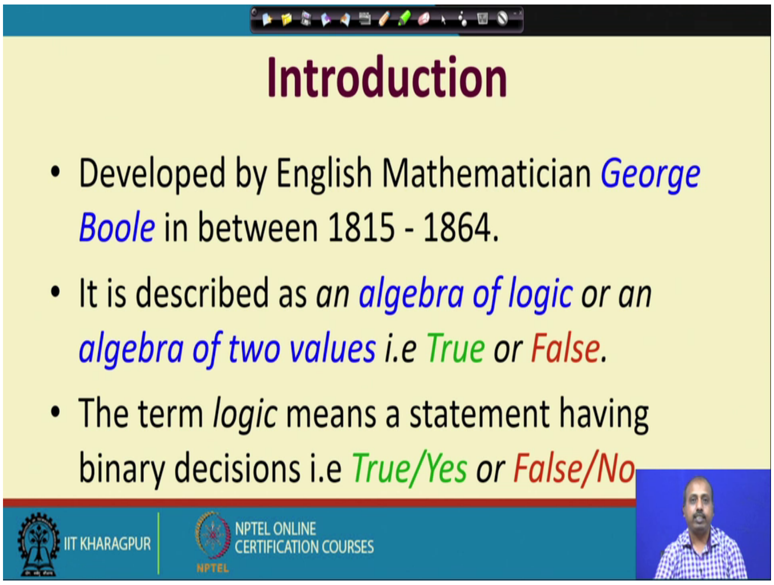
Digital Circuits

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

Lecture - 08
Boolean Algebra

So, we will now start with another algebra method which is known as Boolean algebra. So, this is the algebraic systems that are used in computer systems and particularly in the digital circuits. So, which is this Boolean algebra; so this was this was invented longer, before these digital computers came into existence, but these are. So, previously it was only for say mathematical interest, but after these digital circuits came this Boolean algebra got a new life. So, that way it is being used very much in this digital systems computer and other digital accessories.

(Refer Slide Time: 01:00)

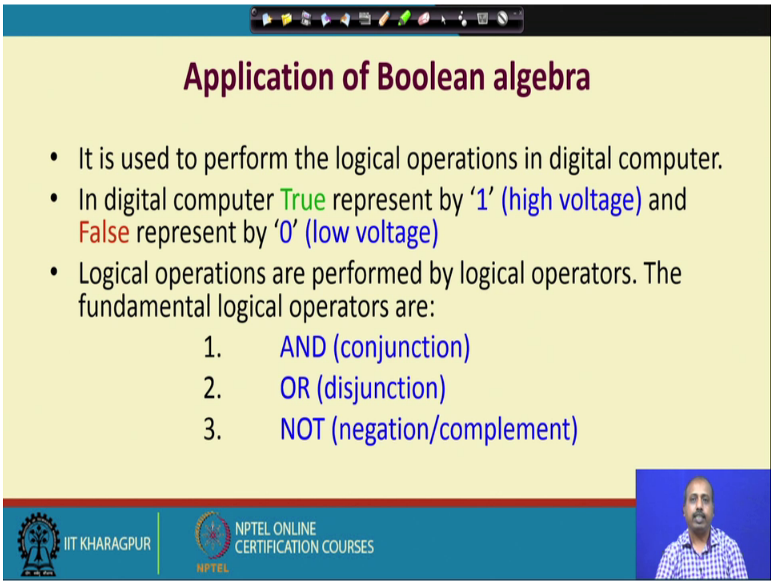


So, to introduce this was developed by English mathematician George Boole, in between 1815 and 1864. So, you can see that it was long before these digital circuits came into existence. So, it is described at the algebra of logic or an algebra of 2 values true or false that is why it is so. So, it is; so the variables that we have in any algebra when it is a about any algebra. So, there are certain things that we have to talk about like what are the values that variables on this algebra can take up and what are the operations that we can do on those variables ok.

So, in case of like when he was normal algebra that we are familiar with so the variables they can take up some values from some domain, may be interior real etcetera. And then we can do some operation the addition, subtraction, multiplication, division type of operations on those values.

Similarly, here in case of Boolean algebra; it if the values are only true or false there are only two values possible true or false. Here the term logic means the statement having binary decision, it is a it may be true or yes and false or no. So, either the statement may be true or false or similarly some cases we may call the true as yes and false as no. So, this is the basic idea behind Boolean algebra.

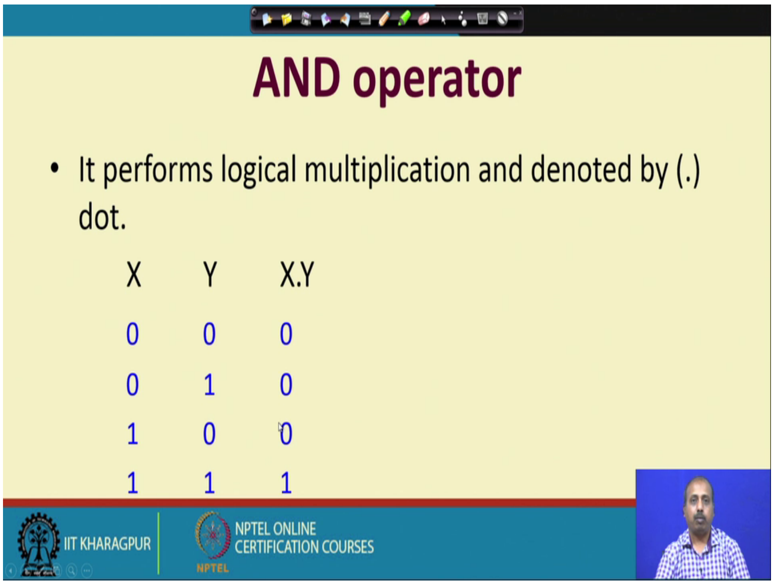
(Refer Slide Time: 02:27)



So, where it is the application of Boolean algebrait used to perform logical operation in digital computers and in fact, in digital circuits we are doing. In digital circuits we are doing it. And this digital computer in case of digital computer digital circuits a true is true is represented by a 1 often it is a high voltage or false is represented by a 0 which is low voltage. Of course, this is high and low voltage. So, these terms are not very much correct, because as we have seen previously that it may be that logic high is a the negative voltage also may be taken as logic high, and positive voltage may be taken as logic low. But for the sake of understanding we can say that when it is high voltage, so this is 1 and low voltage is 0. So, it is logic high and logic low.

So, logical operations that perform that on this by some logical operators and there are some fundamental logical operators that are there in Boolean algebra: one is called and or conjunction or disjunction and not which is negation or complement. So, these are the three fundamental operations that you have in Boolean algebra. Of course, there are many other derived operations like say; NAND, NOR, XOR etcetera XNOR. And also you can think about any operation that that can be built around these basic operations, and that way we can get some newer operations in the Boolean algebra.

(Refer Slide Time: 04:00)

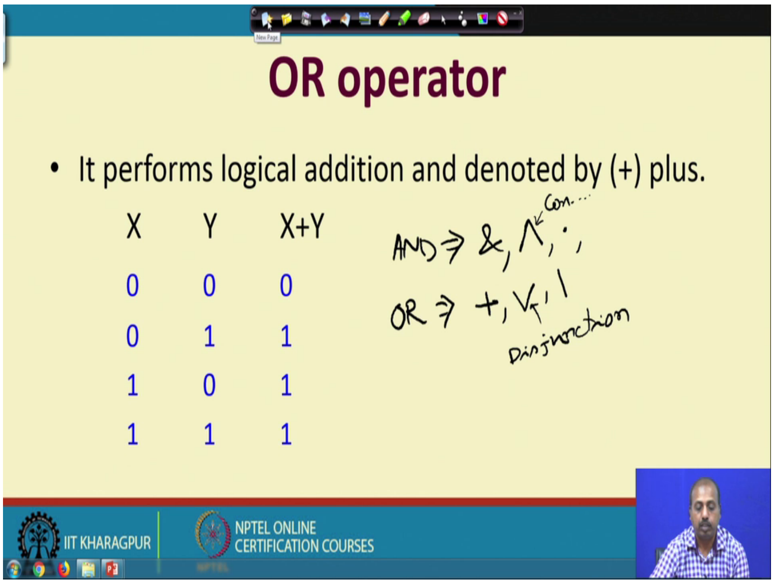


So, the first simple operation is the AND operation. So, it is often denoted by this dot symbol. So, if X and Y are two Boolean variables their AND is denoted by X dot Y. Many times this dot is not written explicitly where it is implied that the operator is AND. So, we just write X Y meaning that it is X dot Y or X and Y.

Now, if you say that in case of AND the operation is defined like this; so if you take X equal to 0 and Y equal to 0 then there AND operation is also the and it value is also 0. If X is 0 and Y is 1, then this is 0 the result will be 0. The 1 and 0 if X equal to one and Y equal to 0 then the result will be 0, when both are 1 X and Y both are 1, then the result will be 1. So, this is a binary operator because it takes two the variables and does the operation on that ok. So, that two operands and there is a two operand operator.

So, it takes two operands X and Y and depending upon the operands value the result is produced. And this is the rule: if both the operands are 1 in that case only the output is 1 otherwise the output is 0.

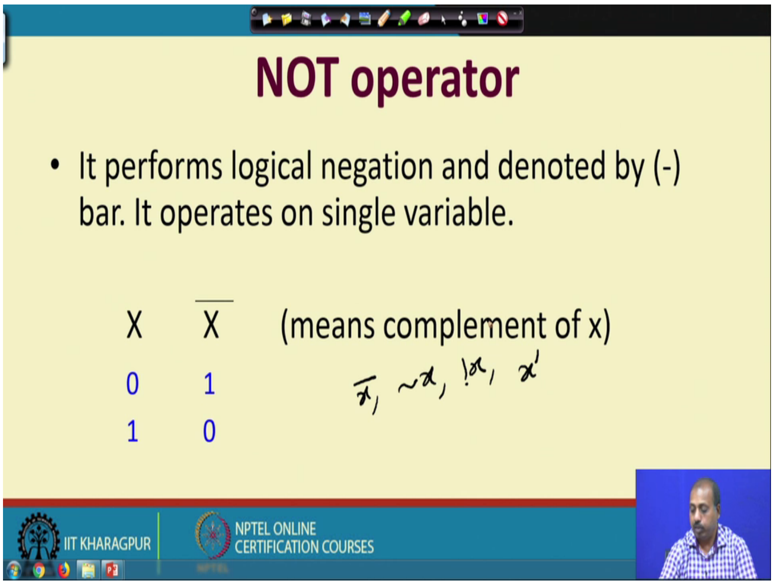
(Refer Slide Time: 05:24)



Next, the other another fundamental operator which is known as the OR operator; so this is like this that if we have got both the operands as 0, then only the result is 0 and otherwise the result is 1. So, this is 0 or 1 is 0 is 1, 1 or 0 is 1, 1 or 1 is 1. So, this logical OR operator is often denoted by a plus symbol. Of course, there are many other notations like for this AND for this AND we have got for this AND we have got notations like this, then we have got notation like this v ok. So, these are the alternate notations that we have.

Similarly, for OR we have got notations like say plus that we have seen and the dot is definitely there, and another one is there is a no symbol. So, the no additional symbols. So, the variables coming just one after the other so that is also AND operation. Now OR for OR we have got plus, we have got this particular symbol, which is known as disjunction. So, this is called conjunction and this is called disjunction. So, this is disjunction and this is called conjunction. So, this is there and sometimes we represent it by a vertical bar. So, that is also OR. So, there are many notations. So, you may you may find different notations at different places all of them in the same thing all of them are equally useful and equally valid.

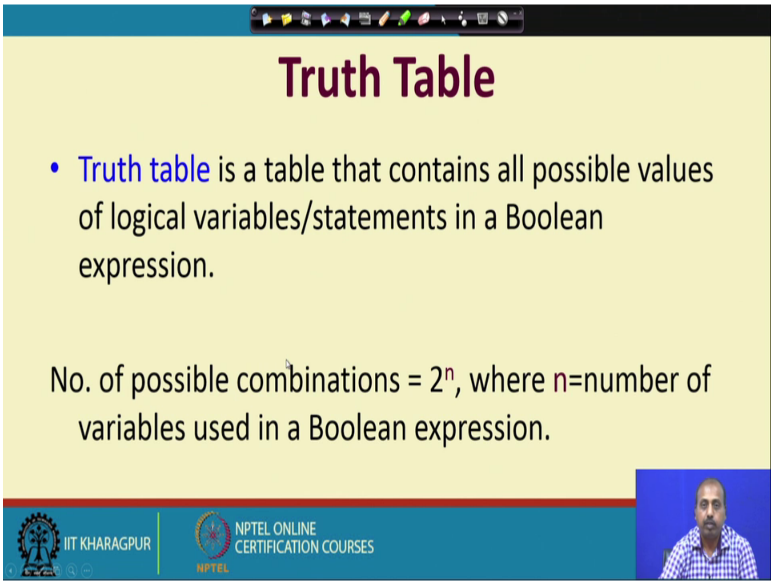
(Refer Slide Time: 07:16)



Now, so, this is the OR operator for the NOT operator. So, it is like this that if it is a unary operator. So, it takes only one operand and does the operation on that. So, this is repre denoted by a by a bar over the symbol. So, this is X if X is the variable its compliment is denoted as X bar, and if X is 0 then X bar is 1. If X is 1 X bar is 0. So, this is just the complement of X.

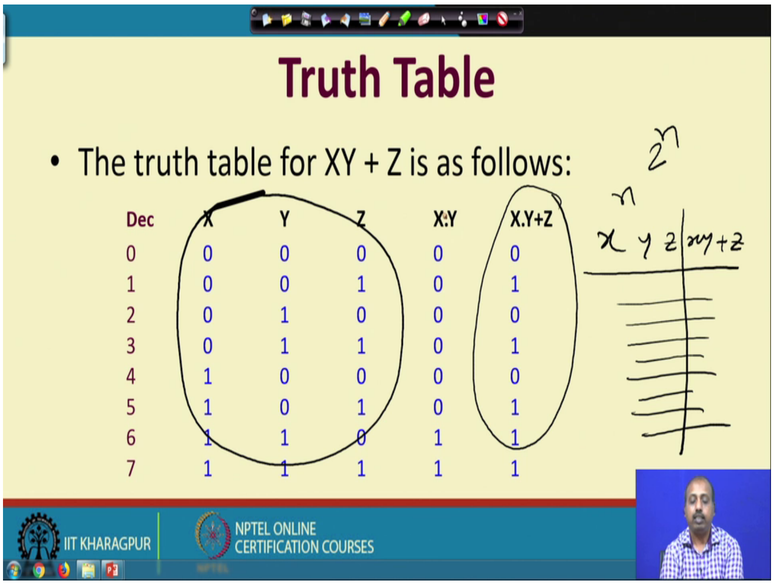
So, again there are other notations also like we have got the notation say. So, x bar is one notation that you have already seen here. So, we sometimes we write it as tilde x, sometimes we write it as not of x. So, sometimes we write it as x then a hash; so or x dash. So, these type different notations are there. So, all these notations in our lecture also many a times will be interchangeably be using these notations. So, meaning all of them in the same thing. So, that is the NOT operator. So, AND or NOT; so these are the 3 fundamental operations that we have in Boolean algebra.

(Refer Slide Time: 08:30)



Next we introduce a concept called Truth Table. So, truth table it is a table that contains all possible values of logical variables or statements in a Boolean expression. So, how does it look like? So, number of. So, if there are n number of variables then there are 2 to the power n possibilities like.

(Refer Slide Time: 08:54)



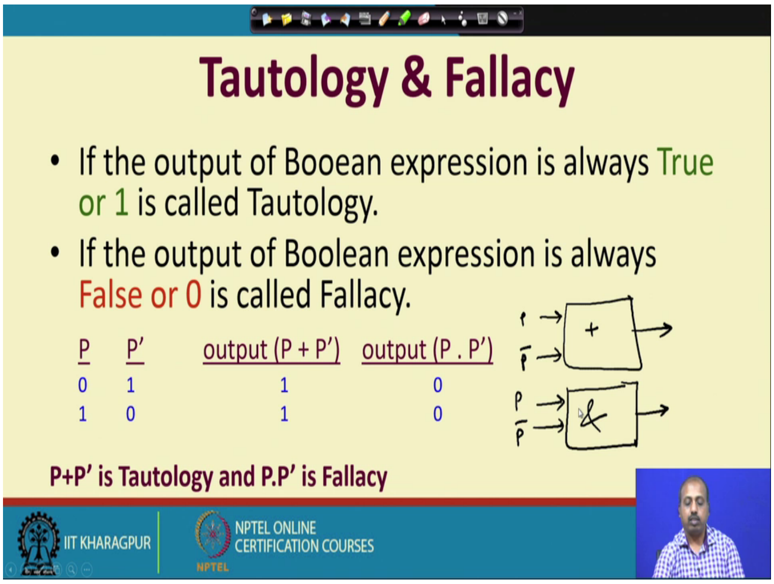
So, like here suppose we have got a Boolean function X Y plus Z. So, it is like this; so so this x y and z. So, these are the inputs to the function now if you enumerate the alternatives that you can have here. So, it can go from 0 0 0 to 1 1 1. So, that way since this is a 3 variable, I have got 8 different alternatives. So, 000 to 111.

Now, if we look into the term X Y or X dot Y. So, it is an AND of X and Y. So, I can write down this part. So, this is 0 x y is 0, similarly this is also 0. So, only when x and y both are one this xy part becomes 1 and xy plus z. So, this z is also there. So, it is whenever x y is 1 or z is 1 the xy plus z is 1. So, I can say. So, here z is 1. So, this is also 1, at this point x y is 1, but z is 0, but we have got this is xy plus z to be equal to 1 similarly here both xy and z both are 1. So, we have got xy plus z equal to 1.

So, this way in a truth table, so you can. So, this is slightly extended version of the truth table, in normally we will be we will be writing say this part and we will be writing the this particular columns only these 2 columns will be rewritten. So, this x y z and the output part which is xy plus z in our case. So, normally we write it like this and there are the there are 3 variables here.

So, I will have 2 power 3 that is 8 such rows there, in general if there are n variables then there will be 2 power n such rows. So, this is the truth table. So, truth table as the name suggests. So, these tells what when the function assumes a true value and when the function assumes a false value, when the input variables are assigned different combinations of values ok.

(Refer Slide Time: 11:15)



So, next we will be coming across 2 terms one is called tautology another is called fallacy. So, sometimes what happens is that, this out outcome of a Boolean expression is always true it is called a tautology. So, for example, if I say that we are going through the digital circuits course now. So, that is always true. So, we are definitely going through it, but if we say that we at the, we are not talking about say tautology now or say Boolean algebra now so, that is always false statement because at now at present we are talking about it.

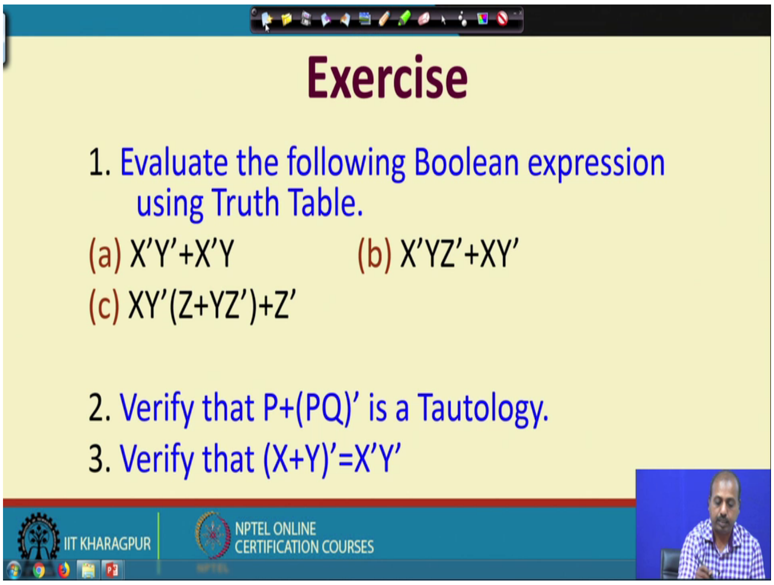
So, in this way the sum of the Boolean expressions may be may be always true or always false. So, when it is always true it is called a tautology, when it is always false it is called a fallacy. So, if the output of the Boolean expression is always false or 0 we call it a fallacy. So, like this say. So, if I have got this P and P bar then output. So, now,. So, the suppose I have a function which is P or P bar ok. So, P or P bar now. So, P or P bar means when this P is 0, P bar will be 1 and when this P is 1 P bar is 0. So, so these 2 are input like if I if I have got a block, if I have got a block which is computing say P plus P bar we which is computing say OR function ok.

Now if I give it this one side I give P, another I give P bar then what will be the output of this function. So, as I am; so this P as. So, so when I am giving P as 0. So, definitely P bar is equal to 1. So, whenever I am feeding a 0 here I am feeding a one at this point, similarly when you are feeding a 1 here I am feeding a 0 at this point.

Now, since this is a OR function. So, for both the combinations it will produce a 1. So, we can say that this particular module that I have. So, this always produces 1 irrespective of the value of P whether the P is 0 or 1. So, this module will always give you 1. Similarly if we have got another module that computes this P and P bar. So, that is. So, this is a AND operation. So, here also I give P and P bar as input and then I see what is the output. Now whatever be the value of this P, whatever be the value of P this P and P bar is always going to be equal to 0.

So, in this way in one case we have got always 1, in the other case we have got always 0 it does not depend on the value of P. So, this P plus. So, this P plus P bar. So, this is a tautology, because this is always true and P dot P bar is a fallacy, because this is always false ok.

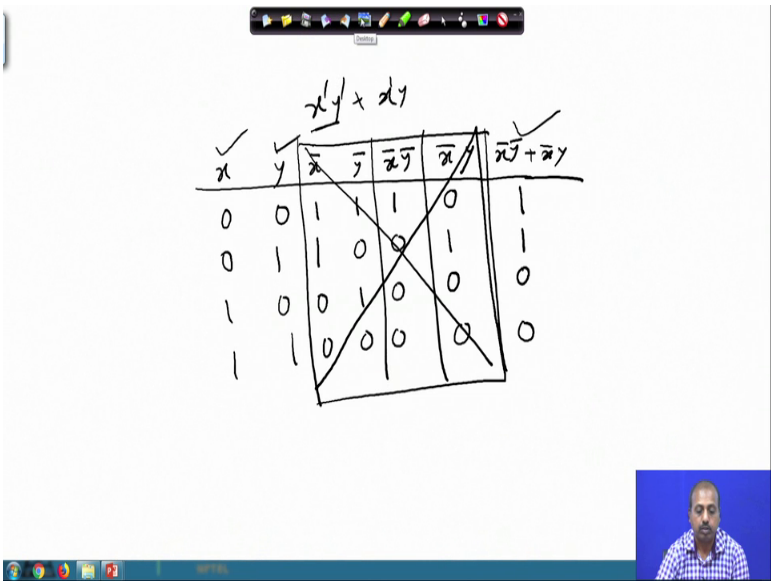
(Refer Slide Time: 14:20)



So, we will see some we try to evaluate some Boolean expression using this using the truth table like say X bar Y bar plus X bar Y. So, what is the value? So, if we want to get the corresponding truth table. So, you have to proceed like this.

Say let us take a new page and do that.

(Refer Slide Time: 14:42)



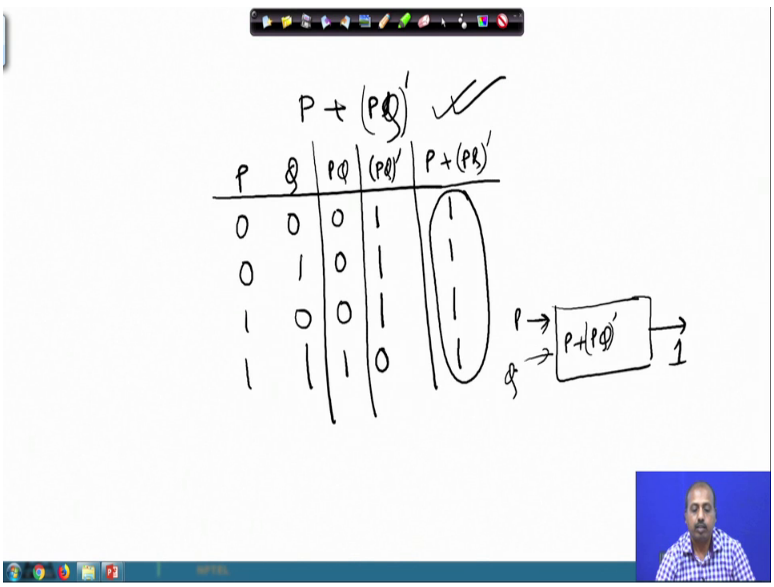
So, it is x bar y bar plus x bar y, I think whatever that x bar y fine. So, if you want to get that truth table then what do you do? You take this x and y. So, it can take up the values 0 0 0 1 1 0 and 1 1. Now we take up this 1. So, I will need x bar and y bar. So, I write down the x bar and y bar. So, x bar value is a 1 1 0 0 because when x is 1, x bar is 0 and when x is 0 x bar is 1 and y bar is 1 0 1 0.

Now, if we have to compute say I am already using this bar and this dash interchangeably. So, if I am have to if I want to compute x bar y bar; so for the first term. So, I have to. So, this x bar y bar is 1 because both of them are 1. So, AND is 1. So, this is 0, this is 0 and this is 0, then I have got x bar y. So, x bar y if we say; so we have to consider this column and this column ok. So, this x bar y. So, this is 0, this is 1, this is 0 and this is also 0, x bar y. Now finally,. So, I have got the column x bar y bar plus x bar y. So, it is the OR of these 2 columns. So, this is 1, this is 1, this is 0 this is 0.

So, ultimately now, you can forego this part of the truth table. So, you can forego this part and you can say that my truth table is this column, this column and this column. So, these 3 columns; so you need not we have done it for our understanding ok, but so, that does not constitute the truth table. So, truth table you will have some an input part and an output part the input part will have all the variable combinations and the output part will have the corresponding value of the function ok. So, we can find out so, that is the function that we are talking about.

So, similarly you can do the other part that this X bar Y Z bar X Y bar. So, this thing; so this is one problem it says that verify that P plus P Q bar is a tautology.

(Refer Slide Time: 17:24)

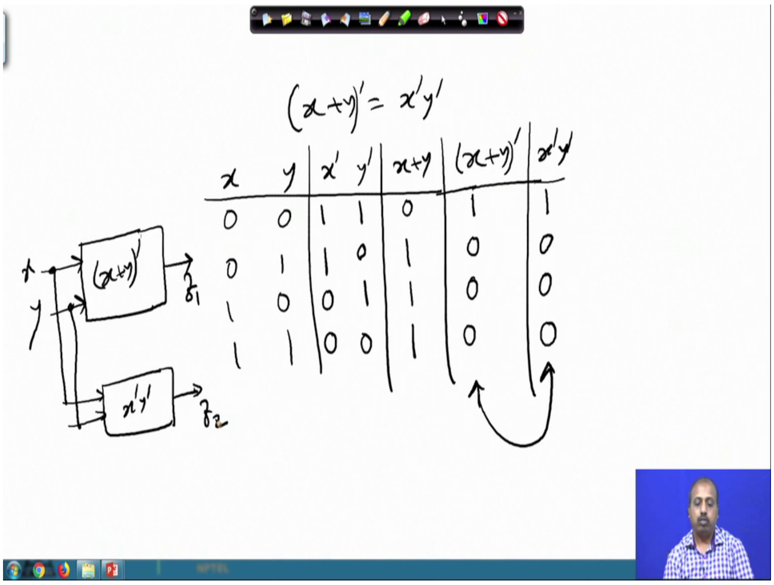


So, let us see how to do this. So, P plus P Q bar P plus P Q bar is a tautology. So, how do we do this? So, there are 2 variables in this expression P and Q. So, accordingly we take 2 columns P and Q. So, they can take up the value 0 0 0 1 1 0 and 1 1. After that we have got P Q here as if as a component function. So, P Q is 0 this is a AND function. So, these are all 0 only when both P Q are 1, then the output is 1. After that we have got this P Q bar. So, when I take P Q bar. So, this is 1, this is 1. So, this is the complement of P Q. So, this is like this. So, what is P plus P Q bar? So, P plus P Q bar so P plus P Q bar; so 0 1. So, this is 1, this is 0 this is 1 or of them. So, this is also 1. So, this is a one and one this is 1 and here P is 1 and P Q bar is 0, but there is an or function. So, that is 1.

So, you see that this is a tautology because at the output column we have. So, it is always 1, it does not depend on the values of P and Q. So, if I have got a functional block, which has got 2 inputs P and Q and it computes this P plus P Q bar; it computes P plus P Q bar then whatever value you give to this P and Q inputs. So, it will always output a 1 or true. So, this is a tautology. So, that is the proof. So, when a whenever if it is required to show that some expression is a tautology or a fallacy, what you need to do is, you have to draw the corresponding truth table and in the truth table you show that the output column is always 1 for a tautology and always 0 for a fallacy.

So, next we take the other another example. So, it says that we verify X plus Y bar is equal to X bar Y bar. So, let us see how can we do this using this truth table. So, our problem is to show that x plus y bar equal to x bar y bar ok.

(Refer Slide Time: 19:41)



So, how to do this? So, we can we can as we are using a truth table method. So, there are 2 variables x and y. So, we take we write down the corresponding input possibilities. So, it is 0 0 0 1 1 0 and 1 1. Now as a component I have got I need this x bar and y bar also. So, I write down the x bar part the compliments of x similarly y bar. So, that is not of y then I have got x plus y. So, what is x plus y? So, this is OR function of these first 2 columns or of first 2 columns. So, this is 0 0 1 1 1.

So, next I can write down this x plus y bar. So, x plus y bar is the complement of this. So, it is 1 0 0 0 and what is x bar y bar? It is the AND of these 2 columns ok. So, this is 1 0 0 0 now you compare between these 2 columns of the truth table. So, they are always same ok. So, that proves that all possible assignments of x and y.

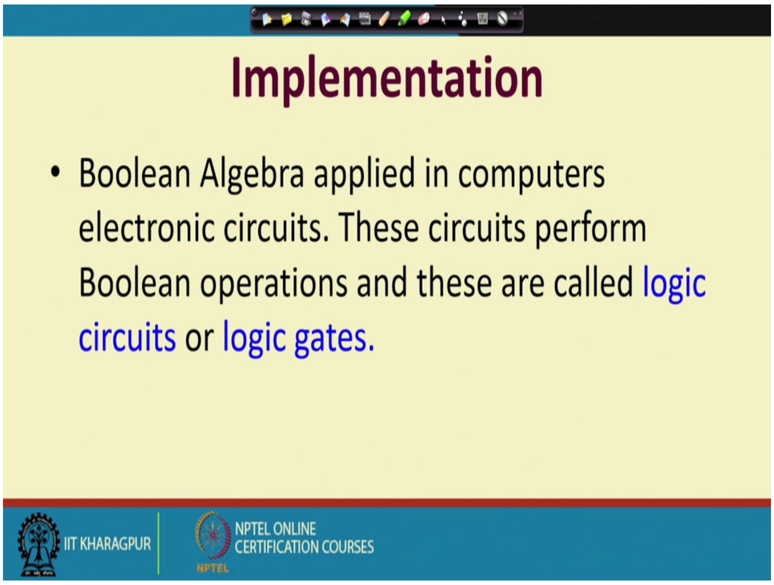
So, so this is the left hand side function and the right hand side function they will produce the same output conceptually. So, if I have got 2 functional blocks one is computing this x plus y bar and another is computing; so x bar y bar. So, if I feed the same values of x and y to both of them. So, I give the same values of x and y to both of them. Then the output that you will get f 1 here and f 2 with there so, they will always be same because these truth table says that if you feed the same values of x and y to both of them. So, they will give the same result.

So, this way by using truth table so, you can verify many of the Boolean formulas and they try to establish equivalency between Boolean expressions and all ok. But of course, if the number of variables are more then this is not a very good method because you will have to draw a table that has got 2 power n rows in it. So, here the number of variable n is equal to 2, but suppose the n becomes equal to 10 or 15 or 20 or 100 like that. So, drawing a table with say 2 power 10 rows 1024 rows or if it is a 100 variable function to 2 power 100. So, that is a very huge number. So, that. So, it is that then this method is not very much suitable ok. So, we have to do we have to use some other tricks and in fact, this is a very difficult problem to solve. So, you do not have any straight cut answer to this thing.

So, let us go back and see. So, we have seen how to check a tautology, how to check some relationship etcetera; now in case of now how this Boolean algebra. So, why this thing happened that, it was invented long back, but it is it was not used in the practice ok. So, as a mathematical tool it was there, but it was not being practiced in the in the say scientific community.

The reason is that the implementation was not there. So, we could not implement this AND gate or as this AND logic OR logic NOT logic like that. So, with the advent of these digital circuits, what has happened is, these implementations became possible. So, this AND OR NOT functions you can implement very easily.

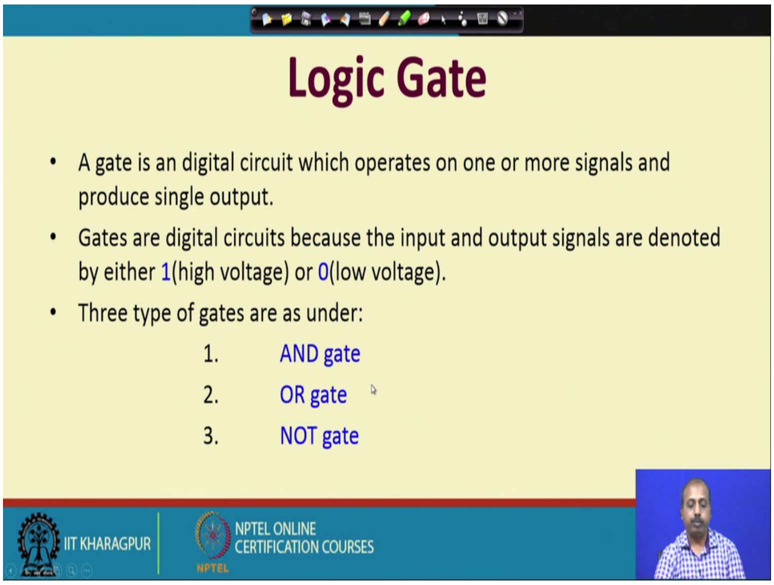
(Refer Slide Time: 23:40)



So, that gives that gives the impetus like why the why should we work on this, on Boolean algebra for this logic circuits.

So, Boolean algebra is applied in computers and electronic circuits and these circuits perform Boolean operations using something called logic circuits or logic gates. So, we will see the some elements. So, logic gates are again you can say it is some sort of electronic structure. So, that can that can implement in AND operation, OR operation, NOT operation like that.

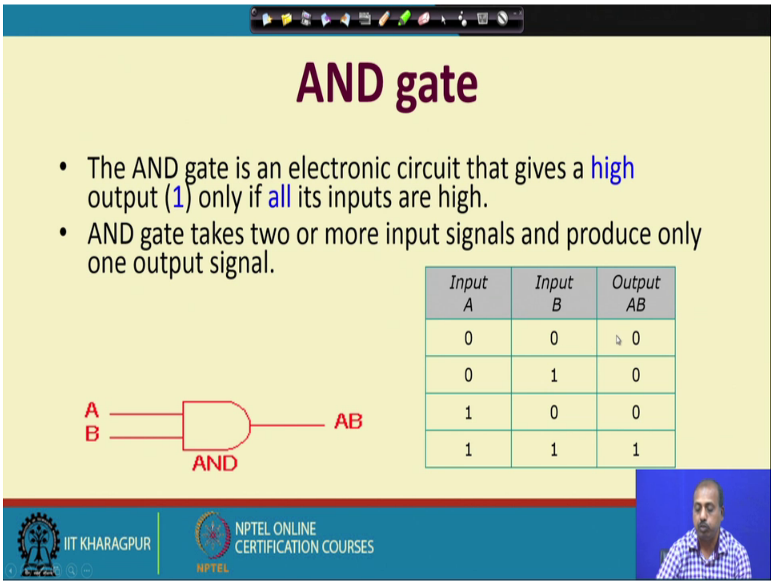
(Refer Slide Time: 24:15)



So, a gate is defined to be a digital circuit, which operates on one or more signals and produce single output. So, it has got a number of inputs and a single output. So, based on the values of input so, it will produce some output. So, gates are digital circuits because the input and output signals are denoted by either high or the either 1 that is often represented by a high voltage and 0 often represented as low voltage. Again the same thing that high voltage low voltage these terms are a bit confusing so, it may be other way also; so we call it at logic high and logic low. So, 1 is logic high and 0 is logic low.

So, as we are considering the 3 fundamental type of operations. So, in case of these digital circuits also. So, you will find that there are 3 basic types of gates that we have. One is called AND gate one is called OR gate another is called NOT gate. So, AND OR and NOT. So, these are the 3 fundamental gates that we have, and then we can. So, using this AND gate. So, we can mimic the behavior of this AND operation, using or gate we can mimic the behavior of OR operation and NOT gate can mimic the behavior of NOT operation.

(Refer Slide Time: 25:36)



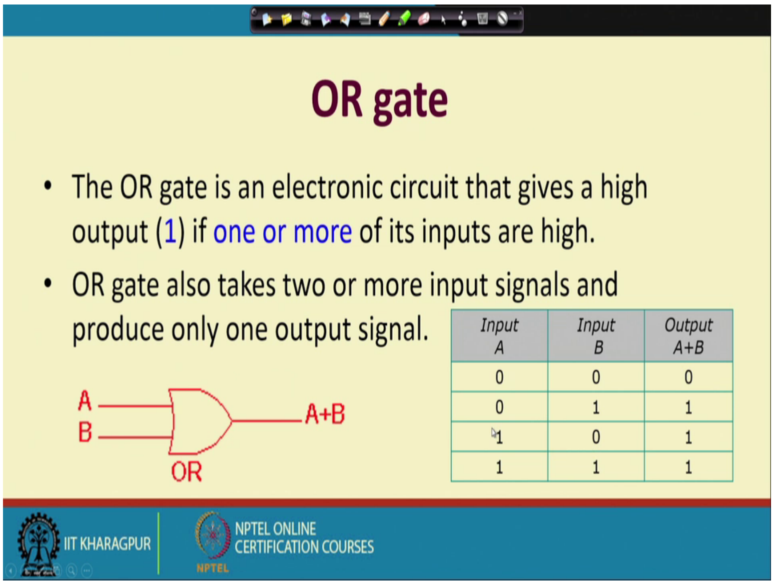
So, let us see what is an AND gate. So, AND gate is again another one electronic circuit that gives high output only if all its inputs are high ok. So, there can be it can take 2 or more input signals and produce only one output signal. So, it takes a number of inputs and it produces output. Output is 1 or high, only when all the inputs are 1. So, that was the AND truth table if you remember. So, the truth table output column. So, it had 1 only when all the inputs were equal to 1, any of the inputs being equal to 0 the output was equal to 0.

So, in case of symbolically it is represented like this. So, we one straight line and then 2 straight line in front of it there is an ellipse. So, this is the symbol. So, whenever we are trying to represent one AND gate. So, we will be using this symbol. So, this is a 2 input and gate. So, there are 2 inputs A and B and one output which is marked as AB here. So, you can have multiple inputs like you can have as a logically there is no limitation on the number of inputs that you can have to an AND gate. So, of course, the minimum is true, but you can have any higher number 3 4 5 6 ok.

So, there is there is no such limit, but practically of course, there will be a limit because there will be some digital circuitry which will be there inside this AND gate and that cannot be infinite. So, logically there were logically the any number of inputs may be there more than 2 more than 1, but there will be a physical limit.

So, this is the behavior of this and circuitry, whatever circuitry we put here it should behave in this fashion that when this A and B both are equal to 0, the output should also be 0. When one when both of them are equal to 1, then the output should be equal to 1 otherwise whenever at least one of the inputs is equal to 0. So, it should be the output should be equal to 0. So, that that is the basic AND operation.

(Refer Slide Time: 27:42)



The next gate next fundamental gate that we will consider, is the OR gate. Again the same thing that OR gate is an electronic circuit that gives high output if one or more of its inputs are equal to high. So, this is again depicted by the truth table that we you see that we have seen for the OR gate for the OR functionality. So, whenever all the whenever at least one of the inputs is high, the output is high and when all the inputs are equal to 0, then only output is 0; so again the same thing. So, you can have at least you should have at least 2 inputs, but theoretically you can have any higher number of inputs, but practically there will be a limit. This is the symbol for the OR gate with the 2 input OR gate. So, of course, you can have more number of inputs.

The OR gate the OR gate truth table is like this. So, this for this whatever circuitry we put here should behave in this fashion, that only when both A and B inputs are at logic low level 0, the output will output should be equal to 0 or logic low otherwise the output should always be high. So, that is the OR operation of A and B.

(Refer Slide Time: 28:56)

Then the NOT gate: so NOT gate is again another electronic circuit that gives high output if its input is low and it takes only one input signal. So, it is a single input gate unlike an AND gate AND gate and OR gate. So, this NOT gate is a single input gate. So, this is if a output is the complement of the input. So, if input is 0, output is 1 and if input is 1 output is 0.

So, this is also often known as inverter. So, this is NOT gate another very common term for this is inverter. So, it is symbolically represented like this, one triangle and a bubble at the beginning at the end of it, and then A and output is written as A bar and the truth table will be like this. So, it is input A and output is A bar. So, whenever this is 0 output is one and whenever input is 1 output is 0.

So, in this way we can have basic AND OR and NOT gates. So, in the digital circuits they can realize the AND OR and NOT functions of Boolean algebra.