_2 + \beta_2\beta_3 + \dots + \beta_{v-1}\beta_v \quad (4.22)$$

.

.

.

$$\sigma_v = \beta_1\beta_2 \dots \beta_v$$

The  $\sigma_i$ 's are known as elementary symmetric functions of  $\beta_i$ 's.  $\sigma_i$ 's are related to the syndrome components by the following Newton's identities:

$$S_1 + \sigma_1 = 0$$

$$S_2 + \sigma_1S_1 + 2\sigma_2 = 0$$

$$S_3 + \sigma_1S_2 + \sigma_2S_1 + 3\sigma_3 = 0 \quad (4.23)$$

.

.

.

$$S_v + \sigma_1S_{v-1} + \dots + \sigma_{v-1}S_1 + v\sigma_v = 0$$

$$S_{v+1} + \sigma_1S_v + \dots + \sigma_{v-1}S_2 + \sigma_vS_1 = 0$$

Error correction procedure consists of three major steps:

1. Compute the syndrome  $S = (S_1, S_2, \dots, S_{2t})$  from the received polynomial  $r(X)$ .

2. Determine the error-location polynomial  $\sigma(X)$  from the syndrome components  $S_1, S_2, \dots, S_{2t}$ .
3. Determine the error-location numbers  $\beta_1, \beta_2, \dots, \beta_v$  by finding the roots of  $\sigma(X)$  and correct the errors in  $r(X)$ .

#### 4.2.1.2 BERLEKAMP'S ALGORITHM

The first step of Berlekamp's iterative algorithm is to find a minimum degree polynomial  $\sigma^{(1)}(X)$  whose coefficients satisfy the first Newton's identity. The next step is to test whether the coefficients of  $\sigma^{(1)}(X)$  also satisfy the second Newton's identity. If the coefficients of  $\sigma^{(1)}(X)$  do satisfy the second Newton's identity, we set  $\sigma^{(2)}(X) = \sigma^{(1)}(X)$ . If the coefficients of  $\sigma^{(1)}(X)$  do not satisfy the second Newton's identity, a correction term is added to  $\sigma^{(1)}(X)$  to form  $\sigma^{(2)}(X)$  such that  $\sigma^{(2)}(X)$  has minimum degree and its coefficients satisfy the first two Newton's identities. The third step of iteration is to find a minimum-degree polynomial  $\sigma^{(3)}(X)$  from  $\sigma^{(2)}(X)$  such that the coefficients of  $\sigma^{(3)}(X)$  satisfy the first three Newton's identities. Again, we test whether the coefficients of  $\sigma^{(2)}(X)$  also satisfy the third Newton's identity. If they do, we set  $\sigma^{(3)}(X) = \sigma^{(2)}(X)$ . If they do not, a correction term is added to  $\sigma^{(2)}(X)$  to form  $\sigma^{(3)}(X)$ . Iteration continues until  $\sigma^{(2t)}(X)$  is obtained. Then  $\sigma^{(2t)}(X)$  is taken to be error location polynomial  $\sigma(X)$ , that is,  $\sigma(X) = \sigma^{(2t)}(X)$ .

This  $\sigma(X)$  will yield an error pattern  $e(X)$  of minimum weight that satisfies the equations of (4.8). If the number of errors in the  $r(X)$  is  $t$  or less, then  $\sigma(X)$  produces the true error pattern.

Let  $\sigma^{(\mu)}(X) = 1 + \sigma_1^{(\mu)}X + \sigma_2^{(\mu)}X^2 + \dots + \sigma_{l_\mu}^{(\mu)}X^{l_\mu}$  be the minimum-degree polynomial determined at the  $\mu$ th step of iteration whose coefficients satisfy the first  $\mu$  Newton's identities. To determine  $\sigma^{(\mu+1)}(X)$  we compute the following quantity:

$$\mathbf{d}_\mu = S_{\mu+1} + \sigma_1^{(\mu)} S_\mu + \sigma_2^{(\mu)} S_{\mu-1} + \dots + \sigma_{l_\mu}^{(\mu)} S_{\mu+1-l_\mu}. \quad (4.24)$$

The quantity  $\mathbf{d}_\mu$  is called the  $\mu$ th discrepancy. If  $\mathbf{d}_\mu = 0$ , the coefficients of  $\sigma^{(\mu)}(X)$  satisfy  $(\mu+1)$ th Newton's identity. In this event, we set  $\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X)$ . If  $\mathbf{d}_\mu \neq 0$ , the coefficients of  $\sigma^{(\mu)}(X)$  do not satisfy the  $(\mu+1)$ th Newton's identity and a correction term must be added to  $\sigma^{(\mu)}(X)$  to obtain  $\sigma^{(\mu+1)}(X)$ . To accomplish this correction, we go back to the steps prior to the  $\mu$ th step and determine a polynomial  $\sigma^{(\rho)}(X)$  such that the  $\rho$ th discrepancy  $\mathbf{d}_\rho \neq 0$  and  $\rho - l_\rho$  [ $l_\rho$  is the degree of the  $\sigma^{(\rho)}(X)$ ] has the largest value. Then

$$\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + \mathbf{d}_\mu \mathbf{d}_\rho^{-1} X^{(\mu-\rho)} \sigma^{(\rho)}(X) \quad (4.25)$$

which is the minimum-degree polynomial whose coefficients satisfy the first  $(\mu+1)$  Newton's identities.

**TABLE 4.4** Berlekamp's iterative algorithm

$\mu$	$\sigma^{(\mu+1)}(X)$	$\mathbf{d}_\mu$	$l_\mu$	$\mu - l_\mu$
-1	1	1	0	-1
0	1	$S_1$	0	0
1				
2				
.				
.				
.				
2t				

To find error location polynomial, fill out the Table 4.4. If the polynomial has degree greater than  $t$ , there are more than  $t$  errors in  $r(X)$ , and generally it is not possible to locate them.

The last step in decoding BCH codes is to find the error-location numbers that are the reciprocals of the roots of  $\sigma(X)$ . The roots of  $\sigma(X)$  can be found simply by substituting  $1, \alpha, \alpha^2, \dots, \alpha^{n-1}$  into  $\sigma(X)$ . Since  $\alpha^n = 1$ ,  $\alpha^{-1} = \alpha^{n-1}$ . Therefore, if  $\alpha^i$  is a root of  $\sigma(X)$ ,  $\alpha^{n-i}$  is an error-location number and the received digit  $r_{n-i}$  is erroneous digit.

The substitution method described above for finding the roots of the error location polynomial was first used by Peterson in his algorithm for decoding BCH codes. Later, Chien formulated a procedure to carry out the substitution and error correction. The received vector

$$r(X) = r_0 + r_1X + r_2X^2 \dots + r_{n-1}X^{n-1}$$

is decoded on a bit-by-bit basis. The high-order bits are decoded first. To decode  $r_{n-1}$ , the decoder tests whether  $\alpha^{n-1}$  is an error-location number; this is equivalent to testing whether its inverse  $\alpha$  is a root of  $\sigma(X)$ . If  $\alpha$  is a root, then:

$$1 + \sigma_1\alpha + \sigma_2\alpha^2 \dots + \sigma_v\alpha^v = 0.$$

Therefore,  $\alpha^{n-1}$  is an error location number and  $r_{n-1}$  is an erroneous digit.

To decode  $r_{n-1}$  the decoder tests the sum

$$1 + \sigma_1\alpha^1 + \sigma_2\alpha^{2i} \dots + \sigma_v\alpha^{vi}.$$

If the sum is zero, then  $\alpha^{n-1}$  is an error location number and  $r_{n-1}$  is an erroneous digit. Error can be corrected by adding 1 to erroneous digit (use modulo-2 addition).

## 5 REED SOLOMON CODES

### 5.1 Historical Overview

Claude Shannon in 1948 had proven the existence of error-correcting codes that under suitable conditions and at rates less than channel capacity, would transmit error-free information for all practical applications.

The evolution of algebraic coding theory in the 1950's and 1960's was driven by the development of simplified encoders and less complex decoders by a host of researchers. The theory of Reed-Solomon codes is inextricably entwined with the history of block and more particularly algebraic coding theory. With the introduction of increasingly complex mathematical structures, broader classes of machine-encodable block codes emerged along with the algebraic means of decoding.

It is the early 1950s, Richard Hamming had already produced the first practical binary codes using the techniques of linear algebra. In fact, he had both introduced and completed the theory of optimal single-error-correcting binary codes. At almost the same time, Marcel Golay gave us the perfect triple-error-correcting code of length 23 and dimension 12. Golay's results opened up a Pandora's box for mathematical theorists searching for perfect optimal binary codes. David Muller, trained as a theoretical physicist, had invented a class of codes in a language of his own called "Boolean net functions". Shortly thereafter, a Caltech Ph.D. in mathematics with a minor in physics, Irving Reed, recognized an inherent algebraic structure in Muller's codes. They were multinomials over the Galois field of two elements.



Using the notion of multinomials over the primitive field  $GF(2)$  and constraining the maximum product degree, Reed constructed an error-correcting code that was equivalent to Muller's codes. The algebraic structure Reed imposed led to a decoding algorithm, the Reed algorithm, the first example of what is now called majority logic decoding. Reed and Muller's codes were demonstrated to be group codes, or vector spaces over  $GF(2)$ . Now called Reed-Muller codes, the codes were introduced in September 1954 at the first International Symposium on Information Theory in Cambridge, Massachusetts. The Grassmann algebra people later recognized this structure as belonging to them and extended Reed-Muller codes to algebraic number fields and other structures. With the work of Neal Zierler, Solomon Golomb and Eugene Prange, these codes were soon generated by linear shift registers (with parity added) and thus became endowed with a cyclic structure.

In the mid-1950s, Reed spent much of his time developing automatic processors for use in radar applications. This work culminated in 1957-1958 in the design of the first all-solid-state (transistor) computer, then called CG-24. This led to many firsts: the first machine to be designed and developed using the RTL language, the first machine to be emulated on another computer, and the first all-purpose machine to have a rudimentary interrupt structure. At this time, Reed became enamored of Galois theory and thought of using nonbinary finite field symbols in byte-level operations as opposed to the traditional focus on bit-oriented algorithms. This thought process ultimately led to Reed-Solomon codes.

In the late 1950s, Gustave Solomon, a young MIT Ph.D. mathematician specializing in algebra, was brought into the field by Reed. Reed introduced him to the world of coding theory and applied algebra through his ideas and conjectures. On January 21, 1959, Irving Reed and Gustave Solomon submitted a paper to the JSIAM. In June of 1960 the paper was published: five pages under the rather unpretentious title "Polynomial Codes over Certain Finite Fields". This paper described a new class of error-correcting codes that are now called Reed-Solomon codes.

## 5.2 Definition

Reed-Solomon codes form a subclass of the nonbinary BCH codes. Reed-Solomon codes are cyclic codes. Although they are a subclass of the nonbinary BCH codes, Reed-Solomon codes offer better error control performance and more efficient practical implementation because they have the largest minimum Hamming distance for fixed values of  $k$  and  $n$ .

Let  $\alpha$  be an element of  $GF(q^s)$  and let  $t_d$  be the designed error-correcting power of a BCH code. For some positive integers  $s$  and  $b \geq 1$ , a BCH code of length  $n$  and minimum Hamming distance  $\geq 2t_d + 1$  can be generated by the generator polynomial  $g(X)$  over  $GF(q)$  with  $\alpha^b, \alpha^{b+1}, \dots, \alpha^{b+2t_d-1}$  as the roots of  $g(x)$ . Let  $\alpha^i$ , a nonzero element in  $GF(q^s)$ , be root of the minimal polynomial  $\phi_i(X)$  over  $GF(q)$  and  $n_i$  be the order of  $\alpha^i$  for  $i = b, b+1, \dots, b+2t_d-1$ . The generator polynomial of a BCH code can be expressed in the form

$$g(X) = \text{LCM}\{\phi_b(X), \phi_{b+1}(X), \dots, \phi_{b+2t_d-1}(X)\}.$$

The length of the code is

$$n = \text{LCM}\{n_b, n_{b+1}, \dots, n_{b+2t_d-1}\}.$$

The degree of  $\phi_i(X)$  is  $s$  or less, and then the degree of  $g(X)$  is, therefore, at most equal to  $2st_d$ .

Reed-Solomon codes can be obtained by setting  $s = 1$ ,  $b = 1$  and  $q = p^m$  where  $p$  is some prime. Let  $\alpha$  be a primitive element in  $GF(p^m)$ . A primitive Reed-Solomon code with symbols from  $GF(p^m)$ :

$$\text{Block length:} \quad n = p^m - 1$$

$$\text{Number of check digits:} \quad c = (n - k) = 2t_d$$

Minimum distance:  $d_{\min} = 2t_d + 1$

An important property of any Reed-Solomon codes is that the true minimum Hamming distance  $t$  is always equal to the designed distance  $t_d$ . This tells us that for a fixed  $(n,k)$ , no code can have a larger minimum distance than a Reed-Solomon code. A Reed-Solomon code is therefore a maximum distance code.

Let  $\alpha^i$  be a root of the  $\phi_i(X)$  and  $n_i$  be the order of  $\alpha^i$ , for  $i = 1, 2, \dots, 2t$ . The generator polynomial of a primitive Reed-Solomon code is

$$\begin{aligned} g(X) &= (X - \alpha)(X - \alpha^2) \dots (X - \alpha^{2t}) \\ &= g_0 + g_1X + g_2X^2 + \dots + g_{2t-1}X^{2t-1} + X^{2t} \end{aligned}$$

### 5.3 Encoding

The code generated by  $g(X)$  is an  $(n, n-2t)$  cyclic code. Let

$$a(X) = a_0 + a_1X + a_2X^2 + \dots + a_{k-1}X^{k-1}$$

be the message to be encoded where  $k = n - 2t$ . In systematic form  $2t$  parity-check digits are the coefficients of the remainder  $b(X) = b_0 + b_1X + b_2X^2 + \dots + b_{2t-1}X^{2t-1}$  resulting from dividing the message polynomial  $X^{2t}a(X)$  by the generator polynomial  $g(X)$ .

$$b(X) = X^{2t}a(X) \bmod g(X)$$

$$v(X) = X^{2t}a(X) + b(X)$$

This is accomplished by using a division circuit like below:

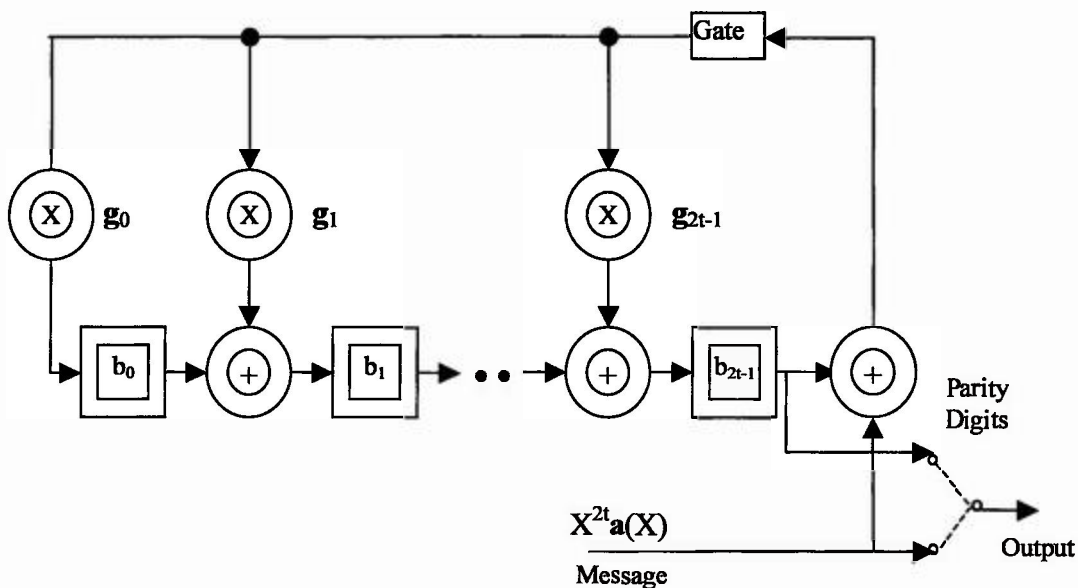


Figure 5.1 Division circuit for encoding

#### 5.4 Decoding

Decoding of Reed-Solomon codes contains the following steps:

1. Compute the syndrome  $S = (S_1, S_2, \dots, S_{2t})$  from the received polynomial  $r(X)$ .
2. Determine the error-location polynomial  $\sigma(X)$  from the syndrome components  $S_1, S_2, \dots, S_{2t}$ .
3. Determine the error-location numbers from the roots of  $\sigma(X)$ .
4. Determine the error values.
5. Correct the error words.

Syndrome computation:

Let

$$v(X) = v_0 + v_1X + v_2X^2 + \dots + v_{n-1}X^{n-1}$$

be the transmitted code vector and let

$$\mathbf{r}(X) = \mathbf{r}_0 + \mathbf{r}_1X + \mathbf{r}_2X^2 + \dots + \mathbf{r}_{n-1}X^{n-1}$$

be the corresponding received vector. Then the error pattern added by the channel is

$$\mathbf{e}(X) = \mathbf{v}(X) + \mathbf{r}(X) = \mathbf{e}_0 + \mathbf{e}_1X + \mathbf{e}_2X^2 + \dots + \mathbf{e}_{n-1}X^{n-1}.$$

Suppose that the  $\mathbf{e}(X)$  contains  $v$  errors at location  $X^{j_1}, X^{j_2}, \dots, X^{j_v}$  where  $0 \leq j_1 < j_2 < \dots < j_v \leq n-1$ . Then

$$\mathbf{e}(X) = \mathbf{e}_{j_1}X^{j_1} + \mathbf{e}_{j_2}X^{j_2} + \mathbf{e}_{j_3}X^{j_3} + \dots + \mathbf{e}_{j_v}X^{j_v}$$

To determine  $\mathbf{e}(X)$ , we need to know the error locations  $X^{j_i}$ 's and the error values  $\mathbf{e}_{j_i}$ 's. As with BCH codes we define for  $l = 1, 2, \dots, v$

$$\beta_l = \alpha^{j_l}$$

as error location numbers. Syndrome components can be obtained like below

$$\begin{aligned} S_1 &= \mathbf{r}(\alpha) = \mathbf{e}_{j_1}\beta_1 + \mathbf{e}_{j_2}\beta_2 + \dots + \mathbf{e}_{j_v}\beta_v \\ S_2 &= \mathbf{r}(\alpha^2) = \mathbf{e}_{j_1}\beta_1^2 + \mathbf{e}_{j_2}\beta_2^2 + \dots + \mathbf{e}_{j_v}\beta_v^2 \\ S_3 &= \mathbf{r}(\alpha^3) = \mathbf{e}_{j_1}\beta_1^3 + \mathbf{e}_{j_2}\beta_2^3 + \dots + \mathbf{e}_{j_v}\beta_v^3 \\ &\vdots \\ &\vdots \\ &\vdots \\ S_{2t} &= \mathbf{r}(\alpha^{2t}) = \mathbf{e}_{j_1}\beta_1^{2t} + \mathbf{e}_{j_2}\beta_2^{2t} + \dots + \mathbf{e}_{j_v}\beta_v^{2t} \end{aligned} \tag{5.1}$$

If there are  $t$  errors in received word (5.1) equations can be given like below:

$$S_{2t} = \sum_{i=1}^t \mathbf{e}_{j_i} \beta_i^k \quad (5.2)$$

The aim of the decoding is to find the error vector with maximum  $t$  errors which produces the syndromes at (5.2).

#### 5.4.1 Peterson's Algorithm

With a few differences we can use this method at RS codes like binary BCH codes. Newton's identities:

$$S_{t+j} + \sigma_1 S_{t+j-1} + \dots + \sigma_t S_j = 0 \quad (5.3)$$

$\sigma_i$ 's are the coefficients of  $\sigma(X)$ .

$$\sigma(X) = X^t + \sigma_1 X^{t-1} + \dots + \sigma_t \quad (5.4)$$

In the first step of decoding  $2t$  syndromes  $S_1, S_2, \dots, S_{2t}$  are computed. Then  $t$  equalities are obtained from (5.4) for  $k \leq t$ . The solution of the equalities gives us the coefficients of the error location polynomial  $\sigma(X)$ . As an example consider a triple-error-correcting RS code:

$$\begin{aligned} S_1 \sigma_3 + S_2 \sigma_2 + S_3 \sigma_1 &= -S_4 \\ S_2 \sigma_3 + S_3 \sigma_2 + S_4 \sigma_1 &= -S_5 \\ S_3 \sigma_3 + S_4 \sigma_2 + S_5 \sigma_1 &= -S_6 \end{aligned} \quad (5.5)$$

We must solve the (5.5) equalities to find the coefficients of the error location polynomial. The number of the equalities are equal to the number of errors at the received word. Writing (5.5) equalities in matrix form, we obtain:



$$\begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} \begin{bmatrix} \sigma_3 \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_4 \\ -S_5 \\ -S_6 \end{bmatrix} \quad (5.6)$$

(5.6) takes a simple case for one and two errors.

$$[S_1][\sigma_1] = [-S_2] \quad (5.7)$$

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_3 \\ -S_4 \end{bmatrix} \quad (5.8)$$

To find the coefficients of the error location polynomial from (5.6), (5.7) and (5.8) the determinant of the coefficient matrix must be different from zero. Taking  $D_2$  the determinant of the (5.8)'s coefficient matrix and  $D_3$  the determinant of the (5.6)'s coefficient matrix:

$$D_2 = S_1 S_3 - S_2^2 \quad (5.9)$$

$$D_3 = S_1 S_3 S_5 + S_2 S_3 S_4 + S_2 S_3 S_4 - S_3^3 - S_1 S_4^2 - S_2^2 S_5 \quad (5.10)$$

$D_3$  and  $D_2$  are computed to determine how many errors are in the received word. This two expressions are equal to zero at one error case. For double erroneous words  $D_3$  is equal to zero. After determining  $\sigma_i$ 's, the roots of the error location polynomial, the error locations, are computed. Error values are computed by substituting error locations into syndrome equation (5.2) and decoding is finished.

Let use Peterson's directly solution method at triple-error-correcting (15,9) RS code over  $GF(2^4)$ .

1. Syndromes  $S_k$  are computed by using (5.1) equations, for  $1 \leq k \leq 6$

$$S_k = r(\alpha^k)$$

If  $S_k = 0$ , for  $1 \leq k \leq 6$  the received word is a code word and we assume no errors are occurred in transmission.

2. Determining the number of errors in the received word:

a. If  $D_3 = S_1 S_3 S_5 + S_3^3 + S_1 S_4^2 + S_2^2 S_5 \neq 0$  we assume there exist three errors.

b. If  $D_3 = 0$  and  $D_2 = S_1 S_3 + S_2^2 \neq 0$  we assume there exist two errors.

c. If  $D_3 = D_2 = 0$  and  $S_1 \neq 0$  we assume there exists one error.

3. Determining the coefficients of the error location polynomial:

a. The state of existing three errors:

$$\begin{aligned} \sigma_1 &= [S_1 S_3 S_6 + S_1 S_4 S_5 + S_2^2 S_6 + S_2 S_3 S_5 + S_2 S_4^2 + S_3^2 S_4] / D_3 \\ \sigma_2 &= [S_1 S_4 S_6 + S_1 S_5^2 + S_2 S_3 S_6 + S_2 S_4 S_5 + S_3^2 S_5 + S_3 S_4^2] / D_3 \\ \sigma_3 &= [S_2 S_4 S_6 + S_2 S_5^2 + S_3^2 S_6 + S_4^3] / D_3 \end{aligned} \quad (5.11)$$

b. The state of existing two errors:

$$\begin{aligned} \sigma_1 &= [S_1 S_4 + S_2 S_3] / D_2 \\ \sigma_2 &= [S_2 S_4 S_6 + S_3^3] / D_2 \end{aligned}$$

c. The state of existing one error:

$$\sigma_1 = \beta_1 = S_2 / S_1$$

4. Determining the error locations:

a. The state of existing three errors:

$$\sigma(X) = X^3 + \sigma_1 X^2 + \sigma_2 X + \sigma_3$$

b. The state of existing two errors:

$$\sigma(X) = X^2 + \sigma_1 X + \sigma_2$$

The roots of the polynomials are computed. In the state of existing one error, error location polynomial  $\sigma(X) = X + \sigma_1$  and the root of the error location polynomial  $\sigma_1 = S_2 / S_1$ . There is an uncorrectable mistake if we can't find the true number of error location.

5. After determining error locations, error values are computed by solving syndrome equations.

a. The state of existing three errors:

$$C = \beta_1 \beta_2^3 \beta_3^3 + \beta_1^3 \beta_2 \beta_3^2 + \beta_1^2 \beta_2^3 \beta_3 + \beta_1^3 \beta_2^2 \beta_3 + \beta_1 \beta_2^3 \beta_3^2 + \beta_1^2 \beta_2 \beta_3^3$$

$$e_{J_1} = [S_1 \beta_2^3 \beta_3^3 + S_2 \beta_2^3 \beta_3 + S_3 \beta_2 \beta_3^2 + S_1 \beta_2^3 \beta_3^2 + S_2 \beta_2 \beta_3^3 + S_3 \beta_2^2 \beta_3] / C$$

$$e_{J_2} = [S_1 \beta_2^3 \beta_3^2 + S_2 \beta_1 \beta_3^3 + S_3 \beta_1^2 \beta_3 + S_1 \beta_1^2 \beta_3^3 + S_2 \beta_1^3 \beta_3 + S_3 \beta_1 \beta_3^2] / C$$

$$e_{J_3} = [S_1 \beta_1^2 \beta_2^3 + S_2 \beta_2 \beta_1^3 + S_3 \beta_2^2 \beta_1 + S_1 \beta_1^3 \beta_2^2 + S_2 \beta_2^3 \beta_1 + S_3 \beta_2 \beta_1^2] / C$$

(5.12)

b. The state of existing two errors:

$$e_{J_1} = [S_1 \beta_2 + S_2] / [\beta_1 \beta_2 + \beta_1^2]$$

$$e_{J_2} = [S_1 \beta_1 + S_2] / [\beta_1 \beta_2 + \beta_1^2]$$

b. The state of existing one error:

$$e_{j1} = S_1^2 / S_2$$

6. Error correction is made by adding the symbols at the erroneous locations of the received word to the computed error values.

7. Syndrome is computed for the corrected word. If the syndrome is not equal to zero corrected word is erroneous, in fact error is not corrected.

Consider a (15,9) RS code, with the generator polynomial  $\alpha^6 + \alpha^9X + \alpha^6X^2 + \alpha^4X^3 + \alpha^{14}X^4 + \alpha^{10}X^5 + X^6$ , the transmitted code vector  $v = (000000000000000)$ , and the received code vector  $r = (000\alpha^700\alpha^300000\alpha^400)$ . The polynomial representation of the received word be  $r(X) = \alpha^7X^3 + \alpha^3X^6 + \alpha^4X^{12}$ . The syndromes are obtained from (5.1),

$$S_1 = \alpha^{12}, S_2 = 1, S_3 = \alpha^{14}, S_4 = \alpha^{10}, S_5 = 0, S_6 = \alpha^{12}.$$

Because of the computing  $D_3 = \alpha^7 \neq 0$  from (5.7) We assume there exists three errors. From (5.8) equalities, we obtain,

$$\sigma_1 = \alpha^7, \sigma_2 = \alpha^4, \sigma_3 = \alpha^6$$

$$\sigma(X) = X^3 + \alpha^7X^2 + \alpha^4X + \alpha^6.$$

Substituting the all elements of the  $GF(2^4)$  into  $\sigma(X)$ , we obtain,

$$\sigma(\alpha^3) = \sigma(\alpha^6) = \sigma(\alpha^{12}) = 0.$$

Here we can see 3th, 6th and 12th symbols are erroneous. We can compute the error locations from (5.12) equations.

$$e_3 = \alpha^7, e_6 = \alpha^3, e_{12} = \alpha^4$$

$$e(X) = \alpha^7 X^3 + \alpha^3 X^6 + \alpha^4 X^{12}$$

$$v(X) = r(X) + e(X) = 0$$

is obtained. It is clear that the syndromes of the corrected word is equal to zero. Errors are corrected for the code word which is transmitted erroneous at three locations.

#### 5.4.2 Berlekamp's Algorithm

It is easy that, the decoding the codes which corrects low number of erroneous symbols, with Peterson's method. Like binary BCH codes, in RS codes we use the Berlekamp's method because of the complexity while computing the coefficients of  $\sigma(X)$  at the codes which has more than six erroneous symbols. Unlike from the decoding of binary BCH codes, in the decoding of RS codes, we must compute the error values. After determining the error locations, error values are computed by determining the error evaluator polynomial  $\omega(X)$ . The relation with  $\omega(X)$  and  $\sigma(X)$  are defined with (5.13).

$$\omega(X) \equiv \sigma(X)[S(X) + 1] \bmod X^{2t+1} \quad (5.13)$$

This equation known as key-equation which is used in decoding of the BCH codes with the Berlekamp's method. In this equation  $S(X)$  known as syndrome polynomial and given by,

$$S(X) = S_1 X + S_2 X^2 + \dots + S_{2t} X^{2t}.$$

Furthermore, the definition of the error location polynomial  $\sigma(X)$  in the key-equation is different from Peterson's method. Now  $\sigma(X)$  is defined like below, where  $\beta_i$ s are error location and  $i = 1, 2, \dots, t$ ,

$$\sigma(X) = (1 + \beta_1 X) + (1 + \beta_2 X) + \dots + (1 + \beta_t X).$$

That is to say the inverse of the roots of  $\sigma(X)$  gives us the error locations. Berlekamp's algorithm is an iterative algorithm which is developed for determining  $\sigma(X)$ . After finding the error location polynomial the inverse of the roots of the polynomial gives us the error locations. Error evaluator polynomial  $\omega(X)$  is found by substituting error location polynomial into (5.13). Error value  $e_{j_i}$ , corresponding to error location  $\beta_i$ , is computed from (5.14) where  $\sigma'(X)$  denotes the derivative of  $\sigma(X)$ .

$$e_{j_i} = \beta_i \frac{\omega(\beta_i^{-1})}{\sigma'(\beta_i^{-1})} \quad (5.14)$$

Using (5.13) error polynomial  $e(X)$  is found and error correction is made by substituting it into  $v(X) = r(X) + e(X)$ .

Consider a (15,9) RS code, with the generator polynomial  $\alpha^6 + \alpha^9X + \alpha^6X^2 + \alpha^4X^3 + \alpha^{14}X^4 + \alpha^{10}X^5 + X^6$ , the transmitted code vector  $v = (000000000000000)$ , and the received code vector  $r = (000\alpha^7 00\alpha^3 00000\alpha^4 00)$ . The polynomial representation of the received word be  $r(X) = \alpha^7X^3 + \alpha^3X^6 + \alpha^4X^9$ . The syndromes are obtained from (5.1),

$$S_1 = \alpha^{12}, S_2 = 1, S_3 = \alpha^{14}, S_4 = \alpha^{10}, S_5 = 0, S_6 = \alpha^{12}.$$

Berlekamp's algorithm is applied by filling out the Table 5.1.



**TABLE 5.1** Application of Berlekamp's iterative algorithm

$\mu$	$\sigma^{(\mu+1)}(X)$	$d_\mu$	$l_\mu$	$\mu - l_\mu$	
-1	1	1	0	-1	
0	1	$S_1 = \alpha^2$	0	0	
1	$1 + \alpha^{12}X$	$\alpha^7$	1	0	take $p = -1$
2	$1 + \alpha^3X$	1	1	1	take $p = 0$
3	$1 + \alpha^3X + \alpha^3X^2$	$\alpha^7$	2	1	take $p = 0$
4	$1 + \alpha^4X + \alpha^{12}X^2$	$\alpha^{10}$	2	2	take $p = 2$
5	$1 + \alpha^7X + \alpha^4X^2 + \alpha^6X^3$	0	3	2	take $p = 3$
6	$1 + \alpha^7X + \alpha^4X^2 + \alpha^6X^3$	-	-	-	

At the 5th step, the polynomial is the error location polynomial which has minimum distance equal to  $d_\mu = 0$ .

$$\sigma(X) = 1 + \alpha^7X + \alpha^4X^2 + \alpha^6X^3$$

Error evaluator polynomial is computed from the key equation given by (5.13).

$$\omega(X) \equiv \sigma(X)[S(X) + 1] \pmod{X^{2t+1}}$$

$$\omega(X) = [\alpha^3X^9 + \alpha X^8 + X^7 + \alpha^6X^3 + X^2 + \alpha^2X + 1] \pmod{X^7}$$

$$\omega(X) = \alpha^6X^3 + X^2 + \alpha^2X + 1$$

$\alpha^3$ ,  $\alpha^9$  and  $\alpha^{12}$  is found as the roots of the  $\sigma(X)$  by substituting the all elements of the  $GF(2^4)$  into  $\sigma(X)$ . The inverse of the roots  $\alpha^{12}$ ,  $\alpha^6$  and  $\alpha^3$  are error locations. That is to say 3th, 6th, and 12th symbols are received erroneous. Using (5.14) to compute the error values;

$$e_3 = (\alpha^3)^{-1} \frac{\omega((\alpha^3)^{-1})}{\sigma'((\alpha^3)^{-1})} = \alpha^7$$

$$e_6 = (\alpha^6)^{-1} \frac{\omega((\alpha^6)^{-1})}{\sigma'((\alpha^6)^{-1})} = \alpha^3$$

$$e_{12} = (\alpha^{12})^{-1} \frac{\omega((\alpha^{12})^{-1})}{\sigma'((\alpha^{12})^{-1})} = \alpha^4$$

are obtained. Error polynomial is found like below:

$$e(X) = \alpha^7 X^3 + \alpha^3 X^6 + \alpha^4 X^{12}.$$

The result is same as the preceding result which is found by using Peterson's method.

We can see from  $v(X) = r(X) + e(X) = 0$  that the error is corrected.

### 5.4.3 Euclid's Algorithm

We can use the Euclid's method when decoding RS codes. Euclid's algorithm, gives us the GCD  $C$  of the any two integers or polynomials  $A$  and  $B$ . Furthermore, it finds the integers or polynomials  $S$  and  $T$  which satisfies the equality  $C = SA + TB$ . If we give the key equation like (5.15), we can find the polynomials  $\sigma(X)$  and  $\omega(X)$  by using Euclid's algorithm.

$$\omega(X) \equiv \sigma(X)[S(X) + 1] + \mu(X)X^{2t+1} \quad (5.15)$$

While applying the algorithm if we choose the polynomials  $A = X^{2t+1}$ ,  $B = S(X) + 1$ , we can find the  $\sigma(X)$  and  $\omega(X)$ . First we will define the Euclid's algorithm for integers and polynomials and we will give an example.

If  $A$  and  $B$  are integers take  $A \geq B$ , if they are polynomials take  $\deg(A) \geq \deg(B)$ . If we choose the initial conditions like  $r_{-1} = A$  and  $r_0 = B$ , then we obtain, at the  $n$ th step of the algorithm, the remainder  $r_{n-1}$  which is found from the division of  $r_{n-2}$  to  $r_{n-1}$  ( $r_{n-2} = q_n r_{n-1} + r_n$ ). For integers  $r_n \leq r_{n-1}$  and for polynomials  $\deg(r_n) \leq \deg(r_{n-1})$ . We find  $r_n$  at the  $n$ th step using the below equation:

$$r_n = r_{n-2} - q_n r_{n-1}. \quad (5.16)$$

Furthermore, we can find the  $s_n$  and  $t_n$  which satisfies the equality  $r_n = s_n A + t_n B$ . For  $s_n$  and  $t_n$  following equations are given:

$$\begin{aligned} s_n &= s_{n-2} - q_n s_{n-1} \\ t_n &= t_{n-2} - q_n t_{n-1} \end{aligned} \tag{5.17}$$

Because of being  $r_{-1} = A = (1)A + (0)B$  and  $r_0 = B = (0)A + (1)B$  we take  $s_{-1} = 1$ ,  $t_{-1} = 1$ ,  $s_0 = 0$ ,  $t_0 = 1$  for the initial conditions. As an example we will use the Euclid's algorithm to find the GCD(42,24). At first step the remainder  $r_1 = 18$  quotient  $q_1 = 1$  which is obtained from the division of 42 to 24.

If we continue while the remainder obtained from division be zero then we find GCD(42,24) = 6. The remainders at the every step can be given like below by the kind of  $s_n$  and  $t_n$ :

$$\begin{aligned} 42 &= (1).42 + (0).24 \\ 24 &= (0).42 + (1).24 \\ 18 &= (1).42 + (-1).24 \\ 6 &= (-1).42 + (2).24 \\ 0 &= (4).42 + (-7).24 \end{aligned}$$

The easiest way to use Euclid's algorithm to make a table for  $r_n$ ,  $q_n$ ,  $s_n$  ve  $t_n$ . Table 5.2 is given at the below for the example:

**TABLE 5.2** Application of Euclid's algorithm for integers

$n$	$r_n$	$q_n$	$s_n = s_{n-2} - q_n s_{n-1}$	$t_n = t_{n-2} - q_n t_{n-1}$
-1	42	-	1	0
0	24	-	0	1
1	18	1	1	-1
2	6	1	-1	2
3	0	3	4	-7

By using Euclid's algorithm at the key-equation

$$\omega(X) \equiv \sigma(X)[S(X) + 1] + \mu(X)X^{2t+1}$$

if we apply it to the  $X^{2t+1}$  ve  $[S(X) + 1]$ , we find the equality

$$r_n(X) \equiv s_n(X)X^{2t+1} + t_n(X) [S(X) + 1]$$

at a certain step in condition that not exceeding the code's capacity. Here  $n$  is the first value which satisfies the condition  $\deg(r_n) \leq t$ . From here we obtain

$$\sigma(X) = t_n(X)$$

$$\omega(X) = r_n(X).$$

Now we know the error location and error evaluator polynomial. So using (5.14), we find error values and decoding finishes.

Let us use the Euclid's algorithm to the preceding RS code. Consider a (15,9) RS code, with the transmitted code vector  $v = (000000000000000)$ , and the received code vector  $r = (000\alpha^7 00\alpha^3 00000\alpha^4 00)$ . The polynomial representation of the received word be  $r(X) = \alpha^7 X^3 + \alpha^3 X^6 + \alpha^4 X^{12}$ . The syndromes are obtained from (5.1),

$$S_1 = \alpha^{12}, S_2 = 1, S_3 = \alpha^{14}, S_4 = \alpha^{10}, S_5 = 0, S_6 = \alpha^{12}$$

$$S(X) = \alpha^7 X^6 + \alpha^{10} X^4 + \alpha^{14} X^3 + X^2 + \alpha^{12} X.$$

Fill out the table to apply the Euclid's algorithm.

**TABLE 5.3** Application of Euclid's algorithm

$n$	$r_n$	$q_n$	$t_n = t_{n-2} - q_n t_{n-1}$
-1	$X^7$	-	0
0	$S(X) + 1$	-	1
1	$\alpha^{13}X^5 + \alpha^2X^4 + \alpha^3X^3 + X^2 + \alpha^3X$	$\alpha^3X$	$\alpha^3X$
2	$\alpha^8X^4 + \alpha^6X^3 + \alpha^{13}X^2 + \alpha^4X + 1$	$\alpha^4X + \alpha^3$	$\alpha^2X^2 + \alpha^6X + 1$
3	$\alpha^7X^3 + \alpha X^2 + \alpha^3X + \alpha$	$\alpha^4X + \alpha^3$	$\alpha^7X^3 + \alpha^5X^2 + \alpha^8X + \alpha$

Because of  $\deg(r_3) = 3 \leq t = 3$  at the third step, we obtain,

$$\omega(X) = \alpha^7X^3 + \alpha X^2 + \alpha^3X + \alpha$$

$$\sigma(X) = \alpha^7X^3 + \alpha^5X^2 + \alpha^8X + \alpha.$$

Notice that the obtained polynomials are  $\alpha$  times of the polynomials which are obtained by using the Berlekamp's method. This condition does not change the result because it does not change the roots of the error location polynomial. If we look the (5.14) which is using for determining the error values we see a division operation between  $\omega(X)$  and  $\sigma'(X)$ . In this operation  $\alpha$  simplifies. Error values are obtained like below:

$$e_3 = (\alpha^3)^{-1} \frac{\omega((\alpha^3)^{-1})}{\sigma'((\alpha^3)^{-1})} = \alpha^7$$

$$e_6 = (\alpha^6)^{-1} \frac{\omega((\alpha^6)^{-1})}{\sigma'((\alpha^6)^{-1})} = \alpha^3$$

$$e_{12} = (\alpha^{12})^{-1} \frac{\omega((\alpha^{12})^{-1})}{\sigma'((\alpha^{12})^{-1})} = \alpha^4$$

Error polynomial is obtained like:

$$e(X) = \alpha^7X^3 + \alpha^3X^6 + \alpha^4X^{12}.$$

We can see from  $v(X) = r(X) + e(X) = 0$  that the error is corrected.

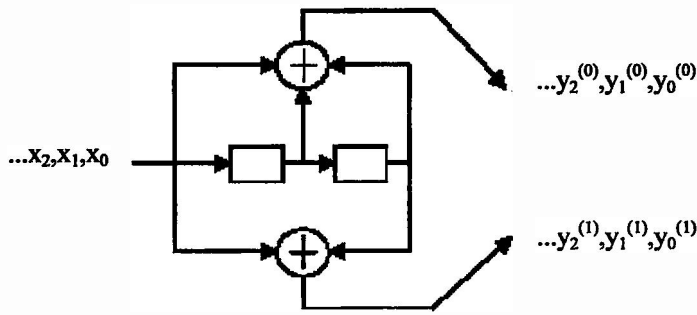
## 6 CONVOLUTIONAL CODING

The encoding process of convolutional codes is significantly different to that of block encoding. Block codes are developed through the use of algebraic techniques. Block encoders group information bits into length  $k$  blocks. These blocks are then mapped into codewords of length  $n$ . A convolutional encoder converts the entire input stream into length  $n$  codewords independent of the length  $k$ . The development of convolutional codes is based mostly on physical construction techniques. The evaluation and the nature of the design of convolutional codes depends less on an algebraic manipulation and more on construction of the encoder.

Convolutional codes were first introduced by Elias in 1955. He proved that redundancy could be added to an information stream through the use of linear shift registers. In 1961, Wozencraft and Reiffen described the first practical decoding algorithm for convolutional codes. The algorithm was based on sequential decoding, however sub-optimal for decoding convolutional codes. Several other algorithms were developed off of Wozencraft and Reiffen initial work. In 1967, Viterbi proposed a maximum likelihood-decoding scheme for decoding convolutional codes. The importance of the Viterbi algorithm is that it proved to be relatively easy to implement given the encoder has a small number of memory elements. It is the work by Viterbi that promotes the motivation here to apply his algorithm for the decoding of this error correction scheme.

Figure 6.1 is a binary rate  $1/2$  linear convolutional encoder. The rate of the encoder is determined by the fact that the encoder outputs two bits for every one bit at the input. In general, an encoder with  $k$  input bits and  $n$  output bits is said to have a rate  $k/n$ . The rate  $k/n$  is defined as the code rate ( $R_c$ ) of the system.





**Figure 6.1** Binary rate 1/2 linear convolutional encoder

In Fig. 6.1 a binary stream of data  $\mathbf{x}=(x_0,x_1,x_2,\dots)$  is fed into a series of memory elements. The bits travel through the shift register, the values of the individual memory elements are tapped off and added modulo-2 according to a fixed pattern. This creates a pair of output coded data streams  $\mathbf{y}=(y_0^{(0)},y_1^{(0)},y_2^{(0)},\dots)$  and  $\mathbf{y}=(y_0^{(1)},y_1^{(1)},y_2^{(1)},\dots)$ . These output streams are multiplexed to create a single encoded data stream  $\mathbf{y}=(y_0^{(0)}, y_0^{(1)}, y_1^{(0)}, y_1^{(1)}, y_2^{(0)}, y_2^{(1)}, \dots)$ . The data stream  $\mathbf{y}$  is the convolutional code word. Each element in the interleaved output stream  $\mathbf{y}$  is a linear combination of the elements in the input stream  $x^{(0)}, x^{(1)}, \dots, x^{(k-1)}$  assuming that the shift register contents are initialized to zero before the encoding process. The linearity of the codes words shows that if  $\mathbf{y}_1$  and  $\mathbf{y}_2$  are code words corresponding to inputs  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , then  $(\mathbf{y}_1+\mathbf{y}_2)$  is the code word that corresponds to the input of  $(\mathbf{x}_1+\mathbf{x}_2)$ . The linear structure of these codes allows for use of powerful techniques from linear algebra theory.

There is a way to characterize the encoder structure of convolutional codes called generator sequences. Generator sequences are obtained by applying an impulse response  $\mathbf{g}_j^{(i)}$  where the  $i$ th output of the encoder is obtained by applying a Dirac delta function  $\delta=(1000\dots)$  data stream at the  $j$ th input. The impulse responses for Fig. 6.1 are

$$\begin{aligned} \mathbf{g}^{(0)} &= (111) \\ \mathbf{g}^{(1)} &= (101) \end{aligned} \tag{6.1}$$

The generators have been terminated at a point where the following output values are all zeros. It should now be evident that the generators sequences can be determined by “counting” the number of taps off the shift register that connect to the  $j$ th generator sequence. Since there are two memory elements each incoming bit can affect at most 3 bits, hence the length of the generator sequence. The constraint length  $K$  of a convolutional code in its simplest terms can be defined as the maximum number of taps off the shift registers in the encoder.

$$K = l + 1,$$

where  $l$  is the number of memory elements of the encoder structure. The memory of the encoder has a direct impact to the complexity of the decoder, specifically the Viterbi algorithm that is used here. In practical implementations of the Viterbi algorithm the complexity is exponential in the constraint length and the number of input bits  $k$ .

There are two popular ways to describe a convolutional encoder. One way is graphically like Fig. 6.1; the other is by a generator matrix (6.2). A generator matrix is formed by interleaving the generator sequences  $\mathbf{g}^{(0)}$  and  $\mathbf{g}^{(1)}$ , where  $m$  the number of generator sequences.

$$\mathbf{G} = \begin{bmatrix} g_0^{(0)} & g_0^{(1)} & g_1^{(0)} & g_1^{(1)} & g_2^{(0)} & g_2^{(1)} & \dots & g_m^{(0)} & g_m^{(1)} & 0 \\ & g_0^{(0)} & g_0^{(1)} & g_1^{(0)} & g_1^{(1)} & g_2^{(0)} & g_2^{(1)} & \dots & g_m^{(0)} & g_m^{(1)} & 0 \\ & & g_0^{(0)} & g_0^{(1)} & g_1^{(0)} & g_1^{(1)} & g_2^{(0)} & g_2^{(1)} & \dots & g_m^{(0)} & g_m^{(1)} & 0 \\ \mathbf{0} & & & \ddots & \ddots & \ddots & \dots & \ddots & \ddots & g_m^{(0)} & g_m^{(1)} \end{bmatrix} \quad (6.2)$$

The action of the convolutional encoder can be described as a discrete convolutional operation, which leads for an appropriate transform that will provide a simpler multiplicative representation for encoding. The D-transform, called the delay transform can be interpreted as a delay operator, with the exponent denoting the number of time delay with respect to the  $D^{(0)}$  term.

$$\begin{aligned}
\mathbf{x}^{(i)} &= (x_0^{(i)}, x_1^{(i)}, x_2^{(i)} \dots) \Leftrightarrow \mathbf{X}^{(i)}(D) = x_0^{(i)} + x_1^{(i)}D + x_2^{(i)}D^2 + \dots \\
\mathbf{y}^{(i)} &= (y_0^{(i)}, y_1^{(i)}, y_2^{(i)} \dots) \Leftrightarrow \mathbf{Y}^{(i)}(D) = y_0^{(i)} + y_1^{(i)}D + y_2^{(i)}D^2 + \dots \\
\mathbf{g}_j^{(i)} &= (g_{j0}^{(i)}, g_{j1}^{(i)}, g_{j2}^{(i)} \dots) \Leftrightarrow \mathbf{G}_j^{(i)}(D) = g_{j0}^{(i)} + g_{j1}^{(i)}D + g_{j2}^{(i)}D^2 + \dots
\end{aligned}
\tag{6.3}$$

The encoding operation of a single input encoder can be represented as follows.

$$\begin{aligned}
\mathbf{Y}^{(i)}(D) &= \mathbf{X}(D) \mathbf{G}_j^{(i)}(D) \\
\mathbf{Y}^{(i)}(D) &= [\mathbf{X}(D)] \begin{bmatrix} \mathbf{G}_0^{(0)}(D) & \mathbf{G}_0^{(1)}(D) & \dots & \mathbf{G}_0^{(n-1)}(D) \\ \mathbf{G}_1^{(0)}(D) & \mathbf{G}_1^{(1)}(D) & \dots & \mathbf{G}_1^{(n-1)}(D) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_{k-1}^{(0)}(D) & \mathbf{G}_{k-1}^{(1)}(D) & \dots & \mathbf{G}_{k-1}^{(n-1)}(D) \end{bmatrix}
\end{aligned}
\tag{6.4}$$

The matrix  $\mathbf{G}(D)$  is called the transfer-function matrix. The number of rows represents the  $k$  input streams and the number of columns represents the  $n$  output streams. If the input stream to Fig. 6.1 is  $\mathbf{x} = (101)$  the corresponding D-transform is  $\mathbf{X} = 1+D^2$ .

The transfer-function matrix for Fig. 6.1 is

$$\mathbf{G}(D) = \begin{bmatrix} G^{(0)} & G^{(1)} \end{bmatrix} = \begin{bmatrix} 1+D+D^2 & 1+D^2 \end{bmatrix}
\tag{6.5}$$

The D-transform of the output coded bits are

$$\begin{aligned}
\mathbf{Y}(D) &= \mathbf{X}(D) \mathbf{G}(D) = [1+D^2] \bullet [1+D+D^2 \quad 1+D^2] \\
\mathbf{Y}(D) &= [1+D+D^2+D^2+D^3+D^4 \quad 1+D^2+D^2+D^4] \\
\mathbf{Y}(D) &= [1+D+D^3+D^4 \quad 1+D^4]
\end{aligned}
\tag{6.6}$$

since the arithmetic here is based on Galois fields of size  $p$ , the addition and multiplication are modulo  $p$ , in this case  $p$  is two. Given

$$\mathbf{Y}(D) = (\mathbf{Y}^{(0)}(D), \mathbf{Y}^{(1)}(D), \dots, \mathbf{Y}^{(n-1)}(D))
\tag{6.7}$$

Inverting the transform yields

$$\mathbf{y}^{(0)} = (11010)$$

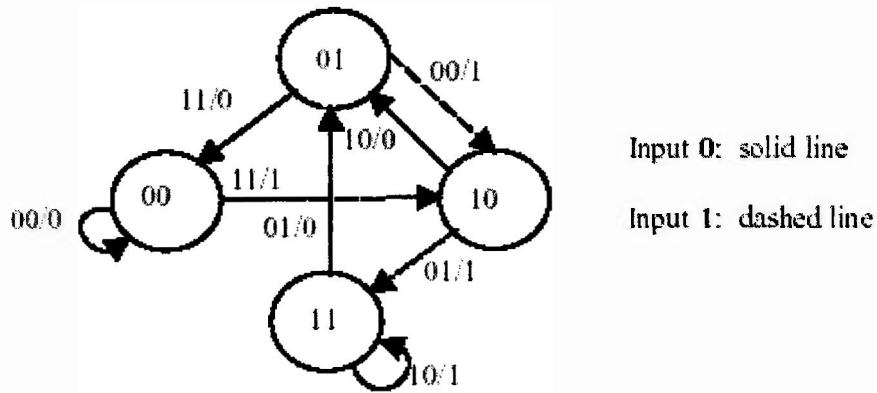
$$\mathbf{y}^{(1)} = (10001)$$

Thus, the output code word for  $\mathbf{y} = (01, 10, 00, 10, 11)$ .

## 6.1 Structural Properties of Convolutional Codes

The techniques used to analyze and compare block codes does not work so well when it comes to convolutional codes. A considerable amount of the analysis of block codes resides in obtaining a fixed length codeword to determine the minimum distance. Convolutional codes are somewhat different in that the encoder can generate code words of arbitrary length. There are three popular methods to describing the performance of convolutional codes: the tree diagram, the trellis diagram, and the state diagram.

The convolutional encoder is a state machine. It contains memory elements whose contents determine the mapping between the next set of input and output bits. Fig. 6.2 below is the state diagram for the encoder in Fig. 6.1.



**Figure 6.2** State diagram for the encoder in Figure 6.1

As with most finite-state machines, the encoder only can move between states in a limited manner. Each branch in the state diagram has a label of the form  $XX/Y$ , where  $XX$  is the output pair corresponding to the input bit  $Y$ . The distance properties and the error rate performance of a convolutional code can be obtained from its state diagram.

Another benefit of the state diagram is that it gives performance measures that are considered key in comparing convolutional codes: the minimum free distance, denoted by  $d_{free}$ . Convolutional code words are linear, therefore, there exists a subspace  $C = \{C_i, C_j, \dots, C_m\}$  in which any two code words  $C_i$  and  $C_j$  added together produce another code word that exists in the subspace  $C$ . The number of places in which two code words differ is referred to as the Hamming distance between the two codewords.

The minimum free distance, denoted  $d_{free}$ , is the minimum Hamming distance between all pairs of code words. In relation to the state diagram,  $d_{free}$  is the minimum Hamming distance between any two different paths of any length  $L$ , where the paths begin in the same state<sup>(i)</sup> and end in the same state<sup>(j)</sup> where  $i$  need not equal  $j$ . The minimum distance is an important metric because it inherently gives the error correcting power of the codeword. As stated previously, in order to correct errors within a codeword the received code word must not land on another code word. The value of  $d_{free}$  gives a measure of how many bits can be “flipped” in order for the

received code word not to be a given code word in the subspace  $C$ . It is important to note that the value of  $d_{free}$  increases as the constraint length increases. Daut derived a simple upper bound the minimum free distance of a rate  $1/n$  convolutional code. It is given by

$$d_{free} \leq \min_{r \geq 1} \left\lfloor \frac{2^{r-1}}{2^r - 1} (K + r - 1)n \right\rfloor \quad (6.8)$$

where  $\lfloor \cdot \rfloor x$  denotes the largest integer contained in  $x$ ,  $K$  is the constraint length,  $r$  is the number of input bits, and  $n$  is the number of encoded output bits. It should be evident by (6.8) that  $d_{free}$  increases if either the constraint length increases or the code rate  $R_c$  decreases. These factors must be carefully examined on a system level because they affect the overall system performance and the implementation complexity of the Viterbi decoder.

## 6.2 Viterbi Algorithm

In 1967 Andrew Viterbi proposed an algorithm as an approach to the decoding of convolutional codes. Shortly after, Forney showed that the Viterbi algorithm is a maximum-likelihood (ML) decoding algorithm for convolutional codes. In 1979, Cain, Clark, and Geist showed that the complexity of the Viterbi algorithm could be greatly simplified through puncturing. The Viterbi algorithm makes for an efficient implementation of the maximum likelihood sequence detection algorithm. Fundamentally the algorithm determines the most likely path taken given a received sequence.

This example used here is for hard decision decoding because it simplifies the decoding to minimum Hamming distance decoding, thus simplifying the explanation. Let the length of a given path be  $B=L/k$  branches, where  $L$  is the length of the information sequence and  $k$  the number of input bits into the decoder. Also, let  $n$  be the number of coded bits per branch. Define the Hamming distance



$$d_j^{(i)} = \sum_{m=1}^n v_{jm} \oplus c_{jm}^{(i)}, j = 1, 2, \dots, B, m = 1, 2, \dots, n \quad (6.9)$$

as the metric between the received sequence  $y$  and a candidate sequence  $C^{(i)}$ ,  $i=1,2,\dots,2^L$  on the  $j$ th branch. The Hamming path metric between the received sequence  $y$  and a candidate sequence  $C^{(i)}$  is,

$$d^{(i)} = \sum_{j=1}^B d_j^{(i)} = \sum_{j=1}^B \sum_{m=1}^n v_{jm} \oplus c_{jm}^{(i)} \quad (6.10)$$

The value  $d^{(i)}$  can be considered as the Hamming distance between the received vector and the candidate sequence. Consider computing the path metric on a branch-by-branch basis for a candidate path  $C^{(i)}$ :

$$d^{(i)} = \underbrace{\sum_{j=1}^B d_j^{(i)}}_{\text{Total Path Metric}} = \underbrace{\sum_{j=1}^J d_j^{(i)}}_{\text{Left of } J} + \underbrace{\sum_{j=J+1}^B d_j^{(i)}}_{\text{Right of } J} \quad (6.11)$$

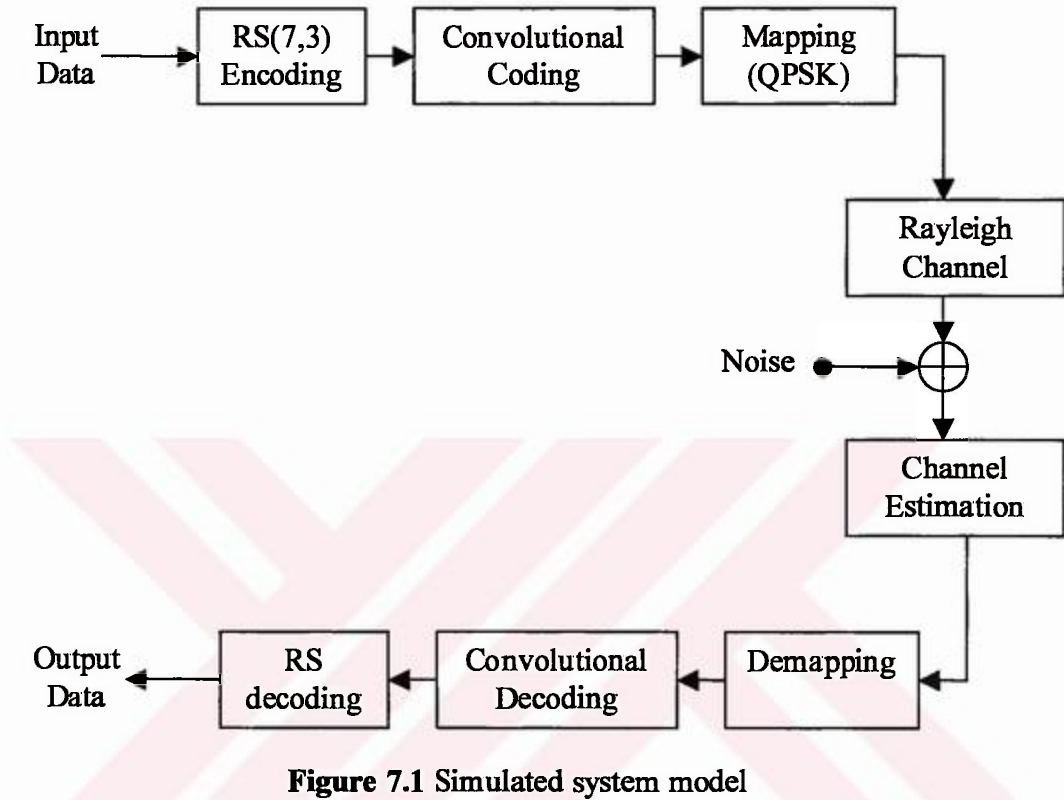
At any arbitrary time  $J$ , the path metric can be broken into two equations, the partial path metric to the left of time  $J$  and the partial path metric to the right of  $J$ . Suppose two candidate paths  $C^{(1)}$  and  $C^{(2)}$  merge at time  $J$  and share a common path for  $j \geq j+1$ . If  $C^{(2)}$  has a smaller path metric than  $C^{(1)}$  for  $j \leq J$  then  $C^{(2)}$  will always have a smaller path metric than  $C^{(1)}$ . Thus,  $C^{(2)}$  is called the survivor path and  $C^{(1)}$  is excluded as a candidate path. For each of the  $2^{k(K-1)}$  states at time  $J$ , store the list of transitions in survivor path and the partial path metric of each survivor path. After a predetermined amount of time, go back through the trellis along the survivor path until the all-zero state is reached. This is the optimal path and the input bit sequence corresponding is the maximum likelihood decoded sequence.

### 6.3 Punctured Convolutional Codes

Since the complexity of Viterbi decoding is exponential in the number of input symbols as  $k$  gets large implementation complexity becomes difficult. Classes of codes called punctured convolutional codes were introduced by Cain and Clark in 1979. By periodically deleting bits via a puncturing matrix these codes allow for higher rate codes, which give a higher coding gain while not suffering the implementation penalty from a large value of  $k$ . If the encoder structure is a lower rate code  $1/n$ , then there are only  $2^k$  computations for each node at the decoding trellis, which is suitable for practical implementations.



## 7 SIMULATION



**Figure 7.1** Simulated system model

First random integers ( $n=4096*3=12288$  symbols) are generated as input data. Then using (7,3) Reed-Solomon encoding, input data is encoded. This code is double error correcting code. By encoding 4 redundant symbols are added to each 3 symbols. Then converting decimal integers to binary, convolutional coding is used. In convolutional coding the code rate is equal to  $R_c=k/n=1/2$ . The rate of the encoder is determined by the fact that the encoder outputs two bits for every one bit at the input. Then the bits are mapped onto QPSK symbols. Now we have 86016 symbols. The symbols are grouped to blocks. Every block has 1024 symbols. For every block channel and noise affect is like below:

$$H_k = \sum_{i=1}^{30} \rho_i e^{j(\theta_i - 2\pi(k/N T_s)\tau_i)} \quad (7.1)$$

$$R(k,n)=H(k)A(k,n)+N(k,n) \quad k=0,1,\dots,1023 \quad (7.2)$$

The dispersive channel is chosen as a multipath Rayleigh model which is suitable for wireless systems operating at the outdoor dispersive environment. We assume that the channel response is only slowly time-varying with respect to the symbol period. That is, it is assumed that the channel is quasi-stationary and its impulse response stay constant throughout all of the symbols.

The following multipath model was employed for the channel impulse response for the duration of  $L_0$  frames.

$$h(t)= \sum_{i=1}^{K_p} A_i \delta(t-T_s), \quad A_i = \rho_i e^{j\theta_i} \quad (7.3)$$

where  $\rho_i$  and  $\theta_i$  are the amplitude and phase of the path associated with the delay  $\tau_i$  and  $K_p$  is the number of paths. The random variables  $\{A_i\}$  are zero-mean complex-valued Gaussian and are mutually independent. The random independent delays  $\{\tau_i\}$  are generated so as to provide an exponential power delay profile with an average delay  $\tau_{av}$  and a maximum delay  $\tau_{max}$ . Parameters' values are:  $K_p=30, \tau_{av}=5\mu s$  and  $\tau_{max}=20\mu s$ . The values of  $\{\tau_i\}, \{\theta_i\}$  and  $\{\rho_i\}$  for the channel are listed in Table. In our simulation, information bits are mapped onto QPSK symbols. The symbol interval is chosen to be  $T_s=0.167\mu s$ .

Only for the first block, 512(or 256) of 1024 symbols are assumed known symbols (pilot symbols). Then the channel estimation is made as below:

$$\begin{aligned} R(k,n) &= H(k)A(k,n) + N(k,n) \\ \hat{H}_p(k) &= R(k,n)/A(k) \end{aligned} \quad (7.4)$$

For 512(or 256) symbols,  $\hat{H}_p(k)$  is computed. Then using an interpolation technique, channel is estimated for all 1024 symbols. For all blocks estimated channel parameters are used.

$$A(k,n)=R(k,n)/\hat{H}_p(k) \quad (7.5)$$

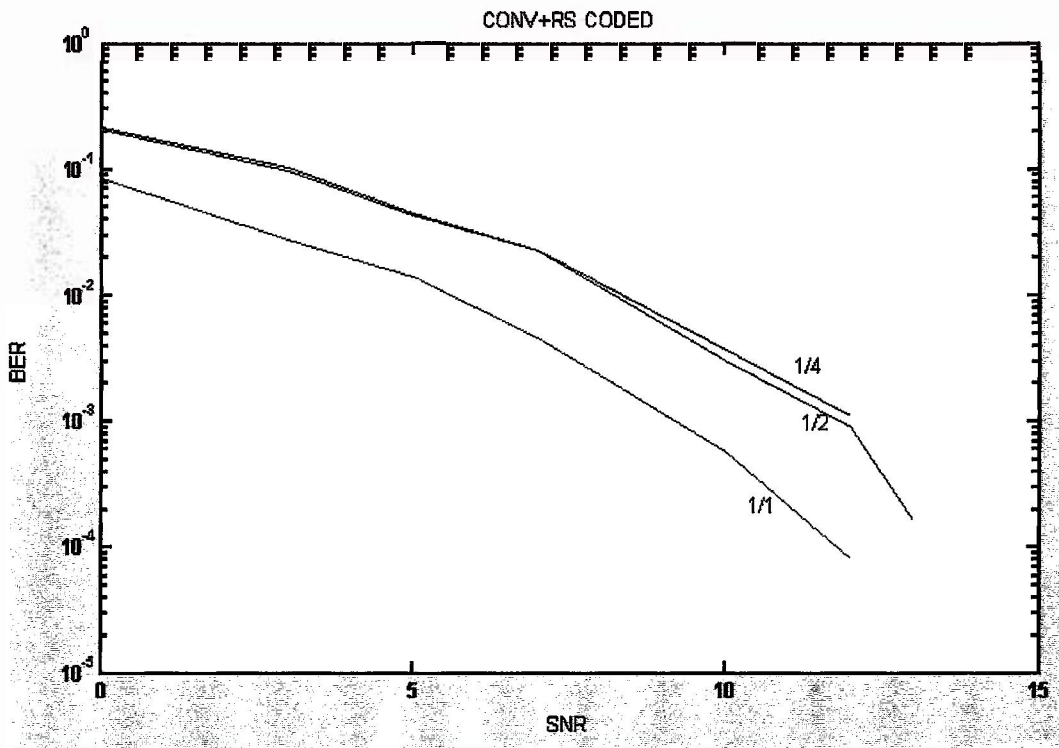
After demapping convolutional decoding is used. Then using RS decoding the output data is obtained. In RS decoding Berlekamp-Massey algorithm is used. For different symbol to noise ratios(SNR), symbol error rates(SER) are shown in Figures.



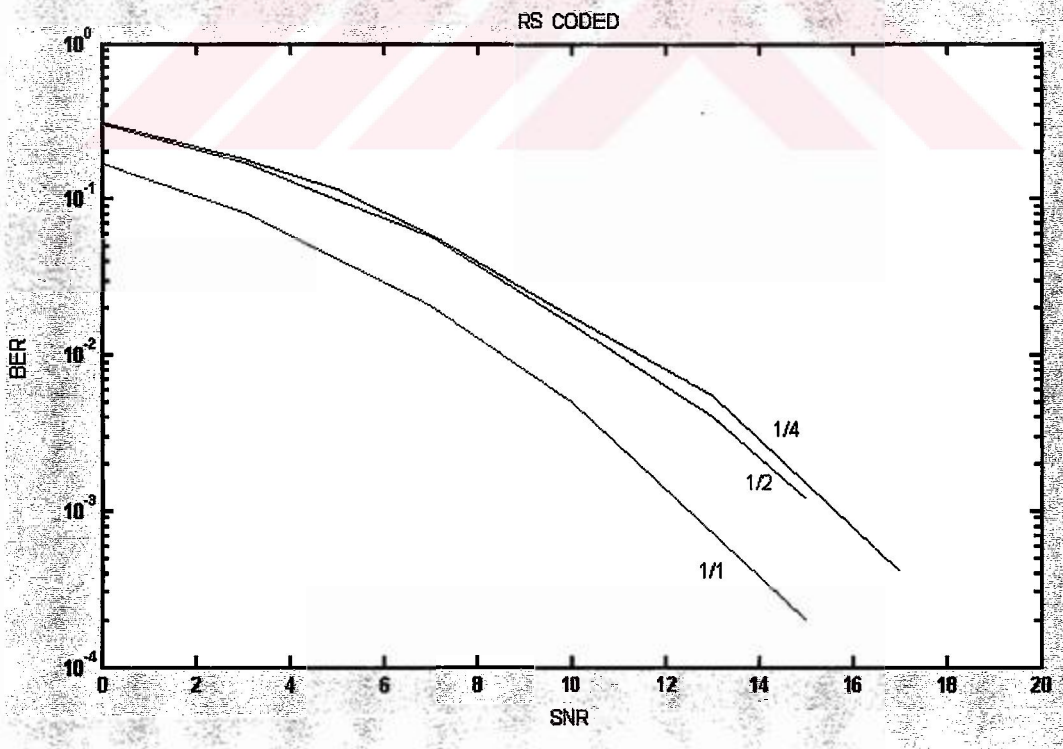
**Table 7.1: Parameters of the Channel Model**

$i$	Delay $\tau_i$ ( $\mu\text{s}$ )	Amplitude $\rho_i$	Phase $\theta_i$ (rad)
1	0.0120	0.4213	5.9010
2	0.2892	0.1543	0.2147
3	0.5593	0.4401	3.9968
4	0.6919	0.4380	4.6862
5	1.0266	0.1864	4.4331
6	1.2347	0.0669	1.1484
7	1.3056	0.0809	4.0282
8	1.9643	0.1647	3.3214
9	2.0906	0.1503	4.0649
10	2.3076	0.1714	3.8432
11	2.3907	0.1289	2.8815
12	2.8962	0.2123	2.8152
13	3.7334	0.3531	5.0859
14	3.7415	0.0982	6.2326
15	3.7630	0.0808	0.7662
16	4.0452	0.1157	5.6671
17	5.4348	0.2199	2.3719
18	5.5246	0.2016	6.0266
19	5.9653	0.1288	5.1854
20	6.6460	0.2004	1.1537
21	6.8295	0.2102	1.3142
22	7.5086	0.2630	4.4436
23	7.9602	0.1199	6.0964
24	8.2400	0.3210	5.0876
25	8.8824	0.1907	1.4835
26	9.7827	0.2379	4.7438
27	10.1142	0.1800	0.1396
28	11.1587	0.2539	1.8221
29	17.6513	0.2767	1.7052
30	18.3765	0.1208	5.3582

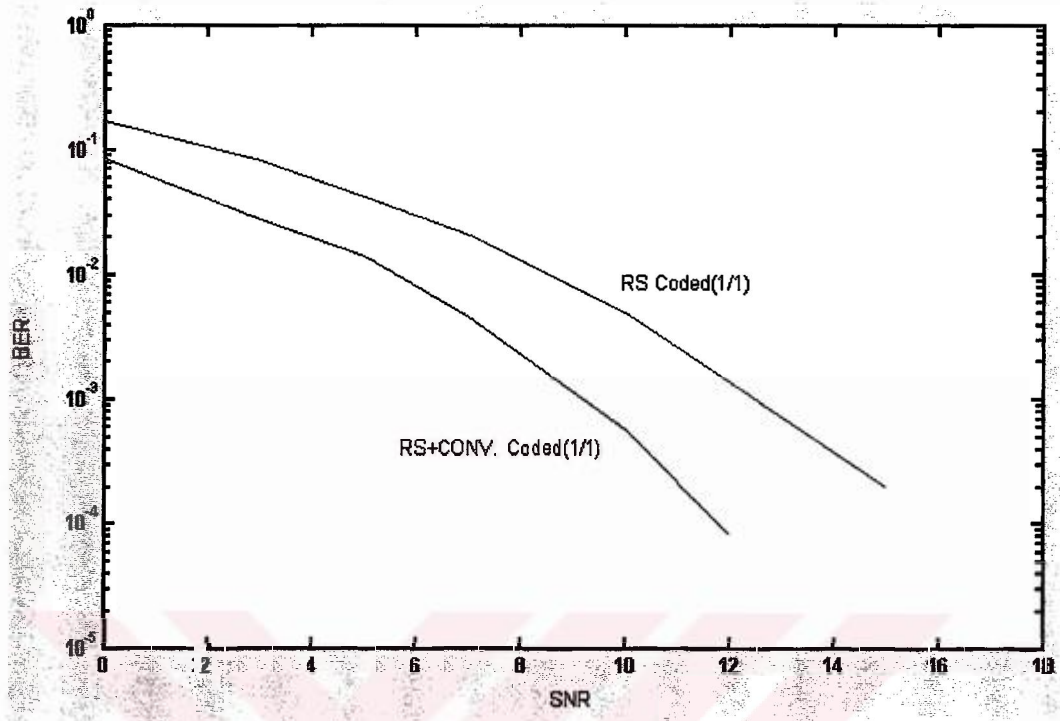




**Figure 7.2** Simulation results of RS and convolutional coded system for pilot rates: 1/4, 1/2, 1/1



**Figure 7.3** Simulation results of RS coded system for pilot rates: 1/4, 1/2, 1/1



**Figure 7.4** Simulation results of RS and convolutional coded system for pilot rate: 1/1

## 8 CONCLUSION

In almost all applications of multi-carrier modulation, satisfactory performance cannot be achieved without the addition of some form of coding. In this thesis, Reed-Solomon and Convolutional coded OFDM system is designed. First RS coded system is simulated. The results are shown in Figure 7.3.

Combining convolutional and block codes in a concatenated code is particularly powerful technique. The block code is the outer code, that is it is applied first at the transmitter and last at the receiver. The inner convolutional code is very effective at reducing the error probability. However when a convolutional code does make an error, it appears as a large burst. This occurs when the Viterbi algorithm chooses a wrong sequence. The outer block code, especially Reed-Solomon code is very effective in correcting that burst error. So, then both convolutional coding and RS coding are used. Simulation results are given in Figure 7.2. RS coded system is compared with RS+Convolutional coded system. It can be seen that from Figure 7.4, concatenated system has better performance.

Channel affect changes amplitudes and phases of the symbols. So channel estimator has to estimate this differences. One way to do this is to use a channel estimator that estimates the reference values based on known pilot values. Based on these pilots, all other reference values can be estimated by performing an interpolation technique. In simulations only for the first block  $1/4, 1/2$  and  $1/1$  of the 1024 symbols are assumed known symbols (pilot symbols). It can be seen that from Figure 7.2 and Figure 7.3, to use more pilot symbol gives better performance.

## REFERENCES

- [1] Richard Van Nee, Ramjee Prasad OFDM for Wireless Multimedia Communications
- [2] Ahmad R.S. Bahai, Burton R. Saltzberg Theory and applications of OFDM
- [3] O.Edfors, M.Sandell, J.J.van de Beek, D.Landström, F.Sjöberg An introduction to OFDM
- [4] A.K.Michelson, A.H.Levesque Error Control Techniques For Digital Communications
- [5] Stephen B. Wicker, Vijay K. Bhargava Reed-Solomon Codes and Their Applications
- [6] L.H.Charles Lee Error-Control Block Codes for Communications Engineers
- [7] P.Sweeney Error Control Coding:an Introduction
- [8] S.Lin, D.J.Costello Error Control Coding Fundamentals and Applications
- [9] C.L.Taylor Punctured Convolutional Coding Scheme for Multi-Carrier Multi-Antenna Wireless Systems
- [10] IEEE Potentials February/March 2001 Error Control Coding *Pages:26-28*
- [11] Weinstein, S.B., Ebert P.M. Data Transmission By Frequency Division Multiplexing Using the Discrete Fourier Transform.
- [12] Alard M., Lassalle R. Principles of Modulation and Channel Coding for Digital Broadcasting for Mobile Receivers.
- [13] Parhi, K. K., T. Nishinati, Digital Signal Processing for Multimedia Systems
- [14] ETSI, Radio Broadcasting Systems: Digital Audio Broadcasting to Mobile, Portable and Fixed receivers.
- [15] Reimers, U., DVB-T: The COFDM-Based System for Terrestrial Television