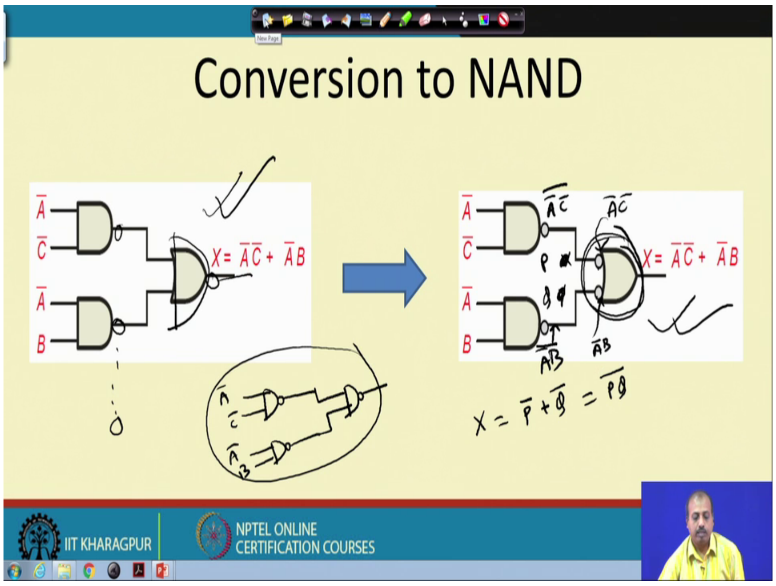
# Digital Circuits
**Prof. Santanu Chattopadhyay**
**Department of Electronics and Electrical Communication Engineering**
**Indian Institute of Technology, Kharagpur**

## Lecture - 16
## Logic Gates (Contd.)

Sometimes we need gates some circuit to be converted into NAND gate circuit, because as I said that NAND is an universal gate so many logic libraries. So, when you are realizing some circuit digital circuit may be only NAND gates are available. So, there are many I integrated circuit chips which has got only NAND gates in them.

So, if we have to realize some circuit using those chips then you have to realize using NAND gates only. So, but normally when you do a Boolean minimization we come up with some sum of product or product of sum expression which consists of a say, AND and OR gates and inverters like for example, in this particular circuit.

(Refer Slide Time: 00:54)


### Conversion to NAND

X= $\bar{A}\bar{C}$ + $\bar{A}$B
X = $\bar{p}+\bar{q}$ = $\overline{pq}$
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, ultimate expression that I have got is X equal to A bar C bar plus A bar B. Now this A bar C bar plus A bar B; so it means that we have got this is the OR and then these are the 2 AND gates.

But if I do not have this AND and OR gates. So, I have got only NAND gates then you can do it like this. So, what we do? This A bar C bar so, this AND gate is converted into

NAND gate. So, this is the NAND operation, and this is so, this A bar B so, it was a NAND gate here we convert it into NAND gate. And after that so, there is another bubble here. So, as so, we can understand that this is converting this a here you are getting A bar C bar whole bar. So, at this point you are getting A bar C bar whole bar. And after this after this inversion so, you are getting A bar C bar.

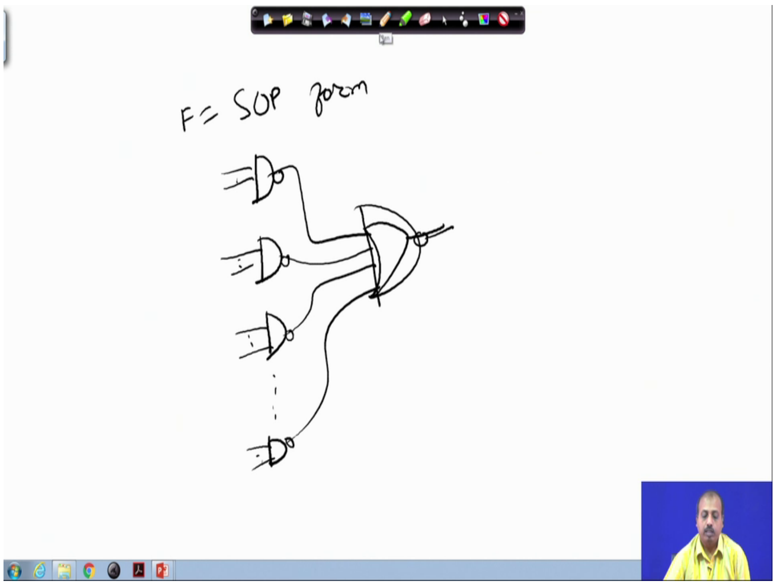
Similarly, at this point at this point you are getting A bar B, and after this bubble here second bubble there. So, you are getting so, sorry, evaluating A bar B whole bar. So, here you are getting A bar B and then as if this or and that is A bar C bar plus A bar B. But what is this gate? Ok so this, gate if I say that this input is X and this input is Y. So, what I am realize sorry this in say instead of X and Y let us write something else.

So, let us say this is P and this is Q. So, what we are getting is X equal to P bar or Q bar because P is inverted here Q is inverted here, and then they are OR'ed. So, if you apply De Morgans theorem so, this is nothing but PQ bar so, this gate is nothing but a NAND gate. So, we you can simplify that drawing as if that this, you can simplify the whole drawing like this that this is a one NAND gate, where we have got this A bar and C bar as inputs and you have got another NAND gate where you have got this A bar and B as a input.

And then you have got here another NAND gate, where these 2 are connected ok. So, this diagram that we have here so, this is actually for the sake of our understanding, but we must appreciate that this is equivalent to this, this is equivalent to this which is convert. So, suppose I have got a sum of product expression from there I have got this realization of the function in terms of AND and OR gate. So, for convert into NAND gate what you have to do is that whatever AND gates you have here. So, convert or if there are many such AND gates many such some product terms.

So, you convert all those ands to NANDs and when this and then from then finally, convert this OR gate to a NAND gate so, that we will solve our purpose ok. So, I will just so, in general I can say that in a sum of product expression so, f is in sum of SOP form.

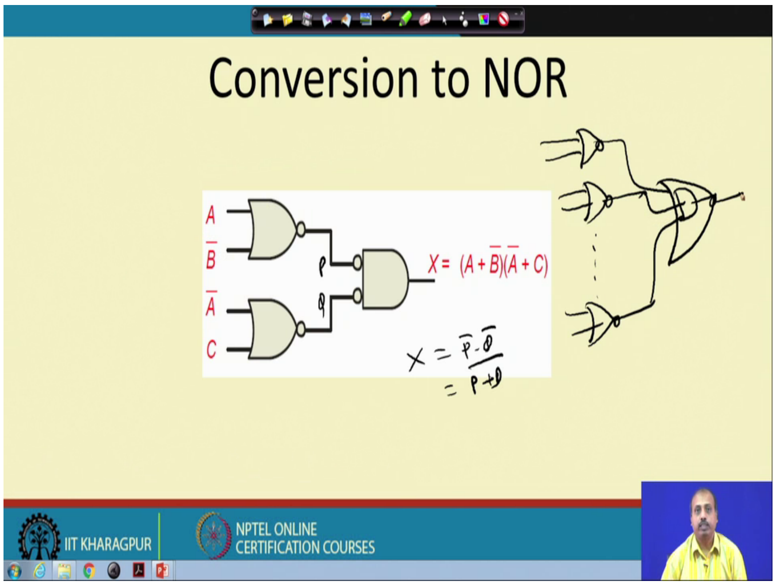
(Refer Slide Time: 04:02)



So, f is in SOP form, then the thus individual sum terms so, they were represented by this AND gates, these were the individual sum terms they were coming there, and then it was feeding the OR gate so, this was the situation.

So, to convert to NAND what you do is you convert each of them to NAND each of them to NANDm and finally, convert this OR gate also to a NAND, ok. So, this way we can convert any Boolean circuit into a circuit consisting of NAND gates, ok. So, next we will be looking into how to convert to nor gate.

(Refer Slide Time: 04:55)



## Conversion to NOR

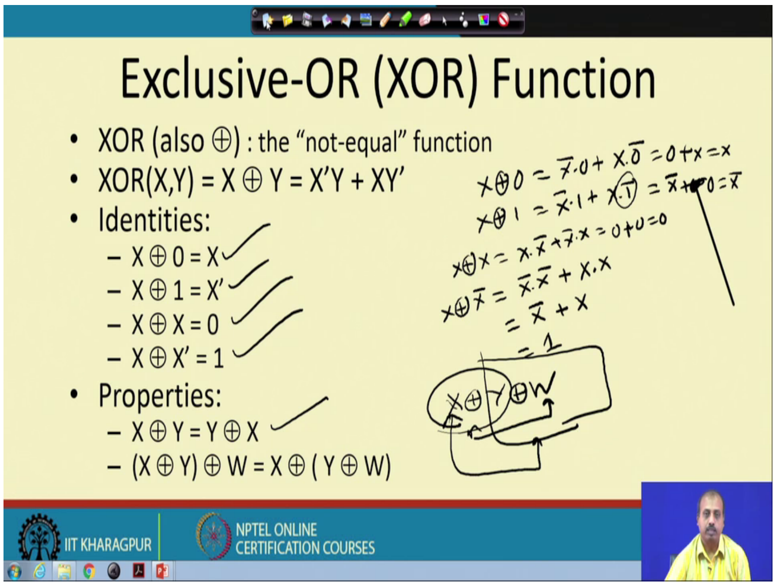
A
$\overline{B}$
$\overline{A}$
C
p
q
$X = (A+\overline{B})(\overline{A}+C)$
$X = \overline{\overline{P}\cdot\overline{Q}}$
$= P+Q$
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, nor gate conversion so, you should start with the product of some expression, and then this A or B bar. So, this is ored here and then ultimately they were ANDed there; so you just so, this gate is nothing but another nor gate, ok.

So, if we go by the previous explanation that we had. So, this is P and Q so, X equal to P bar dot Q bar so, that is nothing but P or Q whole bar. So, if you have got a product of sum realization of set of terms, ok. So, you the so, Boolean function realized as a product of sum expression. So, you have got individual inputs there and they are ultimately feeding one AND gate, if this is the situation. Then what you can do you can replace each of this OR gates by nor gates and then replace this AND gate by another nor gate. So, that will give you the NOR realization of the function.

So, it start with the product of sum form and then convert it into, convert the OR gates to nor gates, and then the AND gate to nor gate. Or if you and if you have to convert to NAND gate form, you start with the sum of product expression of the function, and then convert all the AND gates to NAND gates and then convert the final OR gate to NAND gate so, they will work.

(Refer Slide Time: 06:30)



## Exclusive-OR (XOR) Function

- XOR (also $\oplus$) : the "not-equal" function
- XOR(X,Y) = $X \oplus Y = X'Y + XY'$
- Identities:
  - $X \oplus 0 = X$
  - $X \oplus 1 = X'$
  - $X \oplus X = 0$
  - $X \oplus X' = 1$
- Properties:
  - $X \oplus Y = Y \oplus X$
  - $(X \oplus Y) \oplus W = X \oplus (Y \oplus W)$

$X \oplus 0 = \bar{X}.0 + X.\bar{0} = 0 + X = X$

$X \oplus 1 = \bar{X}.1 + X.\bar{1} = \bar{X} + 0 = \bar{X}$

$X \oplus X = X.\bar{X} + \bar{X}.X = 0 + 0 = 0$

$X \oplus \bar{X} = \bar{X}.\bar{X} + X.X$

$= \bar{X} + X$

$= 1$

$X \oplus Y \oplus W$

So, next we will be looking in more detail this exclusive or function because this is very interesting function. So, XOR which is represented as this plus within a circle so, these or this is also known as not equal function. Why is it not equal? So, this you see that this Boolean expression for this is X XOR Y written as X bar Y plus X Y bar. So, whenever

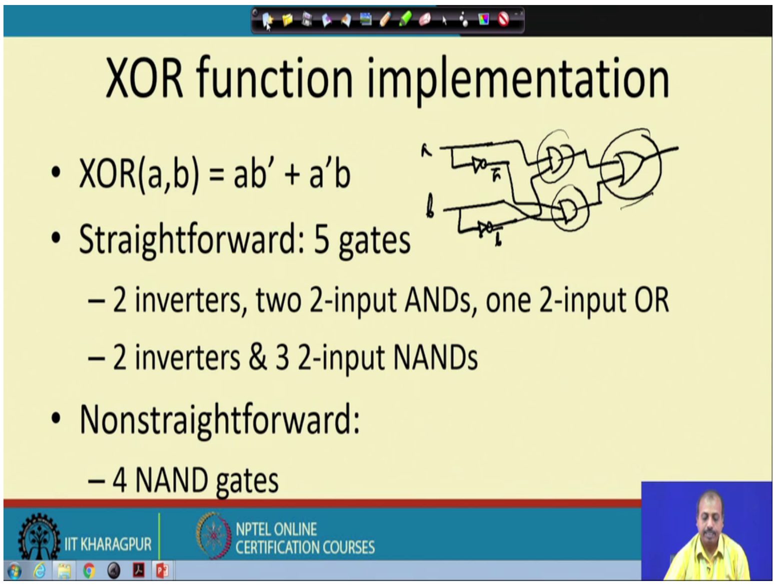
X and Y they are not equal output is equal to 1, ok, for the XOR gate. So, that is why it is called a not equal function so, there are certain identities like X XOR 0 is X XOR one equal to X bar ok. So, we can we can just put this so, X XOR Y sorry X XOR, 0 X XOR 0, if you put into this expression. So, this is X bar into 0 plus X into 0 bar. So, X bar into 0 is 0 plus X into 0 bar 0 bar is 1. So, that is equal to x so, that is equal to x so, you have got the first identity.

Similarly, X XOR 1 is equal to X bar into 1 plus X into 1 bar. So, this one bar is equal to 0, and this X bar into 1 is X bar so, that is X bar plus 1. Sorry, X bar plus X bar plus 0 so, that is equal to X bar. So, that is the second 1 now X XOR X. So, X XOR X is so, X dot X bar plus X bar dot X. So, X dot X bar is equal to 0 so, that is 0 plus 0 equal to 0, ok. So, X XOR X equal to 0 and X XOR X bar is nothing but X bar dot X bar plus X dot X, ok. So, X bar dot X bar is equal to X bar, and X dot X is equal to X and now X plus X bar is equal to 1. So, this is the 4th identity X plus X bar equal to 1.

So, these are some identities that involve X XOR operation, and this XOR operation is commutative like this X XOR Y is same as Y XOR X. So, that is true for any other Boolean function also and it is also associative like if you have to do an XOR of 3 variables X Y and W XY and W. So, this property what it tells is that you can either do this XOR first and then with that result you XOR W. So, that may be one possibility or you first do this YW XOR, and then with that result you XOR X. So, both are both will give you the same result so, this is associative in nature.

So, these are some of the properties of this XOR gates that that makes is makes it interesting in various domains or of communication systems Boolean circuits and all circuit design and all ok.

(Refer Slide Time: 09:42)



## XOR function implementation

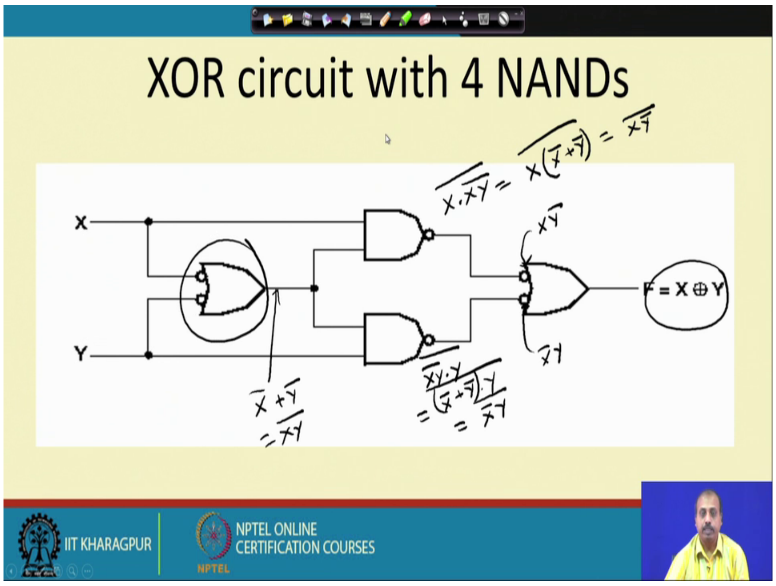
- XOR(a,b) = ab' + a'b
- Straightforward: 5 gates
  - 2 inverters, two 2-input ANDs, one 2-input OR
  - 2 inverters & 3 2-input NANDs
- Nonstraightforward:
  - 4 NAND gates

So, XOR function implementation so, if I have got since the XOR ab is ab bar plus a bar b. So, if I do a straightforward implementation in terms of say and OR gates. So, what I will do? I will have this ab bar so, I will be taking this a and this is the line a. So, I will take an inverter and this is the line b I will take another inverter so, I will get a bar here b bar there.

And now I have to do AB bar so, AB bar means I have to take one AND gate, where this a line has to be connected and this B bar line has to be connected and then a bar b. So, a bar b so, this A bar line has to be connected and this B line has to be connected, and then they are to be ORed like this. So, as a result number of gates needed is equal to 5 the 2 inverters 2 a to 2 2 input AND gates, and one 2 input OR gate ok.

So, you can also do this by using this part. So, they can be they can be converted to NAND gates each of them can be converted to NAND gates. as a result so, you can do the whole thing using 2 inverters and 3 NAND gates so, that also can be done. However, there exist another clever implementation of this XOR function that includes only 4 NAND gates. So, we will see that.

(Refer Slide Time: 11:18)



## XOR circuit with 4 NANDs

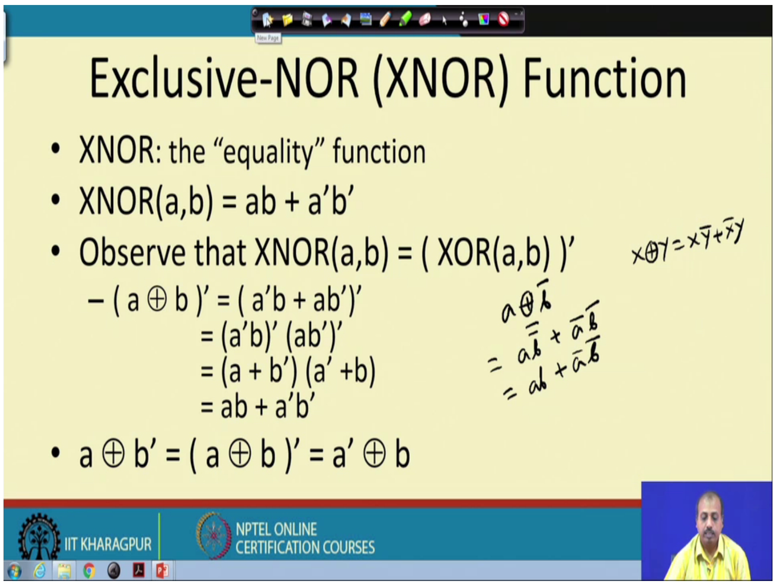
X
Y
$\overline{x.\overline{xy}} = \overline{\overline{x(\overline{x}+\overline{y})}} = x\overline{y}$
$x\overline{y}$
$F = X \oplus Y$
$\overline{x}y$
$\overline{x}+\overline{y} = \overline{xy}$
$\overline{\overline{xy}.y} = \overline{(\overline{x}+\overline{y})y} = \overline{x}y$
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, this is the 4 NAND gate realization of the XOR gate so, we have got this so, here this is um this is basically x bar or y bar, sorry. So, this point we are getting x bar or y bar at this point we are getting x bar or y bar so, which is nothing but XY bar; so, this is also a NAND gate. So, here I am getting xy bar, and here it is NAND’ed with x, ok. And then so, you can say, so, this is this is x dot xy bar whole bar, ok. So, if we simplify so, you will get here as; so x into x bar plus y whole bar so, you will get xy bar. So, you will get xy bar from there.

Similarly, from this side if you look into so, you will be getting here you will be getting this XY bar, XY bar dot Y whole bar. So, if you again simplify like that; so, you will get this X bar plus Y bar into Y whole bar. So, you will get here X bar Y whole bar oh, sorry, I did a mistake here. So, this is X Y X XY bar. So, I will ultimately the term that survives is XY bar there and X bar Y here.

Now, here this is another NAND gate so, the after this inversion so, this term is XY bar and this term is X bar Y, and then I am doing an odd. So, I am getting this or of these 2 so, this is basically the XOR function. So, this way I can do this XOR realization 2 input XOR gate realization using 4 NAND gates, ok. So, that is say for like we can save some gates, like in the previous implementation in the previous implementation. So, we had here 2 inverters and 3 2 input NAND gates, but in the next implementation we have got only 4 NAND gates. So, we are saving some gates so, this may be useful in some situation.

(Refer Slide Time: 14:07)



## Exclusive-NOR (XNOR) Function

- XNOR: the "equality" function
- XNOR(a,b) = ab + a'b'
- Observe that XNOR(a,b) = ( XOR(a,b) )'
  - ( a ⊕ b )' = ( a'b + ab')'
    = (a'b)' (ab')'
    = (a + b') (a' +b)
    = ab + a'b'
- a ⊕ b' = ( a ⊕ b )' = a' ⊕ b

$x \oplus y = x\bar{y} + \bar{x}y$

$a \oplus \bar{b}$
$= a\bar{\bar{b}} + \bar{a}\bar{b}$
$= ab + \bar{a}\bar{b}$

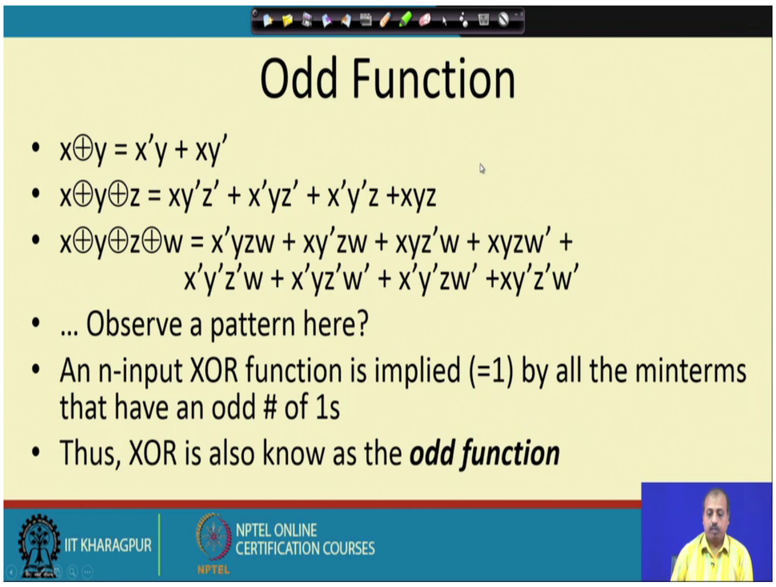
So, just the opposite of XOR is XNOR so, XNOR is known as the equality function so, XOR was non equality function XNOR is equality function, and XNOR of ab is defined as ab plus a bar b bar ok. So, whenever both the inputs are equal so, if a and b both are equal both are equal to 1 then ab is equal to 1, if both of them are equal to 0, then a bar b bar is equal to 1 as a result this XNOR function will be giving us 1, when either both a and b are equal to each other.

So, we can see that X a in the XNOR of ab is the invert of XOR of ab. So, how do you prove it so, this a XOR b the this XNOR is nothing but a XOR b whole bar, ok, now this a XOR b is A bar b plus a bar b bar whole bar. So, this can be you can apply De Morgan's theorem. So, we get a bar b bar and ab bar so, a bar b bar will give us a plus b bar and this ab bar whole bar will give us a bar plus b so, if you multiply ultimately you get ab plus a bar b bar.

So, we so, if you take the invert of XOR so, you come to AB plus a bar b bar which is nothing but XNOR of ab, ok, now if you if you just take one of the variables as a complemented one so, a XOR b bar that will give us the XNOR of a and b. So, let us check that; so, let us say a XOR b bar so, a XOR b bar so, it is given by AB bar, just apply this ax X XOR Y formula. So, it is X Y bar plus X bar Y ok. So, this X Y bar I have written plus X bar Y, fine? So, this is b double bar is nothing but simple b so, ab plus a bar b bar so, this is the XNOR.

So, this a b bar a XOR b bar so, this will give us this XNOR of a and b. And similarly by the similar logic this a XOR a bar XOR b will also give us the XNOR of a and b. So, these are the relations between XOR and XNOR.

(Refer Slide Time: 16:36)



## Odd Function

- $x \oplus y = x'y + xy'$
- $x \oplus y \oplus z = xy'z' + x'yz' + x'y'z + xyz$
- $x \oplus y \oplus z \oplus w = x'yzw + xy'zw + xyz'w + xyzw' + x'y'z'w + x'yz'w' + x'y'zw' + xy'z'w'$
- ... Observe a pattern here?
- An n-input XOR function is implied (=1) by all the minterms that have an odd # of 1s
- Thus, XOR is also know as the ***odd function***

So, this XOR is also known as odd function; so, why a odd function? So, you see that if you if you take a 2 input XOR gate, then this is the logic expression. So, x XOR y is x bar y plus xy bar.

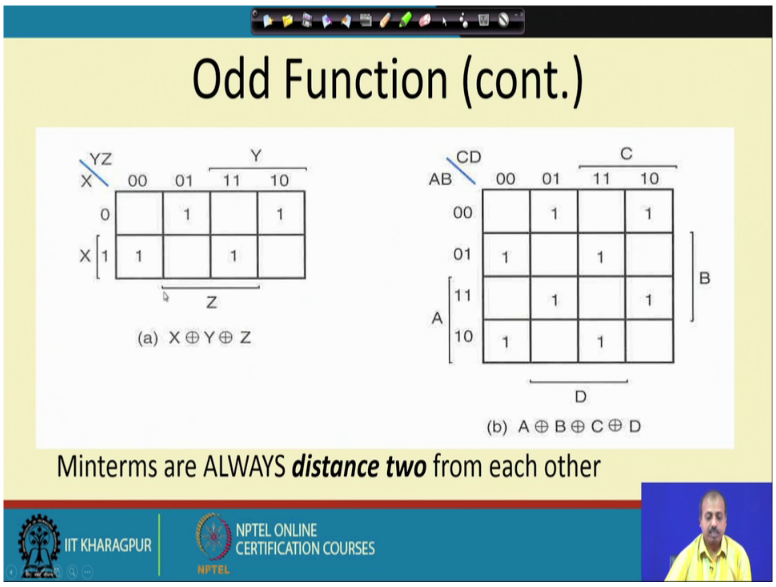
So, if you take a 3 input XOR gate, 3 input XOR function x yx XOR y XOR z, then this if you expand so, you will get an expression like this x y bar z bar plus x bar yz bar plus x bar y bar z plus xyz. If you take a 4 input XOR function, then the expression will be like this. So, can we observe a pattern here, of course, there is a pattern. So, in all of them so, the terms where odd number of inputs are equal to 1, it is surviving, like for the 2 variables so, here you see that these 2 terms they have got only one input equal to 1, that here only y equal to 1 here only x equal to 1.

So, if you look into say this term so, here only x equal to 1 y and z are equal to 0. Similarly in this term only, y is equal to 1 x bar ax and z are equal to 0. If you like take a 4 variable term, this xy z bar w so, here x y and w they must be equal to 1, and z must be equal to 0. So, whatever you do? So, you see that here 3 variable, the 3 variable should get the value one and one of them should get value 0.

Similarly here so, only x variable should get the value 1, and 3 of them should get value 0. So, either one variable is getting in a 4 variable XOR function, either only one of the input is equal to 1 or 3 of the inputs are equal to 1, then only output is equal to 1. So, in a general case we can say that when odd number of inputs are equal to 1, then we will have the XOR gate output equal to 1.

So, an n input XOR function is 1, by all minterms that have an odd number of ones in it. So, that is why XOR is also known as odd function. Because it will it will have the odd number of minterm, odd number of ones equal to 1 in those minterms.

(Refer Slide Time: 18:52)



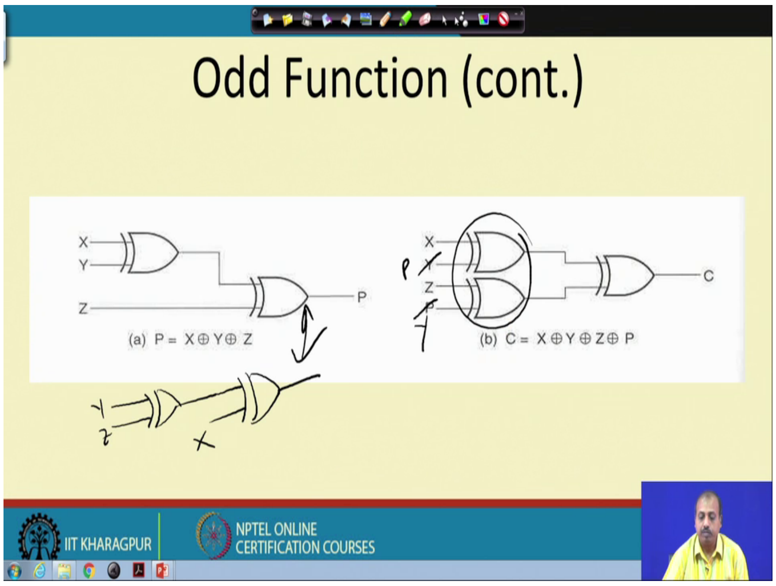
## Odd Function (cont.)

Minterms are ALWAYS ***distance two*** from each other

Now, if we if we just look into that truth table the karnaugh map; so, this is for the 3 variable so, XY and z. So, you see that they will be distributed like this. So, this so, they do not group in a quad or pair or octet like that so, they get distributed like this

Similarly, a 4 variable karnaugh map so, this ones are getting distributed in such a fashion, that they are they do not form any pair, ok. So, minterms are always distance 2 from each other so, this is the property of odd function.

(Refer Slide Time: 19:29)



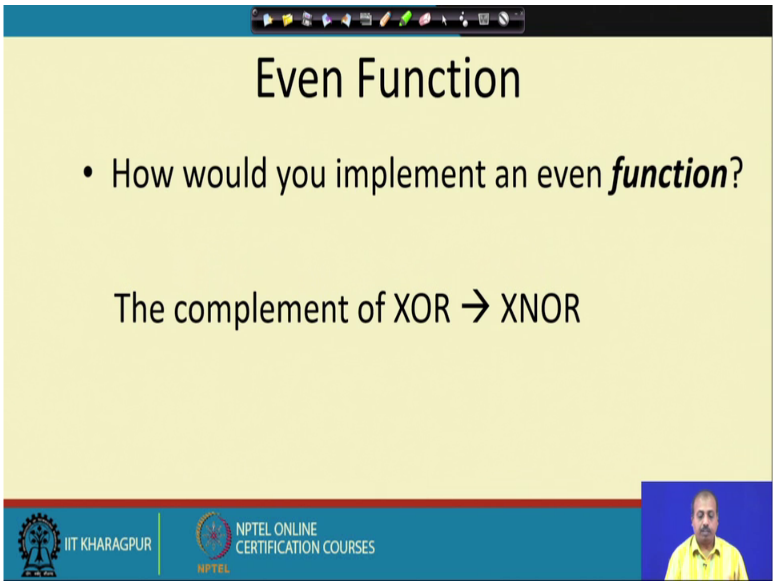
## Odd Function (cont.)

So, you can realize these odd functions like this. So, this X XOR Y coming here and X then Z so, X XOR Y XOR Z so, we can realize by means of 2 XOR gates.

If I have take 4 variable XYZ and P, then this X and Y can be XOR'ed first then Z and P can be XOR, and then this one can be XOR. And it does not matter in whatever order we do it. So, somebody may realize the same function like this also somebody may say that I will do this Y and z XOR first, I will do this Y and z XOR first, and then I will XOR this output with the X, ok. So, that is because of this associativity property of this XOR operation. So, it does not matter the order in which we do the XOR'ing so, these 2 are exactly same.

Similarly here also so, you can change the pair, like in instead of taking Y here you can take P here, and instead of P you put the Y here. So, that can be done ok. So, that will not change the expression at all it will remain all the same, due to this associativity property of this XOR.

(Refer Slide Time: 20:47)



## Even Function

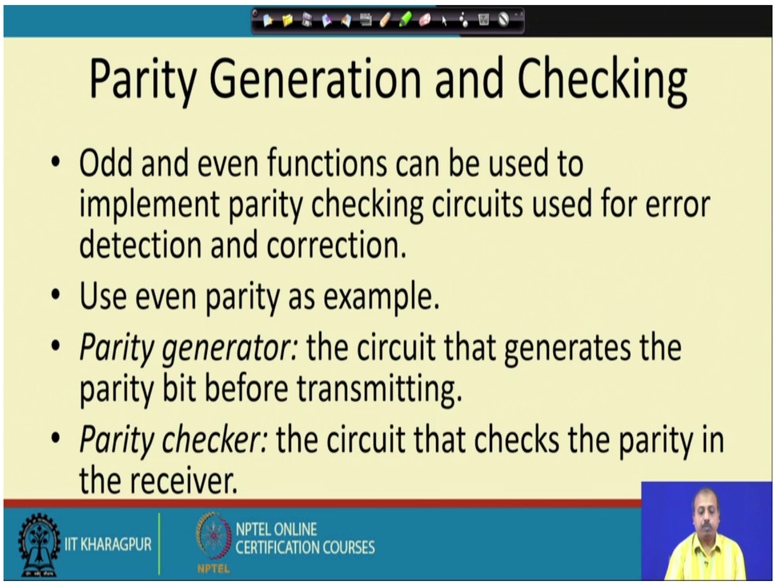
- How would you implement an even ***function***?

The complement of XOR → XNOR

So, we can also have something called an even function. So, even function if you want to realize. So, how can we do this? Like complement of XOR which is the XNOR.

So, XNOR gives us the even function.

(Refer Slide Time: 21:00)

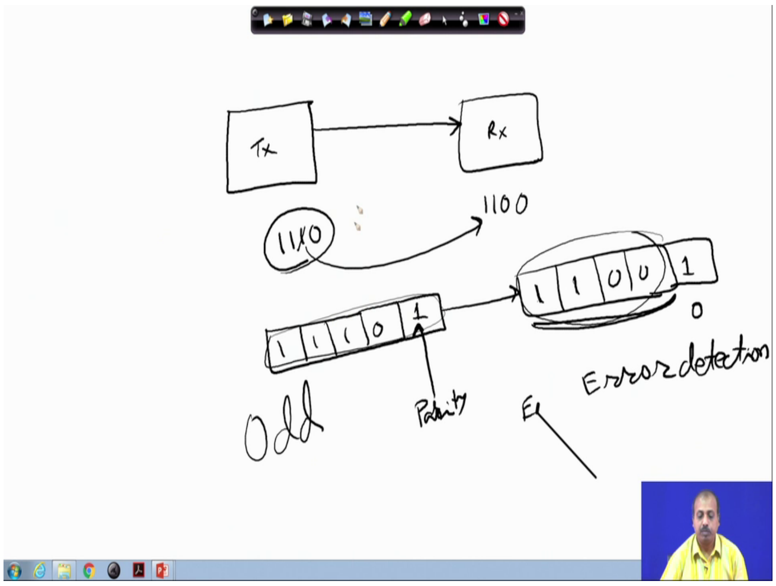


## Parity Generation and Checking

- Odd and even functions can be used to implement parity checking circuits used for error detection and correction.
- Use even parity as example.
- *Parity generator:* the circuit that generates the parity bit before transmitting.
- *Parity checker:* the circuit that checks the parity in the receiver.

So, many a times we use this XOR to generate the parity. So, whenever we have got an odd and even function, they can be used for implementing parity checking circuits in error detection and correction. So, what is parity? So, parity is something like this, that suppose, you are you have got a communication system or a where there is a transmitter and a receiver.

(Refer Slide Time: 21:26)



So, this is the transmitter and this is the receiver so, you are transmitting some bits from here to here. Now suppose I transmitted the bit pattern 1 1 1 0, now at the receiving end, I should receive I get I should get this 1 1 1 0, but due to this communication channel, the some error may into the keep into the transfer. So, it may so happen that this one at the receiving end it has become 0 so, at the receiving end it receives 1 1 0 0. Now at the receiver side how do we know that we have received a possibly correct pattern.

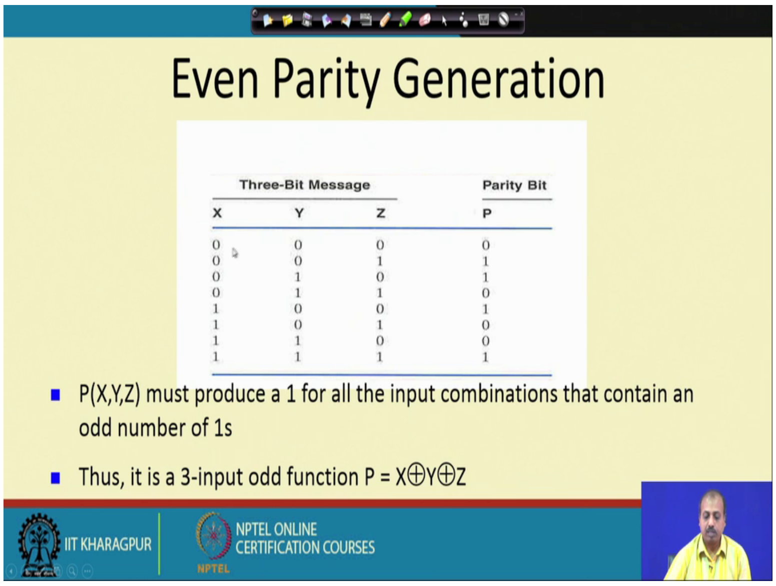
So, for that what is done is; apart from this 4 bits that we are transmitting. So, this 1 1 1 0, this 1 1 1 0 another bit is added. So, this bit is called parity, this bit is called parity. So, parity is actually it keeps an watch on the number of ones that you have sent onto the bit stream. Like, if you say that this parity so, we can say that whenever we transmit we always transmit an even number of ones. So that means, in this case we are transmitting 3 ones so, that is odd number so, this bit we make it one.

So, that at the receiving end so, if you receive a pattern like this so, assuming that this parity bit has been transmitted properly; so, the received bit pattern is something like this so, you have got 1 1 0 0 and then parity bit is 1. But from this if we calculate the parity bit, since, we have taken the protocol that we will always be transmitting even number of 1's. So, based on the received bits so, since these 2 bits are 1 this bit should actually should is expected to be 0. But the receiver finds that it has receive the bit one. So, it understands that that is must be something wrong in this transmission so, receiver can detect the situation ok.

So, this is called error detection so, this is called error detection. So, the receiver can detect that there is some error in the transmission so this parity is actually used for this part. So, apart from the bits that you are transmitting; so you transmit the another extra bit now you may follow even parity, where you check where you make the number of ones even, or you can make you can follow the other option other alternative also you can say it is a odd parity that is the total number of 1's, that you will transmit you will make it odd, ok.

So, both of them can be done using this um using this XOR gate. So, this parity generator so, this circuit generates the parity bits before transmitting, and this parity check so, this circuit is present in the transmitter. So, this will generate the parity bit, and that the receiver side you will have a parity checker circuit. So, it will check the parity after the bit pattern has been received.

(Refer Slide Time: 24:36)



## Even Parity Generation

| Three-Bit Message | | | Parity Bit |
|---|---|---|---|
| X | Y | Z | P |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

- P(X,Y,Z) must produce a 1 for all the input combinations that contain an odd number of 1s
- Thus, it is a 3-input odd function $P = X \oplus Y \oplus Z$

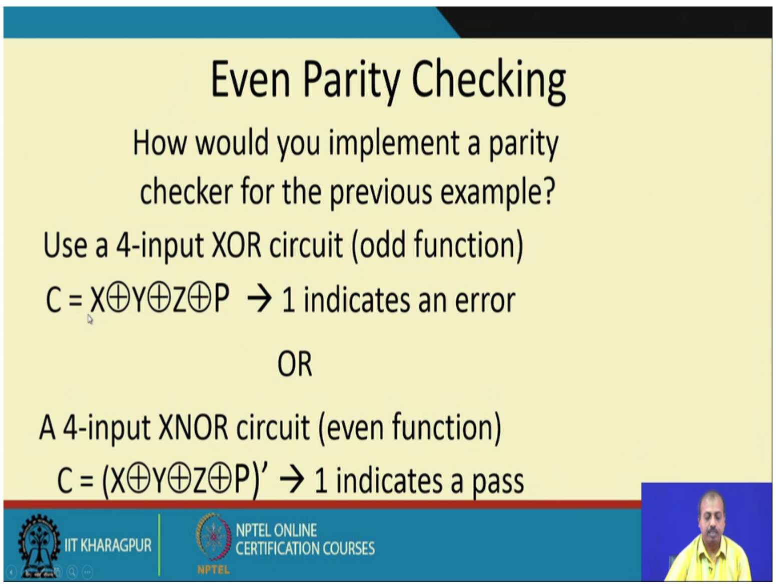
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

So, like this so, suppose I have got this situation so, even parity. So, I will make the number of ones transmitted as even.

So, if it is the bits transmitted at 0 0 0, then the parity bit is also 0; because number of ones is equal to 0, that is even if the message bits are 0 0 one then the parity bit will be equal to 1 so, that number of ones in the whole message plus parity is even. Similarly, 0 1 0 also parity bit is 1, 0 1 1 the parity bit is 0. So, if you so, this way for the for a 3 bit message so, we can write down what is the parity bit.

So, PXYZ the parity must produce one for all input combination that contain an odd number of 1's. And whenever we get this statement, we understand that without doing any Boolean minimization anything, this statement straightway tells that this is an odd function so, this is basically XOR of xy and Z. So, this parity so, even parity generation can be done by means of the XOR gates. So, that is why XOR is a very popular gate in the communication system.

(Refer Slide Time: 25:54)



## Even Parity Checking

How would you implement a parity checker for the previous example?

Use a 4-input XOR circuit (odd function)

$C = X \oplus Y \oplus Z \oplus P \rightarrow$ 1 indicates an error

OR

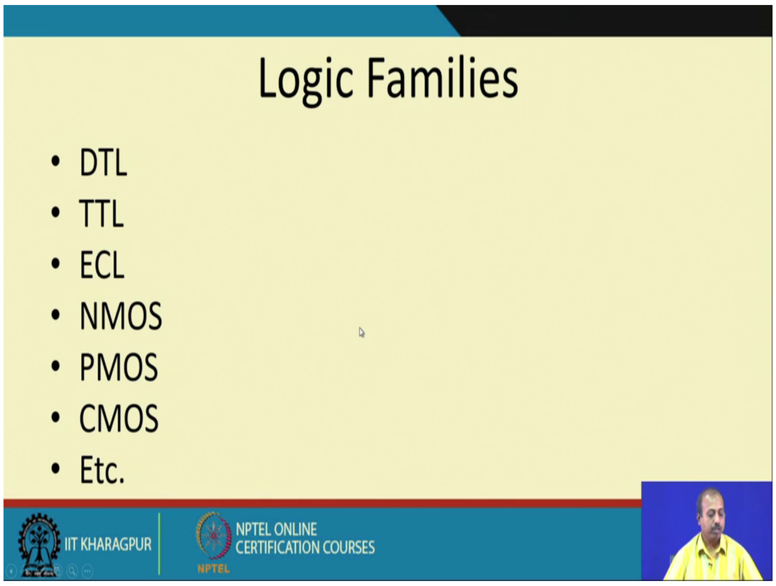
A 4-input XNOR circuit (even function)

$C = (X \oplus Y \oplus Z \oplus P)' \rightarrow$ 1 indicates a pass

So, how do we check the parity? So, you can do the operation like this. So, you can have a 4 input XOR circuit so, you can have this since you are getting 4 bit XYZ and P and if you do an XOR operation. So, if you get an one; that means, there is an error because even parity means if you XOR with the parity bit also, if you get always get a 0 so, if you get a one; that means, there is an error.

So, for if you are on the other hand, if it is an odd parity, then if you are doing this if you can use a 4 input XNOR gate, and so, in this case if you get a one; that means, it is it is all right so, if it is 0 then there is a problem, so this way using this XOR and OR gate. So, we can use this even parity checking. So, for even parity checking either you can use a 4 input XOR gate or in that case say one will indicate an error. Or you can you can use a 4 input XNOR gate. In that case, the output one will indicate that it is correct, 0 will indicate that there is a problem, ok.

(Refer Slide Time: 27:03)



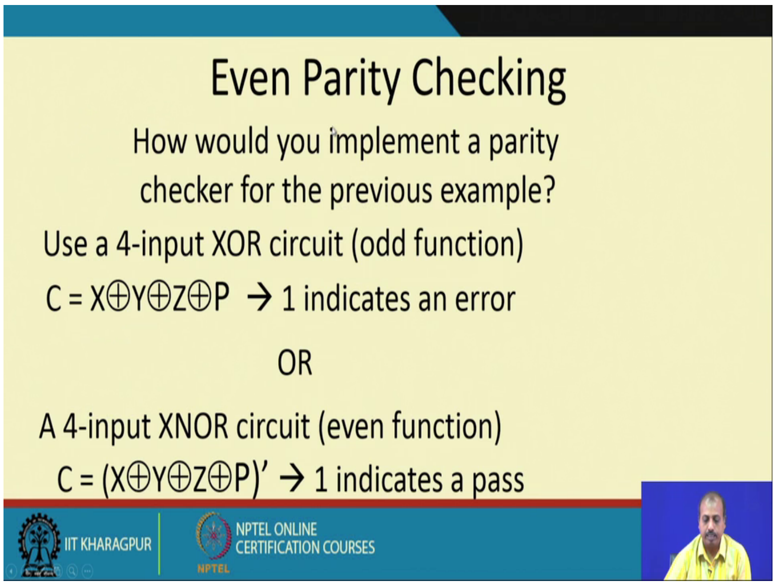
## Logic Families

- DTL
- TTL
- ECL
- NMOS
- PMOS
- CMOS
- Etc.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, you can also use a, similarly you can also take care of the odd parities. Next we will be looking into logic families. Like this different this so, for we are talking about logic gates and logic symbols. Now it is the same logic gate how do you realize like, one implementation we have shown in terms of switches and lamps at the very beginning of this discussion, now there are many circuit elements that can be utilized in like say diodes transistors etcetera.

(Refer Slide Time: 27:48)



## Even Parity Checking

How would you implement a parity checker for the previous example?

Use a 4-input XOR circuit (odd function)

$C = X \oplus Y \oplus Z \oplus P \rightarrow$ 1 indicates an error

OR

A 4-input XNOR circuit (even function)

$C = (X \oplus Y \oplus Z \oplus P)' \rightarrow$ 1 indicates a pass

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

That can be utilized for realizing different for getting different circuits. For example this DTL so, this a stands for this stands for this diode transistor logic, and then that, sorry, this diode transistor logic. So, this is this consist of diodes and transistors, then we have got TTL which is transistor logic. So, here all transistors are used for getting the circuit, and for getting this basic logic gates. Then there is a ECL logic. So, this is again another family, then do you have got NMOS which is metal oxide semiconductor family. And this n stands for n channel metal oxide semiconductor. Then PMOS P channel metal oxide semiconductor, then CMOS this is a complementary metal oxide semiconductor.

So, this way there are many such logic families that has been used for developing this basic logic gates realizing this basic logic gates over the years. So, DTL is one of the first logic families that we had involving diodes and transistors, TTL came after that. And at present so, we are working with the CMOS family, ok. This CMOS family is the most popular now, because of several reason, because of it is compactness, because of it is power consumption less power consumption and all. So, this CMOS is going to be, is right now it is the de facto standard that we have for this integrated circuit chips ok.

So, we will look into this CMOS in detail in the successive lectures.