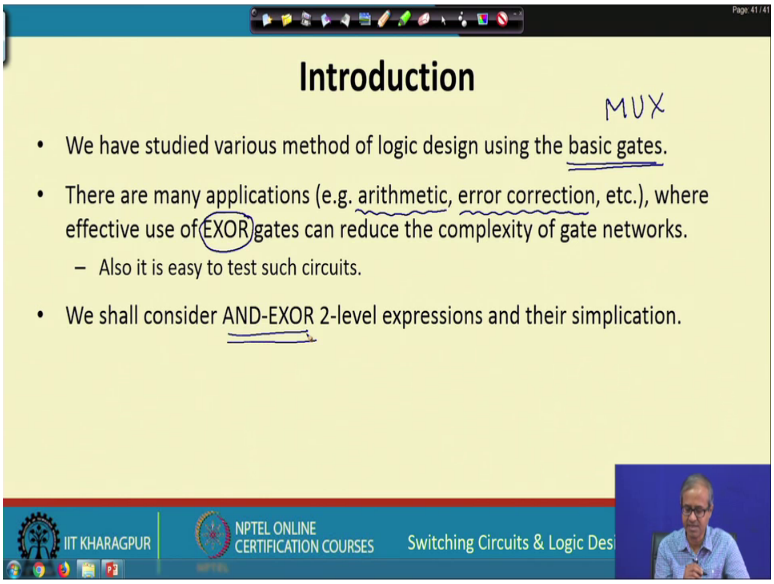
Switching Circuits And Logic Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 29
Logic Design using AND-EXOR Network

So, in this lecture, we shall be looking at some unconventional ways in which a circuit can be designed. Normally when you talk about logic design, the kind of methods that you talked about; we have said that we can use the different kind of gates to implement logic AND, OR, NOT, NAND and NOR gates are functionally complete; we can use them as well. We are also seen various ways in which the basic building blocks in design like multiplexers, decoders, etcetera can be used to implement logic.

Now in this lecture, we shall be talking about something called AND and EXOR network. This is not very conventional. So, we will be using AND gates and EXOR gates to realize logic functions, let us look into this in a little more detail.

(Refer Slide Time: 01:23)

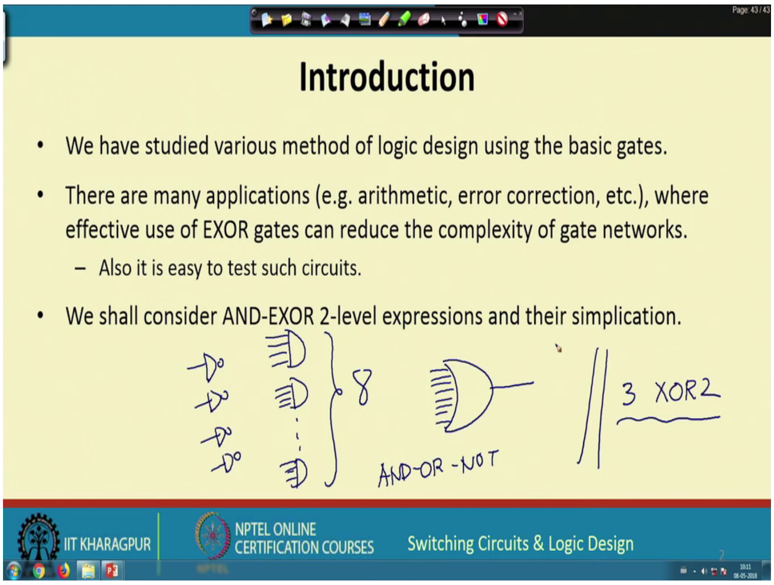


So, what we have just now mentioned is that we have already studied various methods of logic design using basic gates and also using functional modules like the multiplexer, but there are many applications; you think of arithmetic, addition, multiplication, this kind of operations, well, you had looked at the hamming error correcting codes earlier. So, error corrections; there are applications in communication decoding encoding various such applications where we use exclusive or operations very heavily.

So, if we allow not only the basic gates, but also exclusive OR gates to be used in our final circuit, then the size and complexity of the circuit can be reduced or great extent. So, one classic example, if you re call is the circuit will generate the parity of a word simple exclusive or of all the bits will generate the parity that was one example where EXOR gates will be very efficient. Not only that; it is also very easy to test such circuits, we shall consider this AND and EXOR kind of representation in this lecture. Now, one thing I want to just mentioned here that I mentioned the example of a parity generator. Let us take a very simple example. Suppose, I have a 4 bit parity generator and I use an EXOR gate to generate it.

So, I can use a large EXOR gate or I can use a cascade of smaller gates, they are equivalent because normally it is much easier to build smaller gates. So, you see; to generate the parity of 4 numbers bits, I need 3 input EXOR gates, right. So, the hardware is not that much, but now one thing I want to implement; the same thing using AND, OR and NOR gates in a conventional way. So, what will be my hardware complexity 4 input EXOR. So, what this function is I just to recall EXOR is nothing, but count counting or the min terms which corresponds to the odd number of ones. So, in 4 variable function, there will be 8 such combinations where the number of ones will be odd.

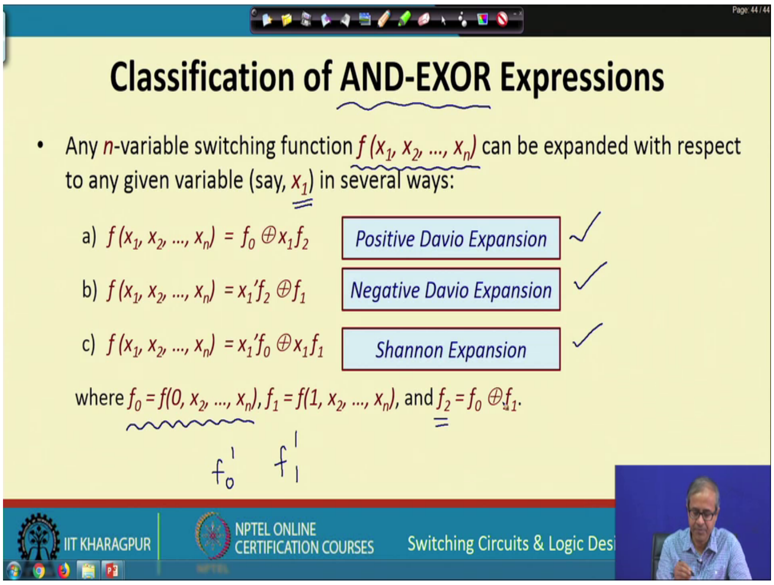
(Refer Slide Time: 04:24)



So, in an AND-OR realization, what will require will be requiring 8 AND gates; each containing 4 inputs each and in the last stage will be requiring a large OR gate with 8 inputs and of course, in the inputs will be using some NOT gates in addition, there will be 4 NOT gates for the 4 inputs.

So, you see will be requiring 8 4 input gates 1 8 input gates and 4 NOT gates for a conventional AND-OR-NOT realization, but if we use EXOR, you need only 3 2 input EXOR gates, you see, this is a classic example which shows, but EXOR gates for some applications can be very very efficient as compared to conventional logic implementations let us move on.

(Refer Slide Time: 05:30)



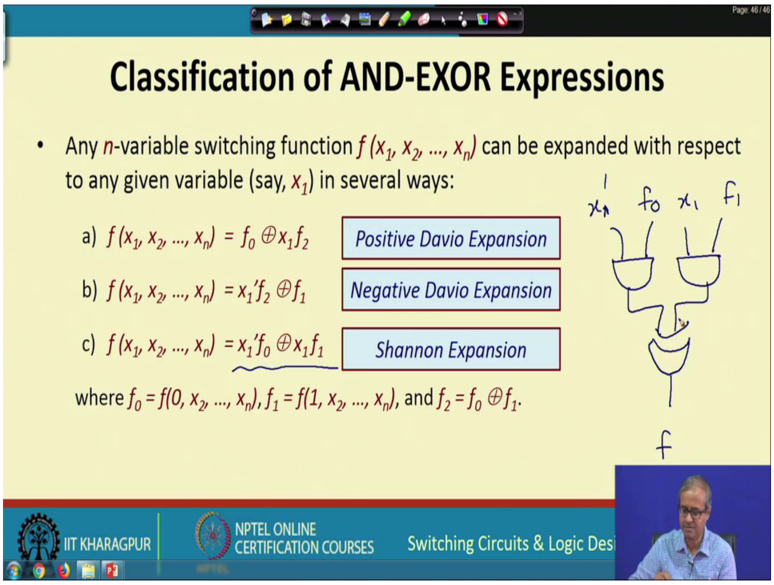
Now, because our subject or discussion in this lecture is and EXOR implementations; let us look at 3 alternatives ways of expansion well, we already talked about Shannon's expansion earlier, let us look at it like in this way consider that we have an n variable function f with variables x on to x n, they can be expanded just like Shannon decomposition or expansion we talked about earlier with.

Respect to any of the variables well in this example, I illustrated with x 1 in 3 possible ways, these are called positive Davio, negative Davio and our familier Shannon's expansion well, here we used 3 notations where f 0 indicates the cofactor of this function f where the input variable x 1 is 0 well earlier, we express it like this f 1 0, but anyway I am showing it only as f 0 and similar f 1 which earlier we showed as f 1 1, but this f 1 means the variable is at one and we introduce another function if 2 which is the exclusive or of f 1 and f 0, right.

Now with this notation, the positive Davio expansion says at the function can be written as f 0 exclusive or x 1 f 2 negative Davio says, it can be expressed as x 1 bar f 2 x or f 1 and Shannon expansion says x 1 bar f 0 EXOR x 1 f 1. Now recall one thing, earlier while talking about multiplexer realization.

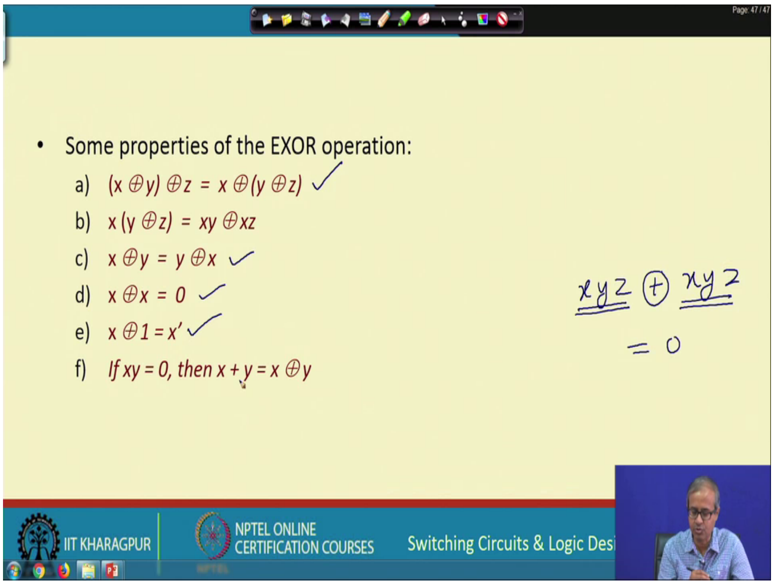
When you introduce the Shannon decomposition theorem you recall we used a or here not on EXOR, but here we are showing an EXOR well, you shall see very shortly that in certain cases, OR AND EXOR can be equivalent this is one such case. So, I can either right or I can write EXOR, they will mean the same thing. So, there are 3 different ways in which I can expand the function you see one thing that if I expanded in this so far example let us say Shannon.

(Refer Slide Time: 08:15)



So, what is this function mean this function means I will be having 2 AND gates 1 x on bar and f 0 1 is feeding with x x sorry x 1 bar other is fed with f 0. The other and gate is x 1 f 1 x 1 and f 1 and it is EXOR of the 2. So, there is EXOR gate. So, you feed this to this and you get the function f this is an AND EXOR kind of realization right.

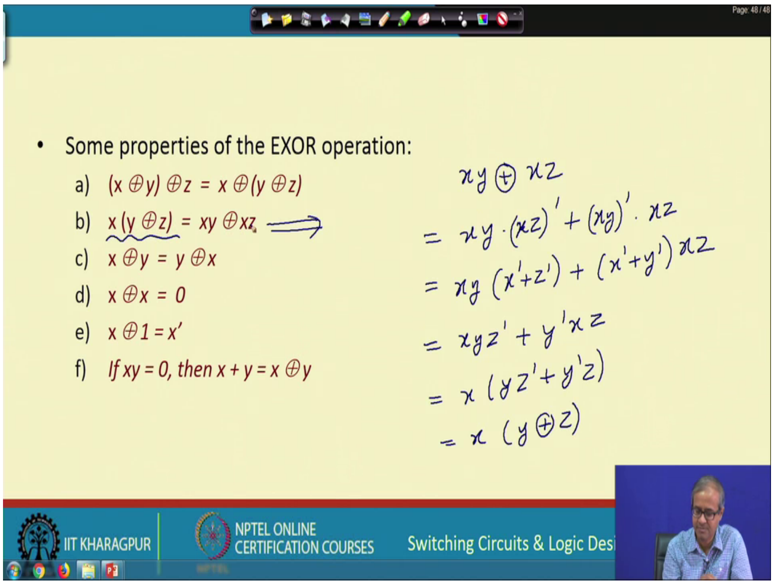
(Refer Slide Time: 08:57)



Now, let us look at some of the properties of EXOR sum of which you already know some of which may not be very apparent, but will see this see sum of the roots you already know the first one that exclusive what is associative. So, whether you take the EXOR of x and y and then EXOR of z or the other way around, it does not make any difference because ultimately EXOR means whether the number of ones is odd or even if it is odd it will be 1, if it is even it will be 0 ok.

The third one is also known commutative whether you take x EXOR y or y EXOR x, it means the something. Now if you take the EXOR of any function with respect to itself or any variable with respect itself it will become 0. This is important like when you have an expression like this, let us say x y z EXOR x y z, it cancels each other this is 0. So, this x need not be a variable only, it can be a function also any function EXOR with itself means 0 and anything EXOR with one means the compliment not right. Now let us look at the other 2 rules which may not be very familiar to you.

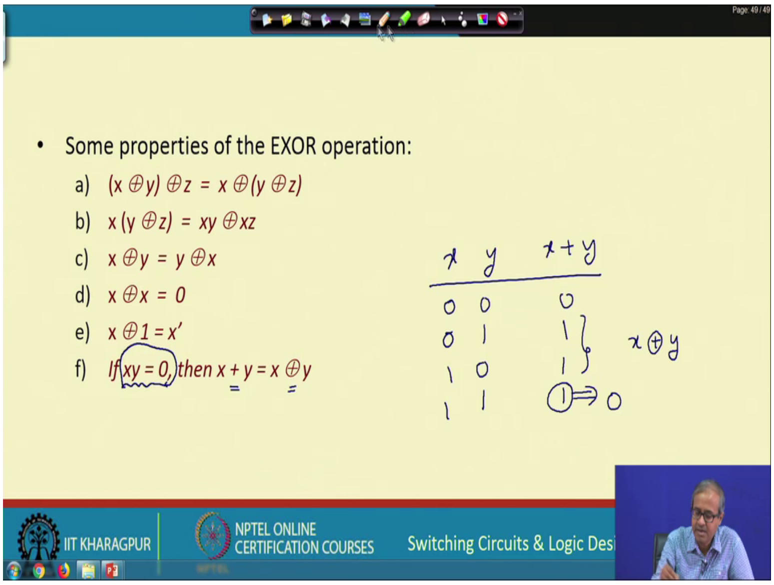
(Refer Slide Time: 10:30)



This one this is some kind of distributive law and an EXOR x and y x or z is equal to this, let us take the right hand side xy EXOR x z. So, let us expand it. So, what is EXOR x y and x z bar or x y bar and x z, this is an EXOR. So, if you apply Demorgan's law x bar or z bar or x bar or y bar x z. Now if you multiply out x y x bar will be 0 x x bar cuts out x y z bar and x bar and x z cuts out x bar x and y bar x z if you take x common y z bar or y bar z which is nothing, but the exclusive or of y and z ok, it will be left hand side.

So, you see that EXOR and they distribute over each other. So, as if you have an and EXOR, you can multiply as if x y EXOR x z, you can do this, right. Now let us look at another interesting rule, it says this x and y, well again, x y need not be variables, they can be any 2 functions.

(Refer Slide Time: 12:18)

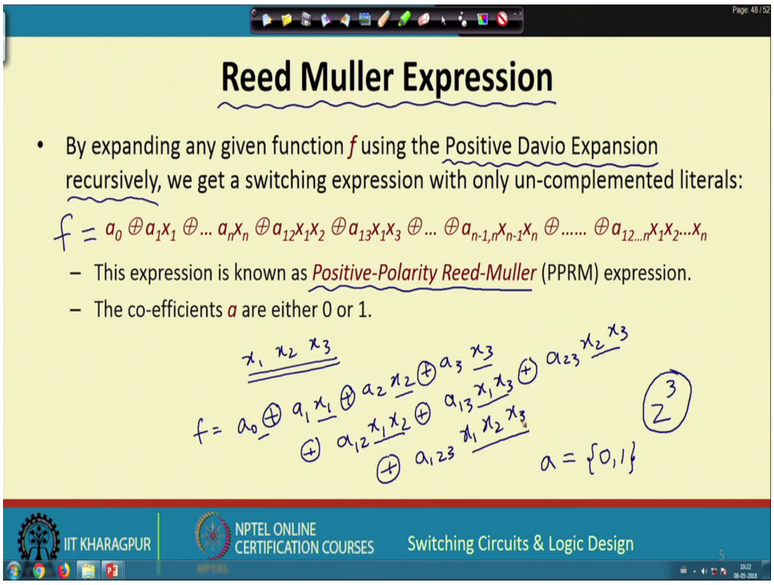


If the end of this 2 is 0, then OR and-EXOR, I can replace their equivalent, well, why it is so? You see; if you look at the truth table of a function; let us say this xy or x or y.

So, if I say EXOR y what is the truth table, this odd says truth table is this, but it addition, I am saying that my x y equal to 0, this condition is true. So, x y 0 means what the end the end of x and y is 0. So, AND means what? X y is the AND so, I am making this is as 0. So, I am left with 0 and 1 0 which is nothing, but x EXOR y so, they are same. So, in any just example, if you see, if you look at the previous example and let us go back to the previous slide, this x bar f 0 x 1 f 1, you see that is x 1 bar and x 1, here if you take and of these 2, iit is 0.

Therefore, this EXOR-AND-OR are equivalent, you can replace EXOR with OR, right. This is the basic idea. So, you remember these rules.

(Refer Slide Time: 14:07)

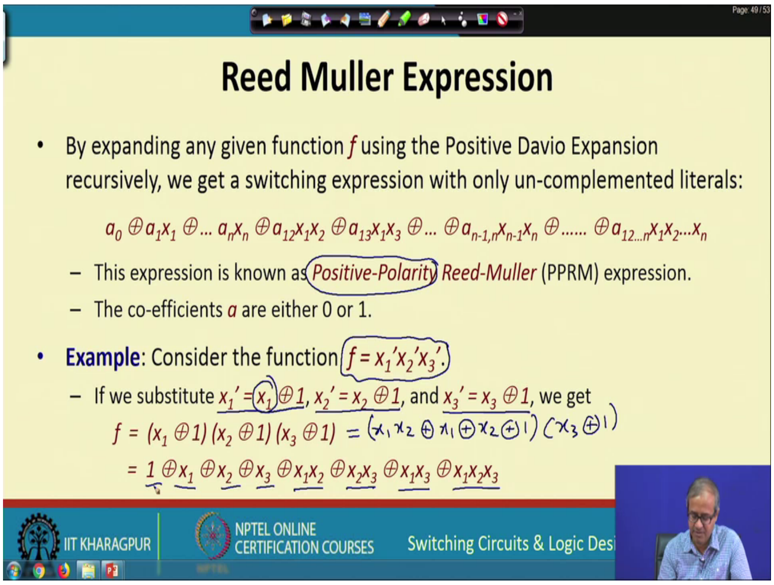


Fine, now let us come to something called Reed Muller expansion. Reed Muller expansion is a classic way of implementing a switching expression using AND and EXOR gates. This was proposed long back, there are many applications of Reed Muller expansion and people have been using it, since, many years many decades. The idea of classic Reed Muller expansion is that, we use positive Davio expansion, let us say to expand a given function repeatedly recursively.

So, if you do that we shall take some examples, later, if we do that then we can get this function f written as EXOR of a number of n terms the may be all possible very; let us say, let us take a very specific example, suppose I have 3 variables x 1, x 2, x 3, this is a general expression that is why this looks complicated for 3 variables. This functional look like this is a 0 plus a 1 x 1 plus a 2 x 2 plus a 3 x 3 not plus using EXOR all EXOR, EXOR a 1 2 x 1 x 2, a 1 3 x 1 x 3, a 2 3 x 2 x 3 and EXOR a 1 2 3 x 1 x 2 x 3. So, you see all possible values of x 1 x 2 x 3 and their combinations and terms are there is only x 1 x 2 x 3 pairs, 2 of them taken together 3 of them taken together and none of them.

So, there are 2 to the power 3; 8 such and terms product terms they are all EXOR together and this a i is this a coefficients, they are either 0 or 1, they may be present in expansion or they may not be present this is actually what is referred to as Positive-Polarity Reed-Mullar expression, Positive-Polarity means all the variables appeared in un complimented forms ok. This is the basic idea.

(Refer Slide Time: 16:59)

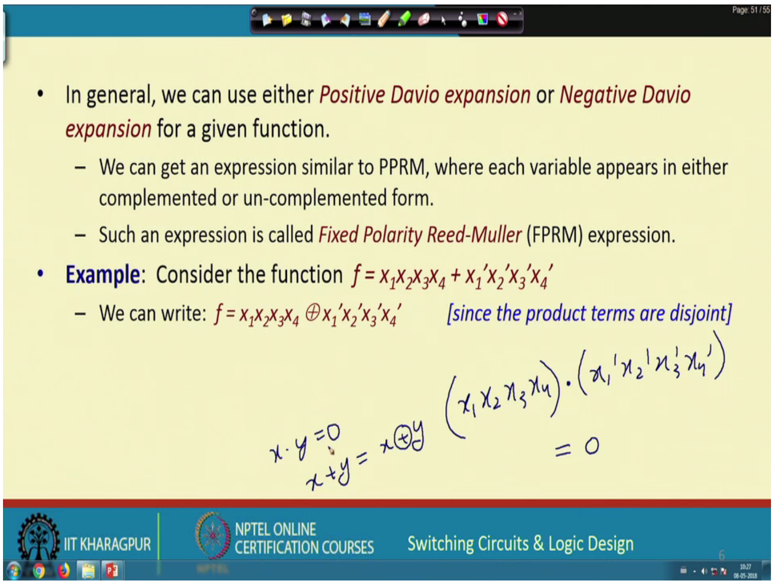


So, let us take an example here, suppose, I have a function like this, I want to express this in the Positive-Polarity form. So, what I do? Well, here I already know the EXOR rules. We saw in the last slide, this x 1 bar; we can write as x 1 exclusive or 1 x 2 bar, we can write like this x 3 bar, we can write like this. Well, why we are writing like this because its bar; we are eliminating; we want to use Positive-Polarity only.

So, this x 1 bar; we have made it x 1 because x or 1 means compliment. So, we have eliminated the bars like this and if you put it here it will be like this and you know that this EXOR is distributive over and you on multiplying. So, I have skipped a step you can do it like this.

The first 2 for example, if you multiply by x 1; so, it will be x 1 x 2 x 1 and 1 EXOR x x 1 1 and x 2 EXOR x 2 1 and 1 EXOR 1, this will be the first term and x 3 and 1 remains. Now if you multiply again like this, you will get you say x 1 x 2 multiply x 3, you get this x 1 x 2 and 1, you get this, then x 1 and x 3, you get this x 1 and 1, you get this x 2 and x 3, you get this x 2 and 1, you get this and finally, 1 and x 3; you get this 1 and 1, you get this, right. So, you see for this function ultimately, whatever you get this is your Positive-Polarity Reed-Mullar form all variables are uncomplimented, right ok.

(Refer Slide Time: 19:05)

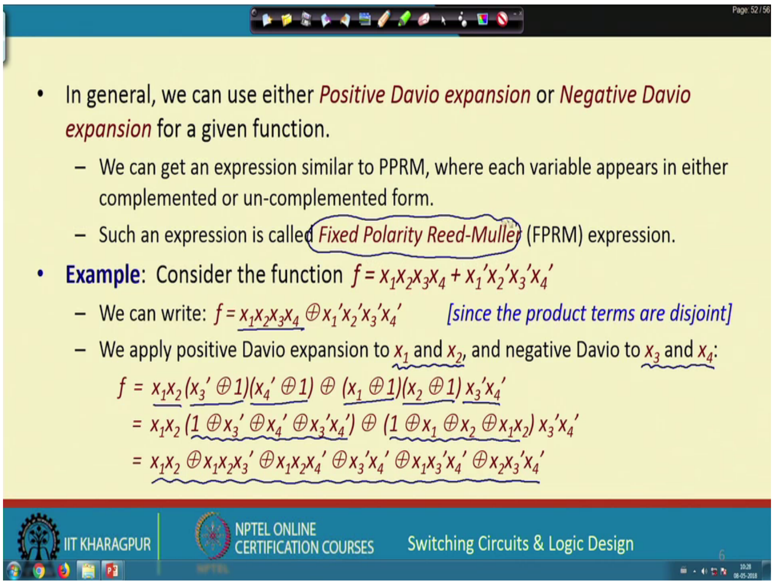


Now, the point is that sometimes, you may not required all in un complimented form. So, you can have or combination of un complimented and complimented form, but you can in force that some variable say x 1 appear in only one form of complementation, let x 1 will always be un complimented x 2 will always be complimented x 3 will be un complimented like that if you have that kind of a restriction in place, then you can have something called fixed polarity Reed-Mullar expansion.

Here, we can use a combination of positive Davio and negative Davio kind of expansions where as I said each variable will appear in either complimented or un complimented form, but not both ok. Let us take an example, take a function like this where there is a term x 1, x 2, x 3, x 4 and there is a term x 1 bar, x 2 bar, x 3 bar, x 4 bar, let us try to arrive at a mixed kind of a Reed-Mullar expansion. For this, the first observation is based on one of the rules, you see this x 1 x 2 x 3 x 4 this term and the other one; x 1 bar x 2 bar x 3 bar x 4 bar, if you take the and of this 2 because there is variable and compliment, they will become 0, this is 0.

So, according to our previous rule, if x y is 0, then you can write x plus y and x EXOR y same ok. So, the original function was or I can replace it by EXOR because their AND is 0 ok, the product terms are disjoint.

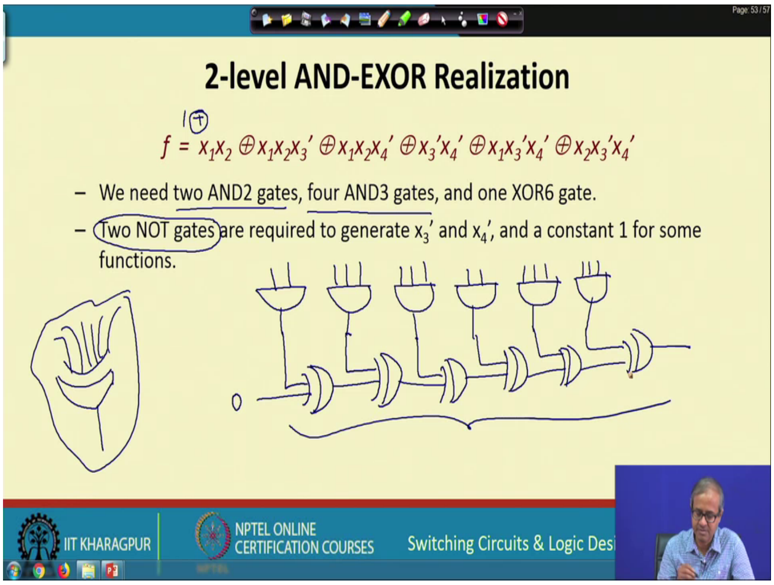
(Refer Slide Time: 21:21)



Now, let us look at this steps of expansion. Now what we are doing? We are applying or we are one thing x 1 x 2 to be in the un complimented form and x 3 and x 4 in the complimented form. So, what we do? So, for the first time because x 1 x 2 are already un complimented, we leave them as it is, but x 3 x 4, you want in complimented form. So, x 3 you replaced by this, we bring in a bar and x 4, we do this, we bring in a bar x 4 bar. Similarly for the other side, x 3 x 4 bar are already bar.

So, we leave them as it is, but x 1 we make it un complimented x 2, we make it un complimented, then using the distributive law, we go on multiplying x 3 plus this and this if you multiply, you get this, if you multiply this and this; you get this. Now straight away, you multiply this with this, you finally, get x 1 x 2 x 3 bar x 1 x 2 x 4 bar like this, you see you get you get a Reed-Mullar expansion where x 1 is un complimented, everywhere x 2 is un complimented, x 3 is always complimented, x 4 is always complimented. So, you have a Fixed Polarity Reed-Mullar expansion where the complementation can be can be either complimented or un complemented, but not both ok, this is what you can have.

(Refer Slide Time: 23:11)

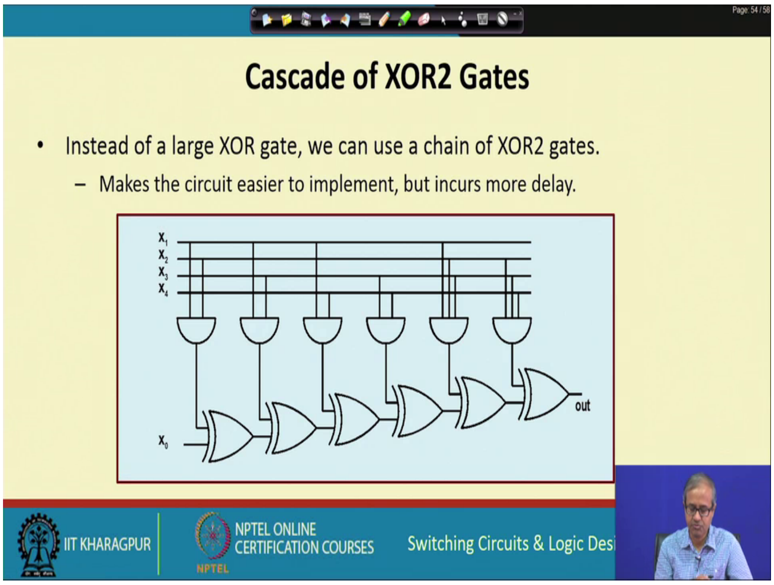


Now, let us talk about 2 level and EXOR realization. So, the function that we just now just arrived at the function is like this. So, in order to implement it what we need; so, how many product terms are there? 1, 2, 3, 4, 5, 6. So, we will be needing 6 AND gates, first AND gate will be having 2 inputs second AND gate will be having 3 inputs, third one also 3, forth one 2, this is 3, and this is 3 and I have to take the EXOR of all of them see, there are 1, 2, 3, 4, 5, 6.

So, one way is to have a large EXOR gate with 6 inputs and connect the output of this AND gates to them directly, but as I said, the large EXOR gate can be difficult to manufacture. So, what we will just another thing is that for the input stages, you may need some complimenting the x 3 x 4, you will be needing 2 NOT gates to compliment x 3 and x 4. So, you need 2 2 input AND gates and 4 3 input AND gates and 1 6 input EXOR gate. Now what I am saying is that in general, you see this is a general expression for some function, you may also have a term 1 EXOR something, right.

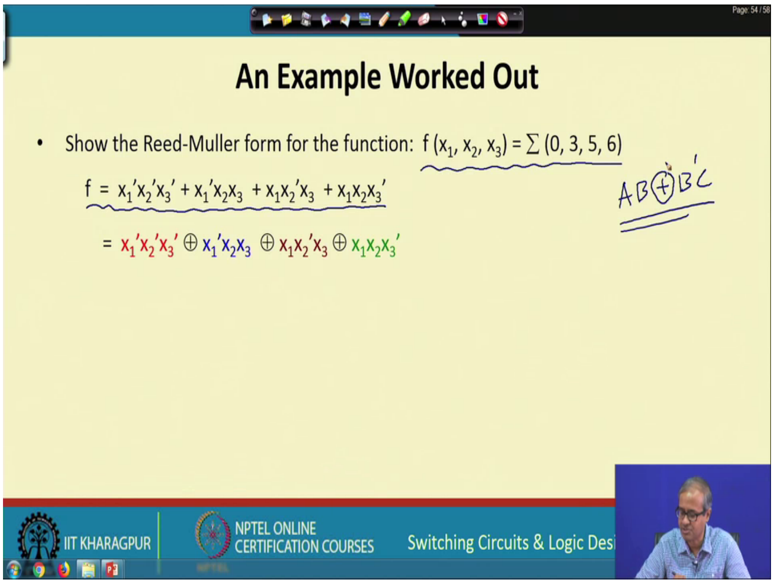
So, general circuit will look like this. So, you can have a chain of 2 input EXOR gates, this is an alternate way of implementing and here, well for this case, you can set it 0, but if it 1, then we will be put it to 1. So, the advantages that you can use small size EXOR gates, but the drawback is that the delay will be more the total delay will be a cascade of the 6 OR gates, right.

(Refer Slide Time: 25:59)



This is what I mentioning that instead of a single logic circuits, we can have a cascade of such gates also right in general, right.

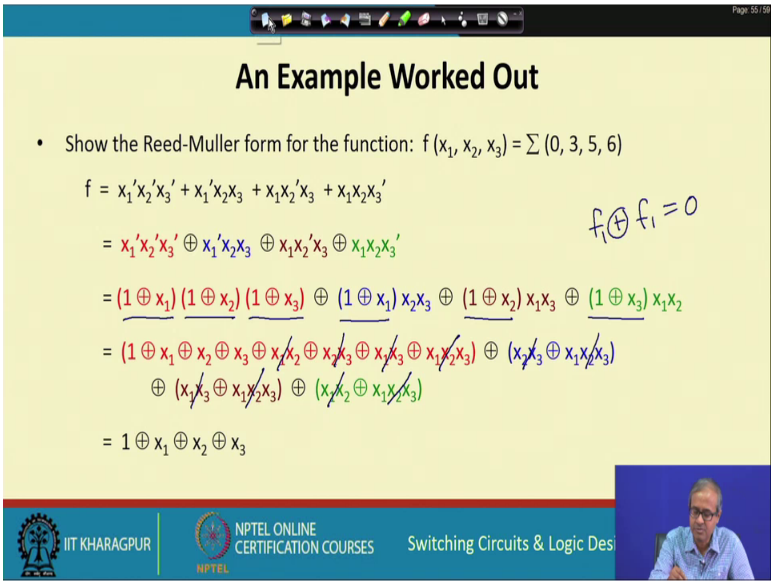
(Refer Slide Time: 26:13)



So, let us work out a complete example here. Suppose, you we want to find the Reed Muller expansion for this function, there are 4 min true min terms 0 3 5 6. So, if you carry out minimization, you will see that your minimize form of the function will be this, there are 4 product.

Now one thing you observe that there is between each pair of product term, if you take and it will be 0. So, this or can be replaced by EXOR that is the idea. So, the first thing is that because they are disjoint this product terms there is nothing in common, but let us say if you have if you had 2 product terms a b or b c, then you could not have done that because a b and b c are something in common a b and b c is not 0, but if you had a b plus b bar c, then you can replace plus by EXOR, right. So, here first step is you replace this or by EXORs.

(Refer Slide Time: 27:35)



Then, suppose I want a positive regular form. So, all variables un complemented. So, wherever that is not; so, x 1 bar, you replace it by this x 2 bar, I replace by this x 3 bar.

Similarly, here x 1 bar x 2 bar and here x 3 bar, then I go on multiplying like in the method, I mentioned earlier, we simply multiply this part, this part is you straight away multiply this red first term gets like this the second term blue, it becomes like this x 2 x 3 or x 1 x 2 x 3, the green term become at this brown term becomes this x 1 x 3 and x 1 x 2 x 3 and EXOR of so many terms. So, now, you can becomes x 1 x 2 and x 1 x 2 x 3 now you see I have an EXOR of so many terms. So, now, you can apply that rule that any function source of function EXOR that same sum function is 0.

So, I can cancel them out. So, let us see which are the terms that are getting canceled out, you say I can get 1 x 1 x 2 x 3 here and 1 x 1 x 2 x 3 here they were canceled out. So, one x 1 x 2 x 3 here and one here they get canceled out this x 1 x 2 and this x 1 x 2 cancels out this x 2 x 3 x 2 x 3 cancels out and finally, x 1 x 3 and x 1 x 3 gets canceled out. So, most of the terms are getting canceled out and what you finally, have his only one EXOR x or EXOR x 2 EXOR x 3. So, the and EXOR you do not need any AND gates for this example you need only x or it is a very simple form of an expansion.

So, with this we come to the end of this lecture you see what we discussed in this lecture are some as I said unconventional ways in which you can represent some functions using and an EXOR gates which show how we can realize any arbitrary function using Reed-Mullar means expression form either in un complemented variables forms are in so called fixed polarity form, some variables are complimented some variables are un complemented.

So, the advantages is that for some functions that can lead to very small realizations and also later on, we will see that when you design a circuit we also may want to test whether the circuit is working correctly or not so far, this kind of Reed-Mullar circuits testing becomes very easy, but this we shall be discussing later.

Thank you.