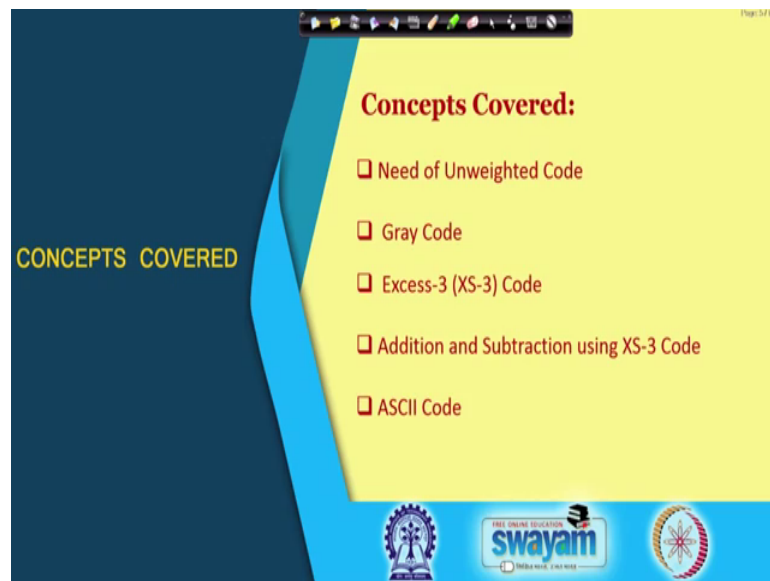


Digital Electronic Circuits
Prof. Goutam Saha
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 28
Unweighted Code

Hello everybody. In today's class we shall discuss Unweighted Code.

(Refer Slide Time: 00:25)



So, we have seen binary coded number and we have seen that the weights associated with binary code is 1, 2, 4, 8, 16, so on and so forth before the binary point and half, 1 upon 4, 1 upon 8 etcetera after the binary point. We shall, we see these are we shall look into what is the need of having unweighted code. Then we shall discuss two important such codes gray code and excess 3 code and also we shall discuss ASCII code.

(Refer Slide Time: 01:01)

Issue with Binary Code

B_3	B_2	B_1	B_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0

2 digits change

3 digits change

2}

3}

4}

Zone No. (Binary coded)	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111

B_3, B_2, B_1 :
(If B_1 change is slower than B_0)

Possibility of decoding a wrong direction of movement

$d_n \dots d_1 d_0, d_1 d_2 \dots d_m = d_n \times r^n + \dots + d_1 \times r^1 + d_0 \times r^0$
 $+ d_{-1} \times r^{-1} + d_{-2} \times r^{-2} + \dots + d_{-m} \times r^{-m}$

1010 in 8421 code = $1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$

8421 : Weights associated with position

Note: 1010 in 2421 code = $1 \times 2 + 0 \times 4 + 1 \times 2 + 0 \times 1$

So, if you remember the quick the visit of the binary code. So, for visit numbers B 3, B 2, B 1 and B naught ok. So, with that with the binary code this formula you remember the corresponding values. So, for any number with base or radix r so this is the digits and this is the corresponding value. So, if we consider this number 1010 and if it is 8421 code normal you know 2 to the power 0 etcetera this way, the standard way of associating weight in the value will be for this one 8 will come, and for this one 2 will come, ok. So, 8 plus 2, 10 will be there.

So, there can be other kind of weighted code where this is not exactly like this r to the power a, r to the power n, in the way it is increasing the integer power is increasing based on the position, ok. So, there can be other way of putting weights. For example, if it is a 2421 code. So, the first position is associated with the digit will be having a code weight of 1, second position 2, third position 4, and the fourth position is having a weight of 2. So, that is for 2421 code. So, that is also a weighted code because weight associated these 2. It is not 2 to the power 3 8 that we see for 8421 code, ok.

And 1010 if we are using 2421 code, what will be the value? So, the one in the; this place the 3 place will be having value 2, and then there is a 0, then there is a 2 for this particular one and then there is a 0. So, 2 plus 2 the value will be corresponding this the equivalent will be that of 4 for this 2421 code in 1010 representation, ok. This part we understand, I mean regarding weighted code.

Now, what could be the issue with weighted code for which we might need to look into different form of unweighted code, ok. So, one of the issue that you can see in this particular case. So, when we are changing the number from 0 to 1 to 2, so this is your 1, this is your 2, then 3, 4 so on and so forth the decimal equivalent the consecutive increase in the number. So, from 1 to 2 you can see that 2 digits are changing, 0 1 it is becoming 1 0, ok.

So, from 3 to 4 you can see 3 digits are changing 1 1 is becoming 1 0 0, ok. So, what could be issue with that? So, this is the way the weighted code you know is 4 2 1 code the why we have seen it is you know is there. So, let us see one possible you know issue with such kind of now coding.

Consider there is a conveyer belt B 1 that you see here, over which some material is there and the conveyer belt is moving in this direction, right direction as has been shown with the arrow, ok. So, it is moving from zone 1 to zone 2, for the where some activities are taking place and the conveyer is goes is suppose to go in this direction, right. And if you are sending a binary code of the zone where the this particular material is lying at a given time to a control unit, through a mode of you know digital transmission of data. So, from zone 1 0 0 1 it is going to 0 1 0, that is the way it will be kind of you know shown in that control room, right.

Now, from 001 to 010 since these two digits are changing, it so happens because of various you know delay our elements associated with the bit positions and the changing of the value the coding of the value and its transmission, there this 1 is changing, when this 0 has not yet changed. So, 1 this 1 is changing faster than the 0 that is suppose to change to 1. They are not exactly you know changing in the same time, and after that it is going to 010 of course, after some delay.

So, at some point of time the 001, after that 000 will come and that will make an in interpretation that the it has moved from zone 1 to zone 0, when it is suppose to go to from zone 1 to zone 2, is not it. So, basically the conveyer belt is moving in the it was duration. So, that might cause or that might cause some alarm or emergency some issues, with this you know particular thing. So, if you would prefer in such a scenario for consecutive numbers that only 1-bit you know if it is possible to make change, ok. So, that gives rise to what we call gray code.

(Refer Slide Time: 06:33)

Gray Code

$B_3 B_2 B_1 B_0$	$G_3 G_2 G_1 G_0$
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1
1 0 1 1	1 1 1 0

0000
 0001
 0011
 0010
 0110
 0111
 0101
 0100
 1100
 1101
 1111
 1110
 ...

- Gray Code is an unweighted code with unit distance i.e. only one position changes between two successive positions.
- It is also called reflected code as by reflecting $(n-1)$ -bit gray code, n -bit gray code can be obtained.

So, in gray code two consecutive numbers do differ by only 1. So, the distance between two numbers in terms of the positions in the is only 1, ok. So, if we look at example of some such some you know gray code from the initial values are 0 to 10, the way we had seen for binary code. So, these are the corresponding equivalent 0, 1, 2, 3, the consecutive number we already know 5, 6, 7, 8, 9 and this is 10, right.

Now, corresponding gray code if you see 0 is 0, right there is no ambiguity about that, 1 is 0 0 0 1 absolutely no issue because only 1-bit is changing, right. Now, here when it comes to it is 1 0, 2 bits are changing here it is 1 1, ok. From 2 to 3 this is 1 0 this is 1 1, from this 1 1 now only 1-bit can change. So, this 1 1 is now 1 0, right. So, is there any rule by which we can get you know this number from you know binary to gray or you know some sort of way the number can be generated, not juristically.

So, one such method, one such method is through reflection. So, how is it that? How is it let us see. So, first 0 and 1 we have no issue, two numbers 0 and 1 is getting you know represented in binary as well as gray. So, this is this 0 1 you reflect, like this is a mirror, so this is 1, this one is getting reflected and this is a 0.

Now, some additional bits are required because the number you know the count we are trying to increase. So, you put 0 0, 0 in the first two and 1 in second; so, 0 0 0 1 1 1 and 1 0. So, that gives you 0 1 2 3. You can see over here you the first thing. So, first 4 number is obtained.

Now, to get next 4 numbers what we shall we do? Again, we will do the deflection. Now, we shall take this 2 digits together, so 0 0, 0 1, 1 1 and 1 0. So, if we reflect, so 1 0 is reflected here 1 1 is reflected here 0 1 is reflected here and 0 0 is reflected here, ok. So, again we shall put 0s here and 1s here. So, by which the third digit comes into play, so that will give us 8 such consecutive numbers.

And similarly, we can go for this 8 will be represented here, so that gets reflected that like 100 is reflected here 101 is reflected here and then we will put to place the 4th digit here first block 0 and second block one we shall get the consecutive numbers. So, this that is the way we can get it and we can see that the gray code is formed, ok. So, the I mean generalized at by reflecting n minus 1-bit gray code we can get in the gray code, ok. So, this is one way of getting the gray code, right.

So, gray code in 10 is 1010 this is 1111. Of course, you can understand that this is not a weighted code, ok. If you as try to at associate some weight a like 8421 2421 any other weight you see that it is it is not working, ok. But good thing about it is that the consecutive numbers are only increasing by 1-bit, so there is no ambiguity of you know one of the number one of the bit is changing slower than the other resulting in some you know interpretation of a different number coming in between, right.

(Refer Slide Time: 10:30)

Code Conversion

B_3, B_2, B_1, B_0	G_3, G_2, G_1, G_0
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1

Logic Diagram: Shows the conversion of binary bits B_3, B_2, B_1, B_0 to Gray code bits G_3, G_2, G_1, G_0 using XOR gates. The diagram shows $G_3 = B_3$ and $G_i = B_{i+1} \oplus B_i$ for $i < n$.

Formulas:

$$G_n = B_n$$

$$G_i = B_{i+1} \oplus B_i \text{ for } i < n$$

$$B_n = G_n$$

$$B_i = B_{i+1} \oplus G_i \text{ for } i < n$$

Handwritten note: $B_3 \oplus B_2$

So, if we have a situation where the input is binary digit and the output is gray code I mean that is what is required and then how we go about it, how we can get a the codes

converted and also vice versa. I mean we have gray code we want a binary code, right for some reason, for some you know application. So, to do that if we look at the you know the mapping the we can come up with a very simple relationship where the n th bit most significant bit, right, if it is a n you know digit conversion that we are looking at that mean the gray code of that at the B_n will be exactly the same, ok.

So, here if we take this 4-bit example; so, B_3 and G_3 , so G_3 will be same as B_3 , right. And for other values, G_i is B_i plus 1 exert with B_i , ok. So, what does it mean? So, B_3 straight away goes as G_3 , so G_2 , ok, so the next gray code is B_i plus 1 this is 2. So, B_3 exert with B_2 , ok. Then we shall get this one. B_1 is B_2 exert with B_1 I mean same way B_1 and as G_1 we get B_2 exert with B_1 , G_{naught} we will get B_1 exert with B_0 . So, if you just put it into the form of a you know bank of example you can get it in this manner. You can see that whether this is working for example, if you say first case, so all 0, right, so 0 comes over here, right, so the 0 0 0 0. So, say this is 0, 0 0 exert is 0, 0 0 exert is 0, 0 0 exert is 0 summation, right.

So, next one is 0 0 0 1, you can see 0 0 is 0. I mean 0 comes here this 0, the next one is 0 0 exert 0, next one is also 0 and 0 1 exert is 1, ok. Now, comes when there is a difference. So, bit binary code is 0 0 1 0, right. What is happening now? So, we get 0 0 1, let us see. So, 0 comes over as 0 directly 0 0 exert is 0, now 0 1 exert is 1 and 1 0 exert is 1, that we can that way you can verify rest of the numbers and we will see that such a simple circuit equal, as a good converter.

Now, for gray code to binary code conversion; so if you look at the reverse way the mapping is done we can see that G_3 can directly go as B_3 and G_2 second B like B_2 can be generated by exerting G_3 and G_2 and B_1 can be exe obtained by exerting B_2 the output that has been generated, using with G_1 , right an G_1 exert with G_{naught} you can get B_{naught} , ok. So, we can again put those values and can see this is the way simply you can get the binary code generated from the gray code, ok.

(Refer Slide Time: 14:13)

Excess-3 Code

Binary Code Decimal + 0011 = Excess-3 Code (XS-3)

Decimal	XS-3
0	0000 0011
1	0001 0100
2	0010 0101
3	0011 0110
4	0100 0111
5	0101 1000
6	0110 1001
7	0111 1010
8	1000 1011
9	1001 1100

Code Conversion:

Approach 1:

- BCD to XS-3: Represent each XS-3 bit, $X_i = F(B_3, B_2, B_1, B_0)$
- XS-3 to BCD: $B_i = F(X_3, X_2, X_1, X_0)$ (Consider 'don't care' for unused input combinations.)

Approach 2:

- BCD to XS-3: Add 0011
- XS-3 to BCD: Subtract 0011 (Consider 4-bit Adder-Subtractor)

Handwritten notes on the slide:

- For decimal 53: $53_{10} = (0110.1011)_{BCD} = (1000.0110)_{XS-3}$
- For decimal 6.9: $(6.9)_{10} = (0110.1001)_{BCD} = (1001.1100)_{XS-3}$

Now, we come to another kind of you know code which is called Excess-3 code. So, here we are trying to represent BCD numbers binary coded decimals. So, 0 to 9 these are the you know 10 numbers. So, in binary coded decimal we have 0000 to 1001, ok. So, in excess-3 code what are doing, this 0 to 9 is represented by 0011 to 1100; that means, whatever is the binary code add to that 0011 we get excess-3 code. Why? Why are you doing this? I mean what is the benefit out of it will be cleared very soon, ok, we shall discuss that part.

So, first we see that how it is represented; so this is the way it is represented. So, if we are representing a number 5 3 in BCD code it will be 0101 for 5 and 3 0011 is not it, but in excess-3 code this will be at 3 0011, so this is 1000 and this is 0110. So, you can see that 5 is 1000 and 3 is 0110, right 5 and 3, ok.

So, similarly 6.9 if you want to represent again 6 is here you can see 1001, right and 9 here is 1100, ok. So, this is the way I mean the representation is there and this is for many other cases we can see for example, if you want to represent 487. So, 4 is 0111 in excess-3 8 is 1011 and 7 is 1010. So, this is the way the number will be represented, ok. So, this is coming in computation with you know BCD code, and accordingly we shall see the benefit, in subsequent slides.

And regarding code conversion; so how to convert from one form to another? So, one approach could be the case where each of this X, XS-3 code. So, if I say if I represent

them as X₃, X₂, X₁ and X_{naught}, right we can form a form a have a truth table where this is X₃. What is the value of X₃ for different representation now? B₃, B₂, B₁ and B_{naught}, right. And the numbers that are not there like one 10 to 15, right, 1010 at the input site 1111, so those are the do not care cases, ok.

We do not care about what will be the value, right. So, with that we can have a you know minimization process Karnaugh Map QM or Boolean algebra by which we can get the value of X₃ then we can get X₂, and get X₁, and X_{naught} as a function of B₃, B₂, B₁ and B_{naught} and you can realize it. So, this is one way of you know getting it.

The other one could be just having a 4-bit adder, right, a 4-bit adder by which you can add and get it and the reverse process, reverse process means from XS-3 to binary. So, we shall have B₃, right. We shall see how it is it can be represented as a function of X₃, X₂, X₁ and X_{naught} for different values, again we shall consider do not care we get this (Refer Time: 17:58) and realized after minimization or we shall add a subtract by which are subtracting 0011 from the excess-3 count we shall get back the binary code, ok.

(Refer Slide Time: 18:20)

Addition of XS-3 Code

- Subtract 0011 from addition result if no carry is produced.

$$\begin{array}{r} (5)_{10} \rightarrow 1000 \text{ (XS-3)} \\ + (2)_{10} \rightarrow 0101 \text{ (XS-3)} \\ \hline (7)_{10} \rightarrow 1101 \text{ (XS-6)} \end{array}$$

$$\begin{array}{r} 1101 \text{ (XS-6)} \\ - 0011 \\ \hline 1010 \text{ (XS-3)} \end{array}$$

No carry up to (1111)_{XS-6} = (1001)₁₀ = (9)₁₀ = (1100)_{XS-3}

- Add 0011 to addition result if carry is produced.

$$\begin{array}{r} (5)_{10} \rightarrow 1000 \\ + (7)_{10} \rightarrow 1010 \\ \hline (12)_{10} \rightarrow 10010 \end{array}$$

$$\begin{array}{r} 0010 \\ + 0011 \\ \hline 0101 \text{ (XS-3)} \end{array}$$

(0)1(00) 0101 (XS-3)
1 2 (Decimal)

$$\begin{array}{r} 101 \\ + 1000 \\ \hline 1110 \\ - 0011 \\ \hline 1011 \\ + 0011 \\ \hline 1011 \\ 101 \text{ (XS-3)} \\ \hline 8 \text{ (Dec.)} \end{array}$$

So, the benefit we shall see as you will see over here, ok. So, compared to BCD arithmetic you shall see that excess-3 arithmetic is simpler, ok. So, let us see how the arithmetic is done. So, if you are looking at addition of say two numbers for which the no carry is produced, no carry is produced, right. What is the implication of it? So, let us

consider two numbers 5 and 2. So, corresponding representation is 5 is 1000 in XS-3 and 2 is 0101 in XS 3, because 3 get added with normal the binary coding, right. So, if you add if you add, so this is 1, this is 0, this is 1 and this is 1. So, no carry is produced here of course, no carry is produced, so we shall look at example where carry is produced.

So, no carry is produced, so this representation 1101, right this is a you know valid XS-3 code, right and this the valid you know number I mean valid number in the sense the number which is which does not produce any carry, right but within it has 2 additional 3s, 2 additional 3s that goes inside it, ok, one from here another from here compared to binary coding. So, basically what you get here is XS 6, the binary coded number plus 0011 coming here as 0011 coming from here.

So, if you want to get the XS-3 version of the result which is our intention, what we need to do? We need to subtract it subtract 3 from it, right because it is XS 3, if you subtract 3 we will get XS 3. So, the result will be if you subtract from here you get 1101 subtracted with by 3 we get 1010, ok. So, this 1010 is the XS-3 version of XS-3 of the number 7 that we see as corresponding decimal equivalent is that clear, right.

So, we no carry is produced the resulting number, resulting number is a valid XS 6 number, ok. To get XS-3 we have to subtract 3 from it and that is how it will it is to work, ok. And carry is not produced up to what? Up to 1111, when this result is there in XS 6. So, that is your in its binary it is 1001, and that is your 9 decimal that is 1100 in XS 3, right is it clear. And when it goes beyond 9 go, right and these addition becomes more than you know 1111. So, the carry it is produced, ok. So, that is the you know in the binary coded decimal cases more than 9 basically we are getting carried, right. So, if look at another example where carry is getting produced. So, 5 and 7, so this is 12. So, 5 is 1010 and 7 is 1010, I will reduce to 1000, 1010. So, if you add them up we get 1 and 0010, ok.

So, how do we go about it? So, in these case when carry is produced. So, this is 0010 you just fine with you know normal kind of cases, but XS-3 we have to add 3 with 8, you have to add 3 with it. So, 0101 will be the case here and for this one if you are using this or another BCD addition another XS-3 addition another. So, the digit is there. So, this is will go as a carry otherwise if it is the end of it then we shall put a put it in the 4-bit; that means, which is getting added with 3 basically, so that is one and the 0 0 will coming

will come over here, that is the XS-3 version XS-3 representation of 1, that is 0100, right. So, this is the way it is it be represented.

So, in what is he that in one, case one is getting added another sorry 3 is getting added in this case then 3 is getting subtracted over here. So, you just need to see whether the carry is present or not accordingly the number 3 which remains constant which that will be either getting added or we get subtracted. So, that is that is going to help us in addition subtraction add or subtracted circuit for XS-3 in comparison to BCD add or subtractor that that we have developed before, ok.

So, if you are looking for you know addition of 2 you know addition of numbers decimal numbers with 2 digits. So, these are the, this is an example, so 25 is added with 57, so 25, 2 and 5, and 5 and 7 they are represented in XS 3. So, this is the addition result, ok. So, in that addition result here carry one is generated, ok. So, when carry one is generated what we have to do? For this particular XS-3 digit, of the decimal number we have to add 3. So, that is what we will get 0101.

So, 0101 is the decimal representation of 2, ok. And when you add carry over here by which what we see the number is 1110 and no carry is produced here no carry is produced here. So, no carry is produced we shall follow this. So, 3 needs to be re subtracted if you subtract 3 from it you get 1011 and this 1011 is decimal equivalent of 8. So, final result is 82, ok.

(Refer Slide Time: 24:45)

XS-3 Adder Unit

XS-3 Adder Unit from 4-bit Adder

XS-3 Input 1: A_3, A_2, A_1, A_0
 XS-3 Input 2: B_3, B_2, B_1, B_0

7483

Carry from previous: C_3

Sum outputs: S_3, S_2, S_1, S_0

7483

Ignore: C_3

XS-3 Digit 1: S_3, S_2, S_1, S_0

XS-3 Digit 2 / Carry: C_3

Handwritten calculations:

$$\begin{array}{r} 1 \\ 0101 \ 1000 \ (25)_{10} \\ + 1000 \ 1010 \ + (57)_{10} \\ \hline 1110 \ 0010 \ (82)_{10} \\ - 0011 + 0011 \\ \hline 1011 \ 0101 \ (XS-3) \\ 8 \ 2 \ (Dec.) \end{array}$$

$$\begin{array}{r} 0011 \\ 1100 \\ + 1 \\ \hline 1101 - 25 \\ \hline 0011 \end{array}$$

So, XS-3 adder unit. So, XS-3 adder unit. So, the example that which was discussed is over here in the left-hand side. So, what we do? This is the basic adder 7483 and then these are the 2 inputs, right. If carry is generated carry is generated what is happening? This is another adder, ok. So, if carry is generated we shall add this is coming as 0 this output will be 0. So, 1 1 is getting added, ok, 1 1 is getting added and this is the carry which might go to the next stage and if it is the end of it then you see this is 0, this is if carry is generated this will be 1 and this is 0 and this is 0 carry is not there this is 0, ok.

And if no carry is generated, if no carry is generated then what happens? So, this will be 0, so this will be 1 and this is this one is permanently connected and then this is 0, right. So, 1101, so this is 1, this is 1 0 1. So, what is this? It is the 2s complement, 2s complement, this 1101 is 2s complement of 0011, ok. Remember that, right. So, 0011, 1s complement is 1100 plus 1 is 2s complement which is this. So, we do not need a carry in over here ladder because it is already there, ok, right. And then we do 2s complement addition we will get the subtraction that. So, this is the way we can get very simply the XS-3 adder circuit meant compared to if you remember BCD 3, BCD addition, ok.

(Refer Slide Time: 26:44)

XS-3 Subtractor Unit

Subtraction using 9's C:
Inverting bits of XS-3 code 9's C
In XS-3 directly obtained.

$(2)_{10} : (0101)_{XS-3}$
9's C of 2: $9 - 2 = 7$
 $(7)_{10} : (1010)_{XS-3}$

- Add 9's C of subtrahend with minuend
- If carry, result +ve, add carry, also add 0011 for XS-3
- If no carry, result -ve, subtract 3 for XS-3, invert its bits

The diagram shows two 7483 adders. The top adder takes the XS-3 Minuend (A₃ A₂ A₁ A₀) and the XS-3 Subtrahend (B₃ B₂ B₁ B₀) with a carry-in of 1. Its carry-out (C₃) is inverted and fed into the second adder. The second adder takes the inverted carry-out (C₃) and the XS-3 Minuend (A₃ A₂ A₁ A₀) with a carry-in of 1. Its carry-out (C₃) is labeled 'ignore'. The final output is the XS-3 Result (S₃ S₂ S₁ S₀).

Now, coming to XS-3 subtractor, one good thing about XS-3 number is that if you just invert the number, invert the bits immediate bits then you get nice complement, right. You do not need for BCD subtractor if you remember we discuss 10s complement generated circuit or if we had used 9s complement you know arithmetic we would have a

used 9s complement generator circuit. Here we do not need an elaborate or you now more complex these 9s complement or 10s complement generator circuit because the inverting the bits we can get the 9s complement of this, ok.

So, 9s complement subtraction we have already discussed before in an earlier class. So, we just do the implementation here. So, say 9s complement of 2, right if you look at, so 0 1 0 1 is in XS-3, right. 0 0 1 0 in binary XS-3 0 0 1 0 1, if you take 9s complement of these 2 it is 9 minus 2 which is 7, and 7 if you just look at it it is 1, if you 1010 in XS-3 and you just invert the bit here you get the 9s complement, is not it. And what is the process? If we remember for 9s complement you know best subtraction.

At 9s complement of B subtract with the minuend if carry result is positive add carry also add 0 0 1 1 for XS-3 because if you want XS-3 we need to add this one. If no carry result is negative and we have to subtract X 3 for XS-3 and we get invert the bit we get the result, ok. So, this particular law by which you know this shows the logic circuit by which you can get it, right.

So, this is showing this is showing when there is a subtraction. So, there is an inversion getting done, ok. And this circuit is familiar the BCD these XS-3 adder circuit and here what you can see if there is no carry if there is no carry result is negative, right and subtraction is done and inversion of the bit. So, inversion of the bits is happening to get you the final result, is it ok. So, compare it with other circuit we do not need as I said 10s complement generator another (Refer Time: 29:22). So, this is why XS-3 is useful and the of course, it is an unwanted code.

(Refer Slide Time: 29:29)

ASCII Code

ASCII: American Standard Code for Information Interchange; standardized for computer hardware

X_3, X_2, X_1, X_0	010	011	X_6, X_5, X_4	101	110	111
0000	SP	0	@	P		p
0001	1	1	A	Q		q
0010	"	2	B	R	a	r
0011	#	3	C	S	b	s
0100	\$	4	D	T	c	t
0101	%	5	E	U	d	u
0110	&	6	F	V	e	v
0111	'	7	G	W	f	w
1000	(8	H	X	g	x
1001)	9	I	Y	h	y
1010	*	:	J	Z	i	z
1011	+	;	K		j	
1100	,	<	L		k	
1101	-	=	M		l	
1110	.	>	N		m	
1111	/	?	O		n	

7 bits: 128 Codes
Includes control codes for peripherals and printable characters

EBCDIC (Extended Binary Coded Decimal Interchange Code) was introduced for IBM devices.

A: 1000001 a: 1100001
B: 1000010 b: 1100010
C: 1000011 c: 1100011
0: 0110000 =: 0111101
1: 0110001 :: 0111110
2: 0110010 :: 0101110

0000000: Null Character
0000001: Start of Heading
0001101: Carriage Return

And finally, this is ASCII code where whatever we are working in the computer or the computer is talking to a printer and all. So, in that case the control code control values, control codes and alpha numeric codes the print of the characters etc these are all generating ASCII code for communication from one place to another. And ASCII code the full form of it is American Standard Code for Information Interchange, right and this has been standardized for computer hardware, ok.

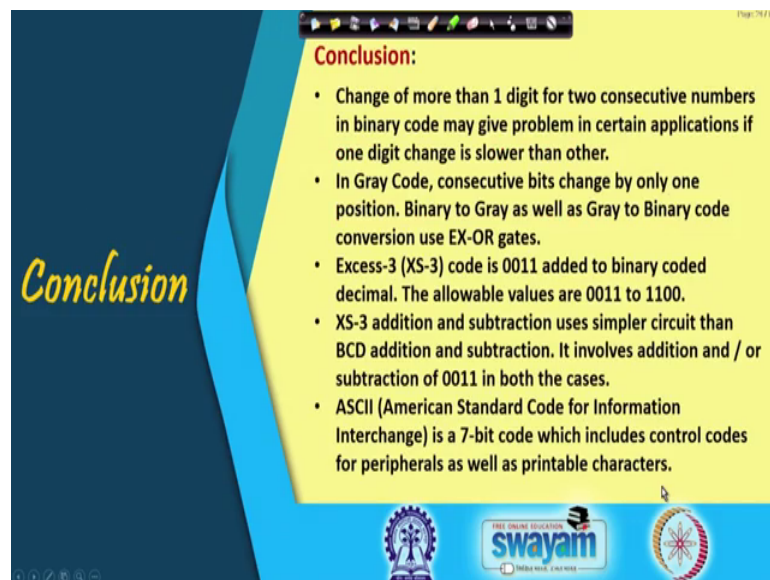
There was some other code which EBCDIC, extended binary coded decimal interchange code which IBM introduced for its device, but this ASCII code is now you will see it is prevalent and you know people are using it. And it is a 7-bit code, so 128 possible codes are there, right and if you just look at how things get represented say one of the character, we type say is capital A, right. So, capital A you can see the in the example. So, these are the more significant bit bits X 6, X 5, X 4 and these are the your significant bits X 3, X 2, X 1 and X naught, right. So, 1 0 0 this column and this row if you see 0 0 0 1 coming from here. It is how capital a is represented.

And how small a is represented? We will see that this is small a. So, between these 2 only one value is changing, this 0 is changing to 1 and rest of the values are remaining same. So, small a is 1 1 0 this instead of 1 0 0, this is 1 1 0, it is from the values are remaining same that is your small a. Similarly, between capital B and small b we see that

only this place the value is changed 6th place, right, 7th bit this is the place it is changing.

Now, how 0? If you type 0, how 0 is represented? So, 0 you can see over here this is 0. So, 0 1 1 and four 0s, 0 1 1 and four 0s, this is the one, right. How 1 is represented this is 1. So, 0 1 1, 0 three 0s and 1, ok. So, these are the different things. So, out of this say carry the detail. So, any place center and all; so then what is the corresponding ASCII code based on which some action is taken? So, carriage return is 0 0 0 1 1 0 1, this list is not exhausting, right or 120 options are not given. These are mostly these are printable characters that you can see, ok.

(Refer Slide Time: 32:16)



Conclusion:

- Change of more than 1 digit for two consecutive numbers in binary code may give problem in certain applications if one digit change is slower than other.
- In Gray Code, consecutive bits change by only one position. Binary to Gray as well as Gray to Binary code conversion use EX-OR gates.
- Excess-3 (XS-3) code is 0011 added to binary coded decimal. The allowable values are 0011 to 1100.
- XS-3 addition and subtraction uses simpler circuit than BCD addition and subtraction. It involves addition and / or subtraction of 0011 in both the cases.
- ASCII (American Standard Code for Information Interchange) is a 7-bit code which includes control codes for peripherals as well as printable characters.

The slide also features a 'swayam' logo and a 'FREE ONLINE EDUCATION' banner at the bottom.

So, with this we come to the conclusion. The change of more than one digit for two consecutive numbers in binary code we give problem in certain applications, if one digit change is slower than the other for which we discussed gray code and we have binary to gray and gray to binary conversion using Ex-OR gates.

Excess-3 code is the one where 0011 is added to the binary coded decimal and if XS-3 addition and subtraction uses simpler circuit than BCD add or subtract the circuit and it involves addition or subtraction of 0011 that is 3 in both the cases. And ASCII character, ASCII code is a 7-bit code by which control codes for peripherals and printable characters can be generated and exchanged.

Thank you.