# Data Communications

**Prof. A. Pal**
**Department of Computer Science & Engineering**
**Indian Institute of Technology, Kharagpur**

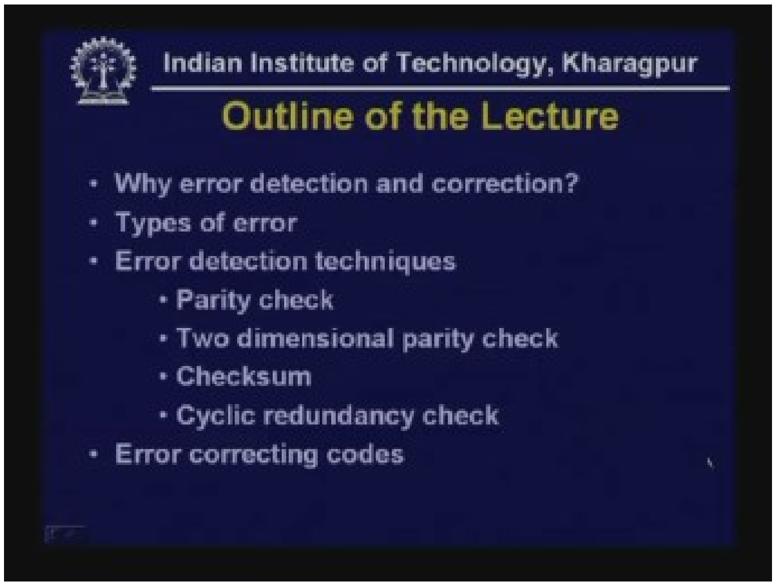
**Lecture-15**

**Error Detection and Correction**

Hello viewers, welcome to today's lecture on Error Detection and Correction**.** We have discussed various issues related to transmission of signal through different transmission media. We have seen that when signal is sent through some transmission media it is subjected to attenuation, distortion and noise. As a result, some of the bits of the data or the signal get corrupted. In other words it leads to some error in sending data. But, we want reliable communication. So, through the unreliable media if we want reliable communication what we have to do. We have to device mechanism for correcting the error that occurs during transmission through the transmission media and that is the topic of today's lecture.

So, here is the outline of the lecture. First we shall discuss why error detection and correction, then the various types of error that can occur during transmission, third topic will be error detection techniques.

(Refer Slide Time: 2:20)



Indian Institute of Technology, Kharagpur

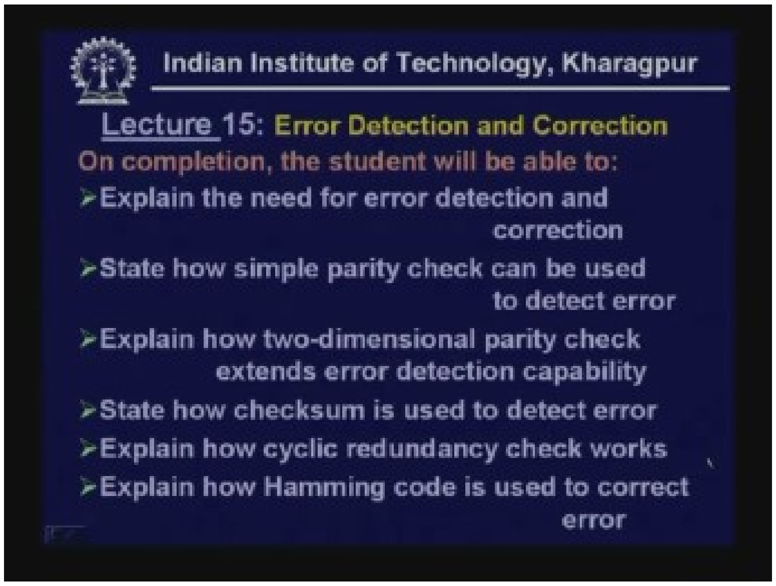
Outline of the Lecture

- Why error detection and correction?
- Types of error
- Error detection techniques
    - Parity check
    - Two dimensional parity check
    - Checksum
    - Cyclic redundancy check
- Error correcting codes

We shall discuss various error detection techniques, such as parity check, two dimensional parity check, checksum, Cyclic Redundancy Check, etc. These are the commonly used error detection techniques. Finally, we shall discuss error correcting codes which can be used not only to detect error but also to correct error if it occurs during transmission.

And on completion of this lecture the students will be able to explain the need for error detection and correction, they will be able to state how simple parity check can be used to detect error, they will be able to explain how two dimensional parity check extends error detection capability because, as we shall see, simple error detection or simple parity check cannot detect all possible errors but only some of the errors are detected.

(Refer Slide Time: 3:03)



Indian Institute of Technology, Kharagpur

Lecture 15: Error Detection and Correction

On completion, the student will be able to:

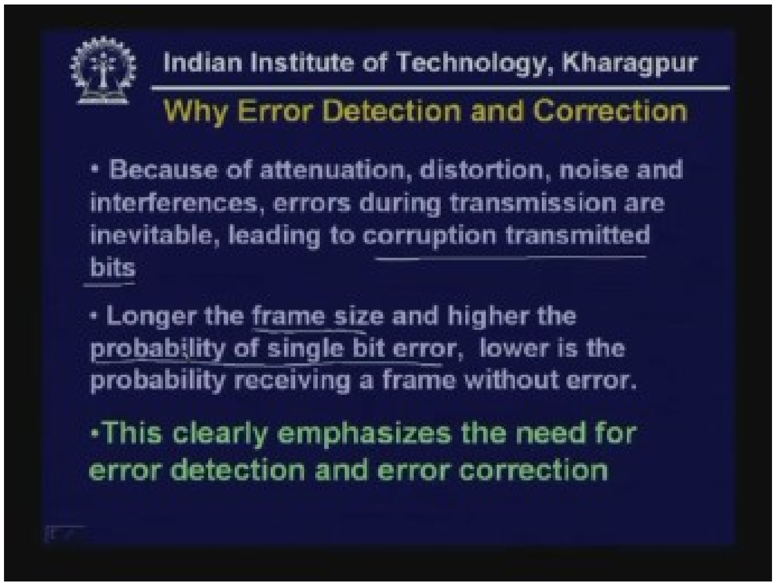
- Explain the need for error detection and correction
- State how simple parity check can be used to detect error
- Explain how two-dimensional parity check extends error detection capability
- State how checksum is used to detect error
- Explain how cyclic redundancy check works
- Explain how Hamming code is used to correct error

So to extend the capability, two dimensional parity check is used and that the students will be able to explain it. Then they will be able to state how checksum is used to detect error; this is another scheme they will be able to state then they will be able to explain how Cyclic Redundancy Check works. Finally, the students will be able to explain how hamming code is used not only to detect error but correct error.

So, we start with why error detection and correction. As I mentioned, because of attenuation, distortion, noise, interferences some errors will occur and this will lead to corruption of some of the transmitted bits. And particularly two parameters which are important in the context of error is your frame size and second is the probability of single-bit error.

Obviously if the transmission media is noisy the probability of single-bit error will be more and when we are sending data in the form of a frame the size of the frame will also decide the probability of error of a frame. That means two important parameters such as frame size and single-bit error will decide whether we are receiving a frame with error or without error. This can be proved mathematically that longer the frame size and higher the probability of single-bit error lower is the probability of receiving frame without error.

(Refer Slide Time: 5:01)



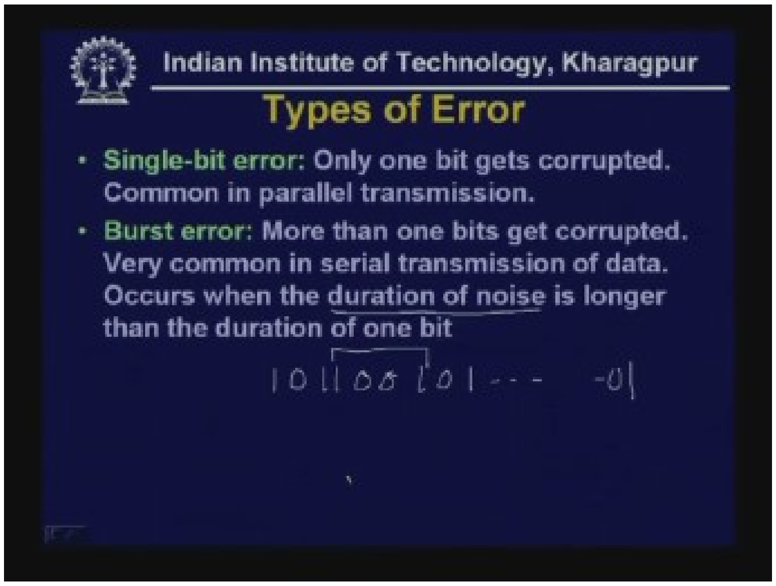
Indian Institute of Technology, Kharagpur

## Why Error Detection and Correction

- Because of attenuation, distortion, noise and interferences, errors during transmission are inevitable, leading to corruption transmitted bits
- Longer the frame size and higher the probability of single bit error, lower is the probability receiving a frame without error.
- This clearly emphasizes the need for error detection and error correction

That means the probability of receiving a frame without error increases with the size of the frame and also the probability of single-bit error increases. This is quite obvious. If the frame size is long, say if you have got a long sequence of zeroes and ones, then if one of the bits get corrupted then the entire frame will be in error. So, this clearly emphasizes the need for error detection and error correction. If we want to communicate in a reliable manner, then we have to device a scheme so that the error which is inevitable can be corrected. Let us look at the various schemes but before we look at the various schemes let us see the types of error.

(Refer Slide Time: 7:06)



Indian Institute of Technology, Kharagpur

## Types of Error

- **Single-bit error:** Only one bit gets corrupted. Common in parallel transmission.
- **Burst error:** More than one bits get corrupted. Very common in serial transmission of data. Occurs when the duration of noise is longer than the duration of one bit

1 0 1 1 0 0 1 0 1 - - - -0 1

First one is single-bit error. Suppose you are sending a bit sequence, may be a character 1 0 1 1 0 0 1 a 7-bit say ASCII code, one of the bits may get corrupted and it may become 1 0 1 0 0 0 1. That means this corresponds to a particular character (Refer Slide Time: 6:28) and this has been transmitted and this will be received by the receiver and obviously the received character is different from the transmitted character. This has occurred due to single-bit error. So this is called single-bit error and this is very common in parallel transmission.

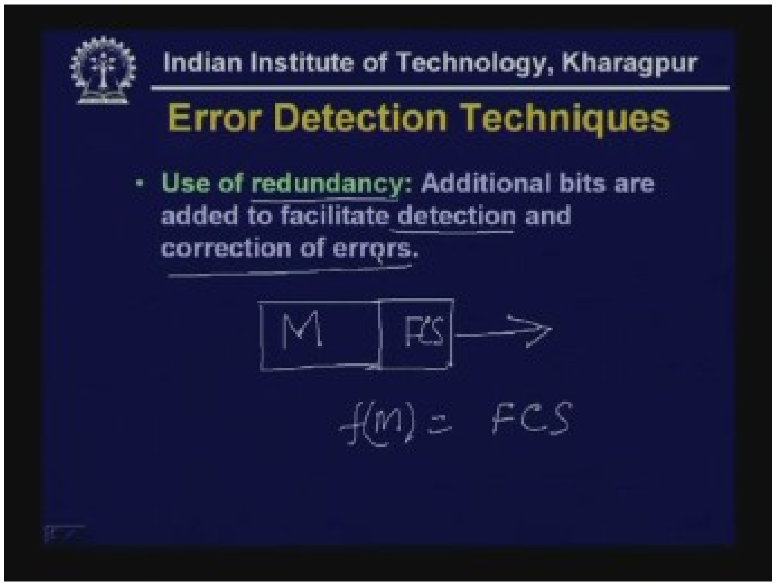
Suppose you are sending eight bit simultaneously using eight lines; so using eight lines you are sending and whenever you are sending eight bits then one of the line may be faulty suppose this is faulty (Refer Slide Time: 7:17) whenever a particular line is faulty obviously the signal coming through this line will be in error. On the other hand, signals coming through other lines may not be in error and as a result this will lead to a single-bit error if a particular line is faulty. This is how single-bit error occurs particularly in parallel transmission.

Then there is another type of error which is known as burst error. Burst error occurs whenever more than one bit gets corrupted. Suppose you are sending a long sequence 1 0 1 1 0 0 1 0 1 and so on, then what can happen is the duration of noise is longer than the duration of one bit. suppose duration of noise is this much; this is the duration of noise (Refer Slide Time: 8:05) so duration of noise is covering four bits, so there is a possibility that some of this bits say this becomes 0, this becomes 1, this remains 0, this becomes 1. So it is not necessary that all the bits will get corrupted. So, out of these four bits the three bits have got corrupted leading to this signal at the receiving end.

This is essentially multiple bit error and we call it burst error because a burst in noise is responsible for this kind of error. This is very common in serial transmission. Whenever you are transmitting serially, then during transmission the transmission line may be exposed to some noise, may be switching on some electrical equipment or some other electric disturbances like it may be due to lightening because of bad weather and so on. So, because of any one of this reasons multiple bits get corrupted because of the duration of noise is longer than one bit and this is the burst error. So, we have to develop technique for detecting not only single-bit error but multiple errors including burst error.

What are the techniques and how it is being done?

(Refer Slide Time: 10:50)


Indian Institute of Technology, Kharagpur

## Error Detection Techniques

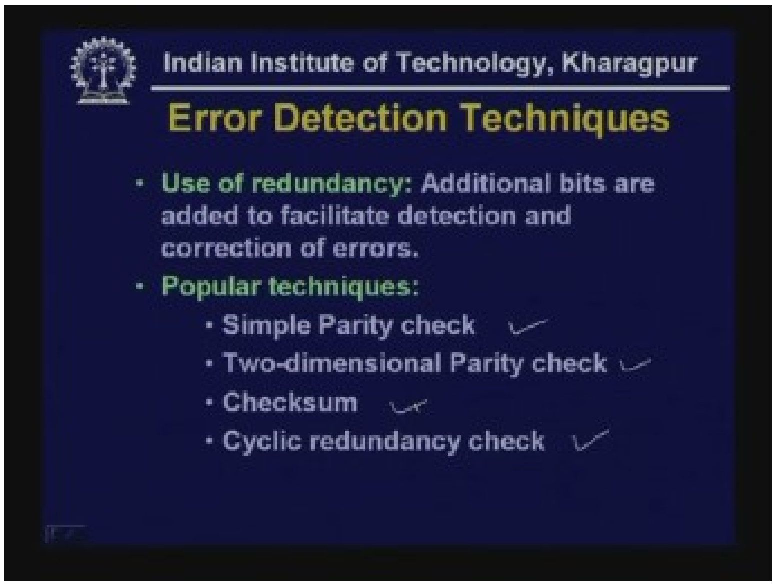
- Use of redundancy: Additional bits are added to facilitate detection and correction of errors.

The basic approach is use of redundancy. Suppose this is the message; the message may comprise several bits in sequence and with that some additional bits are used known as (Frame Check Sequence) FCS. This Frame Check Sequence is added with that additional bit and then sent over the transmission line. So, these are the additional bits (Refer Slide Time: 10:05) which you may call redundancy incorporated in the message and that is being sent and these additional bits can be used for not only for detecting error but also for correction of error. That is the basic mechanism of error detection and correction.

Obviously, this FCS is a function of M, so FCS will be dependent on this message. So, based on this message bit this Frame Check Sequence is calculated and then it is sent and these together are sent to the receiver and they are using this Frame Check Sequence and message bits and by this the error detection and correction is done. We shall see how exactly it is being done.

The popular techniques are simple parity check where single-bit is added, then to enhance the performance two dimensional parity check is commonly used and the third technique we shall discuss is checksum which adds several bits to detect error and error detection capability is better than simple parity check when compared to two dimensional parity check. Finally, we shall discuss Cyclic Redundancy Check which is very popular and common and possibly this is the most sophisticated technique used for error detection. We shall discuss these techniques one after the other.

(Refer Slide Time: 11:31)


Indian Institute of Technology, Kharagpur

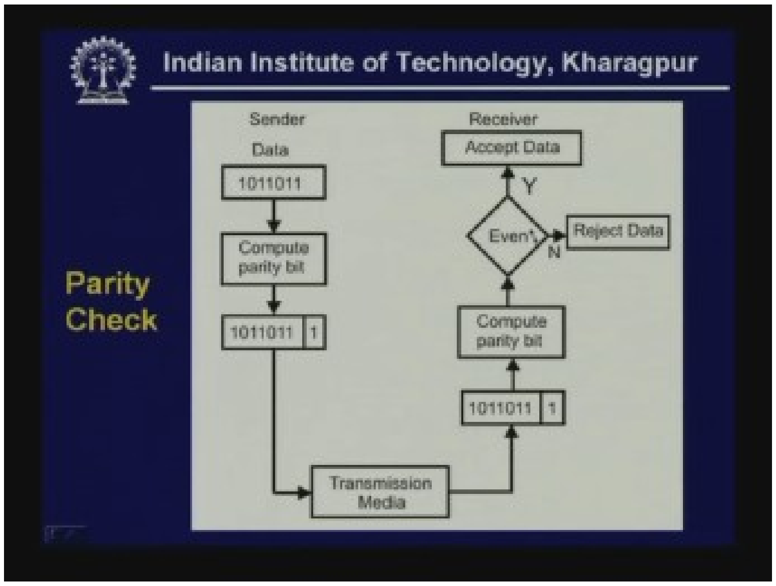
Error Detection Techniques

- Use of redundancy: Additional bits are added to facilitate detection and correction of errors.
- Popular techniques:
  - Simple Parity check ✓
  - Two-dimensional Parity check ✓
  - Checksum ✓
  - Cyclic redundancy check ✓

As I mentioned, the parity check is the simplest and the most popular error detection scheme. What it does is, it appends a parity bit to the end of the data. Suppose you are sending a character say 1 0 1 1 0 0 1, so 7-bit now you can add a parity bit and that parity bit is added based on whether you are using even parity or odd parity. If you are using even parity, then including that bit the number of ones should be even. So, here you have got four ones, so this is made 0 whenever you are using even parity, so that the total number of bits in the message that is being sent including parity is even.

On the other hand, if we are using odd parity, then it will be 1 0 1 1 0 0 1 1. So, instead of 0, 1 is added so that the number of ones is odd, so this is the odd parity. Particularly, as we have discussed, the asynchronous and synchronous mode of transmission, in asynchronous mode commonly the odd parity is used and in synchronous mode of transmission normally the even parity is used; so both are in use. Let’s see how it is exactly being done. So, this is the basic scheme of parity check. Here it is based on even parity as it is shown here.

(Refer Slide Time: 13:10)

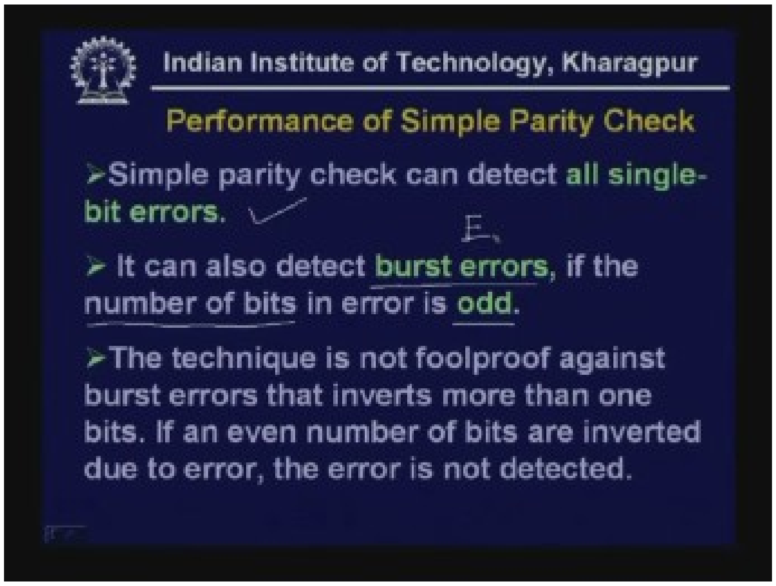

Indian Institute of Technology, Kharagpur

## Parity Check

This is the data 1 0 1 1 0 1 1 that is being sent and the parity is calculated using very simple hardware using (Exclusive OR) Ex-OR gate for computing the parity bit and as you can see here you have got odd number of ones in the message so another one is added as parity bit to make the number of ones even. So, these data bits along with this parity bit are sent through the transmission media and it reaches the receiving end. In the receiving end the parity bit is computed. That means here all the bits are used to check whether the number of ones is even or not.

If it is even, then you accept the data because it is the correct one and on the other hand if the number of bits is odd then you have to reject the data. In that case some error has occurred leading to single-bit error. So, in that case you have to reject the data at the receiving end. This is the basic scheme of using the parity check. Obviously, to compute the parity bit you can use Exclusive OR gates in both cases. This is the performance of simple parity check scheme.

(Refer Slide Time: 15:03)



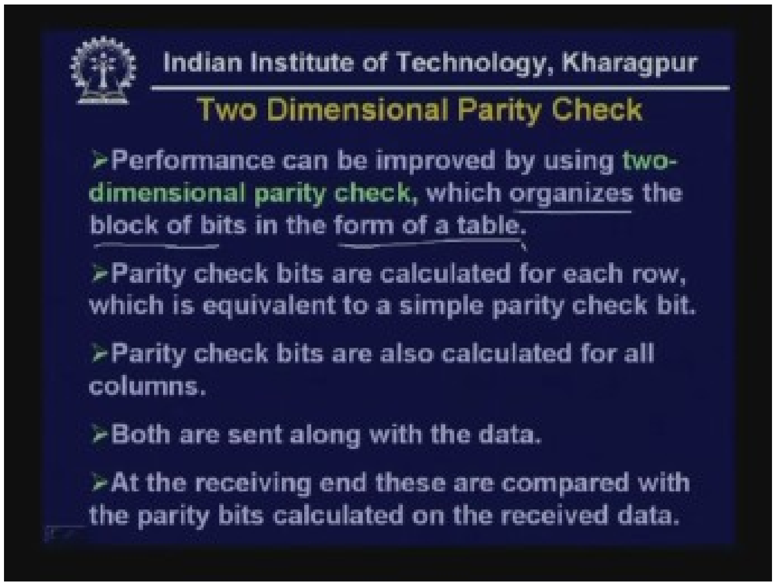
Indian Institute of Technology, Kharagpur

Performance of Simple Parity Check

- Simple parity check can detect all single-bit errors.
- It can also detect burst errors, if the number of bits in error is odd.
- The technique is not foolproof against burst errors that inverts more than one bits. If an even number of bits are inverted due to error, the error is not detected.

Here, all single-bit errors are detected and it can detect all the single-bit errors and also it can detect burst errors. However, this can detect burst errors only when the number of bits in error is odd. Obviously, when the number of bits in error is even then if it is 0, it becomes 1, if another one is 1 it becomes 0 leading to the same parity bit. That's why whenever even number of bits is corrupted then this simple parity check cannot detect the error. So, we find that this technique is not foolproof against burst errors that inverts more than one bit. Particularly, if the even number of bits is inverted due to error the error cannot be detected by using this simple parity check scheme.

(Refer Slide Time: 16:05)



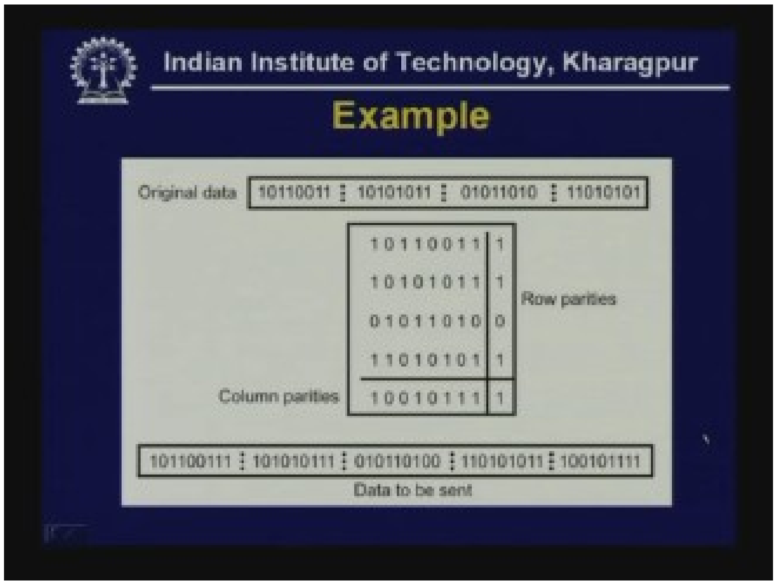
Indian Institute of Technology, Kharagpur

Two Dimensional Parity Check

- Performance can be improved by using two-dimensional parity check, which organizes the block of bits in the form of a table.
- Parity check bits are calculated for each row, which is equivalent to a simple parity check bit.
- Parity check bits are also calculated for all columns.
- Both are sent along with the data.
- At the receiving end these are compared with the parity bits calculated on the received data.

To overcome this limitation, we can use two dimensional parity check. In two dimensional parity check, the performance is improved by using more number of bits that means more redundancy. Here the data is recognized in blocks of bits in the form of a table. So, here since it is organized in a form of a table, it takes the shape of a two dimensional data and parity check bits are calculated for each row which is equivalent to simple parity check bits. For each row there is a parity bit and parity check bits are also calculated for all columns. So you have parity check for each of the rows and parity check bit for each of the columns and then both are sent along with the data and at the receiving end these are compared with the parity bits calculated on the received data.

(Refer Slide Time: 16:47)



Indian Institute of Technology, Kharagpur

## Example

Original data 10110011 ⋮ 10101011 ⋮ 01011010 ⋮ 11010101

| | | | |
|---|---|---|---|
| | 1 0 1 1 0 0 1 1 | 1 | Row parities |
| | 1 0 1 0 1 0 1 1 | 1 | |
| | 0 1 0 1 1 0 1 0 | 0 | |
| | 1 1 0 1 0 1 0 1 | 1 | |
| Column parities | 1 0 0 1 0 1 1 1 | 1 | |

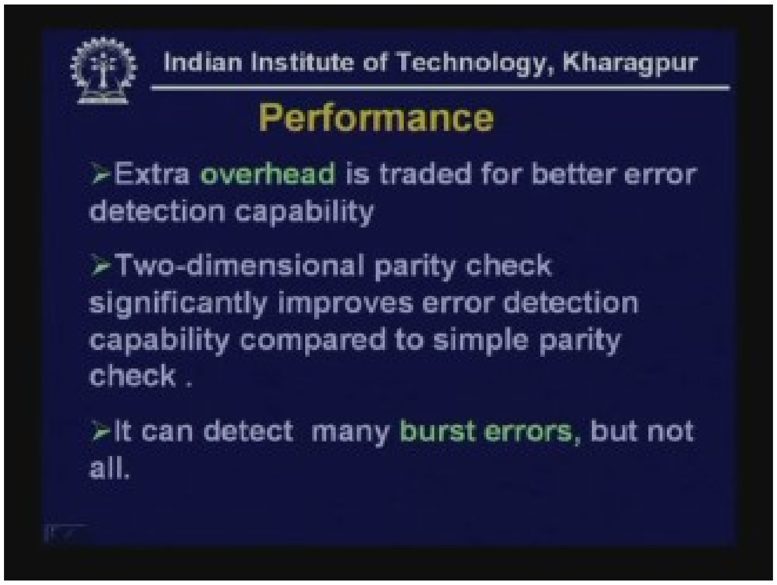
101100111 ⋮ 101010111 ⋮ 010110100 ⋮ 110101011 ⋮ 100101111

Data to be sent

So, let us take up an example and see how it exactly works. This is the original data. Here the entire data has been divided into four segments or blocks then they are organized in the form of a table. Here you have got four rows corresponding to each segment or block then for this particular row the even parity is calculated so you can see here the total number of bits is even including the parity bit. So in this way, for each of these rows the parity bits are calculated and then for each of these columns also the parity bits are calculated and even parity is added at the bottom.

So, for each of these columns you can see the total number of ones you will find is even because this is the even parity. Then for each row, parity bits are also calculated which makes the total number of parity bits. So, we find that we send four segments of data each eight bit. That means thirty two bits are to be sent and to send that the total number of additional bits is now nine bits in each block and you have got five blocks so nine into five that means forty five bits. So, you see here that in simple parity check only one additional bit is needed but here the number of additional bits is increasing. In other words, the overhead is much more than the simple parity check. Obviously, this is done to achieve better performance.

(Refer Slide Time: 18:53)


Indian Institute of Technology, Kharagpur

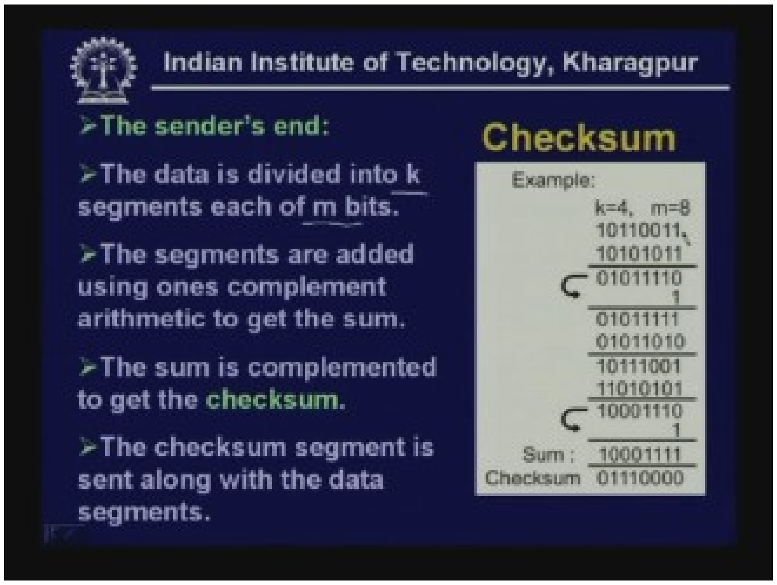
## Performance

- Extra overhead is traded for better error detection capability
- Two-dimensional parity check significantly improves error detection capability compared to simple parity check .
- It can detect many burst errors, but not all.

So, the extra overhead is traded for better error detection capability. How? For example, it can detect not only the odd number of bits but also it significantly improves the error detection capability compared to simple parity check. It can detect most of the vast number of errors but obviously it cannot detect all. Let us take up that example. Here suppose these two of bits get altered and here also these two bits get altered so four bits get altered. Now, if only these two bits get altered then column parity bits can be used to detect error. However, in this column one of these two bits gets altered and in the same column these two bits get altered then the parity bits here remain unaltered. As a consequence, error detection is possible. That's why all the burst errors cannot be detected. However, it detects most of the burst errors.

Now, let us see the third scheme that is based on checksum. In checksum at the sender's end again the data is divided into k segments each of m-bits. So, just like the two dimensional parity bit, here also data is divided in segments. In this example we have taken up the same example as you have shown for two dimensional parity bits. So you have got four segments k = 4, each of eight bits m-bits.

(Refer Slide Time: 20:38)



Indian Institute of Technology, Kharagpur

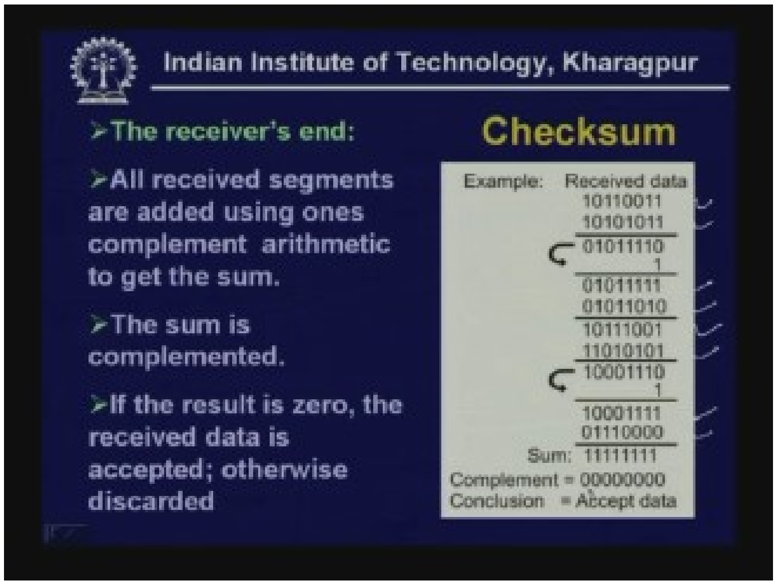
## Checksum

- The sender's end:
- The data is divided into k segments each of m bits.
- The segments are added using ones complement arithmetic to get the sum.
- The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.

```
Example:
             k=4,  m=8
             10110011
             10101011
             --------
             01011110
                    1
             --------
             01011111
             01011010
             --------
             10111001
             11010101
             --------
             10001110
                    1
             --------
Sum :        10001111
             --------
Checksum     01110000
```

Now the segments are added using ones complement arithmetic to get the sum. You may be familiar with ones complement arithmetic. Here what is done is here the addition is done for each pair of the segments one after the other and as we know whenever there is an end around carry it is added with the least significant bit to get the sum. So this is the partial sum by adding first two segments then the third segment is added and here there is no end around carry so this is the partial sum then the fourth segment is added and here we find there is end around carry so it is added so in this way we get this partial sum 1 0 0 0 1 1 1 1. This partial sum is now complemented to get the checksum. So here we get the checksum 0 1 1 1 0 0 0 0. So this checksum is sent along with the four segments. So what we are doing here is along with the data segments we are sending a message segment equal to m-bits. These are now received at the receiving end and the checking is done in this manner. All the received segments are added using ones complement arithmetic to get the sum.

(Refer Slide Time: 22:54)



Indian Institute of Technology, Kharagpur

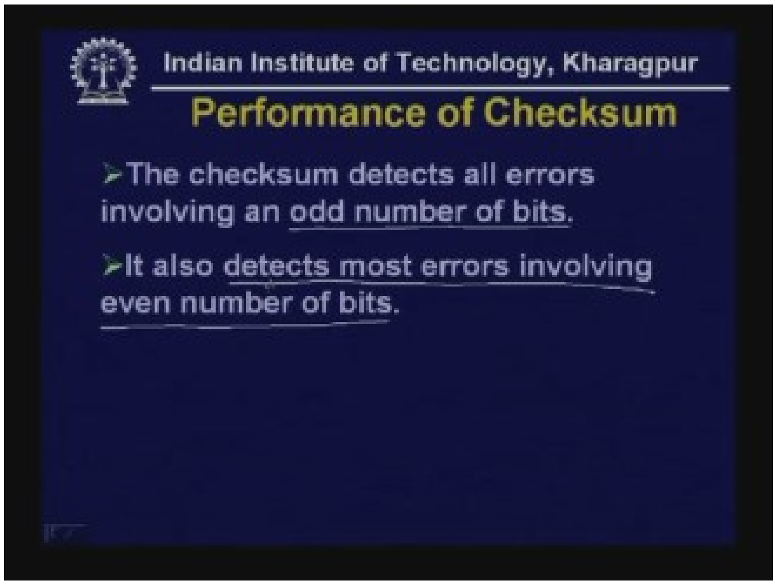
## Checksum

- The receiver's end:
- All received segments are added using ones complement arithmetic to get the sum.
- The sum is complemented.
- If the result is zero, the received data is accepted; otherwise discarded

```
Example:   Received data
              10110011
              10101011
              --------
              01011110
                     1
              --------
              01011111
              01011010
              --------
              10111001
              11010101
              --------
              10001110
                     1
              --------
              10001111
              01110000
              --------
        Sum:  11111111
Complement = 00000000
Conclusion = Accept data
```

As you can see here the first part is the same as the previous case. We have assumed here that no error has taken place so first two segments are added to get this partial sum then the third segment is added to get this partial sum then the fourth segment is added to get this partial sum and then the additional segment which is essentially the checksum-bits are added to get this partial sum. Now the sum is complemented and if the result is all 0 then it is assumed that no error has taken place and conclusion is accept data. So, if it is all not 0 then this data is not accepted and it is considered that some errors occurred in anyone of these bits.

(Refer Slide Time: 23:52)



Indian Institute of Technology, Kharagpur

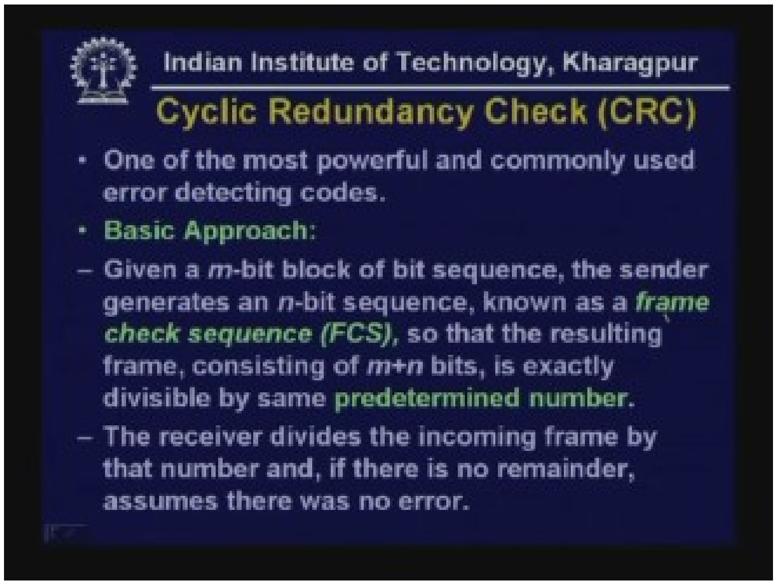
## Performance of Checksum

- The checksum detects all errors involving an odd number of bits.
- It also detects most errors involving even number of bits.

Let's see the performance of checksum. Like single parity or two dimensional parity it can detect all errors involving odd number of bits. So here the performance is same as the simple parity check. However, it can detect most of the errors involving even number of bits. That means the burst errors can also be detected and whenever the number of errors is even in most of the cases it can be done but not in all cases. So we find that the performance of checksum is comparable to that of two dimensional parity check but the overhead is little less.

As you can see here (Refer Slide Time: 24:07) overhead is lesser than the two dimensional parity check. Here we are sending for 32 bit so here the data was 32 bit so it is 32 bit + 8 bit that means it is 40 compared to 45 in case of two dimensional parity check. So in this case we have some extra overhead but less than the two dimensional parity check.

(Refer Slide Time: 24:53)



Indian Institute of Technology, Kharagpur

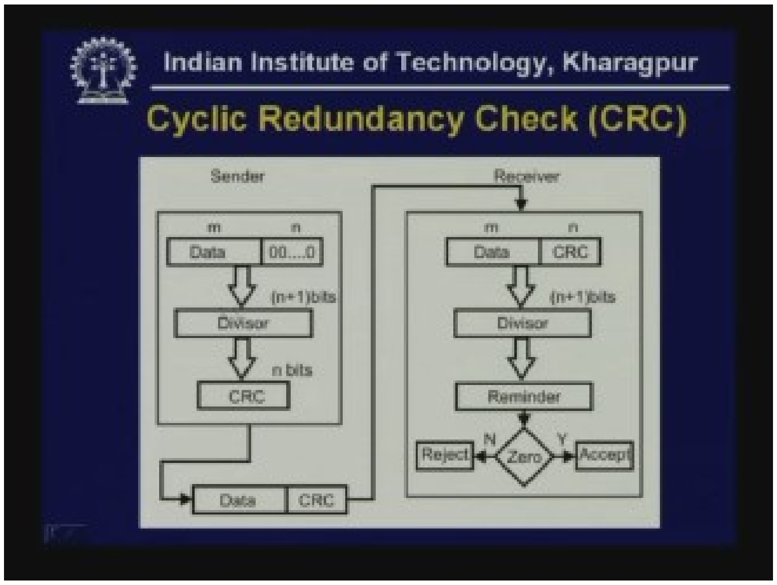
## Cyclic Redundancy Check (CRC)

- One of the most powerful and commonly used error detecting codes.
- Basic Approach:
  - Given a *m*-bit block of bit sequence, the sender generates an *n*-bit sequence, known as a *frame check sequence (FCS)*, so that the resulting frame, consisting of *m+n* bits, is exactly divisible by same predetermined number.
  - The receiver divides the incoming frame by that number and, if there is no remainder, assumes there was no error.

Now let us come to the fourth scheme and the final scheme for error detection that is the Cyclic Redundancy Check. The Cyclic Redundancy Check is possibly one of the most powerful and commonly used error detecting codes. We shall see why it is so. We shall see that Cyclic Redundancy Check gives you the best performance in terms of the error detection quality. The basic approach is given here.

Given a m-bit block of bit sequence the sender generates an n-bit sequence and additional bits known as Frame Check Sequence FCS. So for each of m-bits if your block of data sequence is m-bits then additional n-bits sequence is calculated based on some scheme so that the resulting frame consisting of m + n-bits is exactly divisible by some predetermined number. Here is the most important part of the story. We are using a predetermined number and this is a very carefully chosen predetermined number which is used to generate this n-bit and as we shall see this bit is generated by division. The parity check and the checksum uses the addition. Here we shall see we are using division to generate the Frame Check Sequence. So the receiver divides the incoming frame by that number and if there is no remainder it assumes that there was no error. So let's see how it is being done. So here you have got the data of m-bits then additional 0s are concatenated with it so (Refer Slide Time: 26:50) this is divided by n + 1 bit divisor.

(Refer Slide Time: 26:51)

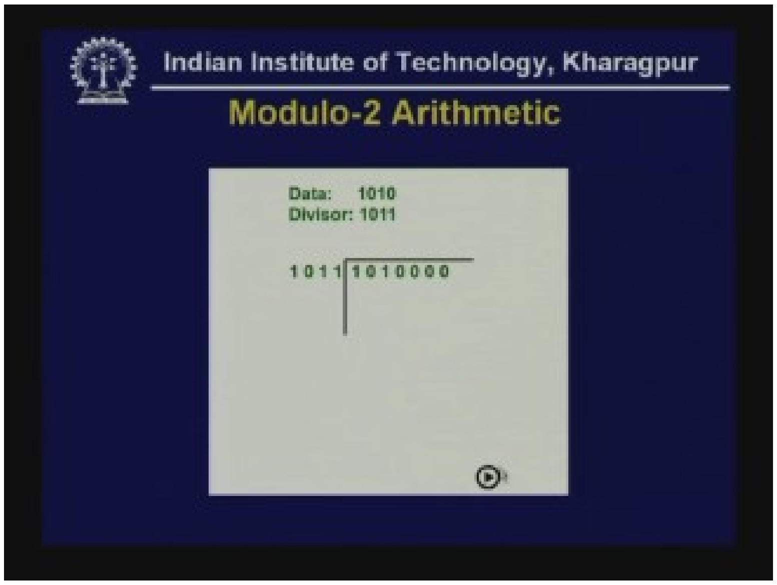


Indian Institute of Technology, Kharagpur

## Cyclic Redundancy Check (CRC)

This is the carefully chosen n-bit sequence which is used as divisor to divide the m + n-bit sequence and it gives you n-bit data. As you know whenever you divide a number by n + 1 bit we get the reminder of n-bits and that n-bit is used as CRC or the Cyclic Redundancy Check bits. These n zeroes are replaced by CRC to get the data so this is what is being transmitted to the transmission media. So, through the transmission media that is sent at the receiving end the receiver receives m + n-bits data plus CRC and it is again divided by the same divisor so the divisor has to be same in both cases both at the transmitting and the receiving end and after division it generates a reminder and if the reminder is 0 then the data is accepted as correct data and if it is not 0 then it is rejected. So we find that at the receiving end as well as in the transmitting end division is performed to generate CRC and also to check whether the received data including CRC is correct or not and then it is accepted or rejected based on whether the reminder is 0 or not. This division is based on Modulo-2 arithmetic and how it is done it is shown with this example.

(Refer Slide Time: 28:38)



Indian Institute of Technology, Kharagpur

## Modulo-2 Arithmetic

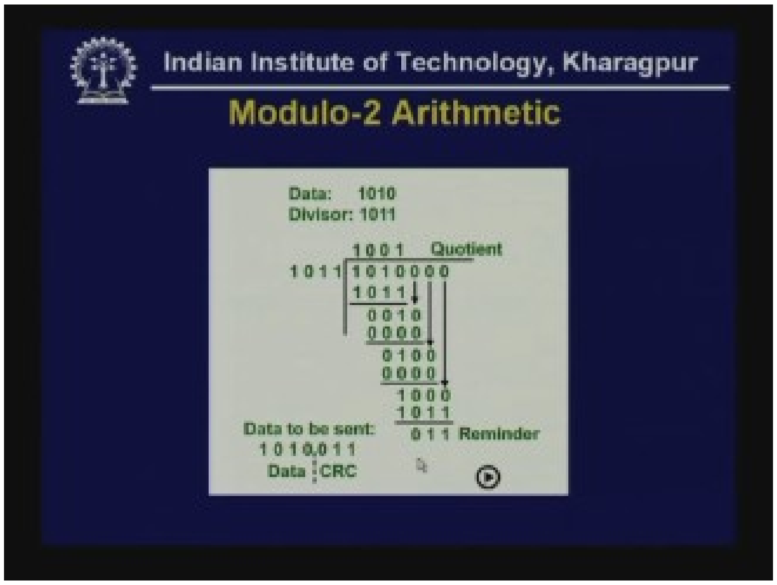
Data: 1010
Divisor: 1011

```
1011 | 1010000
```

So here it is assumed that data is of only four bits 1 0 1 0 and divisor is 1 0 1 1. Although I have shown here four bits for simplicity the data length can be 8-bit, 16-bit or even much longer and as we have seen then the divisor is n + 1 bits so n zeroes are added with it 1 0 1 0 then 0 0 0 so this is being divided by this divisor. This division process is simple this is the partial quotient so since this is 1 so 1 0 1 1 then-bit by bit Exclusive OR operation is performed to get this remainder (Refer Slide Time: 29:25).

So one bit is excluded because the remainder has to be n-bit so 0 0 1 and this comes from this bit and again it is divided and since this quotient bit is 0 so it is 0 0 0 0 here and again Exclusive OR operation is performed to get this 0 1 0 0 and this comes here again this is divided to get the quotient bit 0 and after Modulo-2 arithmetic we get the 1 0 0 then this 0 comes (Refer Slide Time: 30:02) and finally the quotient is 1 0 1 1 so you get 1 0 1 1 and by Modulo-2 arithmetic we get the remainder 0 1 1 and the quotient 1 0 0 1.

(Refer Slide Time: 30:22)



Indian Institute of Technology, Kharagpur

## Modulo-2 Arithmetic

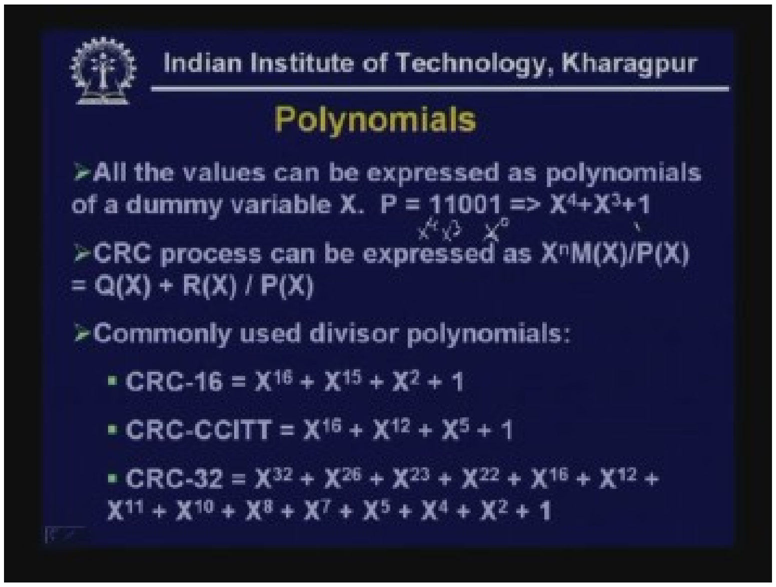
Data: 1010
Divisor: 1011

```
         1 0 0 1     Quotient
1 0 1 1 |1 0 1 0 0 0 0
         1 0 1 1
           0 0 1 0
           0 0 0 0
             0 1 0 0
             0 0 0 0
               1 0 0 0
               1 0 1 1
                 0 1 1  Reminder
```

Data to be sent:
1 0 1 0 0 1 1
Data ¦ CRC

So this remainder as we know is used as CRC to give you the data to be sent. So here we get the data plus the CRC. This is the CRC and here we can see how it is generated by Modulo-2 division (Refer Slide Time: 30:24). So this data and CRC is sent to the transmitting end and at the transmitting end in place of 1 0 1 0 0 0 0 this 1 0 1 0 0 1 1 is divided by 1 0 1 1. And you will find that if we do the division in the same way the remainder will be 0 0 0 so this is how Modulo-2 arithmetic division is performed to generate the CRC by the transmitter and to check whether the received data is correct or not at the receiving end.

(Refer Slide Time: 32:13)



Indian Institute of Technology, Kharagpur

## Polynomials

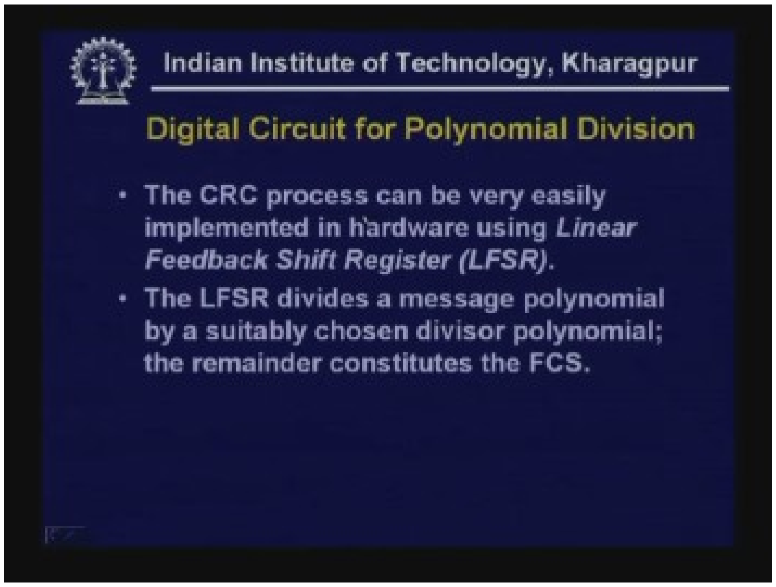
- All the values can be expressed as polynomials of a dummy variable X. P = 11001 => $X^4+X^3+1$
  $X^4$ $X^3$ $X^0$
- CRC process can be expressed as $X^nM(X)/P(X) = Q(X) + R(X) / P(X)$
- Commonly used divisor polynomials:
  - CRC-16 = $X^{16} + X^{15} + X^2 + 1$
  - CRC-CCITT = $X^{16} + X^{12} + X^5 + 1$
  - CRC-32 = $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + 1$

Actually the same approach can be expressed in a different way. The 1 0 sequence can be represented or quite often it is represented by a polynomial. So here we find a bit sequence 1 1 0 0 1 can be represented as a polynomial of a dummy variable x. So this corresponds to the bit x to the power 0 so this is 1, this corresponds to the bit x to the power 1 (Refer Slide Time: 31:46) and since it is 0 it does not appear in the polynomial, this bit is also 0 it does not appear in the polynomial then this corresponds to x to the power 3 so this is present here then this bit corresponds to x to the power 4 so corresponding to each of these ones there is a factor in the polynomial.

On the other hand the zeroes do not give you any factor in the polynomial. So we find that this is a more concise way of representing the bit sequence and particularly the division operation can be expressed as…….. that means the message which comprises of m-bits so that means comprising m-bits whenever you multiply x to the power n this becomes m + n-bits and that is being divided by standard polynomial known as characteristic polynomial P(X) where we get a remainder and that remainder is used as the CRC check bits.

There are some standard polynomials based on some strong mathematical property where these are used for the purpose of generating CRC check bits. For example, CRC-16 is commonly used in ATM and it has got some other applications. Here this has the degree of 16 but only four factors are here as you see; x to the power 16, x to the power 15, x power 2 + 1. That means the remaining bits are 0 CRC, CCITT is used in many applications, this is also 16-bit but it has also got four factors and it is based on that it should not be divisible by x and it should be divisible by x + 1 so these are the two minimum properties that is being satisfied and based on that these are generated. And there is another polynomial x to the power 32 bit and x to the power 16 and so on. So we see this is a concise way of representing the bit sequence in the form of a polynomial of dummy variable x.

(Refer Slide Time: 34:30)


Indian Institute of Technology, Kharagpur

## Digital Circuit for Polynomial Division

- The CRC process can be very easily implemented in hardware using *Linear Feedback Shift Register (LFSR).*
- The LFSR divides a message polynomial by a suitably chosen divisor polynomial; the remainder constitutes the FCS.

Now how do you really perform the division?
I have shown how the division can be done manually. Question arises how the machine will do it both at the transmitting and at the receiving end. This can be done with a help of hardware which is known as Linear Feedback Shift Register or in short LFSR. there is a shift register with some feedback which are used and this Linear Feedback Shift Register can be used to generate the Frame Check Sequence or the CRC and also at the receiving end it can be used for checking whether the received information is correct or not. The Linear Feedback Shift Register will have n number of flip-flops at most n Ex OR gates. The number of Ex OR gates can be less but the number of flip-flop will be exactly n and the LFSR divides a message polynomial by a suitably chosen divisor polynomial. The divisor polynomial is of degree n. then the LFSR will have n flip-flops and the remainder will constitute n-bit which is the Frame Check Sequence. Let us see how exactly it is done.

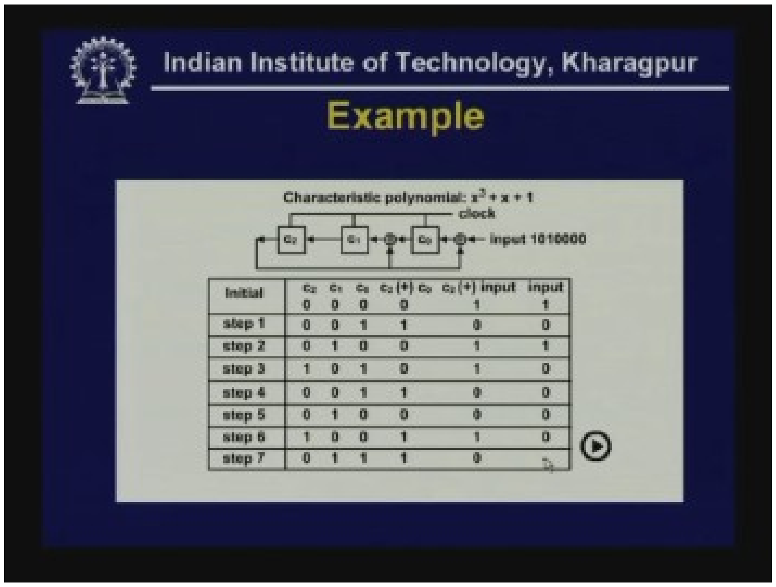
(Refer Slide Time: 36:35)

This is a simple characteristic polynomial of degree 3. Since it is of degree 3 obviously the number of flip-flops will be 3 that is 1 2 3 $C_0$ $C_1$ $C_2$ so there are three flip-flops and here we find two Exclusive gates are used in the feedback path corresponding to x1 and x0 and x3 is the feedback then there is no need for another Exclusive OR gate so two Exclusive OR gates and these three flip-flops will make the necessary hardware for performing the polynomial division. let's see how exactly it can be done.

This is the initial condition (Refer Slide Time: 36:59) where these flip-flops are initialized with the value 0 0 0 so $C_0$ $C_1$ and $C_2$ all bits are reset, these are reset pins these are reset initially and as you can see the input here is 1 so the input to this flip-flop is 1 on the other hand this bit is 0 and this bit is also 0 so input to this flip-flop is 0. That means input to this and input to this flip-flop is given here on this column and this column and since this output is directly connected this $C_1$ output is directly connected to the input of $C_2$.

On the other hand to $C_0$ $C_2$ Exclusive OR input bit is applied and to $C_1$ $C_0$ Exclusive OR $C_2$ bit is applied. So after we apply one clock whatever is here will be latched into this flip-flop and whatever is here will be latched into this flip-flop and the output of $C_2$ will be latched into $C_2$. So let us apply one clock and see how it happens. Therefore this $C_1$ will come here, $C_0$ will come here and this $C_1$ will come here after applying one clock (Refer Slide Time: 38:15) next we apply another clock and we find that this 0 will come here, this 1 will come here and this 0 will come here so in this way we have to apply seven clocks because we have got 7-bits so each of them has to be fed into….., in the step 3 again this 1 comes here, 0 comes here and 1 comes here and in step four same way this 1 comes here, 0 comes here and this 0 comes here, in step 5 again this 0 comes here this 1 comes here and this 0 comes here and these inputs are already fed and then finally these seven inputs are fed. So, this 0 comes here, this 0 comes here and this 1 comes here and finally we have to apply another clock to get the final CRC. So seven clocks are to be applied 1 2 3 4 5 6 7.

(Refer Slide Time: 39:12)



Indian Institute of Technology, Kharagpur

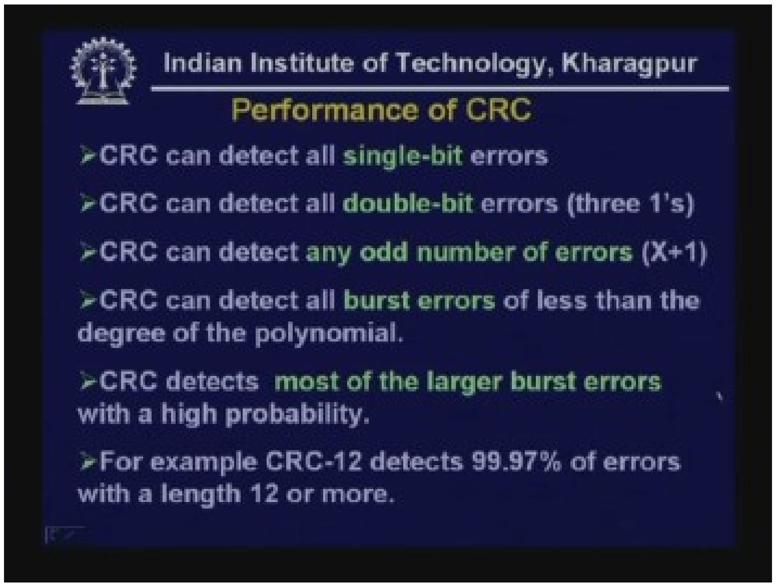
## Example

Characteristic polynomial: $x^3 + x + 1$

| | $C_2$ | $C_1$ | $C_0$ | $C_2$ (+) $C_0$ | $C_2$ (+) input | input |
|---|---|---|---|---|---|---|
| Initial | 0 | 0 | 0 | 0 | 1 | 1 |
| step 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| step 2 | 0 | 1 | 0 | 0 | 1 | 1 |
| step 3 | 1 | 0 | 1 | 0 | 1 | 0 |
| step 4 | 0 | 0 | 1 | 1 | 0 | 0 |
| step 5 | 0 | 1 | 0 | 0 | 0 | 0 |
| step 6 | 1 | 0 | 0 | 1 | 1 | 0 |
| step 7 | 0 | 1 | 1 | 1 | 0 | |

So after applying seven clocks we get the result in this $C_0$ $C_1$ and $C_2$ and indeed this is the CRC 0 1 1 so data to be sent is 1 0 1 0 0 1 1 that means these three 0s are replaced by 0 0 1 and we get 1 0 1 0 0 0 1 this is the data to be sent. Now at the receiving end instead of applying 1 0 1 0 0 0 0 the receiver will apply this 1 0 1 0 0 1 1 to this Linear Feedback Shift Register and again they will apply seven clocks and at the end you will see that result will be 0 1 1 if no error has occurred and if any error occurs then of course the remainder will not be 0 0 0 but it will be something else. So this is how the polynomial division is performed by using Linear Feedback Shift Register. Let us see the performance of CRC.

(Refer Slide Time: 40:18)



Indian Institute of Technology, Kharagpur

## Performance of CRC

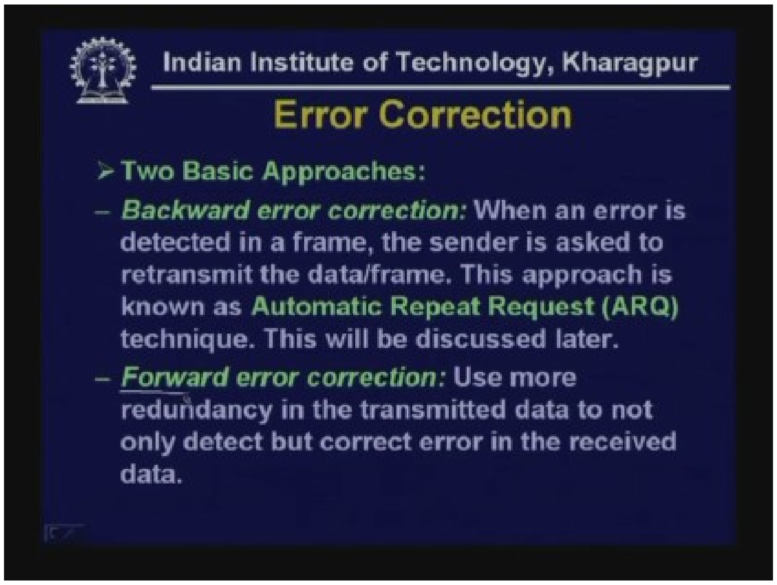
- CRC can detect all single-bit errors
- CRC can detect all double-bit errors (three 1's)
- CRC can detect any odd number of errors (X+1)
- CRC can detect all burst errors of less than the degree of the polynomial.
- CRC detects most of the larger burst errors with a high probability.
- For example CRC-12 detects 99.97% of errors with a length 12 or more.

CRC can detect cyclic redundancy can detect all single-bit errors. This is similar to the simple parity check. It can detect all double-bit errors provided there are three ones in the characteristic polynomial and usually in a number of characteristic polynomials we have seen that total number of ones is indeed more than 3. So the CRC can detect all double-bit errors. CRC can detect any odd number of errors provided it is divisible by x + 1 and CRC is chosen such that it is divisible by 1 so it can detect single-bit error, double-bit error and any odd number of errors. Not only that it can detect all burst errors less than the degree of the polynomial. So degree of the polynomial is n so it will detect all the burst errors less than the degree of the polynomial.

We have seen that the CRCs are usually 16-bit or 32-bit. So in this case up to 15-bit burst errors can be detected and moreover CRC can detect most of the larger burst errors. That means if the polynomial is of 16-bit then if the degree of the polynomial is n then the number of burst errors that means more than 15-bit errors can also be detected. In other words most of the larger burst errors that means more than 15 bits can be also detected with high probability. However, all the larger burst errors cannot be detected.

For example for CRC-12 that means whenever degree of the polynomial is 12 it will detect 99.97% of all errors with length 12 or more so we find that the CRC gives you a very powerful error detection scheme and that is the reason why it is so widely used. Now let us shift gear and come to the last topic that is error correction.

(Refer Slide Time: 43:43)


Indian Institute of Technology, Kharagpur

## Error Correction

- Two Basic Approaches:
  - *Backward error correction:* When an error is detected in a frame, the sender is asked to retransmit the data/frame. This approach is known as Automatic Repeat Request (ARQ) technique. This will be discussed later.
  - *Forward error correction:* Use more redundancy in the transmitted data to not only detect but correct error in the received data.

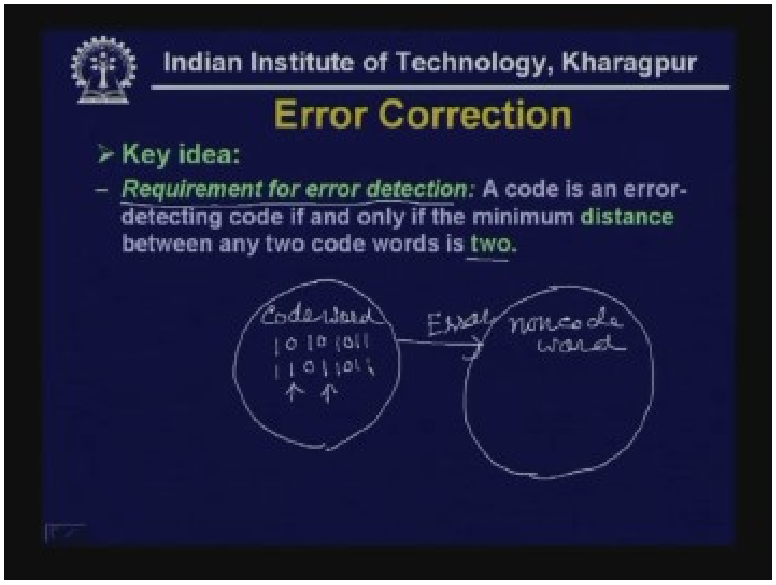
In error correction two basic approaches are used. The first approach is based on backward error correction. This backward error correction is based on this; when an error is detected in a frame the sender is asked to retransmit the data or frame. That means the receiver will detect the error by one of the schemes as we have discussed then it will send information to the ender and ask the sender to retransmit the frame once again. So this approach is known as Automatic Repeat Request and this scheme is known as backward error correction because we are going back to transmission once again. this particular ARQ scheme will be discussed in our next lecture. in this lecture we will be focusing on forward error correction.

In this forward error correction what is being done is by using more redundancy the transmitted data is received and not only error detection is performed but error correction is also performed on the received data so it is based on the received data it is not based on the transmission and after the data is received using the additional bits or redundant bits error correction is performed. Let us see what the basic ideas are. The basic idea is to have more redundancy particularly using this property.

For example, whenever you are performing error detection the requirement is an error detecting code if and only if the minimum distance between any two code words is two. So what you can do is….. all the codes can be divided into two sets codeword and non codeword then whenever you choose two code words say you are choosing 1 0 1 0 1 0 1 1 another codeword must have a distance of two what do you really mean by that they must defer in two bit position say it can be 1 1 0 1 1 0 1 1 so here we find they differ in these two positions. So, whenever they differ in two bit positions and if error occurs in any one of these it will fall here. Whenever error occurs it will not be a part of codeword but it will be part of non codeword that's how error is detected.

However, if the distance is only one then again it will be part of this so any single-bit error will lead will generate a non codeword that is the basis for error detection. That means the codes are selected in such a way that they differ in two number bits.

(Refer Slide Time: 46:10)

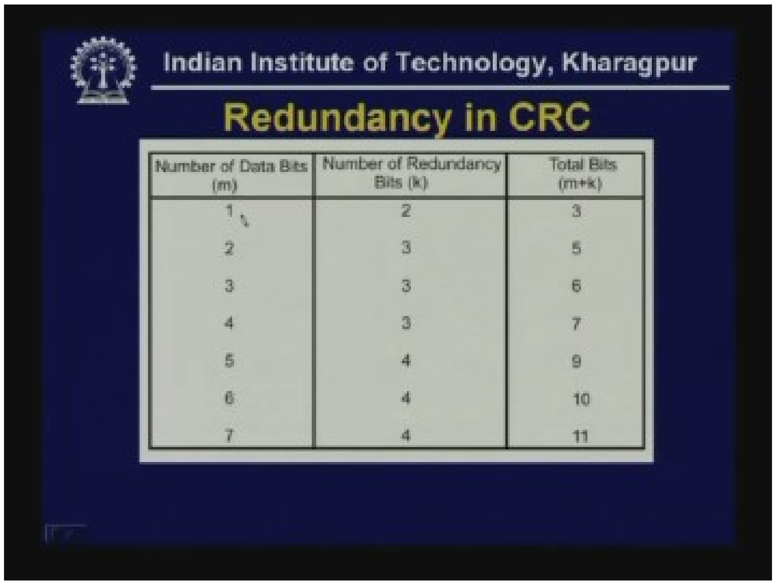

Indian Institute of Technology, Kharagpur

## Error Correction

- Key idea:
  - *Requirement for error detection:* A code is an error-detecting code if and only if the minimum distance between any two code words is two.

If you check the codes that are generated by parity bit you will find that this property is satisfied. That means any two code words will differ in two bits. But for error correction requirement is more. The minimum distance between any two code words must be more than two and the number of additional bits should be such that it can point the position of the bit in error. So here the requirement is we are not only interested in detecting error but identifying where the error has occurred. So the additional bits are to be selected such that it is possible to point the bit in error. So requirement is if k additional bits are used then it can point to 2 to the power k positions and the requirement is that 2 to the power k has to be more than m + k + 1 so this condition has to be satisfied only then the error correction can be done. This table shows the requirement. For example here the number of data bits is varying from 1 to seven and the number of redundancy bit required to satisfy the conditions is given here.

(Refer Slide Time: 47:31)



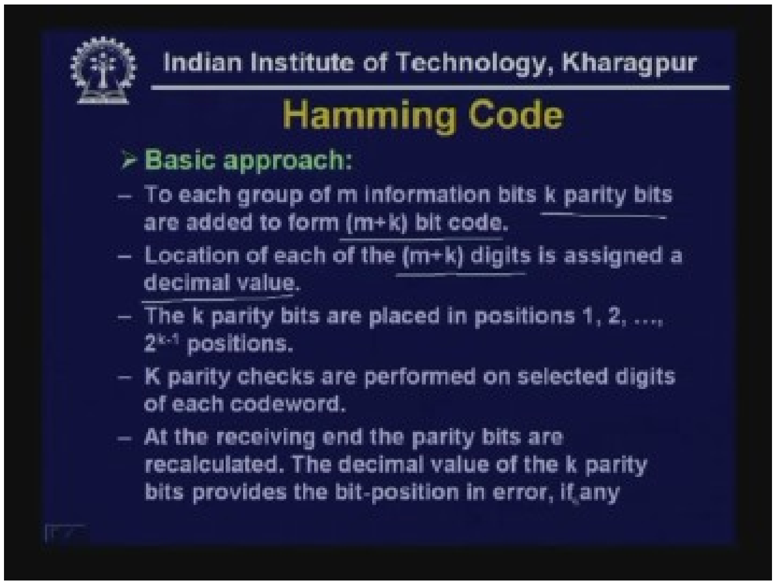
Indian Institute of Technology, Kharagpur

## Redundancy in CRC

| Number of Data Bits (m) | Number of Redundancy Bits (k) | Total Bits (m+k) |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 3 | 5 |
| 3 | 3 | 6 |
| 4 | 3 | 7 |
| 5 | 4 | 9 |
| 6 | 4 | 10 |
| 7 | 4 | 11 |

For one data bit the total number of bits has to be 3 including the redundancy bits but for seven data bits the redundancy required is four having total number of bits as 11. So, if the number of data bits is 16 it can be shown that we will require 5 bits that means m plus k in this case is 21. To point to one of the 21 bits you will require 5 bits so total number of bits will be 21 whenever your number of data bits is 16. Let us see how exactly it is done.

(Refer Slide Time: 49:02)



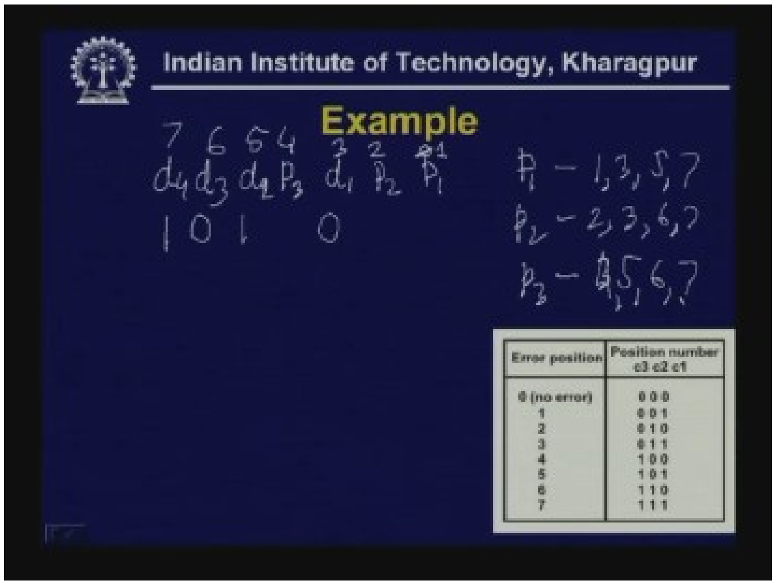
Indian Institute of Technology, Kharagpur

## Hamming Code

- Basic approach:
  - 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, …, $2^{k-1}$ positions.
  - 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 provides the bit-position in error, if any

So, to each group of n information bits k parity bits are added to form m + k bit code and the location of each of the m plus k digits is assigned a decimal value. Then the k parity bits are placed in position 1, 2, 2 to the power k positions then the k parity checks are performed on selected digits of each codeword and at the receiving end the parity bits are recalculated. The decimal value of the k parity bits provides the bit position in error if any. Let us see how exactly it is being done with the help of an example.

(Refer Slide Time: 51:01)



Indian Institute of Technology, Kharagpur

## Example

| 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|
| $d_4$ | $d_3$ | $d_2$ | $p_3$ | $d_1$ | $p_2$ | $p_1$ |
| 1 | 0 | 1 |  | 0 |  |  |

$p_1$ – 1, 3, 5, 7

$p_2$ – 2, 3, 6, 7

$p_3$ – 4, 5, 6, 7

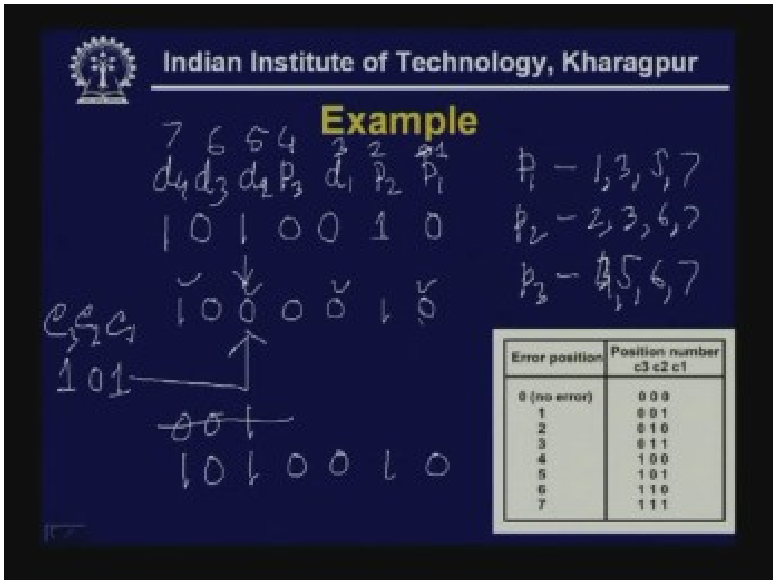
| Error position | Position number c3 c2 c1 |
|---|---|
| 0 (no error) | 0 0 0 |
| 1 | 0 0 1 |
| 2 | 0 1 0 |
| 3 | 0 1 1 |
| 4 | 1 0 0 |
| 5 | 1 0 1 |
| 6 | 1 1 0 |
| 7 | 1 1 1 |

Let's assume that your data is of four bits, let's assume your data is 1 0 1 0. So what is done is the parity bits $p_1$ $p_2$ these are added in one two and four positions and here you ad the data bits $d_1$ $d_2$ $d_3$ and $d_4$ then as you can see with the help of this three parity bits we have to identify the positions. Let's assume these to be (Refer Slide Time: 50:00) 1 2 3 4 5 6 7 per each position so seven positions have to be pointed with the help of parity bits by a code and here the codes are shown. So here we find 0 0 0 whenever there is no error and when error is found then it will point to this position 0 0 1. Whenever it is 0 1 0 it will point to this position by checking parity bit on this. So these parity bits p1 has to be calculated wherever it is 1 that means using the bit positions 1 3 5 and 7 and $p_2$ has to be calculated by 2 3 6 and 7 and p3 has to be calculated using 4 5 6 and 7. Let's see how it is done in this example.

So this p one has to be calculated using 1 3 5 7 so these three are given so already these two are one so two even number of ones are there so this has to be 0 (Refer Slide Time: 51: 24) and these 2 3 6 7 so here the number of ones is odd so you have to add 1 to make it even and here 5 6 7 is already even so it is 4 5 6 7 so here it will be 0. So this is the code words generated by calculating the bits and then it is sent. Suppose error has occurred here this has become 0 so this is the data that is being received at the receiving end.

Now you calculate the parity based on the same manner. So here we get $C_1$ $C_2$ and $C_3$. So by using 1 3 5 and 7 we find it is odd. So this has to be 1 by using this bit, this bit, this bit and this bit so it is even so this has to be 1 to make it 1. Then 2 3 6 7 so we find it is already even. So this is 0 and here it is 4 5 6 7 it has become odd so to make it even it has to be 1 again. So we find it is 5 that means it is pointing to a two bit number 5 so the corrected data has to be 1 0 1 0 0 1 0. With this simple example we have illustrated how the code is generated by using hamming code. This is known as the hamming code scheme and how the correction is done at the receiving end. This is how the error correction is performed by using hamming code.

(Refer Slide Time: 53:17)

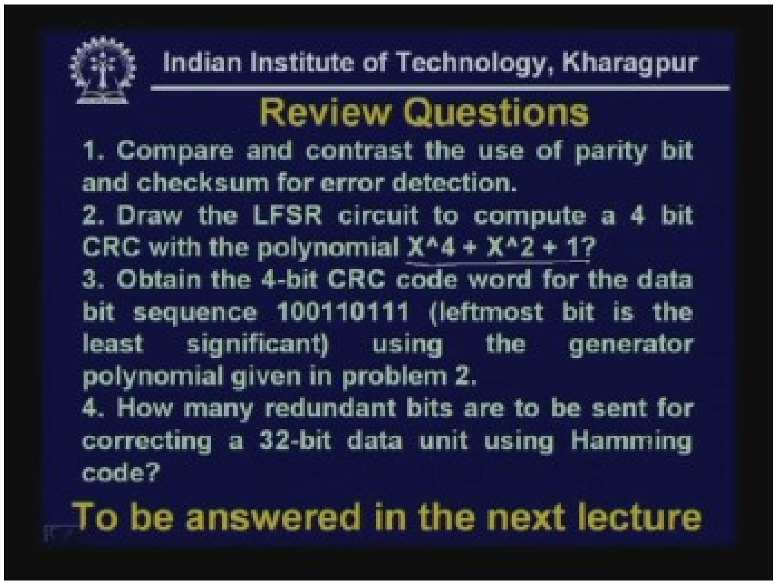

Indian Institute of Technology, Kharagpur

## Example

| Error position | Position number c3 c2 c1 |
|---|---|
| 0 (no error) | 0 0 0 |
| 1 | 0 0 1 |
| 2 | 0 1 0 |
| 3 | 0 1 1 |
| 4 | 1 0 0 |
| 5 | 1 0 1 |
| 6 | 1 1 0 |
| 7 | 1 1 1 |

Therefore in this lecture we have discussed how error correction is done by using three schemes; parity check, checksum and CRC and how error correction is performed. Obviously only single-bit error correction can be performed by using hamming code. However, when the number of bits in error is more you have to add more redundancy. Now it is time to consider the review questions.

(Refer Slide Time: 54:44)



Indian Institute of Technology, Kharagpur

## Review Questions

1. Compare and contrast the use of parity bit and checksum for error detection.
2. Draw the LFSR circuit to compute a 4 bit CRC with the polynomial X^4 + X^2 + 1?
3. Obtain the 4-bit CRC code word for the data bit sequence 100110111 (leftmost bit is the least significant) using the generator polynomial given in problem 2.
4. How many redundant bits are to be sent for correcting a 32-bit data unit using Hamming code?

To be answered in the next lecture

1) Compare and contrast the use of parity bit and checksum for error detection.

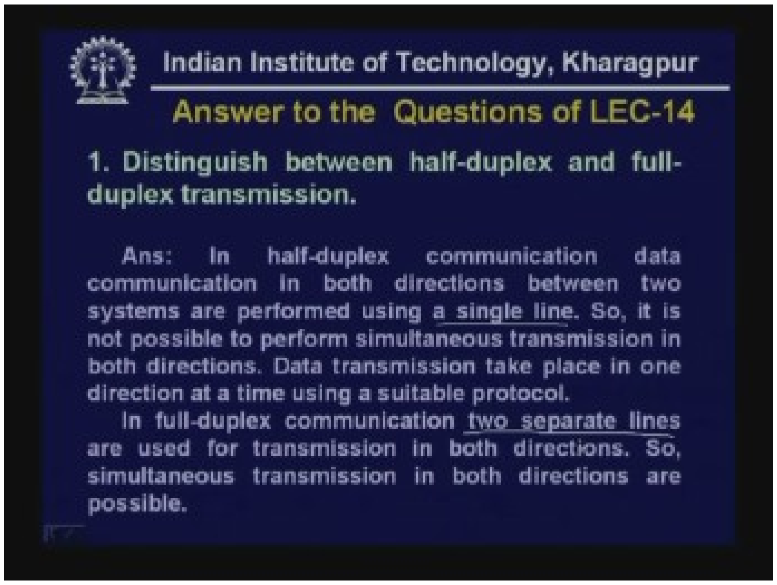
2) Draw the LFSR circuit to compute a four bit CRC with the polynomial x to the power 4 + x to the power 2 + 1.

3) Obtain the four bit CRC code word for the data bit sequence 1 0 0 1 1 0 1 1 1 and the left most bit is least significant using the generator polynomial given in problem 2. That means this is the generator polynomial.

4) How many redundant bits are to be sent for correcting a 32 bit data unit using hamming code?

These questions will be answered in the next lecture.

(Refer Slide Time: 55:48)


Indian Institute of Technology, Kharagpur

Answer to the Questions of LEC-14

1. Distinguish between half-duplex and full-duplex transmission.

Ans: In half-duplex communication data communication in both directions between two systems are performed using a single line. So, it is not possible to perform simultaneous transmission in both directions. Data transmission take place in one direction at a time using a suitable protocol.

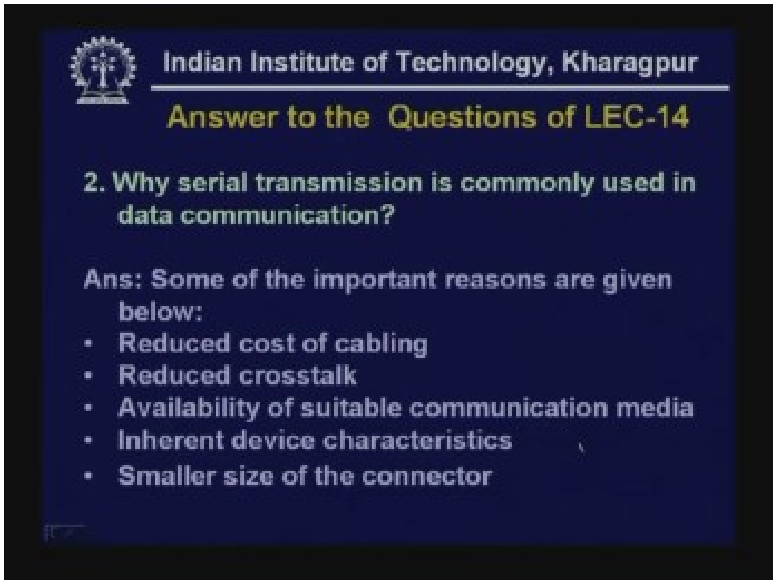
In full-duplex communication two separate lines are used for transmission in both directions. So, simultaneous transmission in both directions are possible.

1) Distinguish between half-duplex and full-duplex transmission.

In half-duplex communication data communication in both directions between the systems are performed using a single line. So it is not possible to perform simultaneous transmission in both directions. It has to be done by using a protocol as we saw in the last lecture that how a policeman communicates with his head office by using a suitable protocol and the other end will say over and then the talking will start.

So, data transmission takes place in one direction at a time using suitable protocol. However, in full-duplex communication two separate lines are used for transmission in both directions so simultaneous transmission in both directions are possible.

(Refer Slide Time: 56:01)



Indian Institute of Technology, Kharagpur

Answer to the Questions of LEC-14

2. Why serial transmission is commonly used in data communication?

Ans: Some of the important reasons are given below:

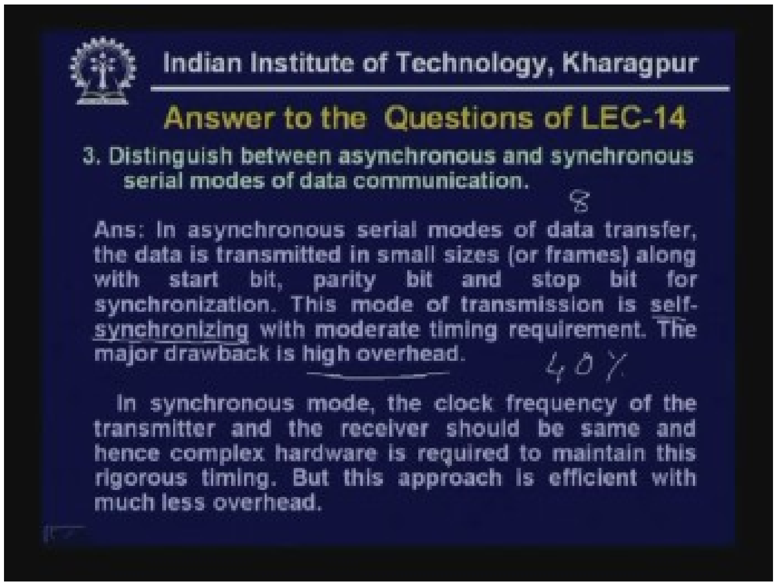
- Reduced cost of cabling
- Reduced crosstalk
- Availability of suitable communication media
- Inherent device characteristics
- Smaller size of the connector

2) Why serial transmission is commonly used in data communication?

Some of the important reasons are given below:

- Reduced cost of cabling
- Reduced cross talk
- Availability of suitable communication media
- Inherent device characteristics and
- Smaller size of the connector

(Refer Slide Time: 56:59)



Indian Institute of Technology, Kharagpur

Answer to the Questions of LEC-14

3. Distinguish between asynchronous and synchronous serial modes of data communication.

8

Ans: In asynchronous serial modes of data transfer, the data is transmitted in small sizes (or frames) along with start bit, parity bit and stop bit for synchronization. This mode of transmission is self-synchronizing with moderate timing requirement. The major drawback is high overhead. 40%

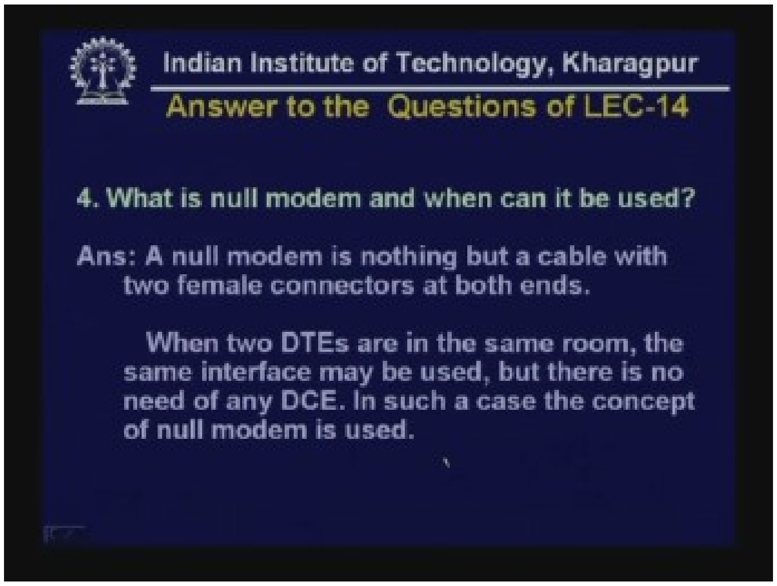
In synchronous mode, the clock frequency of the transmitter and the receiver should be same and hence complex hardware is required to maintain this rigorous timing. But this approach is efficient with much less overhead.

3) Distinguish between asynchronous and synchronous serial modes of data communication.

In asynchronous serial modes of data transfer the data is transmitted in small sizes or frames along with start bit, parity bit, stop bits and as we have seen the maximum number is 8. And this mode of transmission is self synchronizing with moderate timing requirements. The major drawback is high overhead; it can be 40% overhead as we have seen.

In synchronous mode the clock frequency of the transmitter and the receiver should be same and hence complex hardware is required to maintain this rigorous timing. But this approach is efficient with much less overhead.

(Refer Slide Time: 57:13)


Indian Institute of Technology, Kharagpur

Answer to the Questions of LEC-14

4. What is null modem and when can it be used?

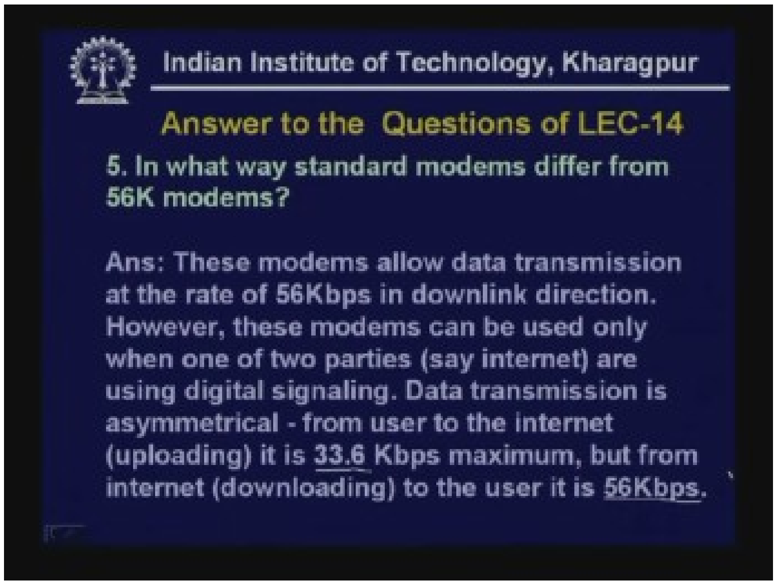
Ans: A null modem is nothing but a cable with two female connectors at both ends.

When two DTEs are in the same room, the same interface may be used, but there is no need of any DCE. In such a case the concept of null modem is used.

4) What is null modem and when can it be used?

A null modem is nothing but a cable with two female connectors at both ends so no active component is present. Here it is just a cable with two connectors at both ends so these are the two female connectors at both ends (Refer Slide Time: 57:20). This can be used when two DTEs are in the same room the same interface may be used but there is no need for any DCE in such a case the concept of null modem is used.

(Refer Slide Time: 58:16)



Indian Institute of Technology, Kharagpur

Answer to the Questions of LEC-14

5. In what way standard modems differ from 56K modems?

Ans: These modems allow data transmission at the rate of 56Kbps in downlink direction. However, these modems can be used only when one of two parties (say internet) are using digital signaling. Data transmission is asymmetrical - from user to the internet (uploading) it is 33.6 Kbps maximum, but from internet (downloading) to the user it is 56Kbps.

5) In what way standard modems differ from 56K modems?

These modems allow data transmission at the rate of 56 Kbps in downlink direction. However, these modems can only be used when one of the two parties such as internet is using digital signaling. Data transmission is asymmetrical. For example, from user to the internet is that is in the uploading direction it is maximum of 33.6 Kbps but from the internet to the user where digital signal can be used it is 56 Kbps.

With this we come to the end of today’s lecture, thank you.