Computer Numerical Control of Machine Tools and Processes
Professor A Roy Choudhury
**Department of Mechanical Engineering**
Indian Institute of Technology Kharagpur
Lecture 2
Binary Logic and Logic Gates

Welcome viewers to the 2nd lecture of the online course Computer numerical control of machine tools and processes. Today we will be discussing about binary logic and logic gates. So what are we really concerned about? We are concerned about 1st of all signals which are employed in case of control of CNC machines, they are in many cases they are binary valued signals that means they can take up 2 values and in that case they undergo a number of combinations and they are used in decision-making in CNC machines and we will be having a quick look at these typical signals.

(Refer Slide Time: 01:26)
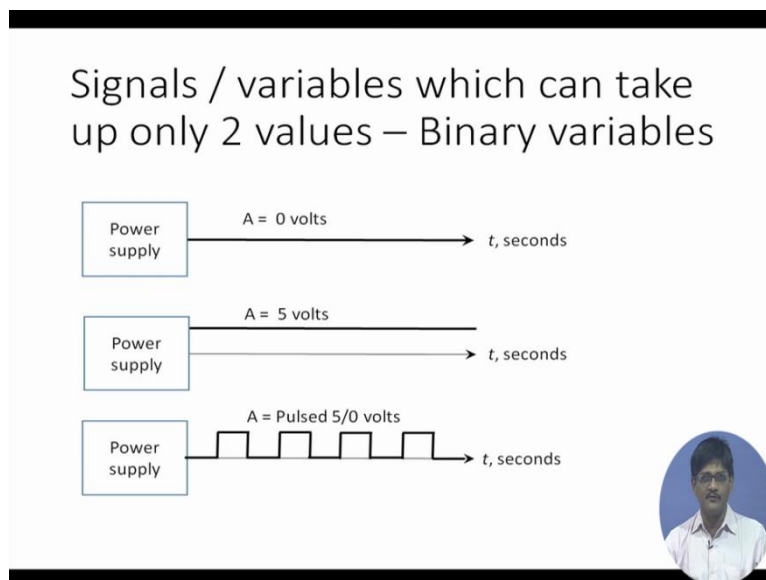
From the power supply you might be having a signal and it can take up say value of 0 volts and which is sometimes referred to as low signal or 0 volts or a value numerically 0 like that, a value of 5 volts may be referred to as high or 1 or 5 volts. It does not necessarily mean that it has to be a high voltage in the in order to be considered high, a low voltage also might be considered to be a high it depends upon what convention we are following. So in the same way a single might be varying between high-voltage and low voltage so that we can have a pulsed voltage and as shown in the 3rd picture that A is taking a value of 5 volts or 0 volts and changing over time, so these are typically some digital signals that we are going to come across. And let see what kind of operation can be done with them and what kind of mathematical or logical operation they follow.

Truth tables for some logic operations

A	B	A′	B′	A + B	A.B	A ⊕ B	A#B
1	1	0	0	1	1	0	0
1	0	0	1	1	0	1	1
0	1	1	0	1	0	1	1
0	0	1	1	0	0	0	1

Now, so generally when we are having these binary valued variables for example, A and B. and as discussed before they can take up 2 values. We are referring to those 2 values as 1 and 0, they might be 5 volts and 0 volts, and they might be high-level, low-level whatever we are referring to them as 1 and 0. So there are certain relations between them, which are defined by these logic operations. Mind you, in some cases they might be having mathematic like symbols, but they are not exactly mathematical operations at least at this moment we are only discussing logic operations. So, how do these logic operations formulate?

For example if I write A′ it means just the opposite of A. A can have only 2 values, so if A is 1, A′ or A´ is 0, it is also sometimes referred to as complement of A. In the same way if B is 1, then B′ or B´ is 0, if A is 0, then A´ is 1 like that. So we have a list, possible values of A, possible values of B giving rise to 4 possible combinations. And for that we have A´ and B´ also written down, so if A is 1 1 0 0, A´ must be 0 0 1 1 just the opposite. In the same way, there are some relational logic operations carried out between A and B.

Say A + B is written, it should be read as A OR B, it is a logic operation which says that A OR B is a function which will be having a value 1 if A and B both are 1 or if A is 1, B is 0 or if A is 0, B is 1 and all the other cases it is 0. So it is something like let me give you an example, suppose you and your friend, you are going for a particular celebration or going to a movie, et cetera and the doorman asks for tickets. Suppose, both of you have your tickets, the doorman passes you, only one of you have the ticket then also say he is very lenient he passes you, but if both of you do not have a ticket then he does not passes you, this is A OR B.

So A OR B is represented as 1110, what is A AND B? A AND B is not that lenient. Suppose the same doorman says both of you have to have tickets otherwise I am not going to let you in, he represents A AND B, only when A is 1 and B is 1 only then A and B will be retaining a value 1, in all the other cases no, 0. A XOR B gives another relation, it says that if either A or B is 1, but not both then the result is 1 so even if both of them are 1 then also it does not return 1, it is called Exclusive OR. And in the last column we have written A # B, just some arbitrary operations is defined as A # B.

If I ask you, can you tell me what kind of logic operation it represents? So in that case you can do one thing, you can apply your common sense to find out okay, does it look like any other logic operation? A OR B? No, A AND B? No, A XOR B? No. But if you complement A AND B, that means A AND B whole´, then you will get A # B, this is a way in which in many cases you will be able to build up circuits by comparing it with existing circuits.

(Refer Slide Time: 07:23)

So this way binary logic operations execute. They have certain number that they obey certain laws which we will have a quick look. For example, if you think of the compliments, A OR A´ = 1, why? Because at least one of them would be 1 and OR means if one of them is 1, you will get 1 as answer. A AND A´ is 0 because at least one of them will be 0 and if you have a 0 ANDED with another variable, you will get 0. So in the same way A OR 1 is 1, A AND 0 is 0, A AND 1 is A, so on and so forth.

The commutative law looks very much like addition, multiplication, et cetera, A OR B is B OR A, A AND B is B AND A. The distributive law is similar to mathematical relation A

AND B OR C = A AND B OR A AND C. Associative law is the place where you will find it does not look like the mathematical fundamental operations that we have, A OR B AND C = A OR B AND A OR C. And De Morgan's laws are very useful for example; (A OR B)′ = A′ AND B′, et cetera. Now we come for some symbols of logic gates.

(Refer Slide Time: 9:11)

Fig 3.1 Logic Gate Symbols
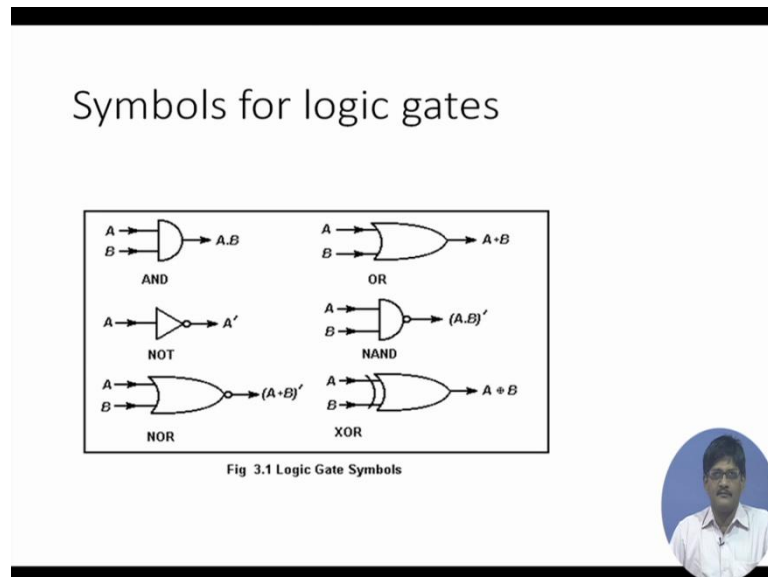
Logic gates can be realized in through electronic circuitry and this is how they are represented in by symbols. For example, two signals are coming into they are binary signals, they can only take up two values A and B and the result is A AND B and the half moon shape represents the AND gate. In the same way, on the right-hand side the signal for, symbol for the OR gate is shown, A OR B is the result. In the same way the NOT gate, that means the complement, the NAND gate the the combination of AND gate and NOT gate, et cetera, these are shown. Of interest to us is the XOR gate which is the Exclusive OR.

Addition of two bits

Decimal	Binary
1 + 1 = 2	1 + 1 = 10
1 + 0 = 1	1 + 0 = 1
0 + 1 = 1	0 + 1 = 1
0 + 0 = 0	0 + 0 = 0

A	B	Sum	Carry
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

| A	B	A'	B'	$A + B$	$A.B$	$A \oplus B$
1	1	0	0	1	1	0
1	0	0	1	1	0	1
0	1	1	0	1	0	1
0	0	1	1	0	0	0

Now let us do some exercises through which we will become conversant with the working of the logic gates. For example, say we set ourselves this particular task that we have to find out a logic circuit which will be able to mathematically add two bits. That means that I if I have 1 and 1, my circuit will be so designed that it can add up these two 1s and give me a result of two, which in binary is 10, it can add 1 and 0, and give a result of 1 and it can add 0 and 0 and give me a result of 0, so this is mathematical addition and with the help of logic circuits, logic operations I have to represent these mathematical operations.

So I have written down the possible results of mathematical additions. Suppose A and B are being added, if you refer to the blue table, 1 + 1 results in 10 which mean basically 2, so I have written down one column the carry column and the sum column, so 1 + 1 is 10, 1 + 0 is 01, 0 and 1 is 01 and 0 + 0 is sum 0 carry 0. So I have also written down the typical results that we have of A´, B´, A OR B, A AND B, etc, etc. So if we have to represent sum and carry or build up circuits which will be giving us results of sum and carry, we can 1st compare quickly with the established standard gates and find out whether they are already there.

For example sum, it is very much there A AND B, so sum is represented by, whenever you are adding two bits, bit means binary digit, when you are adding up two bits then sum is represented by A AND B and what is carry represented by? Carry is represented by A XOR B, so if I build a particular circuit where I am simply employing A and B for obtaining sum and A XOR B for obtaining carry, my job is done , let us have a look at that.

A half adder – can add up two bits
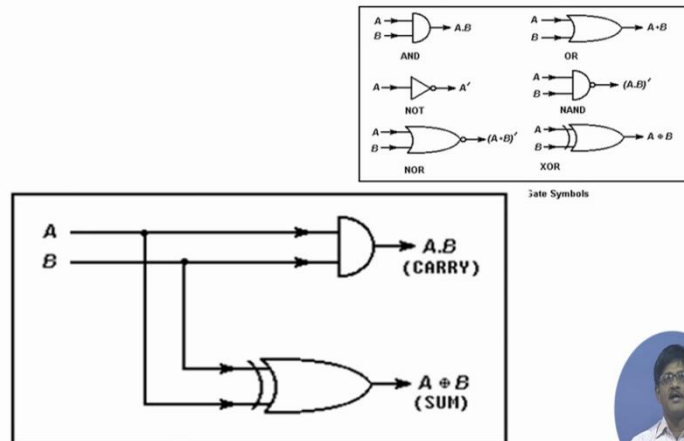
That's it, I am having A and B coming in, I have divided them and sent one pair of A and B to AND gate we are achieving carry as a result of this AND gate and we are also having an XOR gate which is giving me the sum, so this way I can add up two bits, whatever be their combination the results are always going to be obtained as carry as the output of AND and sum as the output of XOR gate, it is called a half adder.

(Refer Slide Time: 13:26)

Addition of 2 bytes

$$
\begin{array}{r}
\text{Carry} \quad 1011111 \\
11011011 \\
+ \ 11001111 \\
\hline
110101010
\end{array}
$$

• You might well have to add 3 bits together

However, if you look at say mathematical addition I have shown here 2 bytes are being added that means 8 digit binary numbers. If you add them, first 1 + 1 will yield 1 0, the 0 comes down, take the unit place and there is 1 carry over in the next column, so in the next column if you see the colour, the colour is different. This blue colored column is therefore having an addition mathematical addition of 3 bits. So you might well have the addition of 3 bits,

whenever you are adding up two bits together, so let us have of quick look how this can be done addition of mathematical addition of 3 bits.

(Refer Slide Time: 14:22)

Whenever we are thinking of mathematical addition of 3 bits, what I have done is I have put them in columns of different possible combinations and results of some and carry are written down and those particular results which are yielding a 1, they are highlighted. For example, A, B and C can take a combination of 1 1 1, which means that their addition will give us 3, this is represented by 11 in binary, so carry will be 1 and sum will be 1. In the 2nd row we have A 1, B 1 and C 0, their addition will yield a 2, which is 10 in binary, carry will be 1, sum will be 0. So this way we have built up this particular table in which different combinations of A, B and C, which are being added together, they are yielding different values of carry and sum.

In that, what we have done is we have pinpointed those values which are having a value of 1 or high, so that way if we look at say sum, I can say that sum takes up the value of 1 in 4 cases, if A, B and C are all 1 or if A is 1 and B is 0 and C is 0 in that case sum is 1, or if A is 0 and B is 1 and C is 0 or if A is 0, B is 0 and C is 1, in none of the other cases the sum is 1. And these combinations can be expressed logically, in what way? We have written down, sum is equal to that means sum is one in cases where A AND B AND C, the 1st condition is written down, A AND B AND C OR, OR is given as the the logical OR sign.

So we are basically writing sum = A AND B AND C OR A AND B´ AND C´ or A´ B and C´ OR A´ AND B´ AND C. That is all the cases where sum is 1 if we have written down and connected them up with proper logical operation. In none of the other cases, we are going to have a 1 for sum. And for these cases we have connected the variables in such a way that those unit building blocks, they are going to yield a 1. So suppose A is 1, B is 1, C is 1, the 1st term of sum is going to yield 1 and others are going to yield 0s, so sum is going to be 1 in that case, so this circuit will represent the value of sum in any of these 8 cases.

Let us say we take A is 0, B is 0, C is 0 all these terms will be 0s, so sum will be equal to 0. So it is built in such a way by taking individually the cases where it is equal to 1, we built up a circuit which is going to yield 1 only in these 4 cases, in other cases it is going to yield 0 and that way we have built up the logical representation of the mathematical operation of sum. In the same way if you take carry, carry is 1 in the highlighted cases and correspondingly we have built up the logical representation of these particular cases and connected them up with logical OR operation, in this case or this case or this case like that.

Now, is it what are we ultimately getting? We are getting sum to be represented by a particular logic circuit or combination of logic gates, how do we utilize it? We can simplify it, let us have a look.

(Refer Slide Time: 19:04)

The value of carry

$$Carry = A.B.C + A.B.C´ + A´.B.C + A.B´.C$$

$$= A.B.C + A.B.C´ + A.B.C + A´.B.C + A.B.C + A.B´.C$$

$$= A.B.(C+C´) + B.C.(A+A´) + A.C.(B+B´)$$

$$= A.B + B.C + A.C$$

For example carry, carry is A AND B AND C OR A AND B AND C´ OR A´ AND B AND C OR A AND B´ AND C. What we are doing here is that, since in logic Boolean algebra, A + A = A, I can add any number of As to itself to get the same value, so in the same way I can

add any number of A AND B AND C to this one without any change in the ultimate result. So what we have done is, we have added 2 more A AND B AND C to these terms, so what do we have?

The 1st term is A AND B AND C + A AND B AND C´ then again we add A AND B AND C, so this way the 1st 2 terms will allow us to take A AND B common with C + C´, the 3rd and the 4th terms will allow us to take B AND C common A + A´, so this way C + C´ = A + A´ = B + B´ = 1, therefore this comes out to be A AND B OR B AND C OR A AND C, so carry is simplified this way.

(Refer Slide Time: 20:34)

The value of Sum

$Sum = A.B.C + A.B´.C´ + A´.B.C´ + A´.B´.C = A . (B.C + B´.C´) + A´. (B.C´ + B´.C)$

Now, $B.C´ + B´.C = B \oplus C$, and

$(B.C + B´.C´)´ = (B.C)´. (B´.C´)´ = (B´ + C´) . (B + C) = B.B´ + B.C´ + B´.C + C.C´$

$= B.C´ + B´.C = (B \oplus C)´$

Hence,

$A . (B.C + B´.C´) + A´. (B.C´ + B´.C) = A.(B \oplus C)´ + A´. (B \oplus C) = A \oplus B \oplus C$

In the same way, after writing out expression of sum, we simplify it by taking A common from the first two terms getting B AND C + B´ AND C´ in bracket OR A´ common from the last 2 terms with B AND C´ + B´ AND C in brackets. Now B AND C´ OR B´ AND C is nothing but B XOR C, so the last bracketed term becomes B XOR C and the 1st bracketed term B AND C OR B´ AND C´ can be shown to be nothing but the complement of B XOR C, in what way?

We have taken its complement, applied De Morgan's law and ultimately obtained (B XOR C)´ and therefore, we are saying this particular term simplifies to A AND B XOR C´ plus sorry, A AND B XOR C´ OR A´ AND B XOR C = A XOR B XOR C, so this is the simplified form of the sum, let us take a small numerical problem.

There is a grinding machine in which coolant is to be applied automatically if the temperature is greater or equal to 60 degree centigrade, the surface speed is greater than 20 meters per second and the table feed is less or equal 15 meters per minute. And suppose we are having sensors which are taking some signals which are developing signals, where A = 0, when temperature is less than 60 degree and equal to 1 otherwise. B = 0 when the surface speed is less than 20 meters per second and 1 otherwise and C is 0, when the feed is less or equal to 15 meters per minute and 1 in other cases.

The question is make a circuit which would send an output signal of S = 1, when coolant is supposed to be applied otherwise 0. So look at the picture, the grinding machine has 3 outputs and there is a black box to be designed by you, which sends out S, which should be 1 when coolant is to be applied and 0 when coolant is not to be applied, so let us see what the circuit should be like.

1st of all, we have pinpointed the 3 cases where the signal is supposed to be 1, the output signal should be 1 in these 3 cases. And therefore we are giving this way that signal should be 1 following the argument of the previous example, signal should be 1 if A AND B AND C is 1 or A AND B´ AND C is 1 or if A AND B AND C´ is 1, so having understood that, what we are doing, we are taking ABC and adding one more ABC at the end.

So from the 1st 2 terms we are getting AC common, A AND C is taken common and it is ANDED with B OR B´ + AB is taken common and it is ANDED with C OR C´. So they are B + B´ I mean B OR B´ = C OR C´ = 1, so that they are ultimately having A AND C OR A AND B, which = A AND B OR C, so we have this simple circuit okay A ANDED with B OR C, this simple circuit would be controlling successfully the flow of coolant for a grinding machine. These 3 inputs just remind you, they are concerned with temperature, surface speed and feed of a grinding machine, thank you.