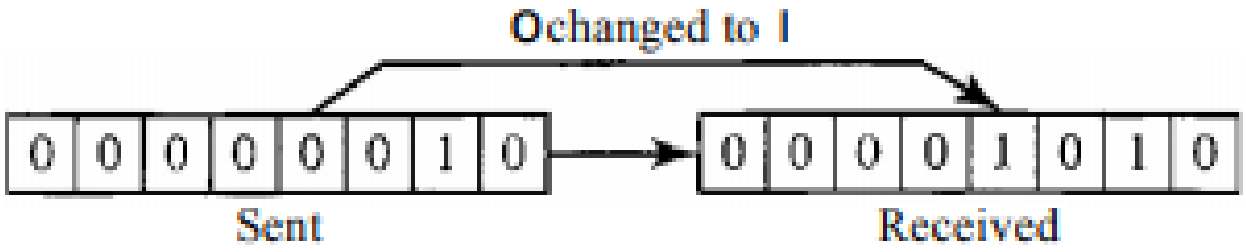
Types of Errors

- Any time data are transmitted from one node to the next, they can become corrupted in passage.

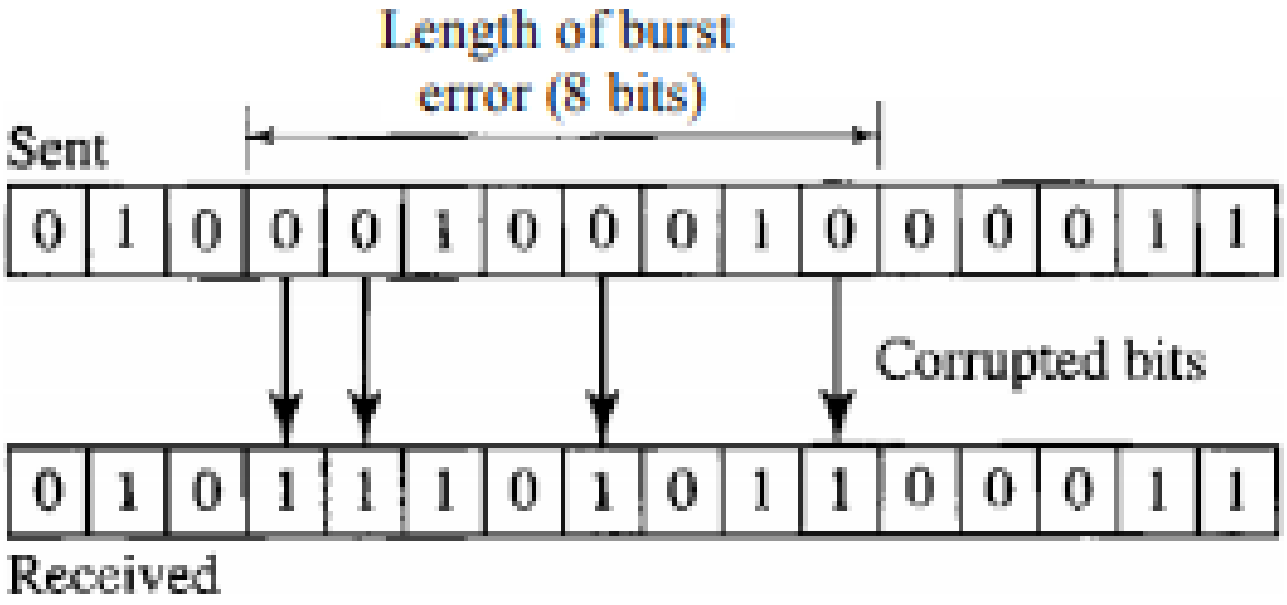
Single-Bit Error

- The term single-bit error means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.



Burst Error

- The term burst error means that 2 or more bits in the data unit have changed from 1to 0 or from 0 to 1. Note that a burst error does not necessarily mean that the errors occur in consecutive bits. The length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.



- A burst error is more likely to occur than a single-bit error. The duration of noise is normally longer than the duration of 1 bit, which means that when noise affects data, it

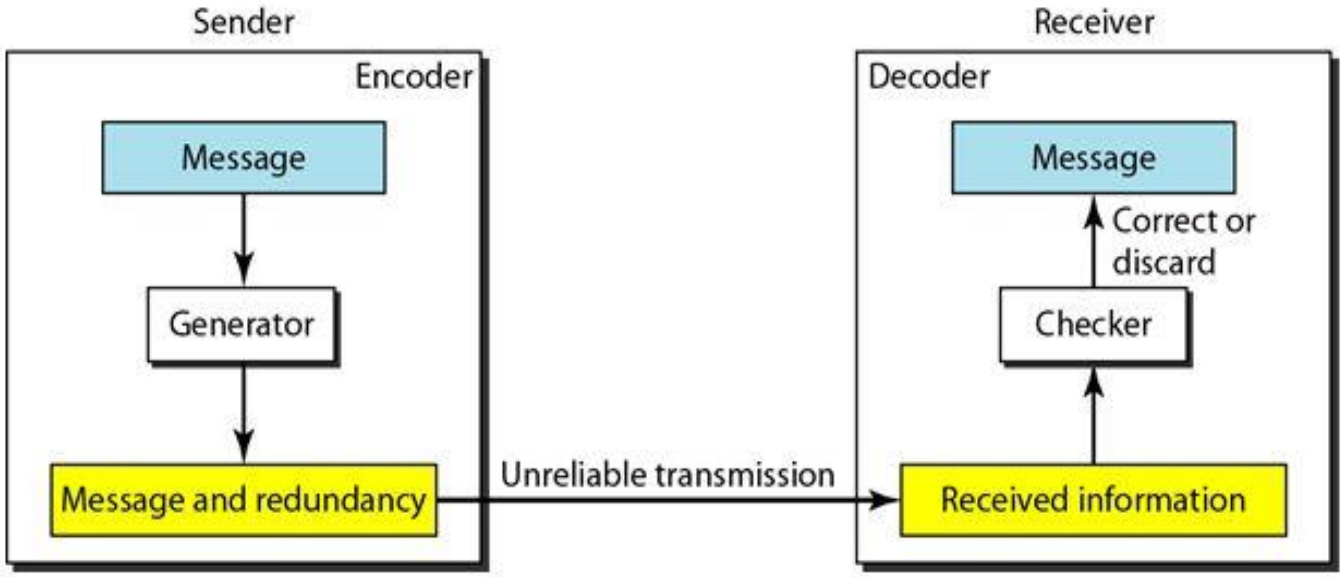
affects a set of bits. The number of bits affected depends on the data rate and duration of noise

Redundancy

- The central concept in detecting or correcting errors is redundancy. To be able to detect or correct errors, we need to send some extra bits with our data.
- These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.
- The correction of errors is more difficult than the detection. In error detection, we are looking only to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of errors. A single-bit error is the same for us as a burst error.
- In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message. The number of the errors and the size of the message are important factors.

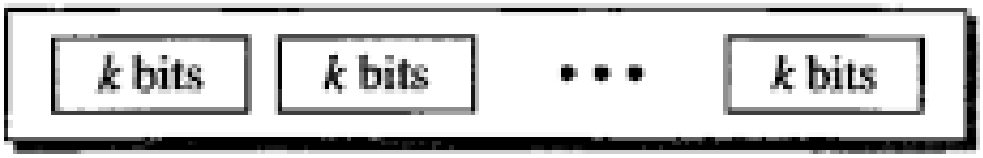
Forward Error Correction Versus Retransmission

- There are two main methods of error correction. Forward error correction is the process in which the receiver tries to guess the message by using redundant bits. This is possible, as we see later, if the number of errors is small.
- Correction by retransmission is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free (usually, not all errors can be detected).
- Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits. The receiver checks the relationships between the two sets of bits to detect or correct the errors.
- We can divide coding schemes into two broad categories: block coding and convolution coding.

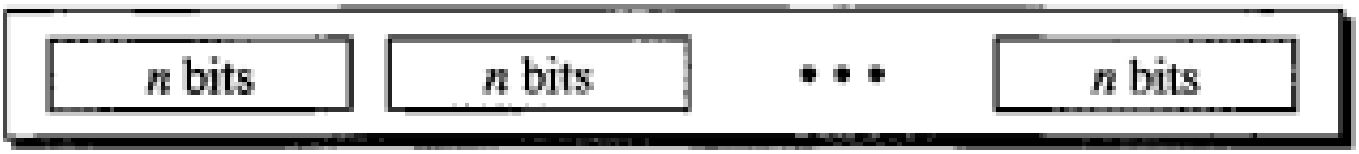


BLOCK CODING

- In block coding, we divide our message into blocks, each of k bits, called datawords. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called codewords.
- it is important to know that we have a set of datawords, each of size k, and a set of code words, each of size of n. With k bits, we can create a combination of 2^k datawords; with n bits, we can create a combination of 2^n codewords.
- Since n > k, the number of possible codewords is larger than the number of possible datawords. The block coding process is one-to-one; the same dataword is always encoded as the same codeword. This means that we have $2^n - 2^k$ codewords that are not used.



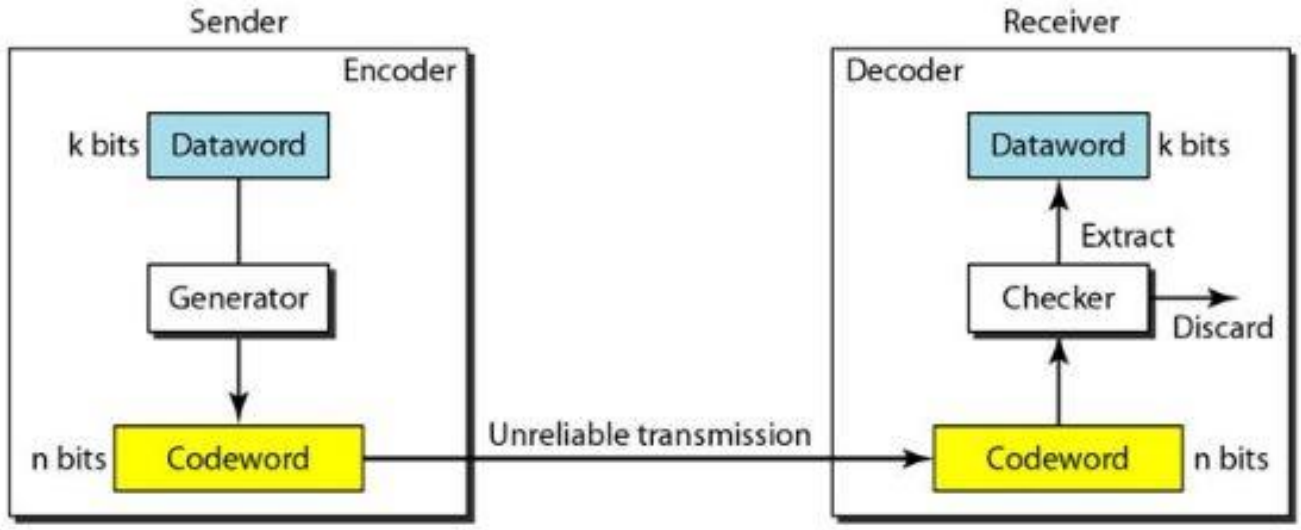
2^k Datawords, each of k bits



2^n Codewords, each of n bits (only 2^k of them are valid)

Error Detection

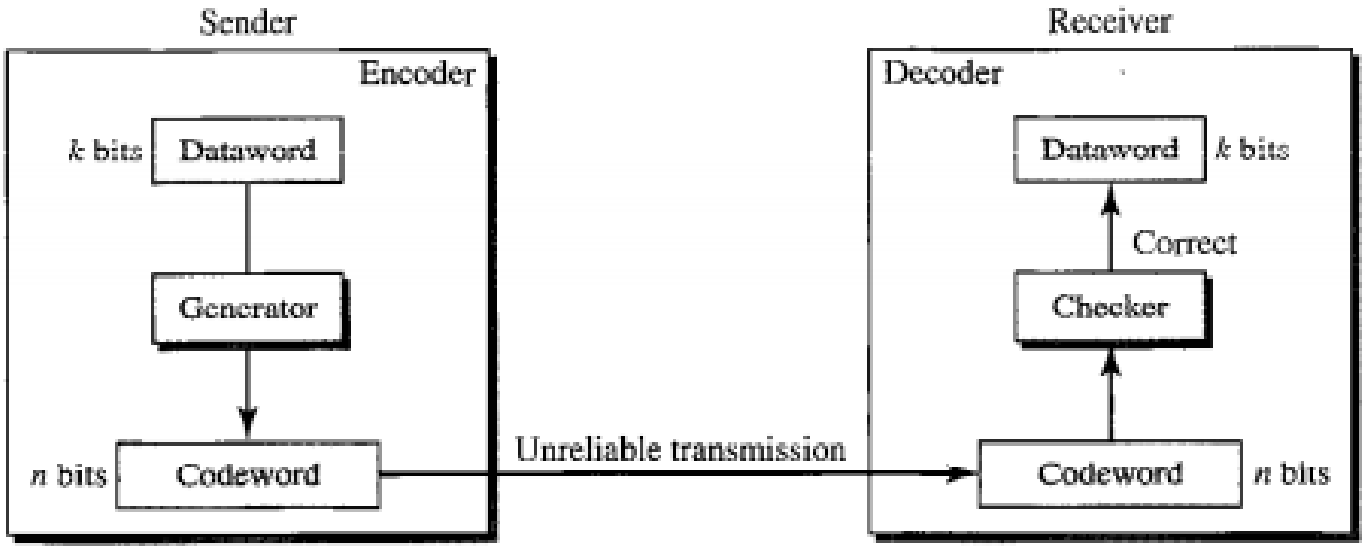
- How can errors be detected by using block coding? If the following two conditions are met, the receiver can detect a change in the original codeword.
- The receiver has (or can find) a list of valid codewords.
- The original codeword has changed to an invalid one.



- The sender creates codewords out of datawords by using a generator that applies the rules and procedures of encoding (discussed later).
- Each codeword sent to the receiver may change during transmission. If the received codeword is the same as one of the valid codewords, the word is accepted; the corresponding dataword is extracted for use.
- If the received codeword is not valid, it is discarded. However, if the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected.

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Error Correction



- Error correction is much more difficult than error detection.
- The checker functions are much more complex.

Hamming Distance

- The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits.
- We show the Hamming distance between two words x and y as ***d(x, y).***
- The Hamming distance can easily be found if we apply the XOR operation on the two words and count the number of 1s in the result.

Example:

1. The Hamming distance d(000, 011) is 2 because 000 $_{XOR}$ 011 is 011 (two 1's).
2. The Hamming distance d(10101, 11110) is 3 because 10101 $_{XOR}$ 11110 is 01011 (three 1s).

Minimum Hamming Distance

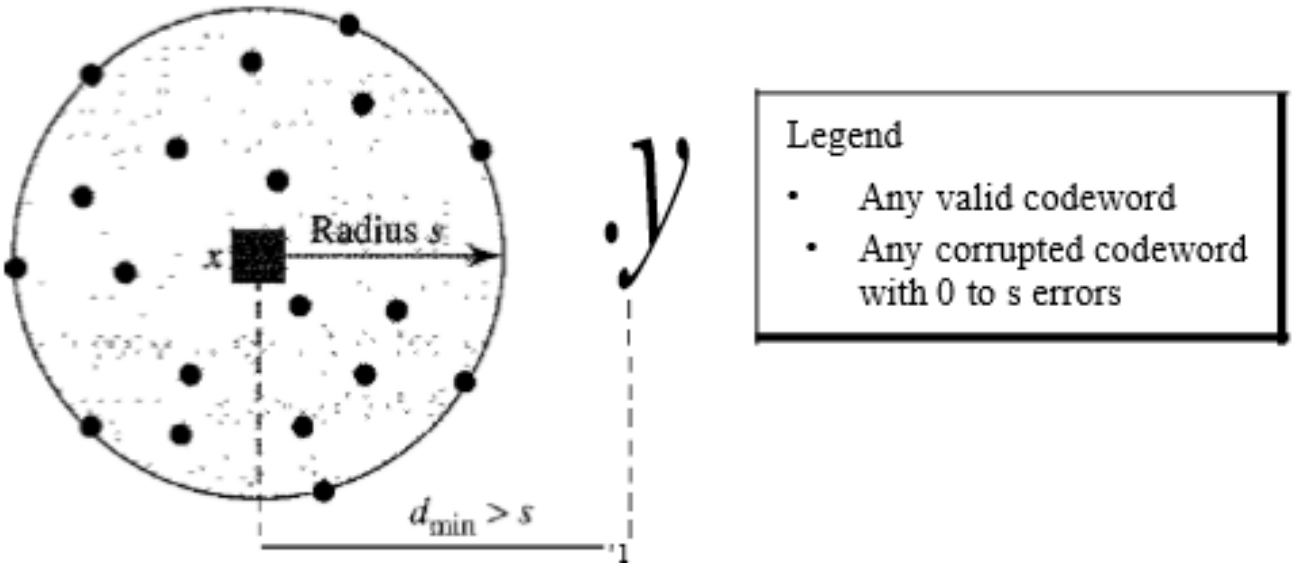
- the minimum Hamming distance is the smallest Hamming distance between all possible pairs.
- d(00000, 01011) = 3 d(01011, 10101) =4 d(00000, 10101) =3
- d(01011, 11110) = 3 d(00000, 11110) = 4 d(10101, 11110) =3
- The d_{min} in this case is 3.

Three Parameters

- Before we continue with our discussion, we need to mention that any coding scheme needs to have at least three parameters: the codeword size n, the dataword size k, and the minimum Hamming distance d_{min}.
- A coding scheme C is written as C(n, k) with a separate expression for d_{min}
- For example, we can call our first coding scheme C(3, 2) with dmin =2 and our second coding scheme C(5, 2) with dmin ::= 3.

Minimum Distance for Error Detection

- Now let us find the minimum Hamming distance in a code if we want to be able to detect up to s errors.
- Ifs errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s. If our code is to detect up to s errors, the minimum distance between the valid codes must be s + 1,
- so that the received codeword does not match a valid codeword. In other words, if the minimum distance between all valid codewords is s + 1, the received codeword cannot be erroneously mistaken for another codeword. The distances are not enough (s + 1) for the receiver to accept it as valid. The error will be detected.



Error Correction

- In error detection, the receiver needs to know only that the received codeword is invalid; in error correction the receiver needs to find (or guess) the original codeword sent.
- We can say that we need more redundant bits for error correction than for error detection.

Q if the receiver can correct an error without knowing what was actually sent. We add 3 redundant bits to the 2-bit dataword to make 5-bit codewords.

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

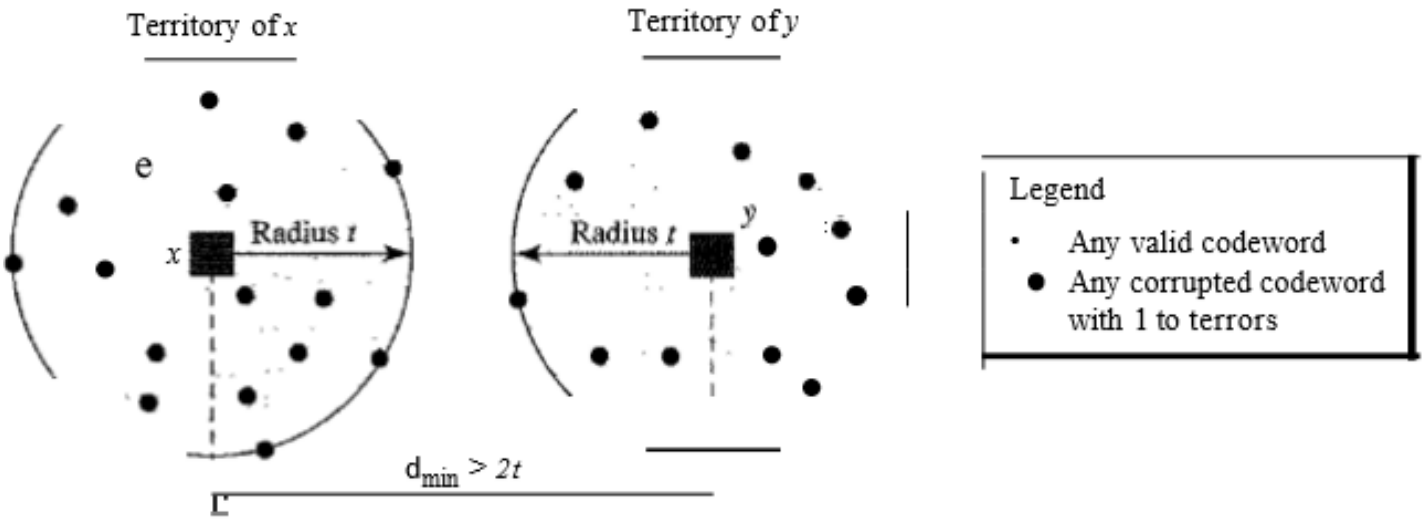
$d(00000, 01011) = 3$ $d(00000, 10101) = 3$ $d(00000, 11110) = 4$
$d(01011, 10101) = 4$ $d(01011, 11110) = 3$ $d(10101, 11110) = 3$

The d_{min} in this case is 3.

Guarantee a double bit error detection

Minimum Distance for Error Correction

- When a received codeword is not a valid codeword, the receiver needs to decide which valid codeword was actually sent. The decision is based on the concept of territory, an exclusive area surrounding the codeword.
- Each valid codeword has its own territory. We use a geometric approach to define each territory. We assume that each valid codeword has a circular territory with a radius of t and that the valid codeword is at the enter.
- For example, suppose a codeword x is corrupted by t bits or less. Then this corrupted codeword is located either inside or on the perimeter of this circle. If the receiver receives a codeword that belongs to this territory, it decides that the original codeword is the one at the centre. Note that we assume that only up to t errors have occurred; otherwise, the decision is wrong.



- Error correction is more complex than error detection; a decision is involved.
- To guarantee correction of up to t errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.

Example: A code scheme has a Hamming distance $d_{min} = 4$. What is the error detection and correction capability of this scheme?
This code guarantees the detection of up to three errors (s = 3), but it can correct up to one error.

Q Consider a binary code that consists only four valid codewords as given below. a
00000, 01011, 10101, 11110
Lets minimum Hamming distance of code be p and maximum number of erroneous bits that can be corrected by the code be q. The value of p and q are: **(Gate-2017) (2 Marks)**
(A) p = 3 and q = 1 **(B)** p = 3 and q = 2
(C) p = 4 and q = 1 **(D)** p = 4 and q = 2
Answer: (A)

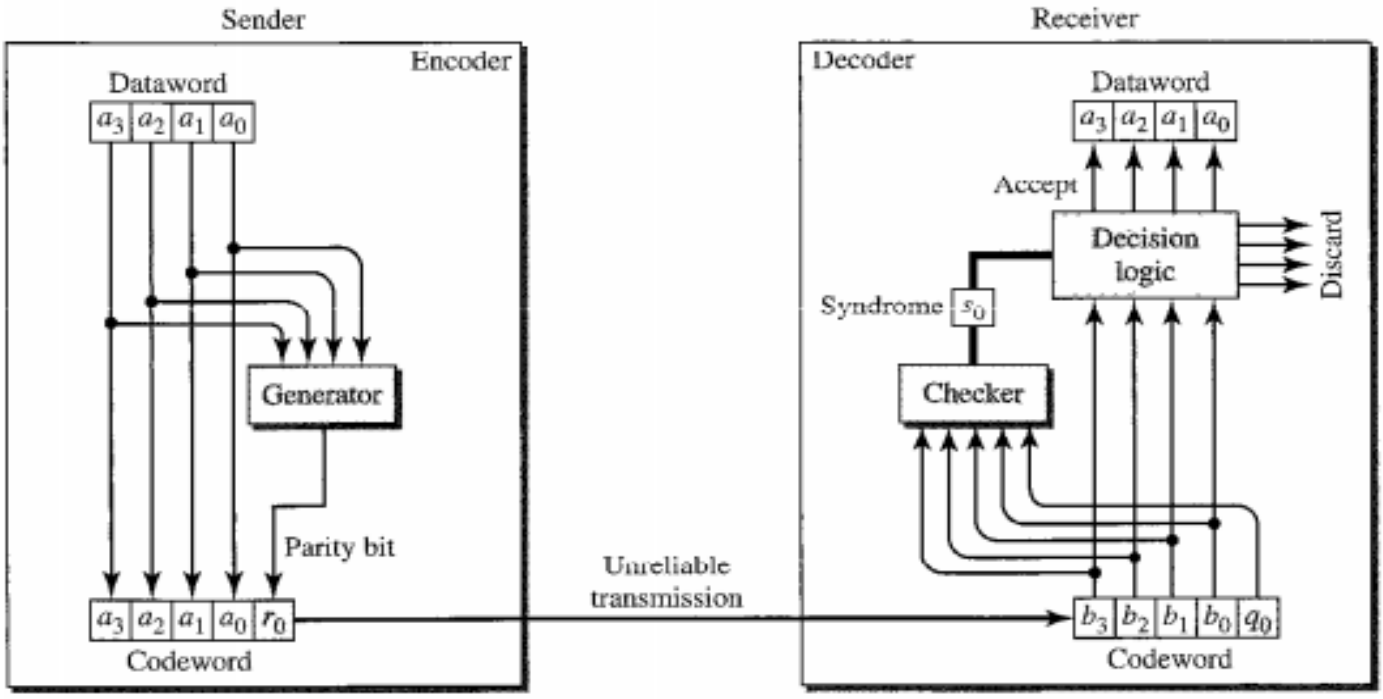
Q An error correcting code has the following code words:
00000000, 00001111, 01010101, 10101010, 11110000.
What is the maximum number of bit errors that can be corrected? **(Gate-2007) (2 Marks)**
(A) 0 **(B)** 1 **(C)** 2 **(D)** 3
Answer: (D)

LINEAR BLOCK CODES

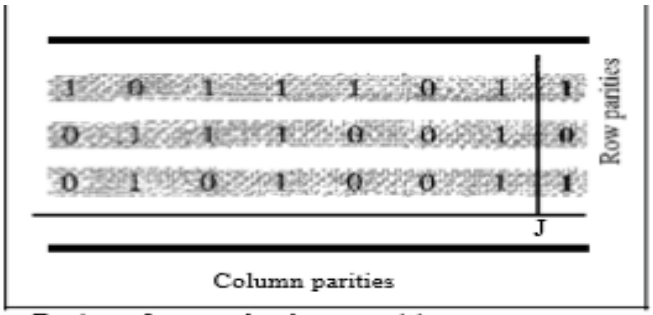
- Almost all block codes used today belong to a subset called linear block codes. The use of nonlinear block codes for error detection and correction is not as widespread because their structure makes theoretical analysis and implementation difficult. We therefore concentrate on linear block codes.

Simple Parity-Check Code

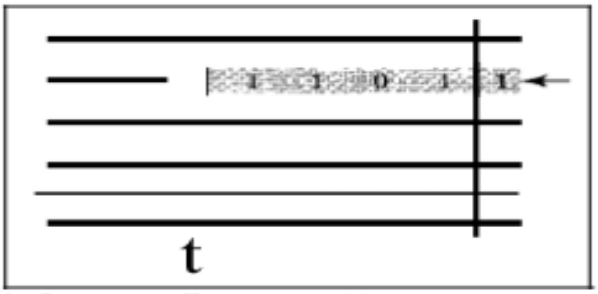
- Perhaps the most familiar error-detecting code is the simple parity-check code. In this code, a k-bit dataword is changed to an n-bit codeword where n = k + 1.
- The extra bit, called the parity bit, is selected to make the total number of Is in the codeword even. Although some implementations specify an odd number of Is.
- A simple parity-check code is a single-bit error-detecting code in which n =k + 1 with dmin =2.



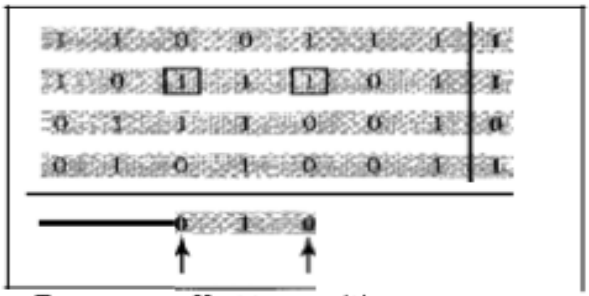
- A simple parity-check code can detect an odd number of errors.
- A better approach is the two-dimensional parity check. In this method, the dataword is organized in a table (rows and columns). the data to be sent, five 7-bit bytes, are put in separate rows. For each row and each column, 1 parity-check bit is calculated.
- The whole table is then sent to the receiver, which finds the syndrome for each row and each column., the two-dimensional parity check can detect up to three errors that occur anywhere in the table (arrows point to the locations of the created nonzero syndromes). However, errors affecting 4 bits may not be detected.



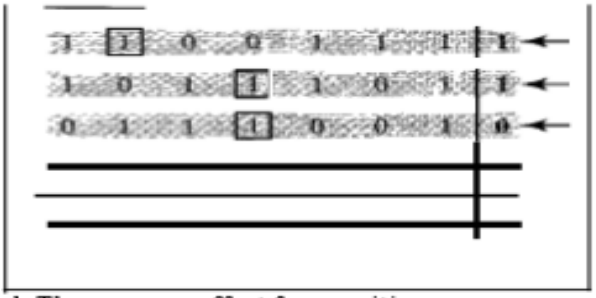
a. Design of row and column parities



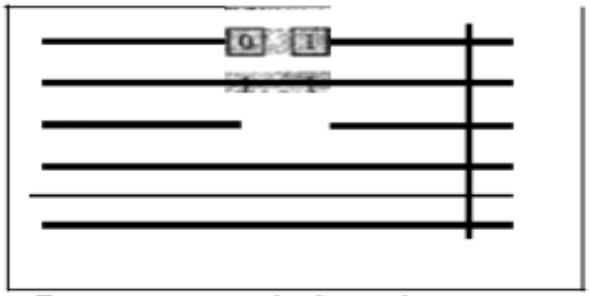
b. One error affects two parities



c. Two errors affect two parities



d. Three errors affect four parities



e. Four errors cannot be detected

Q Data transmitted on a link uses the following 2D parity scheme for error detection: Each sequence of 28 bits is arranged in a 4×7 matrix (rows r_0 through r_3, and columns d_7 through d_1) and is padded with a column d_0 and row r_4 of parity bits computed using the Even parity scheme. Each bit of column d_0 (respectively, row r_4) gives the parity of the corresponding row (respectively, column). These 40 bits are transmitted over the data link. The table shows data received by a receiver and has n corrupted bits. What is the minimum possible value of n? **(Gate-2008) (2 Marks)**

(A) 1 **(B)** 2 **(C)** 3 **(D)** 4

	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
r_0	0	1	0	1	0	0	1	1
r_1	1	1	0	0	1	1	1	0
r_2	0	0	0	1	0	1	0	0
r_3	0	1	1	0	1	0	1	0
r_4	1	1	0	0	0	1	1	0

Answer: (C)

Hamming Codes

- Now let us discuss a category of error-correcting codes called Hamming codes. These codes were originally designed with dmin = 3, which means that they can detect up to two errors or correct one single error.
- Although there are some Hamming codes that can correct more than one error, our discussion focuses on the single-bit error-correcting code.
- First let us find the relationship between n and k in a Hamming code. We need to choose an integer m >= 3. The values of n and k are then calculated from m as $n = 2^m - 1$ and k = n - m. The number of check bits r = m.
- All Hamming codes discussed in this book have $d_{min} = 3$. The relationship between m and n in these codes is n = 2m - 1.
- For example, if m = 3, then n = 7 and k = 4. This is a Hamming code C(7, 4) with dmin =3. shows the datawords and codewords for this code.
- For each integer $r \geq 2$ there is a code with block length $n = 2^r - 1$ and message length $k = 2^r - r - 1$.

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		**p1**	**p2**	**d1**	**p4**	**d2**	**d3**	**d4**	**p8**	**d5**	**d6**	**d7**	**d8**	**d9**	**d10**	**d11**	**p16**	**d12**	**d13**	**d14**	**d15**	
Parity bit coverage	**p1**	X		X		X		X		X		X		X		X		X		X		
	p2		X	X			X	X			X	X			X	X			X	X		...
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

Q Consider a parity check code with three data bits and four parity check bits. Three of the code words are 0101011, 1001101 and 1110001. Which of the following are also code words? **(Gate-2004) (2 Marks)**

I. 0010111 **II.** 0110110 **III.** 1011010 **IV.** 0111010

(A) I and III **(B)** I, II and III **(C)** II and IV **(D)** I, II, III and IV

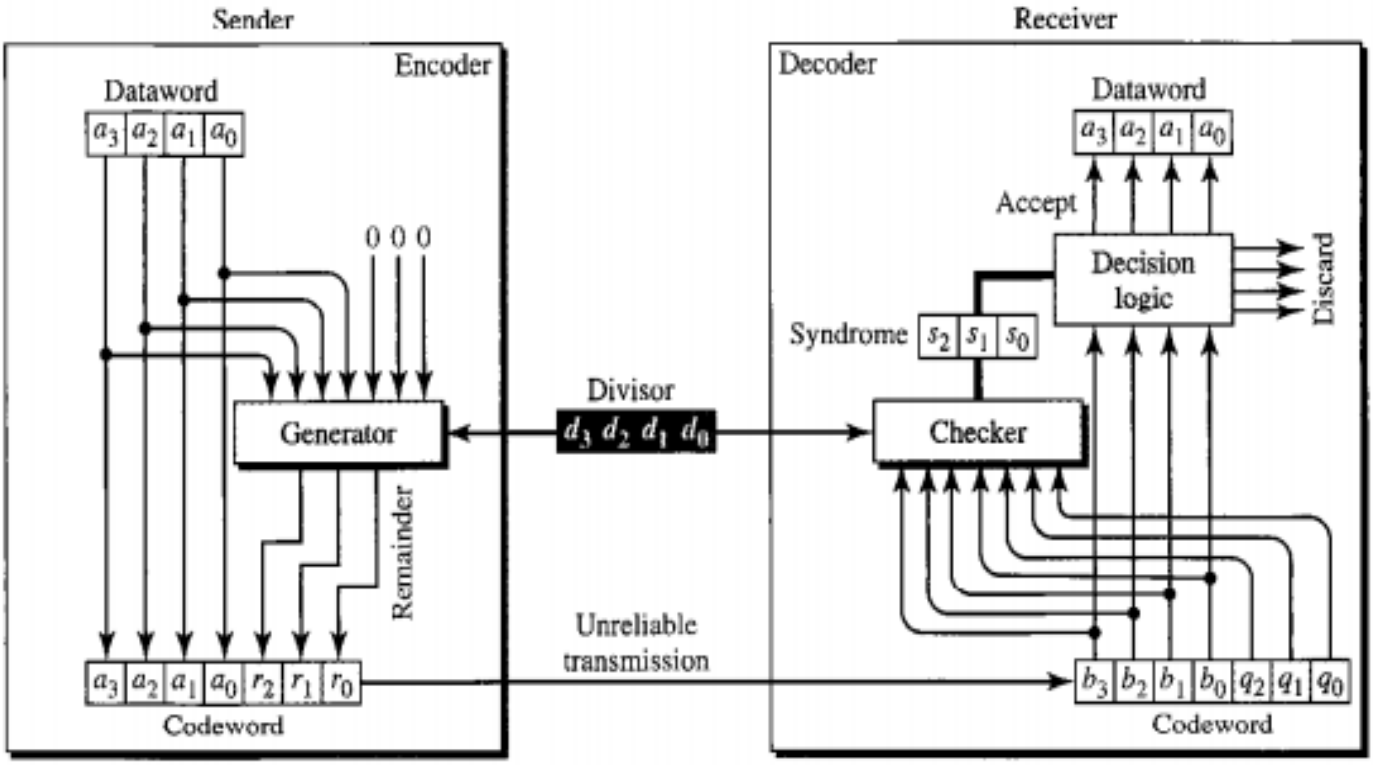
Answer: (A)

CYCLIC CODES

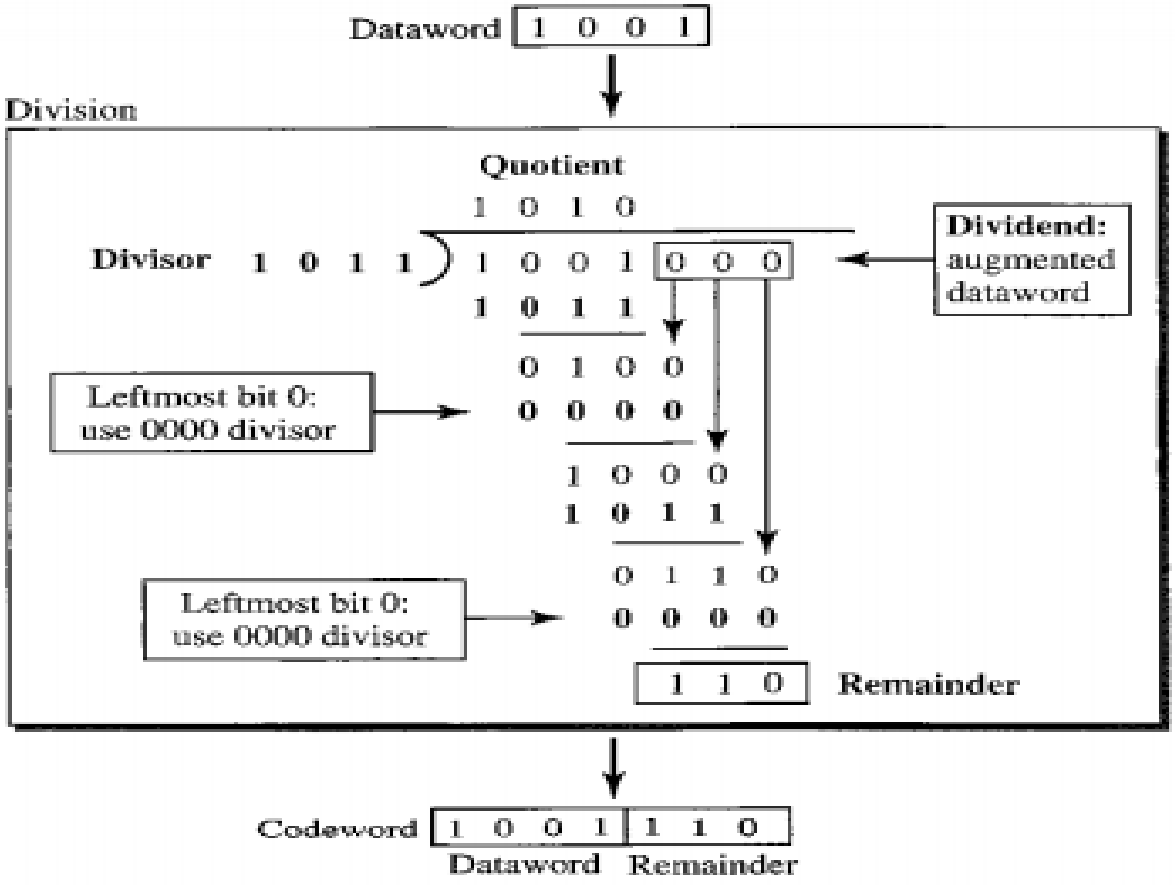
- Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.
- Example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword.

Cyclic Redundancy Check

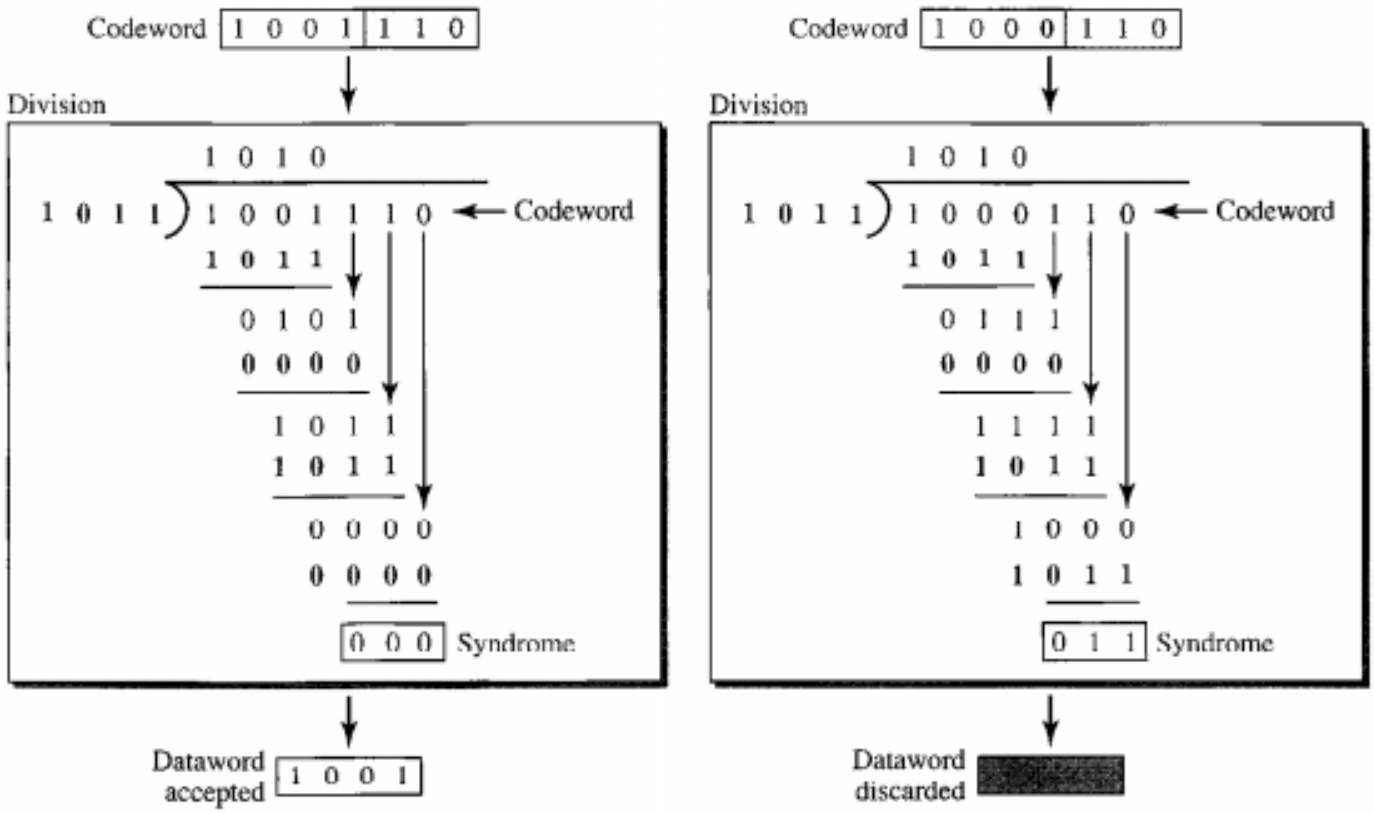
- We can create cyclic codes to correct errors. In the encoder, the dataword has k bits (4 here); the codeword has n bits (7 here).
- The size of the dataword is augmented by adding n - k (3 here) Os to the right-hand side of the word.
- The n-bit result is fed into the generator. The generator uses a divisor of size n - k + I (4 here), predefined and agreed upon.
- The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (r_2 r_1 r_0) is appended to the dataword to create the codeword.
- The decoder receives the possibly corrupted codeword. A copy of all n bits is fed to the checker which is a replica of the generator.
- The remainder produced by the checker is a syndrome of n - k (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all as, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).



Encoder



Decoder


Codeword 1 0 0 1 1 1 0
Division
1 0 1 0
1 0 1 1) 1 0 0 1 1 1 0 ← Codeword
1 0 1 1
0 1 0 1
0 0 0 0
1 0 1 1
1 0 1 1
0 0 0 0
0 0 0 0
0 0 0 Syndrome
Dataword accepted 1 0 0 1
Codeword 1 0 0 0 1 1 0
Division
1 0 1 0
1 0 1 1) 1 0 0 0 1 1 0 ← Codeword
1 0 1 1
0 1 1 1
0 0 0 0
1 1 1 1
1 0 1 1
1 0 0 0
1 0 1 1
0 1 1 Syndrome
Dataword discarded

CHECKSUM

- The checksum is used in the Internet by several protocols although not at the data link layer.
- Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers.
- For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12,0,6,36), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum.
- If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted
- We can make the job of the receiver easier if we send the negative (complement) of the sum, called the checksum. In this case, we send (7, 11, 12,0,6, -36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

One's Complement

- The previous example has one major drawback. All of our data can be written as a 4-bit word (they are less than 15) except for the checksum.
- One solution is to use one's complement arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and 2^n - 1 using only n bits.
- If the number has more than n bits, the extra leftmost bits need to be added to the n rightmost bits (wrapping). In one's complement arithmetic, a negative number can be represented by inverting all bits (changing a 0 to a 1 and a 1 to a 0). This is the same as subtracting the number from 2^n - 1.

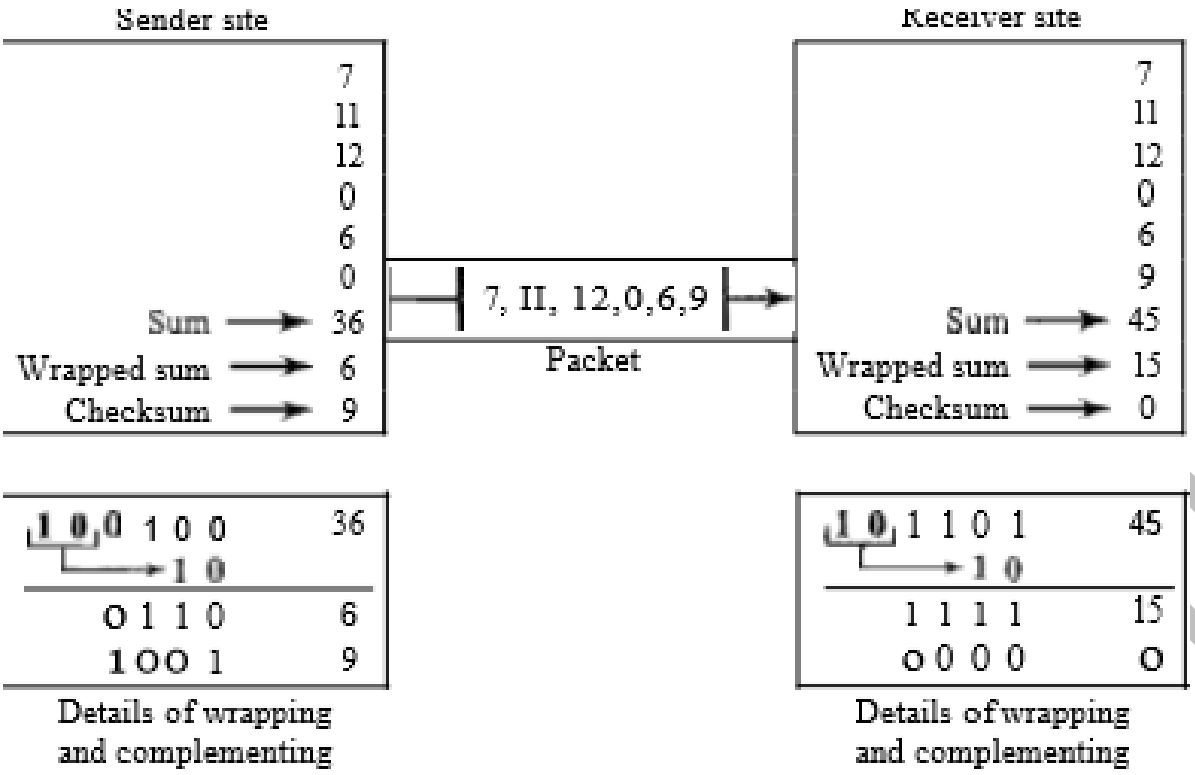
Q How can we represent the number 21 in one's complement arithmetic using only four bits?

Solution The number 21 in binary is 10101 (it needs five bits). We can wrap the leftmost bit and add it to the four rightmost bits. We have (0101 + 1) = 0110 or 6.

Example 10.21

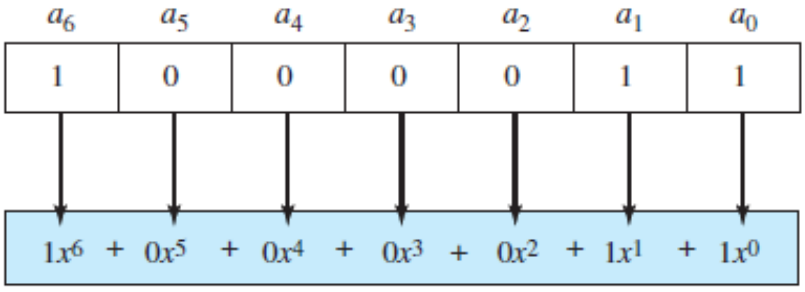
How can we represent the number-6 in one's complement arithmetic using only four bits?

Solution In one's complement arithmetic, the negative or complement of a number is found by inverting all bits. Positive 6 is 0110; negative 6 is 100I. If we consider only unsigned numbers, this is 9. In other words, the complement of6 is 9. Another way to find the complement of a number in one's complement arithmetic is to subtract the number from 2n - I (16 - 1 in this case).

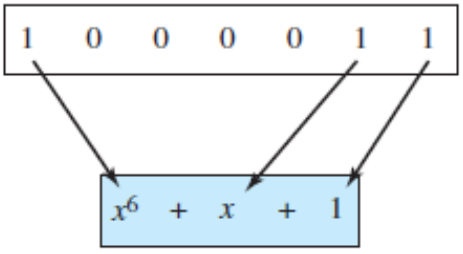


Polynomials

- A pattern of 0s and 1s can be represented as a **polynomial** with coefficients of 0 and 1.
- The power of each term shows the position of the bit; the coefficient shows the value of the bit.



a. Binary pattern and polynomial



b. Short form

- An advantage is that a large binary pattern can be represented by short terms.

Degree of a Polynomial

- The degree of a polynomial is the highest power in the polynomial. For example, the degree of the polynomial $x^6 + x + 1$ is 6.
- The degree of a polynomial is 1 less than the number of bits in the pattern. The bit pattern in this case has 7 bits.

Adding and Subtracting Polynomials

- Adding and subtracting polynomials in mathematics are done by adding or subtracting the coefficients of terms with the same power.
- Addition and subtraction are the same.
- Adding or subtracting is done by combining terms and deleting pairs of identical terms.

Example: Adding $x^5 + x^4 + x^2$ and $x^6 + x^4 + x^2$ gives us:

$x^6 + x^5$ (i.e. we delete the pairs of identical terms)

Note:

- If we add, for example, three polynomials and we get $x2$ three times, we delete a pair of them and keep the third.

Multiplying or Dividing Terms

- In multiplying a term we just add the powers. Example, $x^3 \times x^4$ is x^7.
- For dividing, we just subtract the power of the second term from the power of the first. Example, x^5/x^2 is x^3.

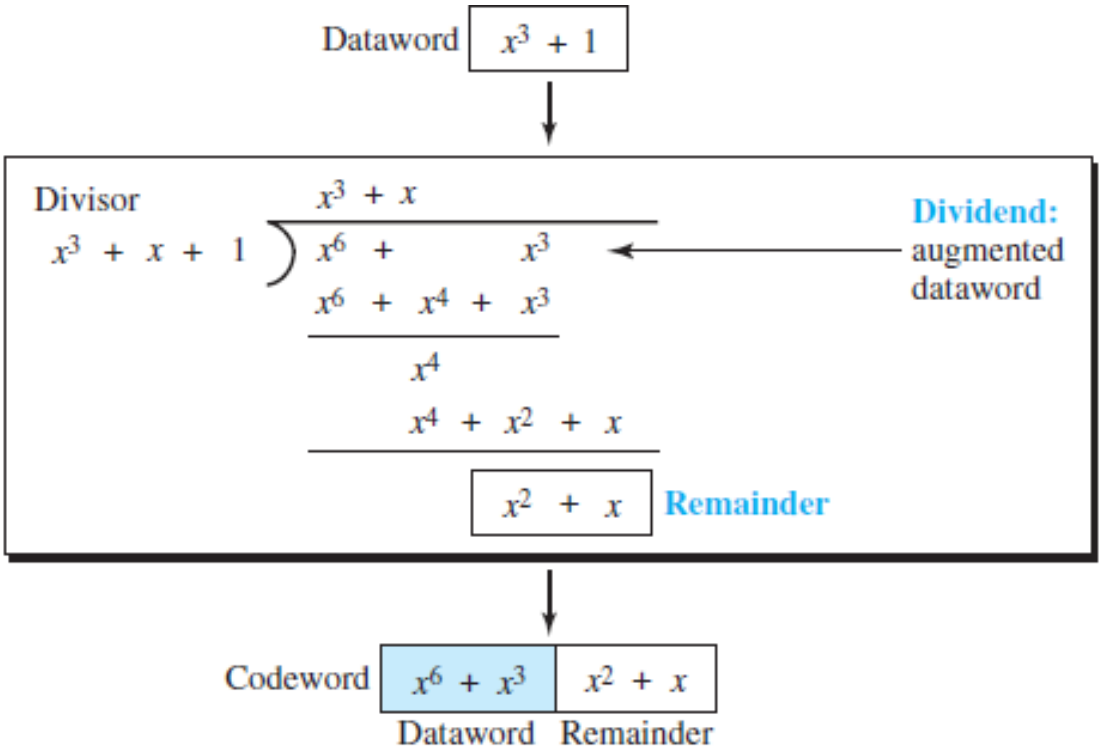
Shifting

Shifting left 3 bits: 10011 becomes 10011000 $x^4 + x + 1$ becomes $x^7 + x^4 + x^3$

Shifting right 3 bits: 10011 becomes 10 $x^4 + x + 1$ becomes x

Cyclic Code Encoder Using Polynomials

- The dataword 1001 is represented as $x^3 + 1$. The divisor 1011 is represented as $x^3 + x + 1$.
- To find the augmented dataword, we have left-shifted the dataword 3 bits (multiplying by x^3).
- The result is $x^6 + x^3$. Division is straightforward. We divide the first term of the dividend, x^6, by the first term of the divisor, x^3.
- The first term of the quotient is then x^6/x^3, or x^3. Then we multiply x^3 by the divisor and subtract (according to our previous definition of subtraction) the result from the dividend.
- The result is x^4, with a degree greater than the divisor's degree; we continue to divide until the degree of the remainder is less than the degree of the divisor.



- The divisor in a cyclic code is normally called the *generator polynomial* or simply the *generator*.

Q The message 11001001 is to be transmitted using the CRC polynomial x^3+1 to protect it from errors. The message that should be transmitted is: **(Gate-2007) (2 Marks)**

a) 11001001000 **b)** 11001001011

c) 11001010 **d)** 110010010011

ANSWER B

Cyclic Code Analysis

- Let us assume: **Dataword: *d*(*x*), Codeword: *c*(*x*), Generator: *g*(*x*), Syndrome: *s*(*x*), Error: *e*(*x*)**
- If *s*(*x*) is not zero, then one or more bits is corrupted. However, if *s*(*x*) is zero, either no bit is corrupted or the decoder failed to detect any errors.
- **Received codeword = c(x) + e(x)**
  Dividing both sides by g(x)

$$\frac{\text{Received codeword}}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

In a cyclic code, those *e*(*x*) errors that are divisible by *g*(*x*) are not caught.

Points to Note about catching errors:

- If the generator has more than one term and the coefficient of x^0 is 1, all single-bit errors can be caught.
- If a generator cannot divide $x^t + 1$ (t between 0 and n - 1), then all isolated double errors can be detected.
- A generator that contains a factor of x + 1 can detect all odd-numbered errors.
- All burst errors with $L \leq r$ will be detected.
- All burst errors with $L = r + 1$ will be detected with probability $1 - (1/2)^{r-1}$.
- All burst errors with $L > r + 1$ will be detected with probability $1 - (1/2)^r$.

Advantages of Cyclic Codes

- Cyclic codes have a very good performance in detecting single-bit errors, double errors, an odd number of errors, and burst errors.
- They can easily be implemented in hardware and software.
- They are especially fast when implemented in hardware.

Error Control

- Error control at the data-link layer is normally very simple and implemented using one of the two methods.
- In both methods, a CRC is added to the frame header by the sender and checked by the receiver.

1. In the first method, if the frame is corrupted, it is silently discarded; if it is not corrupted, the packet is delivered to the network layer.
2. In the second method, if the frame is corrupted, it is silently discarded; if it is not corrupted, an acknowledgment is sent (for the purpose of both flow and error control) to the sender.

Q A computer network uses polynomials over GF (2) for error checking with 8 bits as information bits and uses $x^3 + x + 1$ as the generator polynomial to generate the check bits. In this network, the message 01011011 is transmitted as **(Gate-2017) (2 Marks)**

(a) 01011011010 **(b)** 01011011011
(c) 01011011101 **(d)** 01011011100
Ans: c

Q Consider the following message M = 1010001101. The cyclic redundancy check (CRC) for this message using the divisor polynomial $x^5 + x^4 + x^2 + 1$ is: (Gate-2005) (2 Marks)

(A) 01110 **(B)** 0101 **(C)** 10101 **(D)** 10110
Answer: (A)