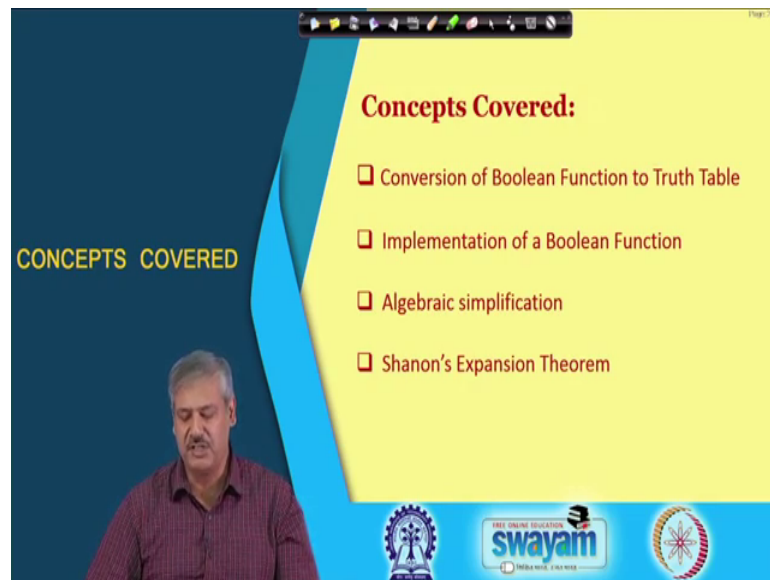


Digital Electronic Circuits
Prof. Goutam Saha
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture –08
Boolean Function to Truth Table and Implementation Issues

Hello everybody, in the last class we discussed fundamentals of Boolean algebra. So, we discussed basic postulates, Huntington postulates discussed and some basic theorems. And we saw proof of those theorems from many of them, and how to do it from the basic postulates.

(Refer Slide Time: 00:37)



Today we shall discuss how to convert a Boolean function to corresponding truth table. And then we shall see how to realize a Boolean function using logic gates. And we shall see that algebraic simplification using Boolean algebra is useful for efficient implementation. And we shall also have a look at Shanon's expansion theorem.

(Refer Slide Time: 00:59)

Boolean Function to Truth Table

$F(x,y) = x + x'y$

$F(0,0) = 0 + 0' \cdot 0 = 0 + 1 \cdot 0 = 0 + 0 = 0$ ✓
 $F(0,1) = 0 + 0' \cdot 1 = 0 + 1 \cdot 1 = 0 + 1 = 1$ ✓
 $F(1,0) = 1 + 1' \cdot 0 = 1 + 0 \cdot 0 = 1 + 0 = 1$ ✓
 $F(1,1) = 1 + 1' \cdot 1 = 1 + 0 \cdot 1 = 1 + 0 = 1$ ✓

x	y	F(x,y)
0	0	0
0	1	1
1	0	1
1	1	1

$F(x,y,z) = (x+y) \cdot (x+z)$

$F(0,0,0) = (0+0) \cdot (0+0) = 0 \cdot 0 = 0$ ✓
 $F(0,0,1) = (0+0) \cdot (0+1) = 0 \cdot 1 = 0$ ✓

x	y	z	F(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

So, the Boolean function some of the function that you have already familiarized with. So, let us take one two of them; one is a two variable function. So, $F(x,y)$ is equal to x plus x prime y ok, so that is x ANDed with x prime complement of x , x ORed with complement of x ANDed with y right. So, this is we already have seen. So, if you want to get the corresponding truth table, what we do for this x y variable, we substitute 0 and 0 like over here what you see, then it becomes 0 plus 0 prime. So, this is 0 standing for x , this is 0 prime that is substituted x value and then y is also 0.

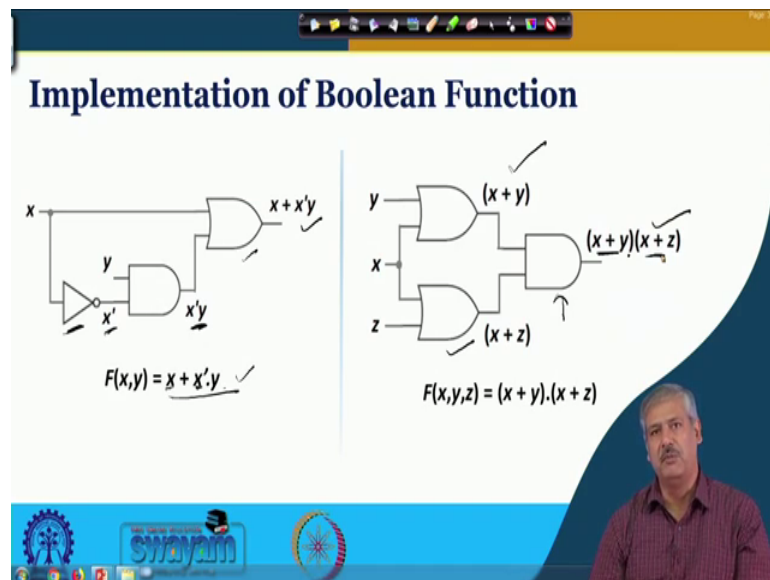
So, this 0 remains 0, and this 0 prime becomes 1 over here, and this 0 remains 0 right. So, this one ANDed with 0 is 0, so 0 plus 0 is 0. So, this way we get one particular row of the truth table, which we see over here filled, filled up. So, then the other option is x and y can take value 0 and 1. So, this is that option. And then you substitute and go on putting the values. And from the AND and OR operation, we get 1 1 1 for the remaining three possibilities, and that way we can come up with the truth table ok.

Say for a three variable case, this was where we had seen that product terms are summed finally, here the sum terms, this is a sum term, this is also a sum term with which we are doing taking an AND. So, there is a product term that is finally formed. So, there also follow the similar thing, this is a three variable problem. So, we shall be substituting x , y , z different combinations of them. So, we start with say 0 0 and 0. So, x and y both are 0. So, we put those values. So, 0 OR 0 becomes 0 over here. Then x and z they are 0, so this

is 0. So, you substitute those values 0 OR 0 is also 0, this two are ANDed we get 0 ok. So, this completes the first row of this particular truth table.

So, then 0 0 1 another a possibility, so this particular row. So, again you substitute and we get 0 from 0 plus 0 of x plus y; and 0 plus 1 we get 1; from x plus z and 0 ANDed with 1 is 0. So, this is the second row that we get, and that we get all the different rows and complete the truth table ok. This is a very simple approach given any function; given any function of any number of variables you know that the number of rows in the truth table will be 2 to the power n. If n variables are there, and we shall go on substituting each values of the combination and work out the AND and OR operation, and complement operation. And then finally, we will get the different entries.

(Refer Slide Time: 04:25)



Now, if you want to implement a Boolean function, any given Boolean function, we want to implement it using hardware. So, this logic gates are there. So, each of these logic operation, each of the Boolean operation that we do in the case of a Boolean function like this right, so for that a corresponding logic gate is there. So, x prime is required complement of x is required. So, for that you have got NOT gate.

So, we shall put not gate x is input at it and output is x prime is very easy to get. So, this is ANDed with y right. So, we will put a two input and gate. So, there are two inputs to this AND operation. So, one is x prime, another is y. So, this is these AND gate output is x prime y. And these x prime y is ANDed with ORed with x. So, we shall put two input

OR gate because there are two input to this odd operation. So, this is the two input OR gate and final output we get this way.

So, every these Boolean operation that you we do in the Boolean expression, there is a corresponding logic gate to convert it to that and we get the corresponding circuit made. Similarly, the other relation that you had seen so, this is x plus y ANDed with x plus z. So, to achieve x plus y, we need a two input OR gate ok. This is or operation to achieve x plus z we need a two input OR gate that is another this is a OR operation. And finally, these two are getting ANDed, these two are getting ANDed over here. So, for this two such input are required for these two input and get and finally, at this place we get x plus y ANDed with y plus z ok.

So, you can just simply follow the Boolean any given Boolean expression operation by operation with the operands that are there. If it is a there are two operands, so two input gates will be required; three operands inputs gates will be required. And finally, the whole expression can be arrived in the form of digital circuit implementation. Remember, in this case, we are not talking about the propagation delay power consumption and all those things it is just simple implementation of the Boolean expression right.

(Refer Slide Time: 06:59)

Efficient Implementation
Use of Boolean Algebra

Equivalence can be verified from Truth Table.

From Adsorption Theorem:
 $F(x,y) = x + x'y = x + y$ ✓
 Circuit diagram showing an AND gate with inputs x and y, and an OR gate with inputs x and the output of the AND gate. The output is F(x,y).
 Earlier: One NOT, One 2 i/p AND, One 2 i/p OR ✓
 Now: One 2 i/p OR ✓

From Postulate (Distributive Law):
 $(x + y)(x + z) = x + yz$ ✓
 Circuit diagram showing two AND gates. The first has inputs x and y, outputting yz. The second has inputs x and z, outputting x + yz.
 Earlier: One 2 i/p AND, Two 2 i/p OR ✓
 Now: One 2 i/p AND, One 2 i/p OR ✓

Now, well we talk about this implementation. In the previous case, you did not bother about those factors that can affect the realization, but in practical cases you see that

amount of power you consume amount of power the circuit consumes the fan in how many input can be there fan out how many output can be there, propagation delays if number of pages get you know becomes very much so many such issues would be there. And we will always look for a efficient implementation. And in efficient implementation we shall try to reduce we try to you know have a most simplified form of representation as much as possible.

So, in that case, the Boolean algebra that we have the fundamentals which you discuss before can be of very useful I mean very important. So, the previous realization that we talked about the previous one that x plus x prime y , we have noted that it requires one NOT gate, this NOT operation here. One to input AND gate for this operation; and one to input OR gate for this final operation ok. This is this was what was required if it go for direct implementation just following the Boolean expression.

But if we if we apply adsorption theorem, we know the same input-output combination the truth table, the relationship that we have seen will be available just by equivalent expression which is x plus y right that is from the basic theorems. So, this x plus y if we implement whatever we get we will get we get exactly the same by implementing x plus x prime y . So, it makes sense that we go for this implementation then this one. So, in that case, we just need one to input OR gate as we have seen. And in the other example, so these x plus y ANDed with x plus z , so from distributive law the basic postulate, we know that this is x plus $y z$. And if you want to implement it we need one to input AND gate over here to realize $y z$ as we have seen; and one to input OR gate to realize x plus $y z$ as it is there

And earlier it was for this case one to input OR one to input OR, so that is two to input OR, and one two input AND gate, so that was the requirement. And in this case one to input AND, one to input OR. So, by using Boolean algebra by simplifying relationship, we see that the hardware requirement is reduced. And we have associated benefit in terms of different performance metric.

(Refer Slide Time: 09:53)

Algebraic Simplification

Simplify, ✓
 $Y = F(A,B,C) = A(A' + C)(A'B + C)(A'BC + C')$ ✓

$Y = (AA' + AC)(A'B + C)(A'BC + C')$: distribut.
 $= AC(A'B + C)(A'BC + C')$: $XX' = 0$
 $= (AC \cdot A'B + AC \cdot C)(A'BC + C')$: distribut.
 $= AC(A'BC + C')$: $XX' = 0$
 $= AC \cdot A'BC + AC \cdot C'$: distribut.
 $= 0 + 0 = 0$: $XX' = 0$

Note that, output Y is always L here and can be connected to LOW voltage directly.

Simplify, $Y = (A + B)(A'(B' + C)') + A'(B + C)$

$Y = (A + B)((\overline{A + (B' + C')}) + A'(B + C))$: De Morgan's
 $= (A + B)(A + BC) + A'(B + C)$: De Morgan's
 $= (AA + ABC + AB + BBC) + A'(B + C)$
 $= (A + AB + ABC + BC) + A'(B + C)$
 $= A(1 + B + BC) + BC + A'(B + C)$
 $= A + BC + A'(B + C)$
 $= (A + A'(B + C)) + BC$
 $= A + B + C + BC$
 $= A + B + C(1 + B)$
 $= A + B + C$ ✓

So, that was a very simple example. There can be more complex relationship arrived at by examining a particular problem, which is given in the form of an English statement. And when we try to convert it, just by following the statement, we may arrive at a complex relationship like this one. This is just an example of the kind of relationship you are getting. So, we can go for direct implementation of this the way we have just seen that how a Boolean expression can be realized in the form of hardware. So, we can have a NOT gate to realize C prime, then another NOT gate to realize A prime from A, then a three input AND gate to realize ABC, then this can be ORed with C prime, the two input OR gate and so on and so forth. And ultimately you can arrive at a final circuit which will provide you the truth table these eight rows and you know the corresponding columns the output function.

So, the same truth table, we can get if we go for a simplification process and whatever circuit finally arrived at if we just go for implementation of that, so that is what is the usefulness of this simplification process. And if you try to do that, in this case by following the partially basic postulates and basic theorems that we have discussed before in the last class, then we can see just a few steps. Let us have a look at this, when you are ending with A prime, this particular thing, so you get this one and the rest of the remaining things remain the same. So, A prime gives a 0. So, you are left with AC only. So, this is the AC from here and the rest of the terms are out as it was.

So, then again you do AND operation of $A C A' B A C$ and C right. So, this A and A' again will give you a 0 and what you are left with is only $A C$, so that is the $A C$ and then the rest the rest of the term then again you go for ending of this. So, $A C$ and with $A' B C$ and then you get $A C C'$. So, this A and A' will give you 0 ; C and C' will give you a 0 , so 0 plus 0 is 0 . So, it is as good as the output Y , you directly connect to logical low whatever you get is the same as do you all the different you know circuit operation effectively it remains the same ok, so that is the equivalence that we are talking about. And also occasionally one you know gives the you know identity to be proved that this is the left hand side. So, that this is equal to 0 I mean this is equal to binary is 0 at the end. So, there also you can find these algebraic simplification terms of use

So, another example we can see where we use the DeMorgan's theorem. So, here in these expression, this we have to simplify, and then implement we can go for again as I said a direct implementation following the equation, but that may be more complex and not economical. So, here in this expression, we see that there the whole expression that is a prime over there. So, we can apply Boolean sorry DeMorgan's theorem over here. So, this becomes A' , again A' double prime actually. So, this is A , and then B' plus C' the whole thing becomes a prime ok, because it was a NAND operation. So, this is the way you are getting it.

Then this $A B$ is remaining. So, B' plus C' this is again a compliments another DeMorgan's theorem. So, this is a NOT operation. So, you can get B' double prime and C' double prime, so if these becomes $B C$ ok. So, this expression then becomes A plus B ANDed with A plus $B C$ and this was what it was. So, if you just get the product of this, then this is $A A B C A B$ plus $B B C$. So, this is the corresponding term over here. And if you take A out, then 1 plus rest of the term; and we know one from the null theorem if one is there the rest of the terms of is of no use.

So, from this you have A and this $B C$ was there. So, this is your $B C$ and this is the other term ok. So, this again you can just take into consideration A and A' plus B plus C . So, in this you can use the adsorption theorem. So, there A plus B plus C is there, and this is your $B C$. So, again you can take C out and 1 plus B again using the null theorem you get from this only C , because 1 OR B is 1 only. So, final expression is A plus B plus C . So, what you require in this case is only early it was just connected to ground here, you need

only a three input OR gate right. And you can realize the same expression clear so that is what do you do if so required before implementing any Boolean expression, we shall see whether it can be further simplified.

(Refer Slide Time: 15:47)

Shanon's Expansion Theorem

$$F(x_1, x_2, x_3, \dots, x_N) = x_1' \cdot F(0, x_2, x_3, \dots, x_N) + x_1 \cdot F(1, x_2, x_3, \dots, x_N)$$

Proof: x_1 can take only 2 values, 0 and 1

For $x_1 = 0$, LHS = $F(0, x_2, x_3, \dots, x_N) = 0' \cdot F(0, x_2, x_3, \dots, x_N) + 0 \cdot F(1, x_2, x_3, \dots, x_N)$
 $= 1 \cdot F(0, x_2, x_3, \dots, x_N) + 0$
 $= F(0, x_2, x_3, \dots, x_N) = \text{RHS}$

For $x_1 = 1$, LHS = $F(1, x_2, x_3, \dots, x_N) = 1' \cdot F(0, x_2, x_3, \dots, x_N) + 1 \cdot F(1, x_2, x_3, \dots, x_N)$
 $= 0 \cdot F(0, x_2, x_3, \dots, x_N) + F(1, x_2, x_3, \dots, x_N)$
 $= 0 + F(1, x_2, x_3, \dots, x_N)$
 $= F(1, x_2, x_3, \dots, x_N) = \text{RHS}$

Dual Form:

$$F(x_1, x_2, x_3, \dots, x_N) = [x_1' \cdot F(0, x_2, x_3, \dots, x_N)] + [x_1 \cdot F(1, x_2, x_3, \dots, x_N)]$$

Example:

$$F(x,y) = x + x'y$$

$$F(0,y) = 0 + 0'y = 1.y = y$$

$$F(1,y) = 1 + 1'y = 1$$

$$F(x,y) = x' \cdot F(0,y) + x \cdot F(1,y)$$

$$= x'y + x \cdot 1$$

$$= x'y + x$$

$$= x + x'y$$

Now, Shanon's expansion theorem is something which you would like to take note of it has got many different uses. So, what Shanon's expansion theorem is says is very interesting. So, given any function of n variables ok. We shall see examples of you know smaller number two variables, three variables and all. Any you know expression of n variables, one of the variable may be taken out, for example, here x 1 it could have been any other variable. So, this variable when you take it out like this x 1 prime and rest of the function wherever x 1 is present, you put 0 ok. And then whatever result you get, you end it with x 1 prime. And then you take x 1 out, and for x wherever x 1 is present you replace 1. And then whatever you get you just or it sum it up then the you get an equivalent relationship ok. This is what Shanon's expansion theorem says right.

And to be simple in the left hand side if you place x 1 is equal to 0, then this is only 0 term is there; and in the right hand side when you place x 1 is equal to 0, so 0 prime is 1. So, this one this one ANDed with this term, so this term will be remaining. So, the other term 0 ANDed with rest of the thing is 0. So, you will be left with this one only and left hand side you have already substituted 0 over here. So, left hand side and right hand side is matching ok. Similarly, if you put x 1 is equal to 1, you will get the other term ok. So,

if x_1 is 0 and if x_1 is 1, then these are the two cases and they are getting summed up and you get the final relationship final term for the function in hand ok.

And you can look at a two variable example how it works it out. So, this if x, y , x plus x prime y you are already familiar. So, if you are taking out x right, if you are taking out x , and we would like to write it in this form. So, x prime $F_0 y$ plus $x F_1 y$ right, it is a two variable problem. So, basically x_1 is your x and x_2 is your y that is way you are doing it. So, we have to calculate $F_0 y$ and $F_1 y$. How will you do that? So, in the expression we shall substitute in place of x_0 . So, if you substitute then 0 plus 0 prime y . So, 0 plus 0 that is 1 prime y ANDed with y . So, this is y right. $F_0 y$ is y and $F_1 y$ is in substitute in place of x_1 , so 1 plus 1 prime y , s , one ANDed with anything is 1 . So, this is your $F g$ $1 y$.

So, this $F_1 y$ is your one and $F_0 y$ is $0 y$, so that way x prime y plus x , and you get back what you had seen x plus x prime y ok. So, this will be true for any other you know example this is small example, but this holds. And it is dual form ok. See in the dual form we know that it is these and is replaced with OR, and odd is replaced with AND. So, in this case, so this is the AND operation over here. So, these AND operation is replaced with OR, and this was the odd operation this is replaced with AND. This was a AND operation over here ok. So, this is replaced with OR ok. So, it was sum of two product terms, this becomes product final product of two sum terms. So, this is the dual form representation of Shanon's expansion theorem ok.

(Refer Slide Time: 20:07)

Example on Simplification

Simplify,
 $F(A, B, C, D, E) = A + \bar{A}.B + A.D.(B+E).(B.C+D.E)$ ✓

Choice of A as expansion variable as it is associated with more number of terms

$F(0, B, C, D, E) = 0 + \bar{0}.B + 0.D.(B+E).(B.C+D.E) = \bar{B}$ ✓ $0 + 1.B + 0 \Rightarrow 0 + B + 0$

$F(1, B, C, D, E) = 1 + \bar{1}.B + 1.D.(B+E).(B.C+D.E) = 1$ ✓

$F(A, B, C, D, E) = \bar{A}.F(0, B, C, D, E) + A.F(1, B, C, D, E)$ ✓
 $= \bar{A}.B + A.1 = A + \bar{A}.B = A + B$ ✓

Now, this can be useful in simplification problem as well right. So, we take one example. So, this is a (Refer Time: 20:18) problem, where the right hand side looks something like this. And if you need to want to apply Shanon's expansion theorem, one variable we want to take out. So, we will generally look for the variable which is present in almost all the terms that you see in the expression. So, here we see that A is present, A is present in all the different individual terms. So, it makes sense that we take A out ok.

So, if you take A out, so then you have to get two relationship, one is F 0, B, C, D, E and another is F 1 B, C, D, E and then we shall end it with a bar F 0 B, C, D, E, and A F 1 B, C, D, E ok. So, this is the Shanon's expansion theorem that we shall be using for taking A out. You could have taken any other variable out B also in that case if A 0 B, C, D, E F A 1 B, C, D, E that would be the case, and here the relationship would have been B bar F A 0 B, C, D, E B ANDed with F A 1 B, C, D, E, so that is the way you would have written the expression.

So, once you take a out then what do you see F 0 B, C, D, E this expression how you will you get. So, if you substitute, this is your 0, this is 0 compliment ANDed with B, and this is 0 ANDed with rest of the things. And whenever this is 0, so in the OR gate it does not you know contribute in any way it is 0, So, it is contribute in any way means it is dependent on other inputs of the OR gate. And this is ANDed with 0 means this generates A 0. So, basically you have got 0 plus 0 prime is 1 ANDed with B plus 0. So,

basically you get 0 OR B OR 0. So, so ultimately the result is B ok. So, F 0 B, C, D, E is your B. And F 1 B, C, D, E what it is. So, this is 1 plus 1 bar B plus 1 rest of the terms ok.

And we know anything ORed with 1 is 1 only from the null theorem. So, this is one this is straight forward ok. So, you have got this two terms. Now, we just need to combine using this Shanon's expansion theorem. So, A bar, so A bar is ANDed with B this particular term is B only, and A ANDed with this term which is one. So, A plus A bar B that is what you get. And after that you can use adsorption theorem, so this is A plus B. So, from this to this, this is your adsorption theorem ok. So, you get A plus B. So, this is the simplified version of A. And we could have take other you know postulates and basic theorems also to simplify it, but we see that Shanon's expansion theorem can also be useful in such a cases.

(Refer Slide Time: 23:41)

More on Shanon's Expansion Theorem

$$F(x_1, x_2, x_3, \dots, x_N) = x_1' \cdot F(0, x_2, x_3, \dots, x_N) + x_1 \cdot F(1, x_2, x_3, \dots, x_N)$$

$$F(x_1, x_2, x_3, \dots, x_N) = x_1' \cdot [x_2' \cdot F(0, 0, x_3, \dots, x_N) + x_2 \cdot F(0, 1, x_3, \dots, x_N)] + x_1 \cdot [x_2' \cdot F(1, 0, x_3, \dots, x_N) + x_2 \cdot F(1, 1, x_3, \dots, x_N)]$$

← Nesting

$$F(x, y) = x' \cdot F(0, y) + x \cdot F(1, y) = x' \cdot [y' \cdot F(0, 0) + y \cdot F(0, 1)] + x \cdot [y' \cdot F(1, 0) + y \cdot F(1, 1)]$$

← For 2 variables

$$= x' \cdot y' \cdot F(0, 0) + x' \cdot y \cdot F(0, 1) + x \cdot y' \cdot F(1, 0) + x \cdot y \cdot F(1, 1)$$

Example:

$$F(x, y) = x + x' \cdot y \rightarrow F(0, 0) = 0 + 0' \cdot 0 = 0; F(0, 1) = 0 + 0' \cdot 1 = 1; F(1, 0) = 1 + 1' \cdot 0 = 1; F(1, 1) = 1 + 1' \cdot 1 = 1$$

x	y	F(x,y)
0	0	0
0	1	1
1	0	1
1	1	1

← In Truth Table, o/p is 1 in 3 rows

$$F(x, y) = x' \cdot y' \cdot 0 + x' \cdot y \cdot 1 + x \cdot y' \cdot 1 + x \cdot y \cdot 1 = x' \cdot y + x \cdot y' + x \cdot y$$

$$\Rightarrow x' \cdot y + x \cdot (y + y') = x' \cdot y + x \cdot 1 = x + x' \cdot y$$

So, finally, we look at some of the use of you know Shanon's expansion theorem. So, this was our the basic Shanon's expansion theorem. So, where x 1 was taken out right, it is clear. So, in this particular function, there are x 2 to x N variable. So, these again can be expanded for another variable. And if you choose it to be x 2, so this one we can write as x 2 prime this is already 0. So, we do not do anything with that. So, x 2 is 0, because x 2 prime has been taken out is the ANDing term, and rest of the terms remaining same plus this one. This whole term is this 1 plus x 2 F 0 in place of x 2 since x 2 has been

taken out this is 1, x^3 and rest of the terms, clear. So, this is for this one and for the other one we can have a similar expansion with x^2 . So, basically we are having x^1 prime outside and inside x^2 prime being considered. Similarly, $x^1 x^2$ prime and x^2 over here and x^1 outside and x^2 prime and x^2 over here, and the corresponding terms here are F_{00} and rest of the terms F_{01} , rest the terms F_{10} rest of the terms and F_{11} rest of the terms ok.

Then we can go ahead with you know further such expansion of this particular at one such term. So, x^3 can be taken out ok. So, the first term will be F_{00} , and second term will be F_{001} and all, so that way what is known as nesting is possible using Shannon's expansion theorem ok. So, you can go on doing it and it has got certain use which we shall discuss later in some of the later classes ok. So, here you see how it actually works out for a two variable you know problem.

So, for a two variable problem, so say $x y$ is there. So, x prime has been taken out, so $F_0 y$ and $F_1 y$, so these are the two terms. So, this y is now you are taking out, so $F_{00} F_{01}$ right. And then $x F_{10}$ and F_{11} and then if you just take the y prime outside the parentheses bracket, so x prime y prime $F_{00} x$ prime $y F_{01} x y$ prime F_{10} and $x y F_{11}$ that is how what we get by expanding it to this level. And for an example like this the one, we had seen before $x F x y$ is x plus x prime y . So, you have seen you form the truth table $F_{00} F_{01}$ and so on and so forth. So, we have seen that F_{00} is 0 F_{01} is 1, and F_{10} is 1, and F_{11} is 1 we have already seen that isn't it.

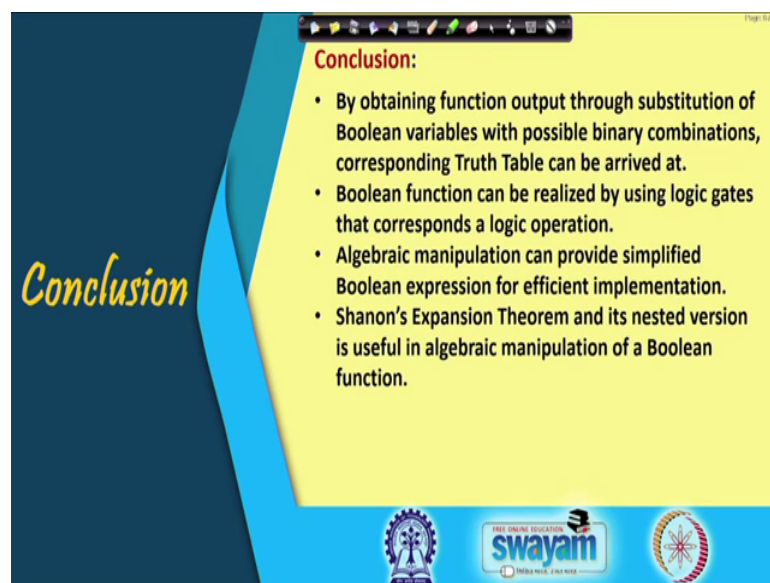
So, this corresponding terms we shall just AND it up from here, we shall use this F_{00} terms, so x prime y prime, so x prime y prime ANDed with F_{00} . So, F_{00} is 00, then x prime y is ANDed with F_{01} . So, F_{01} is we have already seen F_{01} has 1. So, this is your 1 ok. So, if $x y$ is ANDed with F_{11} ok, so that is your the final 11 over here. So, all the terms that you can see are coming here. So, this is being 0, this term is not there, and this would be terms x prime y $x y$ prime and $x y$ that would be there in the final expression ok.

Anyway expand it to that level ok. You can further simplify it. We are not talking about simplification over here. You have just saying that this is the way you can present up to this level you can if you know go on expanding it to that level. And then if you just this x prime y , and you can take x out and y plus y prime this from compliment postulate a

complementary we can see that $x \oplus y$ and this is one. So, this $x \oplus y$, we can get back this one that is a simplification process. But if you get all the terms in terms of individual variables, these are the three terms that we will get.

So, you can see in the truth table, this one is there in three cases right. And this there are three such terms where all the variables expanded to the in the maximum possible way that is what we see ok. This has got it is use in our subsequent discussion we shall begin our next lecture next class from where we leave here.

(Refer Slide Time: 29:05)



Conclusion:

- By obtaining function output through substitution of Boolean variables with possible binary combinations, corresponding Truth Table can be arrived at.
- Boolean function can be realized by using logic gates that corresponds a logic operation.
- Algebraic manipulation can provide simplified Boolean expression for efficient implementation.
- Shanon's Expansion Theorem and its nested version is useful in algebraic manipulation of a Boolean function.

So, to conclude what we have seen today in this particular class is we can obtain a function output by substituting variables with possible binary all possible binary combinations. And we can get the truth table out of it. And this any Boolean function given to us, we can realize it using logic gates by getting those logic gates which corresponds to a specific logic operation. And algebraic manipulation helps to get a simplified a Boolean expression, Boolean algebra comes very handy in that. And Shanon's expansion theorem and its nested version is useful in algebraic manipulation of a Boolean function, ok.

Thank you.