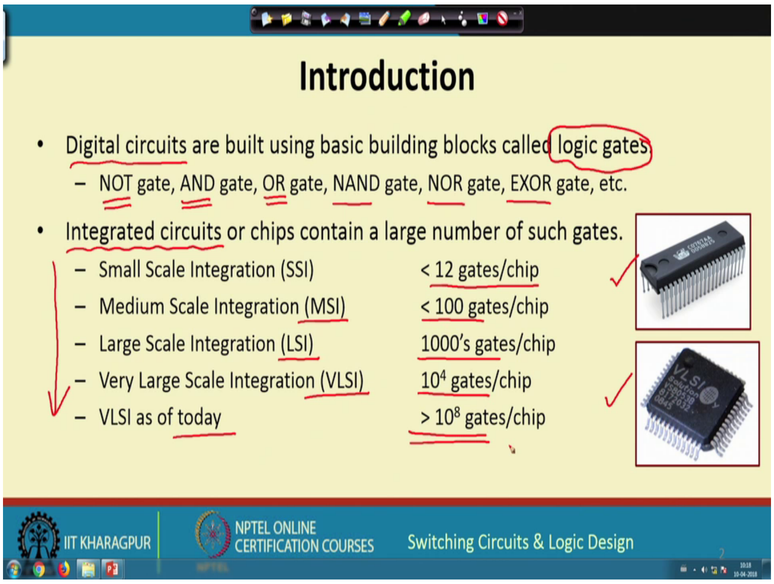
Switching Circuits and Logic Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 07
Logic Gates

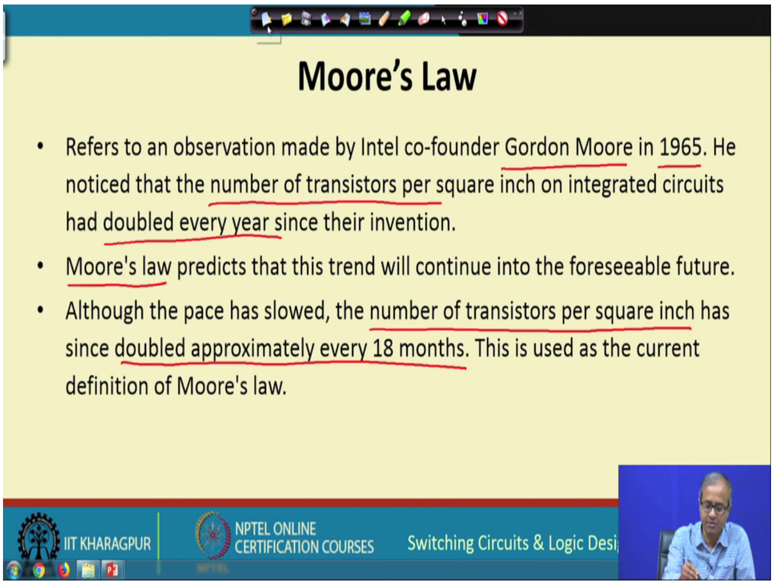
So, we start with our discussion on so called Logic Gates from this lecture. The title of the lecture is Logic Gates. Now, if you recall we have talked about numbers, number systems, binary primarily and other derived number systems like octal hexadecimal, decimal grey code etcetera BCD. Now, when we talk about binary number system the next step or the objective of the present course will be to built some circuits that are based on the binary number system or something which is called switching circuits.

This logic gates form the basic building block of such switching circuits. So, we first talk about the different kinds of common logic gates that are used which we shall be using in our subsequent lectures and discussions ok, logic gates fine.

(Refer Slide Time: 01:19)



(Refer Slide Time: 01:21)



So, the first thing is that we are talking about designing digital circuits this logic gates will be the basic building block of such circuits. There are many different types of logic gates which have been reported by various people. The most common logic gates are NOT and AND, OR this three are the most basic once and of course, NAND NOR or exclusive or EXOR these are also quite popular, there are few others.

Now, when you design a circuit nowadays we talk about integrated circuits there is a large amount of circuitry which are packed or integrated within a single integrated device. So, you can see such integrated circuits in every circuit board you open a computer, open a laptop, open any open a mobile phone, anything that you see you open a washing machine, open a air condition whatever we have you will see the there are lot of electronic circuitry and there all in terms of integrated circuits.

Now, here have shown you some typical pictures of integrated circuits how they look like they can vary in shapes and sizes. And over the years there have been several generations of integrated circuits which have come up; starting with the 19 late 1960's where we talked about something called small scale integration where inside these are called chips; integrated circuits are sometimes are also called chips.

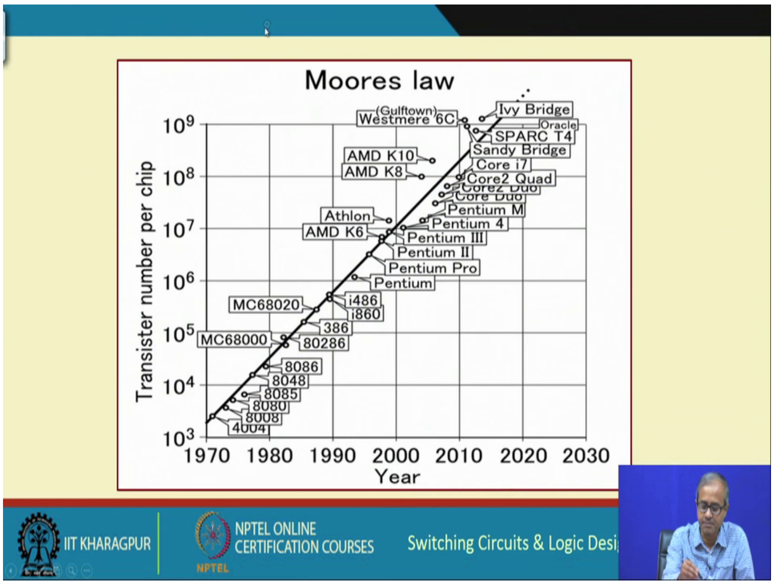
There will be about 10 to 12 maximum gates per chip very small circuits, then came in the 70's something called medium scale integration, MSI where we were able to pack about 100 gates in a chip. Then came large scale integration where we are talking about

1000’s of gates, then very large scale integration 10’s of thousands of gates and this VLSI is actually progressing as of today we are able to pack almost in excess of 10 to the power 8 or 100 million gates per chip. So, you see today we have chips of fantastic complexity with so huge number of components or basic gates inside a chip that allows us to build incredibly complex systems in a very compact space because the sizes of this integrated circuits are very small ok. This gives us a very big advantage in system design.

Now, another thing I just wanted to mention there is a emphercial observation which was made by Gordon Moore who was the co-founder of Intel long back in the 19 year 1965. So, what he predicted based on his observations is that the number of transistors in a chip basically should double every year. So, in the initial years this trend was found to be hold holding good. The so called Moore’s law predicts that such trend will continue in the foreseeable future.

See the pace has slowed down over the years ok, but what we see today is that still today the kind of technological advancements that we see that has allowed us to have the number of transistors per chip or per square inch whatever you say, actually we are talking a per chip to double approximately every 18 months. See the initially Moore talked about 1 year, but now is about 18 months. This actually means an exponential growth it is doubling every fix number of months right.

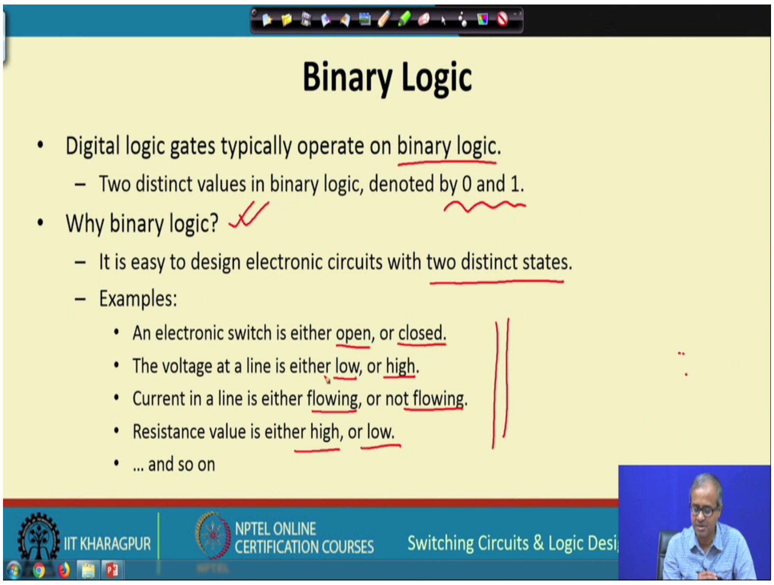
(Refer Slide Time: 05:50)



So, a very typical curve is shown here. So, here the x-axis shows the years this is projected up to 2030 and y-axis shows the number of transistor per chip on a logarithmic scale. So, straight line means exponential growth.

So, here we are only showing the Intel's and a few others Motorola, AMD mostly Intel the ones on the bottom these are Intel processors. So, you see that the trend the showing that exponential growth and this is continuing till today right.

(Refer Slide Time: 06:35)

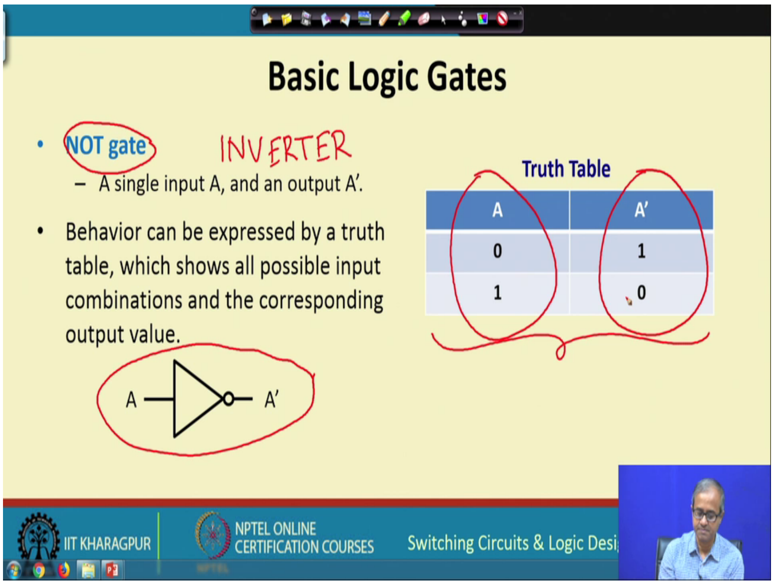


Now, let us come to binary logic. This digital logic gates that we are going to talk about they typically operate on binary logic. Now, in binary logic we have already seen we talked about two digits 0 and 1. So, the question is why do we want to go for binary logic and not other logic like decimal octal anything else. The first thing is that when we talk about designing circuits, circuits are built using small electronic components, most of these components can be very easily visualize has having two distinct states.

If you can map this distinct states in to those two values 0 and 1 then there can be a direct one to one correspondence between the circuits and the binary numbers that we want to talk about or we want to compute using this circuits ok. So, this electronics circuits can be visualized as having let us say two distinct states.

Some typical example a miniature electronics switch can be either open or closed two states. The voltage at a line you can say it is either low voltage or high voltage two states; current is either a flowing or not flowing, resistance value is either high or low. So, there are many technologies which is either opening or closing of switches, voltages currents resistances in various ways to represent this two states ok. This is the basic idea behind binary logic.

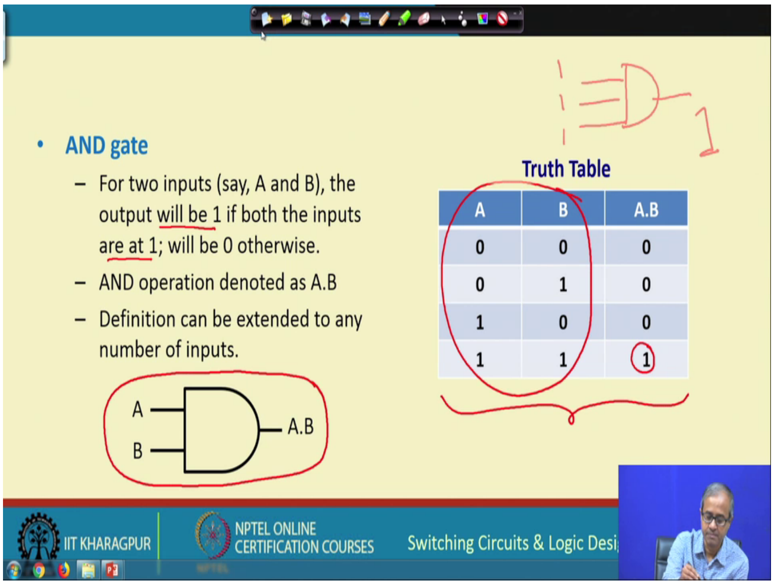
(Refer Slide Time: 08:28)



Now, let us come to the basic logic gates. We start with the simplest gate which is called a NOT gate and also sometimes it is referred to as an inverter because it inverts the state of the input line and NOT gate has a single input and a single output; pictorially it can be depicted by symbol like this. So, your input is A and the output you refer to as A bar. Now, the behavior of a NOT gate can represented by something called the truth table.

Now, what is truth table in a truth table on one side we show the inputs, on the other side we show the expected outputs. Now, here there is a single inputs so I am only showing A and I am listing all possible values of A can be 0 and 1 and what is the expected value it is the invert of that reverse of that if the input is 0 output will be 1. If the input is 1 output will be 0 that is how a NOT gate works ok.

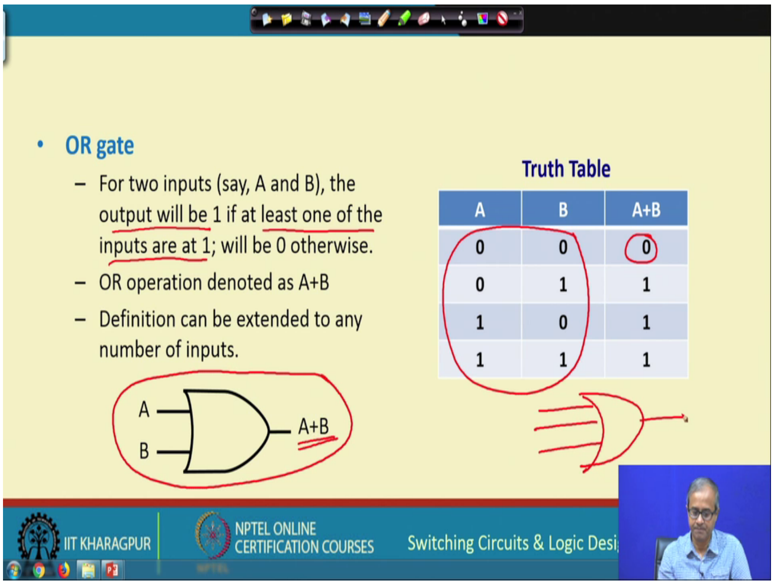
(Refer Slide Time: 09:52)



Let us move on to the other gates. Next comes AND gate, well AND gate the first thing is that AND gate can have two or more inputs. So, here I am showing you the symbol of an AND gate with two inputs A and B and the output the AND function is denoted like this with a dot A dot B. Let us first understand what or how an AND gate works this is depicted by the truth table again. So, here are your inputs where all possible values of A and B are shown is 0 0 0 1 1 0 1 1 and this is the expected output.

So, what does the AND gate do, the output will be 1 if both the inputs are at 1. Otherwise it is 0 you see when both the inputs are 1 only then the outputs are 1 otherwise it is 0. Now, this concept you can also extend to an AND gate with three number of inputs or any number of inputs where the output will be 1 only when all the inputs are 1 and it will be 0 otherwise. This is the definition of an AND gate that the output is 1 if A and B and C whatever inputs are there all of them are 1 and if all of them are not 1 then the output will be 0; this is and gate.

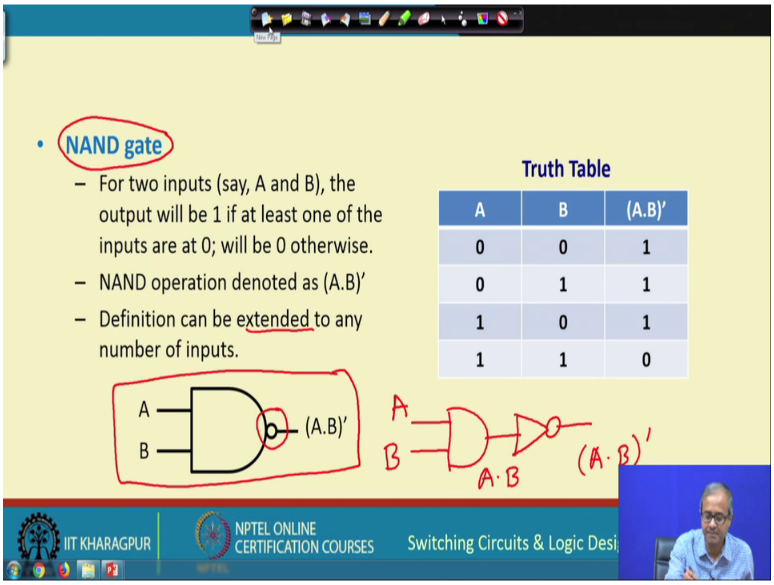
(Refer Slide Time: 11:28)



Similarly, you can have something called an OR gate where firstly, this symbol is like this, this is the OR gate symbol and OR you denote as plus A plus B in logic denotes not addition, but odd operation. Here the idea is that the output will be 1 if at least one of the inputs are at 1.

So, you see for a two input gate these are the inputs, the output will be 1 if either the input are 0 1 1 0 or 1 1; only when both of them are 0 only then the output is 0 otherwise the output is 1. So, here again you can extend the definition of a OR gate to include any number of inputs. So, the output will be 0 if all the inputs are 0 otherwise the output will be 1 fine.

(Refer Slide Time: 12:30)

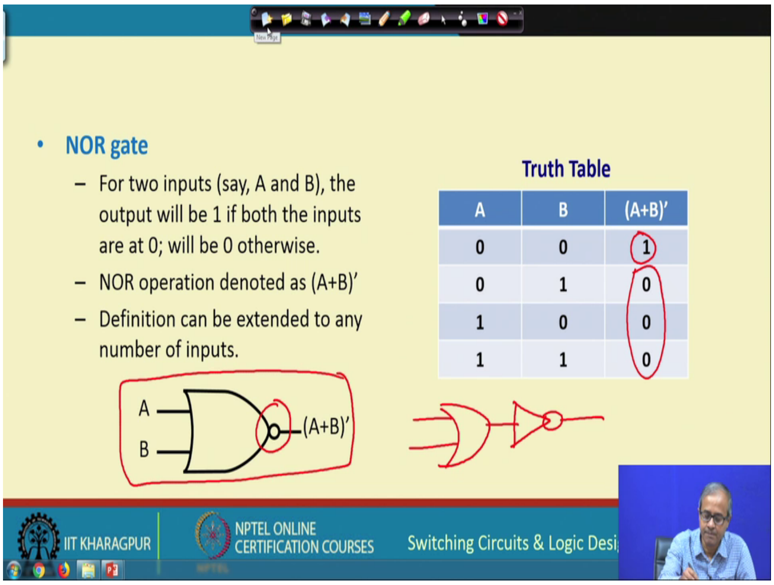


Now, let us come to slightly you can say this is like a composition of two gates. So, I will explain what I mean by that something called a NAND gate. First look the symbol you see NAND gate it uses the symbol of an AND gate with a small bubble at the output and symbolically I show it A dot means and I told you and B whole bar.

So, essentially NAND gate means that I have an AND operation followed by a not; this is my NAND operation. So, if this is my A and B so, I have A and B out here and A and B bar out here right. So, the operation will be just opposite to AND you see the output will be 0 if both the inputs are 1 otherwise the output is 1, for AND gate the output was 1 when both the inputs were 1.

But since, because there is a NOT operation in the output so, the output will be 0 if both the inputs are 1. Now, here gain you can extend NAND gate to any number of inputs right. So, functionally a NAND gate is nothing, but an AND gate followed by a NOT gate that is why I have said this is like a derived gate; if you have AND and NOT you can also make a NAND.

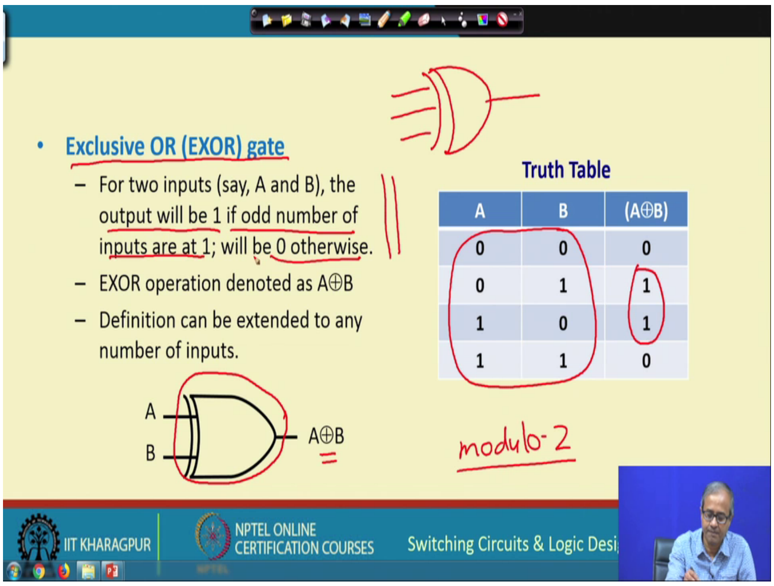
(Refer Slide Time: 14:19)



Next comes NOR gate which is an extension of OR gate again. So, again NOR gate looks like this it is like an OR gate with a bubble indicating a NOT operation in the output. So, here again if we have an OR gate and a NOT gate this makes a NOR gate.

So, here for an OR gate you recall the output was 0 only when both the inputs are 0 for other cases output was 1. So, you have a NOT so it has reversed for 0 0 the output is 1 for all other combination the output is 0. This is what a NOR gates means.

(Refer Slide Time: 15:11)

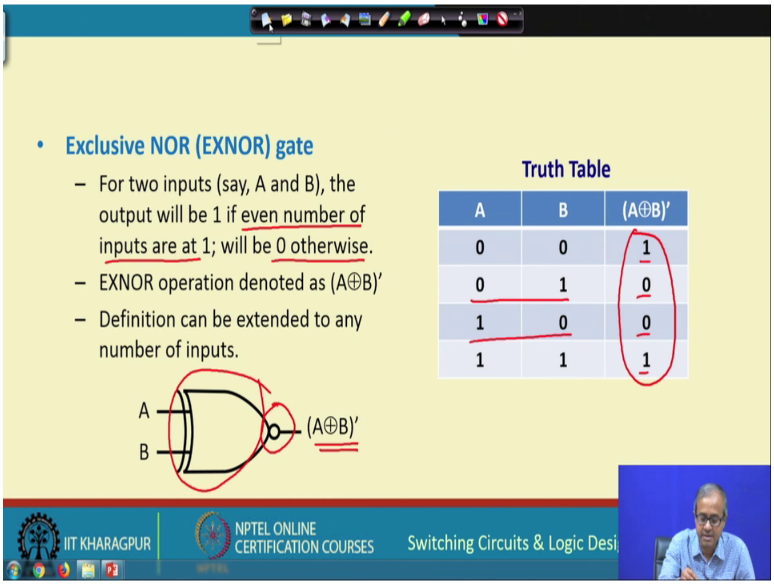


Now, let us come to something called exclusive OR gate well you have already seen this symbol. So, I mentioned the symbol is modulo 2 sum when we talked about it. Now, let me tell you this modulo 2 sum is nothing, but the exclusive OR operation because now you are introducing the specific gate exclusive OR gate or EXOR in short. This symbol is like this a OR with a double line in the input side this is a symbol of an exclusive OR. Well for a two input exclusive OR, if you look at the truth table this is the input. The output will be 1 if the inputs are 0 1 and 1 0 which means exactly one of the inputs are at 1, but if both the inputs are 0 or both the inputs are 1 then the output is 0.

So, here again you can extend an exclusive OR definition to any number of inputs, but what will be the definition of such a gate. Well so, instead of saying that 1 is 0, 1 is 1 we say that the output will be 1 if odd number of inputs are at 1 you see the output is 1 if odd number of inputs are 1 or how many are 1, 1 1 is odd, 1 odd, 2 is even, 0 is even and if it is even it will be 0.

So, for larger gates the output can be defined like this right. So, if odd number of inputs are at 1 the output will be 1, if even number of inputs are 1 the outputs will be 0 fine.

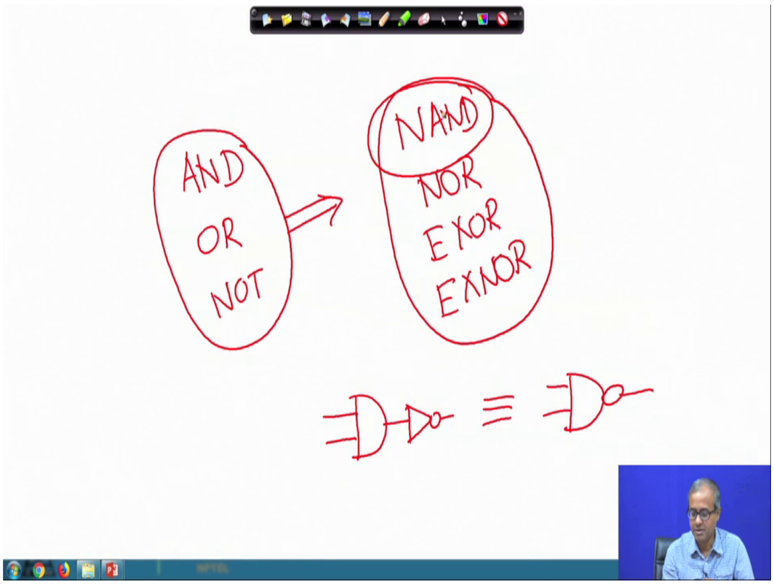
(Refer Slide Time: 17:12)



Now, let us come to something called exclusive NOR gate. Now, exclusive NOR gate is just the exclusive OR you see there is an exclusive OR symbol and there is a NOT symbol or a bubble at the end. So, in the output column you see the just the reverse when you have 0 1 or 1 0 for EXOR the output is 1 now it is 0, but for the other combination it is 1.

So, definition wise the output will be 1 if even number of inputs are 1; that means, 0 0 number of 1's is 0 even 1 1 number of inputs is two that is also even, but if it is odd it will be 0 and symbol and symbolically and expression operation is denoted like this EXOR bar.

(Refer Slide Time: 18:20)



So, one thing you see here you have the basic operations like I talked AND, I talked about OR, I talked about NOT. These three constitute a basic operations and there are derived operations like NAND NOR EXOR and EXNOR; derived means these can be realized using basic gates also. Like as I said an AND gate followed by a NOT gate is equivalent to NAND gate ok. Later on we see that not only this way some of these gates for example, NAND. This NAND gates can also be used to realize any other types of gates or also NOR gates, this we shall see later.

So, we have talked about the various kind of gates this AND OR NOT NAND NOR EXOR EXNOR. So, in our next lecture we shall be talking about how some of this gates can be actually implemented or realized following which we shall be going into the design right, when you have some function or some application how do use this gates to design our functionality.

So, this will be our flow during the next few lectures. So, with this actually we come to the end of this lecture. So, as I had said in the next lecture we shall be talking about some of the ways in which these gates we had discussed today can be actually built.

Thank you.