Switching Circuits and Logic Design

Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

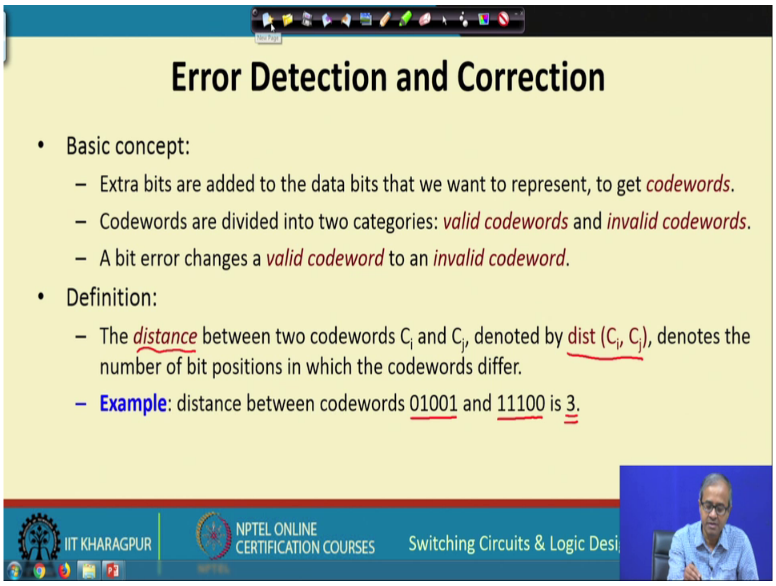
Lecture - 06
Error Detection and Correction

In this lecture we shall be talking about Error Detection and Correction when we store or communicate some digital data. The idea is very simple when you have some data, we store it somewhere maybe in a hard disk or maybe other place with time some bits might get changed or corrupt. Similarly, when you are sending some data over a network, over a communication channel due to some noise in the communication channel some of the bits again might get corrupt.

So, it is a very important requirement to provide some error detection and error correction mechanism during this kind of storage and communication process. Now, there are many methods available, many sophisticated algorithms and techniques available. Now, here we shall be just giving you the basic concept and one simple method using which you can carry out error detection and correction ok.

This is the objective of the present lecture.

(Refer Slide Time: 01:19)



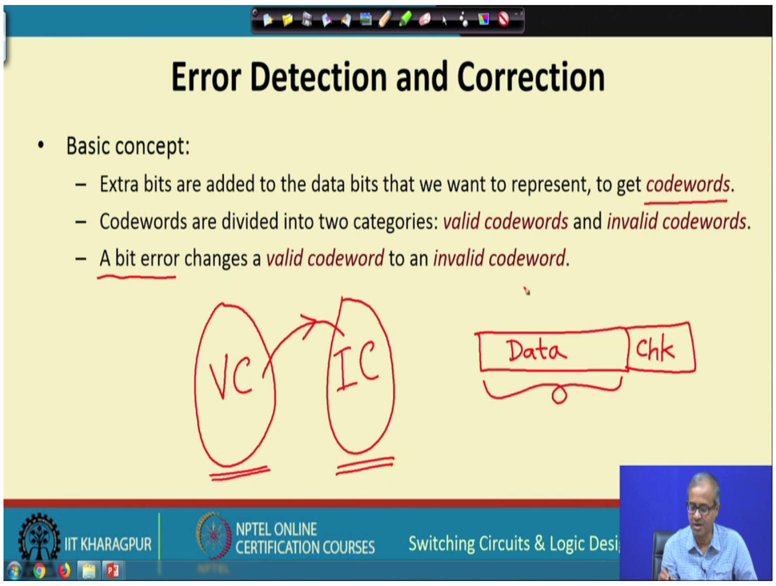
Error Detection and Correction

- Basic concept:
  - Extra bits are added to the data bits that we want to represent, to get *codewords*.
  - Codewords are divided into two categories: *valid codewords* and *invalid codewords*.
  - A bit error changes a *valid codeword* to an *invalid codeword*.
- Definition:
  - The *distance* between two codewords C_i and C_j, denoted by dist (C_i, C_j), denotes the number of bit positions in which the codewords differ.
  - **Example**: distance between codewords 01001 and 11100 is 3.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

So, the basic concept of error detection and correction is suppose I have a data or a number which I want to store or communicate, this is my basic data.

(Refer Slide Time: 01:31)



Error Detection and Correction

- Basic concept:
  - Extra bits are added to the data bits that we want to represent, to get *codewords*.
  - Codewords are divided into two categories: *valid codewords* and *invalid codewords*.
  - A bit error changes a *valid codeword* to an *invalid codeword*.

VC
IC
Data
Chk
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

So, one thing to understand is that whenever we are talking about error detection and correction first thing is that we must add some extra bits to my data because if I these are sometimes called check bits it is called check. So, if I do not add some additional bits it will not be possible for me to tell just by looking at my data that whether the data is correct or some error has taken place.

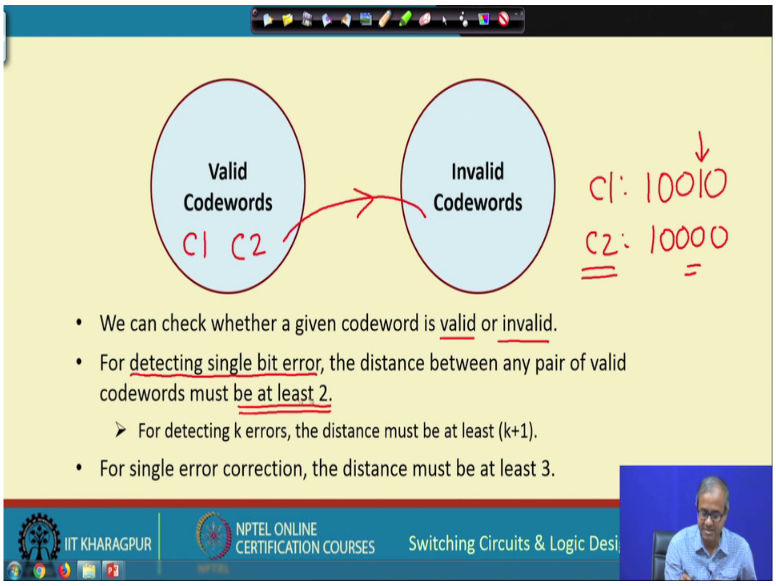
So, it is mandatory to provide some kind of additional information in the form of check bits and the data along with the check bits will form something called code words ok. So, we have the concept of code words, extra bits are added to the data and we get code words. Now, now the concept behind error detection is when you talk about code words we say there can be valid codes, there can be invalid codes. So, code words can be valid or invalid.

Normally, whenever we are storing or communicating some data they correspond to valid code words. Now, if I design my codes in such a way let us say I am talking about a single bit error 1 bit is changing. So, if I design my code in such a way that every single bit error will convert a valid code word into an invalid code word then my task is done. I should have a way to test whether a code word is invalid or not. So, if I see it is invalid which means that some error has taken place. So, it was a valid code word and some bit error has converted that valid code word into an invalid code word; this is a basic concept fine. So, let us talk about one definition here distance between code words.

Let us say we have two code words C i and C j the distance between them sometimes it is denoted as dist distance C i and C j. It basically means how many bit positions are different between C i and C j. Let us take a specific example, consider a code word 01001 and 11100. So, how many positions are different first position is 0 and 1 it is different, second position is same, third position is different, fourth position is same, fifth position is different. So, number of differing position is 3.

We say that the distance between the code word is 3. So, the reason we are talking about distance is that the distinction between valid and valid and invalid code words are often based on the concept of this distance.

(Refer Slide Time: 05:04)



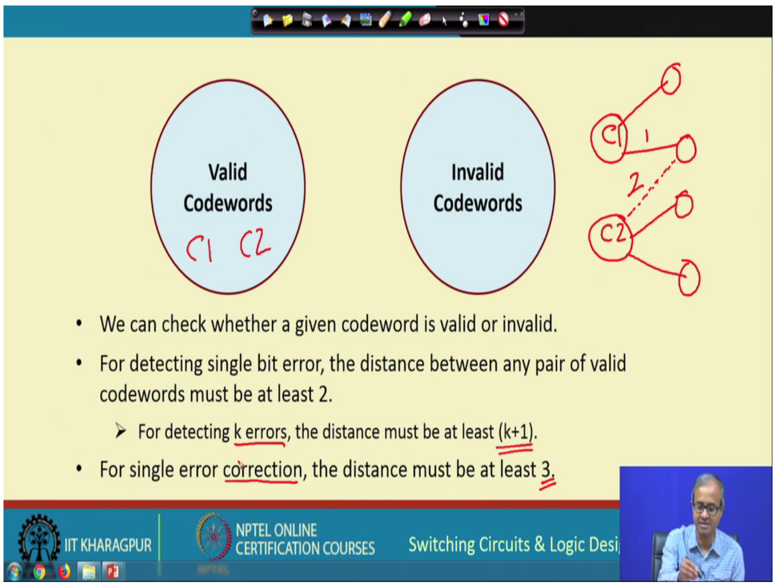
- We can check whether a given codeword is valid or invalid.
- For detecting single bit error, the distance between any pair of valid codewords must be at least 2.
  - For detecting k errors, the distance must be at least (k+1).
- For single error correction, the distance must be at least 3.

So, this is the kind of scenario I talked about I have valid code words, I have invalid code words and there should be a mechanism I said to check whether a code word is valid or invalid. Let us say our task is to detect single bit error, what I claim is that if this is my requirement then the distance between any pair of valid code words must be at least 2. Why? Consider 2 valid code word let us say c 1 and c 2. I am saying that any single bit error we will transform it into an invalid code right.

Single bit error means what, I am increasing the distance by 1 with respect to c 1 or c 2. Now, if distance of c 1 and c 2 was less than 2; suppose c 1 and c 2 distances 1. Let us say let us take an example c 1 is 10010 and c 2 is 10000 only 1 bit position is changing, let us say the distance is 1. So, it can happen that the valid code word c 1 there is a single bit error on this position and it has become 10000 which is also a valid code word. So, valid remains valid ok.

So, distance of 1 will not satisfy my property, distance must be at least 2, so that any single bit error will transform c 1 into some other code which is not in this set, which is in the other set.

(Refer Slide Time: 07:23)



Valid Codewords
C1 C2
Invalid Codewords
C1
1
2
C2
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

- We can check whether a given codeword is valid or invalid.
- For detecting single bit error, the distance between any pair of valid codewords must be at least 2.
  - For detecting k errors, the distance must be at least (k+1).
- For single error correction, the distance must be at least 3.

So, across all valid code words between any pair distance must be at least 2. This is the mandatory requirement and generalizing this concept if we are talking about detecting k errors.

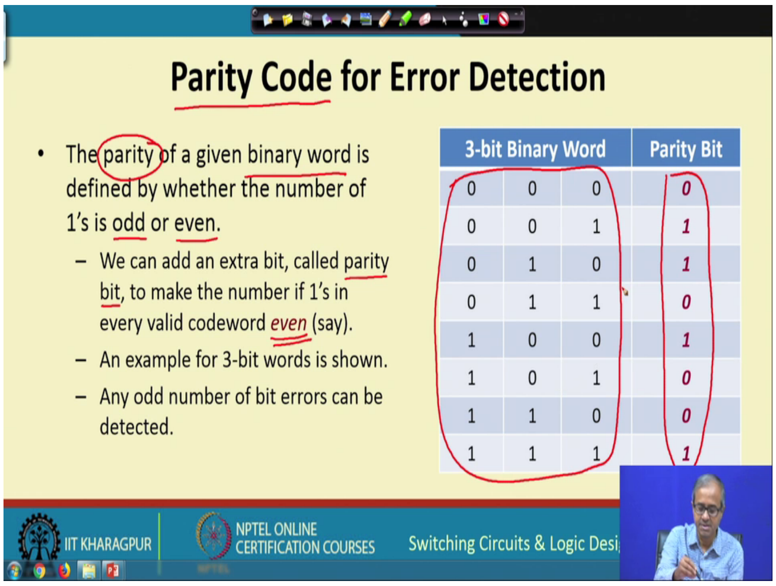
So, k numbers of bit errors can be there and k or less errors should not convert one valid code word into another valid code word. So, the distance here must be at least k plus 1. These are some basic theory behind the idea. Now, if I want to talk about not detection also correction, correction means suppose an error has taken place while I detect the error not only I detect the error I also find out the location of the error. So, that I can correct it, this is what is meant by error correction.

Now, if I want to do error correction what is the mean, what is the concept you consider c 1 and c 2. Suppose, I have this code c 1 I have this code c 2 there can be single bit errors. Due to single bit error some bit c 1 can go to some other states c c 1 can go to some other state, c 2 can go somewhere, c 2 can go somewhere. Now, what I am saying is that if c 1 and c 2 are going to the same place. Let us say here, then I cannot say that this is because of an error in c 1 or because of an error in c 2.

What I am saying is that for single error correction, you must be uniquely able to tell that which is the code from where it has there was a error and you had come to that, which means so, every such say due to a single error you must arrive here. But if you want to go from c 2 to here at least 2 errors must be there.

So, the distance between c 1 and c 2 must be at least 3. If we have a requirement like this satisfied then only you can say that you can also provide error correction in addition ok. This is the basic concept.

(Refer Slide Time: 09:34)



Parity Code for Error Detection

- The parity of a given binary word is defined by whether the number of 1's is odd or even.
  - We can add an extra bit, called parity bit, to make the number if 1's in every valid codeword *even* (say).
  - An example for 3-bit words is shown.
  - Any odd number of bit errors can be detected.

3-bit Binary Word			Parity Bit
0	0	0	*0*
0	0	1	*1*
0	1	0	*1*
0	1	1	*0*
1	0	0	*1*
1	0	1	*0*
1	1	0	*0*
1	1	1	*1*

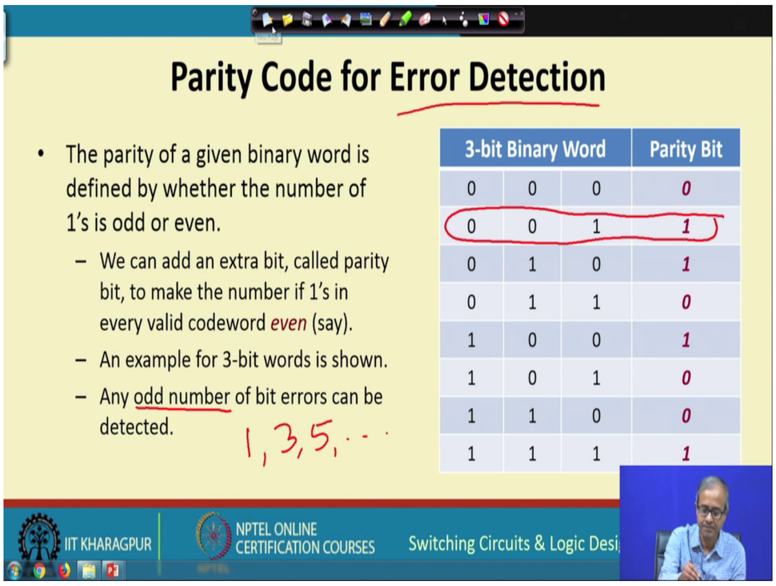
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

So, let us first talk about a very simple scheme for detecting errors. This is a very popular technique using something called parity codes. Parity is very simple. Parity is defined with respect to a given binary word, you check whether the number of 1's in the word is odd or whether it is even. Well, there can different conventions, but as a matter of convention you can say that you add a parity bit to your data so, that the total number of 1's in every valid code word becomes even.

Let us take an example: you consider 3-bit binary words like I show here let us say these are my data bits. It can be 0 0 0 up to 1 1 1, I add 1 additional parity bit to every word in such a way that the number of 1's in every row you can check becomes even; 0 0 1 I add a 1. So, that number of 1's become 2. 0 1 0 again 0 1 1 already it is 2 so, I make it 0 even 1 1 1 3 and 1 4 so, like this.

So, parity bit you can define like this it will be 0, if the number of 1's in the binary code word is even, it is 1 if the number of 1's is odd you see it is 1 if the number of 1's is odd, if the number of 1's is odd. So, this is how we add binary bits these parity bits.

(Refer Slide Time: 11:34)



Parity Code for Error Detection

- The parity of a given binary word is defined by whether the number of 1's is odd or even.
  - We can add an extra bit, called parity bit, to make the number if 1's in every valid codeword *even* (say).
  - An example for 3-bit words is shown.
  - Any odd number of bit errors can be detected.

1, 3, 5, . . .

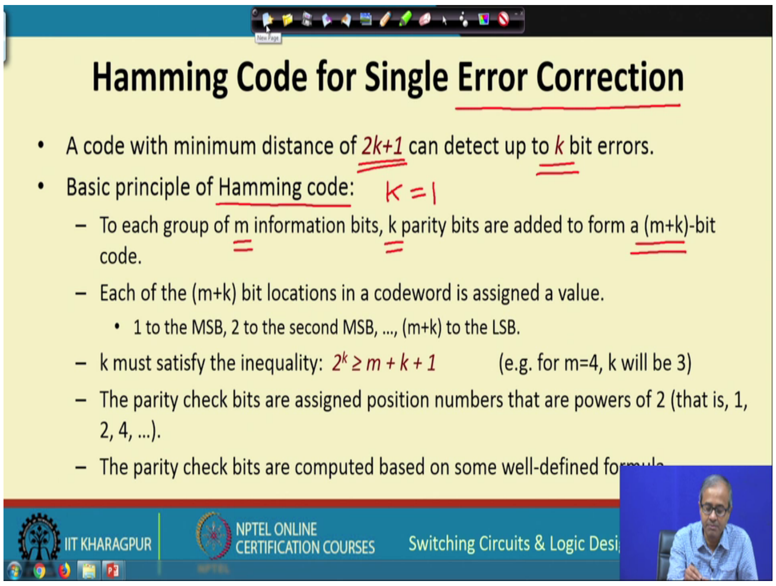
3-bit Binary Word			Parity Bit
0	0	0	*0*
0	0	1	*1*
0	1	0	*1*
0	1	1	*0*
1	0	0	*1*
1	0	1	*0*
1	1	0	*0*
1	1	1	*1*

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

Now, once you add the parity bits what is the advantage you gain, you see that any odd number of errors will be detected. Why? Because, now my code word will consist of not only my data, but also the parity right the whole thing and we know that the number of 1's is even. Now, any odd number of errors which means single error, 3 errors for larger number 5, 7, 9 we will convert it into a code where the number of 1's will become odd. So, if I have a checking mechanism where I check whether the number of 1's is remaining even or not.

So, any odd number of errors will make the number of 1's odd so, it can be detected. So, I can use it for error detection, very simple mechanism. Not only single bit errors, but rather any odd number of bit errors single error, 3 errors, 5 etcetera right.

(Refer Slide Time: 12:36)



Hamming Code for Single Error Correction

- A code with minimum distance of $2k+1$ can detect up to k bit errors.
- Basic principle of Hamming code: $K=1$
  - To each group of m information bits, k parity bits are added to form a (m+k)-bit code.
  - Each of the (m+k) bit locations in a codeword is assigned a value.
    - 1 to the MSB, 2 to the second MSB, …, (m+k) to the LSB.
  - k must satisfy the inequality: $2^k \geq m+k+1$ (e.g. for m=4, k will be 3)
  - The parity check bits are assigned position numbers that are powers of 2 (that is, 1, 2, 4, …).
  - The parity check bits are computed based on some well-defined fo

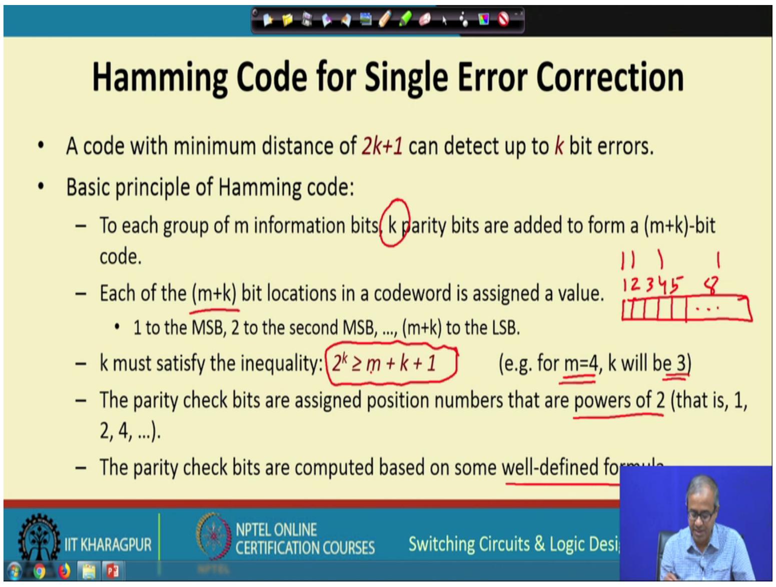
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

Now, let us come to a more complex mechanism where I also want to do error correction. The rule for error correction I mention for single error that for 1-bit error correction I need a distance of 3 so, here I am generalizing that statement. If I want to correct up to k bit errors others except more complex statement, the minimum distance between every pair of valid code words should be at least 2 k plus 1.

Now, Hamming code is one method where we can correct single bit errors k equal to 1 right. So, let us see how Hamming code works. The Hamming code that we shall be explaining here that is for k equal to 1.

The idea is suppose I have my data bits or information bits there are m such bits, we add a set of parity bits not 1, but k number of parity bits ok, do not confuse this k with this k we add k number of parity bits so, that we create a m plus k bit word.

(Refer Slide Time: 14:23)



Hamming Code for Single Error Correction

- A code with minimum distance of $2k+1$ can detect up to k bit errors.
- Basic principle of Hamming code:
  - To each group of m information bits, k parity bits are added to form a (m+k)-bit code.
  - Each of the (m+k) bit locations in a codeword is assigned a value.
    - 1 to the MSB, 2 to the second MSB, ..., (m+k) to the LSB.
  - k must satisfy the inequality: $\boxed{2^k \geq m+k+1}$ (e.g. for m=4, k will be 3)
  - The parity check bits are assigned position numbers that are powers of 2 (that is, 1, 2, 4, ...).
  - The parity check bits are computed based on some well-defined fo

1 1 1 1
1 2 3 4 5 8
. . .
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

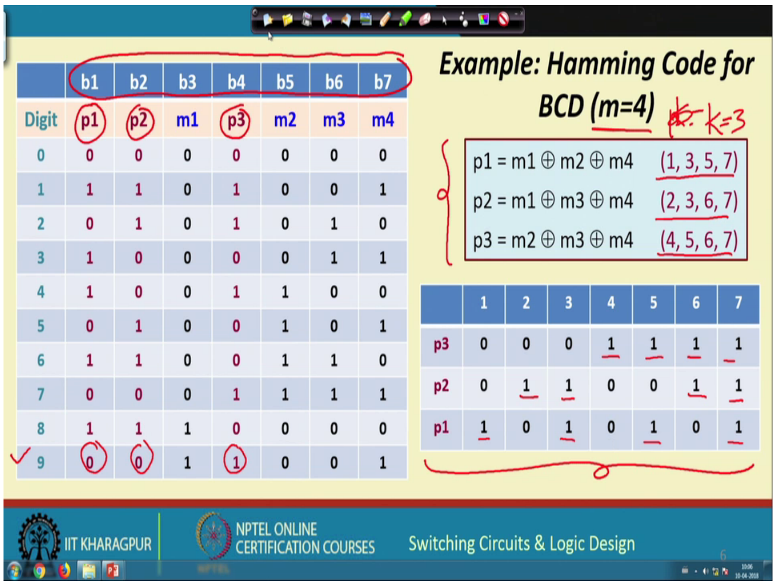
Now, these m plus k bit words, I shall take an example will have a bit location like what I mean is that; this each of this m plus k bit location they will be having value suppose I have this is my code word these are my bits. So, I assign numbers 1 2 3 4 from left to right like this.

So, each of this bit positions I am assigning a number and how many parity bits are to be added, there is a rule we will have to follow this rule. The smallest value of k that satisfies this inequality, 2 to the power k should be greater than or equal to m plus k plus 1.

So, suppose my information bits are 4 in number m equal to 4. So, you see what smallest value of k will satisfy this. So, right hand side will be 4 plus k plus 1, if k is 2 left hand side is 4 2 square and right hand side is 4 plus 2 plus 1, 7 so, it does not satisfy. If k is 3 this is 8 4 plus 3 plus 1, 8 so, it satisfies. So, the smallest value of k is 3 here and in this bit assignment 1 2 3 4 5 that I had said so, I will be adding k number of parity bits, which of these will be the parity bit.

They will be the positions which are powers of 2 like position 1, position 2, position 4, a possible position 8 ok. So, the powers of 2 I will be inserting the parity bits in those positions only and there are some well defined formula using which we can calculate the parity bits I will illustrate.

(Refer Slide Time: 16:23)



Example: Hamming Code for BCD (m=4)

~~k~~ k=3

	b1	b2	b3	b4	b5	b6	b7
Digit	p1	p2	m1	p3	m2	m3	m4
0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1
2	0	1	0	1	0	1	0
3	1	0	0	0	0	1	1
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1
6	1	1	0	0	1	1	0
7	0	0	0	1	1	1	1
8	1	1	1	0	0	0	0
✓ 9	0	0	1	1	0	0	1

$p1 = m1 \oplus m2 \oplus m4$ (1, 3, 5, 7)

$p2 = m1 \oplus m3 \oplus m4$ (2, 3, 6, 7)

$p3 = m2 \oplus m3 \oplus m4$ (4, 5, 6, 7)

	1	2	3	4	5	6	7
p3	0	0	0	1	1	1	1
p2	0	1	1	0	0	1	1
p1	1	0	1	0	1	0	1

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Let us take this example for this m equal to 4 as I illustrate said.

So, for m equal to 4 we had said that we have to have k equal to 3 the value of k must be equal to 3 sorry, k is 3. So, we need 7 bits 4 plus 3 b 1 b 2 to 7 and I said we inserted the parity bits in those positions which are powers of 2; that means, bit position 1, position 2 and 4. These are powers of 2 and my 4 information bits are located in the remaining positions 3rd, 5th, 6th and 7th. So, if you look at the remaining I am only showing decimal digits you can also add the other numbers also.

So, only 4 I am showing only this 10 I am showing; you see 0 is if you look at the information bits m 1 m 2 m 3 0 0 0 0. 1 is 0 0 0 1. 2 is 0 0 1 0 and so on, 9 is 1 0 0 1. Now, we are inserting parity bits in a suitable way. How? The rules are as follows you look at this table.

So, I write down the 3 parity bits p 1 p 2 p 3 and other than the all 0 combination I write down all possible combinations 0 0 1 1 2 3 4 5 6 and 7. 0 1 0 is 2, 0 1 1 is 3, 1 0 0 is 4. So, other than the all 0 combination I write down all 7 combinations and this 7-bit positions I write on top 1 2 3 4 5 6 7. For p 1 you look at which positions are having 1 here, here, here and here. So, those are positions 1 3 5 and 7.

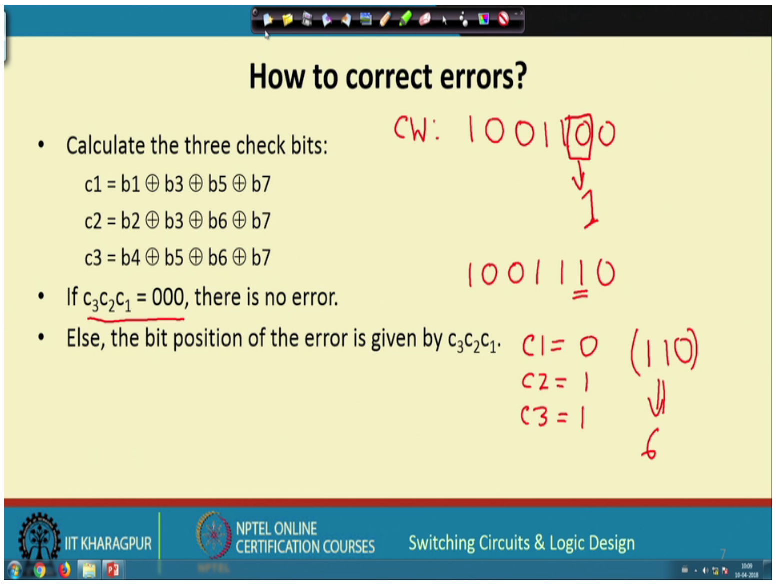
So, I note them down here 1 3 5 7; p 2 is having here 2 3 6 and 7, 2 3 6 and 7 and p 3 is having in position 4 5 6 7 here and now I can write down the parity generation formula like this; this symbol indicates modulo 2 sum. This we introduced earlier modulo 2 sum; see p 1 corresponds to 1 3 5 7 right 1 3 5 7, out of them 1 corresponds to the bit position 1 corresponds to p 1.

So, remaining are 3 5 7. So, in 3 what you have placed m 1, 5 is m 2, 7 is m 4. So, m 1 m 2 m 4 modulo 2 sum of these; talk about p 2 p 2 is 2 3 6 7 and p 2 is already in position 2 so, 3 6 7. 3 is m 1, 6 is m 3 and m 4 1 m 1 m 3 m 4. Similarly, 4 5 6 7, 4 is your p 3 5 6 7.

So, m 2 m 3 m 4; So, like this you can calculate the parity like let us take and take an example; let us consider that 9 the last one p 1 is m 1 m 2 m 4. So, recall this parity indicates whether the number is odd or even, if it is odd it is 1, if it is even it is 0. This modulo 2 sum works like that that is parity.

So, m 1 m 2 m 4 you see m 1 1 0 1 1 is even number of 1's so, it is 0. Then p 2 is m 1 m 3 m 4 1 0 1 again even so, p 2 is also 0; p 3 is m 2 m 3 m 4 0 0 1 it is odd that is why p 3 is 1. So, in this way you fill up all the parity bits right, this is how you do.

(Refer Slide Time: 21:04)



How to correct errors?

- Calculate the three check bits:
  - $c1 = b1 \oplus b3 \oplus b5 \oplus b7$
  - $c2 = b2 \oplus b3 \oplus b6 \oplus b7$
  - $c3 = b4 \oplus b5 \oplus b6 \oplus b7$
- If $\underline{c_3c_2c_1} = 000$, there is no error.
- Else, the bit position of the error is given by $c_3c_2c_1$.

CW: 1001100
1
1001110
C1 = 0
C2 = 1
C3 = 1
(110)
6
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Now, after you have done this now, you need to detect and correct errors. See correction of errors are fairly simple, see in the previous slide you see that p 1 p p 2 p 3 you have calculated using these numbers 1 3 5 7, 2 3 6 7, 4 5 6 7 you remember these numbers. You straight away find out three check bits c 1 c 2 c 3 just by taking modulo sum of those corresponding base 1 3 5 7, 2 3 6 7 and 4 5 6 7 you see these are these numbers 1 3

5 7, 2 3 6 7, 4 5 6 7 and this modulo sum means I am repeating if these bits number of 1’s are even then c 1 will be 0. If number of 1’s is odd it will be 1.

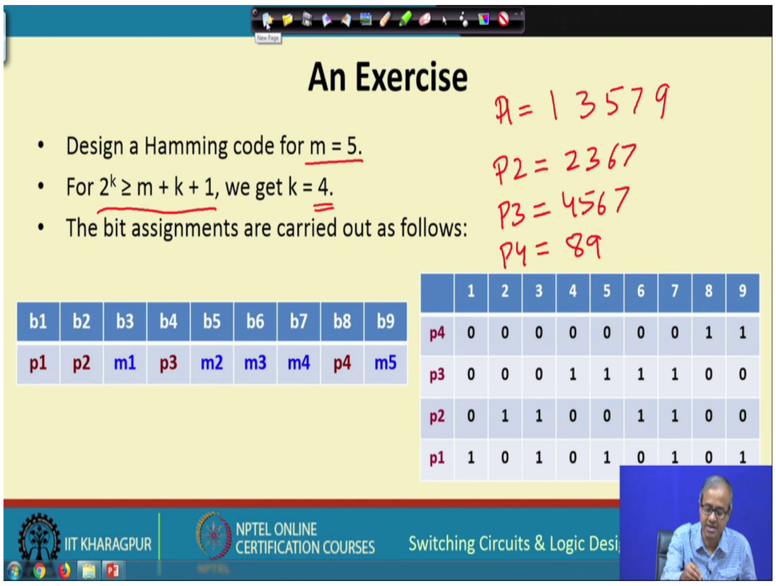
Now, the rule is simple after you have calculated these check bits, if you find that it is all 0 which means there is no error. But if it is non-zero which means there is an error and whatever you have got that will directly give you the position of the error. Let us take an example, suppose I have a code word let us take one of the valid code words I am taking from that table; let us say 1 0 0 1 1 0 0. This was the 7 bit number, let us say this 0 there is an error and this 0 has become 1. So, now my code word has become 1 0 0 1 1 1 0 this bit is in error.

Let us try to calculate using these formulas. So, what will be the value of c 1 1 3 5 7 this is 1 3 5 7 even number, it will be 0; c 2 2 3 6 7 2 3 6 7 odd number it will be 1; c 3 4 5 6 7 1 1 1 0 again odd number this will be 1. So, you see c 3 c 2 c 1 will respond to 1 1 0. So, what does 1 1 0 mean in binary it is 6.

So, bit number 6 is in error 1 2 3 4 5 6 it will tell you exactly which bit position is in error and since you know that position you can change this 1 back to 0, you can correct it. So, this you can verify with some other examples also you see that for any bit position error out of this 7 single bit error can be corrected in this method.

This is the advantage of Hamming code right, using just few parity bits you can detect and correct it.

(Refer Slide Time: 24:16)



An Exercise

- Design a Hamming code for m = 5.
- For $2^k \geq m + k + 1$, we get k = 4.
- The bit assignments are carried out as follows:

b1	b2	b3	b4	b5	b6	b7	b8	b9
p1	p2	m1	p3	m2	m3	m4	p4	m5

	1	2	3	4	5	6	7	8	9
p4	0	0	0	0	0	0	0	1	1
p3	0	0	0	1	1	1	1	0	0
p2	0	1	1	0	0	1	1	0	0
p1	1	0	1	0	1	0	1	0	1

P1 = 1 3579
P2 = 2367
P3 = 4567
P4 = 89

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Desi

Just let us work out another one, I am just giving you the schematic the rest I shall leave you as an exercise to verify. Suppose, we extend the concept we want to design a Hamming code for m equal to 5. So, again look at that inequality how many parity bits you require, smallest value of k you can check it will be 4 because 2 to the power 5 is 32 and 5 plus 4 plus 1 is 10. But if you take it 4 16 and 4 is 6 sorry, k is 4 2 to the power 4 is 16 5 plus 4 plus 1 is 10, 16 greater than equal to 10, but if you take k equal to 3 then the right hand side will become larger.

So, the first thing is the bit assignment. So, I said there will be 5 plus 4 9 bits b 1 b 2 to b 9. The 4 parity bits you assign the powers of 2 1 2 3 4 and 8, the rest you fill up with the information bits m 1 m 2 m 3 m 4 m 5. So, 4 parity bits as usual you write down their binary equivalents other than all 0's then you see that how these numbers is stacking up. Let us say p 1, p 1 will be correspond to 1 3 5 7 9 1 3 5 7 and 9, p 2 will be 2 3 6 7 2 3 6 7, p 3 will be 4 5 6 7 and p 4 will be only there are two 1's you can see 8 and 9.

So, in this way you can actually design the check bit mechanism, you can also fill up the table, you can fill up the parity bits using these formulas just like the previous one. So, I will leave it as an exercise for you to work it out and see how it works and here also you can check that if there are any single errors, bit errors it will be detected and also corrected right.

So, with this we come to the error come to the end of our very short discussion on error detection and correction. From the next lecture onwards we shall be moving on to the actual logic gates and their implementation.

Thank you.