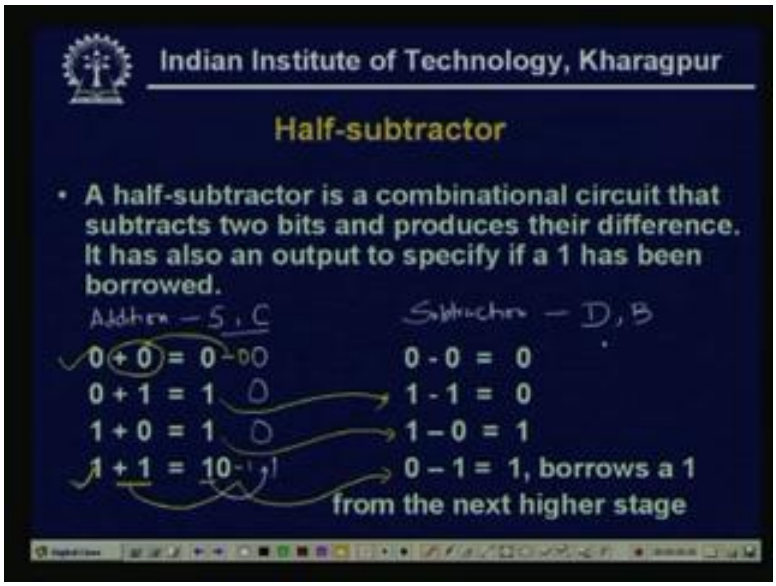
Digital System Design
Prof. D.Roychoudhury
Department of Computer Science and Engineering
Indian Institute of Technology, Khargpur

Lecture - 14
Design of Subtractor Circuits

In the last class, we have read how to design the different of adders, mainly the half adders and full adders and the different type realization using a variety of gates. Today, we will see how we can Design the subtractor Circuit.

(Refer Slide Time: 01:14)



Now, a half subtractor is a combination circuit; that subtracts two bits and produces the difference. It has also an output to specify, if a one has been borrowed, so the similarity with the adders is that, in case of addition, it was for addition, it was the carry signal, for addition one sum and one carry as been generated. For subtraction, that sum replaced by the difference, because it computes the difference and the carry concept is actually they borrow.

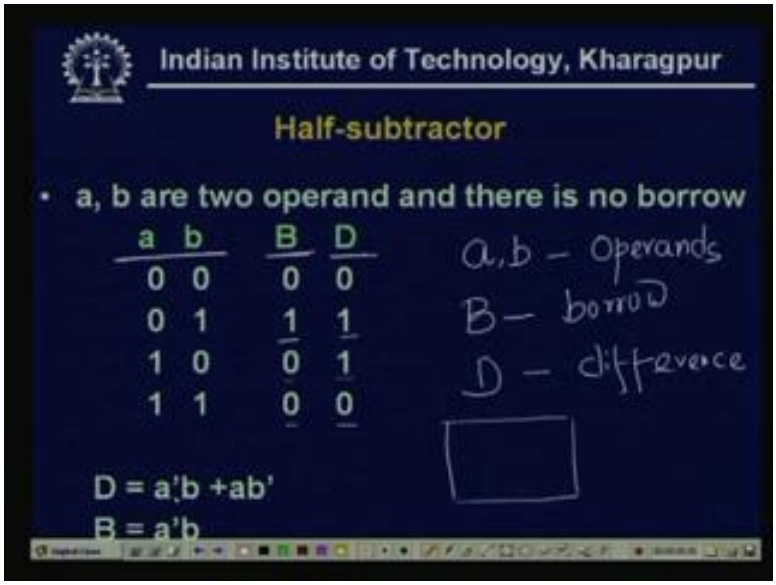
So, we denote these as a B and the difference is D, now how the operator operates on the two operands. So, again first we see the addition operation, see here that 0 plus 0 equal to 0 and the carry is 0, 0 plus 1 is 1, carry is 0, 1 plus 0 is 1 carry 0, 1 plus 1 is actually 1 0. So, sum is 0 and carry is 1, now see if I want change this operation addition as subtraction.

Then, what we can do the 0 plus 0 equal to 0, so as if 1 operand from here, say this 0, I can shift here, that means, say if I want to shift this 0 here, means this would be 0 minus 0. So, what will happen say for this addition, if this plus 0, shifts on the right side, then it becomes minus 0 and left hand side will be the result; that means, this is a 0, so the that thing we have expressed in the right side.

Now, for the same combination of the two operands, what we have done, that 0 plus 0 is 0, so 0 minus 0 is 0, again 0 plus 1 is 1, so here we can tell this see that 1 minus 1 is 0. Similarly, 1 plus 0 is 1; that means, 1 minus 0 is 1 and 1 plus 1 is 1 0, means the sum is 0, so sum is 0, so 0 minus 1. This plus 1 will go here, that will be minus 1, so that has been retain here 0 minus 1 and the result will be this 1, so 0 minus 1 this 1.

Now, what happens with this carry, what is the effect of this carry, so if 0 from 0, I am subtracting 1, result is also 1 provided that it has borrowed a 1 from the next higher stage. So, 0 minus 1 is 1 borrow 1, so in this way, we can define the subtraction operation from the addition operation itself.

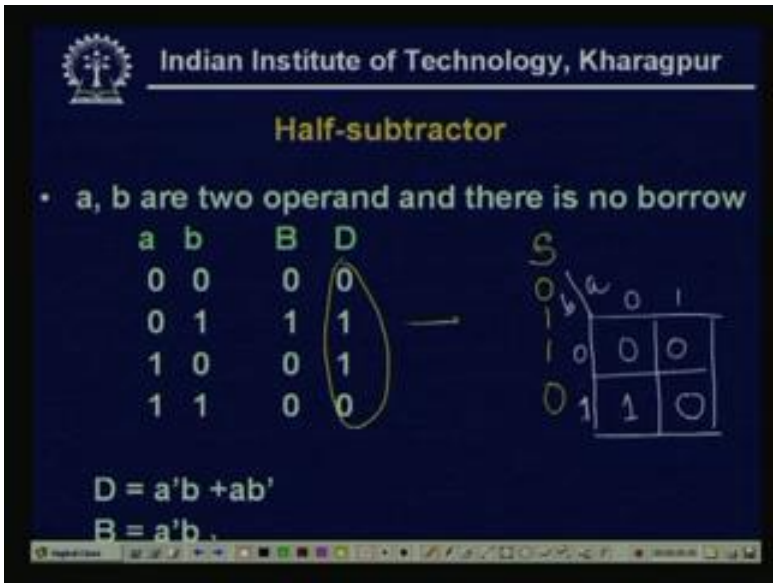
(Refer Slide Time: 05:29)



So, just now if I write in the truth table form, that mean a b are the two operands, these are the two operands B is the borrow and D difference. So, a b are the operands B is borrow and D difference, that we want to compute. Now, 0 0 borrow is 0, difference is 0, 0 1, 0 minus 1, this is the situation that, when borrow is 1 difference is 1, 1 minus 0 differences 1, borrow 0, 1 minus 1 difference is 0 borrow 0.

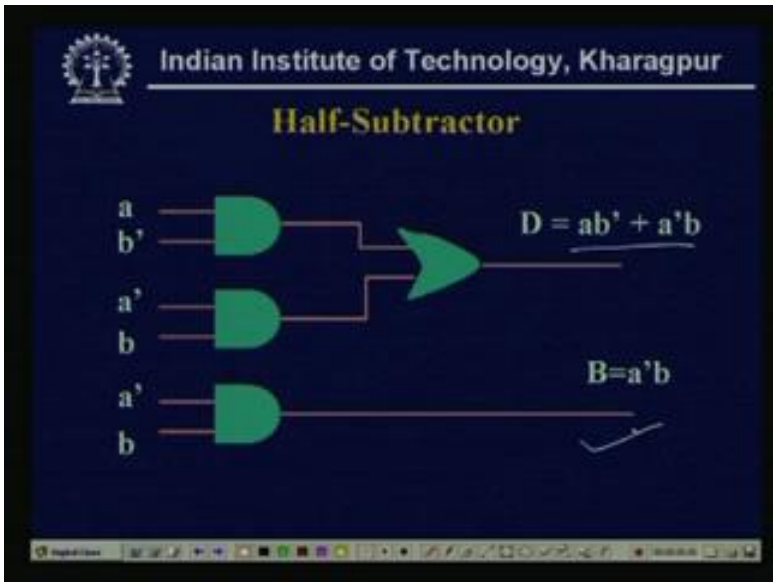
So, now if this truth table, we can express or we can draw the Karnaugh map, for this 1 for B and 1 for D. Then, will get the expression for D and B and in this way we can get D equal to a dash b plus a b dash.

(Refer Slide Time: 07:11)



Because, now just we can notice that, this D, D value is same as that of our sum of the addition operation, for the sum also it is 0 1 1 0. So, if this already we have seen, the last class, that this is nothing but a EXOR two input EXOR a dash b plus a b dash. Now, the borrow is 0 1 0 0, so if we draw the Karnaugh map, then if this is a b, 0 1, 0 1. Then, only for 0 case; that means, a is 0, b is 1, there is only 1 and all are zeros, so this minterm is a dash b see here, that this is my borrow is a dash b.

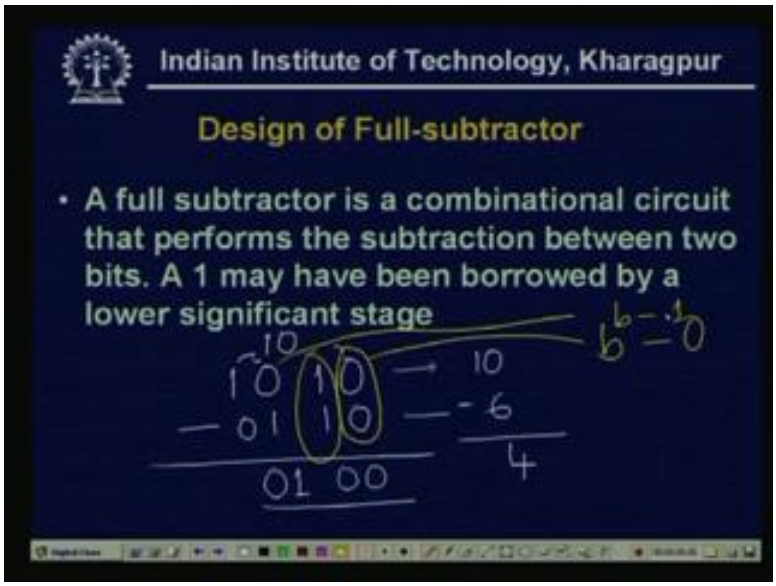
(Refer Slide Time: 08:15)



So, half subtractor circuit, again there are there will be two output, one for the difference, one for borrow, already we have seen the difference is same, at the carry of addition operation, that is two input XOR. And, that is AND, OR realization, then a b dash and this is a dash b, the 2, two AND gate and then the output of the AND gate is fade to a two input OR gate. So, this gives you the difference a b dash plus a dash b and the borrow is a dash b.

So, again it is a two input AND gate, only one input is a dash or another input is b, so this is totally similar with my half adder. So, half adder and half subtractor is almost same, only the expression of borrow differs from the expression of the carry, but the sum and the difference is totally same.

(Refer Slide Time: 09:29)

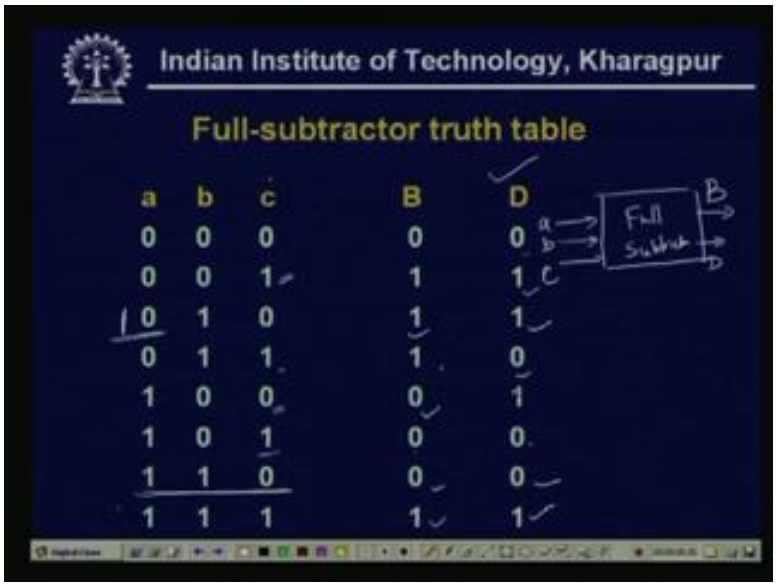


Now, design of a full subtractor, so just like the addition operation or the full adder circuit. Now, we can define full subtractor as a combinational circuit; that performs the subtraction between two bits, A 1 may have been borrowed by a lower significant stage. So, mainly the half subtractor for the two bit, mainly the subtraction is being done between two bits. So, once difference is generated and one borrow is generated, but if it is a n bit subtractor, then these we cannot neglect this borrow.

So, this one borrow, may have been borrowed by a lower significant stage, means when just like the full adder circuit, say if we take 1 0 1 0 and we want to subtract say 0 1 1 0. Then, say 0 minus 0 is 0 1 minus 1 is 0, but 0 minus 1, say here actually 1 0 as if we have borrowed a 1 here, this 0 become 1 0. So, 1 0 minus 1, we know from the addition operation this 1, so borrowed 1, then this 1, so 0 1 0 0.

So, if I take the decimal equivalent, we see that 1 0 1 0 is nothing but 10, this is 1 1 0, means 6 and if I take the subtraction, then it will be 4, which is nothing but our 0 1 0 0. See here, nothing has been borrowed for this two bits, so borrow bits are 0, for this two bits, borrow bits are 0, b is 0, but this 1, but third 1, the borrow is 1. So, this is a concept of full subtractor, again which is same or similar to our full adder only borrow is replaced or borrow is replaced by carry is replaced borrow, whatever you call.

(Refer Slide Time: 12:21)



Now, again if the input output relationship, we have seen that here also, there are two outputs 1 is the borrow output and 1 is the difference. So, now as borrow is considered; that means, here this should by this carry actually this should be my borrow b and that is the way borrow. Because a b two operands, we have taken, so we have retained as if this c is borrow.

So, for all possible now there are three inputs, so what we can draw, see this is my full subtractor circuit there will be three input, a b the two operands. And other is borrow denoted by c again and one is the next borrow generated and other is the difference of these two bits. So, this is my full subtractor circuit, now for all possible combinations, we have retained the B and D values.

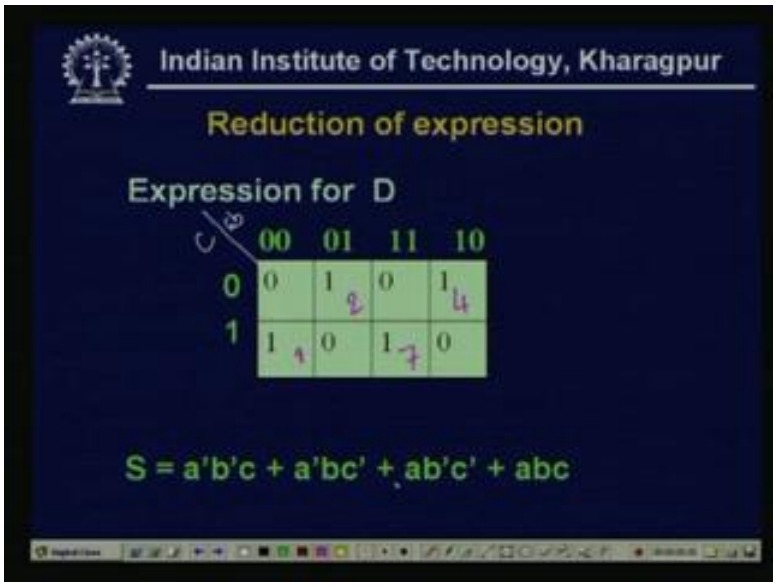
So, first we consider the difference, see that if a b is 0 0, already we know that 0 minus 0 is 0, then see 0 minus 0, but this borrowed 1, 1 borrow is there. So, this should be a difference, should be a 1. Similarly, 0 minus 1, this should be again that difference is 1, borrow is 1. Because, already we have seen that if it is 0 minus 1, as if from the highest stage, 1 is borrowed and actually it is 1 0.

Now, it is 0 minus 1 and there is 1 borrow also, so here also, this 1 minus 1 actually, this 0 and borrow 1, simple 1 minus 0, because that, there is no previous borrow, so this is 1 borrow 0, again here 1 minus 0, borrow 1. So, that means, actually 1 minus 1, so

difference is 0, borrow is 0. Now, 1 minus 1 borrows 0, so simple all are 0, borrow 0 as well as difference 0.

Here 1 minus 1, but 1 borrow is there, it was 0, just the previous case as the borrow is 0, so 1 minus 1 we know that difference is 0 borrow is 0. But, here 1 minus 1 as 1, borrow is there, previous borrow is there, that is why the difference will be 1 and borrow will also be 1. So, we can write the input output relationship of the full subtractor.

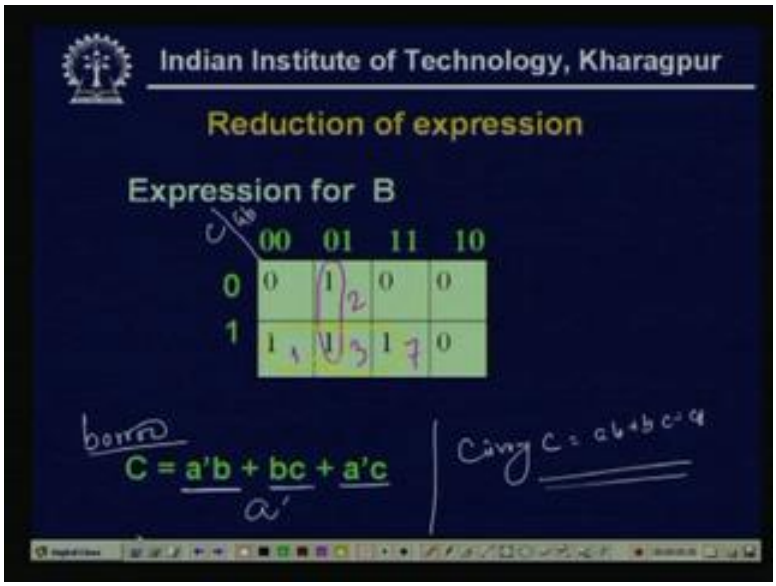
(Refer Slide Time: 16:14)



Now, again in the same way, will draw 2, Karnaugh map for two output, one for the difference d and another for the borrow b. So, when we write the expression for D, so again, if we see that for D, there are the number of 1's or the minterm exist or for 0 0 1 means for decimal 1, then 2, then 4 and then 7. So, 1, 2, 4, 7 the decimal, the minterm will be there, now see if we draw that, there are three inputs.

So, these are a b, this is c, so 0 0 1, this is my 1, then this is 0 1 0, this is 2, this is 1 0 0 4 and this is 7. So, this four minterm exist in the expression and these are nothing but a dash b dash c plus a dash b c dash plus a b dash c dash plus a b c, which is same as set up our full adder circuit.

(Refer Slide Time: 17:58)

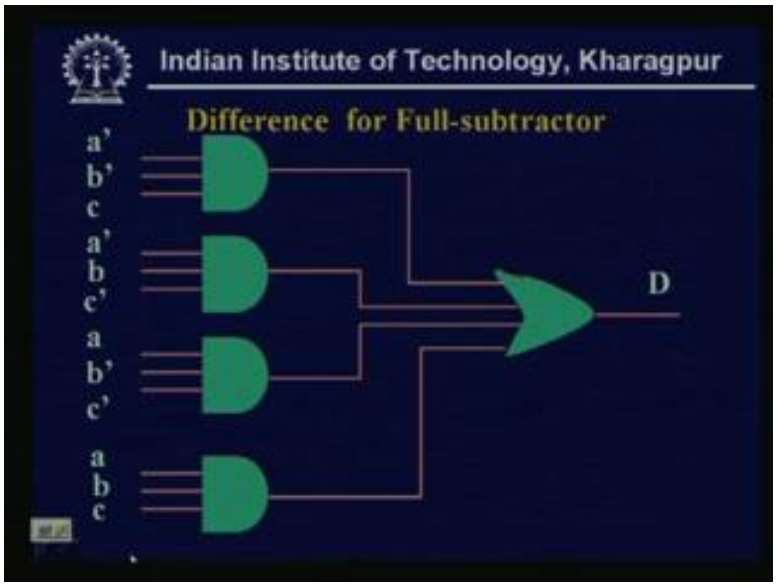


Now, what will be the expression for B, borrow, again if we see borrow, the minterm exist for 0 0 1, means 1, then 2, then 3 and for 7, so minterm exist for 1, 2, 3, 7. Now, we see for this is again, if we put this is my 1, 2, 3, 7. So, only these four terms, now see here then, we can form some couple, these will be 1 couple. For this, it will be a dash b, if we take a b and c, so this will be my a dash b, this term as been, this for this couple.

Now, for this couple, it will be a dash c, now for this couple, so this will be b c, so a dash b plus b c plus a dash c, there a similarity between the carry of the full adder circuit and borrow of the full subtractor circuit. Only the minterms are slightly different, there it was for at the full adder, that carry c was a b plus b c plus c a. So, again that, it was a sum of product and there are 3 AND gets and needed and 1. three input OR gates.

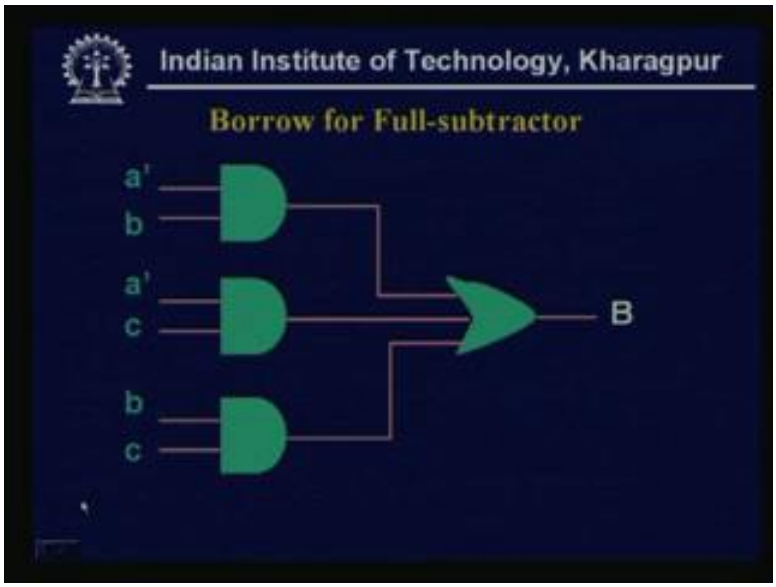
Here also the same number of gates are needed, totally it similar type of structure or similar type expression only here one complemented variable as the input, which my a dash. Here, there is no complement variable exist this is the only difference between the full adder and the full subtractor circuit.

(Refer Slide Time: 21:01)



So, if we see that the full subtractor, the difference for full subtractor, then it is totally is same as that of the full adder circuit. Because, that expression same, so already we have discuss this a 4, three input AND gate and 1, four input OR gate is needed.

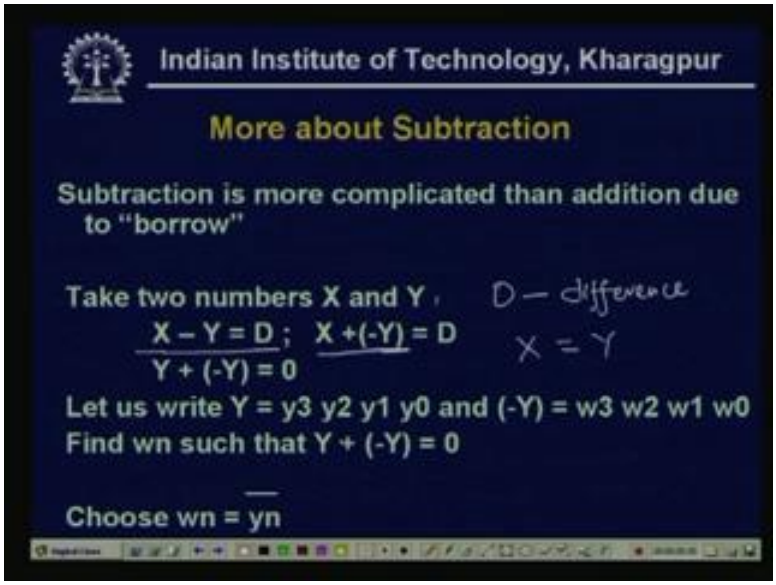
(Refer Slide Time: 21:30)



Now, if it is borrow full subtractor, then only the inputs are different, otherwise are structure is totally same. For the first AND gate the inputs are a dash and b one complemented variable is there. For the second AND gate, also one complemented variable a dash is there, for the third word, it is only b c and 1, three input OR gate is

needed. So, structure is totally same only the inputs are one complemented inputs are added.

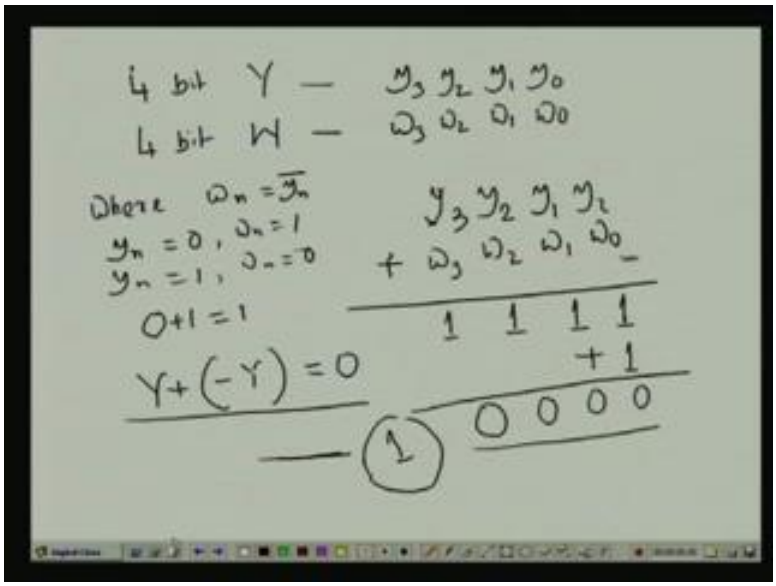
(Refer Slide Time: 0:22:14)



Now, again we see some more about or we read some more about the subtractor, because subtraction is more complicated, then addition due to the concept of borrow. Now, we take 2 numbers say X and Y, see X minus Y, we have defined as the difference D, so D is my D is my difference. Now, what will happen that if I can write the same expression X minus Y equal to D, if X plus Y minus D, I am instead of subtracting, I am doing addition, I am of a negation of Y or the minus Y.

Then, it will be same the X plus minus Y equal to D, now if I would replace X by Y; that means, it my X equal to Y, then this is of Y minus Y. Means the difference should be 0 or if I again arrange or these expression, if I write this should be Y of minus Y equal to 0, now we take a 4 bit Y and see that, how it works.

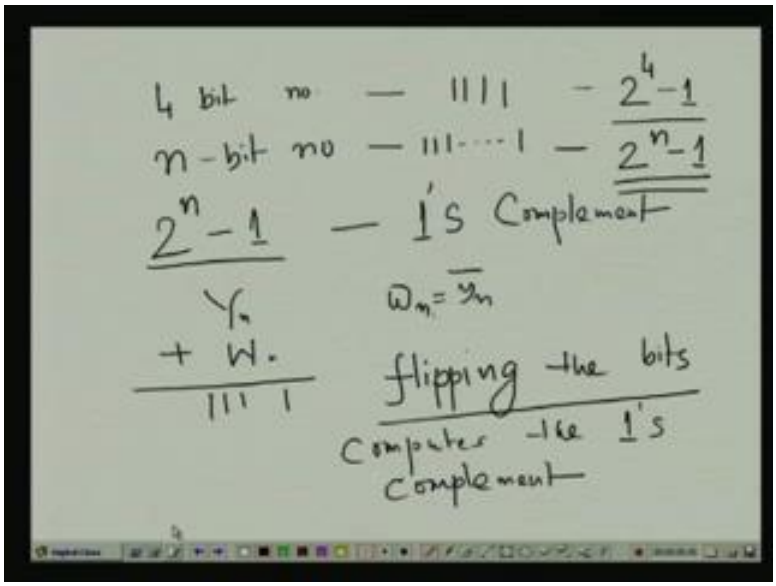
(Refer Slide Time: 24:08)



See, I choose a 4 bit Y, say this a y 3, y 2, y 1 and y 0, say I am considering an another 4 bit, W as if w 3, w 2, w 1, w 0, were w n is y n bar. So, that means, if my y n is 0, w n will be 1, if my y n is 1, w n now y n will be 0. So, now if I take the addition of this 2, say y 3, y 2, y 1, y 0 and then w 3, w 2, w 1, w 0, then if I take addition as they are complimented. So, all ways is I will get all 1, because these are always complemented as 0 plus 1 is 1.

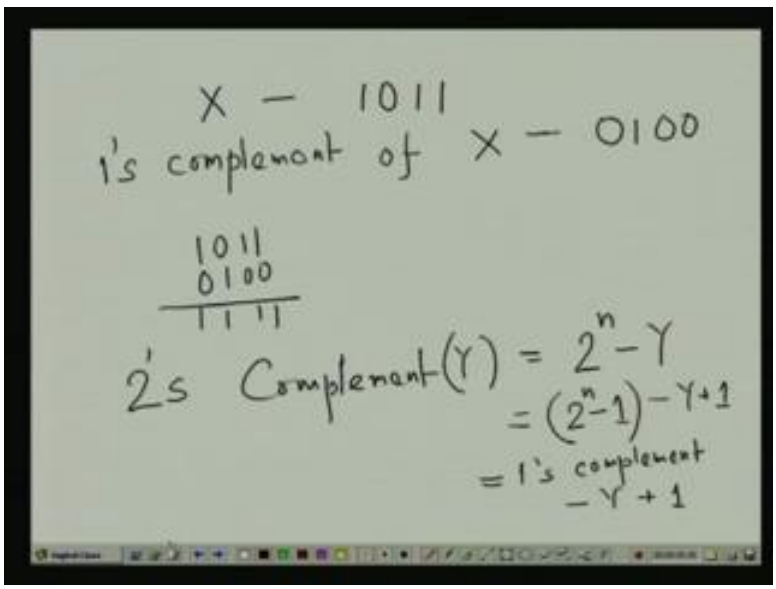
Now, I want that Y plus of minus Y equal to 0, this should be that is it, now if I add 1, then what will happen then only it becomes 0 and a carry 1. So, now if I neglect this carry or this is a overflow. So, if I drop this 1, then my Y plus w becomes 0, now see Y plus of minus Y equal to 0; that means, I have replaced this w as y complement. So, if I forcefully, I am making as if Y plus minus Y equal to 0, then this w is my subtraction, then what we can call now, see that this all 1.

(Refer Slide Time: 25:14)



Now, I define in this way, say for 4 bit number, this all 1, so this is actually 2 to the power 4 minus 1, so this actually 15, so for in general if it is a n bit number, then this n number of 1's means 2 to the power n minus 1, this will give all 1's, all n bit 1's. And this 2 to the power n minus 1, we are defining 1's compliments, so that means, Y plus W, if I take that W is or W n is y n bar, better we take that, small w n, y n bar, then this is always 1, whatever be the bits, if it is n bit, if it is n bit then 1. So, this is just of 1's complement is the flipping the bits, so we call the flipping the bits and by flipping the bits, it computes the 1's complement.

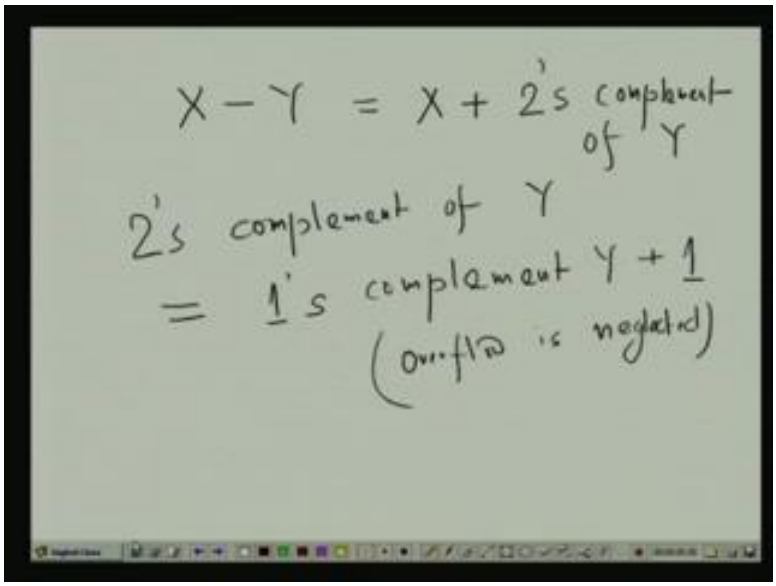
(Refer Slide Time: 29:55)



That means, say if one number X is 1 0 1 1, then 1's complement of X is 0 1 0 0, now see 1 0 1 1 and 0 1 0 0, this becomes always all 1. So, in this way, we define that 1's complement, now we take the 2's complement also and normally we define that 2's complement as 1's complement, plus 1, how say the 2's complement of a number, say 1 number Y, we have taken, see the 2 to the power n minus Y.

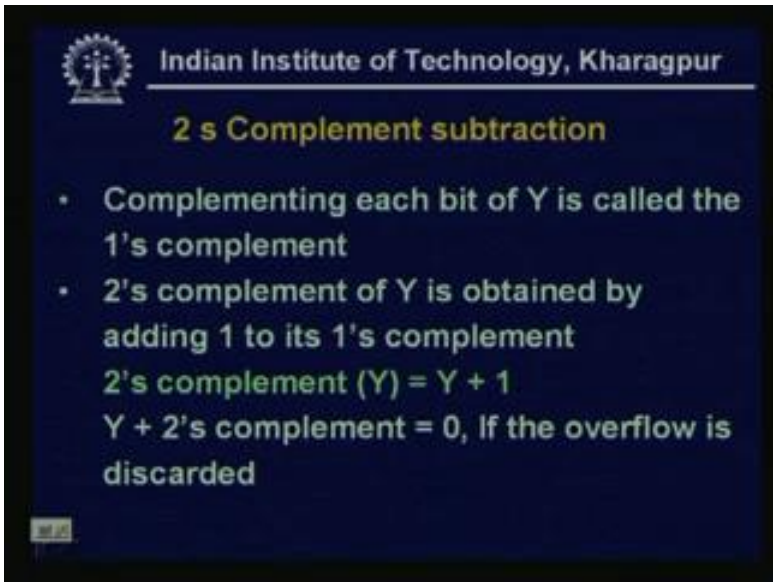
So, that we can write in this way, that 2 the power n minus 1, actually minus 1 plus 1 means Y plus 1. Now, see 2 to the power n minus, already we have define, this is as a 1's complement, so 1's complement minus Y plus 1. Now, if the overflow is or this is 1's complement plus 1 2's complement is, so what we can do that, if now I want to add X.

(Refer Slide Time: 33:03)



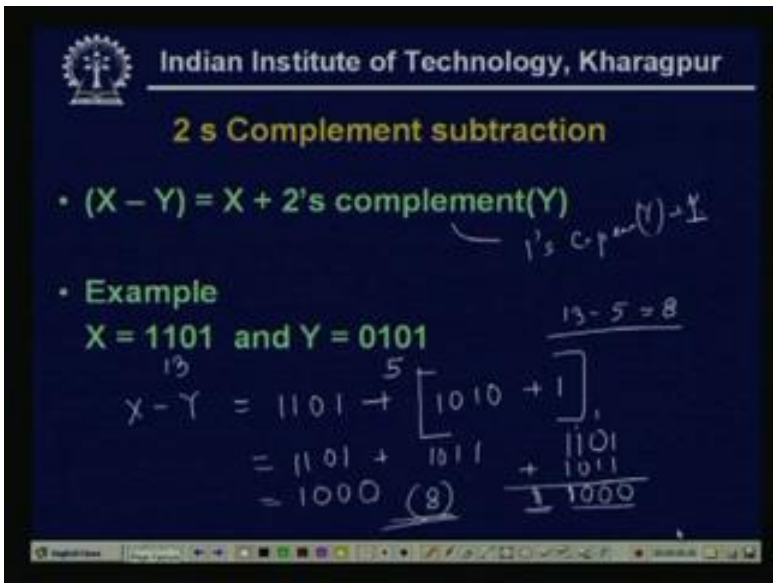
So, what will happen that X minus Y will be X plus 2's complement Y and what we have define that 2's complement Y is 1's complement Y plus 1 and the overflow is neglected.

(Refer Slide Time: 34:07)



So, complementing each bit of Y is called the 1's complement; that means, the flipping the bits just now we have discussed. Now, 2's complement of Y is obtained by adding 1 to it is 1's complement. Just now we have seen and we have derived that thing, then 2's complement Y is Y plus 1 and Y plus 2's complement equal to 0, if the overflow is discarded.

(Refer Slide Time: 34:41)



So, in this way will get that X minus Y is X plus 2's complement of Y, so this is a very important result, that what we see, that if I want to add or if I want to subtract Y from X;

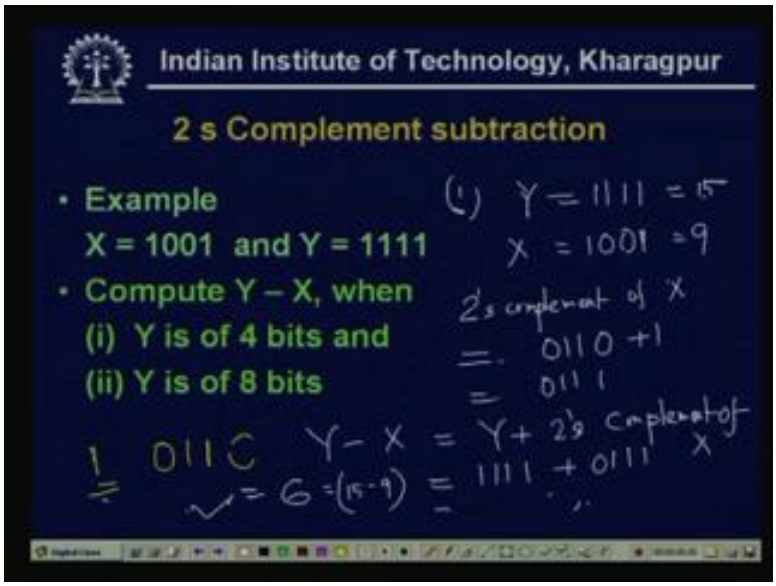
that means, X minus Y. So, there is no separate subtracted is needed, if my 2's complement of the one number available, then what we can do the same adder circuit, the can be use for the subtraction.

See this X minus Y, so first I will compute the 2's complement of Y and we know that 2's complement Y is will take the 1's complement Y again plus 1. So, this is again 1's complement of Y plus 1, this is again some addition and then it will be added with the X. So, no subtraction is there, only the addition circuit is sufficient for subtraction. We take one example first, see there are two numbers X is 1 1 0 1, so decimal equivalent is 3 0 1 3 and here the decimal equivalent is 5 Y.

So, I want to do X minus Y, which should be 13 minus 5 equal to 8, now Y we have done, first X minus Y should be 1 1 0 1 plus say 1's complement of Y, means this is 2's complement of Y, means first 1's complement of Y plus 1. So, 0 1 0 1 if I flip, then it will be 1 0 1 0 plus 1, so these will be 1 1 0 1 plus 1 0 1 1. So, this 1 plus 1 or if we add here 1 1 0 1 and 1 0 1 1, then again 1 1 0 carry 1, again 1 1 0 carry 1. Now, 1 11 0 and 1, carry1 so; that means, this is 1 1, so 1 sum is 1 and 1 carry is 1.

Now, if we neglect or discard this overflow bit, then it will be 1 0 0 0, which is the decimal equivalent is decimal equivalent is 8 and which satisfies are normal results. So, instead of doing the subtraction, actually we have done the addition. So and simple addition circuit, we can easily use for this provided, that 2's complement of that number is available.

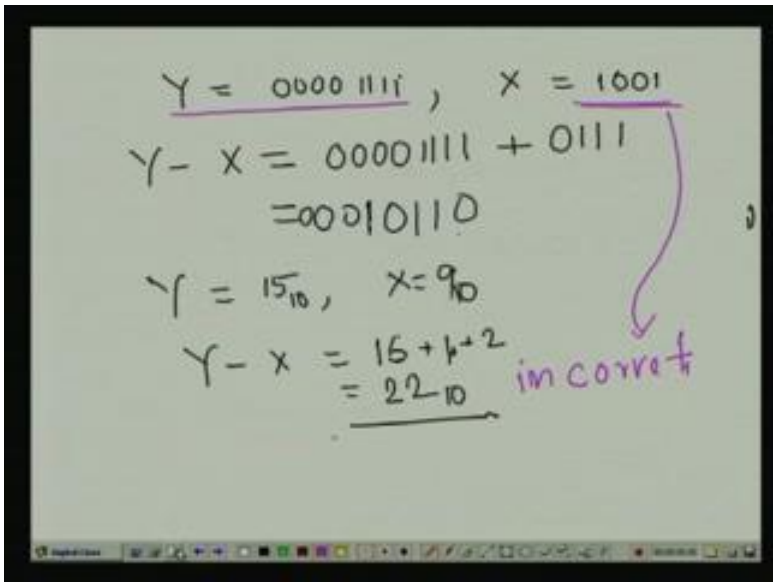
(Refer Slide Time: 38:22)



Now, we take another example say X is 1 0 0 1 and Y is all 1, then compute Y minus X, when Y is a 4 bits and Y is a 8 bits. So, first we take the simple 1, the Y equal to 4 bits, this on my here we have computing Y minus X, so Y is all 1 X is 1 0. Now, 2's complement of X this equal to 1 complement X is 0 1 1 0, flipping the bits plus 1, this equal to 0 1 1. Now, instead of doing the difference, I will add Y minus X is Y plus 2 complement of X.

So, Y is 1 1 1 plus 0 1 1 1, if we add this 1 plus 1 0, carry 1, so we are writing here, 0 carry 1, 1 plus 1 1 0, so this is 1, again 1 plus 1 1 0 and 1 1 1 carry1, 1 plus 1, 0 carry 1. So, this is my overflow bit, I will neglect this 1 and this should be my 0 1 1 0. So, the result is if I take the decimal equivalent, just to verify whether my difference is correct or not. So, 0 1 1 0, means this is my 6 and Y was 15 and X was 9 in decimal, so 15 minus nine this 6 equal to 15 minus 9, so in this way will get the result.

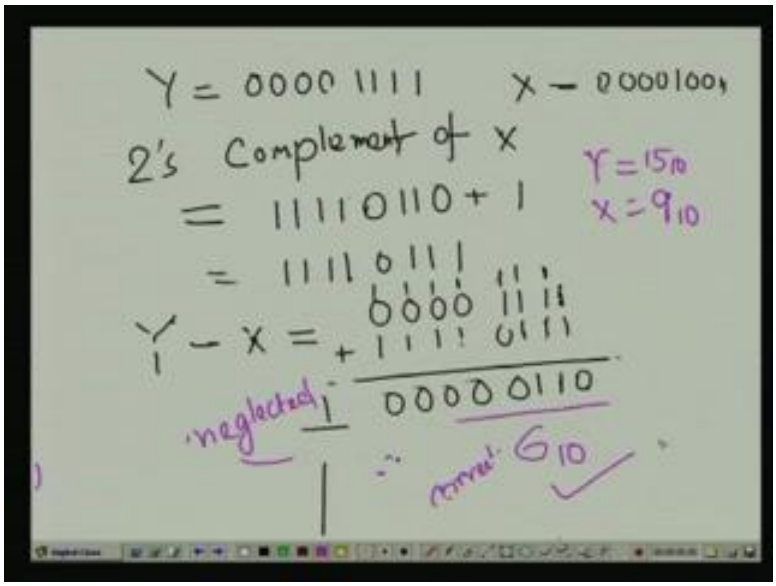
(Refer Slide Time: 42:09)



Now, Y is of 8 bits; that means, Y 1 1 1 1, always I can represent as 8 bit numbers, so now we see say my Y is actually 4 0 1 1 means 8 bit representation, X is as it is 1 0 0 1. Now what will happen, already we have seen that this is Y minus X equal to Y. And this is 2's complement of X means, this 0 0 1 1 0, 0 1 1 1, now if we add, it will be 0 1 1, 1 1 0 1 and 0 0.

So, what will be the value, see here it was Y was 15, X is 9, here what is Y minus X, Y minus X we have got that this is 0, 1, 2, 4, 8; that means, 16 plus 4 plus 2, so 22 this a decimal. See, I have 2 numbers Y equal to 15 and X equal to 9 and when I have taken the difference, I got 22, 15 minus nine is 22. So, this is a incorrect results, why what mistake I have done, see here Y is represented as 8 bit and I have taken X is 4 bit, so this the incorrect thing. So, I have to take X is also of the same bit size as that of Y, so X should be of 8 bit numbers.

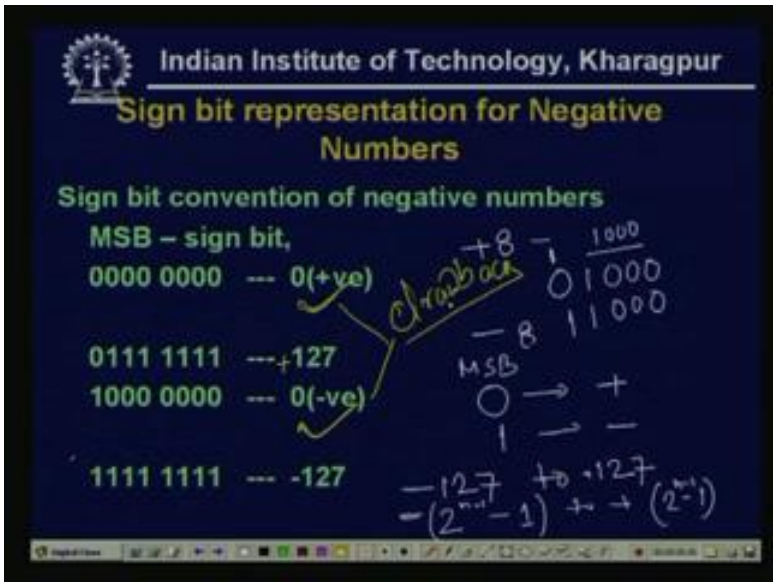
(Refer Slide Time: 45:14)



Now, we check whether it gives a correct result or not, now Y is and X we have taken 8 bits, then if I take 2's complement of X, then it will be 1 1 1 1 flipping the bits and this 1 0 0 1 means 0 1 1 0, then it will be plus 1, so this 4 1 0 1 1 1. Now, if I do Y minus X, so these becomes, we have to add this Y and 2's complement of X, which is this number 1 1 1 1 0 1.

So, 1 plus 1 0 carry 1, again 3 1 carry1, 1 plus 1 0 carry 1, then 1 plus 1 0 carry 1, 0 carry 1, 0 carry 1, 0 carry 1, overflow and this neglected or discarded. Then, see again the value becomes 6 decimal values and earlier it was 15 Y was 15 in decimal X was 9 in decimal and the result is correct now, so this is a correct result. So, what we have seen that, if my any one of the number is some n bit number. Then the other number should also be of the same bit size, otherwise the result would be wrong, just for this example, we have check this thing.

(Refer Slide Time: 49:00)



Now, we see the negative number representation, one thing we know that sign bit representation for negative numbers. Because we have mainly discussing the subtractor and subtractor we have represented as if this is addition. So, addition of negative numbers, so how we can represent the negative number that is one very important thing in this context.

Now, sign bit convention of negative numbers, normally we take MSB as the sign bit, see that if I want to represent say 8, we know binary of 8 is 1 0 0 0, if this is a plus 8. So, this is the 4 bit representation 8. If it is a plus 8, I am taking MSB it is 0 1 0 0, if it is a minus 8, this MSB value I am taking a 1 and there remaining as same. So, this is the MSB is represented as a sign bit, so 0 for plus 1 for minus and this is my MSB bits.

So, normally this is the convention for sign bit and this is the sign bit convention for negative numbers. So, if it is a say 8 bit numbers and the MSB; that means, this 1, this number is the sign bit, it represents the sign 1, see here it is 0. So, this is the positive number here it is a 0, so this must be this must be a positive number, this is 1. So, this is a negative number, this is 1, so this is a negative number.

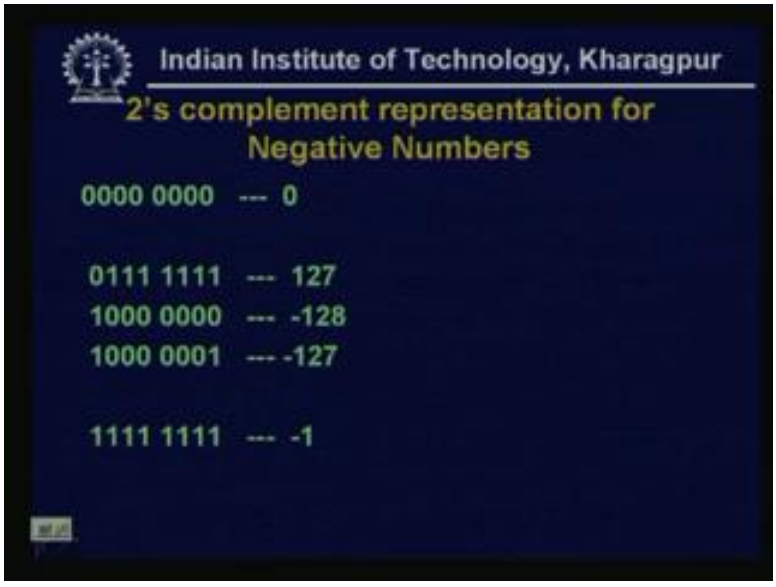
Now, see that 0, 0 means, if it is a 7 bit representation, all bits will be 0 and MSB is 0. So 0, I am telling this is a positive 0, why see again 1, all 0, this is negative number, say as if this is a negative 0, for the remaining. So, this is 1 type of confusion, that 0 we know in our number system, only 1 in the number axis, it appears only 1's. Now, I am when

representing, I am getting two type of representation, 1 is positive 0, 1 is negative 0 in sign bit convention.

But, for the rest of the numbers, it is fine, up to 0 as 7, for 7 bit, we know that, it can represent 127. So, 0 7 1's means, this is plus 127 and 1 7 1 means this is a minus 127, so it can represent minus 127 to plus 127; that means, a in general for this is for 8 bit. So, for in general, if it is a n bit, then it is for minus 2 to the power n minus 1, see 127 means minus 1, 1 less than 1 2 to the power 7. So, this is minus 2 to the power n minus 1 to plus 2 to the power n minus 1 minus 1.

This is a range that, if it is sign bit convention 1 n bit number can represent, now only one drawback of this method is there are 2 representation of 0's, one we call negative 0 and positive 0. So, just now, what we have seen that, it is for this 0, this is positive 0, this is a negative 0 and this is a drawback of sign bit number. So, for this we use the 2's complement representation, which overcome this drawback.

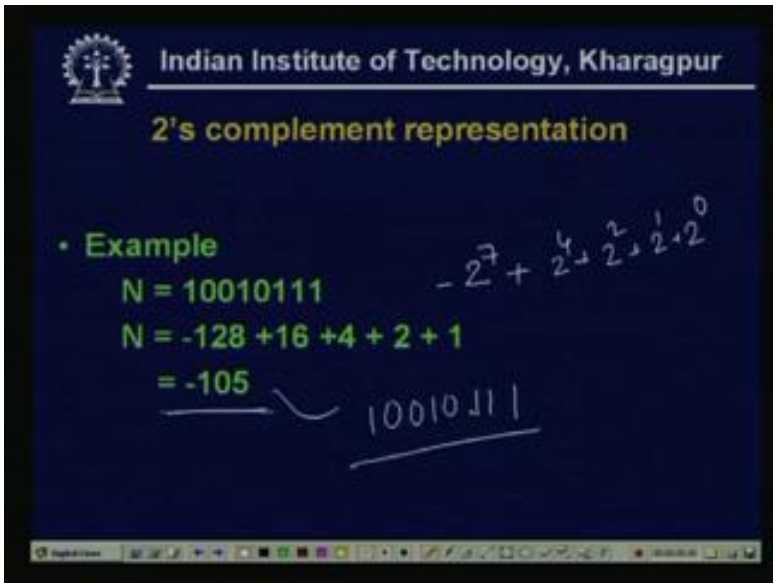
(Refer Slide Time: 54:25)



So, 2's complement representation, just now we have rate that, how we can compute 2's complement, if it is a 8 bit, 8 0 means 0, then 0 to 7 1 means 127. Now, if it is a 1 0 0 0 means this is a 2's complement number and this is minus 128. So, see this minus 128 and 1 3 0 1 mean 0 0 0 0 0 0 1 means this is a minus 127 and the all 1 means this is a minus 1.

So, here 0 means only unique representation of 0 is there, so this is value this overcomes the problem of the double representation of 0, that confusion it overcomes. So, we represent that negative numbers by 2's complement.

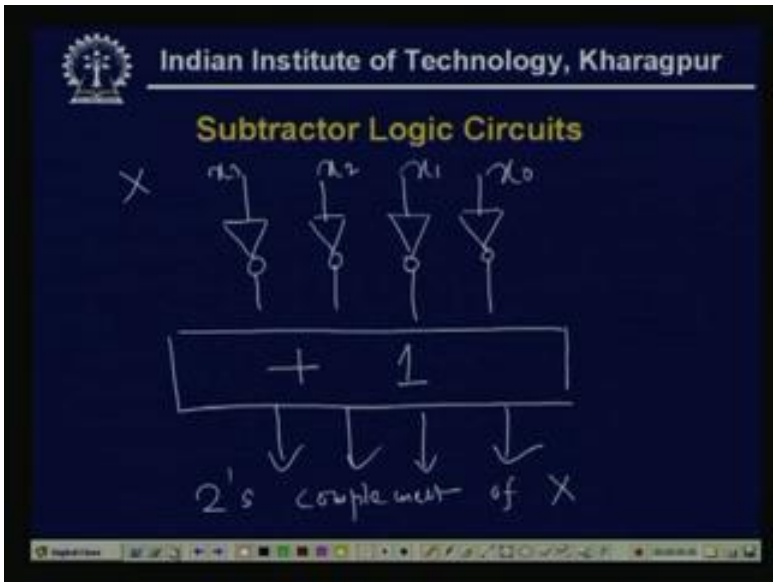
(Refer Slide Time: 55:41)



See here if I take N number, we take 1 example that N equal to 1 0 0 1 0 1 1 1, then what will happen for N equal to this is minus 128, because this is 0, 1, 2, 3, 4, 5, 6, 7 means this my 2 to the power 7. So, the first one is the 2 to the power 7, this is minus and these are plus 2 to the power 4 plus 2 to the power 2 plus 2 to the power 1 plus 2 to the power 0. So, this becomes minus 128 plus 16 plus 4 plus 2 plus 2 plus 1 and this that is minus 105.

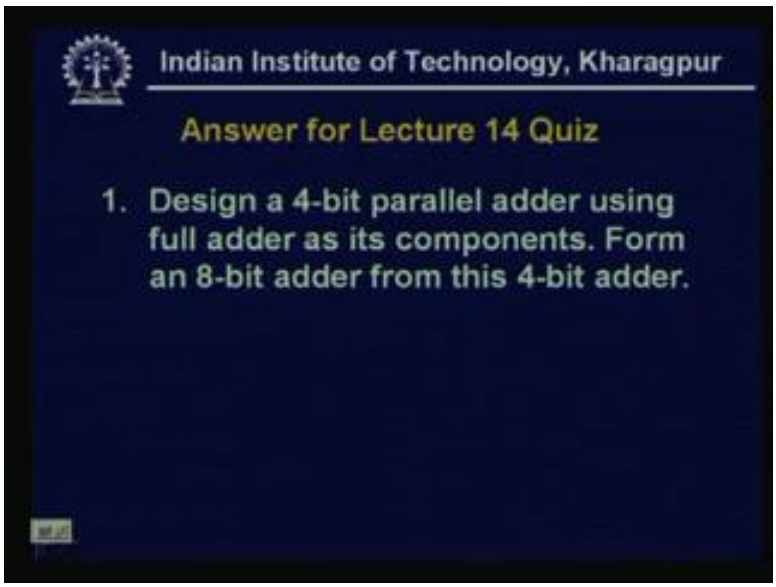
So, this is now representation; that means, for negative number minus 1 0 5 can be represented as 1 0 0 1 0 1 1 1. So, this is the 2's complement representation of negative numbers.

(Refer Slide Time: 57:05)



So, the subtractor logic circuit, we can easily draw that because it is a nothing but that addition only the 2's complement we have to do, so if it is 4 bit number, then only some 4 inverter and needed because this is flip or flipping the bits. So, there will be four inputs say this is that x 3, x 2, x 1 x 0 that X is the input and then we add this plus 1 is added and it represents, it represents the 2's complement of X.

(Refer Slide Time: 58:30)



Now, one question for quiz question for today's lecture, see design a 4 bit parallel adder using full adder as it is components. So, already we have rate the half adder circuit, the

full adder circuit and verity of full adder, half adder circuits. Now, will treat this half adder or full adder as if give as the one of the module just like a gates, say when we have drawn the full adder, we have use the AND, OR as the element digital element.

Now, as if the full adder is one module and then form an 8 bit adder, from this 4 bit adder. So, first will we have to draw, we have to design a 4 bit parallel adder from the one full adder as the components, then from 8 bit adder from this 4 bit adder.

Thank you.