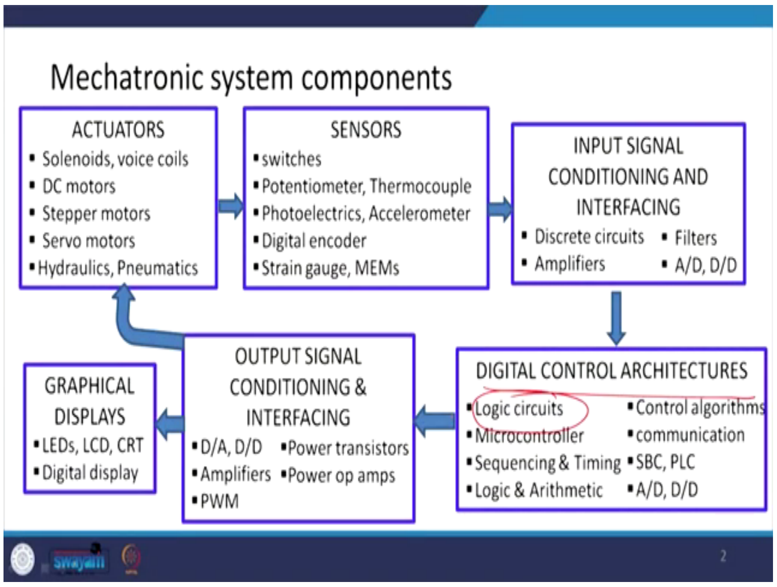
Mechatronics

Prof. Pushparaj Mani Pathak
Department of Mechanical and Industrial Engineering
Indian Institute of Technology, Roorkee

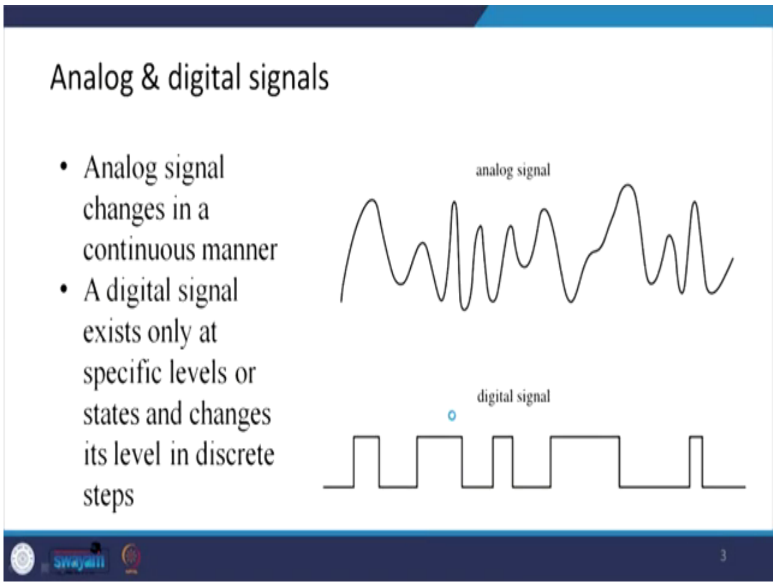
Lecture - 21
Digital Circuits – I

I welcome you all to today's NPTEL online certification course lecture on Mechatronics. Today, we are going to talk about Digital Circuits. So, digital circuits are part of every control system. Today all the microprocessors, microcontrollers, and computers are all operated with the help of digital circuits. So, in this block diagram for the components of a mechatronics system, we will focus on the digital control architecture today.

(Refer Slide Time: 01:17)



(Refer Slide Time: 01:20)

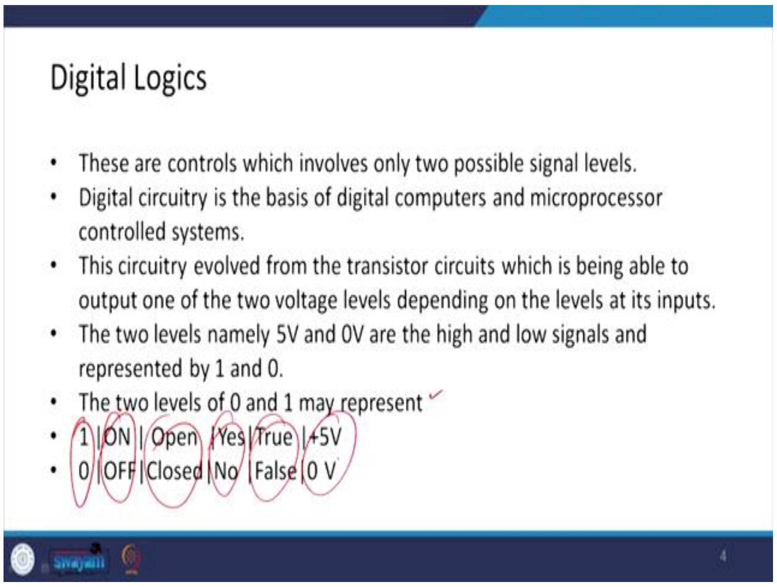

Analog & digital signals

- Analog signal changes in a continuous manner
- A digital signal exists only at specific levels or states and changes its level in discrete steps

analog signal
digital signal
3

Furthermore, we will be seeing the logic circuits. So, as I have been discussing with you earlier, analog and digital are the two types of signals, which are available. Analog signal changes in a continuous manner and the digital signals exist only at a specific level or a state and change their level in a discrete step, as you can see here. These digital signals form the basis for the control of these modern electronic devices.

(Refer Slide Time: 02:01)

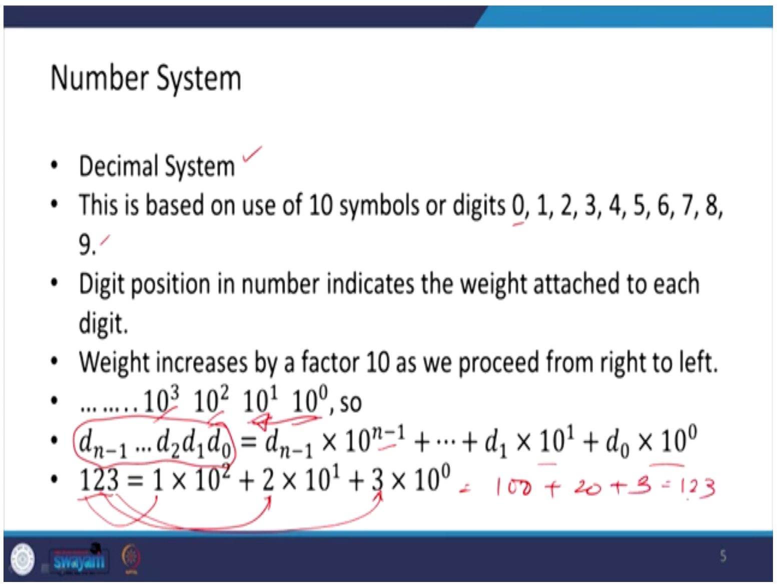

Digital Logics

- These are controls which involves only two possible signal levels.
- Digital circuitry is the basis of digital computers and microprocessor controlled systems.
- This circuitry evolved from the transistor circuits which is being able to output one of the two voltage levels depending on the levels at its inputs.
- The two levels namely 5V and 0V are the high and low signals and represented by 1 and 0.
- The two levels of 0 and 1 may represent
- 1 | ON | Open | Yes | True | +5V
- 0 | OFF | Closed | No | False | 0 V

Digital logics are controls that involve only two possible signal levels. Digital circuitry is the basis for digital computer and microprocessor control systems. This circuitry evolved from the transistor circuits, which can output one of the two voltage levels depending on its input level.

The two levels could be 5 volts and 0 volts, or the high and low signal represented by 1 and 0. So, the two levels of 0 and 1 may be what we can call either ON-OFF, open-closed, or yes-no, or true-false, or by +5 volt and 0 volt. So, this is how 0 volts can correspond to 0 signal, and + 5 volts can correspond to the one signal.

(Refer Slide Time: 03:21)



Number System

- Decimal System ✓
- This is based on use of 10 symbols or digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. ✓
- Digit position in number indicates the weight attached to each digit.
- Weight increases by a factor 10 as we proceed from right to left.
- …….. 10^3 10^2 10^1 10^0, so
- $d_{n-1} \dots d_2 d_1 d_0 = d_{n-1} \times 10^{n-1} + \dots + d_1 \times 10^1 + d_0 \times 10^0$
- $123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 = 100 + 20 + 3 = 123$

Before we proceed, let us look at the number system and various types of number systems. Furthermore, we will operate the binary signals with this understanding very proficiently. So, first of all, let us look at the decimal system. As we all know, this system is based on ten symbols or digits from 0 to 9, as all of us know.

Moreover, the digit position in a number indicates the weight attached to each digit, and this weight increases by a factor of 10 as we proceed from right to left. So, this is how the weights are attached that is 10^0, 10^1, 10^2, 10^3. It is increasing in this direction as we move from right to left. So, suppose a general number is there this one. This number

$$10^0 = d_{n-1} \times 10^{n-1}$$

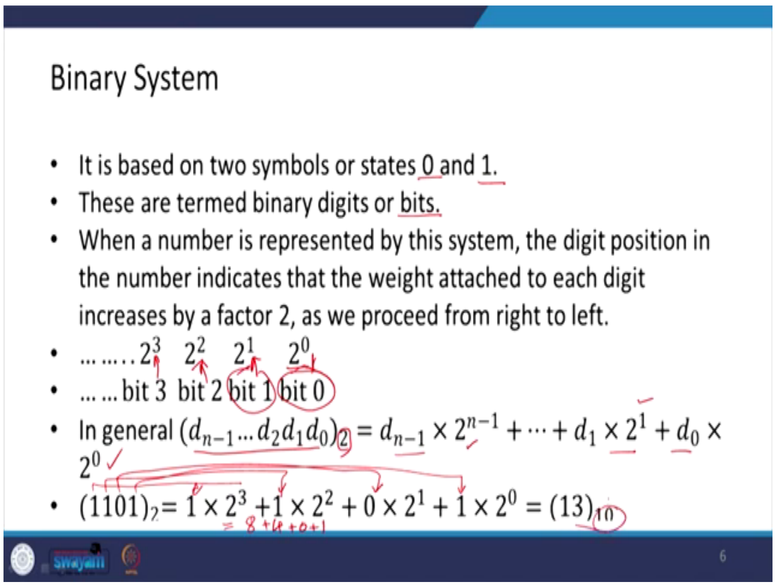
And, here $d_2 \times 10^2$, $d_1 \times 10^1$ and $d_0 \times 10^0$. So, for example, if you have a number,

$$123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

So, this way it forms what?

100+ 20+ 3 and we get 123. So, this is decimal system by which we all are very well aware of.

(Refer Slide Time: 05:31)

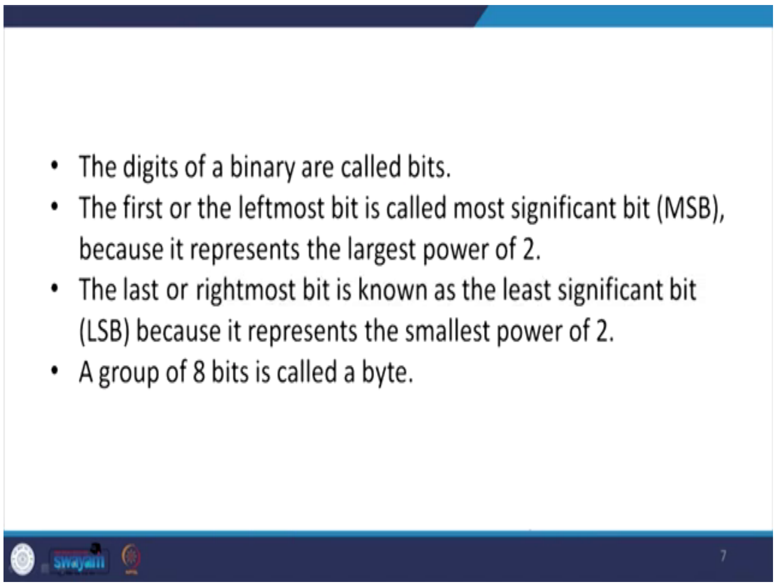


Binary System

- It is based on two symbols or states 0 and 1.
- These are termed binary digits or bits.
- When a number is represented by this system, the digit position in the number indicates that the weight attached to each digit increases by a factor 2, as we proceed from right to left.
- $\dots\dots 2^3 \quad 2^2 \quad 2^1 \quad 2^0$
- … … bit 3 bit 2 bit 1 bit 0
- In general $(d_{n-1} \dots d_2 d_1 d_0)_2 = d_{n-1} \times 2^{n-1} + \dots + d_1 \times 2^1 + d_0 \times 2^0$ ✓
- $(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (13)_{10}$
  $= 8 + 4 + 0 + 1$

Next let us see the binary system. This binary system is based on two symbols or a state that is 0 and 1 . These are termed binary digits or bits and when a number is represented by this system the digit position in the number indicates that the weight attached to each digit increases by a factor 2, as we proceed from right to left. So, here we have a factor the first digit will have a factor 2^0, the second will have a factor 2^1, the third will have a factor 2^2, and so on. So, these are this is are called bit 0, bit 1, bit 2 and this is bit 3. So, in general, if I have got a number this one here the two subscripts is used to indicate that it is in binary. So, this means that $d_{n-1} \times 2^{n-1}$ and here $d_1 \times 2^1$ and $d_0 \times 2^0$. So, for example, if I have got a number binary number 1101. So, here this 1 corresponds to this one. And its weight is 2^0; this 0 is this one with weight 2^1, this 1 is this 1 multiplied by 2^2 and this 1 is this 1 multiplied by 2^3. So, we can see that this is (8 + 4 + 0 + 1). So, this is 13 in the decimal system.

(Refer Slide Time: 07:45)

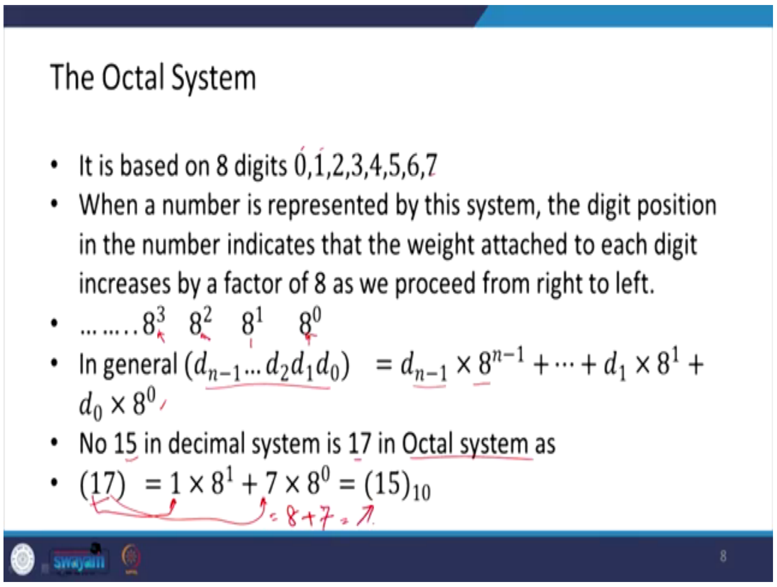


- The digits of a binary are called bits.
- The first or the leftmost bit is called most significant bit (MSB), because it represents the largest power of 2.
- The last or rightmost bit is known as the least significant bit (LSB) because it represents the smallest power of 2.
- A group of 8 bits is called a byte.

7

So, the digits of a binary are called bits. The first or the leftmost bit is called the most significant bit, we call it in short as MSB because it represents the largest power of 2. The last rightmost bit is known as the least significant bit LSB because it represents the smallest power of 2. And a group of 8 bits is called a byte. So, it is a little bigger slightly bigger unit.

(Refer Slide Time: 08:59)



The Octal System

- It is based on 8 digits 0,1,2,3,4,5,6,7
- When a number is represented by this system, the digit position in the number indicates that the weight attached to each digit increases by a factor of 8 as we proceed from right to left.
- $\ldots\ldots\ldots 8^3 \quad 8^2 \quad 8^1 \quad 8^0$
- In general $(d_{n-1} \ldots d_2 d_1 d_0) = d_{n-1} \times 8^{n-1} + \cdots + d_1 \times 8^1 + d_0 \times 8^0$
- No 15 in decimal system is 17 in Octal system as
- $(17) = 1 \times 8^1 + 7 \times 8^0 = (15)_{10}$

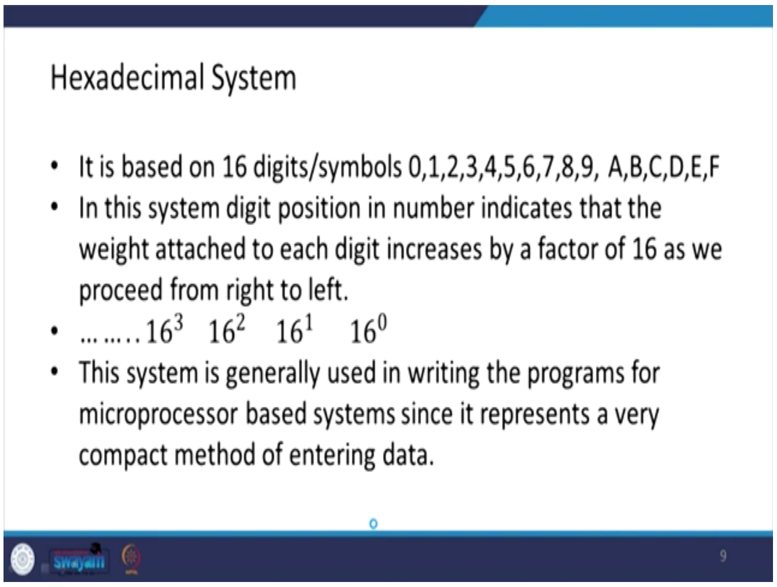
$= 8 + 7 = 15$

8

Next, is the octal system. This system is based on 8 digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 7 . These are the 8 digits, and when a number is represented by this system that digit position in the number indicates that the weight attached to each digit is the increases by a factor of 8 as you proceed from right to left. So, we have 8^0, then 8^1, 8^2, and 8^3. So, there is an increment of 8 that is by a factor of 8. So, in general, if this is my number so, this is what is meant by it $d_{n-1} \times 8^{n-1} + d_1 \times 8^1 + d_0 \times 8^0$.

So, number 15 in the decimal system is number 17 in the octal system. Let us see how it is 17 in the octal system. We have 1 over here into 8^1 and we have 7 over here into 8^0. So, this is 8 + 7. So, this is our 15 in the decimal system.

(Refer Slide Time: 09:58)



Hexadecimal System

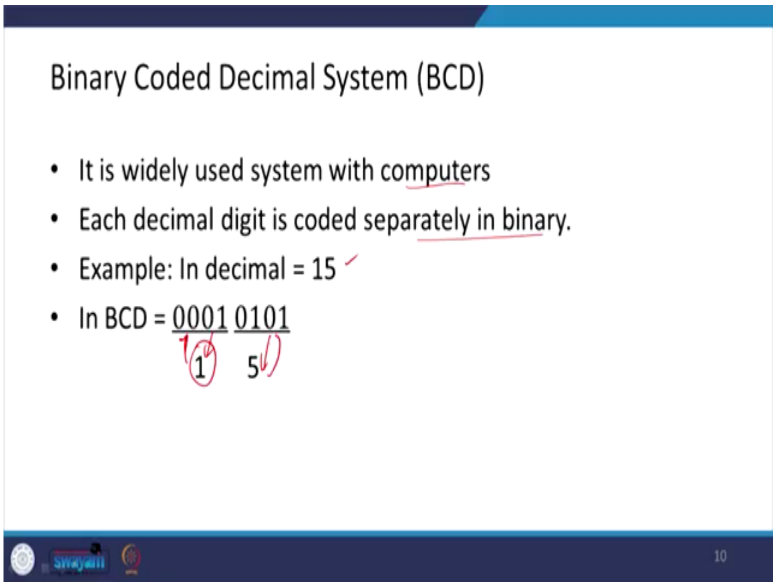
- It is based on 16 digits/symbols 0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F
- In this system digit position in number indicates that the weight attached to each digit increases by a factor of 16 as we proceed from right to left.
- $\ldots\ldots\ldots 16^3 \quad 16^2 \quad 16^1 \quad 16^0$
- This system is generally used in writing the programs for microprocessor based systems since it represents a very compact method of entering data.

9

Next, let us look at the hexadecimal system and this hexadecimal system as the name indicates is based on the 16 digits and symbols. So, these are 0 to 9 and then A to F. So, 10 + 6, 16. So, based on these 16 digits and symbols the hexadecimal system is formed.

Now, in this system digit position in the number indicates that the weight attached to each digit increases by a factor of 16 as we proceed from right to left. So, here you have 16^0, then 16^1, 16^2, 16^3, and so on. Now, this system is generally used in writing the programs for microprocessor-based systems since it represents a very compact method of entering data.

(Refer Slide Time: 11:11)

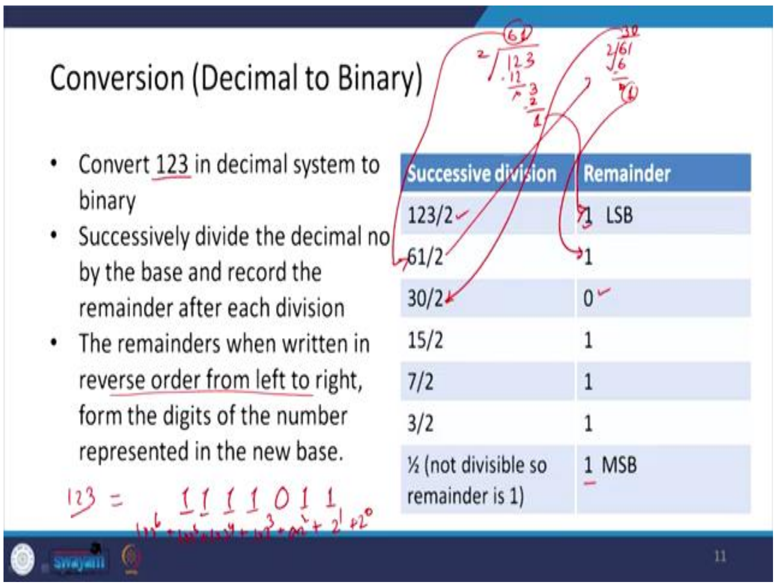


Binary Coded Decimal System (BCD)

- It is widely used system with computers
- Each decimal digit is coded separately in binary.
- Example: In decimal = 15
- In BCD = $\underbrace{0001}_{1}\ \underbrace{0101}_{5}$

Next, let us see the binary coded decimal system . The binary coded decimal in the binary coded decimal system is a widely used system with the computer. Each decimal digit is separated is coded separately in binary. So, for example, 15 in a binary coded decimal (BCD) will be 0 0 0 1 and 0 1 0 1 like this. So, we can see that this is 1 and this is 2^2 that is 4 +1. So, this is 15. Here the 1 is coded in the binary like this and 5 is coded in the binary like this.

(Refer Slide Time: 12:30)



Conversion (Decimal to Binary)

- Convert 123 in decimal system to binary
- Successively divide the decimal no by the base and record the remainder after each division
- The remainders when written in reverse order from left to right, form the digits of the number represented in the new base.

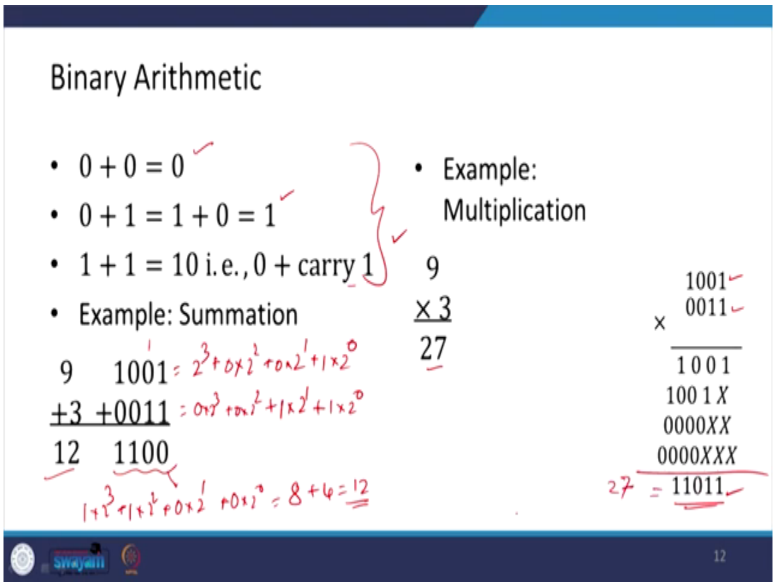
Successive division	Remainder
123/2	1 LSB
61/2	1
30/2	0
15/2	1
7/2	1
3/2	1
½ (not divisible so remainder is 1)	1 MSB

123 = 1 1 1 1 0 1 1

$1\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 2^1 + 2^0$

Then, the question arises how do we convert from decimal system to binary system ? So, it is a very simple. I can explain it by taking an example, suppose that 123 is in the decimal system and I want to convert it into the binary. So, what we do is we successively divide the decimal number by the base and record the remainder after each division. So, 123, I divide by 2 and I record the remainder. So, for example, 123 if I divide by 2 it will be 2 6s are 12. So, nothing is there then 3. So, 2 1s are 2. So, by here I subtracted so, I get 1. So, this 1 is the remainder and this 61 is going for the next successive division, and now, I can do the division of 62 with 2. So, again similarly I can do that. So, 2 3s are 6 this is 0 and so, I get a reminder 0 and so, I get a remainder 1 over here. So, this 1 comes over here and this 30 comes over here and, then again 30 by 2 is 0. So, this way I can keep on dividing, and the last I get 1 by 2, and since this is not divisible. So, the remainder remains as 1. So, this 123 I can write in a binary as this is my least significant bit. So, this will be 1 1 0 and 1111. So, this is there and so, this is how this can be represented over here. You can evaluate this using the formula which we have seen in the last few slides. So, this $2^0 + 2^1 + 2^2 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6$. So, this way we can 2 3 4 5 6. So, 0 1 2 3 4 5 6 and if you evaluate you will get 1 2 3. So, the remainders when written in the reverse order from left to right from the digits of the number represented in the new base.

(Refer Slide Time: 17:00)



Binary Arithmetic

- $0 + 0 = 0$ ✓
- $0 + 1 = 1 + 0 = 1$ ✓
- $1 + 1 = 10$ i.e., $0 +$ carry 1 ✓
- Example: Summation

$$\begin{array}{rr} 9 & 1001 \\ +3 & +0011 \\ \hline 12 & 1100 \end{array}$$

$1001 = 2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$

$0011 = 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$1\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0 = 8 + 4 = \underline{\underline{12}}$

- Example: Multiplication

$$\begin{array}{r} 9 \\ \times 3 \\ \hline 27 \end{array}$$

$$\begin{array}{rr} & 1001 \\ \times & 0011 \\ \hline & 1001 \\ & 1001X \\ & 0000XX \\ & 0000XXX \\ \hline 27 = & 11011 \end{array}$$

12

Next, let us look at binary arithmetic. So, in the binary arithmetic 0 +0 is 0, 0+ 1 is 1, 1+ 0 is 1, and 1 + 1 is 10 or that is 0 + carry 1. So, I can see it here. So, 9+ 3 in the decimal system, this is 12. And, 9 in binary is represented by 1001, which is,

$$2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0.$$

And this is what?

$$0 \times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0.$$

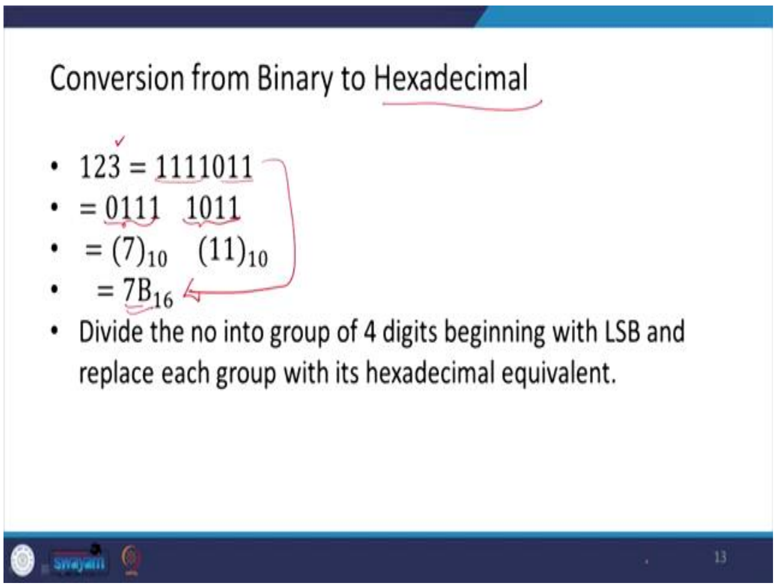
So, this is 3 and this is 8 +1, 9. So, this is how it is. So, this is how we can add it. So, 1 1 0 carry 1 so, 1 and 1 0 carries 1. So, that 1 is here and this 1 is here. So, this is 12 and this is what? This is,

$$0 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1\times 2^3.$$

So, this is 8 + 4 and so, this is 12. So, this is how the binary summation is carried out. Likewise, we can also have binary multiplication. So, for example, 9 × 3 is 27. So, this way 9 is your 1001 and 3 is 0 0 11.

And likewise, as we do the normal multiplication we can form the table and then do the summation using these three basic principles and we can get the required number. So, here is what we are going to get. So, you can check this must be equal to 27 in the decimal system.

(Refer Slide Time: 18:32)

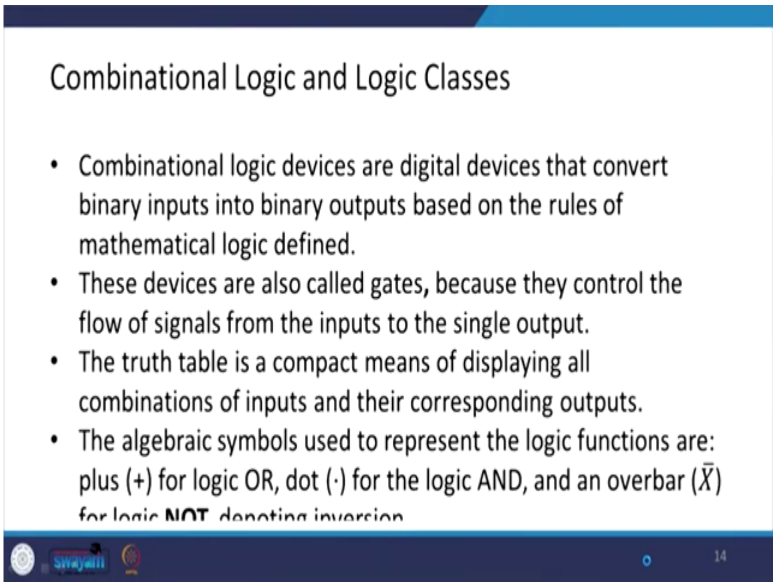


Conversion from Binary to Hexadecimal

- $123 = 1111011$
- $= 0111 \quad 1011$
- $= (7)_{10} \quad (11)_{10}$
- $= 7B_{16}$
- Divide the no into group of 4 digits beginning with LSB and replace each group with its hexadecimal equivalent.

Conversion from binary to hexadecimal: suppose, I want to convert from binary to hexadecimal. So, a binary number we have taken is 123. And for that 123 we have seen over here its representation here there are 111 1 0 1 1. For converting from binary to hexadecimal, what we do is we form the group of 4, 4 bits . So, 1 0 1 1 is formed together and only 11 1 are here. So, I put 0 from my side and then we write and then we write their decimal equivalent. So, for example, this is 7 in the decimal system and this is 11 in the decimal system. And so, the 7 in the decimal is written as 7, and 11 is written as B. So, this is a hexadecimal representation. So, this is how from binary we can convert it into that is hexadecimal. So, we divide the number into the group of 4 digits beginning with the least significant bit and replacing each group with a hexadecimal equivalent.

(Refer Slide Time: 20:47)



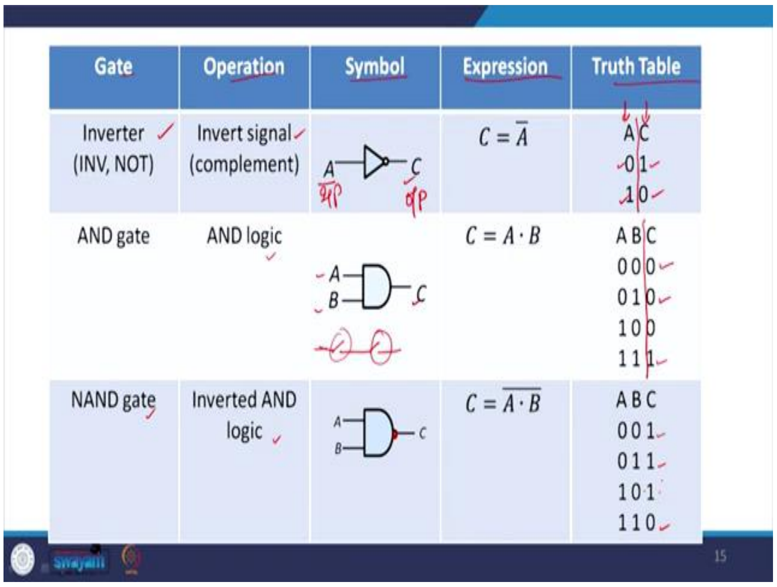
Combinational Logic and Logic Classes

- Combinational logic devices are digital devices that convert binary inputs into binary outputs based on the rules of mathematical logic defined.
- These devices are also called gates, because they control the flow of signals from the inputs to the single output.
- The truth table is a compact means of displaying all combinations of inputs and their corresponding outputs.
- The algebraic symbols used to represent the logic functions are: plus (+) for logic OR, dot (·) for the logic AND, and an overbar ($\bar{X}$) for logic **NOT** denoting inversion

14

Next, let us look at combinational logic and logic classes. Combinational logic devices are digital devices that converts binary input into binary output based on the rules of the mathematical logic which is defined. And, these devices are also called gates because they control the flow of signals from input to the signal output. The truth table is a compact mean of displaying all possible combinations of input and their corresponding outputs. The algebraic symbol used to represent the logic functions are plus for logic OR, dot for logic AND, and an over the bar for the NOT denoting inversion.

(Refer Slide Time: 21:21)



Gate	Operation	Symbol	Expression	Truth Table
Inverter (INV, NOT)	Invert signal (complement)	A C	$C = \overline{A}$	A C 0 1 1 0
AND gate	AND logic	A B C	$C = A \cdot B$	A B C 0 0 0 0 1 0 1 0 0 1 1 1
NAND gate	Inverted AND logic	A B C	$C = \overline{A \cdot B}$	A B C 0 0 1 0 1 1 1 0 1 1 1 0

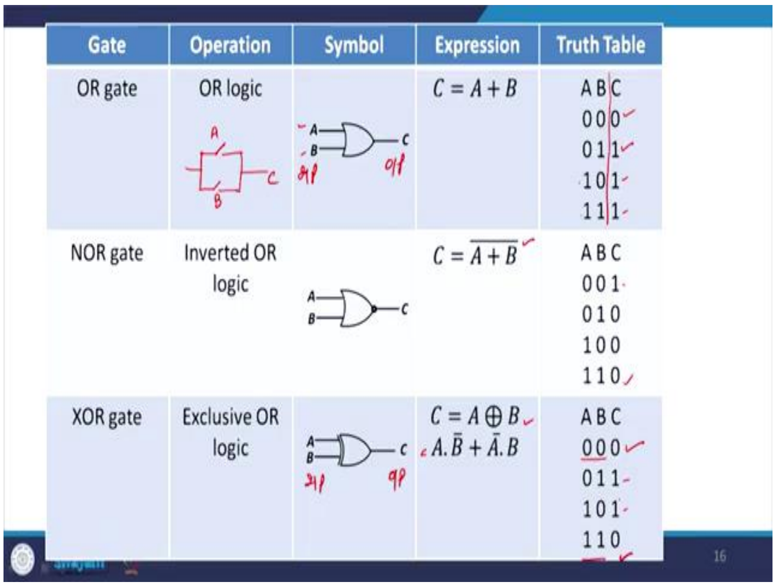
So, here are a table explaining the type of gate and the operation which is being carried out how it is symbolically represented and what is the expression for that and what is the truth table for that. So, let us begin with that.

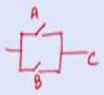
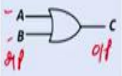
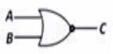
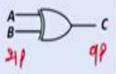
First is the inverter or NOT gate what we call it. So, it inverts the signal. This is how symbolically it is represented that is an amplified signal with a circle at the vertex over there and the expression for this is A is the input here. So, this is your input and C is your output. So, here this C is equal to A invert. Now, the truth table for this is like this. So, what is my A, and what is my C. So, here is the input, and here is the output. So, if my A is 0, then it is inversion will be 1 and if it is 1, then it is inversion is going to be 0. So, this is going to be the truth table for that.

Next is the AND gate and operation is the AND logic. So, suppose this is the symbolical representation of that. So, A and B are the input and C is the output and its expression is C is equal to A into B. And, if I look at the truth table so, again here A and B are the input and C is the output. So, here you see that if A is 0 and B is 0 you have C is 0; if A is 0 and B is 1, then your C is 0 and if A is 1 and B is 0, then also C is 0 and if A is 1 and B is 1, then C is 1. So, this can also be explained with the help of there are two switches like this. So, you are going to have output only when both the switches are ON. So, here also you are getting to the going to get the output only when both A and B are ON. When both A and B are 1, then you are going to get the output 1 otherwise it is going to be 0 only. So, this is the AND gate. Next is the NAND gate and it is nothing, but the inverted AND gate. And, here just this inversion is represented here by putting a circle here.

And, in the expression, we put a bar at the top of it to represent the inversion. And, here whatever has been the AND output that has to be inverted. So, the AND output for this has been 0, so, inverted 1. Here it has been 0 so, inverted 1; here it has been 0, so, inverted 1 and here it has been 1 so, inverted 0. So, this is how the NAND gate is defined.

(Refer Slide Time: 24:37)



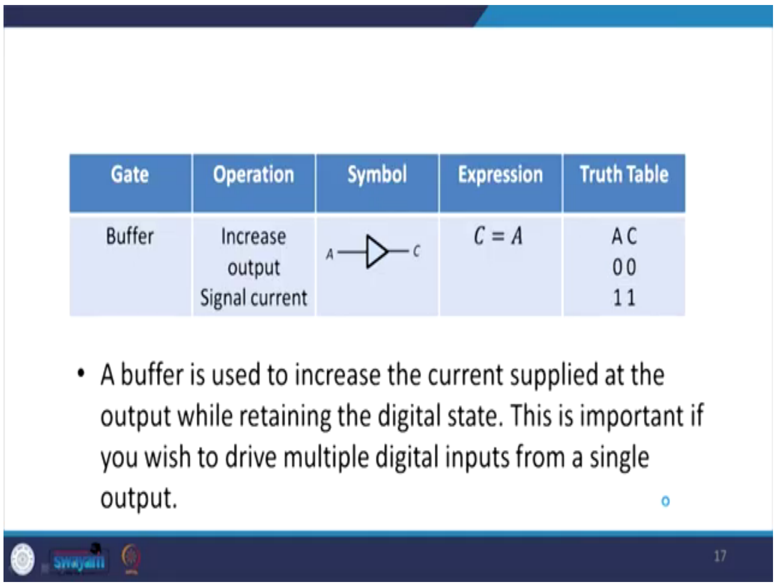
Gate	Operation	Symbol	Expression	Truth Table
OR gate	OR logic 		$C = A + B$	A B C 0 0 0 0 1 1 1 0 1 1 1 1
NOR gate	Inverted OR logic		$C = \overline{A + B}$	A B C 0 0 1 0 1 0 1 0 0 1 1 0
XOR gate	Exclusive OR logic		$C = A \oplus B$ $A.\bar{B} + \bar{A}.B$	A B C 0 0 0 0 1 1 1 0 1 1 1 0

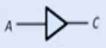
Then, let us look at let us look at the OR gate, the logic is OR logic. This is the symbolic representation of it. We have the two input here and here is the output. Expression for OR is given by C is equal to A plus B. Now, here let us look at this input side and output side. So, if A and B both are 0, then naturally your C is 0. If A is 0, B is 1 then output is 1; if A is 1, B is 0, then also the output is 1 and if both are 1 then also the output is 1. So, this is; this is what is meant by it. So, it can be symbolically, or if you want to understand it is just like this that you have a switch here and you have a switch here both are in parallel. And, this is A and this is B and you have the output over here C. So, you see that if both are open then you have 0; if either 1 is either 1 is closed then you have the output and if you both are closed then you have the output. So, this is how it is.

Then the NOR gate is the inverted OR gate. So, here you have the inversion and the output here will be the inversion of this one. So, 0 inversion is 1 over here and for the 1 inversion 0 is over here.

Then we have the XOR gate or what we call the exclusive-OR gate. So, here A and B are the input and C is the output over here. This is how it is represented in the expression. This is what is mean by it that is A and B inverse plus A inverse and B and corresponding your table is going to be this one. So, here if either both are 0 both inputs are 0, then output is 0 and if both inputs are 1, then also the output is 0 and if either input is 1 then you have the sorry, if either of the input is 0, then you have the 1 over, here.

(Refer Slide Time: 27:50)



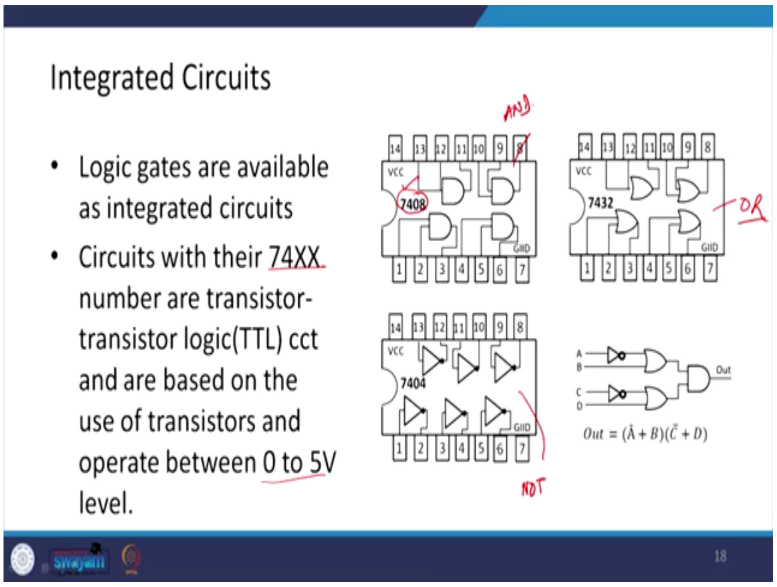
Gate	Operation	Symbol	Expression	Truth Table
Buffer	Increase output Signal current		$C = A$	A C 0 0 1 1

- A buffer is used to increase the current supplied at the output while retaining the digital state. This is important if you wish to drive multiple digital inputs from a single output.

17

Then, we have a buffer the operation is increased output signal current and here the expression is C is equal to A. So, whatever A is there the same C is there and a buffer is used to increase the current supplied at the output when while retaining the digital state. This is important if you wish to drive multiple digital inputs from a single output.

(Refer Slide Time: 28:41)



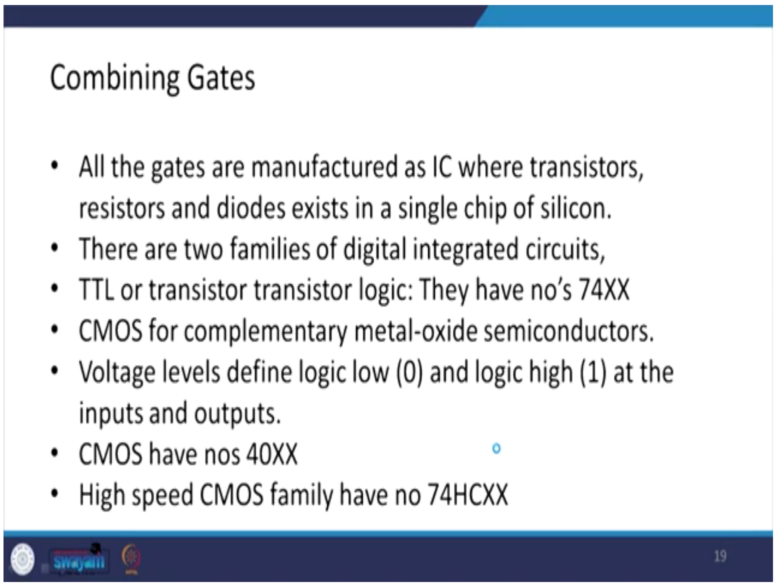
Integrated Circuits

- Logic gates are available as integrated circuits
- Circuits with their 74XX number are transistor-transistor logic(TTL) cct and are based on the use of transistors and operate between 0 to 5V level.

AND
7408
VCC
GND
7432
OR
7404
NOT
Out = $(\bar{A}+B)(\bar{C}+D)$

Now, the coming to the integrated circuits, these logic gates, which I talked to you just now these gates are available at as integrated circuits. And circuits with their 74XX number are the TTL circuit or what we call transistor-transistor logic circuits and based on the use of transistor and operate between 0 to 5-volt level. For example, you can see that the IC 7408 is this is based on the AND that is if you want to have the AND gates, then you can have the 7408, 7408 IC. And, if you want to use the OR, so for OR gate purpose a 7432; this is for AND gate purpose and this is for inverter or NOT gate purpose. So, these types of ICs are available.

(Refer Slide Time: 29:37)

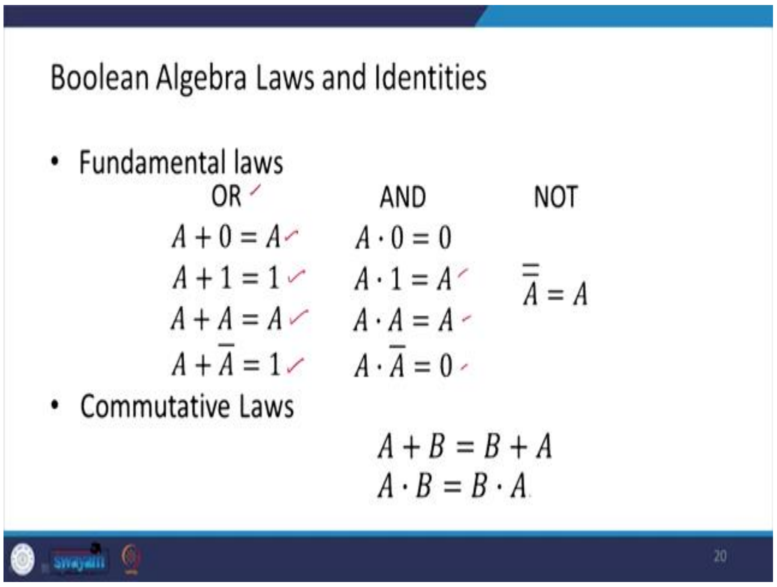


Combining Gates

- All the gates are manufactured as IC where transistors, resistors and diodes exists in a single chip of silicon.
- There are two families of digital integrated circuits,
- TTL or transistor transistor logic: They have no's 74XX
- CMOS for complementary metal-oxide semiconductors.
- Voltage levels define logic low (0) and logic high (1) at the inputs and outputs.
- CMOS have nos 40XX
- High speed CMOS family have no 74HCXX

Then combining of gates all the gates as I just said are manufactured as ICs where transistor, resistor and diode exist in a single chip of silicon. Now, there are two families of digital integrated circuits one is the TTL or the transistor-transistor logic and they have a number 74 dash dash and the other is CMOS for Complimentary Metal Oxide Semiconductors. And, voltage level device logic low and logic high at the input and output and CMOS have numbers 40 and so on, and high-speed CMOS family have number 74H is there and then CXX.

(Refer Slide Time: 30:50)



Boolean Algebra Laws and Identities

- Fundamental laws

OR	AND	NOT
$A + 0 = A$	$A \cdot 0 = 0$	
$A + 1 = 1$	$A \cdot 1 = A$	$\overline{\overline{A}} = A$
$A + A = A$	$A \cdot A = A$	
$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$	

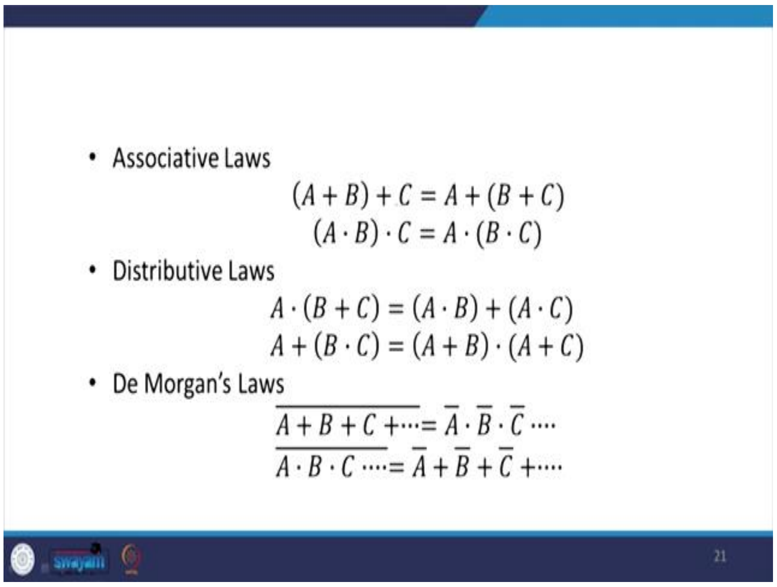
- Commutative Laws

$$A + B = B + A$$
$$A \cdot B = B \cdot A$$

20

Now, let us look at the Boolean algebra laws and identities. So, there are some fundamental laws for OR. So, A + 0 is A, A+ 1 is 1, A+ A is A and A + A invert is A. And, for the AND A × 0 is 0, A×1 is A, A×A is A, A × A invert is 0. And, for the NOT A is same as A invert, invert. And there are commutative laws that are A + B is equal to B + A, A × B is equal to B × A.

(Refer Slide Time: 31:44)



- Associative Laws

$$(A + B) + C = A + (B + C)$$
$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

- Distributive Laws

$$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$$
$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

- De Morgan's Laws

$$\overline{A + B + C + \cdots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdots$$
$$\overline{A \cdot B \cdot C \cdots} = \overline{A} + \overline{B} + \overline{C} + \cdots$$

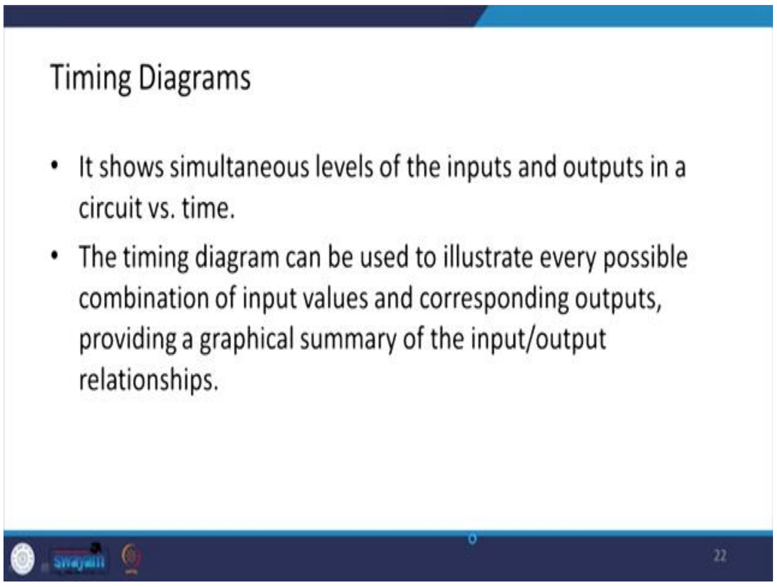
21

Associative laws we have (A + B) + C is equal to A +(B+ C). So, either you can associate A + B or you can associate B and C and similarly, (A × B) × C is equal to A × (B × C), then we have the distributive laws A × (B + C) is equal to A × B + A × C. Similarly, for A + B × C is A + B × A + C.

Then, there are De Morgan's laws . So, these laws are there to convert the OR operations into AND operation or AND operation into OR operation, .

So, here you can see that A + B + C all invert is equal to invert of individual ABC with the and that is with their multiplication or with the and form. And, A and B and C invert is equal to invert of A plus invert of B plus invert of C. So, here what we can see that here the OR has been converted into AND, and below the AND has been converted into OR.

(Refer Slide Time: 33:19)

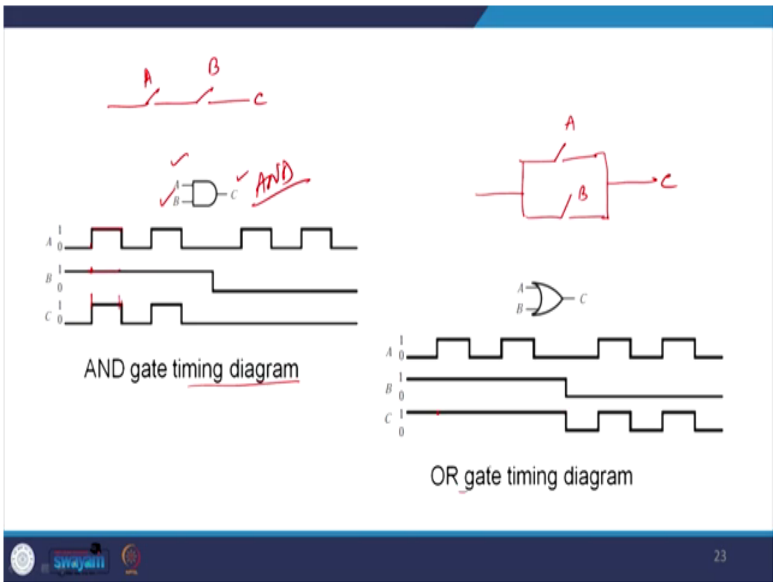


Timing Diagrams

- It shows simultaneous levels of the inputs and outputs in a circuit vs. time.
- The timing diagram can be used to illustrate every possible combination of input values and corresponding outputs, providing a graphical summary of the input/output relationships.

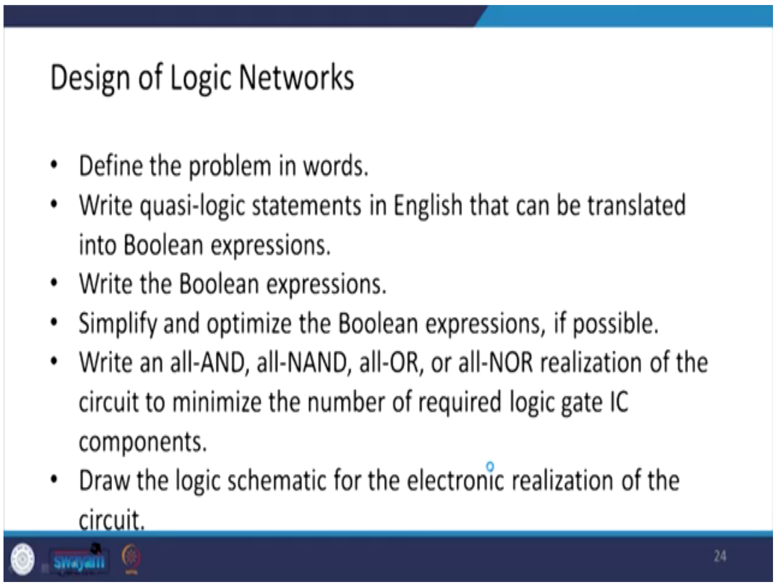
Then many times we use to draw the timing diagram. It indicates that with respect to time what are the signal. So, it shows the simultaneous level of the input and output in a circuit versus time. And, the timing diagram can be used to illustrate every possible combination of input values and corresponding output, providing a graphical summary of the input-output relationship, and many times it does help us in understanding.

(Refer Slide Time: 33:59)



So, for example, AND gate timing diagram, so, here A and B are the input and C is the output, and as you see that this is the AND. So, here you see in the case of AND for 1 to have you need to have both the signal as 1. So, AND is the two switches in series that way you can understand. So, if this is A and B and so, you have C. So, if both are 1 then only you will have C is equal to 1. So, you can see here, here A is 0 and here B is 1. So, you are going to get 0 over here, but at this place A is 1 up to here and B is already 1. So, you are going to have the 1 over here for the C. Similarly, with the OR gate for the OR one, it is as I have been as I explained to you so, this if A, B, and your C are over here. So, here if your A is 0 and B is 1, then you are going to have C is equal to 1. So, you can see over here that the C value is 1. Likewise, you can explain the OR gate using the timing diagram.

(Refer Slide Time: 35:42)



Design of Logic Networks

- Define the problem in words.
- Write quasi-logic statements in English that can be translated into Boolean expressions.
- Write the Boolean expressions.
- Simplify and optimize the Boolean expressions, if possible.
- Write an all-AND, all-NAND, all-OR, or all-NOR realization of the circuit to minimize the number of required logic gate IC components.
- Draw the logic schematic for the electronic realization of the circuit.

24

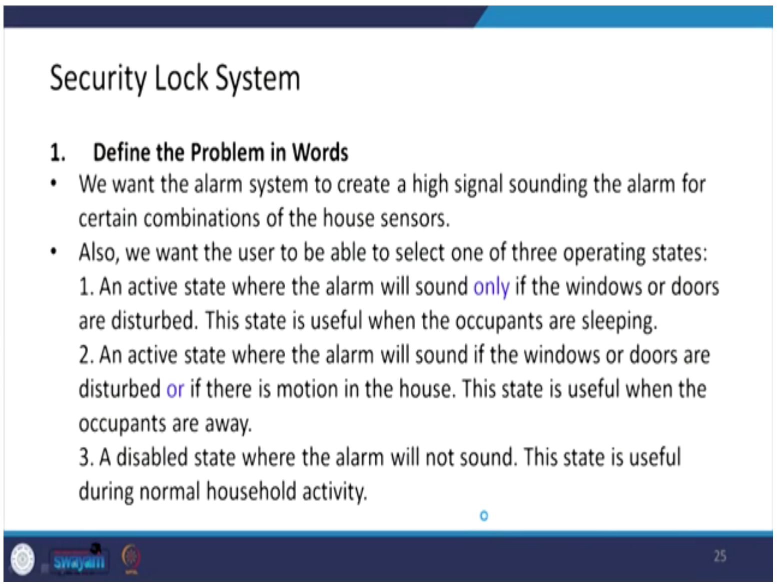
Now, after all these basic concepts let us use these basic concepts in designing a logic network. So, for designing of logic network the example which I am going to take I have taken from Alciatore and Histand book on mechatronics. So, those who want to read further you may do refer to that. So, for the design of a logic network, there are basic steps we can formulate. First, you can define the problem in word and then you can write a quasi-logic statement in English that can be translated into a Boolean expression. And, then from that quasi aesthetic statement, you write the Boolean expression, and then you can simplify and optimize the Boolean expression.

Why do we need that? Because ultimately we are going to realize that with the help of some gates and gates how we are going to get? Gates we are going to get in the form of ICs. So, if you simplify that then you are going you may require a few numbers of ICs or a lesser number of ICs to realize that particular Boolean expression.

Then we write an all AND, all NAND, all OR. Why do we do write this all type that is a similar type of gates expression in terms of the similar type? That is because we need to use only a similar type of ICs, one IC rather based on the requirement similar type of ICs. So, write an all AND, all NAND, all OR, or all NOR realization of a circuit to minimize the number of required logic gates ICs components.

And, then finally, draw the logic schematic for the electronic realization of the circuit. So, let us take the security lock system. So, first of all, we define the problem involved. So, suppose we want the alarm system for a house. And we want that the system should create a high signal sounding the alarm for the certain condition of the house sensor.

(Refer Slide Time: 38:34)



Security Lock System

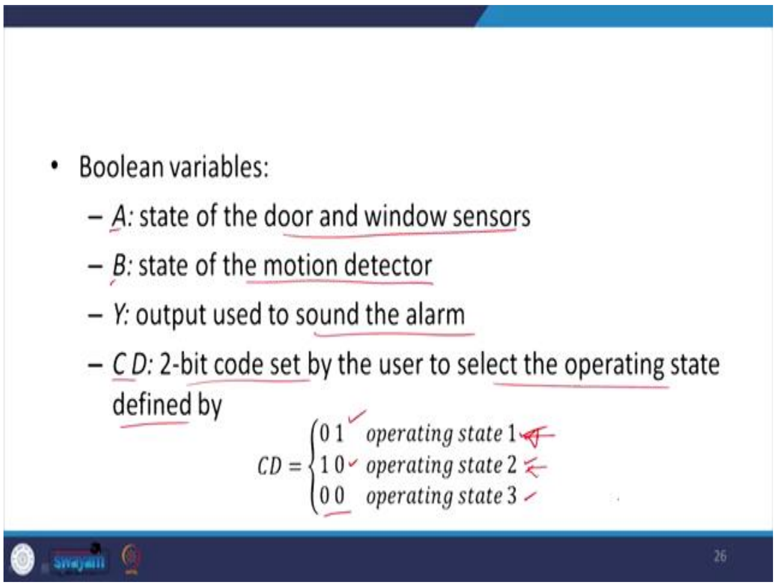
1. **Define the Problem in Words**

- We want the alarm system to create a high signal sounding the alarm for certain combinations of the house sensors.
- Also, we want the user to be able to select one of three operating states:
  1. An active state where the alarm will sound only if the windows or doors are disturbed. This state is useful when the occupants are sleeping.
  2. An active state where the alarm will sound if the windows or doors are disturbed or if there is motion in the house. This state is useful when the occupants are away.
  3. A disabled state where the alarm will not sound. This state is useful during normal household activity.

25

Now, also why we want the user to be able to select one type of the operating state. Now, there could be various types of operating states also and we want a user to set, which type of operating state the alarm system he or she wants to operate. So, let us look at these three classifications. Suppose, the user wants an active state when the alarm will sound only if the windows or the doors are disturbed. Now, this state is useful with the occupant is sleeping. So, if the occupant is sleeping and there is some disturbance to the windows or door, one should get the alarm. The other state could be an active state where the alarm will sound if the windows or dates are disturbed or if there is a motion inside of the house. So, this state is useful when the occupants are away. So, if occupants are away so, if the thieves are coming to the house by breaking the doors or windows or if there is a motion also. Because somebody breaking doors and window and then if they are moving inside and nobody is there in the house so, in that condition also there should be an alarm. And, the third state could be a disabled state where the alarm will not sound and this state is useful during normal household activity. So, suppose this is the vague requirement, which I want and now I want to design a system a security lock system.

(Refer Slide Time: 40:45)



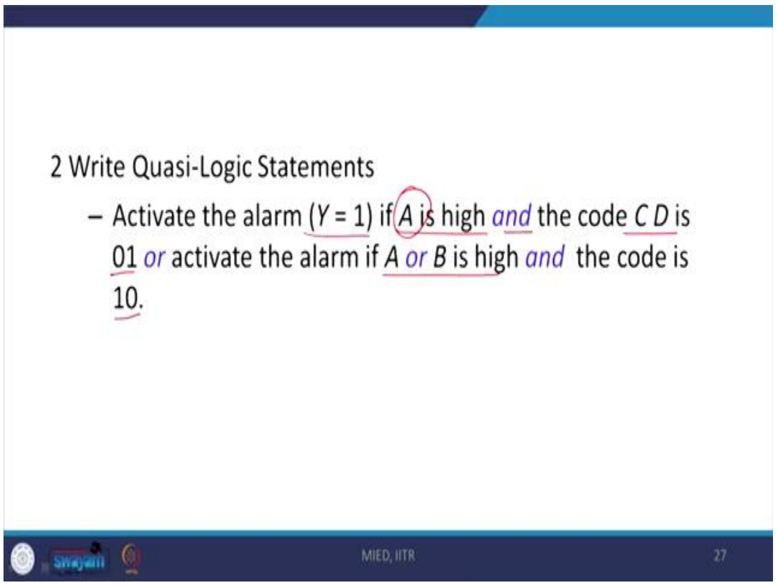
- Boolean variables:
  - A: state of the door and window sensors
  - B: state of the motion detector
  - Y: output used to sound the alarm
  - C D: 2-bit code set by the user to select the operating state defined by

$$CD = \begin{cases} 0\,1 & operating\ state\ 1 \\ 1\,0 & operating\ state\ 2 \\ 0\,0 & operating\ state\ 3 \end{cases}$$

26

So, I define the Boolean variable A is the state of the door and window sensor; B is the state of the motion detector which can detect the motion of the thief inside of the house, and Y is the output used to sound alarmed. So, Y is my output. Now, I said that my user should be able to pick any of the three states. So, for that, I generate a 2-bit code set by the user to select the operating state. Suppose C D is 0 1 if a user sets 0 1 then operating state is 1, if a user sets 1 0 then operating state is 2 and if a user sets 0 0 it means that the operating state 3. That is operating state is 1 is that the door and window sensor-activated; operating state 2 is that when apart from door and windows sensor-activated there is motion detector also activated and third is for the normal case.

(Refer Slide Time: 41:53)



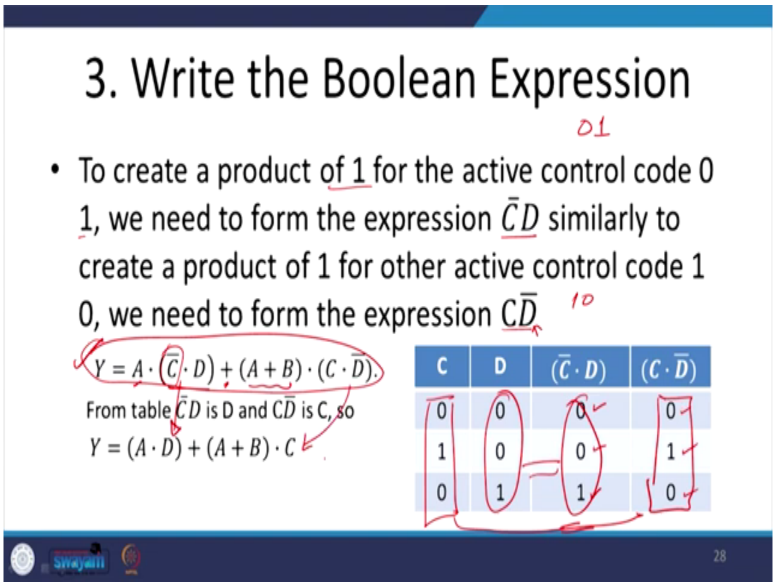
2 Write Quasi-Logic Statements

- Activate the alarm ($Y = 1$) if A is high *and* the code $C\ D$ is 01 *or* activate the alarm if A *or* B is high *and* the code is 10.

MIED, IITR
27

Now, so, what is our quasi-logic statement? So, the quasi-logic statement could be like this – Activate the alarm that is (Y = 1) if A is high and code C D is 01. 01 we wrote for the first state. So, an alarm should be there for fulfilling both these requirements that are A is high and C D is 01 or activate the alarm if A or B is high and code is 10. So, this is my quasi-logic statement. So, what does this means? This means that output should be there if your A is high and the user had set code that is he or she is sleeping or the alarm should be activated that is if either doors and windows are disturbed, or the motion of the person inside the house is detected. And the user has said that he is not in the house that is this state 10. So, we can write the Boolean expression. To create a product of 1 for the why product 1? Because we want the alarm for the active control code 01, we need to form the expression C invert D, because we have first state 01. So, if I want 1 product of 1 naturally this 01 I have to invert this 0 I have to invert, then only I can have 1.

(Refer Slide Time: 44:05)



3. Write the Boolean Expression

01

- To create a product of 1 for the active control code 0 1, we need to form the expression $\bar{C}D$ similarly to create a product of 1 for other active control code 1 0, we need to form the expression $C\bar{D}$ 10

$Y = A \cdot (\bar{C} \cdot D) + (A + B) \cdot (C \cdot \bar{D})$.

From table $\bar{C}D$ is D and $C\bar{D}$ is C, so

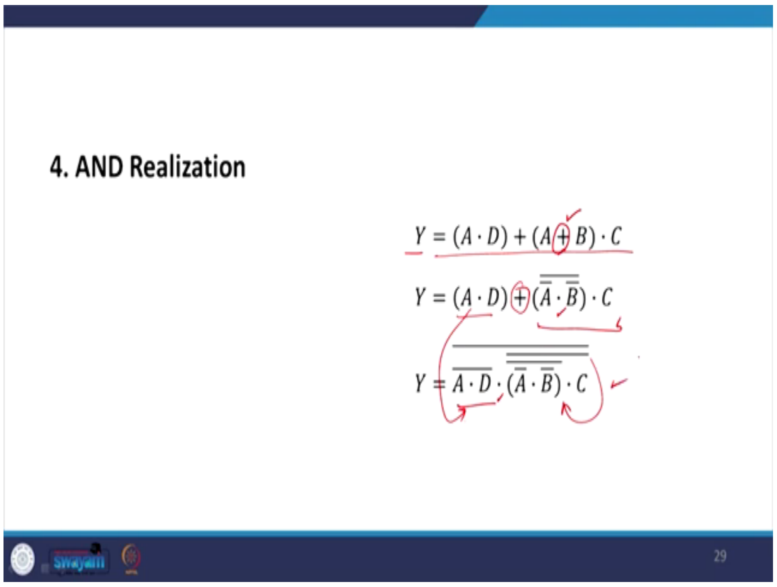
$Y = (A \cdot D) + (A + B) \cdot C$

C	D	$(\bar{C} \cdot D)$	$(C \cdot \bar{D})$
0	0	0	0
1	0	0	1
0	1	1	0

28

Similarly to create a product of 1 for the active control code 10 , we need to form the expression C D invert. So, here I need to invert D so that this 1 0 could be made 11. So, my expression output Y is this what I have to create a AND operation with A AND C invert D and then I have also AND operation with A + B AND C. D invert and then I have the OR operation. So, this single expression is the representation for this quasi-logic statement. And, as I said we have we want an alarm to be ON. So, that is we want Y to be 1 and here our code was 01 that is why I inverted it over here to create 11, and then I put the AND operation with A and, here we have we want A+ B over here that is in either case here that is either A or B is high so, I want the or over here. So, A +B here and then I have created here I want an AND operation with C and D invert over here. So, from the table, I can see here that if my C is 0 1 0 and D is 0 0 1. So, what is my C invert D? C invert D is 0, if C is 1 and D is 0 then C invert D is again 0 and if C is 0 and D is 1, then C invert D is 1, and what is my C D inverse? So, if C is 0, D is 0 then this is 0 if C is 1 and the D is 0 then D invert is going to be 1, and 1 into 1 is going to be 1 and in this case, this is going to be 0. Now, if I look at this table, then you see that C invert D is 0 0 1 and this is the same as D. So, what does this mean that? I can replace this C invert D with simply D and here you see C D invert is 0 1 0 and here C is 0 1 0 this one. So, this is the same as this one and this is the same as this one. So, I can replace C D invert by Cover here. So, this is my simplified expression for this.

(Refer Slide Time: 47:29)



4. AND Realization

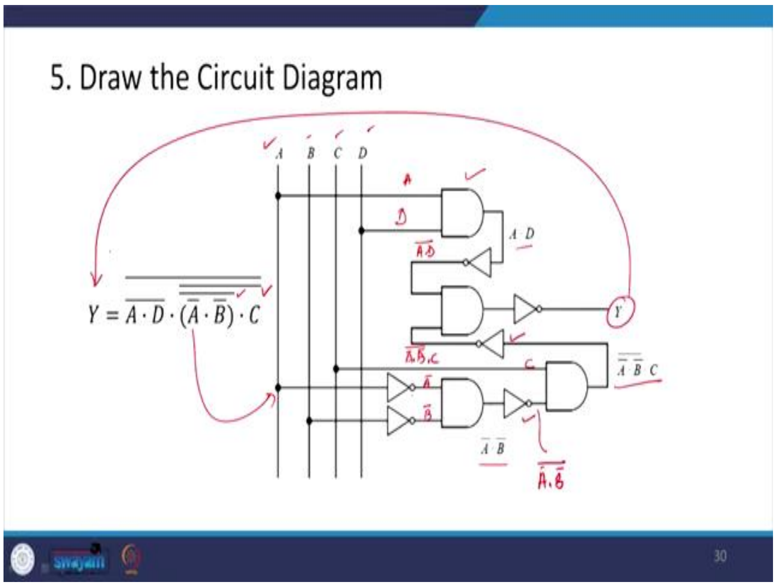
$$Y = (A \cdot D) + (A + B) \cdot C$$

$$Y = (A \cdot D) + (\overline{\overline{A} \cdot \overline{B}}) \cdot C$$

$$Y = \overline{\overline{A \cdot D} \cdot \overline{(\overline{\overline{A} \cdot \overline{B}}) \cdot C}}$$

So, this is my simplified expression. Now, I want to convert it for making the AND realization. So, that I can do so, first take this OR operation, I convert it into a AND one because I want the realization of AND, and the next step I convert this OR operation into the AND operation. So, this is the invert of individual that is A AND B and its whole invert and similarly, here it is going to be the invert of this and invert of this and then the whole invert. So, invert of this is this one, and invert of this is this one and then you have the whole invert. So, this is the AND realization.

(Refer Slide Time: 48:45)



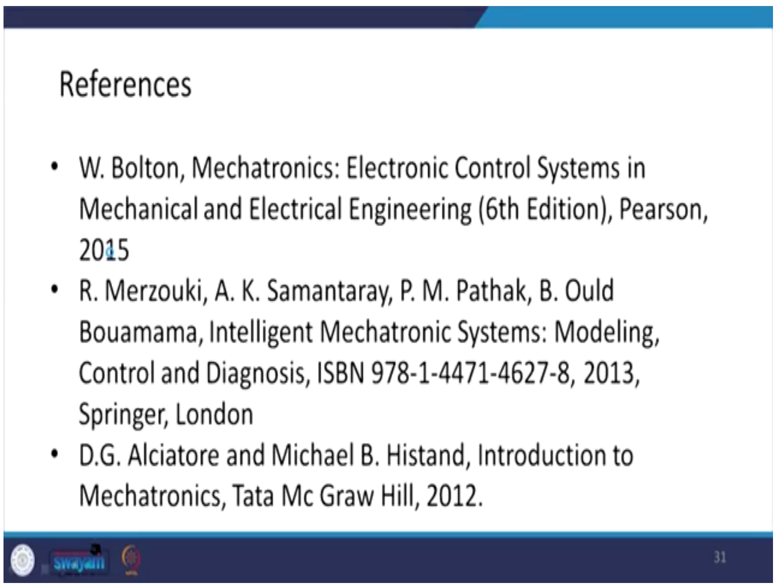
5. Draw the Circuit Diagram

$$Y = \overline{\overline{A \cdot D} \cdot \overline{(\overline{\overline{A} \cdot \overline{B}}) \cdot C}}$$

A
B
C
D
A
D
A · D
$\overline{AD}$
Y
$\overline{A}$
$\overline{B}$
$\overline{A} \cdot \overline{B}$
$\overline{\overline{A}.\overline{B}}$
C
$\overline{\overline{A}.\overline{B}}.C$
$\overline{A} \cdot \overline{B} \cdot C$

And, in the last we can draw a circuit diagram for this AND realization. So, suppose I have got the four inputs A, B, C, D over here. So, you see this is how it is done. So, we had the A invert so, I can take A from here invert it over here then here is. So, this is what A invert and here I have got B, I put it an invert. So, I have a B invert. In the AND operation I get A invert dot B invert over here. Now, I have to take this whole invert over here. So, I put one NOT gate here. So, I invert this. So, here is A invert dot B invert whole invert over here, and then I have C directly from here and, I take a product of that. So, AND gate, I use for that. So, this is my output and after this what I do is that I take another invert here. So, for this operation and then I feed it to the AND gate to take it with A and D. So, here I have AND gate. So, I take A from here; A is here, D is here, A and D is here I take invert of that. So, here I get A and D invert and this A and D invert and what is here is A invert and B invert and C and A and B invert. So, this is the input. So, here I get the output. So, I get the multiplication or AND operation of that and then the whole invert. So, this is my Y, which is equal to this one.

(Refer Slide Time: 51:10)



References

- W. Bolton, Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering (6th Edition), Pearson, 2015
- R. Merzouki, A. K. Samantaray, P. M. Pathak, B. Ould Bouamama, Intelligent Mechatronic Systems: Modeling, Control and Diagnosis, ISBN 978-1-4471-4627-8, 2013, Springer, London
- D.G. Alciatore and Michael B. Histand, Introduction to Mechatronics, Tata Mc Graw Hill, 2012.

31

So, these are the references for you can look at Alciatore and Histand, as I told at the beginning for this. You can refer other two books also are mentioned over here.

Thank you.