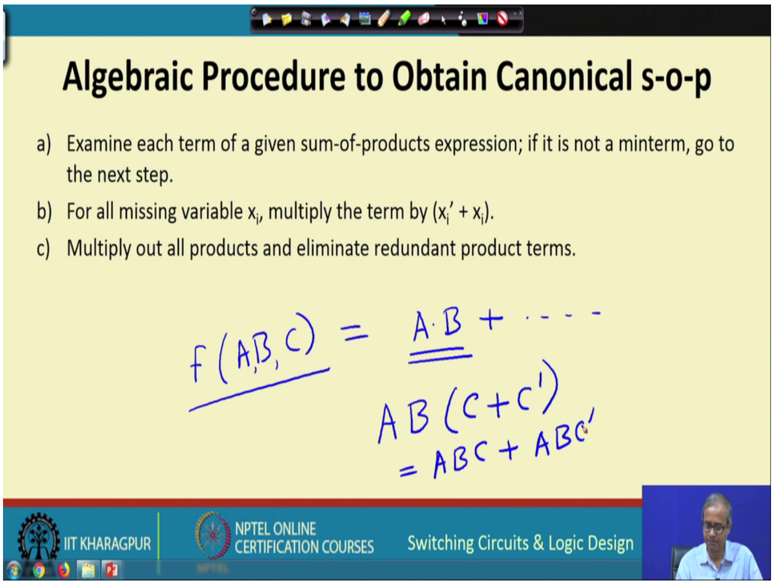
Switching Circuits and Logic Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 14
Obtaining Canonical Representations of Function

If you recall in our last lecture, we had talked about how to obtain the canonical sum of product and product of some representations of a function from the truth table. So, we continue with our discussion in this lecture as well, we shall see that not only from the truth table even from a given expression; we can systematically apply some algebraic manipulation technique to obtain the canonical expression. So, let us try to look at it and see that what kinds of transformations are possible?

So, the title is Obtaining Canonical Representations of Functions.

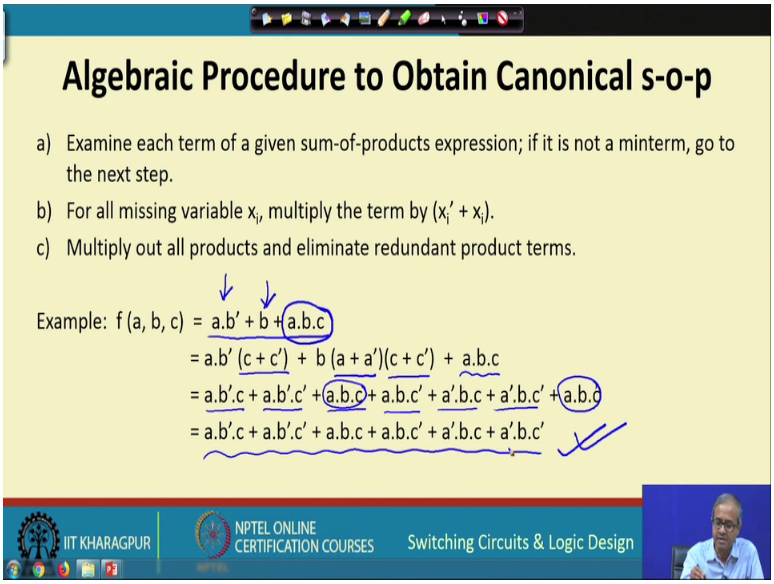
(Refer Slide Time: 00:56)



So, we first look at the algebraic procedure. So, what is the algebraic procedure? So, what we are saying is that for a given function let us say I have a function of let us call A B C a function of 3 variables A B C. And in this function I have one term let us say A B or something else is there; you see this is certainly not a minterm because C is absent. So, what you are saying is that for all missing variables you multiply the product term by that variable plus its complement. The reason is very simple if you multiply them out it will be nothing but A B C or A B C bar just like this right.

So, this same rule is being followed here you see; here we examine each term of a given sum of product expression. Well, if it contains all the variables that mean it is a minterm do not do anything, but if it is not a minterm; some of the variables must be missing. So, for every missing variable let say x i multiply that term by as I said x i bar plus x i. Then using the algebraic rules, you multiply out all the terms and finally, you will be getting a sum of minterms; some of the minterms some of them may be repeated. So, you eliminate the repeated once redundant once and what you get will be your canonical sum of product let us take an example here.

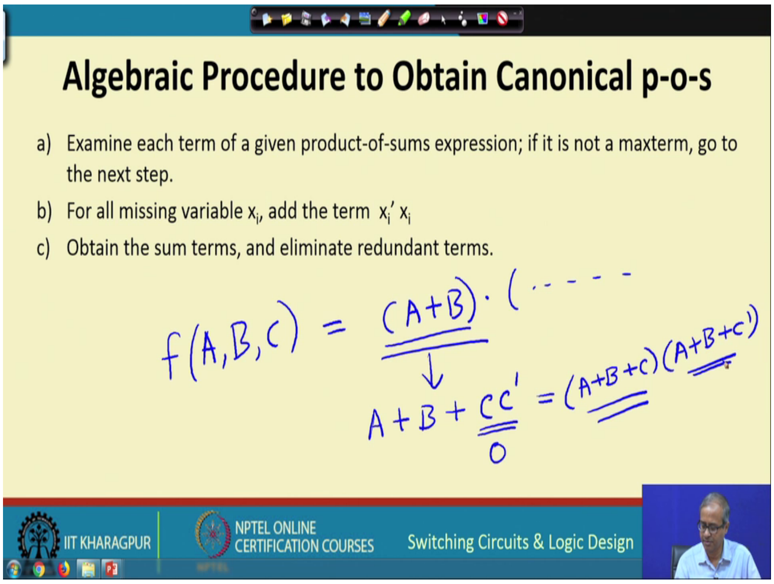
(Refer Slide Time: 02:46)



Consider a 3 variable function given like this. So, here you see the first the last one is; obviously, a minterm all the variables are there, but the first product term c is missing, the second product term both a and c are missing. So, for the first one what we do we multiplied by c or c bar. For the second one we multiplied by both a or a bar and also c or c bar, for the last one we do not do anything because all the 3 are there. Then you simply multiply them out a b bar c a b bar c bar, and here you multiply them out I have just skip this step see a c and b a c bar and b, a bar c and b, a bar c bar and b you saw multiplied all of them out and a b c is there.

Now, you see which of them are repeated you see this a b c is appearing twice other than that all are unique, there is no further repetition. So, you write down the expression you have 6 true 6 minterms and this will be your canonical sum of product expression. So, given any expression you can apply these rules to expand the function and using this multiplicable and multiplication rules switching algebra rules; you can obtain the sum of product expression.

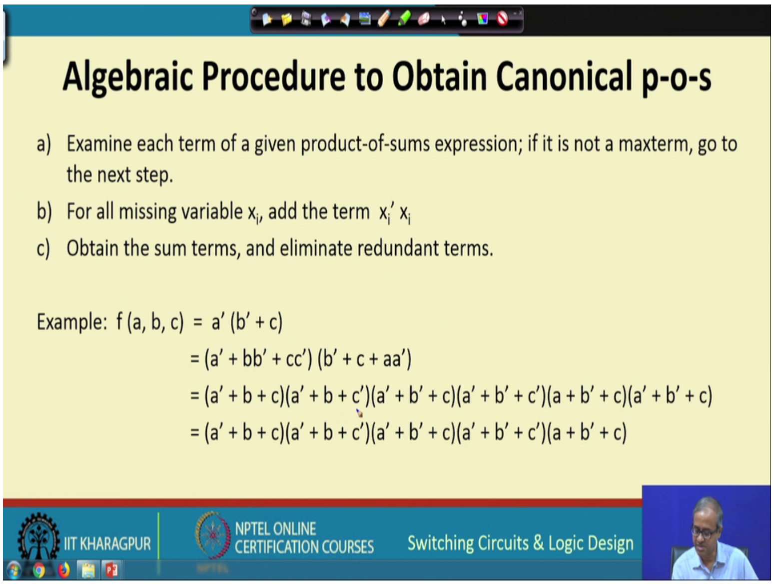
(Refer Slide Time: 04:46)



Similarly, let us look at the reverse; product of sum product of sum means here I am saying let us say again let us taken example a function of 3 variables A B C suppose I have term A plus B and something else let us look at this one, this is certainly not a maxterm because C is missing. So, if C is missing what I do I add A term C C bar to this, this I can do because C C bar is nothing but 0 and of a variable and its complement. So, anything odd 0 does not change the function.

But if I do it then I can apply the distributed law A plus B plus C A plus B plus C bar. So, I can write it as A plus B plus C and A plus B plus C bar. So, now, I have the maxterms this is the basic idea here right. So, let us see we examine each term of a product of sum expression if it is a maxterm, we do not do anything; if it is not a maxterm then you look at all missing variables and for every missing variables you add the term x i x i bar. Then as I said you can apply this simplification rules and distributive laws to obtain the sum terms and you see if any of them are redundant if the redundant eliminate them.

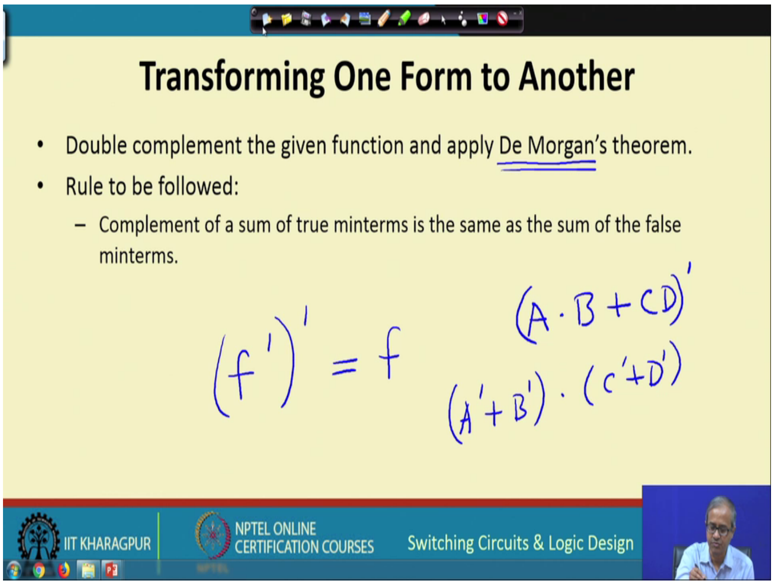
(Refer Slide Time: 06:40)



Let us take an example, suppose I have a function like this where there are 2 sum terms the first one contains only a bar second one contains b bar and c. So, what I do? For the first one b and c are missing. So, we add b b bar also c c bar the third one a is missing. So, we add a a bar now after doing this; now we apply the distributed law. So, here we have skipped a step again these two things we are showing in a combined way say a bar b c a bar b c bar a bar b bar c a bar b bar c bar. So, there will be 4 terms generated and for this one b bar c a and b bar c a bar one of them is repeated I am just eliminating the repetition. So, I am only showing the final form.

No sorry here there is a repetition let see which one is repeated a bar b c is one, a bar b c bar a bar b c bar is one, a bar b bar c a bar b bar c is repeated. So, you just use only one copy of it other one you delete and you get the final form here this is your canonical product of some expression. So, you see that means even using algebraic technique; you can start with any given expression and you can find out either the product of sums product of maxterms or sum of minterms, both of them are canonical expressions of a function, alright.

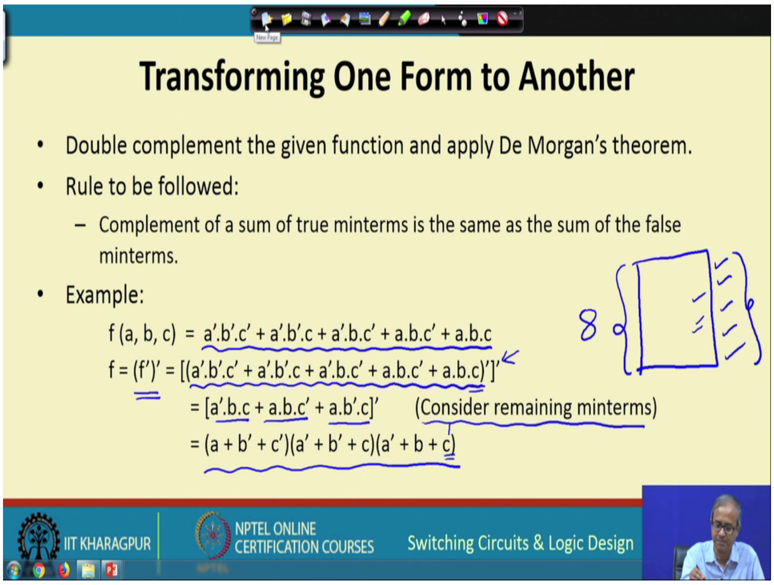
(Refer Slide Time: 08:44)



Now, let us see how we can transform the forms from one to other sum of product to product of sum or product of sum to sum of product. So, I shall show you one, the other one you can try out to the examples. Here the idea is to repeatedly apply De Morgan's law or De Morgan's theorem. The idea is this say a given function if you complement it twice, the function value remains the same. So, our idea is the same like intuitively I am telling if I have A B or C D and if I compliment this once; what I will get? If I compliment this once I will get A bar plus B bar and C bar plus D bar by 2 applications of De Morgan's law you can see.

Then we are doing another complementation let us see what you are doing this let us take an example.

(Refer Slide Time: 10:00)

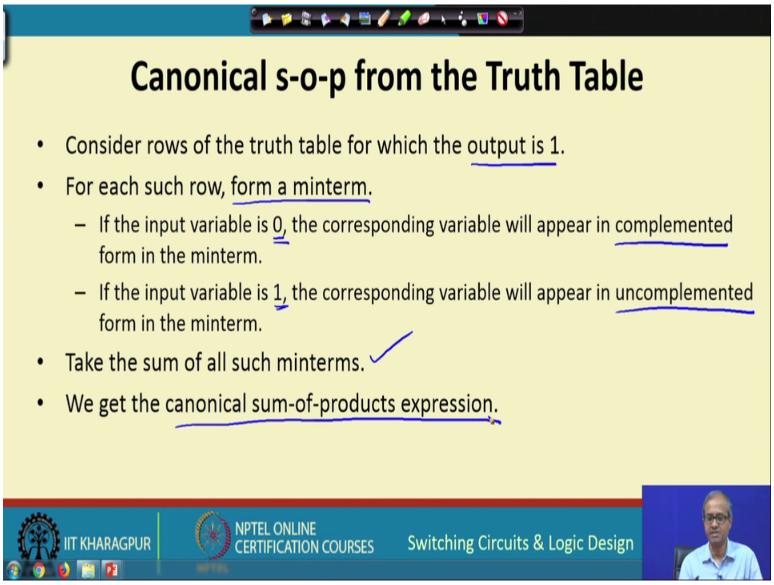


Let us take an example; suppose I have a given sum of product expression like this is a canonical sum of product expression. So, you consider that this is a 3 variable functions. So, if I consider the truth table consider the truth table. So, how many rows will be there? There will be 8 rows out of this 8 rows; 5 of them are one the output is one for 5 of them these are the combinations. So, what I am saying that the functional remain the same if I complement it twice ok.

So, f bar this was f; so, within the inner bracket f bar whole again bar. Now let us look at what is the meaning of this inner thing; sum function bar. So, what this function represents? This function represents these 5 rows of the truth table for which the output was 1; not of that what is the not of that? Not of; that means, the remaining 3 rows which was 0. So, if I take the not of this instead of algebraic manipulation I can do a shortcut, I can simply look at which of rows are not there see a bar b c was not here a b c bar was not here and a b bar c was also not here. So, if I want to complement this we simply consider the remaining minterms; it is same as this. So, if I am verify you can also compliment it do a lot of multiplications and simplifications you will be arriving at this form.

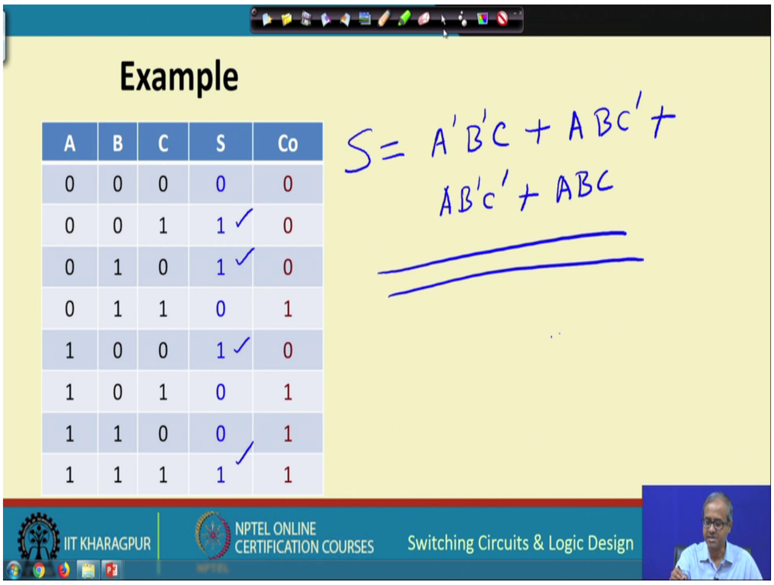
But here we have done a shortcut and finally, this again there is a not the last not here; so, if you do this now we apply De Morgan's law straight away; this a bar b c becomes a plus b bar plus c bar plus becomes and a bar b bar c plus becomes and a bar b c bar this means c bar sorry this means c bar fine. So, like this you can convert canonical sum of product expression to a canonical product of sum expression. Now see for this example, this sum of product had 5 terms, but product of sum is having 3 terms it will be smaller, but it may not be the same for all the functions for some functions it can be the other way round also.

(Refer Slide Time: 12:51)



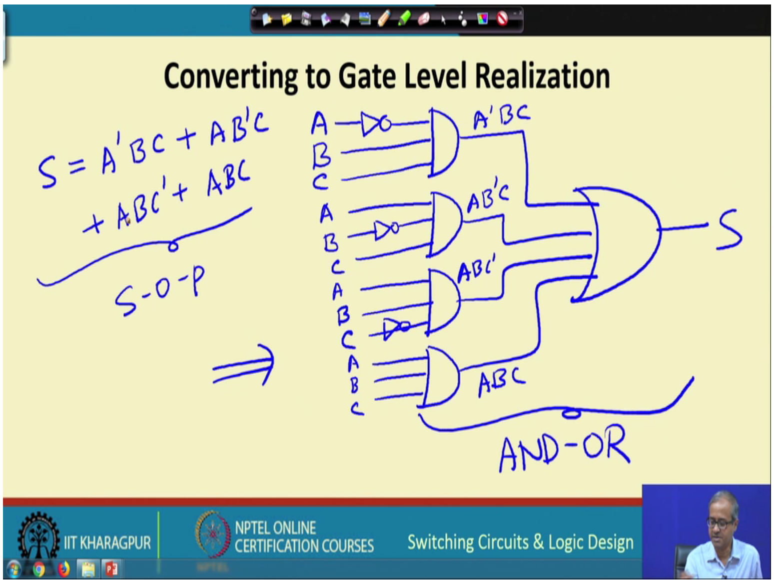
Now, this we have already seen earlier let us look at it once more; there is no harm. So, whatever saying is that is how to obtain the canonical sum of product expression for truth table. This we have already shown with the help of an example of a full adder in the last lecture. You consider rows of the truth table for which the output is 1. For each such row form a minterm where the literals are defined like this if a variable is 0; you compliment the corresponding variable, if the variable is 1, you leave the variable as it is do not compliment take this sum; what you get is the canonical sum of product expression. This is what we discussed in our last lecture let us work out the example once more there is no harm in working out again.

(Refer Slide Time: 13:55)



Let us take the example of this here just to recall what you said; just for this sum we said that we look at the rows for which the output is 1 and it is 0 0 1, we write A bar B bar C 0 1 0 we write A B C bar plus 1 0 0 A B bar C bar and finally, 1 1 1 A B C. So, you can straight away write like this right this already we discussed earlier. Now since our ultimate objective is to design the circuit once we do it what next?

(Refer Slide Time: 14:54)



So, can we convert this function into a gate level circuit? Let us see; how let us write down that expression once more.

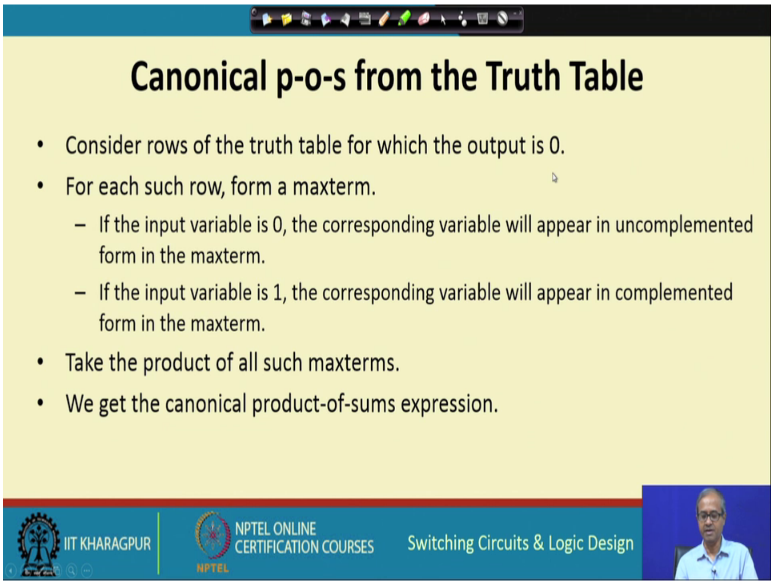
So, what you saw is sum is nothing but A bar B C plus A B bar C plus A B C bar plus A B C. You see you have a function; I can straight away convert this function into a gate level circuit; how? I use 4 AND gates one corresponding to every product term. And the output of all AND gates, I take together and I connect them in an OR gate; this will be my final output.

Now let us look at the individual and gets first one is having A bar B C. So, I have a variable A I was a not A bar B I connect straight C, I connect straight. So, A bar B C this output will be giving A bar B C. Second one is A B bar C; so, I connect A straight away B bar C. So, I have here A b bar C then A B C bar I have A I have B then of course, there will not get here there will be C bar this will be A B C bar.

And last one will be A B C no need of any NOT gates here just connect A B C. So, the point to notice that given any sum of products expression; you can directly convert it into a so, called and or circuit realization. There will be one level containing only AND gates, there will be one level containing a single OR gate and in the input you may be requiring sum NOT gates because sum of the variables maybe complemented right

So, for this function you need 4 AND gates with 3 input each one OR gate with 4 inputs and 3 NOT gates right this is how we can convert a function into a circuit.

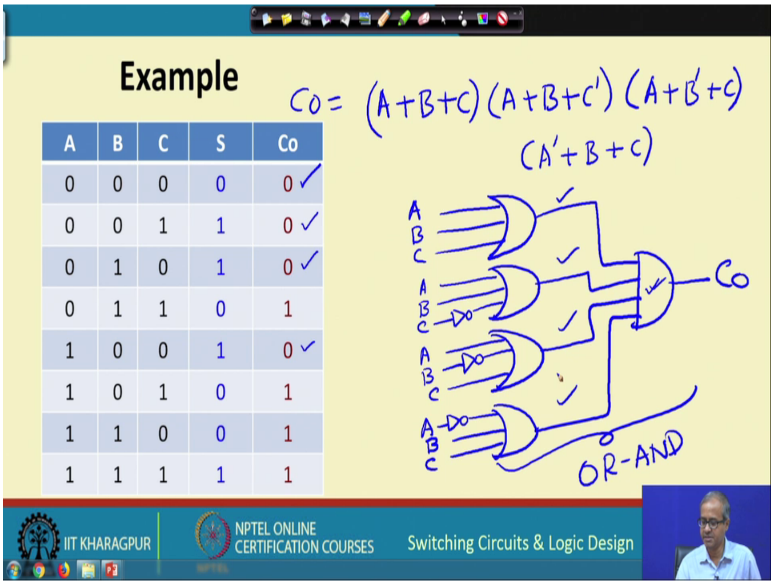
(Refer Slide Time: 17:44)



Similarly, you can do the same thing for a product of sum realization; here you recall this also we discussed earlier. We consider rows of the truth table for which the output is 0, for each row you form a maxterm where the convention for the variable polarity is just reverse with respect to sum of product. If it is 0 then you use it in uncomplemented form, if it is 1 then you complement and take the product of all the maxterms what you get is the canonical product of sum expression.

This already we illustrated earlier with the help of an example.

(Refer Slide Time: 18:35)



A	B	C	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

So, for this full adder example again let us look at the carry let us look at the carry. So, here with the conventions says you look at the outputs where the value is 0 and A B C all 3 are 0 0 0 convention is do not compliment them A plus B plus C. Second one; 0 0 1 0 0 and 1 means you compliment, third one 0 1 0 0 one 0 and last one is 1 0 0 1 0 0.

So, this will be the carry out now again just like a sum of product expression I told you; if we have this kind of an expression, you can directly convert this into a gate level circuit. But here it will be different in the first level you will be using OR gates, there will be 4 OR gates in the first level and there will be a single AND gate in the second level; this will generate the final output. So, the output of the OR gates will be connected here, now you see the, or connections first one says A B C. So, you connect A B C straight away, second one is A B; C bar A B and C with an NOT gate C bar.

Third one A B bar C A B with a NOT gate b bar and C last one A bar B C A bar B and C; so, you have this 4 product terms appearing here which you finally, and them together using this AND gate it is very simple. This is also a 2 level realization this is called a OR AND realization. Sum of products get mapped into and or, but here you get map into or an and right this is the basic convention. So, actually what we have seen is that given a function, you can either convert it into a sum of product expression; from the sum of product expression you can realize the corresponding circuit or you can have a product of sum expression from there you can convert it to a circuit.

Now, it depends whether you need an and or an OR AND. So, you will have to start with either a sum of product or a product of sum, but one thing you should remember ultimately our objective is to implement the circuit using gates and the implementation cost should be reduced as much as possible. So, one simple metric or measure of the cost is how many gates you need. Now it is not possible to say that which of the 2 will give you the better solution sum of product or better or the product of sum means a and OR AND.

So, this is depends on the circuit and it varies from function to function; you will have to analyze and find out that for the sum of product how many true minterms are there; are they larger in number or small in number? If you see that there are a large number of true minterms, then possibly product of sum will be better and the converse means also; if it is less then sum of product will be better.

So, with this we come to the end of this lecture. In our next lecture, we shall be talking about another characteristic of these gates and these basic functions. And we shall be discussed in the notion of functional completeness this we shall be discussing in the next lecture.

Thank you.