

Chapter 2: Computer Hardware

In this chapter:

Basic Logic Gates.....	
Derived Gates.....	
Universal Gates.....	
How NAND gate is a Universal gate?.....	
Let us try some other gates using NAND gate.....	
How NOR gate is a Universal gate?.....	
Let us try some other gates using NAND gate.....	
Applications of logic gates	
Adder	
Half Adder.....	
Full Adder.....	
Alternate circuit diagram for full adder.....	
Subtractor.....	
Half subtractor.....	
Full subtractor.....	
Encoder.....	
Decimal to Binary (10 to 4) Encoder.....	
Octal to Binary (8 to 3) Encoder.....	
Hexadecimal to Binary (16 to 4) Encoder.....	
Decoder	
Binary to Decimal (4 to 10) Decoder	
Binary to Octal (3 to 8) Decoder.....	
Binary to Hexadecimal (4 to 16) Decoder.....	
Multiplexer	
Demultiplexer.....	
Practice questions	

Logic Gates (Introduction)

A gate is simply an electronic circuit which operates on one or more signals to produce an output signal. Gates are two-state digital circuits with input or output signals are at either 0 (low voltage) or 1 (high voltage) Gates are called logic circuits because they can be analyzed with Boolean algebra.

Different categories of logic gates

Logic gates can be categorized into three categories, they are as follows-

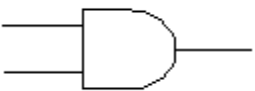

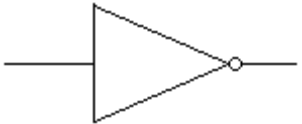
1. Basic Gates
2. Derived Gates
3. Universal Gates

1. Basic Gates

These are the elementary gates at the lowest level. There are 3 basic gates, they are – AND gate, OR gate and NOT gate.

- **AND gate** – It has two or more input lines and only one output line. When all the input lines are at 1, then only the output shows 1 otherwise 0.
- **OR gate** – It has two or more input lines and only one output line. If any one of the input line, or any combination of input line or all of them are at 1, then the output shows 1 and if all the input lines is at 0 then only the output is 0.
- **NOT gate** – It has only one input line and one output line. The output line always shows the opposite of the input line i.e. when input is at 1 then the output is 0 and when input is at 0 then the output is 1.

1.1 Basic logic gates with their circuit diagram and truth table

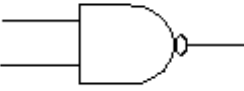



Name of the gate	Graphical symbol/Logic diagram	Algebraic Function	Truth Table															
AND		$X.Y$	<table border="1"><thead><tr><th>X</th><th>Y</th><th>X.Y</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></tbody></table>	X	Y	X.Y	0	0	0	0	1	0	1	0	0	1	1	1
X	Y	X.Y																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$X+Y$	<table border="1"><thead><tr><th>X</th><th>Y</th><th>X+Y</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></tbody></table>	X	Y	X+Y	0	0	0	0	1	1	1	0	1	1	1	1
X	Y	X+Y																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		\bar{X} or X'	<table border="1"><thead><tr><th>X</th><th>X'</th></tr></thead><tbody><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></tbody></table>	X	X'	0	1	1	0									
X	X'																	
0	1																	
1	0																	

2. Derived Gates

These gates are created as alternative gates to some combinations of two or more basic gates. There are 4 derived gates, they are – NAND gate, NOR gate, XOR gate and XNOR gate.

- **NAND gate** – This gate is an alternate to the combination of (AND + NOT) gate. It has two or more input lines and only one output line. **When all the input lines are at 1, then only the output shows 0 otherwise 1** i.e. in words this gate actually work opposite to AND gate. (Inverter of AND gate)
- **NOR gate** – This gate is an alternate to the combination of (OR + NOT) gate. It has two or more input lines and only one output line. **When all the input lines are at 0, then only the output shows 1 otherwise 0** i.e. in words this gate actually work opposite to OR gate. (Inverter of OR gate)
- **XOR gate** – This gate is also known as Exclusive OR gate. It has two or more input lines and only one output line. **When count of the input lines at 1 is odd, then only the output shows 1 otherwise 0.**
- **XNOR gate** – This gate is also known as NOT of Exclusive OR gate. It has two or more input lines and only one output line. **When count of the input lines at 1 is odd, then only the output shows 0 otherwise 1.**

2.1 Derived logic gates with their circuit diagram and truth table

Name of the gate	Graphical symbol/Logic diagram	Algebraic Function	Truth Table																				
NAND		$\overline{X \cdot Y}$	<table border="1"> <thead> <tr> <th>X</th> <th>Y</th> <th>$X \cdot Y$</th> <th>$\overline{X \cdot Y}$</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	X	Y	$X \cdot Y$	$\overline{X \cdot Y}$	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1	0
X	Y	$X \cdot Y$	$\overline{X \cdot Y}$																				
0	0	0	1																				
0	1	0	1																				
1	0	0	1																				
1	1	1	0																				
NOR		$\overline{X + Y}$	<table border="1"> <thead> <tr> <th>X</th> <th>Y</th> <th>$X + Y$</th> <th>$\overline{X + Y}$</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	X	Y	$X + Y$	$\overline{X + Y}$	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	0
X	Y	$X + Y$	$\overline{X + Y}$																				
0	0	0	1																				
0	1	1	0																				
1	0	1	0																				
1	1	1	0																				
XOR		$X \oplus Y = X \cdot Y' + X' \cdot Y$	<table border="1"> <thead> <tr> <th>X</th> <th>Y</th> <th>No. of 1</th> <th>$X \oplus Y$</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0(even)</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1 (odd)</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1 (ood)</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>2(even)</td> <td>0</td> </tr> </tbody> </table>	X	Y	No. of 1	$X \oplus Y$	0	0	0(even)	0	0	1	1 (odd)	1	1	0	1 (ood)	1	1	1	2(even)	0
X	Y	No. of 1	$X \oplus Y$																				
0	0	0(even)	0																				
0	1	1 (odd)	1																				
1	0	1 (ood)	1																				
1	1	2(even)	0																				
XNOR		$X \odot Y = X \cdot Y + X' \cdot Y'$	<table border="1"> <thead> <tr> <th>X</th> <th>Y</th> <th>No. of 1</th> <th>$X \odot Y$</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>2</td> <td>1</td> </tr> </tbody> </table>	X	Y	No. of 1	$X \odot Y$	0	0	0	1	0	1	1	0	1	0	1	0	1	1	2	1
X	Y	No. of 1	$X \odot Y$																				
0	0	0	1																				
0	1	1	0																				
1	0	1	0																				
1	1	2	1																				



3. Universal Gates

These are the two derived gates – **NAND** and **NOR**. These are called universal gates because, by using these gates we can create any of the basic gates – AND, OR and NOT and thus any form of logic circuit can be made with the help of these two gates only.


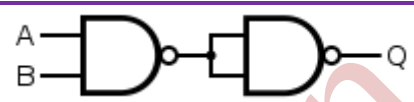
3.1 How NAND gate is a Universal gate?

A NAND gate is a universal gate, meaning that any other gate can be represented as a combination of NAND gates.


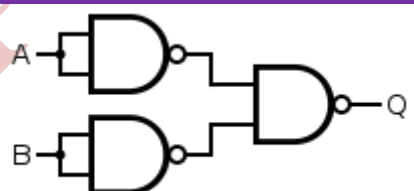
NOT A NOT gate is made by joining the inputs of a NAND gate. Since a NAND gate is equivalent to an AND gate followed by a NOT gate, joining the inputs of a NAND gate leaves the NOT part.

Desired Gate	NAND Construction	Proof								
		$Q = (A \cdot A)' = \bar{A} + \bar{A}$ $= \bar{A} \text{ (by Idempotent Law)}$								
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Output Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>		Truth Table		Input A	Output Q	0	1	1	0	
Truth Table										
Input A	Output Q									
0	1									
1	0									

AND An AND gate is made by following a NAND gate with a NOT gate as shown below. This gives a NOT (NAND), i.e. AND.


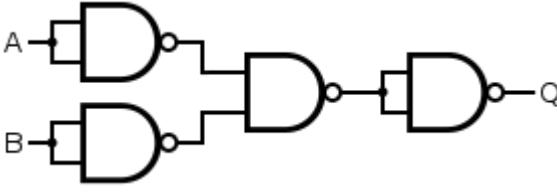
Desired Gate	NAND Construction	Proof																		
		$Q = \overline{\overline{A \cdot B}}$ $= \overline{(\overline{A \cdot B})'}$ $= A \cdot B$ (by Involution Rule)																		
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="3">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Input B</th> <th>Output Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>		Truth Table			Input A	Input B	Output Q	0	0	0	0	1	0	1	0	0	1	1	1	
Truth Table																				
Input A	Input B	Output Q																		
0	0	0																		
0	1	0																		
1	0	0																		
1	1	1																		

OR If the truth table for a NAND gate is examined, it can be seen that if any of the inputs are 0, then the output will be 1. To be an OR gate, however, the output must be 1 if any input is 1. Therefore, if the inputs are inverted, any high input will trigger a high output.


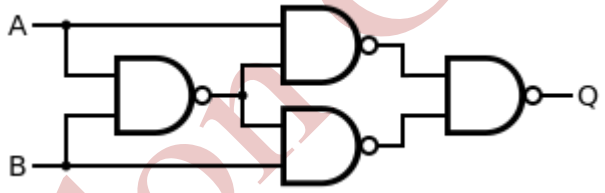
Desired Gate	NAND Construction	Proof																		
		$Q = \overline{\overline{A} \cdot \overline{B}}$ $= \overline{(\overline{A})' + (\overline{B})'}$ (By De Morgan's Law) $= A + B$ (By Involution Rule)																		
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="3">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Input B</th> <th>Output Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>		Truth Table			Input A	Input B	Output Q	0	0	0	0	1	1	1	0	1	1	1	1	
Truth Table																				
Input A	Input B	Output Q																		
0	0	0																		
0	1	1																		
1	0	1																		
1	1	1																		

3.2 Let us Try some other gates using NAND gate

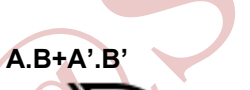
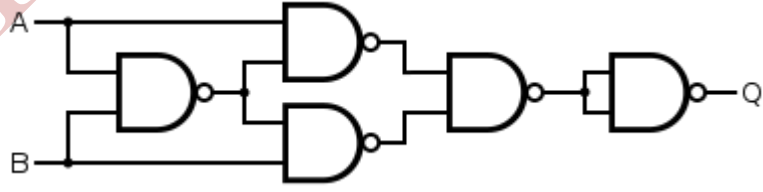
NOR A NOR gate is simply an inverted OR gate. Output is high when neither input A nor input B is high:

Desired Gate	NAND Construction																		
																			
<table border="1" style="margin: auto;"> <thead> <tr> <th colspan="3">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Input B</th> <th>Output Q</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>		Truth Table			Input A	Input B	Output Q	0	0	1	0	1	0	1	0	0	1	1	0
Truth Table																			
Input A	Input B	Output Q																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	

XOR An XOR gate is constructed similarly to an OR gate, except with an additional NAND gate inserted such that if both inputs are high, the inputs to the final NAND gate will also be high, and the output will be low. This effectively represents the formula: "NAND(A NAND (A NAND B)) NAND (B NAND (A NAND B))".

Desired Gate	NAND Construction																		
 $F(A,B) = A \cdot B' + A' \cdot B$																			
<table border="1" style="margin: auto;"> <thead> <tr> <th colspan="3">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Input B</th> <th>Output Q</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>		Truth Table			Input A	Input B	Output Q	0	0	0	0	1	1	1	0	1	1	1	0
Truth Table																			
Input A	Input B	Output Q																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	



XNOR - An XNOR gate is simply an XOR gate with an inverted output:

Desired Gate	NAND Construction																		
 $A \cdot B + A' \cdot B'$																			
<table border="1" style="margin: auto;"> <thead> <tr> <th colspan="3">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Input B</th> <th>Output Q</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>		Truth Table			Input A	Input B	Output Q	0	0	1	0	1	0	1	0	0	1	1	1
Truth Table																			
Input A	Input B	Output Q																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	


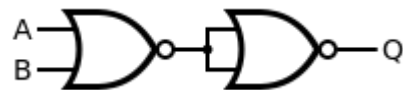
3.3 How NOR gate is a Universal gate?

A NOR gate is a universal gate, meaning that any other gate can be represented as a combination of NOR gates


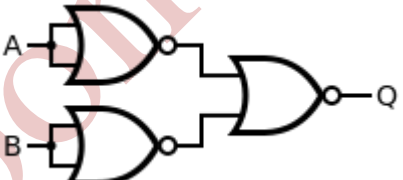
NOT This is made by joining the inputs of a NOR gate. As a NOR gate is equivalent to an OR gate leading to NOT gate, this automatically sees to the "OR" part of the NOR gate, eliminating it from consideration and leaving only the NOT part.

Desired Gate	NOR Construction	Proof								
		$Q = \overline{A + A}$ $= \overline{A} \text{ (by Idempotent Law)}$								
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Output Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>		Truth Table		Input A	Output Q	0	1	1	0	
Truth Table										
Input A	Output Q									
0	1									
1	0									

OR The OR gate is simply a NOR gate followed by a NOT gate.


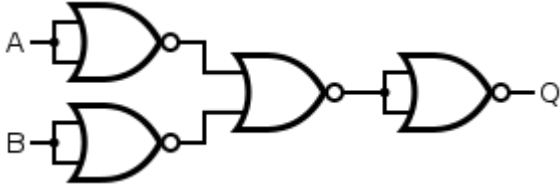
Desired Gate	NOR Construction	Proof																		
		$Q = \overline{\overline{A + B}}$ $= A + B$ $= A + B \text{ (by Involution Rule)}$																		
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="3">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Input B</th> <th>Output Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>		Truth Table			Input A	Input B	Output Q	0	0	0	0	1	1	1	0	1	1	1	1	
Truth Table																				
Input A	Input B	Output Q																		
0	0	0																		
0	1	1																		
1	0	1																		
1	1	1																		

AND An AND gate gives a 1 output when both inputs are 1; a NOR gate gives a 1 output only when both inputs are 0. Therefore, an AND gate is made by inverting the inputs to a NOR gate.


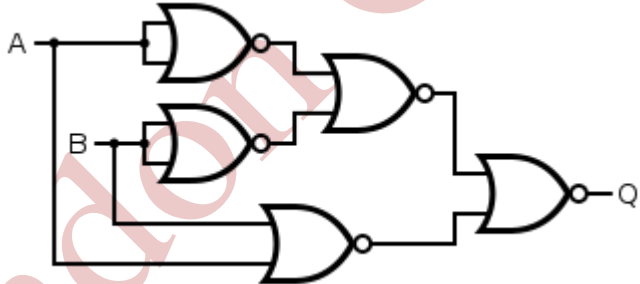
Desired Gate	NOR Construction	Proof																		
		$Q = \overline{\overline{A} + \overline{B}}$ $= \overline{\overline{A} \cdot \overline{B}} \text{ (By DeMorgan's Law)}$ $= A \cdot B \text{ (By Involution Rule)}$																		
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="3">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Input B</th> <th>Output Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>		Truth Table			Input A	Input B	Output Q	0	0	0	0	1	0	1	0	0	1	1	1	
Truth Table																				
Input A	Input B	Output Q																		
0	0	0																		
0	1	0																		
1	0	0																		
1	1	1																		

3.4 Let us try some other gates by using NOR gate

NAND A NAND gate is made using an AND gate in series with a NOT gate:

Desired Gate	NOR Construction																		
																			
<table border="1"> <thead> <tr> <th colspan="3">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Input B</th> <th>Output Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>		Truth Table			Input A	Input B	Output Q	0	0	1	0	1	1	1	0	1	1	1	0
Truth Table																			
Input A	Input B	Output Q																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	

XOR An XOR gate is made by connecting the output of 3 NOR gates (connected as an AND gate) and the output of a NOR gate to the respective inputs of a NOR gate. This expresses the logical fomula $(A \text{ AND } B) \text{ NOR } (A \text{ NOR } B)$. This construction entails a propagation delay three times that of a single NOR gate.

Desired Gate	NOR Construction																		
																			
<table border="1"> <thead> <tr> <th colspan="3">Truth Table</th> </tr> <tr> <th>Input A</th> <th>Input B</th> <th>Output Q</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>		Truth Table			Input A	Input B	Output Q	0	0	0	0	1	1	1	0	1	1	1	0
Truth Table																			
Input A	Input B	Output Q																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	

4. Applications of Logic gates

Logic gates have several applications to the electronic circuit of the computer and other digital circuits. Some of them are as follows:-

1. **Adder** - These are the circuits in ALU that performs the addition on two or more binary numbers.
2. **Subtractor** - These are the circuits that perform subtraction operation on two or more binary numbers.
3. **Encoder** - These are the circuits that convert decimal / octal / hexadecimal numbers to their equivalent binary number.
4. **Decoder** - These are the circuits that convert binary number to its equivalent decimal / octal / hexadecimal equivalent number.
5. **Multiplexer** - It is a combinational circuit that selects binary input from one of the many input lines and directs it to a single output line.
6. **Demultiplexer** - It is a combinational circuit that selects one of the many output lines and directs the input to that output line.

4.1 Adder

These are the circuits in ALU that performs the addition on two or more binary bits.

4.1.1. Half Adder

It is a logic circuit that adds two binary bits. It produces the output as SUM and CARRY. The Boolean equation for the SUM and CARRY is given below:

$$\text{SUM} = A \oplus B$$

$$\text{CARRY} = A \cdot B$$

(Min-term is a term in a Boolean Expression that always results 1 and it should be an AND operation. Now in a truth table, what will be the combination of each literal in a particular row that can yield 1 as output against a given set of input is the Min-term for that particular row)

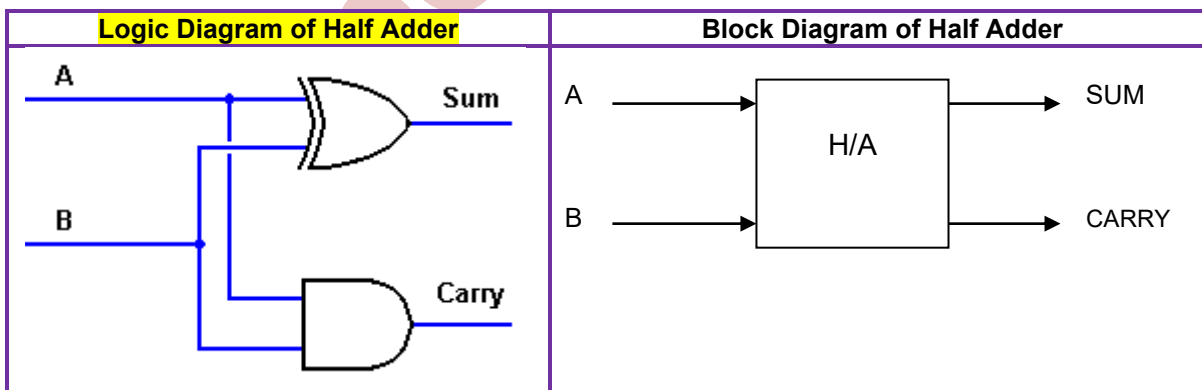
Let us see the truth table for adding two binary bits

A	B	SUM	CARRY	Min-term (1)
0	0	0	0	$A' \cdot B'$
0	1	1	0	$A' \cdot B$
1	0	1	0	$A \cdot B'$
1	1	0	1	$A \cdot B$

From the truth table, now we can derive the Boolean expression as

$$\text{SUM} = A' \cdot B + A \cdot B' = A \oplus B$$

$$\text{CARRY} = A \cdot B$$



4.1.2. Full Adder

It is a logic circuit that adds three binary bits. It produces the output as **SUM** and **CARRY**. The Boolean equation for the SUM and CARRY is given below:

$$\text{SUM} = A \oplus B \oplus C$$

$$\text{CARRY} = A.B + B.C + A.C$$

Let us see the truth table for adding three binary bits

A	B	C	SUM	CARRY	Minterm
0	0	0	0	0	$A'.B'.C'$
0	0	1	1	0	$A'.B'.C$
0	1	0	1	0	$A'.B.C'$
0	1	1	0	1	$A'.B.C$
1	0	0	1	0	$A.B'.C'$
1	0	1	0	1	$A.B'.C$
1	1	0	0	1	$A.B.C'$
1	1	1	1	1	$A.B.C$

From the truth table, now we can derive the Boolean expression as shown in the table below:

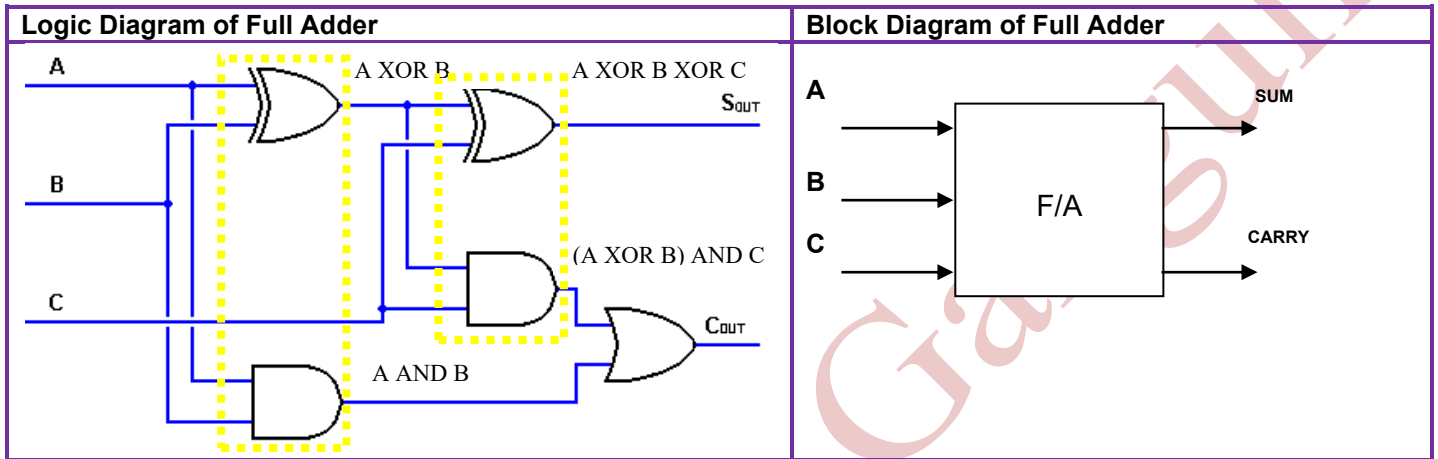
Expression for Sum & Carry	Logic Diagram of Full Adder
<p>SUM = $A'B'C + A'BC' + AB'C' + ABC$ $= (A'B'C + ABC) + (A'BC' + AB'C')$ $= (A'B' + AB).C + (A'B + AB').C'$ $= (A \oplus B).C + (A \oplus B).C'$ $= A \oplus B \oplus C$</p> <p>CARRY = $A'BC + AB'C + ABC' + ABC$ $= (A'BC + ABC) + (AB'C + ABC) + (ABC' + ABC)$ $= BC(A'+1) + AC(B'+1) + AB(C'+1)$ $= BC.1 + AC.1 + AB.1$ $= B.C + A.C + A.B$</p>	

4.1.3 Alternate circuit diagram for Full Adder (using two half adders & an OR gate)

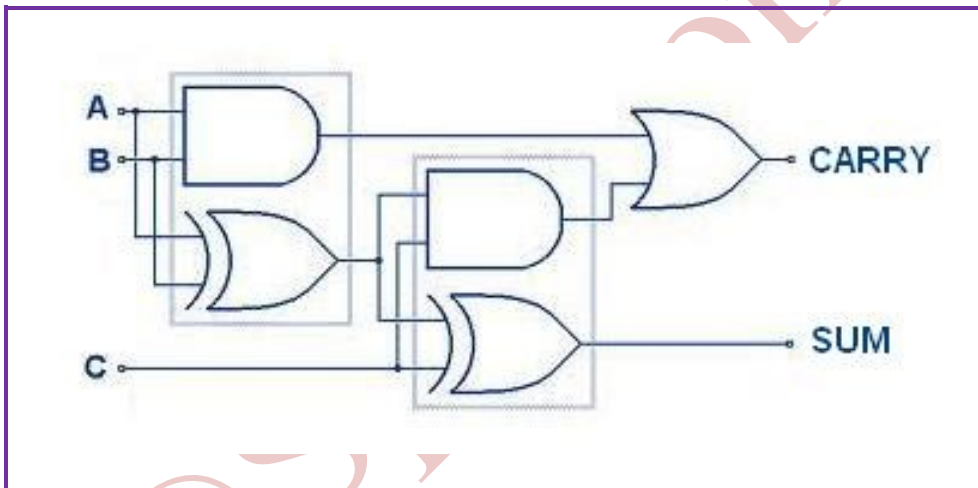
Here we can see two half adders –

$$\begin{aligned} \text{SUM} &= A'B'C + A'BC' + AB'C' + ABC = A \oplus B \oplus C \\ &= (A \oplus B) \oplus C \end{aligned}$$

$$\begin{aligned} \text{CARRY} &= A'BC + AB'C + ABC' + ABC \\ &= (A'BC + AB'C) + (ABC' + ABC) \\ &= (A'B + AB') \cdot C + AB(C' + 1) \\ &= (A \oplus B) \cdot C + AB \end{aligned}$$



Lets identify the half adders in the Full Adder diagram given below:



$$\begin{aligned} \text{SUM} &= A'B'C + A'BC' + AB'C' + ABC \\ &= (A \oplus B) \oplus C \\ \text{CARRY} &= A'BC + AB'C + ABC' + ABC \\ &= (A \oplus B) \cdot C + AB \end{aligned}$$

4.2 Subtractor

There are two types of subtractor circuits. They are given below:

4.2.1. Half Subtractor

It is a logic circuit that subtracts one binary bit from another binary bit. It produces the output as **DIFFERENCE** and **BORROW**. Let us see the truth table for subtracting two binary bits

A	B	DIFFERENCE	BORROW	Minterm
0	0	0	0	$A'.B'$
(1) 0	1	1	1	$A'.B$
1	0	1	0	$A.B'$
1	1	0	0	$A.B$

From the truth table, now we can derive the Boolean expression as

Expression for Difference & Borrow	Logic Diagram of Half Subtractor
<p>DIFF = $A'.B + A.B'$ = $A \oplus B$</p> <p>BORROW = $A'.B$</p>	

4.2.2. Full Subtractor

It is a logic circuit that subtracts three binary bits. It produces the output as DIFFERENCE and BORROW. Let us see the truth table for adding three binary bits ((A-B) - C)

A	B	C	DIFFERENCE	BORROW	Minterm
0	0	0	0	0	$A'B'C'$
0	(1) 0 = 10	1	1	1	$A'B'C$
(1) 0 = 10	1	0	1	1	$A'BC'$
(1) 0 = 10	1	1	0	1	$A'BC$
1	0	0	1	0	$AB'C'$
1	0	1	0	0	$AB'C$
1	1	0	0	0	ABC'
1	1	1	1	1	ABC

From the truth table, we can now derive the Boolean expression as shown in the table below:

Expression for Difference & Borrow	Logic Diagram of Full Subtractor
<p>DIFF = $A'B'C + A'BC' + AB'C' + ABC$</p> <p>= $(A'B'C + ABC) + (A'BC' + AB'C')$</p> <p>= $(A'B' + AB).C + (A'B + AB').C'$</p> <p>= $(A \oplus B).C + (A \oplus B).C'$</p> <p>= $A \oplus B \oplus C$</p> <p>BORROW</p> <p>= $A'B'C + A'BC' + A'BC + ABC$</p> <p>= $(A'B'C + ABC) + (A'BC' + A'BC)$</p> <p>= $(A'B' + AB).C + A'B(C' + 1)$</p> <p>= $(A \oplus B).C + A'.B$</p>	

4.3 Encoder

The process of converting one form to another form is known as encoding. The combinational circuit that converts a number in a denary system (decimal or octal, or hexadecimal) to another system (binary) is called an encoder. The encoder is also known as a Many-to-fewer circuit.

There are three types of Encoder circuits. They are given below:-

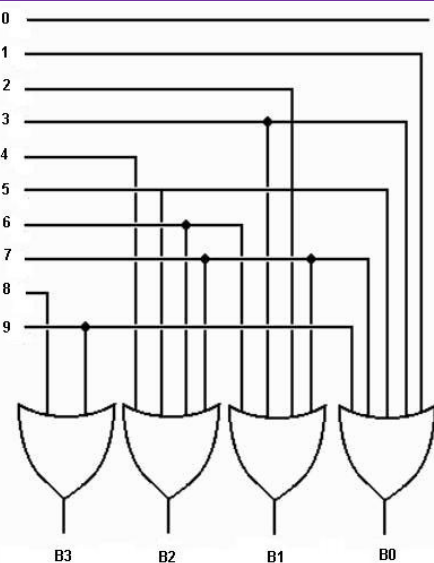
4.3.1. Decimal to Binary (10 to 4) Encoder

This encoder converts decimal numbers to equivalent binary numbers. As decimal numbers are 0 to 9 (i.e. 10 in numbers) and need to be converted to binary form, this circuit requires 10 input lines and 4 output lines. Thus this circuit is also called a 10-to-4 encoder circuit.

The truth table for 10-to-4 encoder:

DECIMAL (INPUT)	BINARY (OUTPUT)			
	B ₃	B ₂	B ₁	B ₀
D0	0	0	0	0
D1	0	0	0	1
D2	0	0	1	0
D3	0	0	1	1
D4	0	1	0	0
D5	0	1	0	1
D6	0	1	1	0
D7	0	1	1	1
D8	1	0	0	0
D9	1	0	0	1

Logic diagram of 10-to-4 Encoder



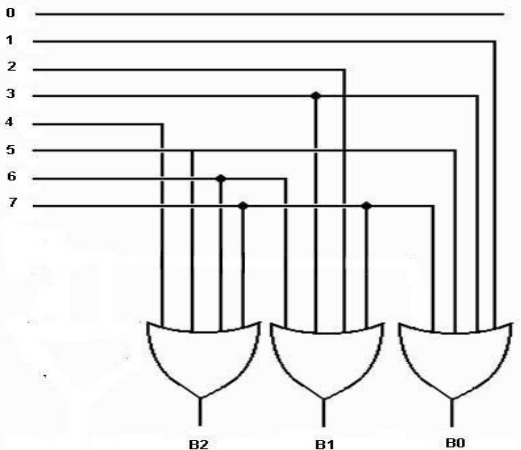
4.3.2. Octal to Binary (8 to 3) Encoder

This encoder converts decimal numbers to equivalent binary numbers. As octal numbers are 0 to 7 (i.e. 8 in numbers) and they are needed to be converted to binary form, this circuit requires 8 input lines and 3 output lines. Thus this circuit is also called 8-to-3 encoder circuit.

Truth table for 8-to-3 encoder:

OCTAL (INPUT)	BINARY (OUTPUT)		
	B ₂	B ₁	B ₀
O0	0	0	0
O1	0	0	1
O2	0	1	0
O3	0	1	1
O4	1	0	0
O5	1	0	1
O6	1	1	0
O7	1	1	1

Logic diagram of 8-to-3 Encoder



4.3.3. Hexadecimal to Binary (16 to 4) Encoder

This encoder converts decimal numbers to equivalent binary numbers. As hexadecimal numbers are 0 to 9 and A to F (i.e. 10+6=16 in numbers) and must be converted to binary form, this circuit requires 16 input lines and 4 output lines. Thus, this circuit is also called a 16-to-4 encoder circuit.

The truth table for 16-to-4 encoder:

HEXADECIMAL (INPUT)	BINARY (OUTPUT)			
	B ₃	B ₂	B ₁	B ₀
Hx0	0	0	0	0
Hx1	0	0	0	1
Hx2	0	0	1	0
Hx3	0	0	1	1
Hx4	0	1	0	0
Hx5	0	1	0	1
Hx6	0	1	1	0
Hx7	0	1	1	1
Hx8	1	0	0	0
Hx9	1	0	0	1
HxA	1	0	1	0
HxB	1	0	1	1
HxC	1	1	0	0
HxD	1	1	0	1
HxE	1	1	1	0
HxF	1	1	1	1

4.4 DECODER

A decoder is a combinational circuit which converts binary number into its equivalent decimal or octal or hexadecimal form. There are three types of Decoder circuits. They are given below:-
Decoders are also known as Fewer-to-many circuit.

4.4.1. Binary to Decimal (4 to 10) Decoder

This decoder converts binary numbers to equivalent decimal numbers. As decimal numbers are 0 to 9 (i.e. 10 in numbers) and the binary numbers are needed to be converted to decimal form, this circuit requires 4 input lines and 10 output lines. Thus this circuit is also called 4-to-10 decoder circuit.

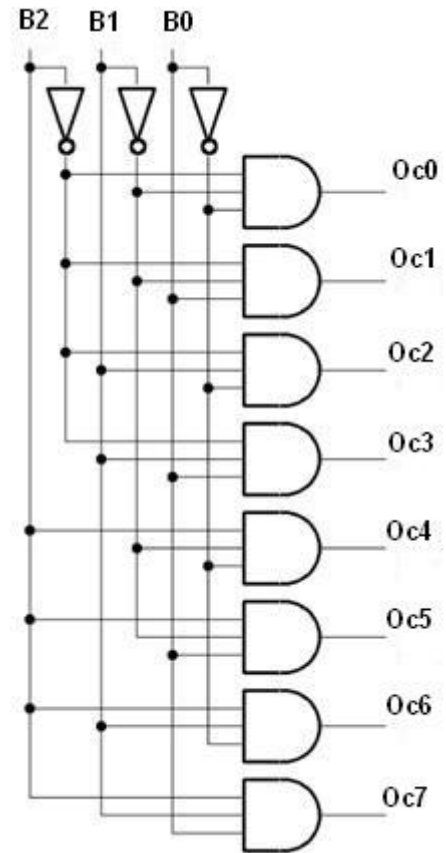
Truth table for 4-to-10 decoder:

BINARY (INPUT)				DECIMAL (OUTPUT)	MINTERM	Logic diagram of 4-to-10 Decoder
B3	B2	B1	B0			
0	0	0	0	D0	$B3'.B2'.B1'.B0'$	
0	0	0	1	D1	$B3'.B2'.B1'.B0$	
0	0	1	0	D2	$B3'.B2'.B1.B0'$	
0	0	1	1	D3	$B3'.B2'.B1.B0$	
0	1	0	0	D4	$B3'.B2.B1'.B0'$	
0	1	0	1	D5	$B3'.B2.B1'.B0$	
0	1	1	0	D6	$B3'.B2.B1.B0'$	
0	1	1	1	D7	$B3'.B2.B1.B0$	
1	0	0	0	D8	$B3.B2'.B1'.B0'$	
1	0	0	1	D9	$B3.B2'.B1'.B0$	

4.4.2. Binary to Octal (3 to 8) Decoder

This decoder converts binary numbers to equivalent octal numbers. As octal numbers are 0 to 7 (i.e. 8 in numbers) and the binary numbers are needed to be converted to octal form, this circuit requires 3 input lines and 8 output lines. Thus this circuit is also called a 3-to-8 decoder circuit.

The truth table for 3-to-8 decoder:

BINARY (INPUT)			OCTAL (OUTPUT)	MINTERM	Logic diagram of 3-to-8 Decoder
B2	B1	B0			
0	0	0	Oc0	$B2'.B1'.B0'$	
0	0	1	Oc1	$B2'.B1'.B0$	
0	1	0	Oc2	$B2'.B1.B0'$	
0	1	1	Oc3	$B2'.B1.B0$	
1	0	0	Oc4	$B2.B1'.B0'$	
1	0	1	Oc5	$B2.B1'.B0$	
1	1	0	Oc6	$B2.B1.B0'$	
1	1	1	Oc7	$B2.B1.B0$	

4.4.3. Binary to Hexadecimal (4 to 16) Decoder

This decoder converts binary numbers to equivalent hexadecimal numbers. As hexadecimal numbers are 0 to 9 and A to F (i.e. 10+6=16 in numbers) and the binary numbers are needed to be converted to hexadecimal form, this circuit requires 4 input lines and 16 output lines. Thus this circuit is also called a 16-to-4 decoder circuit.

The truth table for 4-to-16 decoder:

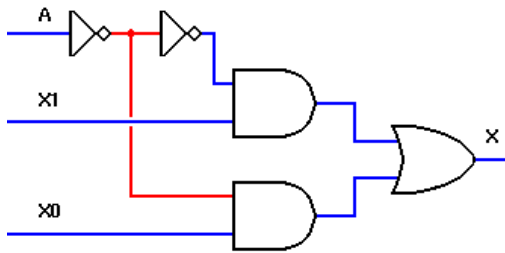
BINARY (INPUT)				HEXADECIMAL
B3	B2	B1	B0	
0	0	0	0	Hx0
0	0	0	1	Hx1
0	0	1	0	Hx2
0	0	1	1	Hx3
0	1	0	0	Hx4
0	1	0	1	Hx5
0	1	1	0	Hx6
0	1	1	1	Hx7
1	0	0	0	Hx8
1	0	0	1	Hx9
1	0	1	0	Hx10
1	0	1	1	Hx11
1	1	0	0	Hx12
1	1	0	1	Hx13
1	1	1	0	Hx14
1	1	1	1	Hx15

5. Multiplexer

5.1 Multiplexer

It means “many into one”. A Multiplexer is a combinational circuit that selects binary input from one of the many input lines and directs it to a single output line. This is a digital circuit with multiple signal inputs, one of which is selected by separate address inputs to be sent to the single output. Multiplexer is also called Data selector as because it selects one of the several inputs and then passes that to the single output.

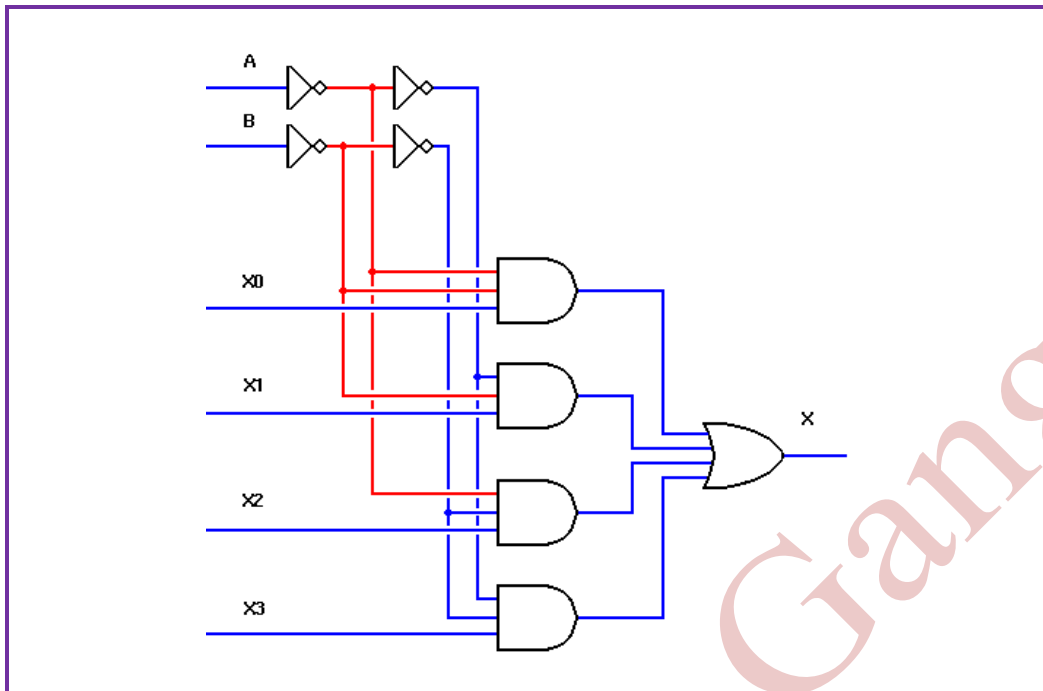
A two-input multiplexer is shown below



In the above logic diagram, A is the select line, that determines which one of the input signal (X1 or X0) will be selected and passed to the output line (x).

Truth Table for the 2-input multiplexer		Block diagram for the Multiplexer	
Select Line (A)	Output (x)		
0	X0		
1	X1		

A four-input multiplexer is shown below



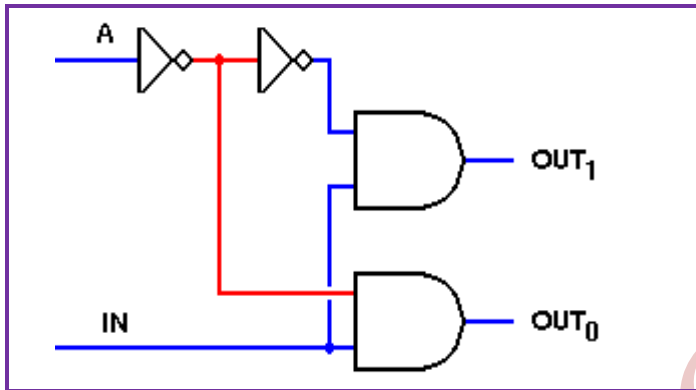
In the above logic diagram, A and B are the select lines, that determines which one of the input signal (X0 or X1 or X2 or X3) will be selected and passed to the output line (x).

Select Lines		Output (x)	Block diagram for the Multiplexer	
A	B		X0	X1
0	0	X0		
0	1	X1		
1	0	X2		
1	1	X3		
1	1	X3		

6. Demultiplexer

6.1 It is a combinational circuit that selects one of the many output lines and directs the input to that output line. The opposite of the multiplexer circuit, logically enough, is the *demultiplexer*. This circuit takes a single data input and one or more address inputs, and selects which of multiple outputs will receive the input signal. The same circuit can also be used as a *decoder*, by using the address inputs as a binary number and producing an output signal on the single output that matches the binary address input. In this application, the data input line functions as a circuit enabler — if the circuit is disabled, no output will show activity regardless of the binary input number.

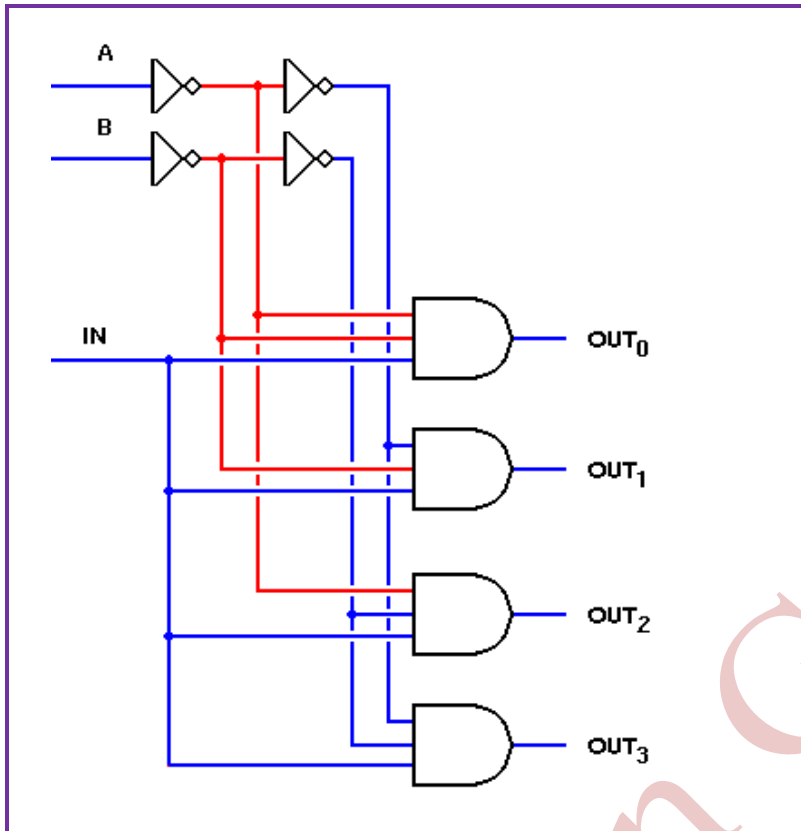
A one-line to two-line demultiplexer is shown below



Truth Table for the 1-input 2-output de-multiplexer		Block diagram for the De-multiplexer	
Select Line (A)	Output		
0	X0		
1	X1		

Like the multiplexer circuit, the demultiplexer is not limited to a single address line, and therefore can have more than two outputs. With two, three, or four addressing lines, this circuit can decode a two, three, or four-bit binary number, or can demultiplex up to four, eight, or sixteen time-multiplexed signals.

A 2-to-4 line demultiplexer is shown below



Truth Table for the 4-output de-multiplexer			Block diagram for the De-multiplexer
Select Lines		Output	
A	B		
0	0	Out0	
0	1	Out1	
1	0	Out2	
1	1	Out3	

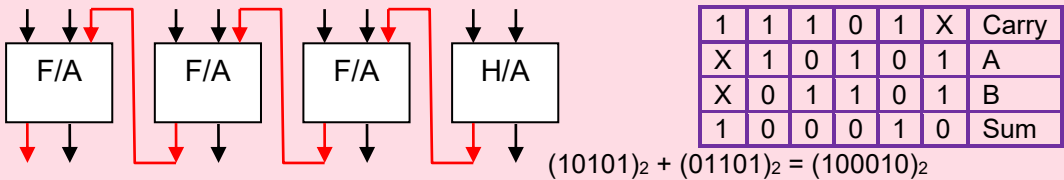
Practice problems

1.

What is an Adder? Draw a logic diagram of 4-bit binary adder and show the addition of 2 binary numbers of 4 bit length. Given the numbers – A = (10101)₂ and B = (01101)₂

Answer:

Adder is a circuit that adds binary digits. It takes more than 1 input and give 2 outputs as Sum bit and Carry bit.



2.

State the difference between multiplexer and decoder.

Answer:

Multiplexer	Decoder
A multiplexer is a device that takes two or more signals and encodes them on fewer (often, on one) wire.	Decoder converts an input binary code into a corresponding single active output.
Multiplexer used to direct one input out of many to a single output.	Decoder is used to convert one code form to another code (e.g. binary to gray) depends on design.
Many input to one output.	One input to many output.

3.

What are select/address lines in a multiplexer?

Answer:

Function of Select lines or address lines is that for a multiplexer of 2ⁿ inputs there will be n select lines. Thus we can say that the select lines are used to select which input line to be send to the output line.

4.

How many select lines does an 8:1 multiplexer have?

Answer:

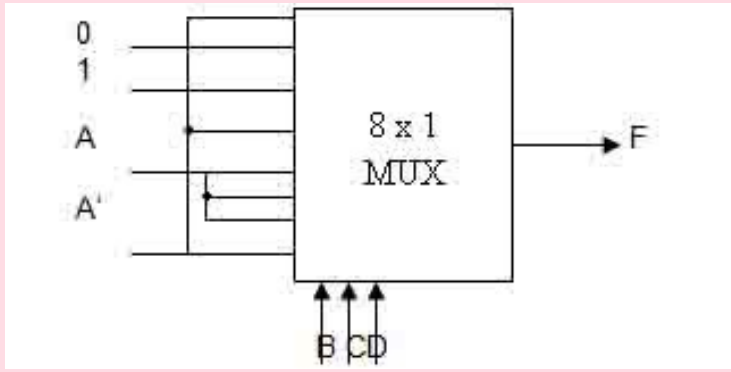
8 Input MUX will have 3 Select Line as by formula, $(2)^3 = 8$.

5.

Implement the Boolean function with 8:1 multiplexer. $F(A,B,C,D) = \sum (0, 1, 3, 7, 9, 12, 13, 14)$

Answer:

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
	000	001	010	011	100	101	110	111
A'	0	1		3				7
A		9			12	13	14	
	A'	1	0	A'	A	A	A	A'



6.

Implement the following Boolean function with 8 x 1 multiplexer.
 $F(A,B,C,D) = \sum(0,3,5,6,8,9,14,15)$

Answer:

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	A	0	A'	0	A'	1	A

0 = I_2, I_4
 1 = I_0, I_6
 $A' = I_3, I_5$
 $A = I_1, I_7$

Similarity between Decoder and Multiplexer

Both multiplexer and decoder are types of combinational digital circuits that are extensively used in the transfer of signals in many communication systems. Basically, the multiplexer and decoder both perform almost identical functions.

Similarity between Decoder and Demultiplexer

Difference between Encoder and Multiplexer

A multiplexer or MUX is a combination circuit that contains more than one input line, one output line and more than one selection line. Whereas, an encoder is also considered a type of multiplexer but without a single output line.

Difference between Encoder and Decoder

ENCODER	DECODER
Encoder circuit basically converts the applied information signal into a coded digital bit stream.	Decoder performs the reverse operation and recovers the original information signal from the coded bits.
In case of an encoder, the applied signal is the active signal input.	Decoder accepts coded binary data as its input.
The number of inputs accepted by an encoder is 2^n .	The number of input accepted by decoder is only n inputs.
The output lines for an encoder is n .	The output lines of an decoder is 2^n .
The encoder generates coded data bits as its output.	The decoder generates an active output signal in response to the coded data bits.
The encoder circuit is installed at the transmitting end.	The decoder circuit is installed at the receiving side.
OR gate is the basic logic element used in it.	AND gate along with NOT gate is the basic logic element used in it.
It is used in E-mail, video encoders etc.	It is used in Microprocessors, memory chips etc.

Difference between Decoder and Demultiplexer

Decoder	Demultiplexer
These are Logic circuit which decodes an encrypted input stream from one to another format.	It is a combination circuit which routes a single input signal to one of several output signals.
n number of input lines and 2^n number of output lines.	n number of select lines and 2^n number of output lines.
In Detection of bits, data encoding.	In Distribution of the data, switching.
It is used for changing the format of the instruction in the machine specific language.	It is used as a routing device to route the data coming from one signal into multiple signals.
Majorly implemented in the networking application.	Employed in data-intensive applications where data need to be changed into another form.

Difference between Decoder and Multiplexer

Multiplexer	Decoder
Multiplexer has n data input lines and one output line.	Decoder has no data input and there is n output line.
There are m control lines.	Inputs are the control bits A, B, C, D.
Depending upon the status of the control line, data on the particular input line is available at the output.	Only one output is high which depends on the status of the control lines.
Converts the unary code to binary code	Converts binary code into unary.

ANSWER THE FOLLOWING

1. What is a Full Adder? Draw the truth table, derive its Boolean expression and draw a logic diagram for Full Adder.
2. What is Encoder? Draw the truth table and logic diagram of 10-by-4 encoder.
3. Prove that $X' \oplus Y = X \oplus Y' = (X \oplus Y)' = X.Y + X'.Y'$
4. State the dual form of the following: $XY' (XY'Z + X + X'Z')$
5. Prove that $F(A,B,C) = \pi (2,3,4,7) = \Sigma (0,1,5,6)$
6. Using NOR gates only draw a logic diagram to construct NAND gate.
7. Verify that $((P' + Q).(Q' + R))' + (P' + R) = 1$
8. What is the principle of duality? Give one example.
9. Find the complement of $F = X + Y.Z$; then show that $F.F' = 0$ and $F + F' = 1$
10. Obtain the logic circuit of the given Boolean expressions using NOR gates only
 - a. $F(X,Y,Z) = (X + Y).(Y + Z).(Z + X)$
 - b. $F(A,B,C,D) = (\overline{A + B}).\overline{B}. \overline{C}. (A + D)$
11. Write the product-of-sum for the Boolean function, $F(A,B,C)$ whose output is 0 only when: $A=1, B=0, C=0$; $A=0, B=1, C=0$; $A=0, B=0, C=1$; $A=1, B=1, C=1$, also draw the logic diagram using NOR gate
12. State any two Applications of Multiplexer.
13. State any two Applications of Decoder.
14. What is a Full Adder? Draw a logic diagram of full adder using two half adders and an OR gate. Write down the SUM and CARRY bit expression also.
15. What is a decoder? What is its application in computer hardware circuit? Draw the decoder circuit of Binary to Decimal converter
16. What is an Adder?
17. Draw a Full Adder using two half adders and an OR Gate.
18. Implement a Full-Adder circuit using a decoder.
19. Implement the following function with a multiplexer: $F(a,b,c,d) = \Sigma(0,1,3,4,8,9,15)$