Chapter

Polynomials in Error Detection and Correction in Data Communication System

Charanarur Panem, Vinaya Gad and Rajendra S. Gad

Abstract

The chapter gives an overview of the various types of errors encountered in a communication system. It discusses the various error detection and error correction codes. The role of polynomials in error detection and error correction is discussed in detail with the architecture for practical implementation of the codes in a communication channel.

Keywords: error detection, error correction, burst error, channel coding, channel decoding, CRC, LDPC

1. Introduction

Different types of errors are encountered during data transmission because of physical defects in the communication medium as well as environmental interference. Environmental interference and physical defects in the communication medium can cause random bit errors during data transmission. Error coding is a method of detecting and correcting these errors to ensure that there are no errors in the information when it is sent from source to destination. Error coding is used for error-free communication in the primary and secondary memory devices such as RAM, ROM, hard disk, CD's, and DVDs, as well as in different digital data communication systems such as network communication, satellite, and cellular communication and deep space combination.

1.1 Need for error coding

Data transmission errors occur in terrestrial mobile communication due to multipath fading, diffractions or scattering in cellular wireless communications, low signal-to-noise ratio, and limited transmitted power and energy resources in satellite communication [1].

Error coding uses mathematical formulae to encode data bits at the source into longer bit words for transmission. The "code word" is then decoded at the destination to retrieve the information. The code word consists of extra bits, which provide redundancy, and at the destination, it will decode the data to find out whether the communication channel introduced any error and some schemes can even correct the errors so that there is no need to resend the data.

There are two ways to deal with errors. One way is to introduce redundant information along with the data to be transmitted, which will enable the receiver to deduce the information that has been transmitted. The second way is to include only enough redundancy to allow the receiver to detect that error has occurred, but not which error and the receiver makes a request for retransmission. The first method uses Error-Correcting Codes and the second uses Error-detecting Codes.

Consider a frame having **m** data bits (message to be sent) and **r** redundant bits (used for checking). The total number of bits in the frame will be **n(m + r)**, which is referred as n-bit code word. Consider two code-words, 11,001,100 and 11,001,111, and perform Exclusive OR and then count number of 1's in the result. The number of bits in which the codewords are different is called Hamming distance. Suppose the code words are Hamming distance d- apart, it will require d single-bit errors to connect one code word to another. The properties of error detection and error correction depend on the Hamming distance.

- A distance (d + 1) code is required to detect d errors because d-single bit errors cannot change a valid codeword into another valid code. Thus the error is detected at the receiver.

- A distance (2d + 1) code is required to correct d errors because the codewords will be so apart that the transmitted codeword will be still closer than any other valid codeword, and thus the error can be determined.

1.2 Types of errors in a communication channel

When the data travels from the sender to receiver, different types of errors are encountered in the communication channel [2].

1.2.1 Noise or electrical distortion

When the data travel through a conductor, there are different influences such as sound waves, electrical signals, noise such as electricity from motors, power switches, impulse noise, because of which data can be corrupted or destroyed. Old conductors are unable to handle these types of interference and heavy data traffic, hence the data transmission suffers.

1.2.2 Burst errors

Burst errors are large clumps of bits and occur when there are a number of interconnected bit errors which occur at many places. These types of errors may occur because of some wrong placement in the data chain. It may contain several hundred or thousand-bit errors.

1.2.3 Random bit errors

Data sent on a communication channel consists of thousands of data bits, sent in a particular order or sequence. However, there is a probability that the bits may be rearranged by accident in the transmission process. These types of errors are known as random bit errors.

1.2.4 Cross talk and echo

Cross talk occurs when the transmission cable through which the data is transmitted, is surrounded by other transmission lines. The data and code words, which

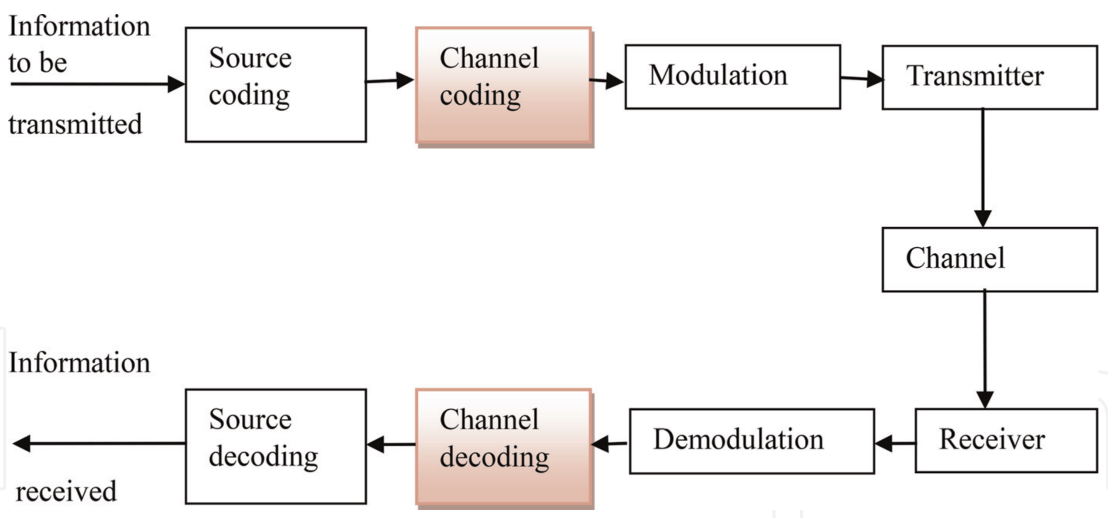

Figure 1.
Wireless communication system with channel coding.

are traveling in the neighboring line crosses over and gets superimposed on the transmission cable. Echo is similar to cross talk; how ever, it occurs in a single transmission line, through which multiple computer ports are sending data at the same time. The data from one port will echo into another, thus resulting in data corruption (**Figure 1**).

2. Error detecting codes

Error detection uses additional bits in the message to be transmitted. This adds redundancy and facilitates detection and correction of errors. Popular techniques of error detection are,

- Simple parity check.
- Two-dimensional parity check.
- Checksum.
- Cyclic redundancy check.

2.1 Simple parity checking or one-dimension parity check

This technique is most common and cheap mechanism for detection. The data unit is appended with a redundant bit known as the parity bit. A parity bit generator is used, which adds 1 to the block of data if it contains odd number of 1's, and 0 is added if there are even number of 1's. At the receiver end, the parity is computed of the block of data received and compared with the received parity bit. These scheme uses total even number of 1's; hence it is known as even parity checking. Similarly, you can use odd number of 1's, known as odd parity checking.

2.2 Two-dimension parity check

Two-dimensional parity check improves the performance. Here, the data bits are organized in the form of a table, computed for each row as well each column and are sent along with the data. The parity is computed for the received data and compared with the received data bits.

2.2.1 Performance

Two-dimension parity checking is mainly used to detect burst errors. It detects a burst error of more than **n** bits with a high probability. However, this mechanism will not be able to detect the errors if two bits in one data unit are damaged. Example if 11000110 is changed to 01000100 and 10101010 is changed to 00101000 the error will not be detected.

2.3 Checksum

This scheme divides the data bits to be sent into **k** segments. Each segment consists of **m** bits. All the segments are added using 1's complement arithmetic. Checksum is obtained by complementing the sum, and the data segments are transmitted together. At the receiver end, again 1's complement arithmetic is used to add all received segments. The sum generated is complemented. The receiver accepts the data if the result of complementing is zero.

2.3.1 Performance

The checksum mechanism detects all errors consisting of odd number of bits. It also detects most errors having even number of bits.

2.4 Cyclic redundancy check (CRC)

Cyclic redundancy check is the most powerful and easy to implement error detection mechanism. Checksum uses addition, whereas CRC is based on binary division. In CRC, the data unit is appended at the end by a sequence of redundant bits, called cyclic redundancy check bits. The resulting data unit is exactly divisible by a second, predetermined binary number. At the receiver end, the incoming data unit is divided by the same predetermined binary number. If the remainder is zero, the data unit is assumed to be error-free and is accepted. A remainder indicates that the data unit has encountered an error in transit and therefore is rejected at the receiver. The generalized technique to generate the CRC bits is explained below:

Consider there is a **k** bit message to be transmitted. The transmitter generates an **r**-bit sequence called as FCS (frame check sequence). These **r** bits are appended to the **k** bit message, so that **(k + r)** bits are transmitted. The **r**-bit FCS is generated by dividing the **k** bit message, appended by **r** zeros, by a predetermined number. This number is **(r + 1)** bit length, and can be considered as coefficient of a polynomial, called generator polynomial. The **r**-bit FCS is generated as the remainder of binary division. Once the **(k + r)** bit frame is received, it is divided by the same predetermined number. If the remainder is zero, it means there was no error, and the frame is accepted by the receiver.

Operations at both the sender and receiver end are shown in **Figure 2**.

CRC is widely used in data communications, data storage, and data compression as a powerful method for detecting errors in the data. It is also used in testing of integrated circuits and the detection of logical faults. A cyclic redundancy code is a non-secure hash function designed to detect accidental changes to raw computer data. CRCs are popular because they are simple to implement in binary hardware, are easy to analyze mathematically, and are particularly good at detecting common

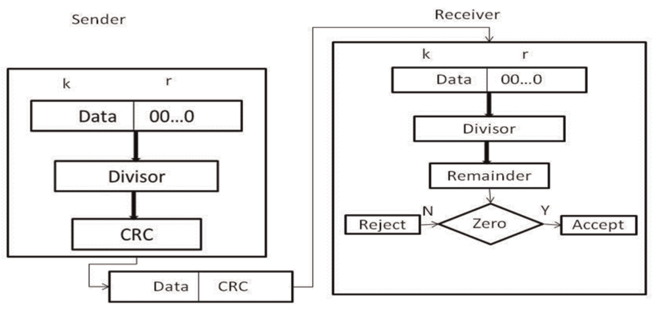

Figure 2.
Basic scheme for cyclic redundancy checking.

errors caused by noise in transmission channels. CRC-32 guarantees 99.999% probability of error detection at the receiver end; hence, this CRC is often used for Gigabit Ethernet packets [3].

Cyclic redundancy codes are a subset of cyclic [4, 5] codes that are also a subset of linear block codes. They use a binary alphabet, 0 and 1. Arithmetic is based on Galois Field GF(2), for example, modulo-2 addition (logical XOR) and modulo-2 multiplication (logical AND). The CRC method treats the data frame as a large Binary number. This number is then divided (at the generator end) by a fixed binary number (the generator polynomial) and the resulting CRC value, known as the FCS (Frame Check Sequence), is appended to the end of the data frame and transmitted. The receiver divides the message (including the calculated CRC), by the same polynomial used during transmission and compares its CRC value with the generated CRC value. If it does not match, the system requests for re-transmission of the data frame.

CRC codes are often used for error detection over frames or vectors of a certain length. The frame can be expressed as a polynomial in x, where the exponent of x is the place marker of the coefficient. The vector $a = a_{L-1}a_{L-2}....a_1a_0$ length L is represented by the degree L-1 polynomial.

$$a(x)=\sum_{i=0}^{L-1} a_i x^i = a_{L-1}x^{L-1} + a_{L-2}x^{L-2} + ... + a_1 x + a_0 \tag{1}$$

CRC coding is a generalization of the parity check bit. Parity bits are used for short vectors to detect one-bit error. However, if there are errors in two-bit positions, it will not detect the error.

2.4.1 Error detection procedure

Let the data to be transmitted consist of a length k binary vector, and represent it by the degree k-1 polynomial.

$$d(x)=d_{k-1}x^{k-1} + d_{k-2}x^{k-2} + ... + d_1 x + d_0 \tag{2}$$

Then, to add redundant bits, so the total length of the code word is n, we should add n-k bits. These redundant bits, which are the CRC bits, can be represented by the degree n-k-1 polynomial.

$$r(x)= d_{n-k-1}x^{n-k-1} + ... + r_1 x + r_0 \tag{3}$$

The polynomial for codeword is written as follows:

$$\begin{aligned} c(x) &= d(x)x^{n-k} + r(x) \\ &= d_{k-1}x^{n-1} + \dots + d_1x^{n-k+1} + d_0x^{n-k} + r_{n-k-1}x^{n-k-1} + \dots + r_1x + r_0 \end{aligned} \tag{4}$$

CRC polynomial is derived using a degree n-k generator polynomial.

$$g(x) = x^{n-k} + g_{n-k-1}x^{n-k-1} + \dots + g_1x + 1 \tag{5}$$

which is a binary polynomial, wherein the highest and lowest coefficients are non-zero ($g_{n-k} = 1$ and $g_0 = 1$).

The CRC polynomial is derived as.

$$r(x) = R_{g(x)}(d(x)x^{n-k}) \tag{6}$$

All coefficients of the polynomial are binary, and modulo-2 arithmetic is used [4].

To see how the receiver side can use this code word to detect errors, we first need to derive some properties of it. Let $z(x)$ denote the quotient in the division $d(x)x^{n-k} / g(x)$ hence, following is the data polynomial.

$$d(x)x^{n-k} = g(x)z(x) + r(x) \tag{7}$$

In modulo-2 arithmetic, addition and subtraction are alike, and the codeword polynomial can be written as.

$$c(x) = d(x)x^{n-k} + r(x) = g(x)z(x) \tag{8}$$

This gives rise to the following theorem [5].

A polynomial $c(x)$ with deg.$(c(x)) < n$ is a code word if and only if $g(x)jc(x)$. If $c(x)$ is transmitted over a channel and there occur errors, they can be represented by an addition of the polynomial $e(x)$, and the received polynomial is.

$$y(x) = c(x) + e(x) \tag{9}$$

Thus $g(x)$ is a factor of each transmitted codeword which can be used by the receiver to detect the error. The error is detected if $g(x)$ is not a factor. To check this, the remainder of the division $c(x) = g(x)$ is derived as.

$$\begin{aligned} s(x) &= R_{g(x)}(y(x)) = R_{g(x)}(c(x) + e(x)) \\ &= R_{g(x)}(R_{g(x)}(c(x)) + R_{g(x)}(e(x))) = R_{g(x)}(e(x)) \end{aligned} \tag{10}$$

This quantity is known as Syndrome. It is directly a function of the error since $R_{g(x)}(c(x)) = 0$. The syndrome plays an important role in coding theory.

2.4.2 Performance

CRC is a very effective and popular error detection technique. The error detection capabilities of CRC depend on the chosen generator polynomial.

- CRC has capacity to detect all single-bit errors.
- CRC has capacity to detect all double-bit errors (three 1's).
- CRC has capacity to detect any odd number of errors (X + 1).
- CRC has capacity to detect all burst errors of less than the degree of the polynomial.
- CRC has the capacity to detect most of the larger burst errors with a high probability.

2.4.3 Implementation

n-bit CRC can be calculated as CRC = Rem [M(x) * (xn/G(x)) J; where M(x) denotes the message polynomial, G(x) denotes the generator polynomial and n is the degree of polynomial G(x). CRC can be calculated using serial or parallel method. **Figure 3** shows the serial data input hardware implementation. The data message input is denoted as Din, clk is used to denote the clock used for the circuits. XOR gates are used before the input of each flip-flop. The output can be obtained from any input or output wire of any flip-flop.

Parallel implementation of CRC is shown in **Figure 4**. The data message input is to be XOR-ed with a calculated input. The calculated input can be obtained by using matrix method [6]. State equation for LFSRs can be written as: X(i + 1) = Fm. X (i) + H.D(i); where Xi is the ith state of register and X(i + 1) is the (i + 1)th state of

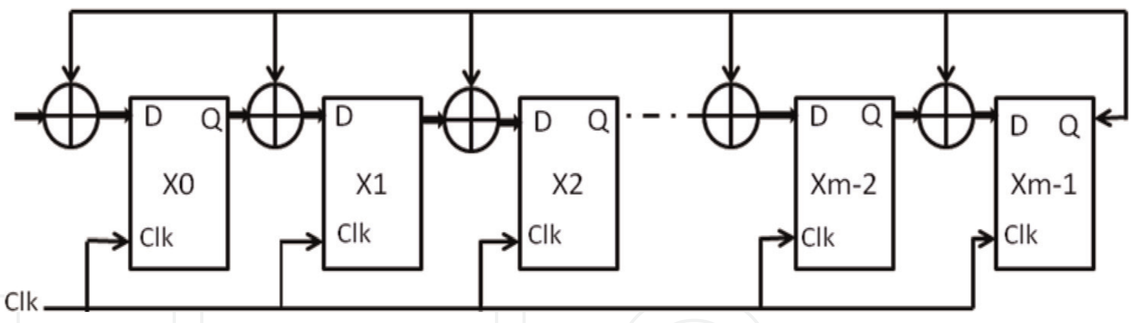

Figure 3.
Serial CRC.

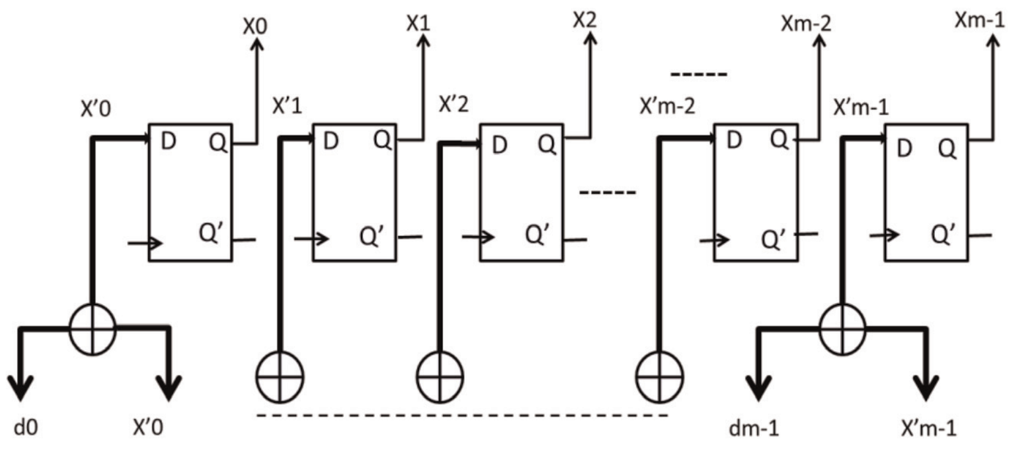

Figure 4.
Parallel CRC.

the register, D(i) is the ith serial input bit, Fm is a m x m matrix and H is a 1 x m matrix. Consider the generator polynomial G = {gm, gm-1. - - -,go}.

$$F^m = \begin{bmatrix} g_{m-1} & 1 & 0 & - & - & 0 \\ g_{m-2} & 0 & 1 & - & - & 0 \\ | & - & - & - & - & - \\ | & - & - & - & - & - \\ g_1 & 0 & 0 & - & - & 1 \\ g_0 & 0 & 0 & - & - & 0 \end{bmatrix} \tag{11}$$

$$H = [0 \quad 0 \quad - \quad - \quad - \quad 0 \quad 1 \quad]^T \tag{12}$$

$$X'_{m-1} = (g_{m-1}, X_{m-1}) \oplus X_{m-2} \tag{13}$$

$$X'_{m-2} = (g_{m-2}, X_{m-1}) \oplus X_{m-3} \tag{14}$$

$$X'_1 = (g_1, X_{m-1}) \oplus X_0 \tag{15}$$

$$X'_0 = (g_0, X_{m-1}) \oplus d \tag{16}$$

The above equations are used for serial computation of CRC. Following equation are used for parallel computation of CRC.

$$X'_{m-1} = (F^m(m-1)(m-1), X_{m-1}) \oplus (F^m(m-1)(m-2), X_{m-2}) \ldots. \oplus (F^m(m-1)(0), X_0 \oplus d_{m-1}) \tag{17}$$

$$X'_{m-2} = (F^m(m-2)(m-1), X_{m-1}) \oplus (F^m(m-2)(m-2), X_{m-2}) \ldots\ldots \oplus (F^m(m-2)(0), X_0 \oplus d_{m-2}) \tag{18}$$

$$X'_0 = (F^m(0)(m-1), X_{m-1}) \oplus (F^m(0)(m-2), X_{m-2}) \ldots\ldots \oplus (F^m(0)(0), X_0 \oplus d_0) \tag{19}$$

Table 1 summaries the commonly used polynomials in different applications and **Table 2** gives a list of primitive polynomials.

Polynomial name	Polynomial	Use
CRC-1	$x+1$	Parity
CRC-4-ITU	x^4+x+1	ITU G.704
CRC-5-ITU	$x^5+x^4+x^2+1$	ITU G.704
CRC-5-USB	x^5+x^2+1	USB
CRC-6-ITU	$x^6+x\ \ +1$	ITU G.704
CRC-7	x^7+x^3+1	Telecom systems, MMC
CRC-8-ATM	x^8+x^2+x+1	ATM HEC
CRC-8-CCITT	$x^8+x^7+x^3+x^2+1$	1-Wire bus
CRC-8-Maxim	$x^8+x^5+x^4+1$	1-Wire bus
CRC-8	$x^8+x^7+x^6+x^4+x^2+1$	General

Polynomial name	Polynomial	Use
CRC-8-SAE	$x^8+x^4+x^3+x^2+1$	SAE J1850
CRC-10	$x^{10}+x^9+x^5+x^4+x+1$	General
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$	Telecom systems
CRC-15-CAN	$x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1$	CAN
CRC-16-CCITT	$x^{16}+x^{12}+x^5+1$	XMODEM,X.25, V.41, Bluetooth, PPP, IrDA, CRCCCITT
CRC-16	$x^{16}+x^{15}+x^2+1$	USB
CRC-24-Radix64	$x^{24}+x^{23}+x^{18}+x^{17}+x^{14}+x^{11}+x^{10}+x^7+x^6+x^5+x^4+x^3+x+1$	General
CRC-32-IEEE802.3	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^6+x^7+x^5+x^4+x^2+x+1$	Ethernet, MPEG2
CRC-32C	$x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}$ $+x^{11}+x^{10}+x^9+x^8+x^6+1$	General
CRC-32K	$x^{32}+x^{30}+x^{29}+x^{28}+x^{26}+x^{20}+x^{19}+x^{17}+x^{16}+x^{15}+x^{11}+x^{10}$ $+x^7+x^6+x^4+x^2+x+1$	General
CRC-64-ISO	$x^{64}+x^4+x^3+x+1$	ISO 3309
CRC-64-ECMA	$x^{64}+x^{62}+x^{57}+x^{55}+x^{54}+x^{53}+x^{52}+x^{47}+x^{46}+x^{45}+x^{40}+x^{39}$ $+x^{38}+x^{37}+x^{35}+x^{33}+x^{32}+x^{31}+x^{29}+x^{27}+x^{24}+x^{23}+$ $x^{22}+x^{21}+x^{19}+x^{17}+x^{13}+x^{12}+x^{10}+x^9+x^7+x^4+x+1$	ECMA-182

Table 1.
Commonly used divisor polynomials [4, 5].

P(x)	
x^2+x+1	$x^{10}+x^3+1$
x^3+x+1	$x^{11}+x^2+1$
x^4+x+1	$x^{12}+x^6+x^4+x+1$
x^5+x^2+1	$x^{13}+x^4+x^3+x+1$
x^6+x+1	$x^{14}+x^{10}+x^6+x+1$
x^7+x^3+1	$x^{15}+x+1$
$x^8+x^4+x^3+x^2+1$	$x^{16}+x^{12}+x^9+x^7+1$
x^9+x^4+1	$x^{17}+x^3+1$

Table 2.
A list of some primitive polynomials [4, 5].

3. Error correcting codes

There are two ways to handle error correction. The first method is known as backward error correction wherein, and the receiver asks for retransmission of data when the error is discovered. The second method is known as backward error connection, where the receiver uses an error correction code to correct certain errors.

The codes required for error connection are more sophisticated compared to error detection codes and require more redundant bits. Most error correction is limited to one, two or at the most three-bit errors since it requires large number of redundant bits multiple bit error or burst errors.

Different types of error detection and correction techniques are required for specific noisy channels/media, like random error or burst error or multi-path distortion or channel effects. There are two approaches for error control coding, forward error correction (FEC) and automatic repeat request (ARQ) [7].

FEC error control is used for one-way system whereas, ECC (Error Correcting Codes) with error detection and retransmission called ARQ is used for two-way communication, such as telephone and satellite communications. The classification of FEC is shown in **Figure 5**.

3.1 Single-bit error correction

A single-bit error can be easily detected using a parity bit; however, for correcting an error, the exact position of the errored bit is required to be detected.

Hamming code is a technique developed by R.W. Hamming, which is used to find out the location of the bit which is in error, Hamming code can be used for data bits of any length and uses the relationship between data bits and redundant bits where $2r \geq d + r + 1$.

Procedure for error detection using Hamming code is as follows:

- To each group of m information bits k parity bits are added to form $(m + k)$ bit code.
- Location of each of the $(m + k)$ digits is assigned a decimal value.
- The k parity bits are placed in positions 1, 2, …, 2 k-1. k parity checks are performed on selected digits of each codeword.
- At the receiving end, the parity bits are recalculated. The decimal value of the k parity bits provide the bit-position in error if any.

Claude Elwood Shannon (1916–2001) and Richard Hamming (1915–1998), were colleagues at Bell Laboratories pioneer in coding theory. Shannon's channel coding theorem proves that if the information transmission rate is less than the channel capacity, it is possible to design an error correcting code (ECC) with almost

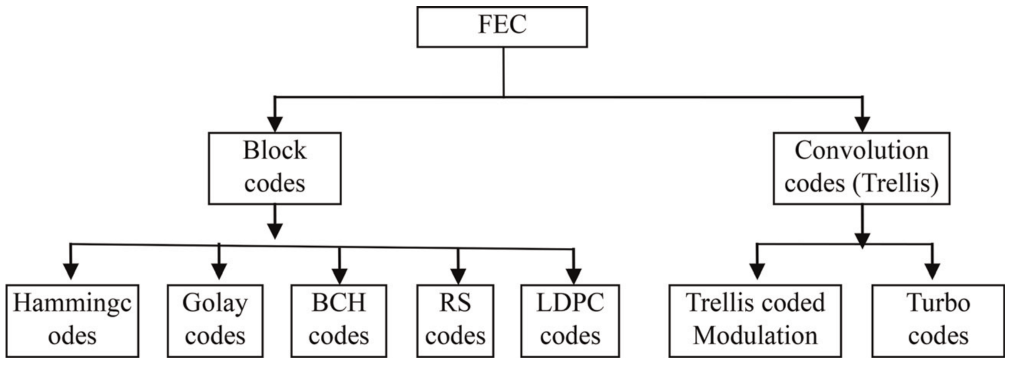

Figure 5.
Classification of FEC.

error-free information transmission. Hamming invented the first error correcting codes (ECC) in 1950. It is known as (7, 4) Hamming code.

3.2 BCH codes

The BCH code design can have a precise control over the number of symbol errors correctable by the code. Binary BCH codes can correct multiple bit errors. BCH codes are advantages since a simple algebraic method known as syndrome decoding can be used which simplifies the design of the decoder for these codes, which uses a small low-power electronic hardware.

BCH codes are used in applications such as satellite communications, compact disc players, DVDs, disk drives, solid-state drives, and two-dimensional bar codes.

BCH codes are a class of linear, cyclic codes. For a cyclic code any codeword polynomial has its generator polynomial as a factor; so the roots of the code's generator polynomial $g(x)$ are also the roots of code words. BCH codes are constructed using the roots of $g(x)$ in extended Galois field; binary primitive BCH codes, which correct multiple random errors, form an important subclass. The error correcting binary BCH code has the following parameters:

Block length $n = 2^m - 1$.

- No. of parity check bits: $n - K £ mt$.
- Minimum distance: $d_{min}\ 2\ t + 1$.

$g(x)$ generates a binary primitive BCH code ith it is the least degree polynomial over GF(2) with $\alpha, \alpha^2, \dots \dots .. \alpha^{2t}$ as roots, α being a primitive element in GF(2^m). With this $g(x)$ must have $(x + \alpha)\ (x + \alpha^2) \dots \dots\ (x + \alpha^{\ 2t})$ as a factor. This leads to g(x) of the form.

$g(x) = LCM\ [\Omega_1(x)\ \Omega_2\ (x)\ \Omega_3\ (x).. .. \Omega_i\ (x)]$.

where $\{\Omega_1(x)\ \Omega_2\ (x)\ \Omega_3\ (x).. .. \Omega_\mathrm{i}\ (x)\}$ is the smallest set of minimal polynomials with $(x + \alpha)\ (x + \alpha^2) \dots \dots\ (x + \alpha^{\ 2t})$ as a factor.

BCH codes can be encoded using similar method.

3.2.1 Decoding of BCH codes

The decoding of BCH codes involves the following steps:

i. Form the syndrome polynomial $s(x) = s0 + s1x + s2x^2 + \cdots + s_{n\text{-}K\text{-}1x}{}^{n-K-1}$ with the set $\{s_0, s_1, s_2 \dots . s_{n\text{-}K\text{-}1}\}$ being the values of $r(x)$ at α, $\alpha 2, \dots . \alpha\ 2\ t$. If $s(x)$ is zero, $r(x)$ itself is a codeword; else proceed as follows.

ii. With the syndromes obtained in step 1 above, form the error-locator polynomial $\sigma(x)$ using any of the algorithms like Berlekamp, Peterson-Gorenstein-Zierler algorithm, form the error-locator polynomial $\sigma(x)$ using the syndromes obtained in Step 1.

iii. Obtain the roots of $\sigma(x)$ and their respective inverses which indicate the error locations.

iv. Complement the bits in the positions indicated by the error locations to obtain the decoded codeword. The syndrome polynomial can be obtained alternately by dividing $r(x)$ by $g(x)$ and evaluating the remainder at $\alpha, \alpha^2, \dots . \alpha^{2t}$. This is same as the syndrome nonbinary BCH codes; nonbinary BCH codes form another class of BCH codes where the coefficients of the code polynomial are also elements from the extended field. Encoding of non-binary BCH codes follows the same procedure as that of binary BCH codes.

3.3 The binary Golay code

The binary form of the Golay code is one of the most important types of linear binary block codes. The t-error correcting code can correct a maximum of t errors. A perfect t-error correcting code has the property that every word lies within a distance of t to exactly one code word. Equivalently, the code has $d_{min} = 2t + 1$, and covering radius t, where the covering radius r is the smallest number such that every word lies within a distance of r to a codeword.

The time complexity for hamming codes is 0(n2) since it is multiplication of two matrices. The time complexity for Golay binary code is as follows 0(n) for the calculating syndrome that is calculating the error.

3.4 Reed-Solomon codes

Reed-Solomon codes are block-based error correcting codes with a wide range of applications in digital communications and storage. Reed-Solomon codes are used to correct errors in many systems such as storage devices, wireless or mobile communications, satellite, DVB and high-speed modems such as ADSL, xDSL. A typical communication channel using Reed-Solomon code is shown in **Figure 6**.

The Reed-Solomon encoder takes a block of digital data and adds extra redundant bits. Errors occur during transmission or storage due to noise, interference, scratch on CD, etc. The Reed-Solomon decoder processes each block and attempts to correct errors and recover the original data. The number and type of errors that can be corrected depends on the characteristics of the Reed-Solomon code.

3.4.1 Properties of Reed-Solomon codes

Reed Solomon codes are a subset of BCH codes and are linear block codes. A Reed-Solomon code is denoted as RS (n,k) with *s*-bit symbols.

This means that the encoder takes *k* data symbols of **s** bits each and adds parity symbols to make an n symbol code word. There are *n-k* parity symbols of s bits each. A Reed-Solomon decoder can correct up to t symbols that contain errors in a code word, where $2t = n-k$.

The following diagram shows a typical Reed-Solomon code word.

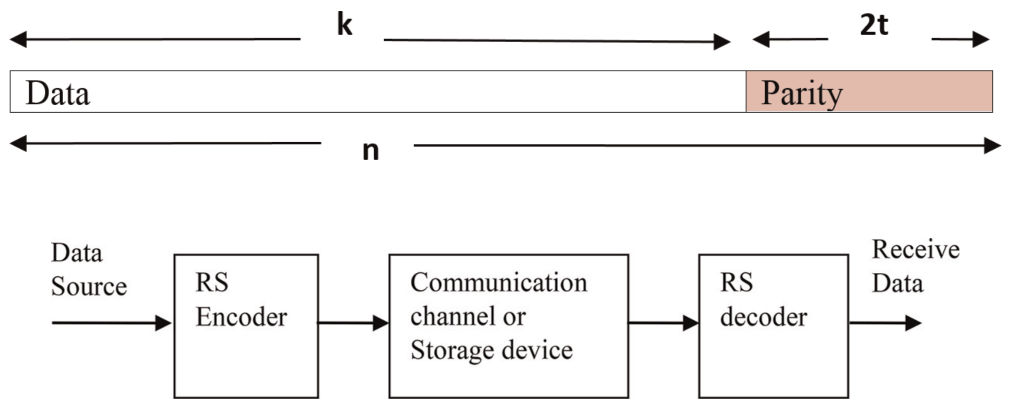

Figure 6.
The Reed-Solomon code with communication channel.

3.4.2 Architectures for encoding and decoding Reed-Solomon codes

Reed-Solomon encoding and decoding can be carried out in software or in special-purpose hardware.

3.4.2.1 Finite (Galois) field arithmetic

Reed-Solomon codes are based on a specialist area of mathematics known as Galois fields or finite fields. A finite field has the property that arithmetic operations (+, −, ×, / etc.) on field elements always have a result in the field. A Reed-Solomon encoder or decoder needs to carry out these arithmetic operations. These operations require special hardware or software functions to implement.

3.4.2.2 Generator polynomial

A Reed-Solomon codeword is generated using a special polynomial. All valid codewords are exactly divisible by the generator polynomial. The generator polynomial is denoted as below:

$$g(x) = \left(x-\alpha^{i}\right)\left(x-\alpha^{i+1}\right)\dots\left(x-\alpha^{i+2t}\right) \tag{20}$$

and the codeword is constructed using:

$$c(x) = g(x).i(x). \tag{21}$$

where g(x) is the generator polynomial, $i(x)$ is the information block, $c(x)$ is a valid codeword and α is referred to as a primitive element of the field.

Example: Generator for RS(255,249).

$$g(x) = \left(x-\alpha^{0}\right)\left(x-\alpha^{1}\right)\left(x-\alpha^{2}\right)\left(x-\alpha^{3}\right)\left(x-\alpha^{4}\right)\left(x-\alpha^{5}\right) \tag{22}$$

$$g(x) = x^{6} + g_5x^{5} + g_4x^{4} + g_3x^{3} + g_2x^{2} + g_1x^{1} + g_0 \tag{23}$$

3.4.3 Encoder architecture

The 2t parity symbols in a systematic Reed-Solomon codeword are given by:

$$p(x) = i(x).x^{n-k} \bmod g(x) \tag{24}$$

Figure 7 shows the architecture for a systematic RS(255,249) encoder:

Each of the 6 registers holds a symbol (8 bits). The arithmetic operators carry out finite field addition or multiplication on a complete symbol.

3.4.4 Decoder architecture

A general architecture for decoding Reed-Solomon codes is shown in **Figure 8**.

Key

r(x)	codeword at receiver
Si	Syndromes
L(x)	Polynomial of error locator

Xi	Locations of error
Yi	Magnitudes of error
c(x)	code word recovered
v	Errors in total

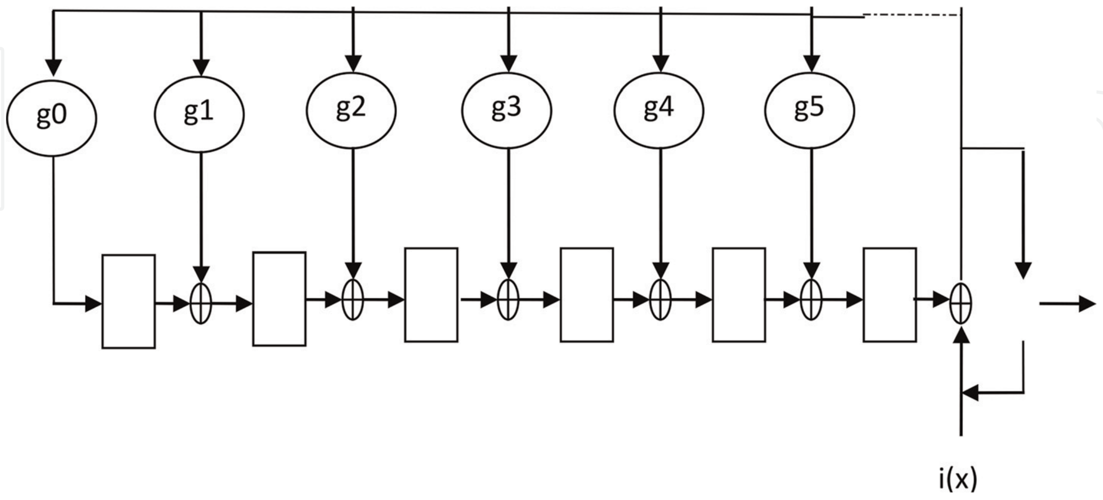

Figure 7.
Block diagram of RS encoder.

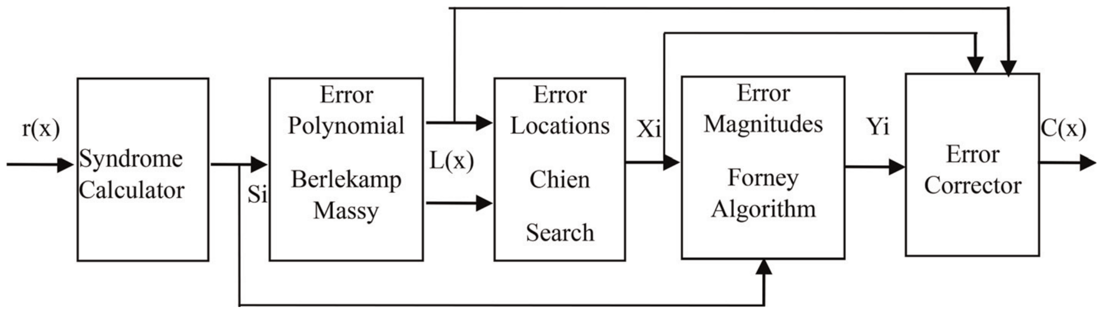
r(x)
Syndrome Calculator
Si
Error Polynomial Berlekamp Massy
L(x)
Error Locations Chien Search
Xi
Error Magnitudes Forney Algorithm
Yi
Error Corrector
C(x)

Figure 8.
Block diagram of RS decoder.

The received codeword $r(x)$ is the original (transmitted) codeword $c(x)$ plus errors:

$r(x) = c(x) + e(x)$.

A Reed-Solomon decoder attempts to identify the position and magnitude of up to t errors (or $2\,t$ erasures) and to correct the errors or erasures.

Syndrome calculation: This is a similar to parity calculation. A Reed-Solomon codeword has $2\,t$ syndromes that depend only on errors (not on the transmitted codeword). The syndromes can be calculated by substituting the 2 t roots of the generator polynomial g(x) into r(x).

Finding the symbol error locations: Error locations are found by solving simultaneous equations with t unknowns. It uses several fast algorithms, which take the advantage of the special matrix structure of these codes and reduce the computational effort. In general two steps are involved.

Find an error locator polynomial: This can be done using the Berlekamp-Massey algorithm or Euclid's algorithm. Euclid's algorithm is more popular because it is easier to implement: however, the Berlekamp-Massey algorithm has efficient hardware and software implementations.

Find the roots of this polynomial: This is done using the Chien search algorithm.
Finding the symbol error values: Again, this involves solving simultaneous equations with t unknowns. A widely-used fast algorithm is the Forney algorithm.

3.5 Low-density parity check codes

Low-density parity check (LDPC) codes are a class of linear block code. The term "Low Density" refers to the parity check matrix which contains only few '1's in comparison to '0's. LDPC codes are arguably the best error correction codes in existence at present. LDPC codes were first introduced by R. Gallager in his Ph.D. thesis in 1960. However, they were forgotten due to introduction of Reed-Solomon codes and since there were problems with implementation of LDPC codes due to limited technological know-how. The LDPC codes were rediscovered in mid-90s by R. Neal and D. Mackay at the Cambridge University.

N bit long LDPC code is defined code in terms of M number of parity check equations, and these equations can be described as an M × N parity check matrix H.

where, M is the number of parity check equations and N is the number of bits in the code word.

Consider the 6-bit long codeword in the form $c = [c_1, c_2, c_3, c_4, c_5, c_6]$ which satisfies 3 parity check equations as shown below.

$$c_1 \oplus c_2 \oplus c_5 = 0 \tag{25}$$

$$c_1 \oplus c_4 \oplus c_6 = 0 \tag{26}$$

$$c_1 \oplus c_2 \oplus c_3 \oplus c_6 = 0 \tag{27}$$

We can now define 3 × 6 parity check matrix as,

$$c_1 \oplus c_2 \oplus c_5 \tag{28}$$

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{29}$$

w_r and w_c changes, therefore this is an irregular parity check matrix.

The density of '1's in LDPC code parity check matrix is very low, row weight (w_c) is the number of '1's in a row, number of symbols taking part in a parity check, column weight is the number of '1's in a column, number of times a symbol takes part in parity checks.

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{30}$$

The $H_{M \times N}$ parity check matrix defines a rate $R = K/N$, $R = K/N$ code where $K = N - M$ Codeword is said to be valid if it satisfies the syndrome calculation $z = c.H^T = 0$.

We can generate the codeword *c* by multiplying message *m* with generator matrix *G*.

$$c = m.G \tag{31}$$

We can obtain the generator matrix *G* from parity check matrix *H* by

1. arranging the parity check matrix in systematic form using row and column operations and

$$H_{sys} = \left[I_M \middle| P_{M\times K}\right] H_{sys} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{32}$$

2. rearranging the systematic parity check matrix.

$$G = \left[P^T_{K\times M} \middle| I_K\right] \tag{33}$$

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

3. we can verify our results as

$$G.H^T = 0 \tag{35}$$

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{36}$$

Tanner graph is a graphical representation of parity check matrix specifying parity check equations. Tanner graph for LDPC codes, as shown in **Figure 9**, consists of N number of variable nodes and M number of check nodes. In Tanner graph, m^{th} check node is connected to n^{th} variable node if and only if n^{th} element in m^{th} row in parity check matrix H, h_{mn} is a '1'.

The marked path $z_2 \rightarrow c_1 \rightarrow z_3 \rightarrow c_6 \rightarrow z_2$ is an example for short cycle of 4. The number of steps needed to return to the original position is known as the girth of the code.

3.6 Convolution codes

Convolutional codes differ from block codes in that the encoder contains memory. The n encoder outputs at any time unit depend not only on the k inputs but also on m previous input blocks. An (n, k, m) convolutional code can be implemented with a k-input, n-output linear sequential circuit with input memory m. Typically, n and k are small integers. Wozencraft proposed sequential decoding as an efficient decoding scheme for convolution codes, and many experimental studies were

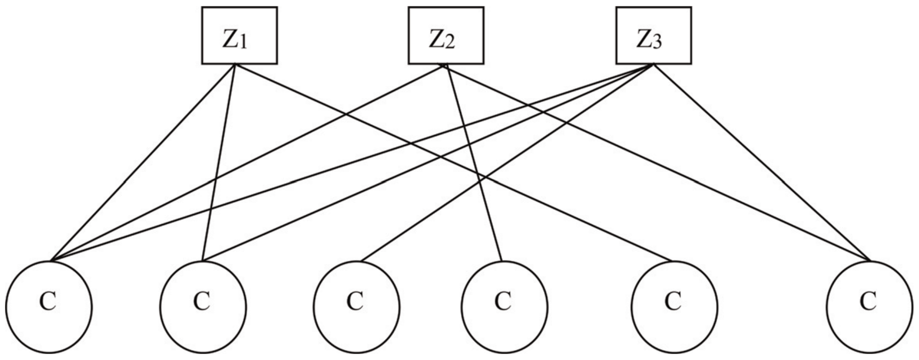

Figure 9.
LDPC codes tanner graph representation.

performed on the same. In 1963, Massey proposed a method which was simpler to implement called threshold decoding. Then in 1967, Viterbi proposed a maximum likelihood decoding scheme that was relatively easy to implement for codes with small memory orders. Viterbi decoding was combined with improved versions of sequential decoding and convolutional codes were used in deep-space and satellite communication in early 1970s. A convolutional code is generated by passing the information sequence to be transmitted through a linear finite-state shift register. In general, the shift register consists of K (k-bit) stages and n linear algebraic function generators.

Convolution codes have simple encoding and decoding methods and are quite a simple generalization of linear codes and have encodings as cyclic codes.

An (n,k) convolution code (CC) is defined by a k x n generator matrix, entries of which are polynomials over F_2.

$$G_1 = [x^2+1, x^2+x+1] \tag{37}$$

is the generator matrix for a (2,1) convolution code CC_1 and

$$G_2 = \begin{pmatrix} 1+x & 0 & x+1 \\ 0 & 1 & x \end{pmatrix} \tag{38}$$

is the generator matrix for a (3,2) convolution code CC_2.

3.6.1 Encoding of finite polynomials

An (n,k) convolution code with a k x n generator matrix G can be used to encode a k-tuple of plain-polynomials.

$$I = (I_0(x), I_1(X), \dots, I_{k-1}(x)). \tag{39}$$

to get an n-tuple of crypto-polynomials.

$$C = (C_0(x), C_1(x), \dots, C_{n-1}(x)). \tag{40}$$

As follows

$$C = I.G. \tag{41}$$

3.6.2 Turbo codes

Turbo codes were proposed by Berrou and Glavieux in the 1993 International Conference in Communications. Turbo codes demonstrated a performance within 0.5 dB of the channel capacity limit for BPSK. Turbo codes use parallel concatenated coding, recursive convolutional encoders, and Pseudo-random interleaving.

Turbo codes have a remarkable power efficiency in Additive White Gaussian Noise (AWGN) and flat-fading channels for moderately low BER, mostly used in delivery of multimedia services. However turbo codes have a long latency and poor performance at very low BER since turbo codes operate at very low SNR, channel estimation and tracking is a critical issue. The principle of iterative or "turbo" processing can be applied to other problems; Turbo-multiuser detection can improve performance of coded multiple-access systems. Performance close to the Shannon Limit can be achieved (E_b/N_0 = −1.6 dB if $R_b \rightarrow 0$) at modest complexity. Turbo codes have been proposed for low-power applications such as deep-space and satellite communications, as well as for limited interference applications such as third generation cellular, personal communication services, ad hoc, and sensor networks.

The information capacity (or channel capacity) C of a continuous channel with bandwidth B Hertz can be perturbed by additive Gaussian white noise of power spectral density $N_0/2$, provided bandwidth B satisfies.

$$C = B\log_2\left(1 + \frac{P}{N_0 B}\right) \quad bits/\sec ond \tag{42}$$

where P is the average transmitted power $P = E_b R_b$ (for an ideal system, $R_b = C$), E_b is the transmitted energy per bit, R_b is transmission rate.

3.6.2.1 Turbo code encoder

The fundamental of turbo encoder is using two identical recursive systematic convolutional (RSC) code arranged in parallel form separated by an interleaver. The nature of the interleaver in turbo code is pseudo-random in order to minimize the correlation between the outputs of encoders that make the best results, and its matrix forms with rows and columns, depending on the block size of the code [8]. The structure of turbo encoder is shown in **Figure 10**.

Interleaver/deinterleaver are used and play an important role in the performance of turbo codes. The interleaver helps to increase the minimum distance and break the low weight of the input sequence by spreading out the burst errors. This is done by mapping the sequence of bits to another sequence of bits. When the length of the interleaver is very large, Turbo codes achieve excellent performance [9]. According to the structure of turbo encoder, puncturing technique will be used to obtain high rate. Puncturing is operating on the parity bits only, but the systematic bits are not punctured [10].

3.6.2.2 Turbo decoder

Turbo decoders consist of a pair of convolutional decoders which cooperatively and iteratively exchange soft-decision information. The information can be passed from one decoder to the other, where each decoder takes the information corresponding to the systematic, parity bits from the encoder and a priori information from the other decoder and the resulting output generated by the decoder should be soft decisions or estimates. The passing of information between the first and second decoder continues until a given number of iterations is reached. With

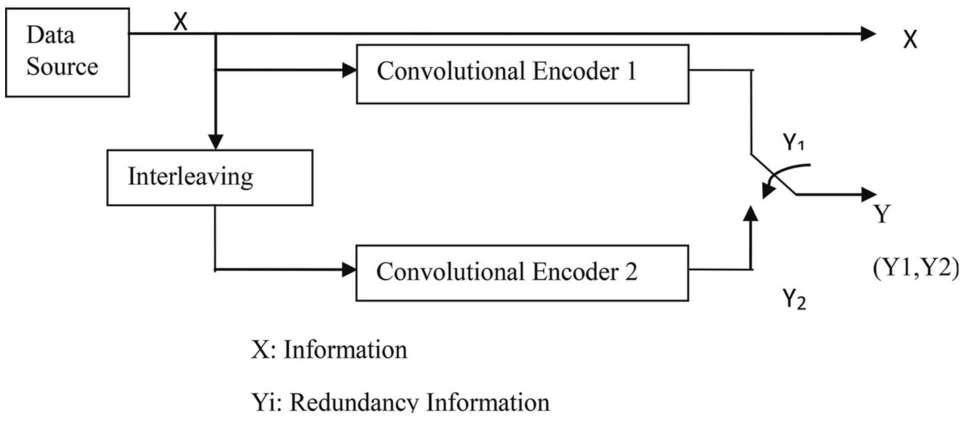

Figure 10.
Turbo code encoder.

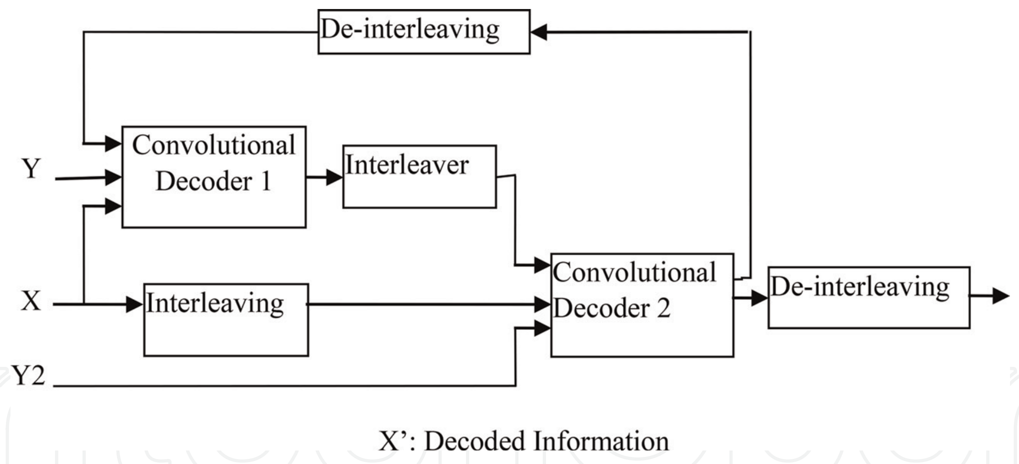

Figure 11.
Turbo code decoder.

each iteration, the estimates of the information bits improve. A correct estimate of the message is achieved by increasing the number of iterations. However, this improvement does not increase linearly. Practically, it is enough to utilize a small number of iterations to achieve acceptable performance [11, 12]. **Figure 11** illustrates the structure of turbo decoder.

The decoder produces a soft-decision to each message bits in logarithmic form known as a log likelihood ratio (LLR) [11, 12]. At the end of this process, a hard decision is carried out at the second decoder to convert the final signal to 1's and 0's and compare it with the original message" [13, 14].

3.6.3 Trellis coded modulation (TCM)

Error probability can be decreased by adding more code bits - the code rate is increased. Combine both encoding and modulation (using Euclidean distance only). Allow parallel transition in the trellis, and it has significant coding gain (3∼4 dB) without bandwidth compromise. It has the same complexity (same amount of computation, same decoding time and same amount of memory needed). Trellis

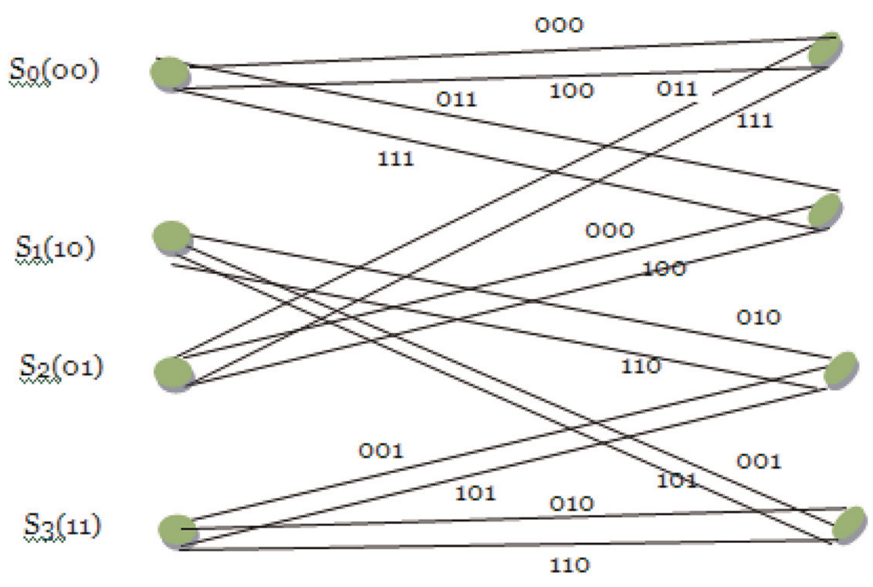

Figure 12.
Encoder for four state Trellis TCM.

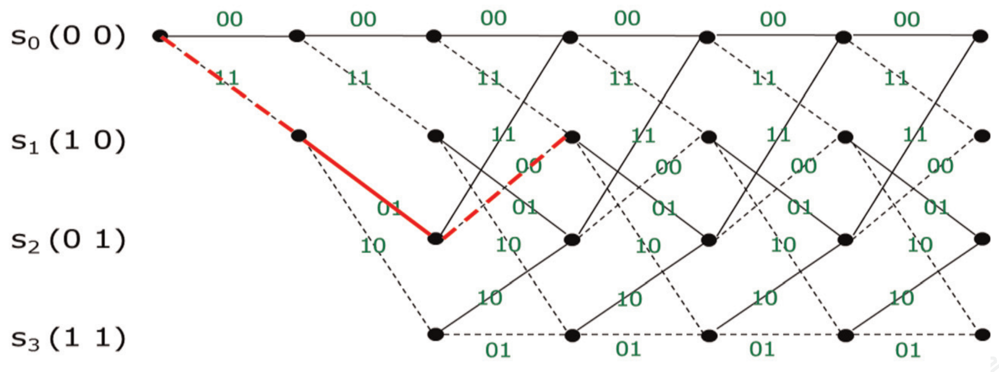

Figure 13.
Trellis representation QPSK.

code has great potential for fading channel and widely used in Modem. **Figure 12** shows encoder for four state Trellis TCM.

There is increase in constellation size compared to uncoded communication, increase in throughput (b/s/Hz), and decline in BER performance due to decrease of $d_{min.}$ Trellis coded modulation (TCM) is used to offset loss resulting from constellation size increase. TCM achieves this higher gain by jointly using the distance properties of the code and the distance properties of the constellation, by carefully mapping coded and uncoded bits to the constellation points. TCM uses “set partitioning” to map the bits to the constellation points. **Figure 13** shows Trellis representation for QPSK.

Input: 101 → Output: 001011.

3.7 Application areas for error correcting codes (ECCs)

Deep Space communication. used a concatenation of Reed-Solomon code and convolutional code.

Storage media. BCH codes and Reed-Solomon codes are used in applications like compact disk players, DVDs, disk drives, NAND flash drives, and 2D bar codes. LDPC codes are used for SSDs and fountain codes are erasure codes used in data-storage applications.

Mobile communication. ARQ is sometimes used with Global System for Mobile (GSM) communication to guarantee data integrity. Traffic channels in 2G standard use convolution code. Convolution and turbo codes are used in 3G (UMTS) networks; convolution coding can be used for low data rates and turbo coding for higher rates.

WiMAX (IEEE 802.16e standard for microwave communications) and high-speed wireless LAN (IEEE 802.11n) use LDPC as a coding scheme.

Satellite communication. For reliable communication in WiMax, optical communication, and power line communication, or in multi-layer flash memories, turbo and LDPC codes are desirable.

Hybrid ARQ is another technique for spectrum efficiency and reliable link. Network coding is one of the most important breakthroughs in information theory in recent years.

4. Conclusion

The chapter describes the different types of errors encountered in a data communication system over channels and focuses on the role of polynomials in implementing various algorithms for error detection and correction codes. It discusses error detection codes such as Simple Parity check, Two-dimensional Parity check, Checksum, Cyclic redundancy check; and error corrections codes such as Hamming code, BCH, Golay codes, RS Code, LDPC, Trellis and Turbo codes. It also gives an overview of the architecture and implementation of the codes and discusses the applications of these codes in various systems.

Author details

Charanarur Panem[1], Vinaya Gad[2] and Rajendra S. Gad[1]*

1 Altera SoC Laboratory, Department of Electronics, Goa University, Goa, India

2 Department of Computer Science, G.V.M.'s College, Ponda, Goa, India

*Address all correspondence to: rsgad@unigoa.ac.in

References

[1] Available at: https://nptel.ac.in/courses/106105080/pdf/M3L2.pdf

[2] Available at: https://www.techwalla.com/articles/types-of-errors-in-data-communication

[3] Bertsekas D, Gallager R. Data Networks. 2nd ed. Prentice Hall; 1992. Available at: web.mit.edu/dimitrib/www/datanets.html

[4] Forouzan B. Data Communications and Networking. 5th ed. McGraw Hill; 2013

[5] Lin S, Costello DJ. Error Control Coding. 2nd ed. Prentice Hall; 2004

[6] McEliece R. Finite Fields for Computer Scientists and Engineers. Springer; 1986

[7] Available at: https://electronicsforu.com/technology-trends/error-correcting-codes-comm-storage

[8] Benkeser C, Burg A, Cupaiuolo T, Huang Q. Design and optimization of an HSDPA Turbo Decoder ASIC. Journal of Solid-State Circuits. 2009

[9] Sadjadpour HR, Sloane NJA, Salehi M, Nebe G. Interleaver design for turbo codes. IEEE Journal on Selected Area In Communications. 2001;**19**(5):831-837

[10] Raad IS, Yakan M. Implementation of a turbo codes test bed in the Simulink environment. In: International Symposium on Signal Processing and Its Applications. Piscataway: IEEE; 2005. pp. 847-850

[11] Kaza J, Chakrabarti C. Design and implementation of low energy turbo decoders. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2004;**12**(9):968-977

[12] Moreira JC, Farrell PG. Essential of Error Control Coding. Wiley; 2006

[13] Moon TK. Error Correction Coding: Mathematical Methods and Algorithms. Wiley; 2005

[14] Yi B-N. Turbo code design and implementation of high-speed parallel decoder. Telkomnika. 2013;**11**(4): 2116-2123