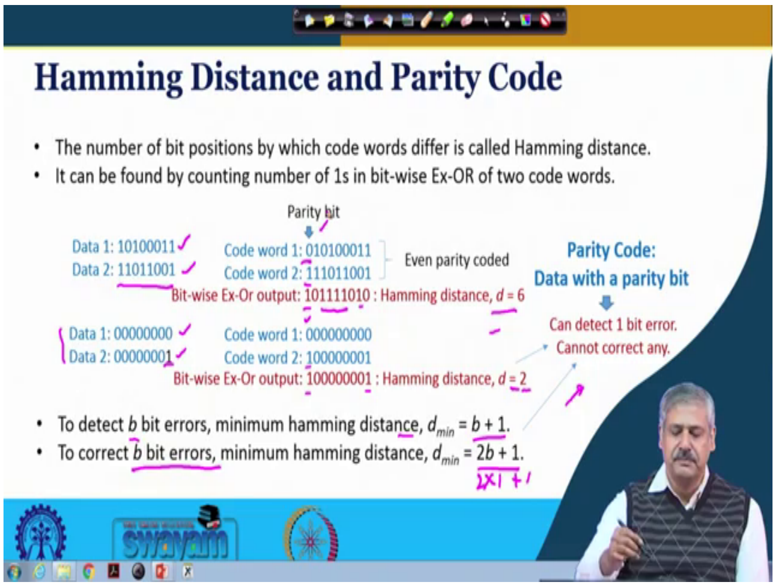
**Digital Electronic Circuits**
**Prof. Goutam Saha**
**Department of E & EC Engineering**
**Indian Institute of Technology, Kharagpur**

# Lecture – 29
# Error Detection and Correction Code

Hello everybody, in today's class we shall discuss Error Detection and Correction Code. In the last class we saw unweighted code, and we continue with the discussion of codes which are not weighted. And here, we shall see the code what that is generated is useful for detecting any error in the bit position. And if possible correction will be done for some specific kind of coding.

So, we shall discuss hamming distance. And while discussing it we will also shall include discussion on parity code that we have seen before; even parity and odd parity. And then we shall look at a specific code called hamming code its generation and how it is used for 1-bit error correction and 2-bit error detection.

(Refer Slide Time: 01:12)



## Hamming Distance and Parity Code

- The number of bit positions by which code words differ is called Hamming distance.
- It can be found by counting number of 1s in bit-wise Ex-OR of two code words.

Parity bit

Data 1: 10100011 ✓ Code word 1: 010100011
Data 2: 11011001 ✓ Code word 2: 111011001
Even parity coded
Bit-wise Ex-Or output: 101111010 : Hamming distance, $d = 6$

Data 1: 00000000 ✓ Code word 1: 000000000
Data 2: 00000001 ✓ Code word 2: 100000001
Bit-wise Ex-Or output: 100000001 : Hamming distance, $d = 2$

Parity Code:
Data with a parity bit
Can detect 1 bit error.
Cannot correct any.

- To detect $b$ bit errors, minimum hamming distance, $d_{min} = b + 1$.
- To correct $b$ bit errors, minimum hamming distance, $d_{min} = 2b + 1$. $2 \times 1 + 1$

So, first we look at what is hamming distance. So, hamming distance is the number of bit positions by which code words differ from one another, ok. If there is one code word and there is another code word, we look at the bit positions where they differ. And how we can get it? We can take bit wise Ex-OR operation of it, ok. So, whenever the bits are of same value; I mean code words is particular position so if it is 1 1 or 0 0 Ex-OR gate output will be 0, and when they differ that is 1 0 or 0 1 the output will be 1.

So, these Ex-OR bank of Ex-OR gates we look at its output and see how many places it is work. So, if those ones we count then that will give us the distance between two 1s, two code words, ok. So, that is what we say as hamming distance, ok.

So, two again group of binary digits and another digit this is the way we can calculate the distance, ok. So, we have seen parity coding, so let us look at an example of even parity coding. And there we look at two data: one is some random data we are talking about, that we shall look at some specific data to find out some minimum distance or so, ok.

So, this has got 1 2 3 4 4 1. So, even parity we are looking at, so the parity bit that is getting generated will be 0. Ultimately all the bits including the parity bit will be even, so that is the objective, right. And another data is this it has got 5 1, so the parity bit is 1 after Ex-ORing, right. So, we refer to that discussion on parity code where Ex-OR gates were used for generation and detection of parity; parity bit parity code related error in the transmission reception, ok.

So, eventually we have got six 1s over here. So, if you look at bit wise Ex-ORing of this two, so we shall see that here is 1 1 four 1s are here another one, so six 1s are there. So, we say between these two the distance is 6, ok. So, that is how we calculate the distance hamming distance between these two code words, ok. But for even parity code. So, if you look at another case say where the original data is differing only in 1-bit position. So, this is one case all 0s right. And just 1-bit is changing, so it has become one, right. So, what will be the corresponding code word; here all 0 so that means the parity beat will be 0. And 1 1, so parity beat becomes 1 so that even number of ones is there finally including the parity bit, right.

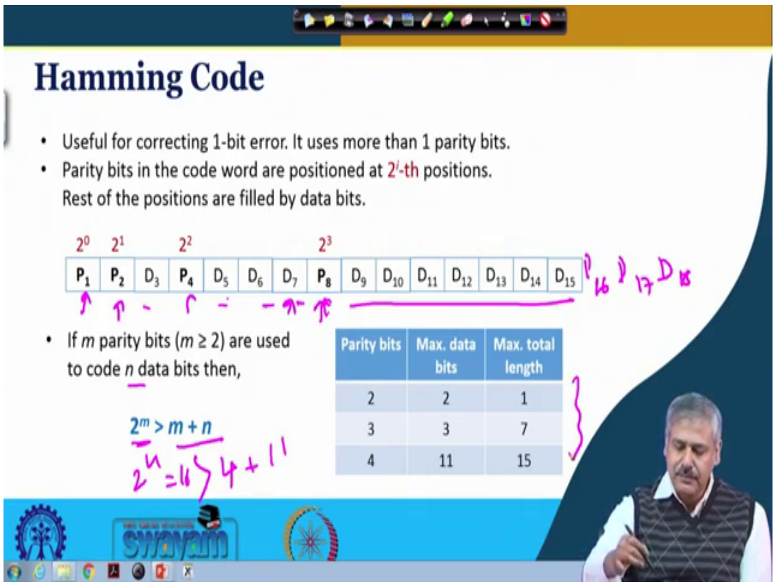
So, now, this is the minimum distance between to start with. Now when we do the Ex-ORing and accumulate we see that two 1s are there. So, this is 2. So, for this kind of parity coding where 1-bit parity code is introduced we shall see the minimum distance is of 2, right. So, that is what we taken out of.

Now the rules is to detect b bit errors number of you know bits if b, then the hamming minimum hamming distance is required is b plus 1; in the code words, So, we take a d code words generated for detecting the this error, so error detection code. So, the minimum distance required is b plus 1. So, hamming distance for this parity coding the kind of parity coding we have seen before is 2; minimum hamming distance 2. So, we can correct 1, 1 we can detect rather 1-bit error, ok. And for correcting b bit errors what is the minimum hamming distance required the rules says; it is 2 b plus 1, right.

So, to correct even 1-bit error we need the hamming distance 2 into 1 plus 1. So, 3, right. So, with this kind of coding we can detect 1-bit error, but we cannot correct any, right. So, this is what we have seen in the past on a error detection code using parity; introducing a parity bit, ok. Here we have taken the example even parity, it will stand good for odd parity also, right.

Yes.

(Refer Slide Time: 06:20)



## Hamming Code

- Useful for correcting 1-bit error. It uses more than 1 parity bits.
- Parity bits in the code word are positioned at $2^i$-th positions. Rest of the positions are filled by data bits.

| $2^0$ | $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | $P_2$ | $D_3$ | $P_4$ | $D_5$ | $D_6$ | $D_7$ | $P_8$ | $D_9$ | $D_{10}$ | $D_{11}$ | $D_{12}$ | $D_{13}$ | $D_{14}$ | $D_{15}$ |

- If $m$ parity bits ($m \geq 2$) are used to code $n$ data bits then,

$2^m > m + n$

$2^4 = 16 > 4 + 11$

| Parity bits | Max. data bits | Max. total length |
|---|---|---|
| 2 | 2 | 1 |
| 3 | 3 | 7 |
| 4 | 11 | 15 |

Now, we look at another kind of coding call hamming code, where we shall see how to correct a bit error as well. Not only detection is what we have seen as an example in parity code, ok. So, hamming code is something where we are putting not 1 parity bit more than 1 parity bit at specific positions. So, what are those specific positions? These are 2 to the power i-th position. So, what are the 2 to the power i-th position? So, 2 to the power 0 is 1, 2 to the power 1 is 2, 2 to the power 2 is 4, 2 to the power 3 is 8. So, these are (Refer Time: 07:05) by 2 to the power i-th position, ok

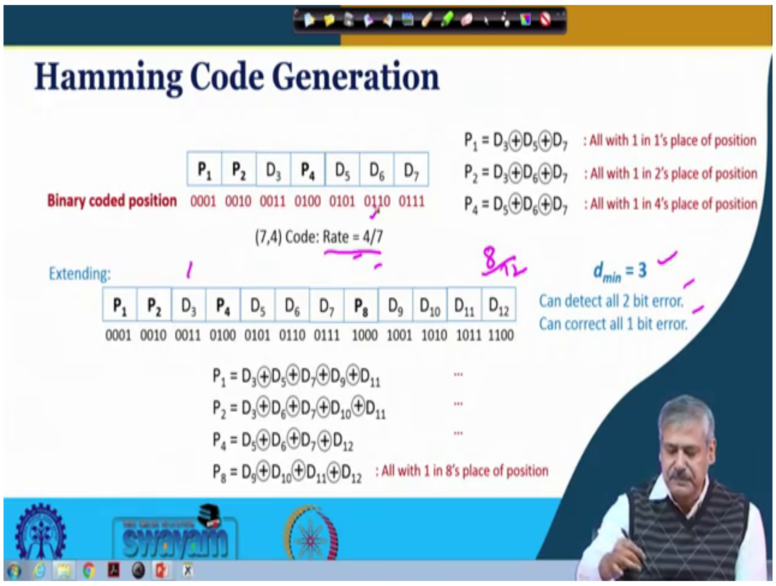
So, if we have a total code word, total code word that includes this parity bits which are included for the coding purpose and the data where which contains the information or the message or the numeric value whatever, right. So, if we include all of them together then, whatever code word is found the 2 to the power 0 that is 1, that is first position 2 to the power 1 2, 2 to the power 2 4; then 2 to the power 3 8. So, these positions will be filled by parity bit, ok.

And the rest of the position will be filled by the data like 3rd position, 5th position, 6th position, 7th position, ok. So, this is the way count will prove this. So, with P 1, P 2, P P 4 these 3 parity bits we can go up to D 7; right that means, total code word will be 7 4 data bit and 3 parity bit will be required in this kind of coding. So, whenever we go beyond that we have to introduce a parity bit over here P 8, so with that we can go up to D 15. And here whenever 16 position comes, right we have to introduce a parity bit D 16 and then again D 17, D 18 will continue. So, this is the way things are done in this kind of coding. Is it clear,

So, with this we can understand that with n parity bits, m greater than equal to 2 the number of bits that we can code message bits or data bit is n, then 2 to the power m 2 to the power m need to be greater than m plus n, ok. So, that is what we can see. Here 4 parity bits are there P 1, P 2, P 4, P 8, so m m is 4. So, 2 to the power 4 is 16, right 16 it need to be greater than m which is 4 plus n. So, here n we can go up to data we can go up to 11. So, 15. So, you see up to 15, because at 16 position another parity bit will come.

So, this is what we can see over here. Is it clear how are how we are introducing the parity bits in the code word hamming code? Fine.

(Refer Slide Time: 09:46)



## Hamming Code Generation

| | $P_1$ | $P_2$ | $D_3$ | $P_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|
| Binary coded position | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |

(7,4) Code: Rate = 4/7

$P_1 = D_3 \oplus D_5 \oplus D_7$ : All with 1 in 1's place of position

$P_2 = D_3 \oplus D_6 \oplus D_7$ : All with 1 in 2's place of position

$P_4 = D_5 \oplus D_6 \oplus D_7$ : All with 1 in 4's place of position

Extending:

| $P_1$ | $P_2$ | $D_3$ | $P_4$ | $D_5$ | $D_6$ | $D_7$ | $P_8$ | $D_9$ | $D_{10}$ | $D_{11}$ | $D_{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 |

$d_{min} = 3$

Can detect all 2 bit error.

Can correct all 1 bit error.

$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11}$ ..

$P_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11}$ ..

$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12}$ ..

$P_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12}$ : All with 1 in 8's place of position

So, you know the position of the parity bit alright, now how we generate the parity bit. So, the values will be 0s and 1s, ok. So, what will be those 0s and 1s, of course it will depend on the data, right. So, how it is generated? So, to generate parity bit for a system which is called hamming 7 4 code, that means total length is 7 and number of data is 4: D 3, D 5, D 6, D 7 and number of parity bits are 3; P 1, P 2, and P 4, ok. So, we will take these example.

So, what we do first? Each of this position 1 to 7 we are coding in binary, right. So 1, its binary coding is 0 0 0 1; 2 binary coding is 0 0 1 0 alright. So, this will continue 7 is 0 1 1 1. So, to generate P 1 to generate P 1 we shall consider all the positions of the data where the units in the units in the units case there is a 1; is it fine. So, D 3, D 5, D 7 so these are the places we have a one in the unit place. So, P 1 will be generated by Ex-ORing these beats; whatever these bits whatever message you are having. So, these are the places whatever the bits are there, we Ex-OR all of them together and then you get P 1.

So, how to get P 2? P 2 will be obtained by looking at ones that are present in the 2 place. So, this is all P 2 will be generated, so it is not coming. So, D 3 right, then D 6 and D 7. They have got one in the twos place, so D 3, D 6, D 7. You can see that and how P 4 will be generate now you can understand you can extend the idea. So, we shall look at 1s present in the 4th place, right. So, 4th place 1 present here in D 5, then D 6, and D 7. So, that is what you can see. Is it ok?

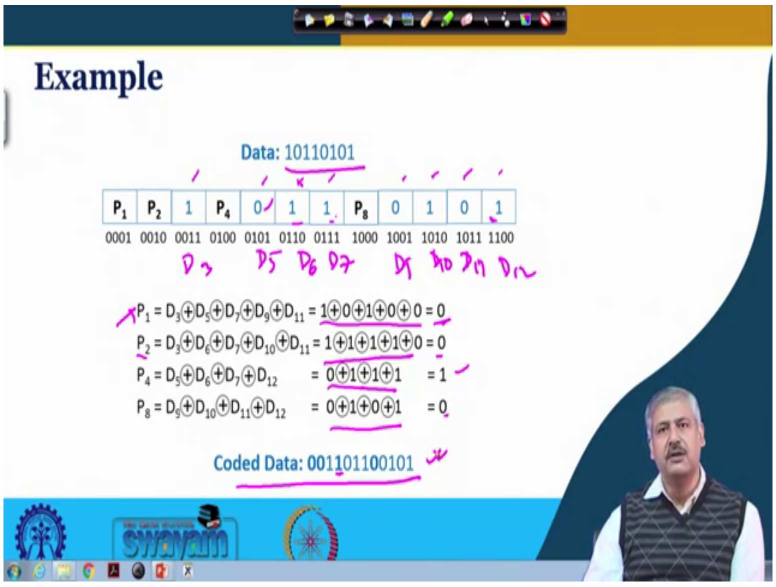
So, if you extend it if you extend it you can see for example, if you are having total code word is code length is 12 code word length is 12 right, and P 1, P 2, P 4, P 8; so 4 parity bits are there and then 8 data bits are there. So, if you want to code it, then the parity bit need to be generated. So, P 1 again you look at places where there is 1 in the units place, so P 1 you are generating. So, D 3, D 5, D 7, D 9, D 11 this is there, right.

Similarly, P 2 similarly P 2 we will look at places where there are 2 right, so D 3, D 6, D 7 already there you can see that D 10, and D 11 these 2 they join in P 2 generation when there are 12 bits 12 bit code word. So, P 4, P 4, D 5, D 6, D 7 was already there D 5, D 6, D 7 up to this then in 4th place you can see over here the D 12 has got a 1, ok. So, D 12 will come. And finally, for P 8 in the 8s place where there is 1. So, P 8, D 9, D 10, D 11, and D 12, ok. Is it clear now?

Now, you can see if you put various numbers and all, so the minimum distance that you can get over here is 3 between 2 code words. And for which it can detect as I said it was b plus 1, so b will be 2 and can correct which was 2 b plus 1, so this is 1. So, this is what it can do, ok. And of course, what you can understand by doing this we are actually increasing the length of the message, right length of the data. So, actually out of which useful data is 8; 8 is something which is unknown and P is the one that you are introducing. So, at the other end; so 8 out of 12 will be useful in this kind of coding and in the 7 4 code 4 data bit out of 7 will be useful. So, this is the way one can actually define the rate at which actually information is going from one place to another.

So, because of the introduction with the this parity bit you can see actual rate though you are sending 7 bits, but your message conveying bit is only 4. So, rate has been reduced. So, this is one thing that we need to take into account that when we are doing it to get some benefit out of this kind of you know methodology there is some cost associated this cost is additional bit. So, that need to be sent along with which will bring down effective data rate; effective transmission rate, fine.

(Refer Slide Time: 15:36)



## Example

Data: 10110101

| P1 | P2 | 1 | P4 | 0 | 1 | 1 | P8 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 |
| | | D3 | | D5 | D6 | D7 | | D9 | D10 | D11 | D12 |

$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 0$

$P_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$

$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} = 0 \oplus 1 \oplus 1 \oplus 1 = 1$

$P_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} = 0 \oplus 1 \oplus 0 \oplus 1 = 0$

Coded Data: 001101100101

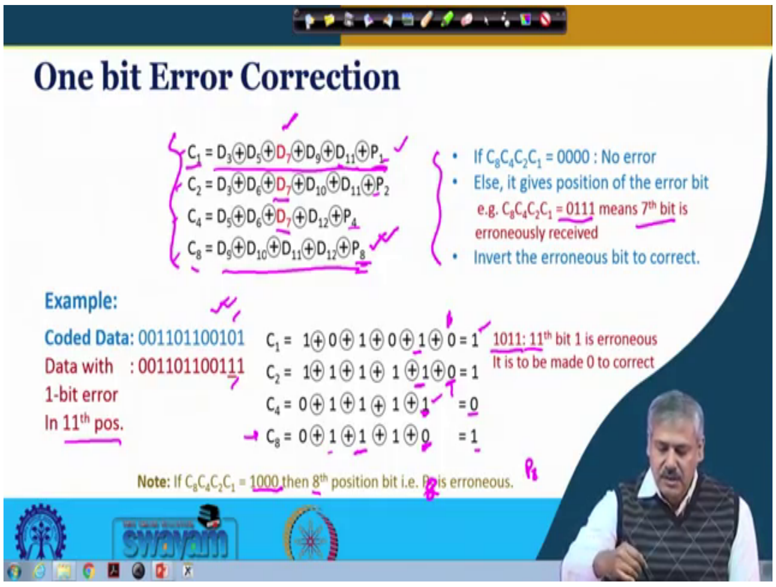
Now, let us look at one example, ok. So, in this case the data is 1 0 1 1 and 0 1 0 1. So, basically 8 data bits are there. So, 4 parity bits will be required. So, data bits are placed D 3 and D 5, D 6, D 7, D 9, D 10, D 11, D 12,, right. So, this is the way they are defined. So, this is D 3, D 5 up to this is D 12, but that start your. So, you know that the nomenclature how this they are designated.

So, to get P 1, so D 3, D 5, D 6, D 7, this is D 9, D 10, D 11, and this is D 12. So, D 3, D 5 D 7, D 9 D 11 this you need to take. So, D 3 is 1, D 5 is 0, D 7 is 1, D 9 is 0, and D 11 is 0, right. This is the way you look at it. Ex-ORing means we shall look at the number of 1s. So, if the number of ones at even, you have to invigilate the number of 1s total number of ones are odd then the output will be 1. So, number of 1s are even in this particular Ex-ORing. So, the output here is 0. Is it clear.

Similarly, for P 2, D 3, D 6, D 7, so you can substitute all those values we can see that this is 0 P 4. Similarly we can see here there are 3 walls D 5, D 6, D 7, D 12. So, D 5 is 0 over here, D 6, D 7; D 6, D 7 are 1 1, 1 1 and D 12 D 12 is 1, ok. So, three 1s are there, so the output is 1. And finally, P 8 you can say that this is 0, right.

So, then the coded data will look like this 0 0 because P 1 is 0, P 2 is 0, P 4 is 1. So, P 4 is coming over here D 3 is 1, so D 3 is there. So, D 1 that are in bold is the parity bit, ok. So, this is the way the coded data will be generated and to be sent, ok.

(Refer Slide Time: 18:03)



## One bit Error Correction

$C_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} \oplus P_1$

$C_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} \oplus P_2$

$C_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} \oplus P_4$

$C_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus P_8$

- If $C_8C_4C_2C_1$ = 0000 : No error
- Else, it gives position of the error bit e.g. $C_8C_4C_2C_1$ = 0111 means 7th bit is erroneously received
- Invert the erroneous bit to correct.

**Example:**

Coded Data: 001101100101

Data with 1-bit error In 11th pos. : 001101100111

$C_1 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 = 1$

$C_2 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$

$C_4 = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$

$C_8 = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$

1011: 11th bit 1 is erroneous
It is to be made 0 to correct

**Note:** If $C_8C_4C_2C_1$ = 1000 then 8th position bit i.e. $P_8$ is erroneous.

Now let us see how the corrections called 1-bit error correction that is possible through this hamming code, ok. So, if we just look at the way it is generated, so for example if we consider that in this particular case D 7 bit is received erroneously, ok. Just for example; actual example we shall see later. So, D 7 bit is generated received erroneously. So, whatever was there it has been flipped, ok. So, what we do, in the receiver side other than what was required to generate the parity bits D 3, D 5 D 7, D 9 the 1s position with them we also Ex-OR the P 1; is it ok.

So, if all these where odd if all these where odd D 3 to D 11 then P 1 would have been 1 or I mean the number of 1s there that was odd. So, together it it would have been even; is it clear. So, if we include P 1 in the Ex-ORing process the way you had done for you know single bit parity code, ok. So, inclusion of P 1 in the receiving side in the Ex-OR operation would have made everything even the whole of it is even. So, if this was not erroneous then these Ex-OR operation together with P 1 would give me 0. So, if all of them together was 0 then P 1 would have been 0 which is the case in in a specific example or if it was 1 it was. So, effectively all of them becomes even parity, right.

So, D 7, if not erroneous this C 1 generated will be 0. But if D 7 is erroneous it will generate because of it has thick, so odd number of 1 will come there, so C 1-bit will become 1, right. D 7 let us see what else which other area where it where it is there. So, C D 7 is over here again, D 7 is over here again and it is not there with C 8, right.

So, D 7 flipping will not affect C 8, right. So, if you now put them together; I mean if you take the receiving side so what will it generate; C 8, C 4, C 2, C 1 it will generate 0 1 11 from this side to this side. And this refers to 7th 0 1 1 1 is 7, ok. So, that indicates that 7 bit is erroneously received. And if you invert it you will get the actual information, right. So, this is how 1-bit error correction is done. So, it will be true for any other bit position. If you will see that only corresponding place over here is getting you know flip a getting a 1, because of the flipping over here, ok.

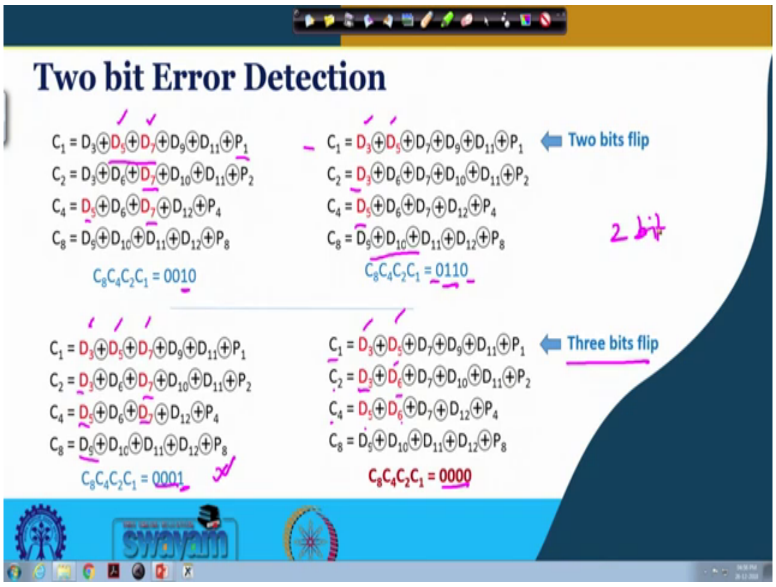
So, the example that we are taking the code word, right we can say a for a change not say D 7; D 7 is already understood that 11 position the bit has changed bit has flipped. So, this 0 has become 1, right So, we shall then perform with the parity bit that is there wherever D 11 is present, D 11 is present D 11 is present here, right. So the D 11, D 11 is present here, and D 11 is present here. So, these where the places which was 0 before in the generation process, now this has become 1, and parity bits bit got added, ok. So, it is not there with this one D 11 is not there.

Now if you do the Ex-ORing operation you can see 1 1 1; three 1s are there. So, output will be 1. This is your P 2, this is your P 4 right, this is your P 1, this is P 2, this is P 4, and this is P 8, so this is 1 this is 1. In this case four 1s because of presence of you know this P 4 this three 1s it finally it will become 0 and because of this three 1s are there this is 1. So, what we get: 1 0 1 1 C 8 to C 8, C 4, C 2, C 1. So, this is what we are getting. So, this indicates 11 in decimal. So, 11 position there is error and if you flip it in you can get the corrected bit well, ok.

And note that the flipping can occur even in the parity bit position, because when the message is coming you know do not know because of the noise or some issue which bit position has it. It is not no lesser down to the data bits will get you know for uptake, ok. So, if say any of the parity bit is getting corrupted it say; P 4 is getting corrupted, ok. So, P 4 is getting corrupted means P 4 is there in the sorry; let us consider that P 8 is getting corrupted. So, P 8 is there with this one only this particular case P 8 is not there in any other places. So, only because of it has changed it will should have been you know even parity whole of this thing. So, these output would have been 0. Now this output will be showing 1 C 8. And since P 4 is P 8 is not there with anyone of them right, so all of them will be showing 0 because they are not getting affected all those bits are in order.

So, 1 0 0 0 is that what you will get and the final you can say that 8th position is erroneous, ok. Is it clear? So, if it is 8 position then P 8 is erroneous; P 8 is erroneous.

(Refer Slide Time: 24:37)



## Two bit Error Detection

$C_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} \oplus P_1$
$C_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} \oplus P_2$
$C_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} \oplus P_4$
$C_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus P_8$

$C_8C_4C_2C_1 = 0010$

$C_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} \oplus P_1$ Two bits flip
$C_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} \oplus P_2$
$C_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} \oplus P_4$
$C_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus P_8$

$C_8C_4C_2C_1 = 0110$

2 bit

$C_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} \oplus P_1$
$C_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} \oplus P_2$
$C_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} \oplus P_4$
$C_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus P_8$

$C_8C_4C_2C_1 = 0001$

$C_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} \oplus P_1$ Three bits flip
$C_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} \oplus P_2$
$C_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} \oplus P_4$
$C_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus P_8$

$C_8C_4C_2C_1 = 0000$

Now let us look at 2-bit error detection that is possible here, ok. So, let us see how what does it mean. So, we consider a case where this D 7 is erroneous and D 5 is also erroneous, ok

So, what will happen in this case: C 1 is 2-bits have flipped. So, C 1 will be 0 only right, if 1-bit flips then it will give a 1 in when you consider the parity bit and all overall it is even parity. Since 2-bit has been flipped, so it will be giving you 0. So, you cannot detect any error over there, ok. So, here D 5 is not there, so D 7. So, in this case this will give a 1 because only 1-bit has flipped D 5, D 6, D 5, D 7 is there in C 4. So, 2-bits have flipped, so it will not detect any error. And C 8 does not have D 5 or D 7,so it will not detected anything.

So, only C 2 will give you a indication that there is a error. But it it is no it cannot correct it because it is indicating 2, ok. if only 1-bit error was there, right to there you could have seen that it is P 2 that has been erroneous, ok. So, what that is not the case here, so more than 1-bit correction means not possible so D 5 D 7 cannot be corrected, right.

So, similarly D 3 and D 5 if you see, so D 3, D 5 both flipped so that means it will this will be 0 it cannot detect anything, so 0. And D 3 is here D 5 is not present D 5 is here D

3 is not present, so there they will generate 11. And you are in this C 8 is in either of D 3 or D 5 are there, so this will be 0, ok. So, this again because of C 2, C 8 2 C 1 is not all 0, so a error is detected, error is detected, ok. So, if there are 2-bits erroneous definitely we can see that the it will be detected. So, the final C 0 C 8 to C 1 will not be all 0, right.

Now more than 2-bit flipping let us see 3 bits case. So, we take an example where D 3, D 5, D 7 they have flipped, ok; which is possible this is one random case, right. So, C 1 3 3 f has changed. So, for which then if 2 then one cancels other, so basically 2. So, here then it will detect a odd parity not even parity, so this will be 1, right the other case D 3 and D 7. So, it will detect both of them is you know flipped, so no error will be detected similarly no error will be detected. And in this case D 3, D 5, D 7 are not there, so 0 0 0 1.

So, error is detected. So, 3 bit error is detected, ok. But we were saying that it can detect 2-bit error. So, 2-bit error detection is guaranteed, 3 bit error in this case detect got detected, ok. But let us see another example whether it is getting detected or not.

So, D 3, D 5 and D 6 let let us consider that this 3 bits have got corrupted have got flipped, ok. So, D 3, D 5 is there in C 1 0 ratio. So, since 2-bits have got flipped, so it will not be detecting it will be 0. D 3 D 6 is over here, 2-bits got flip it will be 0. D 4 D 6 is over here 2-bits got got flipped it will be 0. And in this case none of the D 3, D 5, D 6 are there, so it will be generating 0. So, 3 bits are erroneous, but the output is 0, so it will be seen as that no bits no error is there in the code word, ok. So, that is why you are saying that it can detect 2-bit, error and can correct 1-bit.

So, this is the basic premise of error detection and error correction, and with hamming distance, and the relation between hamming distance, and the number of bits it can detect and number of bits it can correct, ok. And there are many other options available. So, the idea here was to give you an idea or foundation and how digital circuits can be made around it. And we have seen that this Ex-OR gates Ex-OR relations are useful in this.

(Refer Slide Time: 29:28)

References

References:

- Donald P. Leach, Albert P. Malvino, and Goutam Saha, Digital Principles & Applications 8e, McGraw Hill

So to conclude: we can see that the number of bit positions by which the code words differ called is called hamming distance and the hamming with distance is related to number of bits it can detect as erroneous or number of bit it can correct as erroneous; I mean erroneous bit it can correct, ok. And in hamming code generation: 2 to the power ith position we need to be need to have parity. And Ex-OR operation is there for generating the parity bit as well as looking at the receiver side. And from that we can arrive at the for the correction bit position and for detection whether there is a error or not, ok.

Thank you.