# Switching Circuits and Logic Design

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

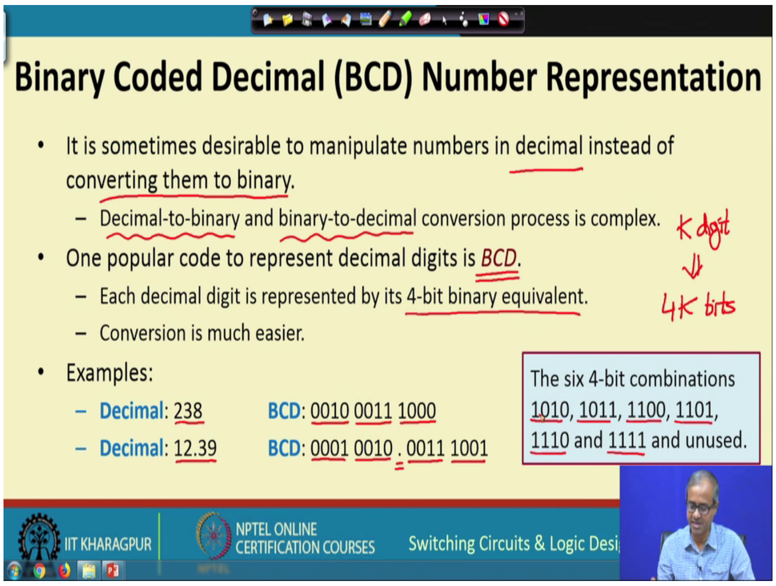
## Lecture - 05
## BCD and Gray Code Representations

So, in the previous lectures if you recall, we had talked about the different number: systems binary, decimal, octal and hexadecimal. And also assured how to convert one number system to the other and also with respect to some examples illustrated, how we can represent numbers both positive and negative in binary and, also carry out some arithmetic operations like addition and subtraction.

Now, there are some applications where the primary numbers that we want to manipulate on are decimal in nature. You take an example of a payroll application, where the employs salaries are calculated. So, in those kind of applications, there is very little arithmetic or calculations involved. See when you have very heavy calculations then it make sense to convert the numbers into binary, because binary circuits are much faster binary adder subtractors multipliers are much more efficient to work on numbers, but when you have such decimal numbers there should be some alternative so that we need not have to convert every time into binary and back, because there is very little calculation in such kind of applications, mostly will be doing some reading from a file and printing stuff like that.

So, in today's lecture we shall be considering on precisely that. So, the topic of the lecture is BCD and gray code representations of numbers.

(Refer Slide Time: 02:00)



## Binary Coded Decimal (BCD) Number Representation

- It is sometimes desirable to manipulate numbers in decimal instead of converting them to binary.
  - Decimal-to-binary and binary-to-decimal conversion process is complex.
- One popular code to represent decimal digits is *BCD*.
  - Each decimal digit is represented by its 4-bit binary equivalent.
  - Conversion is much easier.
- Examples:
  - Decimal: 238 BCD: 0010 0011 1000
  - Decimal: 12.39 BCD: 0001 0010 . 0011 1001

The six 4-bit combinations 1010, 1011, 1100, 1101, 1110 and 1111 and unused.

K digit $\Downarrow$ 4K bits

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

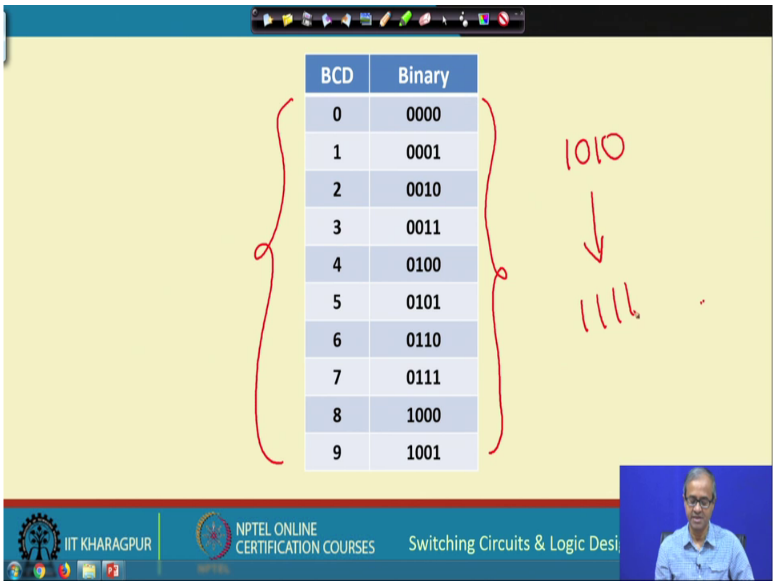
So, let us see what the idea is. We start with something called binary coded decimal numbers. Now, as I said that it is sometimes desirable to manipulate numbers in the decimal number system directly, instead of converting them to binary why the reason is very simple we have seen that of course we can do decimal to binary and binary to decimal conversion. But this conversion processes are not very trivial, we will have to do either repeated multiplication, or repeated division to carry out the process.

So, instead if we have an alternate representation where, you do not have to make this conversion do not have to carry out these multiplications and divisions, then it will be very efficient in terms of the overall competition. So, what we talk about are something called binary coded decimal or BCD, BCD is the acronym for binary coded decimal.

So, what is the idea in BCD here every decimal digit is represented by 4 bit binary equivalent. So, you can say if I have a K digit number, then each of the digits will be represented by 4 bits. So, you will be having finally, 4 into K that many bits, this is the basic idea. So, here are some examples, suppose I have a decimal number 238. So, if you want to convert it to BCD you convert it by digit by digit 2, the 4 bit binary equivalent is 0 0 1 0 3, it is 0 0 1 1 8, it is 1 0 0 0. So, this is your BCD representation talk about a number with decimal point let us say 12.39 12 1 and 2 1 2, then the point 3 and 9.

So, you see the conversion is 1 to 1 straight forward just the only point to noted is that we are talking about the numbers 0 to 9 which are the decimal digits and we are converting them into 4 bit binary. Now, you recall in 4 bit binary, we can represent 2 to the power 4 or 16 distinct numbers. So, out of them we are using only 10 the remaining 6 were not using right. So, which 6 combination these 6 combinations, we are not using 1010, 1011. 1100, 1101, 1110 1 and 1111, this 6 combinations we are not using in case of the BCD system right. So, let us move on fine.

(Refer Slide Time: 05:25)

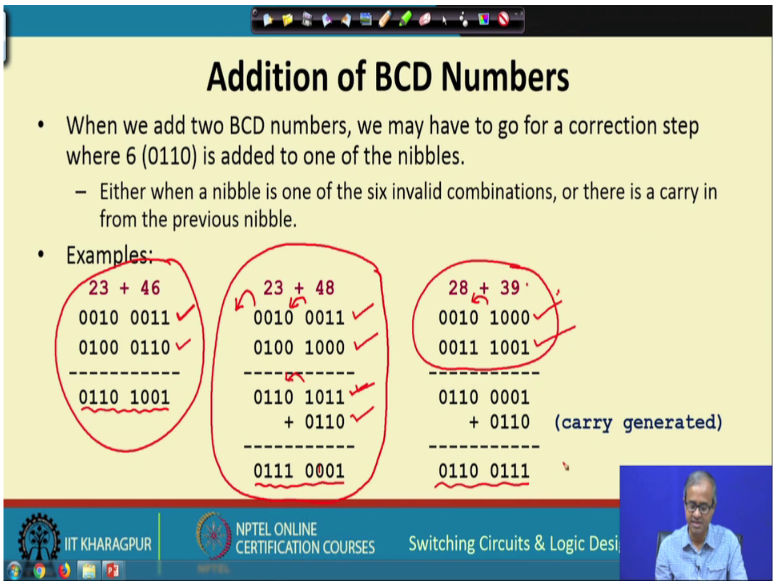


| BCD | Binary |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |

1010
1111
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

So, here you see the 10 decimal numbers digits 0 to 9 and, they are corresponding 4 bit binary equivalents. Now, as I said the remaining 6 combinations which corresponds to 10 which is 1 0 1 0 up to 1 1 1 1 these are not used in this case, right.

(Refer Slide Time: 06:00)



## Addition of BCD Numbers

- When we add two BCD numbers, we may have to go for a correction step where 6 (0110) is added to one of the nibbles.
  - Either when a nibble is one of the six invalid combinations, or there is a carry in from the previous nibble.
- Examples:

```
 23 + 46          23 + 48          28 + 39
0010 0011        0010 0011        0010 1000
0100 0110        0100 1000        0011 1001
-----------      -----------      -----------
0110 1001        0110 1011        0110 0001
                    + 0110           + 0110   (carry generated)
                 -----------      -----------
                 0111 0001        0110 0111
```

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

This is something you have to remember, Now, let us next let us see we said that the advantage of BCD is that, we can convert decimal numbers into BCD coded binary representation very easily, that we had seen how. Next is, if you want to do some simple calculations let us say some additions how do we do it directly in BCD. So now, shall we talking about some rules for addition of BCD numbers see the rule is fairly simple you see that, when you add BCD numbers there may be some correction steps we may have to go for there can be some correction steps now in the correction steps.

What you do we may have to add 6 which in binary 0 1 1 0 to 1 of the nibbles. Now, the rules are very simple when do we do the correction we do the correction, if either 1 of the nibbles is corresponding to an invalid combination, which means that the combination from 10 to 15. So, the one of those 6 so, if it is one of those 6 it is an invalid combination then we apply the correction step, or there is another alternative if there is a carry from the previous stage from the previous nibble, then also you carry out this correction. Let us look into an example to illustrate this point fine. So, here there are 3 examples which are worked out, let us look at the first example here.

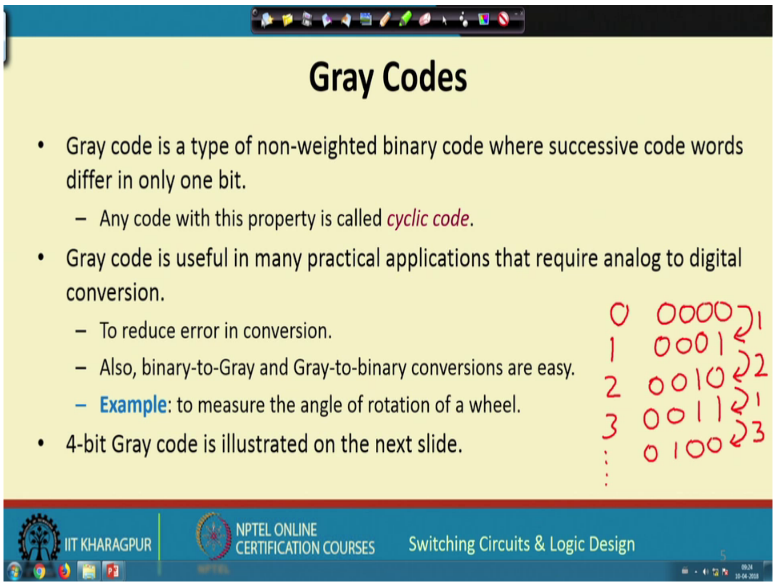
So, here we are adding 23 and 46. So, you see 23 we have coded in BCD like this 2 0 0 1 0 and 3 0 0 1 1, 46 we have coded like this 4 is 0 1 0 0, 6 is 0 1 1 0. Now, when you add you see bit by bit we follow the same rule for binary addition 1 plus 0 is 0 is 1 no carry, 1 and 1 is 0 with the carry of 1 1 and 1 is 0 with the carry of 1 1 0 0 is 1 no carry or 0 0 1 and 0 is 1 no carry 0 1 is 1 0 1 no carry 0 0 0. So, you see after this addition these both this nibbles are valid decimal digit BCD digit, this is 6 this is 9 and also there is no carry from one stage to the next. So, none of these 2 correction conditions are true here therefore, whatever you got here this is our final result you see 69, 23 plus 46 is 69 fine, this is one example. Let us take another example where one of the nibbles become invalid, take a example 23 plus 48. So, this is 23 2 and 3 and this is 48 4 and 8.

So, let us again add 1 plus 0 is 1 no carry 1 and 0 is 1 no carry, all 0 0 0 and 1 is 1 no carry all 0 1 0 is 1 no carry, 0 1 is 1 no carry 0. So, you see here there is no carry from either this stage to here, or carry out from the final stage, but one of the nibbles here this one is invalid 1 0 1 1, 1 0 1 1 is an invalid combination in BCD ok, that is why we apply a correction step on this nibble, we add 6 we add 6 again following the rule of binary addition, 1 and 0 is 1 1 and 1 is 0 with the carry of 1 1 0 1 is 0 with the carry of 1 1 1 0 is 0 with the carry of 1. So now, there is a carry coming ok.

So, this 1 and 0 will become 1 1 1 0 so, you see the result is coming to 71, this is 7, this is 1 which is 23 plus 48. So, the result is coming to correct now take another example, let us say 28 plus 39 we take this example here, this is 28 this is 39. Let us do the addition again 0 and 1 is 1 0 0 0, 0 1 1 is 0 with the carry of 1 you see now there is a carry coming from here to here, this 1 and 1 is 0 with the carry of 1 1 1 1 is 1 with a carry of 1 1 0 0 1 0. So, you see after addition the nibbles are both valid 0 1 1 0 or 1 0 0 1 both valid, but there was a carry here from the first nibble to this again. So, you add a correction steps 6 here. So, after addition this will become this which is 67, 28 plus 39 is 67.

So, now you understand these are the simple rules for BCD addition. So, whenever we have such BCD numbers, you simply just add the numbers first in binary, then see either one of the nibbles are becoming invalid, or there is a carry well if either of the condition is true, then you add 6 to one of the digits. Now, here the example I have taken applies to the first nibble it can also happened to the second nibble, then also you will have to do the same thing right, you will have to add 6 to it. So, the rule is like this.

(Refer Slide Time: 12:30)



## Gray Codes

- Gray code is a type of non-weighted binary code where successive code words differ in only one bit.
  - Any code with this property is called *cyclic code*.
- Gray code is useful in many practical applications that require analog to digital conversion.
  - To reduce error in conversion.
  - Also, binary-to-Gray and Gray-to-binary conversions are easy.
  - **Example**: to measure the angle of rotation of a wheel.
- 4-bit Gray code is illustrated on the next slide.

0 0000 1
1 0001 2
2 0010 1
3 0011 3
0100
⋮

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

Next let us talk about another kind of code using which also you can represent decimal digit not only decimal digits, any binary number any multi bit number you can represent in gray code.

Now, let us try to see why we need gray code first. And what is the basic reason that we are moving to a new code ok. Let us first talk about the conventional number system, like when we talk about the numbers let us say in decimal 0 1 2 3 etcetera. So, when you write the representations in let us a 4 bit representation 0 0 0 0, 1 is 0 0 0 1, 2 is 0 0 1 0, 3 is 0 0 1 0, 4 is 0 1 0 0 and so on. Now, the issue is let us see from 1 combination to the next, how many bits are changing. Let us look at that see from 0 to 1 only the last bit is changing. So, only 1 bit is changing 1, but from 1 to 2 both the last 2 bits are changing. So, there are 2 bits changing 2 to 3 again only last bit only 1, but 3 to 4 you see all the 3 last bits are changing this 3.

Now, there are many applications we shall be talking about, where it is good if you can reduce the number of bits that are changing between successive counts, if multiple number of bits are changing there can be a problem, we shall take an example and show how that that kind of problem can come, but for the time being assume that if multiple bits change there can be a problem. So, we are trying to avoid that, we are trying to come up with the counting system, where only 1 bit is changing across successive count values this is what we are trying to. So, let us see what is a gray code.

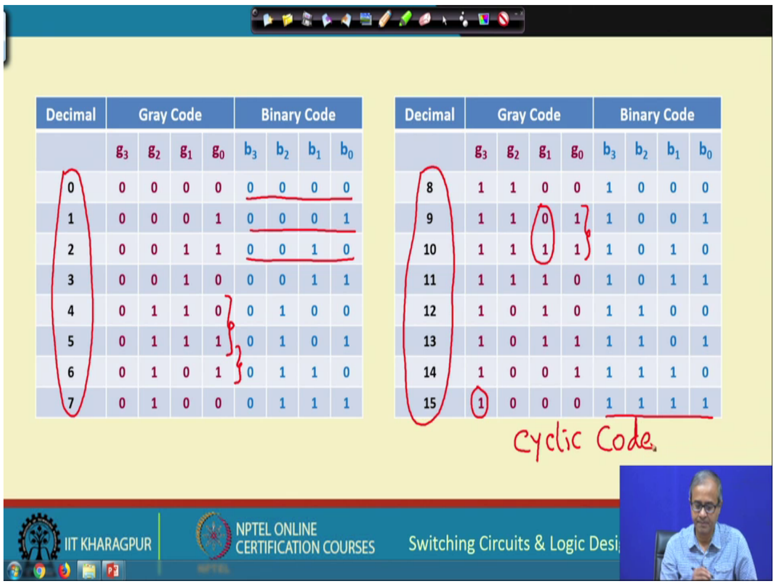
The first thing about gray code is it is non weighted, means unlike binary numbers where every bit position has a weight 2 to the power 0 2 to the power 1 like that, in a gray code number the bit positions you cannot assign in weight like that and, any and the second important thing which I have already mentioned that successive code words deferred in only one bit and any code where this property holds, where there is a 1 bit difference between successive words is called a cyclic code.

So, gray code is a cyclic code. Now, the second part I shall be explaining a little later with the help of an example, but for the time being let us assume that for any application where do you need to convert from analogue to digital analogue to digital means say, any value which is continuous in nature, like you think of temperature you think of motion, you think of the rotation of a wheel these are all continuous events. So, you cannot say the I will only rotate by 1 degree, 2 degree, 3 degree I can rotate by 1.5 degree, 1.2 degree, 1.15 degree anything I want right these are continuous motion, but when you want to represent this kind of a continuous quantity or parameter in binary. So, there has to be a mechanism for this conversion alright.

So, if you convert it into 4 bit representation, then you can have 16 possibilities. Suppose the rotation of a wheel you are representing in 4 bits. So, you can have 16 possible rotation angles because, in 4 bits you can represent 0 to 15, but if you represent it in let us say 8 bits binary, then you can have 256 rotation angles that you can represent 2 to the power 8 is 256 right. So, gray code is useful here, and this point I shall be explaining why it reduces error in conversion this I will explain a little later, the second thing is that it is fairly simple to convert a binary number to gray code or a gray code number to binary we shall also see through example.

So, this example of a wheel we shall be seeing later and explaining this points with respect to that. Let us talk about a 4 bit gray code.

(Refer Slide Time: 17:45)



| Decimal | Gray Code | | | | Binary Code | | | |
|---|---|---|---|---|---|---|---|---|
| | $g_3$ | $g_2$ | $g_1$ | $g_0$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

| Decimal | Gray Code | | | | Binary Code | | | |
|---|---|---|---|---|---|---|---|---|
| | $g_3$ | $g_2$ | $g_1$ | $g_0$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ |
| 8 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 9 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 10 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 11 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 12 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 13 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 14 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 15 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

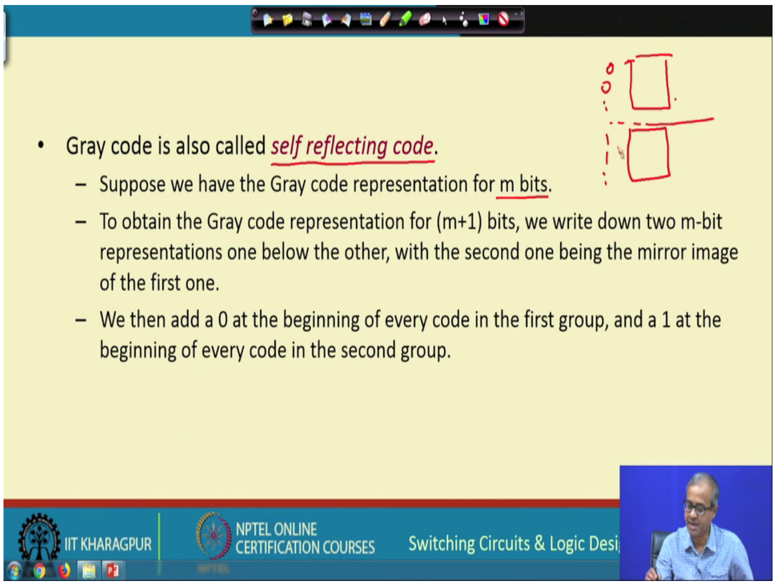
Cyclic Code.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

This table shows a 4 bit gray code and also the corresponding binary representations, the first column shows the decimal numbers 0 to 7, 8 up to 15. And these are first the binary representations so, as we have already seen 0 1 2 3 4 up to 15 side by side, we have also shown the corresponding gray codes, these are the grey codes for this decimal number 0 is represented as 0 0 0 zero 0, 1 as 0 0 0 1 2 as 0 0 1 1 3 as 0 0 1 0 and so, on you see between any pair of codes exactly 1 bit is changing, let us say if you consider these 2 last bit is changing. If you consider these two this second large middle bit is changing.

So, between any pair of bit patterns so you see here take let us say this and this. So, here also the middle bit is changing exactly 1 bit change. Similarly from 15 back to 0 only the first bit changes; this is what gray code basically is. So, across successive codes exactly 1 bit positions change the states right, this is one property you mentioned of so, called cyclic codes, this is an example of a cyclic code, fine.

(Refer Slide Time: 19:30)


- Gray code is also called *self reflecting code*.
  - Suppose we have the Gray code representation for m bits.
  - To obtain the Gray code representation for (m+1) bits, we write down two m-bit representations one below the other, with the second one being the mirror image of the first one.
  - We then add a 0 at the beginning of every code in the first group, and a 1 at the beginning of every code in the second group.

0
0
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

There is another interesting thing about gray code, for which it is also sometimes called self reflecting code. Now, the idea of self reflecting code is like this; suppose, I have my gray bit gray code representation for 4 bits. Now, I want to have it extended to 5 bits I want to have a 5 bit gray code representation. Now from 4 bits it is very easy to convert it into 5 bits from 5 bits it is very easy to convert it to 6 bits. This is the basic idea and such codes is sometime called self reflecting code, for reason that we will see how the idea is very simple.

Suppose I have an m bit representation, I have m bit gray code representation already available with me. So, I am showing it is as a box, let us say this is my m bit representation, what I do I write down that m bit representation again not directly, but as a mirror image. The last number goes first the first number goes last, that is called self reflecting as if it is reflecting and, in the first segment I am adding a bit 0 in the beginning and in the second segment I am adding the bits 1 in the beginning.

So, I get the gray code for m plus 1 bits this is a basic idea. So, see what do you mention to obtain the gray code representation for m plus 1 bits, we write down 2 m bit representations one below the other just like I showed, with the second 1 being the mirror image of the first one, it is just the mirror image. Then we add 0 in the beginning of every code in the first block and add a 1 at the beginning of every code in the second block.

(Refer Slide Time: 21:34)

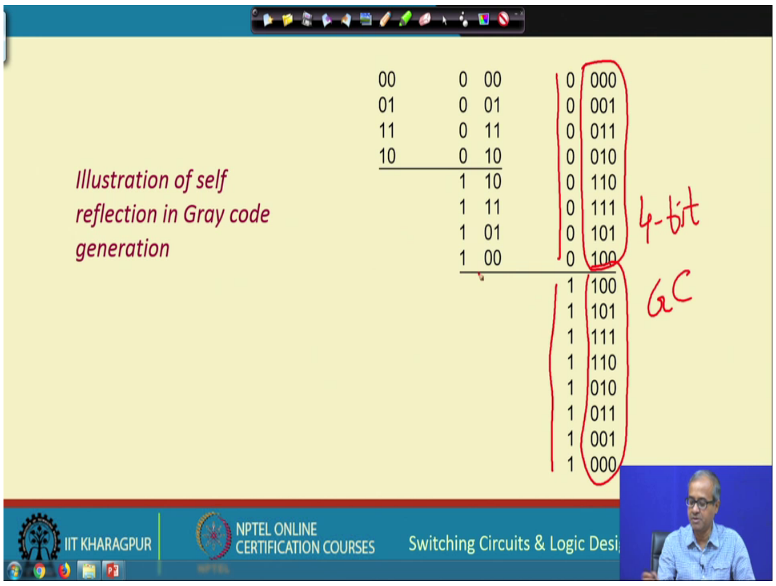


*Illustration of self reflection in Gray code generation*

| | | |
|---|---|---|
| 00 | 0 00 | 0 000 |
| 01 | 0 01 | 0 001 |
| 11 | 0 11 | 0 011 |
| 10 | 0 10 | 0 010 |
| | 1 10 | 0 110 |
| | 1 11 | 0 111 |
| | 1 01 | 0 101 |
| | 1 00 | 0 100 |
| | | 1 100 |
| | | 1 101 |
| | | 1 111 |
| | | 1 110 |
| | | 1 010 |
| | | 1 011 |
| | | 1 001 |
| | | 1 000 |

4-bit GC

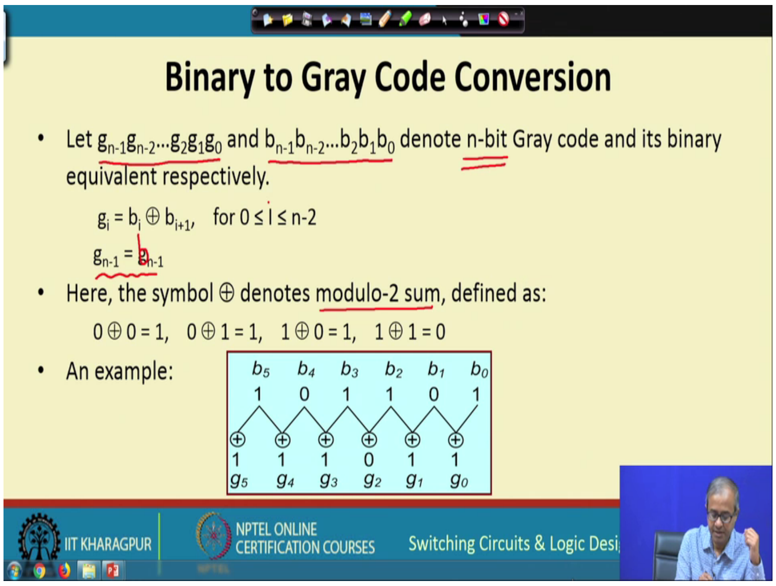
IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES Switching Circuits & Logic Desi

So, let us take an example this one. Suppose I have a gray code for 2 bits, this is the gray code representation 0 1 2 and 3. So, when I want to generate the gray code for 3 bits, you see I write this twice first as it is second as a mirror image mirror image means, 1 0 comes first 1 1 next 0 1 next and 0 0 last. Then I add 0's in the beginning of the first block and 1 in the beginning of the second block.

So, I get 3 bit gray code you see check here across every pair of codes exactly 1 bits are changing, 1 bit is changing. Similarly from 3 bit to 4 bit you follow the same principal, you write down the 3 bit gray code block twice, 1 the mirror image of the other same way just reflecting add 0 in the beginning of the first block add 1 in the beginning of the second block. So, what you get is 4 bit gray code.

So, generating gray code from m bits to m plus 1 bit is fairly simple as you can see right. So, this is also a reason why it is called self reflecting code. Now, let us look at binary to gray code and gray code to binary conversions.

(Refer Slide Time: 23:07)



## Binary to Gray Code Conversion

- Let $g_{n-1}g_{n-2}...g_2g_1g_0$ and $b_{n-1}b_{n-2}...b_2b_1b_0$ denote n-bit Gray code and its binary equivalent respectively.

  $g_i = b_i \oplus b_{i+1}$, for $0 \le I \le n-2$

  $g_{n-1} = b_{n-1}$
- Here, the symbol $\oplus$ denotes modulo-2 sum, defined as:

  $0 \oplus 0 = 1$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$
- An example:

| $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 |
| $\oplus$ | $\oplus$ | $\oplus$ | $\oplus$ | $\oplus$ | $\oplus$ |
| 1 | 1 | 1 | 0 | 1 | 1 |
| $g_5$ | $g_4$ | $g_3$ | $g_2$ | $g_1$ | $g_0$ |

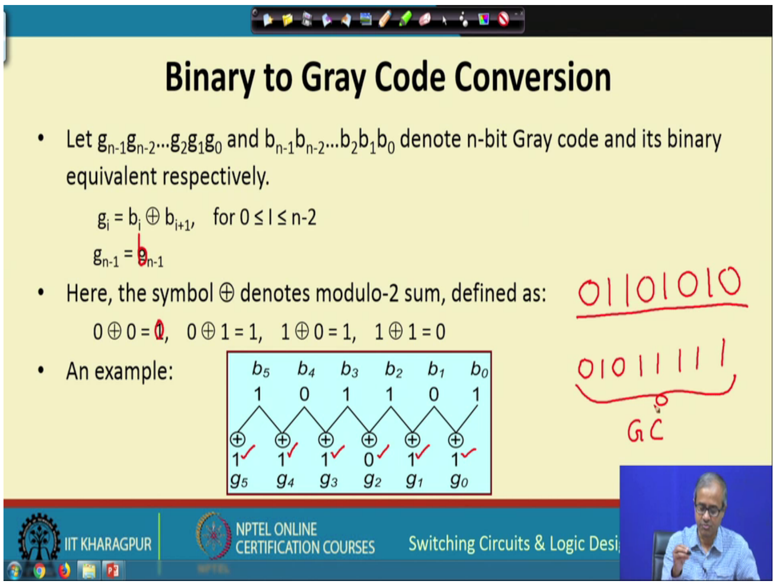
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Desi

First let us look at how we can convert a binary number to gray code. Let us consider in general that we have n bit representations, suppose my binary number is represented like this b 0, b 1 up to b n minus 1 and, the corresponding gray code number will be g 0, g 1 up to g n minus 1. T he rule is fairly simple the most significant bit straight away copied this should be b read it as b right.

So, g n minus 1 equal to b n minus 1, the most significant bit is straightaway copied and for all other bits, you take the modulo 2 sum, this is called modulo 2 sum of the corresponding binary bit. And the next more most significant binary bit bi plus 1 for all other values of I this is I. This is the rule and this modulo 2 sum is defined as follows 0 modulo sum 0 is sorry this is 0 this is 0 just a second.

So, what I said is that this is actually b and this is actually 0 and 0 modulo sum 1 is 1 1 modulo sum 0 is 1 1 modulo sum 1 is 0. Now, when you convert there is an example let us say 1 0 1 is 1 0 1 is a binary number, the most significant part 1 you simply copy here. The next bit of the gray you take as the as modulo 2 sum of the corresponding binary bit and the previous one, this is 0 and 1 0 and 1 model 2 sum is 1 it becomes 1. The next bit is 1 modulo sum 0 this is also 1 1, next bit is 1 and 1 0 next bit is 0 and 1 1, next bit is 1 and 0 1 let us take another example.

(Refer Slide Time: 25:37)



## Binary to Gray Code Conversion

- Let $g_{n-1}g_{n-2}...g_2g_1g_0$ and $b_{n-1}b_{n-2}...b_2b_1b_0$ denote n-bit Gray code and its binary equivalent respectively.

  $g_i = b_i \oplus b_{i+1}$, for $0 \le I \le n-2$

  $g_{n-1} = \overset{b}{g}_{n-1}$

- Here, the symbol $\oplus$ denotes modulo-2 sum, defined as:

  $0 \oplus 0 = \overset{0}{1}$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$

- An example:

| $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 |
| $\oplus$ | $\oplus$ | $\oplus$ | $\oplus$ | $\oplus$ | $\oplus$ |
| 1 ✓ | 1 ✓ | 1 ✓ | 0 ✓ | 1 ✓ | 1 ✓ |
| $g_5$ | $g_4$ | $g_3$ | $g_2$ | $g_1$ | $g_0$ |

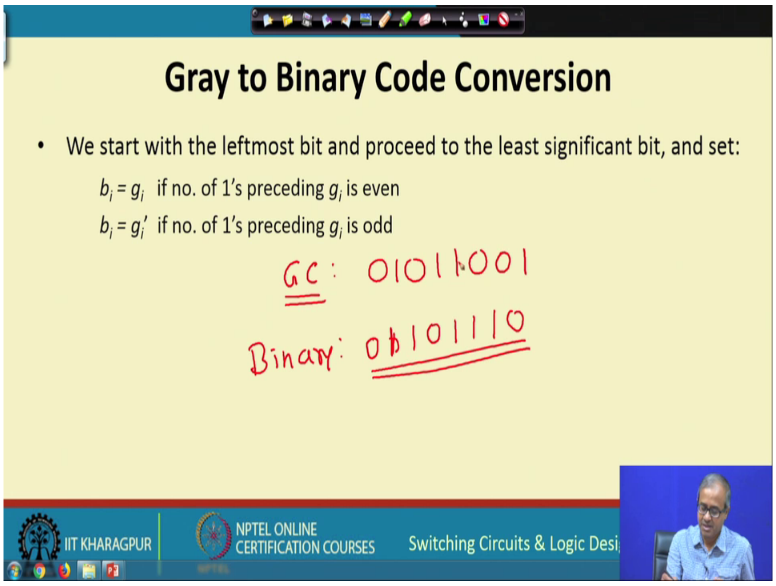
$\underline{01101010}$

$\underbrace{01011111}_{\overset{0}{GC}}$

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

Suppose I have a binary number 0 1 1 0 1 0 1 0 it is an 8 bit number. So, let us follow the same rule the first bit is straight away copied 0, the next bit will be the modulo 2 sum of this bit to previous bit 1 and 0 this is 1, next 1 is 1 and 1 which is 0, next 1 is 0 and 1 which is 1 1 and 0 which is 1 0 and 1 is 1 1 and 0 is 1 0 and 1 is 1. So, this is the corresponding gray code representation of this binary number. So, given a binary number you can very easily convert it into gray code right ok.

(Refer Slide Time: 26:30)



## Gray to Binary Code Conversion

- We start with the leftmost bit and proceed to the least significant bit, and set:

  $b_i = g_i$ if no. of 1's preceding $g_i$ is even

  $b_i = g_i'$ if no. of 1's preceding $g_i$ is odd

$\underline{\underline{GC}}$: 01011001

Binary: $\underline{\underline{01101110}}$

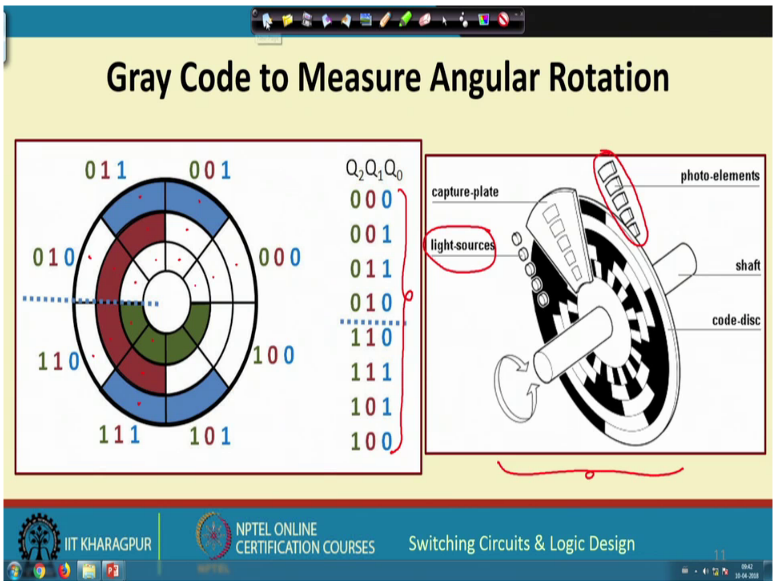
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Desi

Let us look at the reverse gray code to binary number conversion well, the rule is fairly simple as you can see. So, we start to the left most bit position and move towards the right. And the rule is the binary bit is said to be the same as the gray code bit, if the number of ones proceeding g i is even and, we set it to the compliment this g i dash indicates it is are the compliment ok. The compliment of that compliment if the number of ones proceeding g i is odd ok.

Let us take an example suppose I have a gray code, let us say 0 1 0 1 1 0. Let us say let us take an 8 bit representation, 8 bit gray code and I want to convert it to binary. So, when you move to the first bit there is no preceding bits. So, number of ones preceding this is 0 which is even. So, I simply copied b i equal to g i this is 0 bits copied I move to the next 1 number of preceding 1 is still 0 even. So, I sorry I straight away copied this will be 1, then I come to 0 the number of preceding ones is 1 which is odd.

So, it will get complimented 0 will become 1, 1 number of preceding 1 is again 1, which is odd this will get complimented, 1 number of preceding 1 is 1 and 2 it is even so, it will be straight away copied 0 preceding 1 2 3 odd. So, it will get complimented 0 again 1 2 3 odd it will be complimented and 1, 1 2 3 odd this will also bit complemented this will be the equivalent binary number for this. So, you see conversion is not that difficult. So, if you know the rule you can straightaway convert it like this fine ok.

(Refer Slide Time: 28:47)



## Gray Code to Measure Angular Rotation

Now, let us talk about that example that we had talking about; that means, why we say that grey code can reduce errors well. Let us take that example of a rotation of a wheel look at this diagram, there is a wheel let us say it is a the wheel of a car, or any kind of a mechanism, which is rotating and you are trying to measure or monitor the degree of rotation of the wheel. So, what you do on the wheel on one place we drill some holes and on one side there are some light sources and on the other side there are some light sensors. So, whenever those holes come under the light.

So, if there is a hole light will pass and the sensor will be sensing it and, if there is no hole there will be no light and the sensor will not find any light ok. So, it is something like this, there are some light sources as you can see on one side of the wheel, they are arranged like this five of them are shown. And on the other side there are some sensors these are some light sensors.

And the wheel we are coding like, if you see some part of the wheel is shown as white, some part of the wheel is shown as colored. The idea is colored means they are opaque they are solid, light cannot pass through it. And white means they are transparent may be they are made of glass light can pass through it. And the way we have made this coloring and whitening base is based on the gray code let us take an example of 3 bits this is at the 3 bit gray code representation.

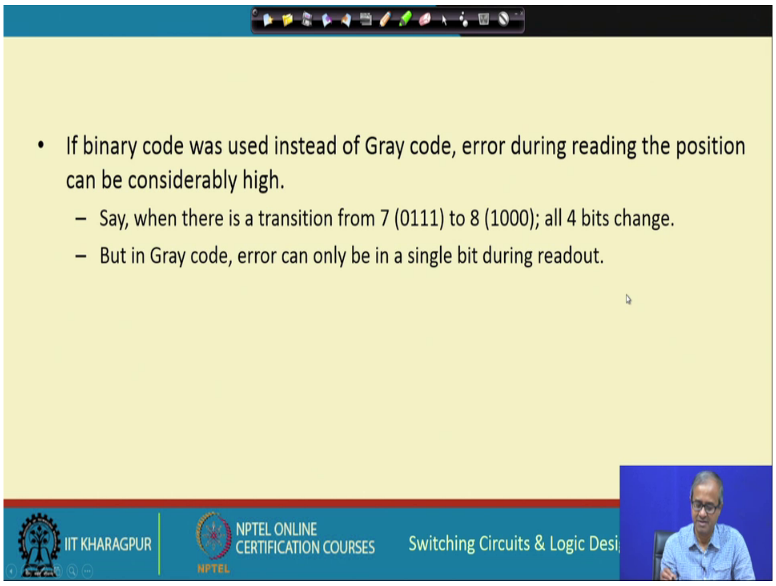
So, the successive numbers you have written down 0 0 0, 0 0 1, 0 1 0, 0 1 0 and so, on. And if it is 0 0 0 you see we have made it all white, white, white. So, white means 0 solid means 1 0 0 1 is 0 0 1 1 is solid, 0 1 1, 0 1 1, 0 1 0, 0 1 0, 1 1 0, 1 1 0, 1 1 1, 1 1 1 and so, on. Now, because it is gray code so, across one segment and the next exactly one of the bits are change in state.

Now, we imagine if we did the same thing with respect to binary, think of the binary numbers. Let us say in 4 bit consider the number 7, 0 1 1 1 next number is 8 1 0 0 0. So, all the 4 bits are changing. So, now, imagine if the wheel is just between 7 and 8 just it is moving. So, some of the lights may be passing some not passing due to some alignment, there can be a little error some of them is sensing some of them is not sensing, in the worst case there can be an error of so, many bits because, all 4 bits are changing, but in gray code because only one bit is changing between 1 segment to the next.

So, during the transition even if there is small error only that 1 bit can be either 0 and 1 other bits will be very stable. So, the error can be at most in a single bit in gray code during this rotation sensing right. This is the main idea behind using this kind of gray coding in, this is just one example I have sited there are many other examples, like here I am just mentioning one example, that in the design of circuits we shall see later.

When you talk about sequential circuits, we shall see that if we use some kind of gray code encoding, then between successive states when some signal is coming, if we ensure that only one of the outputs are changing state at a given time, then it can provide us with the mechanism with which you can reduce the amount of power dissipation, because power dissipation or energy dissipation is very important nowadays, because most of a devices are working on batteries. So, minimizing power is very important and this gray code encoding also helps in that all right.

(Refer Slide Time: 33:20)



- If binary code was used instead of Gray code, error during reading the position can be considerably high.
  - Say, when there is a transition from 7 (0111) to 8 (1000); all 4 bits change.
  - But in Gray code, error can only be in a single bit during readout.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
Switching Circuits & Logic Desi

So, this is what he mentioned binary code was used instead of gray code multiple bit position might be changing across segments and so the error could have been higher, but in gray code you can have only in a single bit position during reading.

So, with this we come to the end of this lecture, in the lecture number 6 we shall be talking about something else. We shall be talking about error detection and correction say in binary we have represent some numbers they might be stored, or they might be communicated from one place to the other. There are chances because of various reason that some bits may get flipped or erroneous; so how to detect and correct them that will be the topic of a next lecture.

Thank you.