Switching Circuits and Logic Design

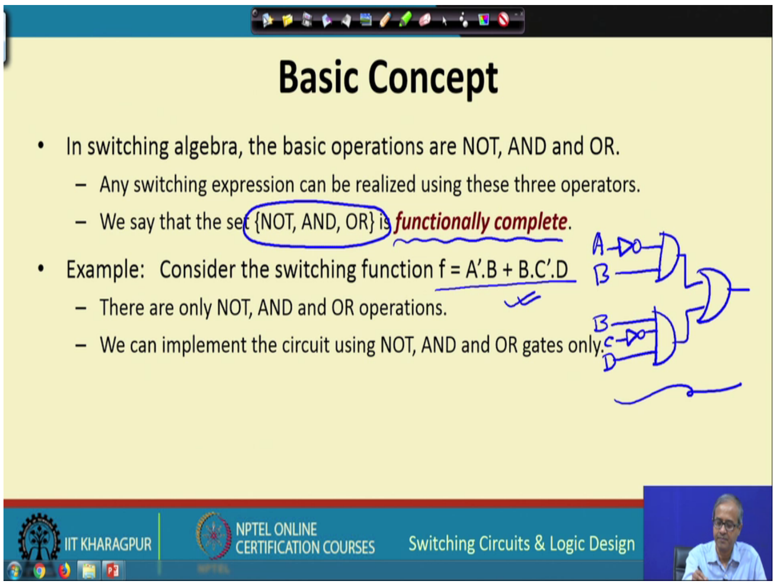
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 15
Functional Completeness

We know talk about something called Functional Completeness. The idea is like this you see we have talked about so many different kinds of gates. Now, if I ask you a question that what kind of gates will be sufficient for designing the circuit; do you need all the different kinds of gates AND, OR, NOT, NAND, NOR exclusive, OR exclusive, NOR everything or even if we have a small subset of gates that will be sufficient to design everything or every circuit. This forms the notion of functional completeness or the universal set of gates.

So, the title of this talk is functional completeness.

(Refer Slide Time: 01:05)



Basic Concept

- In switching algebra, the basic operations are NOT, AND and OR.
  - Any switching expression can be realized using these three operators.
  - We say that the set {NOT, AND, OR} is *functionally complete*.
- Example: Consider the switching function f = A'.B + B.C'.D
  - There are only NOT, AND and OR operations.
  - We can implement the circuit using NOT, AND and OR gates only.

A
B
B
C
D
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Let us understand the notion. Now, in switching algebra whatever we have seen from the basic definition, the operations defined are NOT AND and OR. What does this mean, that any switching expression that you write is just a combination of AND OR and NOT nothing else. So, if I have sufficient number of AND OR and NOT gates available with me that should be enough to design any circuit I want.

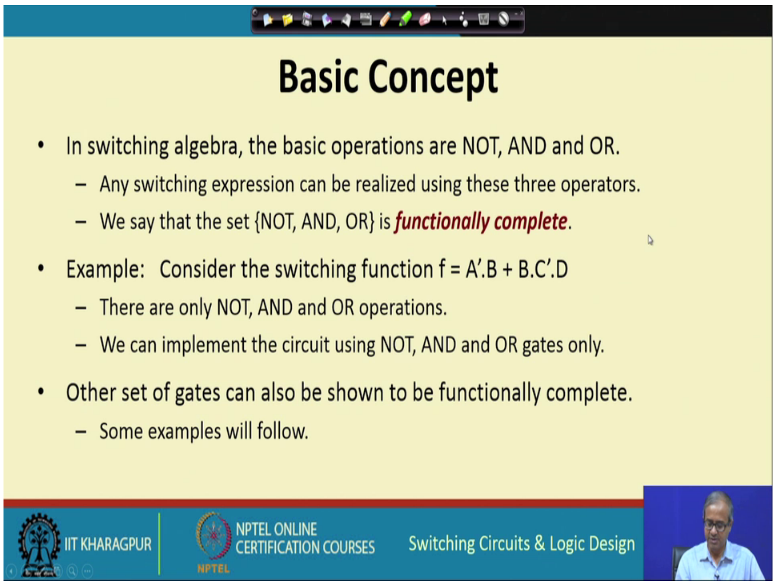
So, this said AND OR NOT is by definition functionally complete because the definition of switching algebra says that with respect to the variables and constants this three operate basic operations are defined AND OR and NOT. So, if you have gates to perform these three basic operations you can realize any switching expressions because switching expressions are formed only using this AND OR and NOT operators. So, the first thing we can say is that NOT AND and OR they are functionally complete.

So, let us take a very simple example say a switching expression f equal to A bar B and B C bar D bar. So, you see that is consist of only not A is not C is not there are some AND and there is a OR. So, if I want to realize it is very simple I need two AND gates, I need to one OR gate, I connect them like this first one is A bar B. So, A I connect using a NOT becomes A bar and B second was a B C bar D, B comes directly C comes through a NOT C bar and D. This is the realization I need only AND OR and NOT. So, given any switching expression we have nothing, but AND OR and NOT in it. So, this set AND OR NOT trivially is functionally complete such a set using which I can realize or implement any function is said to be functionally complete, right.

So, with this notion let us look at what other gates or setup gates they possess this property. But one thing we shall be assuming we shall be assuming that this AND OR NOT is our basic set. So, if I say that some other gate x y z is also functionally complete then you have to show that using x y z I can implement AND, I can implement OR and I can also implement NOT.

So, if I can show that then only I can claim that x y z is also functionally complete fine.

(Refer Slide Time: 04:22)



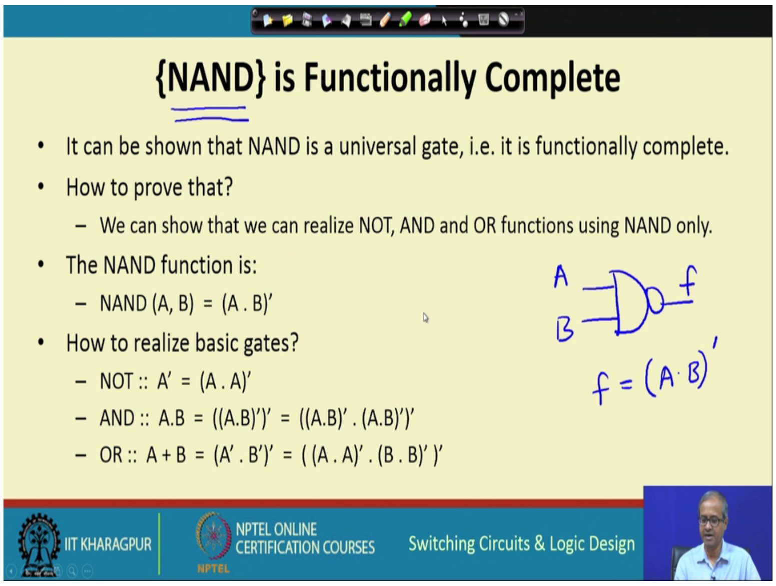
Basic Concept

- In switching algebra, the basic operations are NOT, AND and OR.
  - Any switching expression can be realized using these three operators.
  - We say that the set {NOT, AND, OR} is ***functionally complete***.
- Example: Consider the switching function f = A'.B + B.C'.D
  - There are only NOT, AND and OR operations.
  - We can implement the circuit using NOT, AND and OR gates only.
- Other set of gates can also be shown to be functionally complete.
  - Some examples will follow.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

So, we shall show some examples now.

(Refer Slide Time: 04:26)



{NAND} is Functionally Complete

- It can be shown that NAND is a universal gate, i.e. it is functionally complete.
- How to prove that?
  - We can show that we can realize NOT, AND and OR functions using NAND only.
- The NAND function is:
  - NAND (A, B) = (A . B)'
- How to realize basic gates?
  - NOT :: A' = (A . A)'
  - AND :: A.B = ((A.B)')' = ((A.B)' . (A.B)')'
  - OR :: A + B = (A' . B')' = ((A . A)' . (B . B)')'

A
B
f
$f = (A \cdot B)'$

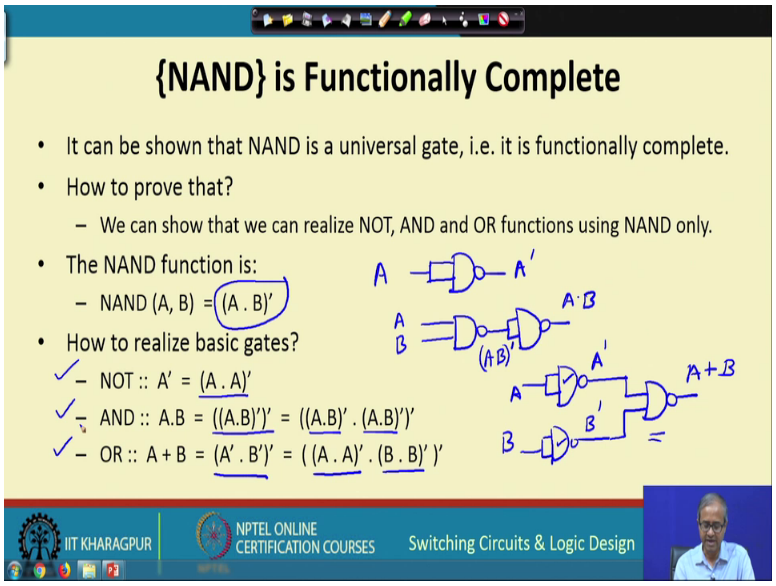
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

The first and most important claim NAND gate; so you know; what are NAND gate is, NAND gate symbolically is shown like this there are two inputs A B and the output let us say if it is f. So, f is nothing, but the AND of the two NOT of that; this is how NAND is defined. Now, what you claim is that NAND is functionally complete what does this mean that if I have only NAND gates then I can build anything.

So, my circuit can consist of NAND gates only no other kind of gates are required ok. So, NAND is some kind of an universal gate this is our claim let us see. So, as I said the NAND function is given by NAND of A B is A and B NOT.

Now, if I want to show that this is functionally complete as I said I have to show that I can realize AND OR and NOT using NAND; let us see how this is possible. NOT just remember this expression NAND of two variable A B is just this A and B NOT. Let us say I take a NAND gate there are two inputs I tie the inputs together and I apply a variable A.

(Refer Slide Time: 05:55)



{NAND} is Functionally Complete

- It can be shown that NAND is a universal gate, i.e. it is functionally complete.
- How to prove that?
  - We can show that we can realize NOT, AND and OR functions using NAND only.
- The NAND function is:
  - NAND (A, B) = $\boxed{(A\,.\,B)'}$
- How to realize basic gates?
  - ✓ NOT :: $A' = \underline{(A\,.\,A)'}$
  - ✓ AND :: $A.B = \underline{((A.B)')'} = ((\underline{A.B})'\,.\,(\underline{A.B})')'$
  - ✓ OR :: $A + B = \underline{(A'\,.\,B')'} = (\,\underline{(A\,.\,A)'}\,.\,\underline{(B\,.\,B)'}\,)'$

A
A'
A
B
(AB)'
A·B
A
A'
B
B'
A+B
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

So, what will be my output? It will be A and A bar this is nothing, but A bar which means I can implement NOT then let us say AND well, if I have a NAND gate with inputs A and B in the output suppose I connect a NOT gate; I already know how to build a NOT gate I connect a NOT gate.

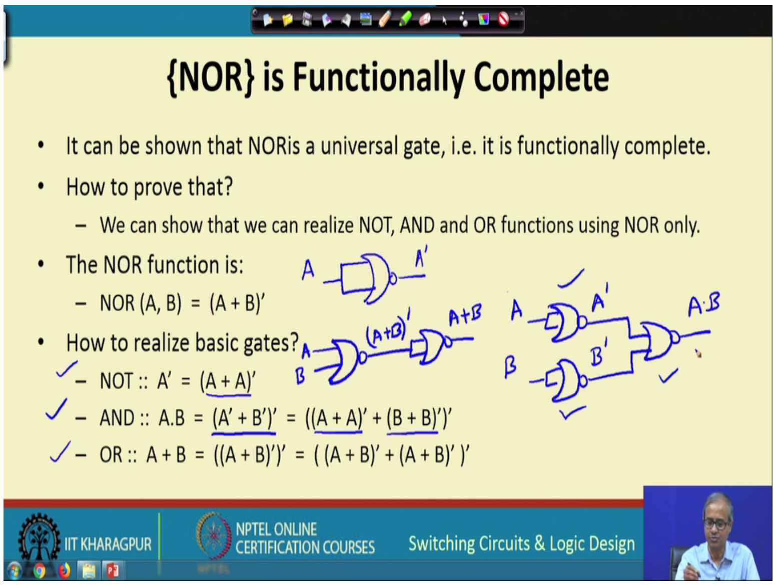
So, this will be NAND NOT of a NAND is nothing, but AND so, I an AND gate you see just algebraically you can write like this AND is A B. So, you can do a double complementation A B bar of bar; now A B bar you can write A B bar and A B bar same thing twice. So, this A B bar is implemented here this is A B bar and A B bar and A B bar this is implemented with the second gate this is basically NOT, NOT of that this implements AND so, you also have a AND.

Well OR is the slightly more complex structure see OR you can implement like this first you need two NOT gates, you apply the two inputs here A and B. So, you have A bar and B bar implemented then you connect these two to another NAND gate. So, what is say is that this will implement OR.

So, the proof is this OR gate by De Morgan's theorem you can write like this A bar and B bar of bar. Now, this A bar is nothing but A and A bar; B bar is nothing but B and B bar. This A and A bar is implemented by this gate, B and B bar is implemented by this gate and whole NAND is implemented by the 3rd gate. So, you can also implement OR.

So, this shows you that NAND gate is functionally complete and any kind of functions switching expression can be realized using NAND gates alone because you have already proved you can design a NOT gate, AND gate and OR gate using NAND only.

(Refer Slide Time: 09:06)



{NOR} is Functionally Complete

- It can be shown that NORis a universal gate, i.e. it is functionally complete.
- How to prove that?
  - We can show that we can realize NOT, AND and OR functions using NOR only.
- The NOR function is:
  - NOR (A, B) = $(A + B)'$
- How to realize basic gates?
  - ✓ NOT :: $A' = \underline{(A + A)'}$
  - ✓ AND :: $A.B = \underline{(A' + B')'} = (\underline{(A + A)'} + \underline{(B + B)'})'$
  - ✓ OR :: $A + B = ((A + B)')' = ((A + B)' + (A + B)')'$

A
A'
A
B
(A+B)'
A+B
A
B
A'
B'
A·B
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

So, continuing with the discussion NOR is also functionally complete just like NAND, the NOR gate is also functionally complete. So, if you have only NOR gates then also you can design and implement any kind of circuit. So, in a similar way to NAND let us also try to proof that NOR gate is functionally complete let see.

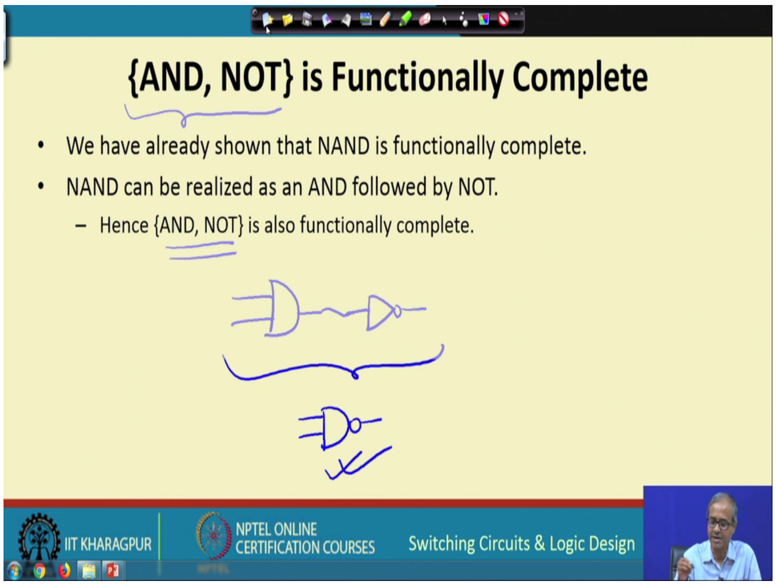
So, the first thing is that the NOR function is nothing, but this A or B NOT of that, now realization of basic gates well in a manner similar to NAND, I am showing here and the corresponding circuits I am showing one by one. First NOT, you take a NOR gate tie the two inputs together just like a NAND if we apply A it will be A or A bar just this A or A means only A. So, the output will be A bar.

So, NOT can be implemented first look at OR because OR is easier you take a NOR gate you apply A and B. So, what you had get here you get NOR of that A plus B bar; now you connect a NOT, NOT you already know how to do a NOT. So, you will be getting finally OR A plus B so, OR can be implemented. Now, AND structure is again similar to OR for a NAND. So, what you do you start with two NOT gates with the two inputs connected together A and B. So, here you have A bar, here you have B bar then you connect them together using another NOR gate you will be getting A and B.

The proof again goes like there from A B by De Morgan's law you can write like this A bar or B bar and this A bar you can write A or A bar B bar you can write B or B bar. This A bar you are implementing by this gate, B bar your implementing by this gate then finally, you are taking NOR of A bar and B bar 3rd gate right.

So, in a manner similar to a NAND we have also shown NOR is also functionally complete. So, if you are given only NOR gate we can implement any circuit that you want to; let us look at some other functionally complete sets also.

(Refer Slide Time: 12:13)



{AND, NOT} is Functionally Complete

- We have already shown that NAND is functionally complete.
- NAND can be realized as an AND followed by NOT.
  - Hence {AND, NOT} is also functionally complete.

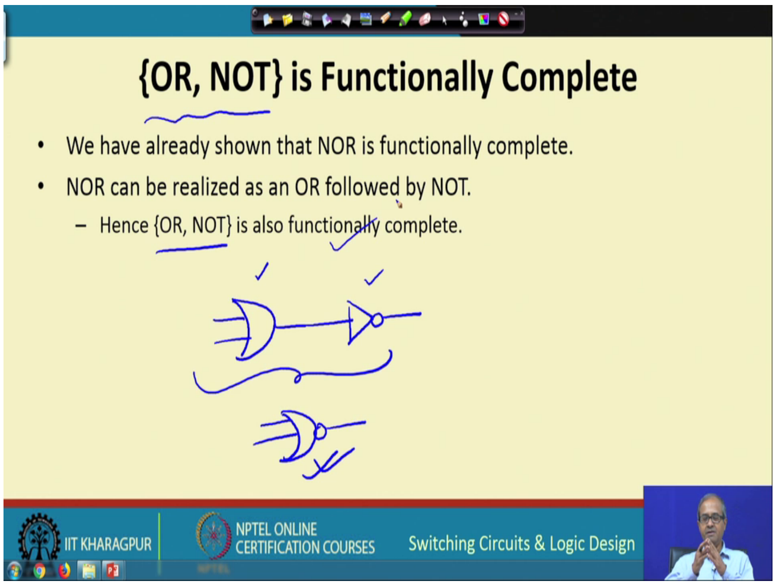
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Well this is quite simple to justify, I am saying that this said suppose I do not have OR gates only AND and NOT.

I am saying this is also functionally complete well what is my justification I have AND gate, I have NOT gate ok. Now, if I connect them together what gate do I get this is equivalent to a NAND gate. Now, now we have we have already proved that NAND gate is functionally complete, now because I can implement a NAND gate using AND and NOT.

So, this will also be a functionally complete AND NOT is also functionally complete ok, simple to say because we have already proved NAND is functionally complete and using AND and NOT I can built a NAND right. So, the justification is very simple.

(Refer Slide Time: 13:23)



{OR, NOT} is Functionally Complete

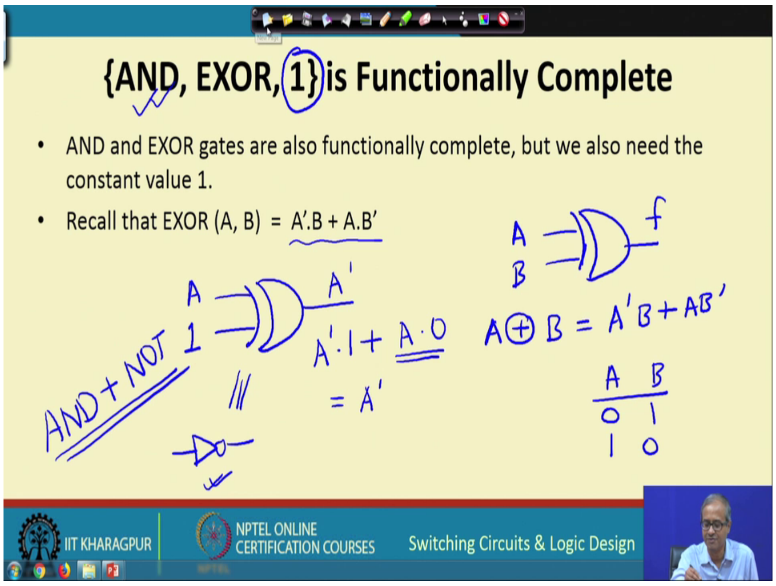
- We have already shown that NOR is functionally complete.
- NOR can be realized as an OR followed by NOT.
  - Hence {OR, NOT} is also functionally complete.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

So, in a similar way you can also say that OR and NOT is functionally complete because you have an OR gate, you have an NOT gate connect them together you get a NOR gate.

Now, NOR gate is functionally complete this we have already proved. So, if I have only OR and NOT. So, we can build a NOR using that which is known to be functionally complete so, this will also be functionally complete. So, we have seen means other than AND OR NOT four sets of gates one is NAND only, NOR only AND NOT OR NOT ok. Now, these gates are functionally complete means without any external means other kinds of inputs because you can realize any function just by using these gates.

(Refer Slide Time: 14:35)



{AND, EXOR, 1} is Functionally Complete

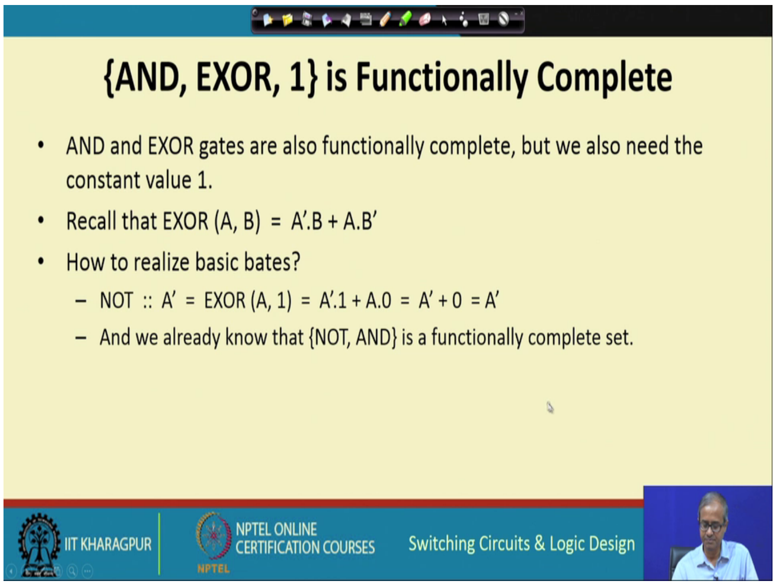
- AND and EXOR gates are also functionally complete, but we also need the constant value 1.
- Recall that EXOR (A, B) = A'.B + A.B'

So, what I mean here is I will be explaining that with the next example. Here what you are saying is there is another set we are talking about we are talking about AND EXOR; well EXOR gate you already know what an EXOR gate is. EXOR gate symbolically is shown like this it two inputs are A B if the output is f well sometimes the EXOR is express symbolically written like this A EXOR B is written as A bar B or A B bar. Because the definition of EXOR I gave earlier is EXOR will be 1 if odd number of inputs are 1; odd number means with respect to A B if A is 0, B is 1 or A is 1, B is 0 odd means one single 1 that is odd so, A bar B A B bar right.

Now, here what you saying that only AND EXOR is not sufficient, we also need the constant value 1. Why? The reason is very simple suppose I have an EXOR gate, EXOR function already I have shown what is EXOR function. Let us say to the first input I apply A, to the second input I apply a constant 1. So, what will be my output just follow this rule A bar B, B is here 1 OR A B bar; B bar will be 0 bar of 1. So, second term will become 0 only the first term will remain which is A bar, which means an exclusive OR gate with one of its input as 1 is equivalent to a NOT gate.

So, I can built a NOT gate using an exclusive OR gate and the constant 1 and I already have AND gate and we have already shown that AND and NOT are functionally complete. So, I have AND and I also have seen how to build a NOT. So, this will also be functionally complete right. So, the justification is fairly straight forward.

(Refer Slide Time: 17:20)



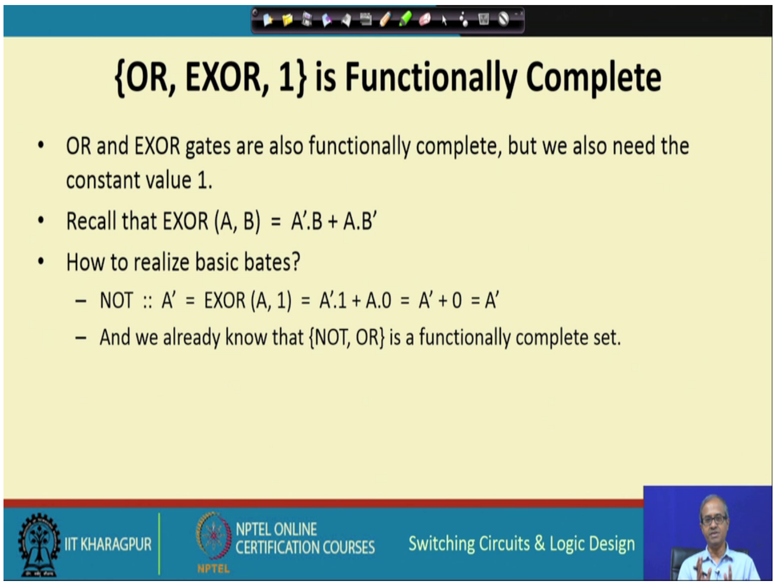
{AND, EXOR, 1} is Functionally Complete

- AND and EXOR gates are also functionally complete, but we also need the constant value 1.
- Recall that EXOR (A, B) = A'.B + A.B'
- How to realize basic bates?
  - NOT :: A' = EXOR (A, 1) = A'.1 + A.0 = A' + 0 = A'
  - And we already know that {NOT, AND} is a functionally complete set.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

This is exactly that is said that you can implement a NOT by setting the 2nd input to 1 and we already have shown earlier NOT and AND is functionally complete.

(Refer Slide Time: 17:35)



{OR, EXOR, 1} is Functionally Complete

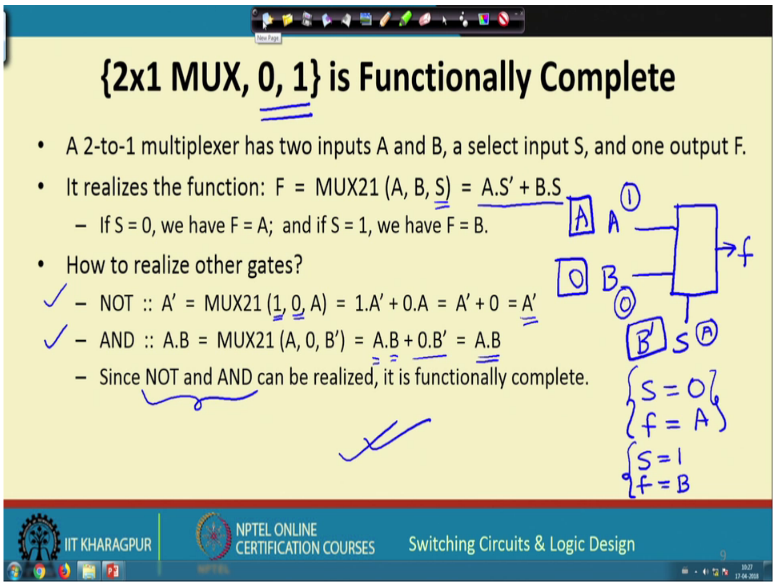
- OR and EXOR gates are also functionally complete, but we also need the constant value 1.
- Recall that EXOR (A, B) = A'.B + A.B'
- How to realize basic bates?
  - NOT :: A' = EXOR (A, 1) = A'.1 + A.0 = A' + 0 = A'
  - And we already know that {NOT, OR} is a functionally complete set.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Switching Circuits & Logic Design

Similarly, instead of AND you can go for OR the logic is similar. So, I am not going into detail because NOT we have seen exclusive OR with a 1, you can implement a NOT and this OR gate is already there OR and NOT we have already shown earlier this is already functionally complete.

Therefore OR, EXOR, 1 will also be a functionally complete set. So, these are the things we have to remember. So, if you are designing any circuit and someone gives you some gates to design you can check whether those gates are sufficient in terms of functional completeness. If there functional complete then you know that well I can design any function using those gates only ok, all right.

(Refer Slide Time: 18:34)



{2x1 MUX, 0, 1} is Functionally Complete

- A 2-to-1 multiplexer has two inputs A and B, a select input S, and one output F.
- It realizes the function: F = MUX21 (A, B, S) = A.S' + B.S
  - If S = 0, we have F = A; and if S = 1, we have F = B.
- How to realize other gates?
  - NOT :: A' = MUX21 (1, 0, A) = 1.A' + 0.A = A' + 0 = A'
  - AND :: A.B = MUX21 (A, 0, B') = A.B + 0.B' = A.B
  - Since NOT and AND can be realized, it is functionally complete.

A
B
S
f
S = 0
f = A
S = 1
f = B
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Let us now look at there are slightly different kind of a functional block not exactly a gate, well we had talked about the multiplexer earlier. So, we considered a 2 line to 1 line multiplexer. So, how does a 2 line to 1 line multiplexer look like; it is the blocks where there are two inputs A and B. There is a select input S and there is an output let us say f, this is how a multiplexer look like.

So, how does it work if the select input is 0 then the output will be equal to A, the first input will be copied to the output, but if S equal to 1 then the second input will be copied. So, it is like a controlled switch depending on what I apply to S either A switched or B switched ok. Let see that whether this is functional here what means we have mentioned nothing that I need both constant 0 and constant 1 to make it functionally complete.

So, talking about the function just you see I can directly write down this expression. The output function of a multiplexer is A S bar B plus S bar for what is the meaning you see if S is 0 output is A, S is 0 means S bar and A, S bar and A. S is 1 means S and B, S and

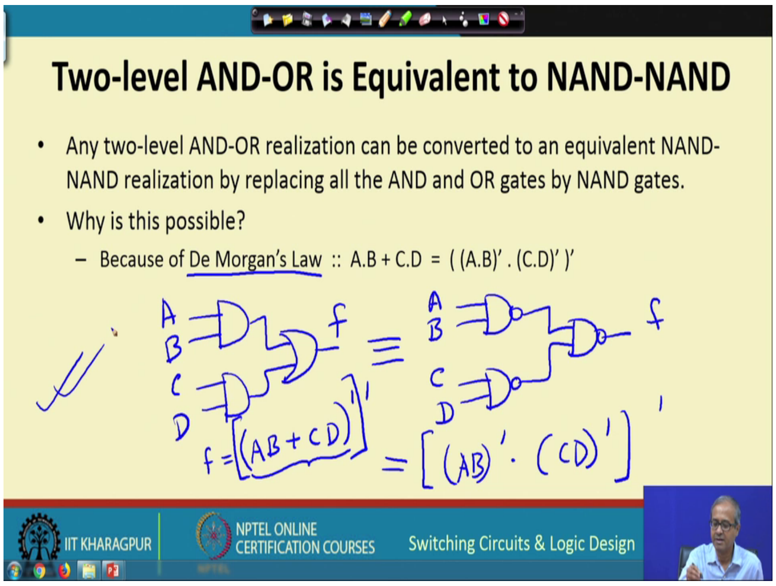
B you can directly write an expression like this ok, this is what a multiplexer is. Now, realization of the other gates can be followed like this just look at these scenarios; first NOT, let us try to build a not.

So, what we are saying that the first two inputs I apply 1 and 0 like in this multiplexer I apply a 1 here, I apply a 0 here let us say and to the select line third-one is a select line, select line I apply the variable A. So, if I substitute 1 0 and A in this expression it becomes A that is 1 A bar plus B B is the second-one 0 A.

If you simplify this is only A bar which means I can implement NOT, but I require a constant 1 and also a constant 0. I can also implement an AND how I can apply this input A in the first-one 0 in the second-one and I already know how to implement a NOT.

So, suppose the other input I do a NOT and I apply B bar to the select line A 0 and B bar then if I just substitute the values in this expression A bar of this becomes B plus second-one 0. So, this will become 0 and A B I have implement AND. So, I can implement a NOT and an AND and you already have seen earlier that NOT and AND are functionally complete. Therefore, this 2 to 1 multiplexer is also functionally complete provided you have the constants 0 and 1 available with us right.

(Refer Slide Time: 22:30)



Two-level AND-OR is Equivalent to NAND-NAND

- Any two-level AND-OR realization can be converted to an equivalent NAND-NAND realization by replacing all the AND and OR gates by NAND gates.
- Why is this possible?
  - Because of De Morgan's Law :: A.B + C.D = ((A.B)' . (C.D)')'

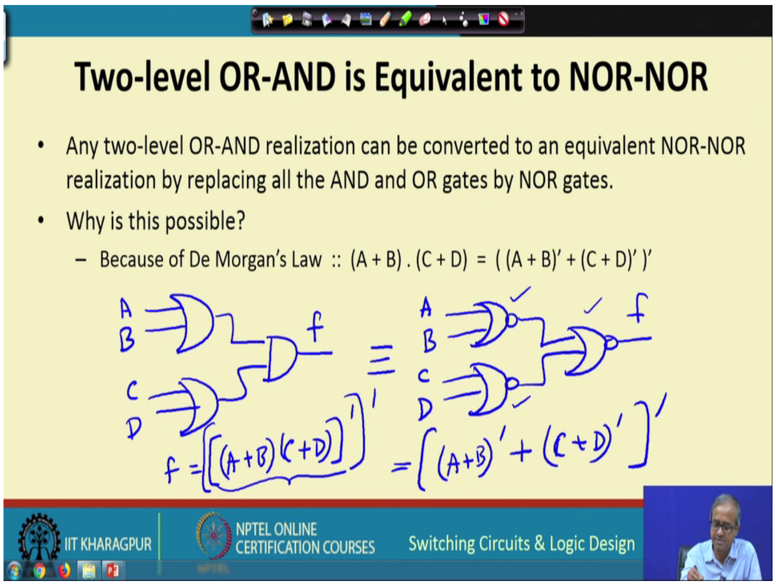
Now, let us look at some interesting observations. So, when you design a circuit sometimes we need to transform a circuit from one form to another, change the kind of gates that you have. So, let us look at somehow the very common transformations that we carry out.

First observation that you make is very important it says that two level AND OR circuit is equivalent to a two level NAND-NAND circuit. So, what I mean is that if I have a two level circuit AND OR like this; this observation says this will be equivalent to keep the circuit structure the same just replace all the gates by NAND. So, function will remain the same. You keep the inputs A B C D has as same, here also it will be A B C D; suppose the function was f here the same function will remain f here. Now, the proof comes from De Morgan's law just you see if I have this A B or C this function f, this realizes what A B or C D. Now, I can do a double complementation complement of complement in two steps let us say.

So, the inner one I can apply De Morgan's law something or something bar what does this mean; this something will be NOT plus will become AND. This something will become NOT and then whole thing I have bar. See AB bar is nothing, but NAND of A B CD bar is nothing but AND of CD and something and something bar is nothing, but NAND of that something and something. So, straight away these two are equivalent.

So, if I have a two level AND OR circuit just blindly I can replace all the gates by NAND gates, not necessary two gates any number of gates can be there in the first level because De Morgan’s law can be extended to any number of inputs right.

(Refer Slide Time: 25:20)



Two-level OR-AND is Equivalent to NOR-NOR

- Any two-level OR-AND realization can be converted to an equivalent NOR-NOR realization by replacing all the AND and OR gates by NOR gates.
- Why is this possible?
  - Because of De Morgan’s Law :: $(A + B) . (C + D) = ((A + B)' + (C + D)')'$

$f = [[(A+B)(C+D)]']' \equiv \;= [(A+B)' + (C+D)']'$

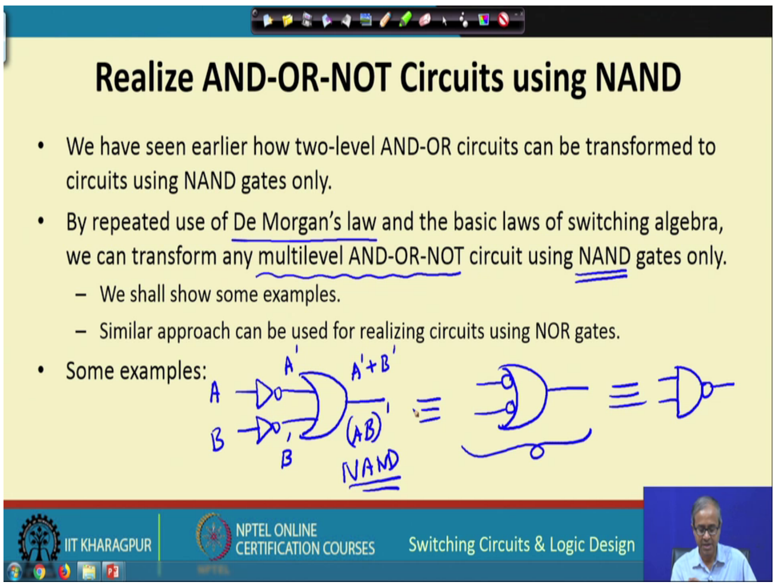
A
B
C
D
f
A
B
C
D
f
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

This is one very important observation you should remember and a similar observation exists for OR AND circuit. Two level OR AND circuit is equivalent to NOR-NOR. So, here also the idea is very similar I have a two level OR AND circuit like this.

So, here also lets say the inputs are A B C D what I say this is equivalent to a stimulus circuit where all the gates are made NOR. So, again this follows from De Morgan's law because if you say the function output is f. So, what is f f will be A or B and C or D. So, if you do it double complementation again a complementation then the inner one something and something bar what will be that something bar or this something bar whole of bar.

So, this means this A plus B bar means this NOR C plus D bar means this NOR and whole thing means this NOR. So, similarly two level OR AND you can directly replace by two level circuit with all NOR gates right. This is very simple.

(Refer Slide Time: 27:08)



Realize AND-OR-NOT Circuits using NAND

- We have seen earlier how two-level AND-OR circuits can be transformed to circuits using NAND gates only.
- By repeated use of De Morgan's law and the basic laws of switching algebra, we can transform any multilevel AND-OR-NOT circuit using NAND gates only.
  - We shall show some examples.
  - Similar approach can be used for realizing circuits using NOR gates.
- Some examples:

A
B
A'
B'
A'+B'
(AB)'
NAND
≡
≡
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

Now, in general if you have means any circuit not necessarily two level right when is it can be a multi level circuit. I mean if you apply De Morgan's law in a suitable way and also if you use some basic transformation rules then you can convert any multi level circuit consisting of AND OR and NOT gates using NAND gates only.

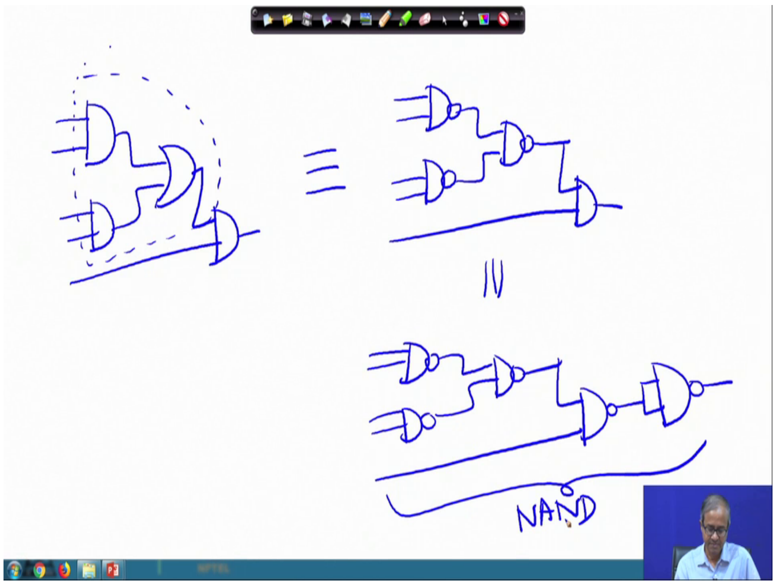
Well, I shall be illustrated in with some examples well similarly, we can do it for NOR gates only, but I am not giving those examples you can try them out yourselves, but what examples I will be giving is given any arbitrary circuit I can convert them into NAND gates, it can be multi level not necessarily two level. But some basic rules are being followed here.

Let us say if I have an OR gate with NOT gates in the input, well sometimes symbolically we write it like this we show it as small bubbles in the input, bubble means NOT these are same thing. So, if you see what this functionally means A B means this will be A bar B bar and OR of that A bar or B bar. So, what is A bar or B bar according to De Morgan's law it is nothing, but A B bar which means this is NAND.

So, any OR gate with a not gates in the input is equivalent to a NAND this you should remember this is nothing, but NAND in some book we will find NAND gates are shown like this because they are equivalent right.

Let us now take some examples to show that how this kind of transmissions can be carried out; just a second.

(Refer Slide Time: 29:40)

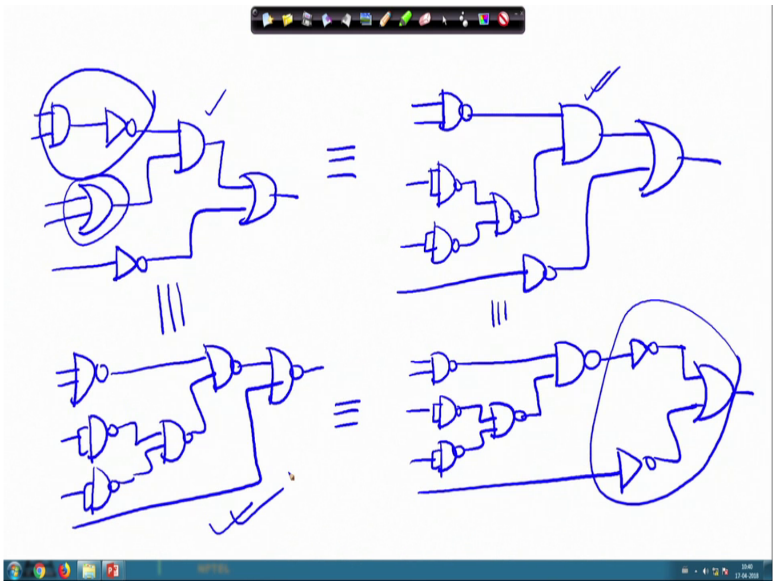


Let us take a simple example. Let us say I have a circuit like this. So, I want to convert it into a pure NAND-NAND circuit it is a multi level circuit. So, there is a AND OR and also there is another AND let us say circuit is like this. So, what are the steps? The first step is that if we look at this part of the circuit separately this part of the circuit, this is like a two level circuit AND OR and two level circuit you already know that we can converted into NAND-NAND blindly because they have already shown that these are equivalent.

Then lastly we have an AND gate here now, what is an AND gate we can write it like this; these are already NAND we have got in NAND and this AND gate you can implement using a NAND gate followed by a NOT gate. So, you know how to implement a NOT using NAND.

So, you see you have obtained a all NAND implementation of the original circuit right simple.

(Refer Slide Time: 31:17)



Let us look at another example which we will involve some OR operations also; suppose I have a circuit like this with a NOT gate here this goes to an AND. There is an OR gate here output of the OR goes here, there is another OR gate and there is a NOT; let us take a circuit like this.

So, step by step let us go through the transformation. First is if you look at these two gates and followed by NOT this is straight forward it is a NAND. You replace it by a NAND and you have an OR gate here. Now, you already know how to implement a OR gate using NAND gates follow that rule; you have a NOT, another NOT, you have a NAND. This is NOR, this NOR gate and both of these will be going to the input of this

AND gate, this AND gate and then you have the final OR gate and the other input coming through a NOT.

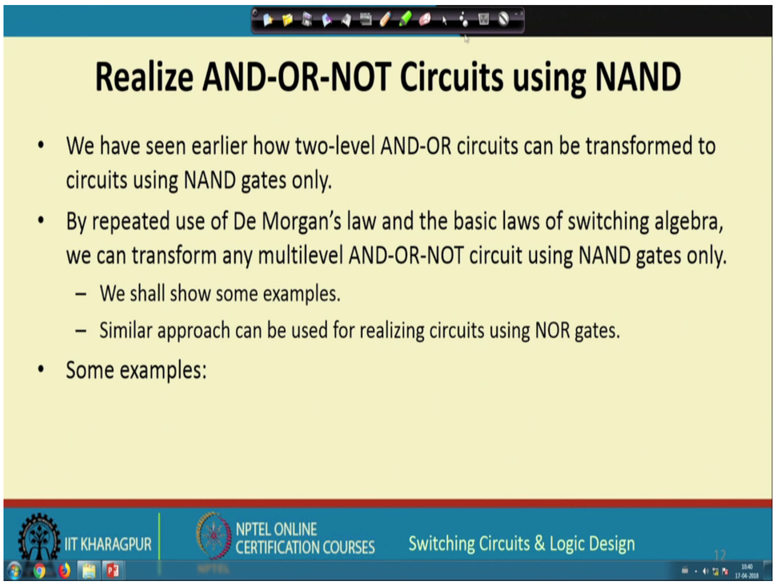
So, the first two part you have taken care of converted them to all NANDS's now, let us look at the rest. This is a this is an AND gate. So, what you can do I am just showing the equivalent circuit here this circuit is already there, I am also drawing this part; this part is there and they are going to the input of this AND gate ok.

Now, what I am saying is that let us make this a NAND gate followed by a NOT gate. Let us say a NOT gate this is equivalent to AND and on the other input also I have NOT gate which goes to an OR gate. Now, we have already shown that a NOR gate with the inputs NOT is equivalent to a NAND gate.

So, finally, you have a circuit where you have this NAND gate this two NOT gates using NAND, another NAND they are connected to another NAND and finally, they are going to the last NAND with the other input is straightaway coming here. So, these two circuits are equivalent.

So, you see just by knowing some of the rules that you have already know and this kind of transmissions that I just talked about some time back you can convert any arbitrary circuit using systematic methods using NAND gates only. Similarly, you can do it for NOR gates also the rules are pretty similar.

(Refer Slide Time: 34:58)



Realize AND-OR-NOT Circuits using NAND

- We have seen earlier how two-level AND-OR circuits can be transformed to circuits using NAND gates only.
- By repeated use of De Morgan's law and the basic laws of switching algebra, we can transform any multilevel AND-OR-NOT circuit using NAND gates only.
  - We shall show some examples.
  - Similar approach can be used for realizing circuits using NOR gates.
- Some examples:

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Switching Circuits & Logic Design

So, with this we come to the end of this lecture. Now, in our next lectures we shall be looking at some systematic ways of minimizing functions. So, far we talked about algebraic methods and just by applying rules by using Ad hoc methods we are try to minimize; means as I said if you remember the rules correctly to be easier for you. But if you do not remember the rules you may not be able to minimize in an effective way.

So, we shall next look at some of the more systematic ways to minimize and analyze switching functions.

Thank you.