BYJU'S EXAM PREP

Computer Science & IT

Digital Logic

SHORT NOTES

Short Notes — DIGITAL LOGIC

Chapter 1: Number System

1. INTRODUCTION TO NUMBER SYSTEMS:

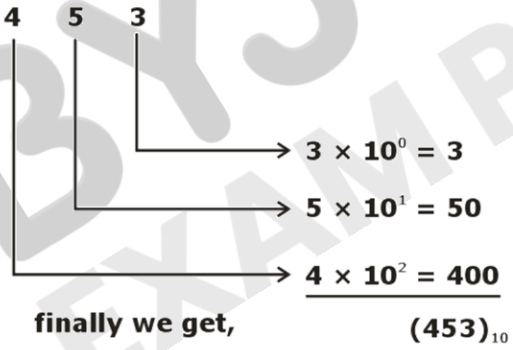
On the basis of number of different symbols used, number systems are classified as:

- Decimal number system
- Binary number system
- Octal number system
- Hexadecimal number system

1.1. Decimal number system:

Example 1:

Let's for example we have $(453)_{10}$ is a decimal number then,



We can say that '3' is the least significant digit (LSD) and '4' is the most significant digit (MSD).

1.2. Binary number system:

4 bits = 1 Nibble, 8 bits = 1 Byte.

Example 3:

The decimal number representation of (101101.10101) is?

Solution:

$(101101.10101)_2$ =

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5}$

$= 32 + 0 + 8 + 4 + 0 + 1 + \frac{1}{2} + 0 + \frac{1}{8} + 0 + \frac{1}{32}$

$= (45.65625)_{10}$

1.3. Octal number system:

Example 4:

Convert $(367.721)_8$ into a decimal system.

Solution:

$(367.721)_8 = 3 \times 8^2 + 6 \times 8^1 + 7 \times 8^0 + 7 \times 8^{-1} + 2 \times 8^{-2} + 1 \times 8^{-3}$

$= 192 + 48 + 7 + 0.875 + 0.03125 + 0.00195$

$= (247.908)_{10}$

1.4. Hexadecimal number system:

Example 5:

Convert $(3A.2F)_{16}$ into a decimal system.

Solution:

$(3A\ .2F)_{16} = 3 \times 16^1 + 10 \times 16^0 + 2 \times 16^{-1} + 15 \times 16^{-2}$

$$= 48 + 10 + \frac{2}{16} + \frac{15}{16^2}$$

$= (58.1836)_{10}$

2. CONVERSION BETWEEN DIFFERENT NUMBER SYSTEM

2.1. OTHER BASE SYSTEM TO DECIMAL

2.1.1. Binary to Decimal Conversion

Example 6:

Convert $(11101.1011)_2$ into decimal.

Solution:

$$(11101 \cdot 1011)_2 = (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) + (1 \times 2^{-4}) = 16 + 8 + 4 + 0 + 1 + 0.5 + 0 + 0.125 + 0.0625 = (29.6875)_{10}$$

2.1.2. Octal to Decimal Conversion

Example 7: Convert $(6327.4051)_8$ into an equivalent decimal number.

Solution:

$(6327.4051)_8$

$= 6 \times 8^3 + 3 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} + 0 \times 8^{-2} + 5 \times 8^{-3} + 1 \times 8^{-4}$

$$= 3072 + 192 + 16 + 7 + \frac{4}{8} + 0 + \frac{5}{512} + \frac{1}{4096}$$

$= (3287.5100098)_{10}$

Thus,

$(6327.4051)_8 = (3287.5100098)_{10}$

2.1.3. Hexadecimal to Decimal Conversion

Example 8:

Convert $(5C7)_{16}$ into decimal.

Solution:

$(5C7)_{16} = 5 \times 16^2 + 12 \times 16^1 + 7 \times 16^0$

$= 1280 + 192 + 7$

$= (1479)_{10}$

2.2. DECIMAL TO OTHER BASE SYSTEM

2.2.1. Decimal to binary conversion:

Example 9: Convert $(13)_{10}$ to binary

Solution:

	Quotient	Remainder	
13 ÷ 2	6	1	LSB
6 ÷ 2	3	0	
3 ÷ 2	1	1	
1 ÷ 2	0	1	MSB

So, $(13)_{10} = (1101)_2$

2.2.2. Decimal to Octal conversion:

Example 10:

Convert $(675 \cdot 625)_{10}$ into octal.

Solution:

For integer part:

	Quotient	Remainder
675 ÷ 8	84	3
84 ÷ 8	10	4
10 ÷ 8	1	2

$\therefore \quad (675)_{10} = (1243)_8$

For fractional part,

$0.625 \times 8 = 5$

$\therefore \quad (0.625)_{10} = (5)_8$

Finally,

$(675 \cdot 625)_{10} = (1243 \cdot 5)_8$

2.2.3. Decimal to Hexadecimal conversion:

Example 11: Convert $(675.625)_{10}$ into hexadecimal

Solution: For Integral part:

	Quotient	Remainder
675 ÷ 16	42	3
42 ÷ 16	2	10 = A
2 ÷ 16	0	2

So, $(675)_{10} = (2A3)_{16}$

For fractional part,

$0.625 \times 16 = 10 = A$

$(0.625)_{10} = (0.A)_{16}$

Finally,

$(675.625)_{10} = (2A3.A)_{16}$

2.3. Octal to Binary Conversion

Example 12:

Convert $(27633)_8$ into binary

Solution:

2	7	6	3	3
↓	↓	↓	↓	↓
010	111	110	011	011

$\therefore (27633)_8 = (010\ 111\ 110\ 011\ 011)_2$

2.4. Binary to Octal Conversion

Example 13:

Convert $(1011011110.11001010011)_2$ into octal.

Solution:

In this conversion, the binary bit stream are grouped into groups of three bits starting at the LSB and then each group is converted into its octal equivalent.

For left side of the radix point, we grouped the bits from LSB:

001	011	011	110
↓	↓	↓	↓
1	3	3	6

Here, two 0's at MSB are added to make a complete group of 3 bits.

For right side of the radix point, we grouped the bits from MSB:

.	110	010	100	110
Radix point	↓	↓	↓	↓
	6	2	4	6

Here, a '0' at LSB is added to make a complete group of 3 bits.

Finally,

$(1011011110.11001010011)_2 = (1336.6246)_8$

2.5. Hexadecimal to binary conversion:

Example 14: Convert $(2F9A)_{16}$ to binary system

2	F	9	A
|	|	|	|
0010	1111	1001	1010

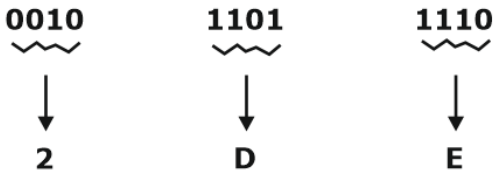
So, $(2F9A)_{16} = (0010\ 1111\ 1001\ 1010)_2$

2.6. Binary to hexadecimal ConversionExample 15:

Convert $(1011011110.11001010011)_2$ into hexadecimal.

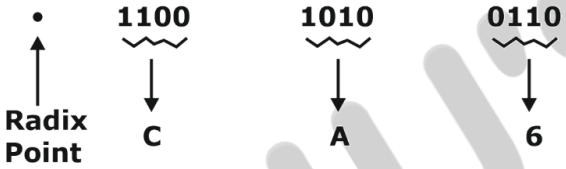
Solution:

For left side of the radix point, we grouped the bits from LSB:



Here, two 0's at MSB are added to make a complete group of 4 bits.

For right side of the radix point, we grouped the bits from MSB:



Here, a '0' at LSB is added to make a complete group of 4 bits.

Finally,

$(1011011110.11001010011)_2 = (2DE.CA6)_{16}$

2.7. Hexadecimal to Octal and Octal to Hexadecimal conversion:

Example 16: A particular number system having base B is given as $(\sqrt{41})_B = 5_{10}$ Find the value of B?

Solution: Squaring both sides:

$(\sqrt{41})_B^{\,2} = (5)^2$

$(41)_B = (25)_{10}$

Now, we have to convert the LHS into decimal to solve the equation:

$(4B + 1)_{10} = (25)_{10}$

So, B = 6

3. BASIC BINARY ARITHMETIC OPERATIONS:

Example 17: Add 110110 and 101101

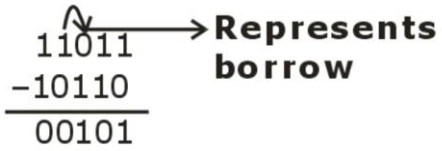
$$\begin{array}{r} \overset{\curvearrowleft\,\curvearrowleft\,\curvearrowleft}{110110} \\ +\ 101101 \\ \hline \text{carry}\ \ \textcircled{1}100011 \end{array}$$

Here, (arrow) indicates carry.

3.1. Binary Subtraction:

Example 18: Subtract 11011 and 10110?

Solution:



3.2. Binary Multiplication:

Example 19: $(1111)_2 \times (1111)_2 = ?$

$$\begin{array}{r} 1111 \times 1111 \\ \hline 1111 \\ 1111\times \\ 1111\times\times \\ 1111\times\times\times \\ \hline 11100001 \end{array}$$

3.3. Binary Division:

4. SIGNED NUMBER REPRESENTATION:

Example 20: 0011 represents a positive number (3), 1011 represents a negative number (−3)

In General, maximum negative to positive number that can be represented by using sign magnitude form is:

$$-(2^{n-1}-1) \text{ to } +(2^{n-1}-1)$$

4.1. r's complement representation:

4.1.1. (r-1) 's complement

For Example: In the binary number system, the base is 2. Hence, its (r-1)'s i.e., (2-1 =1)'s complement can be obtained by subtracting each bit from 1, i.e., 1's complement for 001 can also be calculated by subtracting 001 from 111 which will be (111-001) = $(110)_2$.

4.1.2. r's complement

For Example: In binary number system, 2's complement of 001 can be calculated by adding 1 to the LSB of its 1'complement (i.e., 110 + 1) = $(111)_2$.

Example 21: For 4-bit maximum possible number:

$= 2^4-1 = (15)_{10} = (1111)_2$

Example 22: Determine 9's complement of decimal number 2689?

Solution: For a decimal number, maximum possible number of 4 digit is 9999

∴ 9's compliment of 2686 is:

$$\begin{array}{r} 9999 \\ -2689 \\ \hline 7310 \end{array}$$

Example 23: Determine 8's complement of an octal number 2670?

Solution: To determine the r's complement, first write (r-1)'s complement of given number then add 1 in the least significant position.

$$\begin{array}{rr} & 7777 \\ & -2670 \\ \hline 7\text{'s compliment} = & 5107 \end{array}$$

Now, 8's complement = 7's complement + 1

$$5107 + 1 = \underset{\substack{\uparrow \\ \text{octal}}}{5108} = 5110$$

Note: 08 in Decimal will be 10 in octal.(As octal numbers range from 0 to 7.)

4.2. One's complement representation:

In a binary number, for one's complement representation, replace each 0 by 1 and each 1 by 0. Maximum negative to positive number that can be represented using one's complement is:

$$\mathbf{-(2^{n-1} - 1) \text{ to } +(2^{n-1} - 1)}$$

Example 24: Find one's complement of 101101

Solution: 101101 1's complement will be: 010010

4.3. Two's complement representation:

By adding a '1' to the one's complement representation, we can get 2's complement of that number. For 'n' bit number, maximum negative to positive number that can be represented is

$$\mathbf{-(2^{n-1}) \text{ to } +(2^{n-1} - 1)}$$

Example 25: Subtract decimal number 22 from 17 using 8 bit 2's compliment method?

Solution:

Binary form of 22 = 00010110

Binary form of 17 = 00010001

$$\begin{array}{rr} \mathbf{17 =} & \mathbf{00010001} \\ \mathbf{-22 =} & \mathbf{+11101010} \\ \hline \mathbf{-5 =} & \mathbf{11111011} \end{array}$$

In addition, there is no carry. The MSB of the result is '1'. Hence, it is a negative number and in its 2's complement form.

∴ 2's complement (11111011) = 00000101 = $(5)_{10}$

∴ The answer = −5 in decimal representation.

5. Sign Bit Extension:

For Signed Magnitude Form:

Ex: -3 = 111 in sign magnitude form in 3 bits.

To represent it in 8 bits , we will write it as : 1 0 0 0 0 0 1 1.

For 1's Complement Form:

Ex: -5 = 1010 in 1's complement in 4 bits.

To represent it in 8 bits , we will write it as : 1 1 1 1 1 0 1 0.

For 2's complement Form:

How is the number extension done?

Here, the number extension is same as 1's complement form.

6. BINARY CODES:

6.1. BCD code (Binary coded decimal):

Example 26: $(943)_{10} = (?)_{BCD}$

Solution:

$$\begin{array}{ccc} 9 & 4 & 3 \\ \downarrow & \downarrow & \downarrow \\ 1001 & 0100 & 0011 \end{array}$$

$\therefore (943)_{10} = (1001\ \ 0100\ \ 0011)_2$

6.1.1. BCD addition:

Example 27: Convert the decimal number $(76)_{10}$ and $(94)_{10}$ in BCD and add them.

Solution:

$$\begin{array}{rll} 76 & = 0111 & 0110 \\ +94 & = 1001 & 0100 \\ \hline (170)_{10} & 10000 & 1010 \end{array}$$

From the rule, we add (0110)

$$\begin{array}{rll} & 10000 & 1010 \\ & +0110 & 0110 \\ \hline 1 & 0111 & 0000 \\ \underbrace{} & \underbrace{} & \underbrace{} \\ 1 & 7 & 0 \end{array}$$

$\therefore$ In BCD $\Rightarrow (170)_{10}$

6.2. Excess-3 code:

Example 28: Convert $(48)_{10}$ into excess-3 code?

Solution:

4	8
+3	+3
7	11
↓	↓
0111	1011

∴ $(48)_{10}$ = (01111011) → Equivalent 4-bit binary number.

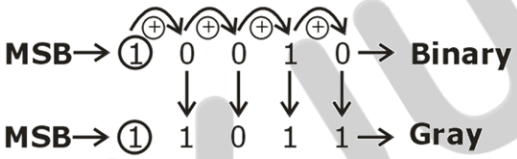
6.3. Gray-code:

- It is an unweighted code.
- Also known as "Reflected code".

7. BINARY TO GRAY CONVERSION:

Example 30: Convert $(10010)_2$ to gray code?

Solution:

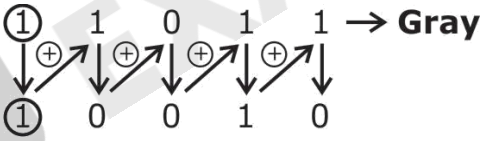


$(10010)_2$ → $(11011)_{gray}$.

8. GRAY TO BINARY CONVERSION:

Example 31: Convert $(11011)_{Gray}$ to binary code?

Solution:



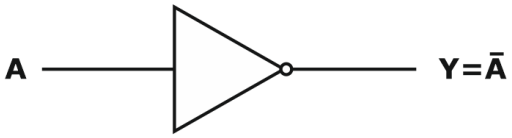
∴ $(11011)_{Gray}$ = $(10010)_2$.

Chapter 2: BOOLEAN ALGEBRA & LOGIC GATES

1. LOGIC OPERATIONS

1.1. NOT operation:

Symbol:



$A \xrightarrow{NOT} \bar{A}$ or A' (Complementation law)

and $\bar{\bar{A}} = A$ ⇒ Double complementation law

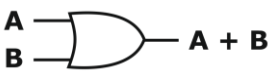
1.2. AND operation:

Symbol:

$A.A = A$, $A.0 = 0$, $A.1 = A$, $A\bar{A} = 0$

1.3. OR operation:

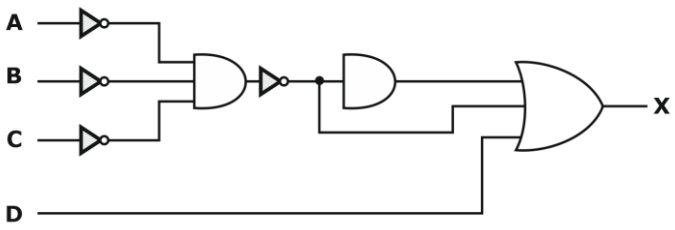
Symbol:



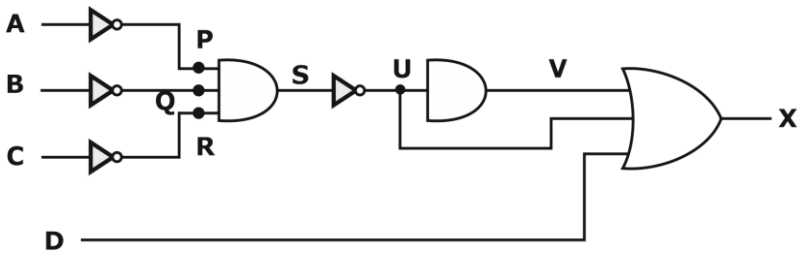
$A + A = A$, $A + 0 = A$, $A + 1 = 1$, $A + \bar{A} = 1$

Example 1.1:

Reduce the combinational logic circuit shown figure such that the desired output can be obtained using only one gate.



Solution:



$P = \overline{A}$, $Q = \overline{B}$, $R = \overline{C}$

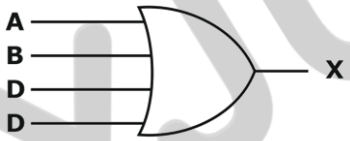
$S = \overline{A} \cdot \overline{B} \cdot \overline{C}$

$V = U = \overline{\overline{A}\,\overline{B}\,\overline{C}}$

$X = U + V + D$

$= \overline{\overline{A}\,\overline{B}\,\overline{C}} + \overline{\overline{A}\,\overline{B}\,\overline{C}} + D$

$= A + B + C + D$

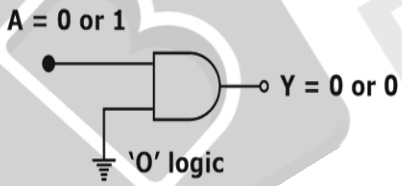


Example 1.2:

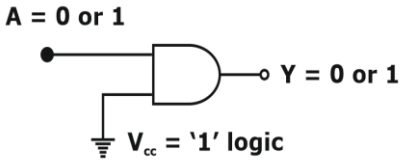
Show the control enable and disable outputs for AND and OR gate.

Solution:

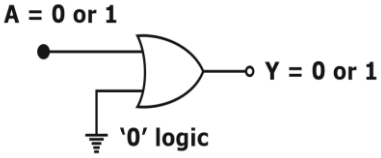
Control '0' disable



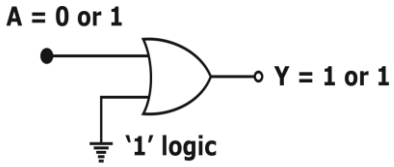
Control '1' enable (Buffer)



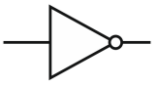
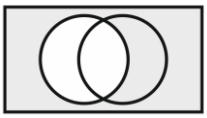
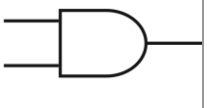
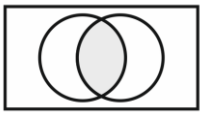
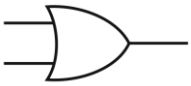
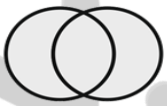
Control '0' enable (Buffer)



Control '1' Always enable



1.4. Venn Diagram:

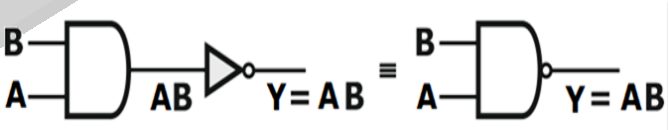
NOT			$\bar{A}$	A		Output
				0		1
				1		0
AND			A.B	A	B	Output
				0	0	0
				0	1	0
				1	0	0
				1	1	1
OR			A + B	A	B	Output
				0	0	0
				0	1	1
				1	0	1
				1	1	1

2. LOGIC GATES

2.1. NAND gate:

The term NAND implies NOT-AND

Symbol:

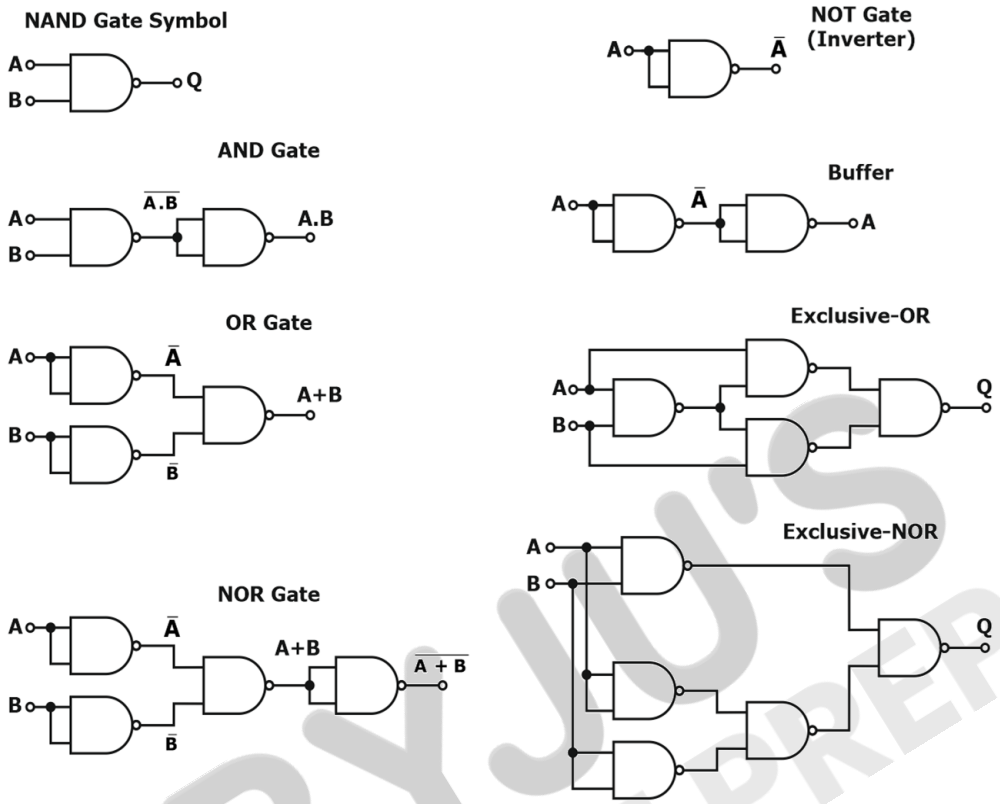


Truth table of 2-input NAND gate.

Input		Output
A	B	$Y = \overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

NAND gate acts as Universal Gate

Logic Gates using only NAND Gates

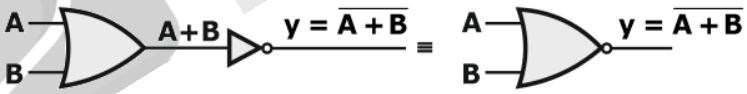


All the logic gate functions can be created using only NAND gates. Therefore, it is also known as a Universal logic gate.

2.2. NOR gate:

A NOR gate is equivalent to OR gate followed by a NOT gate.

Symbol:

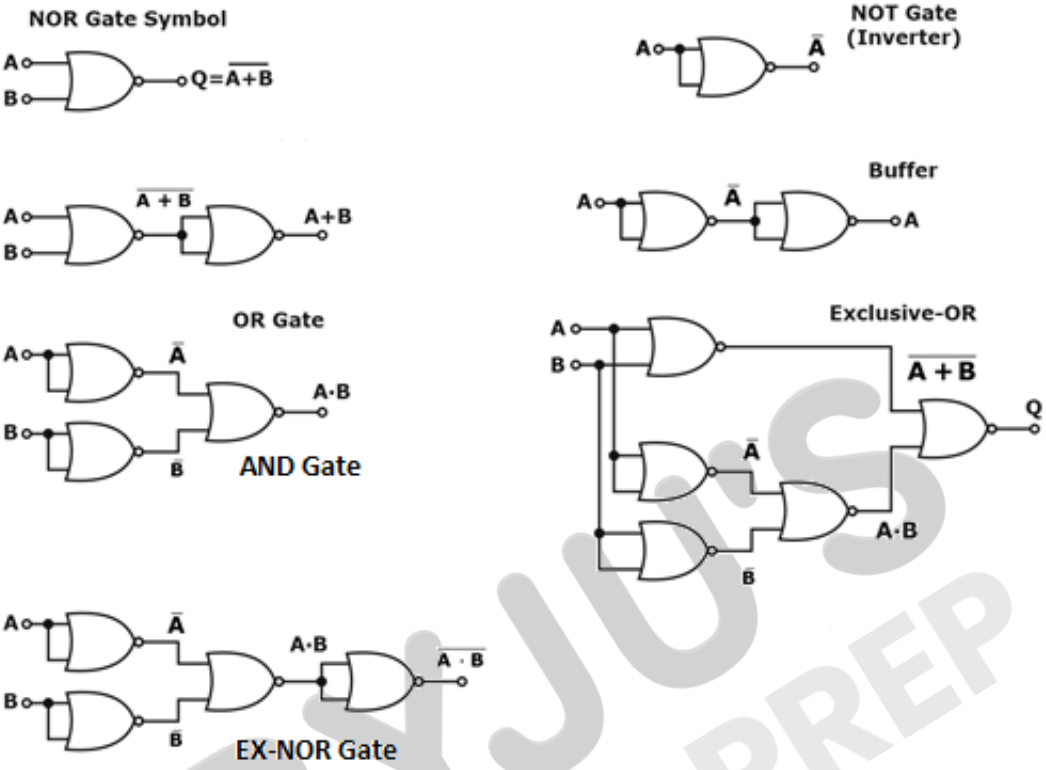


Truth Table for 2-input NOR gate

Input		Output
A	B	$Y = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

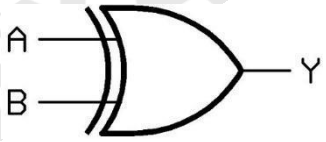
NOR gate acts as Universal Gate.

Logic Gates using only NOR Gates



2.3. EX-OR gate:

Symbol of two input XOR gate



$$A \oplus B = A\overline{B} + \overline{A}B$$

EX-OR Gate using basic Logic Gates:

- **NOR Gate:**

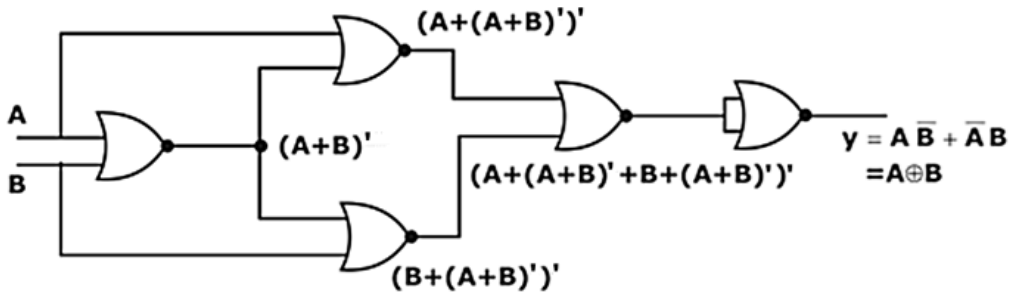
$$y = A \oplus B$$

$$= \overline{A}B + A\overline{B}$$

$$= \overline{A}B + A\overline{B} + A\overline{A} + B\overline{B}$$

$$= (A + B)(\overline{A} + \overline{B})$$

Implementing this using NOR Gates:



· **NAND Gate:**

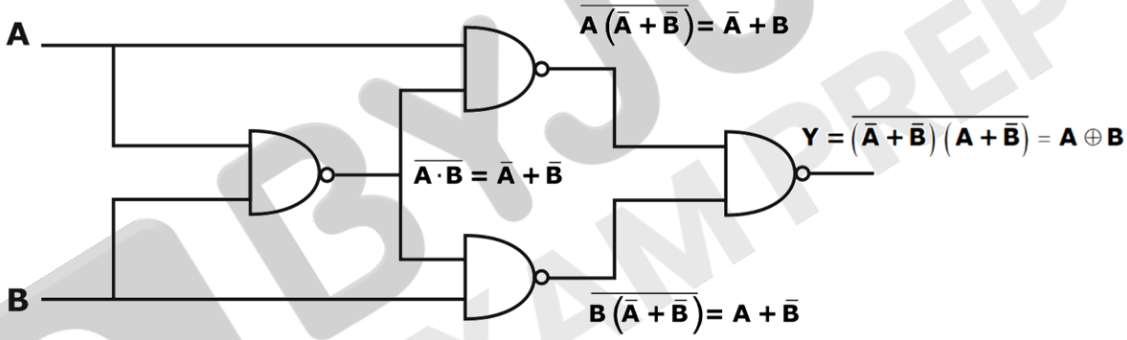
$y = A \oplus B$

$= \overline{A}B + A\overline{B}$

= ((AB' +A'B)')'

= ((AB')'(A'B)')'

Implementing this using NAND Gates:

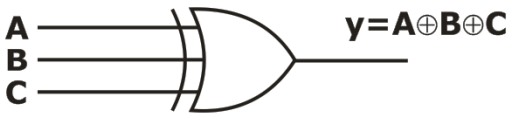


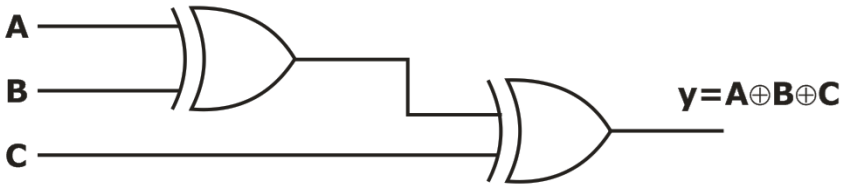
3-input Ex-OR Gate:

The Boolean function for the 3- input XOR gate is

Q = A⊕B⊕C = ABC + A'B'C + A'BC' + AB'C'.

3-Input Ex-OR gate logic symbol:





Laws:

a. Commutative Law:

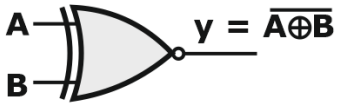
$A \oplus B = B \oplus A$

b. Associative Laws:

$A \oplus (B \oplus C) = (A \oplus B) \oplus C$

2.4. EX-NOR gate:

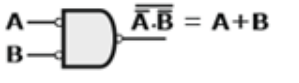
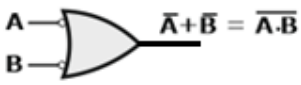
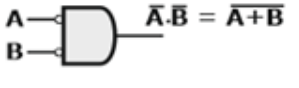
Symbol for two input X-NOR gate



2.5. EX-OR & EX-NOR Properties:

$A \odot A = 1$	$A \oplus A = 0$
$A \odot 0 = A'$	$A \oplus 0 = A$
$A \odot 1 = A$	$A \oplus 1 = A'$
$A \odot A' = 0$	$A \oplus A' = 1$
$A' \odot B' = A \odot B$	$A' \oplus B' = A \oplus B$
If $A \odot B = C$ Then $A \odot C = B$ $B \odot C = A$ $A \odot B \odot C = 1$	If $A \oplus B = C$ Then, $B \oplus C = A$ $A \oplus C = B$ $A \oplus B \oplus C = C \oplus C = 0$

3. ALTERNATE LOGIC GATE REPRESENTATION

Logic	Normal symbol	Alternate symbol
NOT		
AND		
OR		
NAND		
NOR		

4. BOOLEAN FUNCTION

4.1. LAWS OF BOOLEAN ALGEBRA

4.1.1. Commutative Law:

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

4.1.2. Associative Law:

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

4.1.3. Distributive Law:

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

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

4.1.4. Idempotence Law:

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

4.1.5. Absorption Law:

$$A + AB = A(1 + B) = A$$
$$A(A + B) = A$$

4.1.6. Consensus theorem:

$$AB + \bar{A}C + BC = AB + \bar{A}C$$

$$(A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C)$$

4.1.7. Involution Law:

$$\bar{\bar{A}} = (A')' = A$$

4.1.8. De-Morgan's theorem:

I. $\overline{A.B} = \bar{A} + \bar{B}$

II. $\overline{A + B} = \bar{A}.\bar{B}$

4.1.9. Duality theorem:

Example: What is the dual of $x + \bar{x}y = x + y$?

Solution:

$$x(\bar{x} + y) = xy$$

5. REPRESENTATION OF BOOLEAN FUNCTIONS

5.1. SOP form:

$$Y = \overline{A}BC + A\overline{B} + AC$$

$$Y = A\overline{B} + B\overline{C}$$

$$Y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

5.2. POS form:

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

$$Y = (A + B + C)\left(A + B + \overline{C}\right)\left(A + \overline{B} + C\right)\left(\overline{A} + B + C\right)$$

Chapter 3: Minimization Techniques

1. CANONICAL FORMS

We can use the Boolean function in two different ways. The minterm canonical form and the maxterm canonical form are two of these methods. A Boolean function is said to be in its canonical form if it can be represented as a sum of minterms or a product of maxterms.

1.1. Literal

A Literal signifies the Boolean variables including their complements. Such as B is a boolean variable and its complements are ~B or B', which are the literals.

1.2. Minterm

Ex. F = x' y z + x y' z + x y z' + x y z

= m3 + m5 + m6 + m7

or

F (x, y, z) = Σ(3, 5, 6, 7)

1.3. Maxterm

Ex. F = (x+y+z) • (x+y+z') • (x+y'+z) • (x'+y+z)

= M0 • M1 • M2 • M4

1.4. Conversion between Canonical Forms:

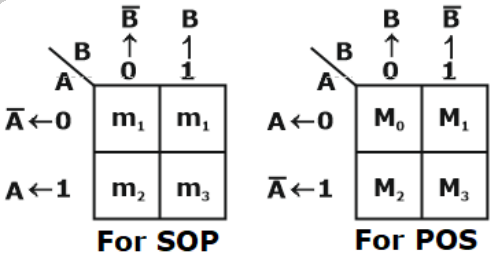
F(A,B,C) = Σ(1, 4, 5, 6, 7)

This function has a complement that can be expressed as

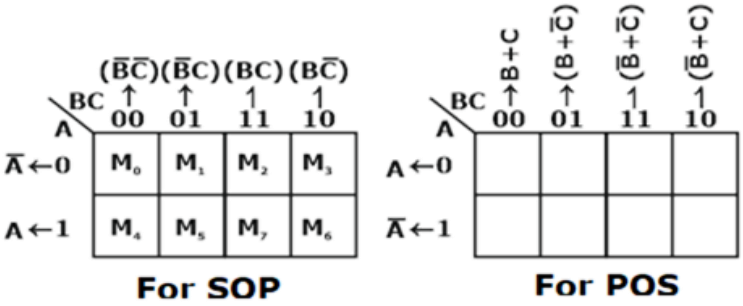
F'(A,B,C) = Σ(0, 2, 3) = $m_0 + m_2 + m_3$

2. KARNAUGH MAP (K-MAP)

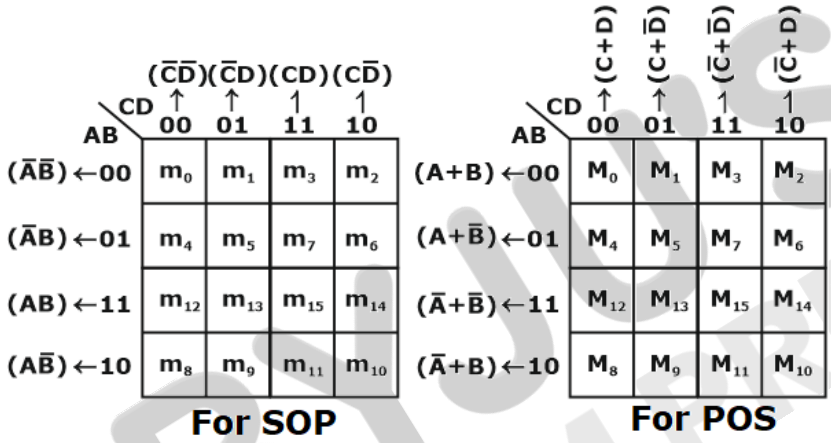
2.1. Two variable K-map:



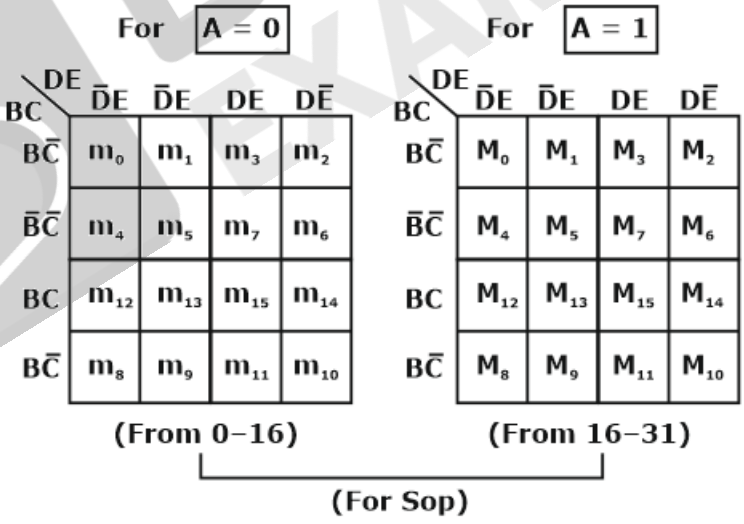
2.2. Three variable K-map:



2.3. Four Variable K-map:



2.4. Five variable K-map:

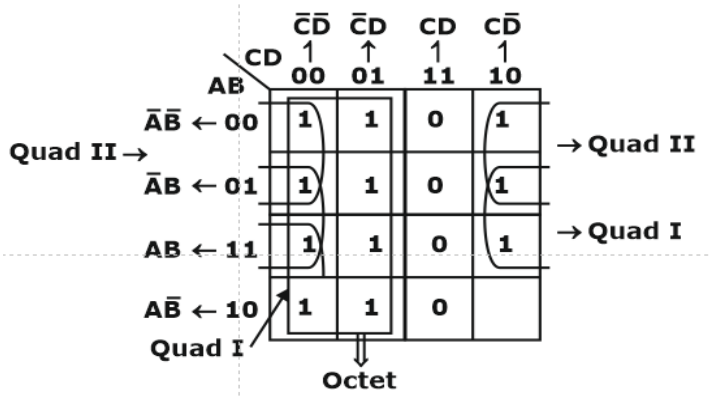


3. SIMPLIFICATION RULES

Example 3.1: Simply a four variable logic function using K-map

f(A, B, C, D) = Σm(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) also implement the simplified expression with AND-OR logic.

Solution:

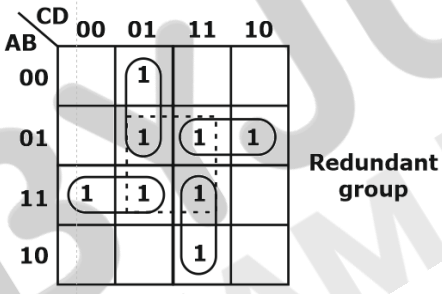


$$f = \bar{C} + \bar{A}\bar{D} + B\bar{D}$$

$$\Downarrow \quad \Downarrow \quad \Downarrow$$

$$\therefore \quad \text{octet} \quad \text{quad} - \text{II} \quad \text{quad} - \text{I}$$

4. REDUNDANT GROUP



5. DON'T CARE CONDITION

Example 5.1:

Simply the given equation in part (i) and (ii)

(i) In terms of SOP. and don't care conditions

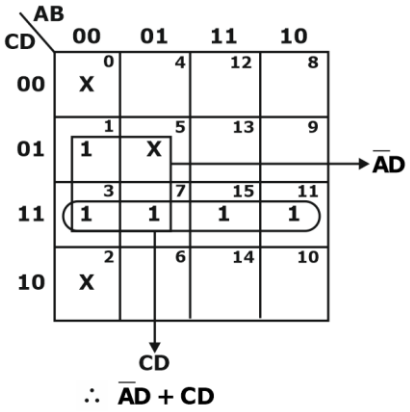
f(A, B, C, D) = Σm(1, 3, 7, 11, 15) + d(0, 2, 5)

(ii) In terms of POS and don't care about conditions.

f(A, B, C, D) = ПM(4, 5, 6, 7, 8, 12) + d(1, 2, 3, 9, 11, 14)

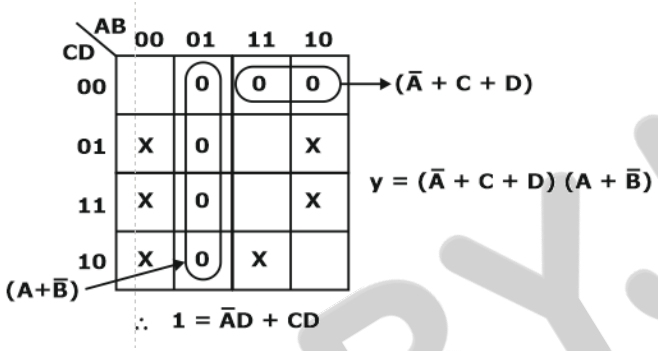
Solution:

(i)



$\therefore\ \bar{A}D + CD$

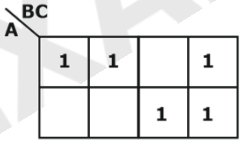
(ii)



$\therefore\ 1 = \bar{A}D + CD$

6. IMPLICANTS, PRIME IMPLICANTS AND ESSENTIAL PRIME IMPLICANTS

Example 6.1: For the given K-map, find Implicant, PI and EPI



Solution:

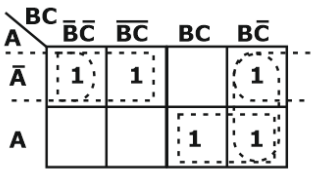
Here, total no of 1's = 5

$\therefore$ Implicants = 5

$\therefore$ Implicants = $(\bar{A}\bar{B}\bar{C}).(\bar{A}\bar{B}C)(ABC)(\bar{A}B\bar{C})(AB\bar{C})$

and prime implicant = $\bar{A}\bar{B},\ \bar{A}\bar{C},\ AB,\ B\bar{C}$

and EPI = $\bar{A}\bar{B},\ AB$



PI = 4

EPI = 2

Chapter 4: Combinational Circuit

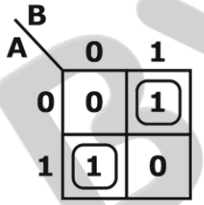
1. Half Adder:

Inputs		Outputs	
A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

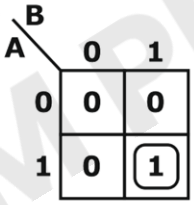
Table 1: Truth Table of Half Adder

1.1. K–map simplification for Carry and Sum:

Boolean expressions for the sum (S) and carry (C) outputs from K – maps:



K-map for sum output



K-map for carry output

Sum, $S = \bar{A}B + A\bar{B} = (A \oplus B)$

Carry, $C = AB$

1.2. Logic Diagram:

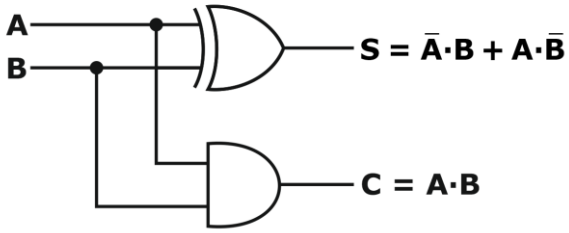


Figure 3: Logic Diagram of Half Adder

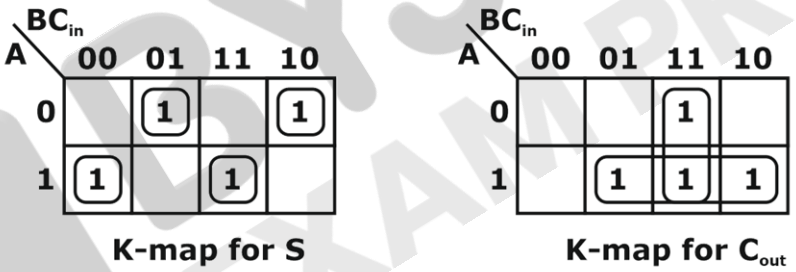
2. Full Adder:

2.1 The Truth Table for Full Adder is given as:

Inputs			Outputs	
A	**B**	**C_{in}**	**Sum (S)**	**Carry C_{out}**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 2: Truth Table for Full Adder

2.2 K – map Simplification for Carry and Sum:



$$\text{Sum,} \quad S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + ABC_{in} + A\bar{B}\bar{C}_{in} = C_{in}\left(\bar{A}\bar{B} + AB\right) + \bar{C}_{in}\left(\bar{A}B + A\bar{B}\right)$$

$$= C_{in}\left(A \odot B\right) + C_{in}\left(A \oplus B\right)$$

$$= C_{in}\left(\overline{A \oplus B}\right) + C_{in}\left(A \oplus B\right)$$

$$\text{Sum,} \quad S = C_{in} \oplus A \oplus B$$

$$\text{Carry,} \quad C_{out} = AB + AC_{in} + BC_{in}$$

2.3 Logic Diagram:

We can realize logic diagram of a full adder using gates as shown in below figure:

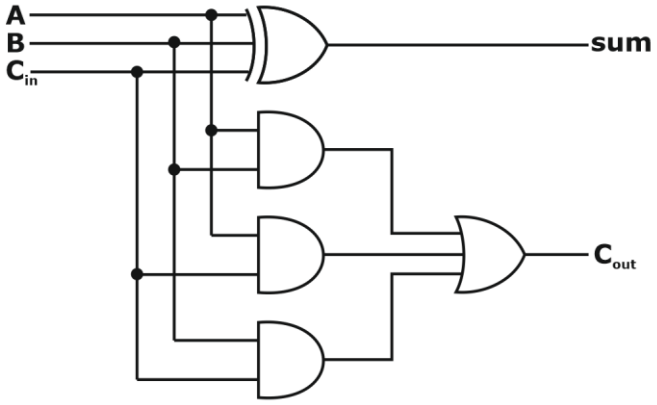


Figure 5: Logic Diagram of Full Adder

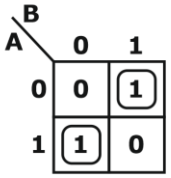
3. SUBTRACTORS

3.1. The truth table of half – subtractor, where A, B are the inputs, and difference (D) and borrow (B) are the outputs.

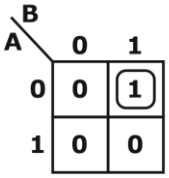
Inputs		Outputs	
A	**B**	**D**	**B_{out}**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Table 3: Truth table of Half – Subtractor,

3.2. K – map Simplification for Difference and Borrow:



K-map for difference output



K-map for borrow output

Difference, $D = \bar{A}B + A\bar{B} = A \oplus B$

Borrow, $B_{out} = \bar{A}B$

3.3. Logic Diagram:

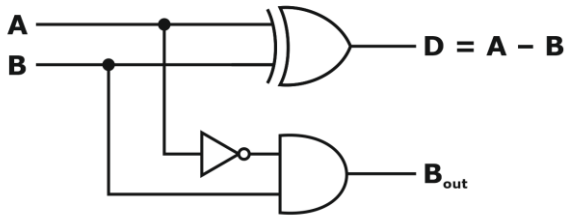
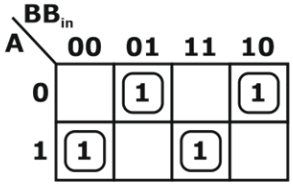
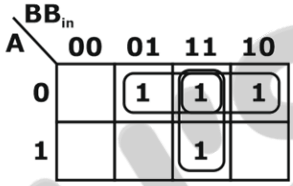


Figure 9: Logic Diagram of a Half subtractor

3.4. Full Subtractor:



K-map for difference output



K-map for borrow output

Difference,

$$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$$

$$= B_{in}\left(AB + \bar{A}\bar{B}\right) + \bar{B}_{in}\left(A\bar{B} + \bar{A}B\right)$$

$$= B_{in}\left(\overline{A \oplus B}\right) + B_{in}\left(A \oplus B\right)$$

$$D = A \oplus B \oplus B_{in}$$

Borrow,

$$B_{in} = \bar{A}B + \bar{A}B_{in} + BB_{in}$$

4. CARRY LOOK AHEAD ADDER

$G = A \cdot B$

Consider the present bit as the n^{th}, then

$G_n = A_n \cdot B_n$

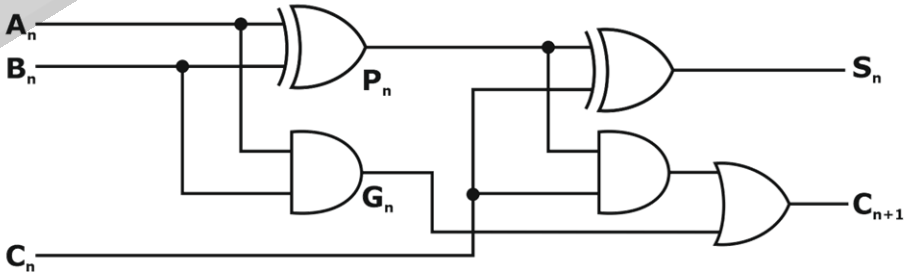


Figure 13: Carry Look – ahead Generator Circuit

4.1. Carry Propagation:

A carry is propagated if any one of the two input bits A or B is 1. If both A and B are 0, a carry will never be propagated. On the other hand, if both A and B are 1, then will not

propagate the carry but will generate the carry. Let P as the carry – propagation function, then

$P_n = A_n \oplus B_n$

4.2. Look ahead Expressions:

Let n^{th} bit adder, the sum (S) and the carry out (C) for the n^{th} bit may be expressed in terms of the carry generation function (G) and the carry propagation function (P) as

$S_n = P_n \oplus C_n$

$C_{n+1} = G_n + P_n \cdot C_n$

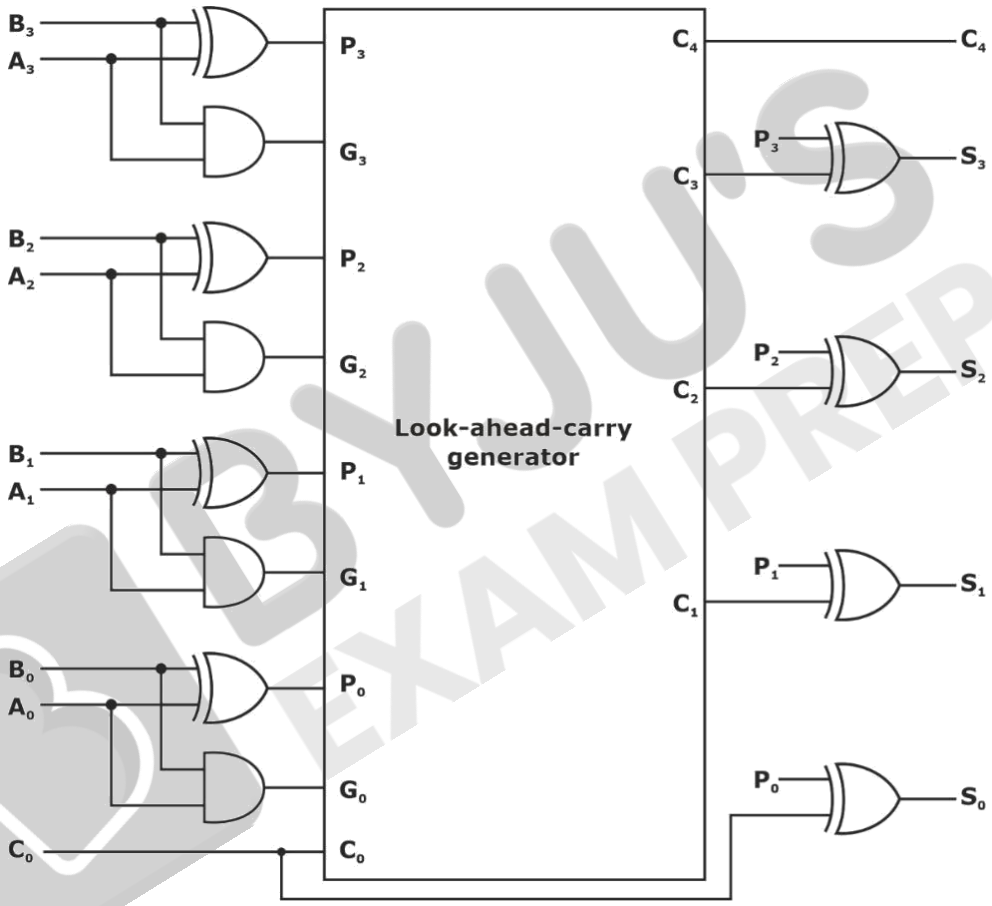


Figure 14: 4-bit Full Adder with a look Ahead Carry Generator

5. COMPARATOR

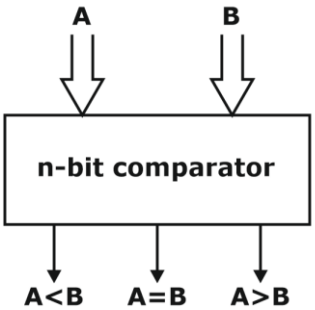


Figure 18: Block diagram of digital comparator

5.1. 1-bit Magnitude Comparator:

The 1-bit comparator is a combinational logic circuit with two inputs A and B and three outputs namely A < B, A= B and A < B.

Inputs		Outputs		
A	**B**	**X (A < B)**	**Y (A = B)**	**Z (A > B)**
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Table 5: Truth Table of a 1-bit Comparator:

5.1.1 Design of 1-bit Magnitude Comparator

We can write the expressions for the three outputs as under:

$$\text{For } (A < B), \quad X = \bar{A}_0 B_0$$

$$\text{For } (A = B), \quad Y = \bar{A}_0 \bar{B}_0 + A_0 B_0 = \overline{A_0 \oplus B_0}$$

$$\text{For } (A > B), \quad Z = A_0 \bar{B}_0$$

5.1.2 Logic Diagram of 1-bit Comparator:

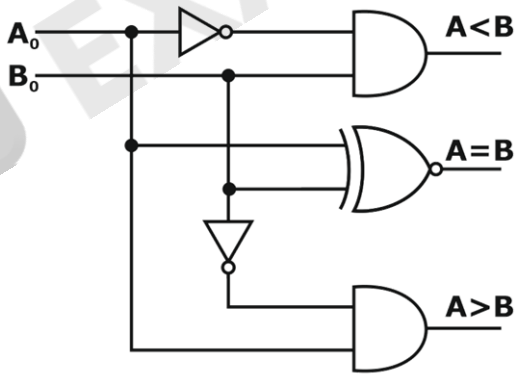
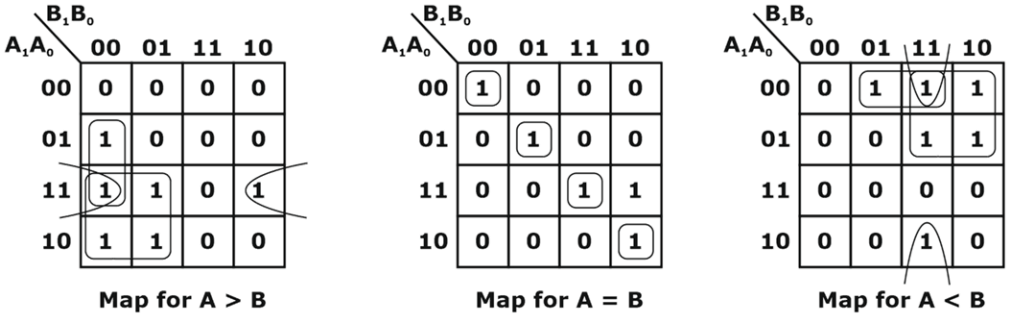


Figure 19: Logic Diagram of 1-bit Comparator

5.1.3 Design of 2-bit Magnitude Comparator:

The K-maps for the three outputs are shown as below:



The simplified expressions can be obtained from the K-map as

For A < B, $X = \overline{A}_1\overline{A}_0B_0 + \overline{A}_1B_1 + \overline{A}_0B_1B_0$

For A = B, $Y = (A_0 \odot B_0)\,(A_1 \odot B_1)$

For A > B, $Z = A_0\overline{B}_1\overline{B}_0 + A_1A_0\overline{B}_0 + A_1\overline{B}_1$

6. MULTIPLEXER

6.1. 2 x 1 MUX:

A 2 to 1 multiplexer has 2 inputs. Since 2 = 2^1, this multiplexer will have one control (select) line. It has two data inputs I_0 and I_1, one select input S, and one output.

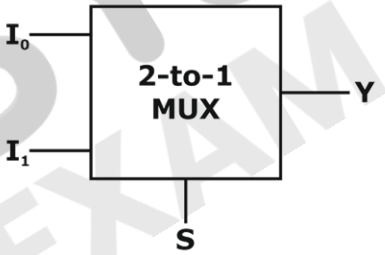


Figure 23: Schematic block diagram of 2:1 Multiplexer

6.1.1 The truth table of this MUX is given below,

Select Line (S)	Output Y
0	I_0
1	I_1

Figure 24

Thus, the SOP expression for the output Y is,

$Y = I_0\overline{S}_0 + I_1S_0$

6.1.2 Realization of a 2:1 MUX using Logic Gates:

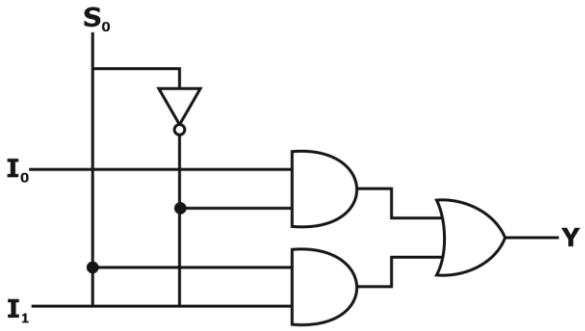


Figure 25: Logic Diagram of a 2 x 1 Multiplexer

6.2. 4 x 1 MUX:

A 4-to-1 multiplexer has 4 inputs and two select lines, where I_0 to I_3 are the four inputs to the multiplexer, and S_0 and S_1 are the select lines.

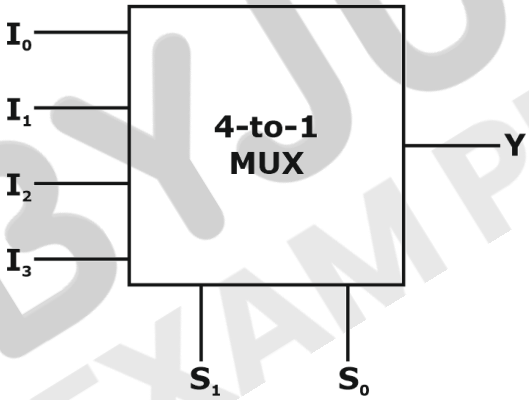


Figure 26: Schematic block diagram of 4 x 1 MUX

6.2.1 Truth Table of a 4-to-1 Multiplexer

Select Inputs		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Output Y for a 4-input multiplexer is

$$Y = I_0\bar{S}_1\bar{S}_0 + I_1\bar{S}_1S_0 + I_2S_1\bar{S}_0 + I_3S_1S_0$$

6.2.2 Realization of a 4:1 MUX Using Logic Gates:

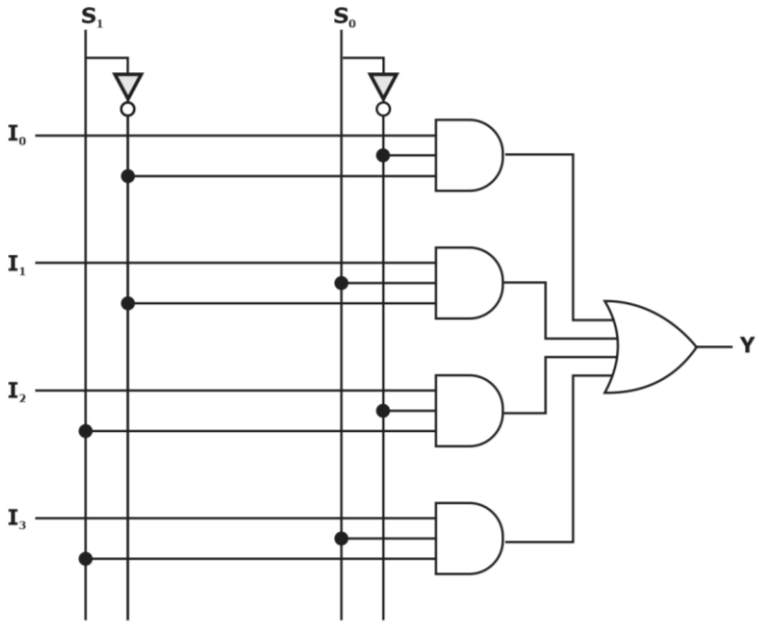


Figure 27: Logic Diagram of 4 x 1 MUX

Example:

Minimum no. of 4 × 1 MUX required to implement 64 × 1 MUX.

Solution:

Using the above formula, we can obtain the same.

$$\frac{64}{4} = 16$$

$$\frac{16}{4} = 4$$

$$\frac{4}{4} = 1$$ (till we obtain 1 count of MUX)

Hence, total number of 4 : 1 MUX are required to implement 64 : 1 MUX = 16 + 4 + 1 = 21.

7. IMPLEMENTATION OF BOOLEAN FUNCTION USING MUX

Example: Implement the following Boolean function using 8:1 MUX

F (A, B, C, D) = $\sum m(0,1,2,4,6,9,12,14)$

Solution:

Select Lines are B, C and D

Total Number of variables, n = 4 (A, B, C, D)

Number of Select Lines, n-1 = 3 (B, C, D)

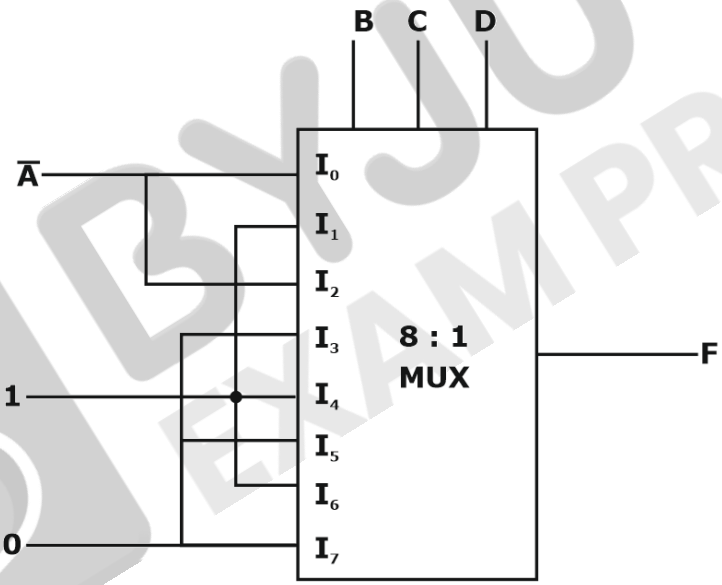
The given function has 4 variables, so 16 possible minterms (0-15) are entered in the implementation table.

All the minterms are divided into 2 groups:

- The first group (0-7) minterms are entered in the first row (Variable A=0)
- The second group (8-15) minterms are entered in the second row (Variables A=1)

Circle the minterms number as per function, which are to be implemented (here $0,1,2,4,6,9,12,14$)

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	(0)	(1)	(2)	3	(4)	5	(6)	7
A	8	(9)	10	11	(12)	13	(14)	15
	$\bar{A}$	1	$\bar{A}$	0	1	0	1	0



8. DEMULTIPLEXER

8.1. 1 x 2 Demultiplexer:

A 1 to 2 demultiplexer has one input and two outputs. Since $2 = 2^1$, it requires only one control (select) line.

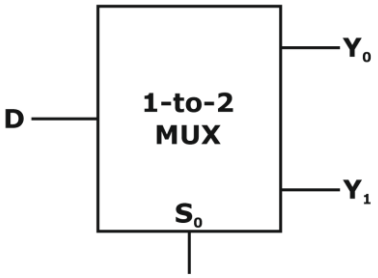


Figure 30: Logic Diagram of 1 x 2 Demux

8.1.1 Truth table of a 1-to-2 demultiplexer

Example 11:

The network shown below implements which gate?

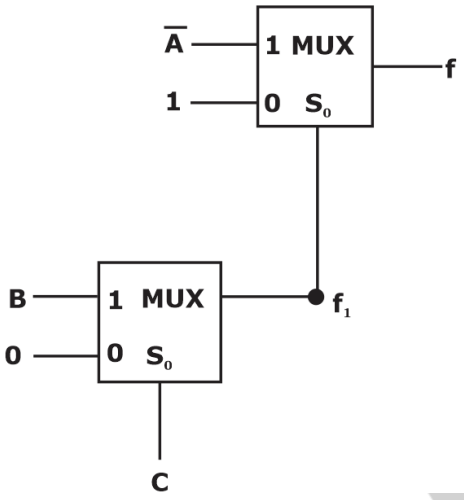


Figure 33

Solution: The function f of the network is given by

$$f = 1 \cdot \overline{S}_0 + \overline{A} \cdot S_0$$

Output of MUX 1 is given to S_0 of the MUX-2, i.e.

$$f_1 = 0 \cdot \overline{C} + B \cdot C$$
$$f_1 = B \cdot C$$

Here, we get the output of the network as.

$$f = 1 \cdot \overline{f_1} + \overline{A} \cdot f_1$$
$$= \overline{BC} + \overline{A}BC$$
$$= (\overline{BC} + \overline{A})(\overline{BC} + BC)$$
$$= (\overline{BC} + \overline{A})$$
$$= (\overline{A} + \overline{B} + \overline{C})$$
$$= \overline{ABC}$$

i.e f is the output of a 3 input (A, B, C) NAND gate.

9. COMPARISON BETWEEN MULTIPLEXER AND DEMULTIPLEXER

S.No.	Parameter of comparison	Multiplexer	Demultiplexer
1.	Type of logic circuit	Combinational	Combinational
2.	Number of data inputs	m	1
3.	Number of select inputs	n	N
4.	Number of data output	1	M

S.No.	Parameter of comparison	Multiplexer	Demultiplexer
5.	Relation between input/output lines and select lines	$m = 2^n$	$M = 2^N$
6.	Operation principle	Many to 1 or as data selector	1 to many or data distributor

Table 9: Comparison between Multiplexer and Demultiplexer

9.1. 2 to 4 Line Decoder:

Consider a 2 to 4-line decoder, where A and B are two inputs whereas Y_0 through Y_3 are the four outputs.

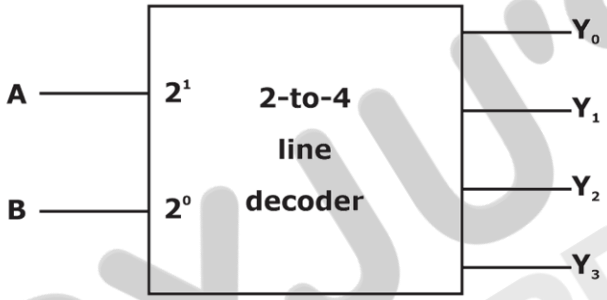


Figure 35: Block Diagram of a 2 to 4 Line Decoder

9.1.1 Truth Table of a 2 to 4 Line Decoder:

Inputs		Outputs			
A	B	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Table 10: Truth Table of a 2 to 4 Line Decoder:

The Boolean expressions for the four outputs is given as:

$Y_0 = \overline{A}\,\overline{B}$ and $Y_1 = \overline{A}B$

$Y_2 = A\overline{B}$ and $Y_3 = AB$

9.1.2 Realization of a 2 to 4 Line Decoder using Logic Gates:

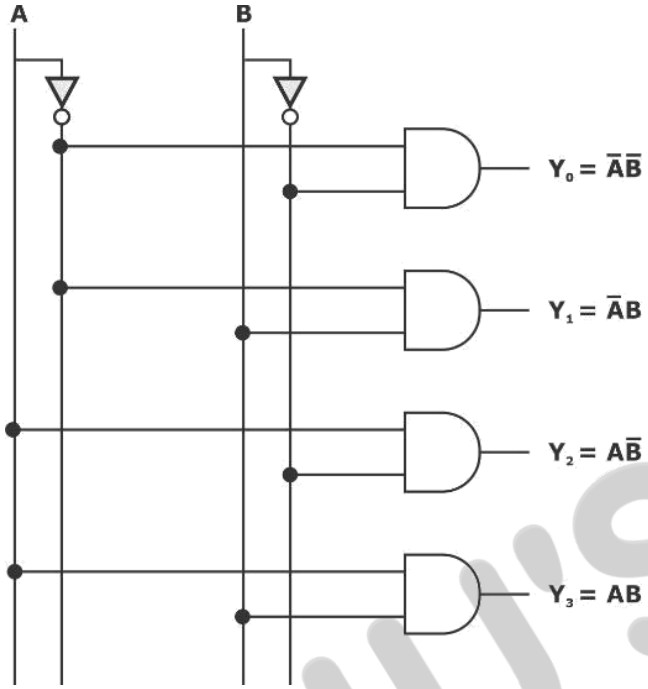


Figure 36: Logic Diagram of a 2 to 4 Line Decoder

Example 13:

The circuit shown below represents a Boolean expression. How many minterms are there in the Boolean expression?

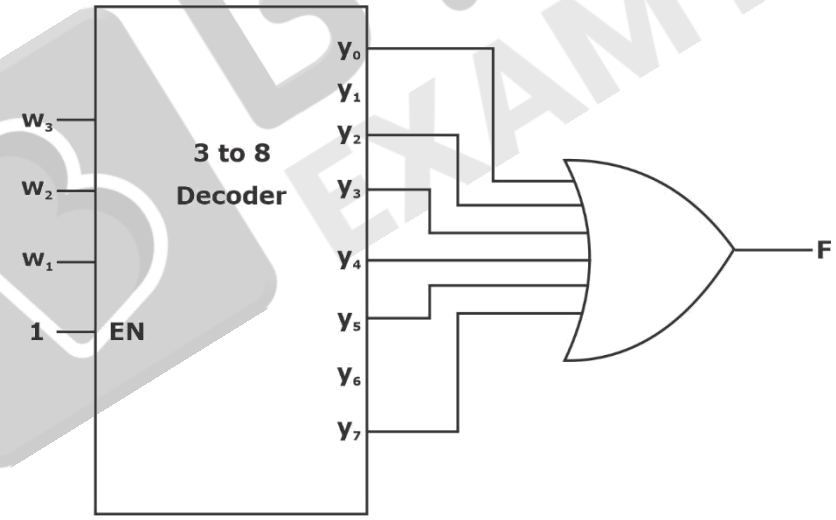


Figure 38

Solution: In the given logic circuit, the enable input (EN) is 1. So, 3 to 8 decoder is activated. So, 3-to-8 decoder is activated.

Therefore, the output function f is given by

$F = y_0 + y_2 + y_3 + y_4 + y_5 + y_7$

Or

$F(w_1, w_2, w_3)$

$= \bar{w}_1 \bar{w}_2 \bar{w}_3 + \bar{w}_1 w_2 \bar{w}_3 + \bar{w}_1 w_2 w_3 +$

$w_1 \bar{w}_2 \bar{w}_3 + w_1 \bar{w}_2 w_3 + w_1 w_2 w_3$

$= 000 + 010 + 011 + 100 + 101 + 111$

$= \Sigma m(0,2,3,4,5,7)$

i.e. Boolean expression with 6 min-terms.

10. ENCODERS

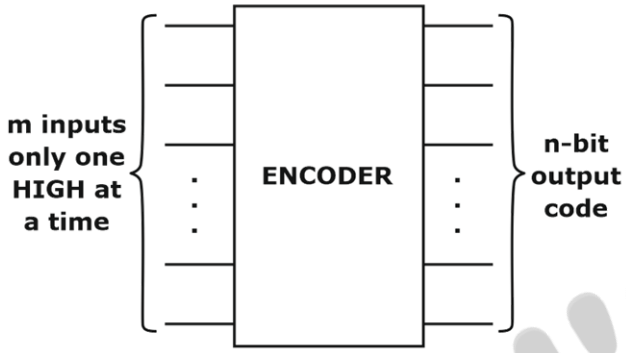


Figure 39: Block Diagram of Encoder

10.1. Octal to Binary Encoder:

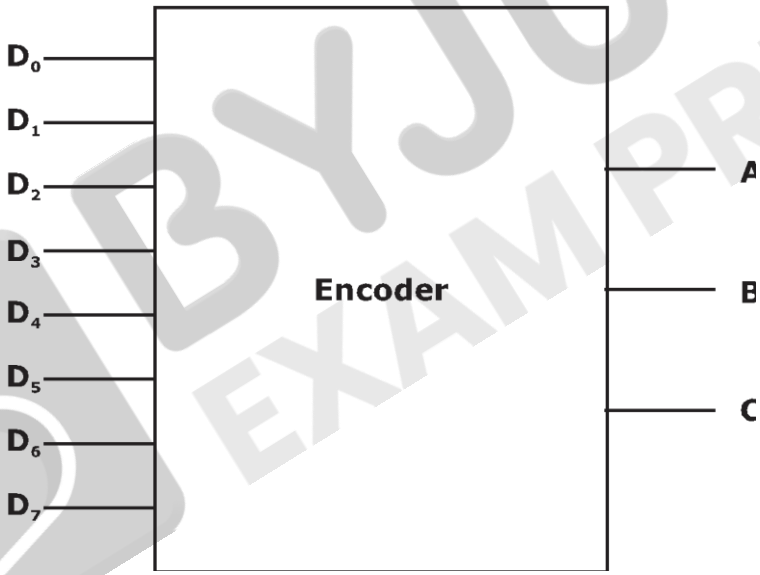


Figure 40: Octal to Binary Encoder

10.1.1 Truth Table of an Octal to Binary Encoder:

The logical expressions for the outputs as follows:

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$

11. PRIORITY ENCODER

11.1 Truth Table of a Four Input Priority Encoder: (Taking LSB as priority)

Inputs				Outputs	
D_0	D_1	D_2	D_3	A	B
0	0	0	0	X	X
1	0	0	0	0	0
X	1	0	0	0	1
X	X	1	0	1	0
X	X	X	1	1	1

Table 13: Truth Table of a Four Input Priority Encoder

According to the truth table, the higher the subscript number, the higher the priority of the input.

The X's are don't care conditions indicating that the binary values they represent may be equal to 0 or 1.

12. CODE CONVERTERS

A code converter is a combinational logic circuit which accepts the input information in one binary code, converts it and produces an output into another binary code.

12.1 The truth table for 4-bit Binary and its Equivalent BCD:

Decimal	Binary Input				BCD Output				
	A	B	C	D	B_4	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	0	1	0
3	0	0	1	1	0	0	0	1	1
4	0	1	0	0	0	0	1	0	0
5	0	1	0	1	0	0	1	0	1
6	0	1	1	0	0	0	1	1	0
7	0	1	1	1	0	0	1	1	1
8	1	0	0	0	0	1	0	0	0
9	1	0	0	1	0	1	0	0	1
10	1	0	1	0	1	0	0	0	0
11	1	0	1	1	1	0	0	1	0

12	1	1	0	1	1	0	0	1	0
13	1	1	0	1	1	0	0	1	1
14	1	1	1	0	1	0	1	0	0
15	1	1	1	1	1	0	1	0	1

Table 14: Truth table for 4-bit Binary and its Equivalent BCD

The minimized expression of outputs are as follows:

$B_4 = AB + AC$

$B_1 = \overline{A}C + AB\overline{C}$

$B_2 = \overline{A}B + BC$

$B_3 = A\overline{B}\overline{C}$

$B_0 = D$

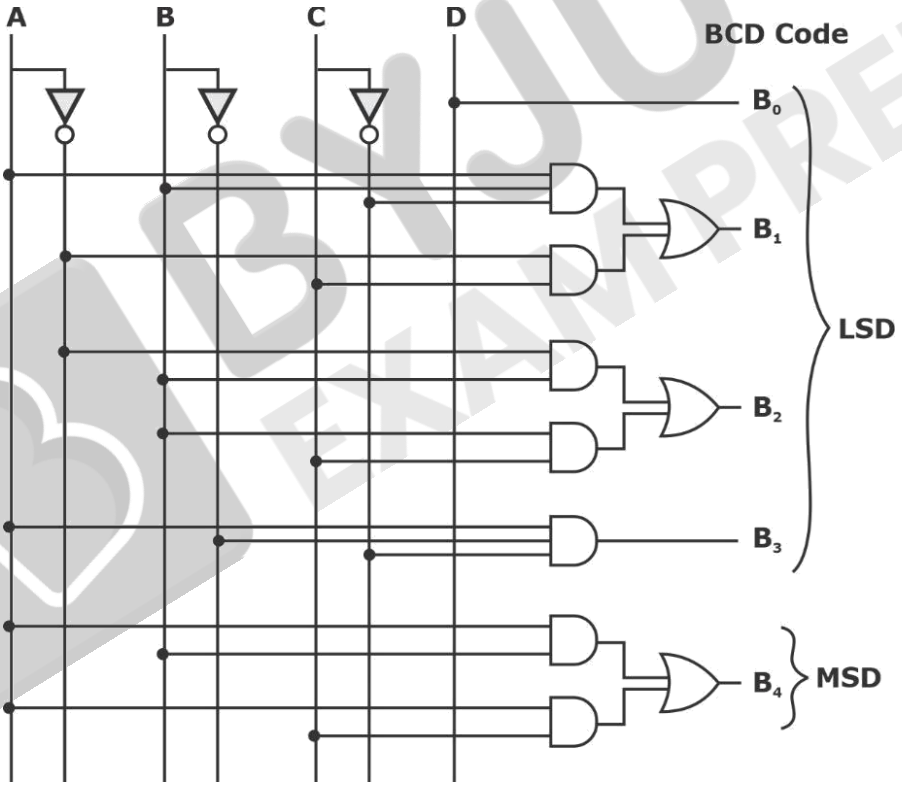


Figure 44: Logic Diagram of a Binary-to-BCD Code Converter

13. PARITY GENERATOR

13.1. Even Parity Generator:

The even parity generator is a combinational logic circuit that generates the parity bit such that the number of 1's in the message becomes even. The parity bit is '1' if there are odd number of 1's in the data stream and the parity bit is '0' if there are even number of 1's in the data stream.

13.1.1 Truth table for 4-bit data with Even Parity:

4-bit data				Even Parity
A	B	C	D	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Table 15: Truth table for 4-bit data with Even Parity

The minimized expression for even parity generator is

$P = A \oplus B \oplus C \oplus D$

The logic diagram for the even parity generator is given as

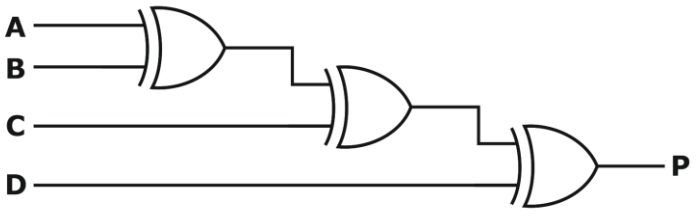


Figure 45: Logic diagram of even parity generator

13.2. Odd Parity Generator:

The odd parity generator is a combinational logic that generates the parity bit such that the number of 1's in the message becomes odd. The parity bit is '0' for odd number of 1's and '1' for even number of 1's in the bit stream.

13.2.1 Truth table for 4-bit data with Odd Parity:

4-bit data				Odd Parity
A	B	C	D	P
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Table 16: Truth table for 4-bit data with Odd Parity

The minimized expression for odd parity generator is

$P = (A \oplus C) \odot (B \oplus D)$

The logic diagram of odd parity generator is given as

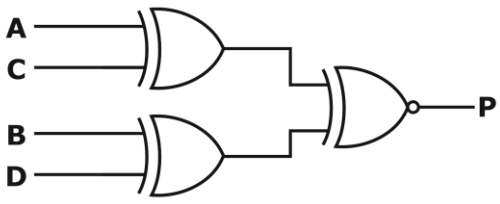


Figure 46: Logic Diagram of Odd Parity Generator

Chapter 5: Sequential Circuits

1. SEQUENTIAL LOGIC CIRCUITS

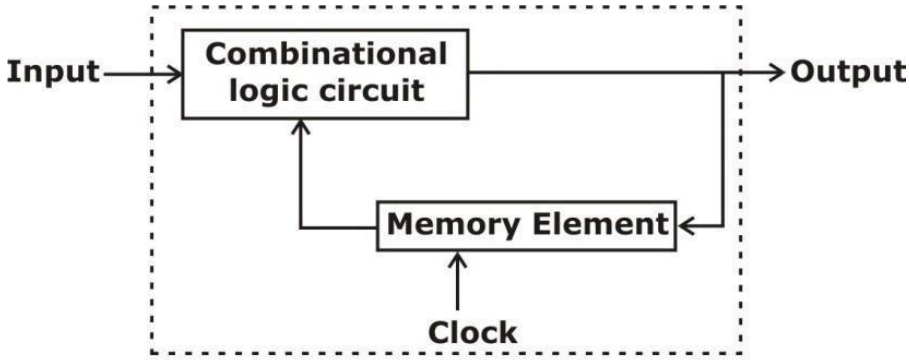
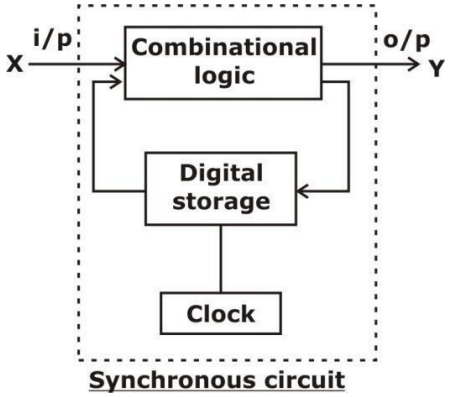
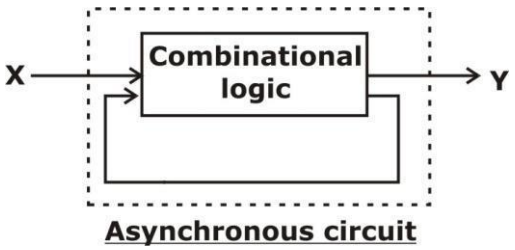


Figure 1: General Block diagram of Sequential Logic Circuit

2. DIFFERENCE BETWEEN SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS

S. No.	Synchronous Sequential Circuits	Asynchronous Sequential Circuits
1.	In synchronous circuits, the change in input signals can affect memory elements upon activation of the clock signal.	In asynchronous circuits, change in input signals can affect memory elements at any instant of time.
2.	In synchronous circuits, memory elements are clocked FF's.	In asynchronous circuits, memory elements are either unclocked FF's or time delay elements.
3.	The maximum operating speed of the clock depends on time delays involved.	Since the clock is not present, asynchronous circuits can operate faster than synchronous circuits.
4.	They are easier to design.	More difficult to design.
5.		

3. LATCHES

3.1. Difference between Latches and Flip-flops:

S.No.	Latch	Flip-flop
1.	A latch is an electronic sequential logic circuit used to store information in an asynchronous arrangement.	A flip-flop is an electronic sequential logic circuit used to store information in a synchronous arrangement. It has two stable states and maintains its states for an indefinite period until a clock pulse is applied.
2.	One latch can store one-bit information, but output state changes only in response to data input.	One flip-flop can store one-bit data, but output state changes with clock pulse only.
3.	Latch is an asynchronous device and it has no clock input.	Flip-flop has a clock input and its output is synchronised with the clock pulse.
4.	Latch holds a bit value and it remains constant until new inputs force it to change.	Flip-flop holds a bit value and it remains constant until a clock pulse is received.
5.	Latches are level-sensitive, and the output tracks the input when the level is high. Therefore, as long as the level is logic level 1, the output can change if the input changes.	Flip-flops are edge sensitive. They can store the input only when there is either a rising or falling edge of the clock.

3.2. SR Latch:

For the SR latch (S stands for set and R for reset). The logic circuit for SR latch is shown in figure below:

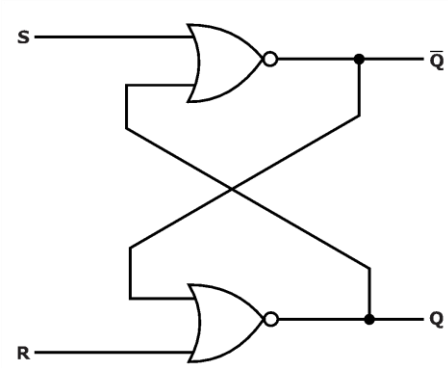
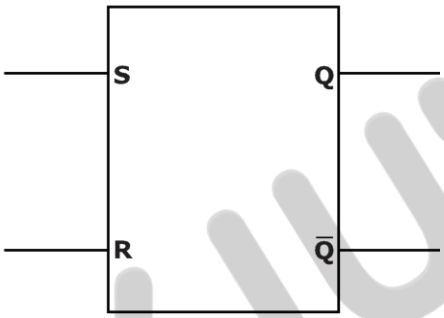


Figure 2: Logic circuit of SR latch with NOR Gates

The symbol for SR Latch is:

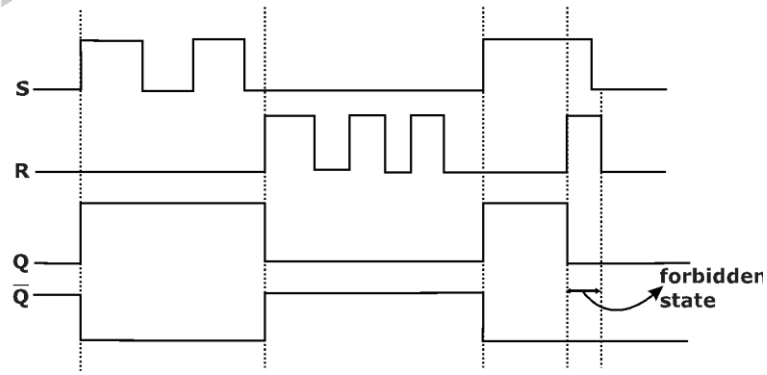


The truth table for SR latch is

S	R	Q_{n+1}	State
0	0	Q_n	Halt
0	1	1	Set
1	0	0	Reset
1	1	x	Forbidden

However, the forbidden state (S = R = 1) is considered a don't care state.

Consider a Timing diagram for SR latch



3.3. $\bar{S}\bar{R}$ Latch:

An $\bar{S}\bar{R}$ latch can be implemented using NAND gates, as shown in figure below

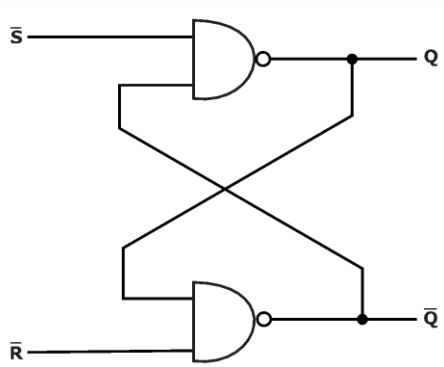
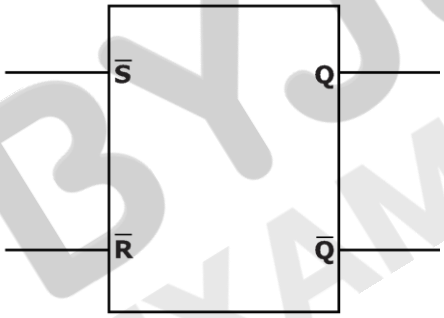


Figure 3: Logic circuit for $\bar{S}\bar{R}$ Latch with NAND Gates

The $\bar{S}\bar{R}$ latch is said to be set-dominant 1,

The symbol for $\bar{S}\bar{R}$ latch is shown below:



The truth table for SR latch using NAND Gates is given as:

S	R	Q_{n+1}	State
0	0	x	Forbidden
0	1	1	Set
1	0	0	Reset
1	1	Q_n	Halt

4. FLIP-FLOPS

Flip-flops are synchronous bistable devices also known as bistable multivibrator. Its output change its state only at a verified paint (i.e. leading or trailing edge) on the triggering input called the clock (CLK), i.e. changes in the output occur in synchronization with the clock. Flip-flops are edge-triggered or edge-sensitive whereas gated latches are level-sensitive.

4.1. Edge-triggered flip-flop:

An edge-triggered flip-flop changes its state either at the positive edge (rising edge) or at negative edge (falling edge) of the check pulse.

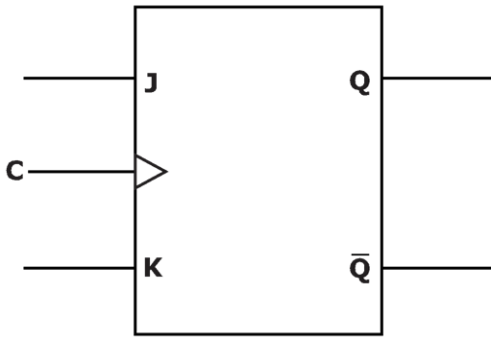


Figure 7: Positive edge triggered flip-flop.

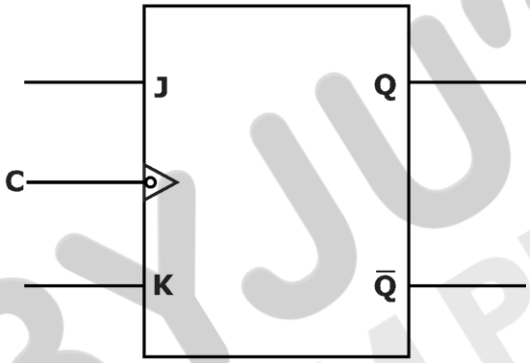


Figure 8: Negative edge-triggered flip-flop.

4.2. Types of Triggering

Following are the two possible types of triggering that are used in sequential circuits.

- Level triggering
- Edge triggering

4.3. S-R flip flop:

The logic circuit diagram, symbol and truth is given as

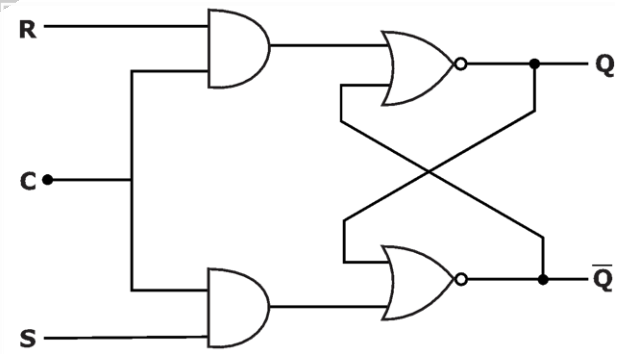


Figure 4: Logic circuit of clocked SR latch.

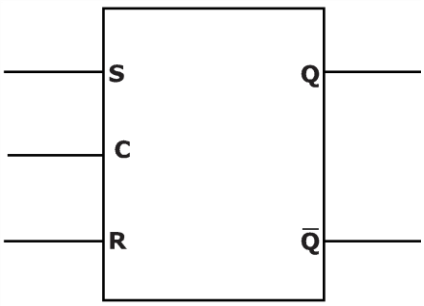


Figure 5: Symbol for clocked SR latch

Truth Table of S-R Flip Flop

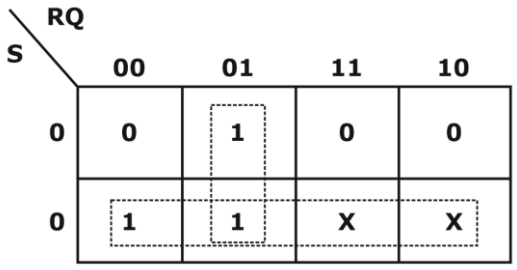
Clock	S	R	Q_{n+1}	State
0	x	x	Q_n	
1	0	0	Q_n	Hold
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	x	Invalid

Characteristic Table of S-R Flip Flop:

C	S	R	Q_n	Q_{n+1}
0	X	X	X	Q
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	1	X
1	1	1	1	X

The characteristic equation for SR flip-flop is given as:

$Q_{n+1}(S, R, Q_n) = \sum m(1,4,5) + d(6,7)$



$Q^+ = S + \bar{R}Q$ or $Q_{n+1} = S + \bar{R}Q_n$

4.4. Basic JK flip-flop:

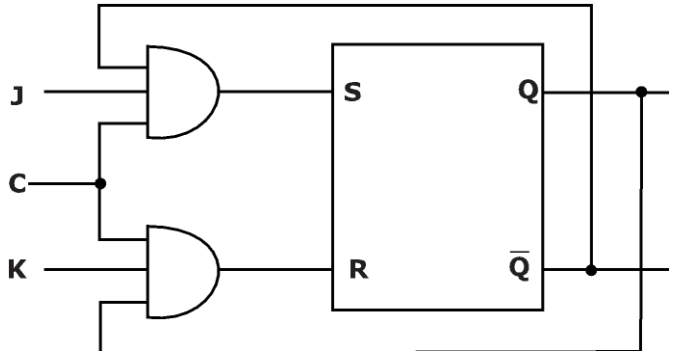


Figure 9: Logic circuit diagram of clocked JK flip-flop.

The state table for the JK flip-flop is given as

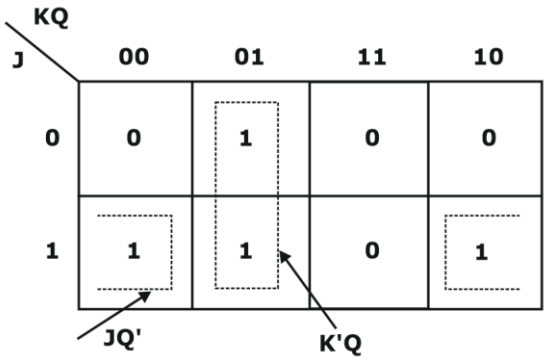
C	J	K	Q_n	Q_{n+1}	State
0	X	X	X	Q_n	Q_n
1	0	0	0	0	Hold
1	0	0	1	1	Hold
1	0	1	0	0	Reset
1	0	1	1	0	Reset
1	1	0	0	1	Set
1	1	0	1	1	Set
1	1	1	0	1	Toggle
1	1	1	1	0	Toggle

Hence, the truth table becomes,

C	J	K	Q^+	$\overline{Q}^+$	
0	X	X	Q	$\overline{Q}$	} ⇒ No change State
1	0	0	Q	$\overline{Q}$	
1	0	1	0	1	⇒ Reset
1	1	0	1	0	⇒ Set
1	1	1	$\overline{Q}$	Q	⇒ Toggle

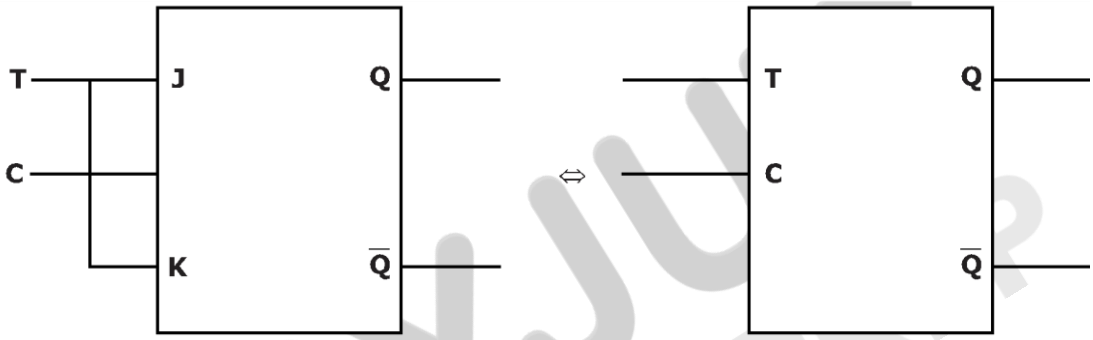
The characteristic equation of the JK flip-flop:

$$Q_{n+1}(J, K, Q_n) = \sum m(1, 4, 5, 6)$$



$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$ or $Q^+ = J\bar{Q} + \bar{K}Q$

4.5. T-flip-flop:



The state or characteristic table for T flip-flop is

C	T	Q	Q^+
O	X	X	Q
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

As J = K = T, we obtain the characteristic equation as

$Q^+ = T \cdot \bar{Q} \cdot C + (\bar{T} + \bar{C}) \cdot Q$

If C = 1, the characteristic the equation is reduced to

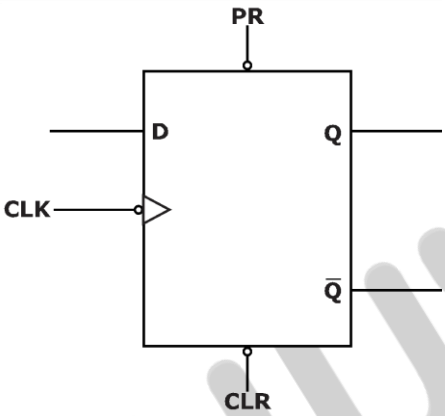
$Q^+ = T \oplus Q$

If C = 0, $Q^+ = Q$

Hence, the truth table of the T-flip flop is given as

C	T	Q^+	$\overline{Q^+}$	
0	X	Q	$\overline{Q}$	⇒ No change state
1	0	Q	$\overline{Q}$	
1	1	$\overline{Q}$	Q	⇒ Toggle

4.6. D Flip-Flop:



The truth table for D-flip-flop is

Input	Output
D_n	Q_{n+1}
0	0
1	1

The characteristic equation of D-flip-flop is:

$Q_{n+1} = D$

The excitation table for flip-flops:

Present state	Next state	SR Flip-flop		JK Flip-flop		T Flip-flop	D Flip-flop
		S	R	J	K	T	D
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	1	0
1	1	X	0	X	0	0	1

5. OPERATING CHARACTERISTICS OF FLIP-FLOPS

5.1. Propagation Delay Time:

6. RACE AROUND CONDITION

6.1. Master Slave Flip flop:

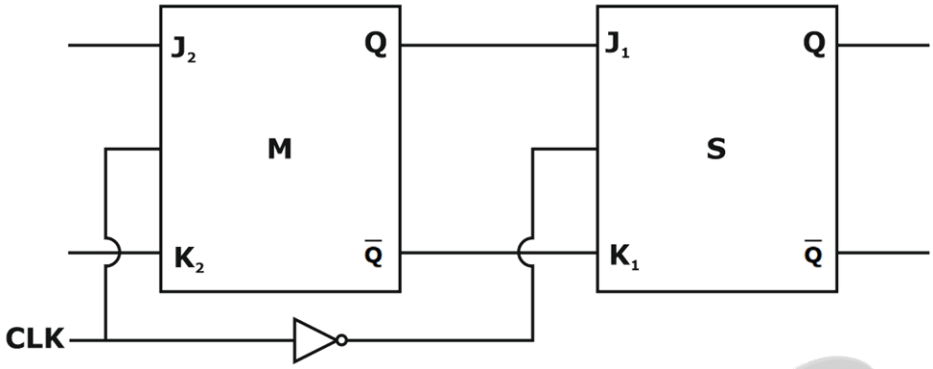


Figure 2: Logic Diagram for Master Slave JK Flip Flop

7. DESIGNING OF ONE FLIP FLOP BY OTHER FLIP FLOP

The steps for designing one flip flop or new flip flop using existing or same existing flip flop.

Step 1: Write the characteristic table for the designed flip flop.

Step 2: Write the excitation table for the available flip flop.

Step 3: Write the logical expression.

Step 4: Minimize the logical expression.

Step 5: Circuit Implementation.

8. SHIFT REGISTERS

8.1. SISO (serial-in, serial-out) Shift Register:

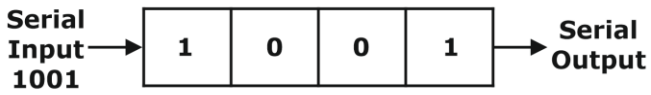


Figure 3: SISO Shift Register

8.2. SIPO (serial-in, parallel-out) Shift Register:

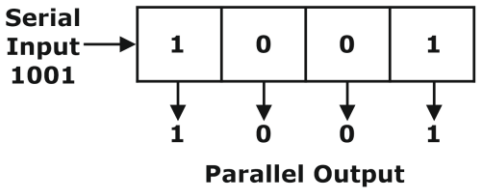


Figure 4: SIPO Shift Register

8.3. PISO (parallel-in, serial out) Shift Register:

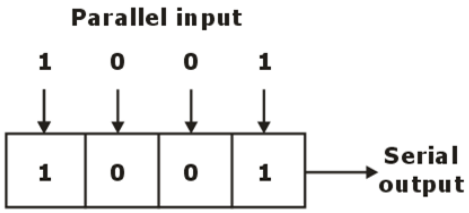


Figure 5: PISO shift register

8.4. PIPO (parallel in, parallel out) Shift Register:

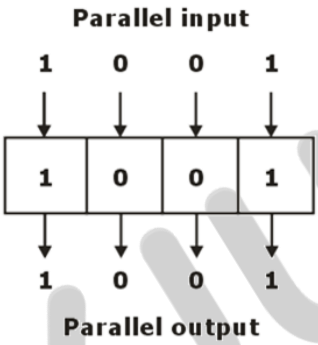


Figure 6: PIPO Shift Register

8.5. UNIVERSAL SHIFT REGISTER:

If the flip-flop outputs of a shift register are accessible, then information entered serially by shifting can be taking out in parallel from the outputs of the flip-flops. If a parallel

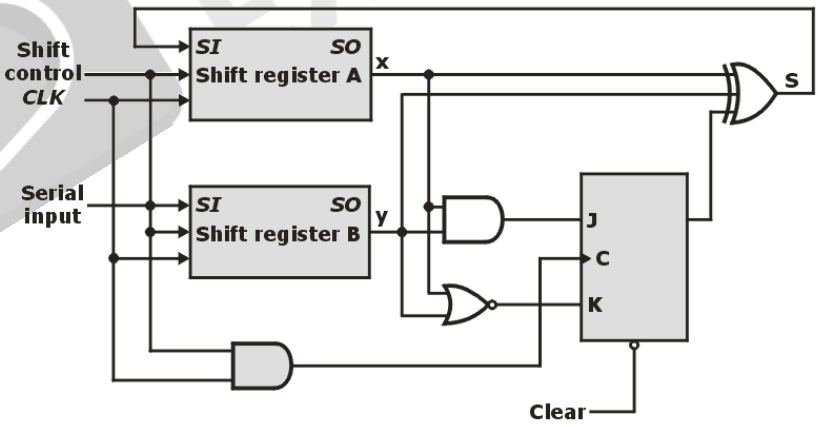


Figure 7: Second form of serial adder

The function for the Universal Shift Register is as follows:

Mode Control		Register Operation
S_1	S_0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

8.6. APPLICATIONS OF SHIFT REGISTERS

(a) Delay line: A shift register can be used to introduce a delay (Δt) in signals

$$\Delta t = N \times \frac{1}{f_c}$$

Where N is number of stages & f_c is the clock frequency.

(b) Serial-to-parallel converter

(c) Parallel-to-serial converter

(d) Divide by 2-circuit with right shift operation.

(e) Multiply by 2-circuit with left shift operation.

(f) Ring counter

(g) Twisted ring counter

9. COUNTERS

9.1. 3-BIT BINARY COUNTER:

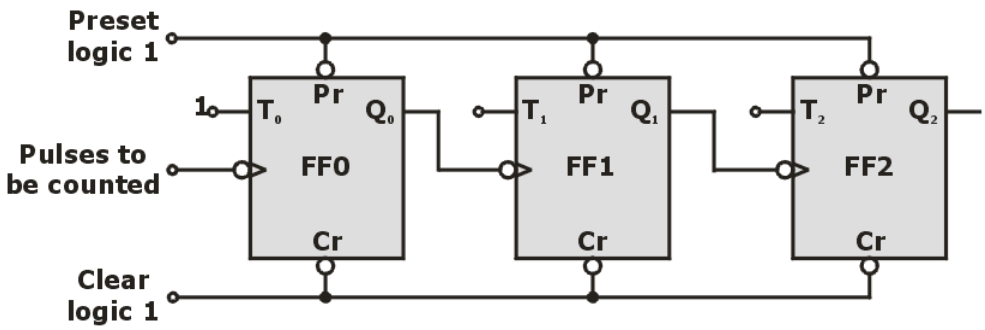
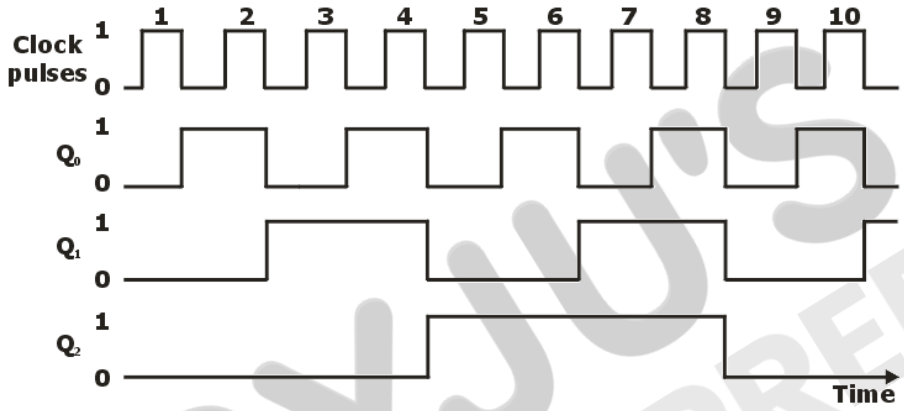


Figure 7: A 3-bit Binary Counter

Counter state	Count		
	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Output waveforms of the above counter is:



The frequency 'f' of clock pulses for reliable operation of the counter is given as

$$\frac{1}{f} \geq N.t_{pd} + T_s$$

Where, N = number of flip-flops

t_{pd} = propagation delay of one flip-flop.

T_s = strobe pulse width.

9.2. MOD-8 UP/DOWN COUNTER:

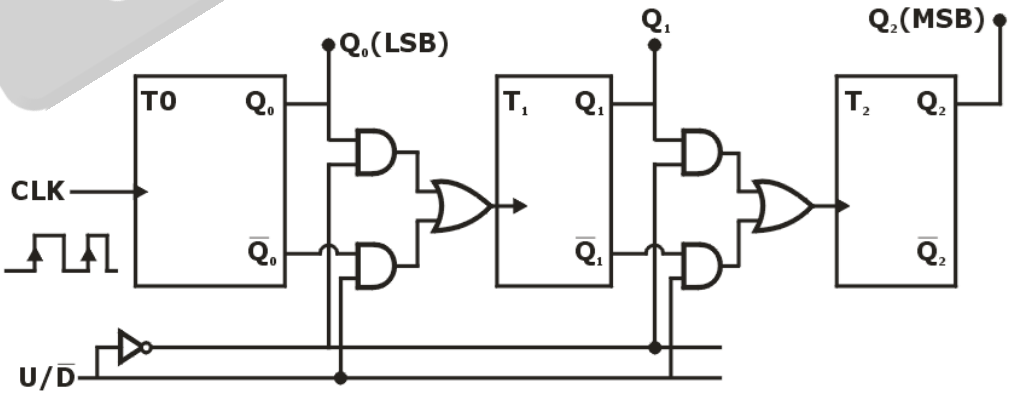


Figure 9: MOD-8 UP/DOWN Counter

9.3. BCD RIPPLE COUNTER

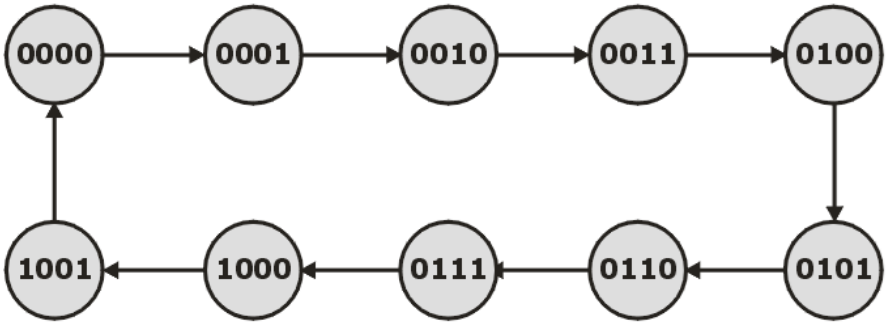
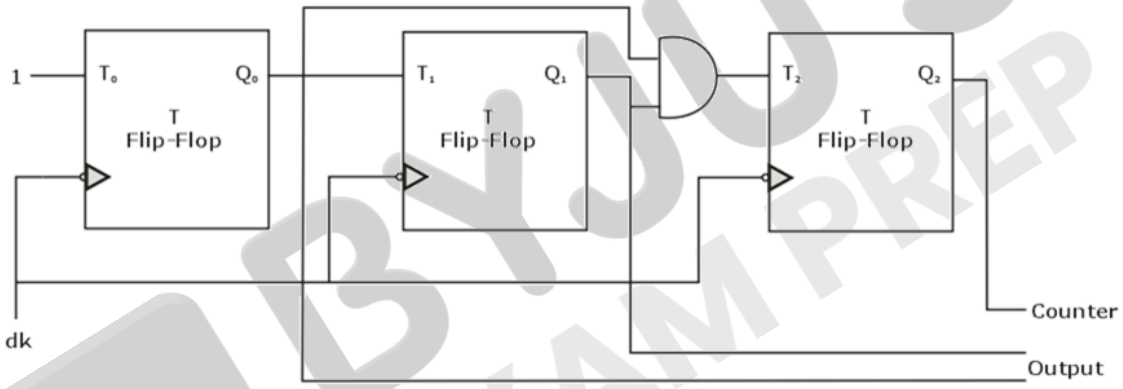


Figure 10: State diagram of a decimal BCD counter.

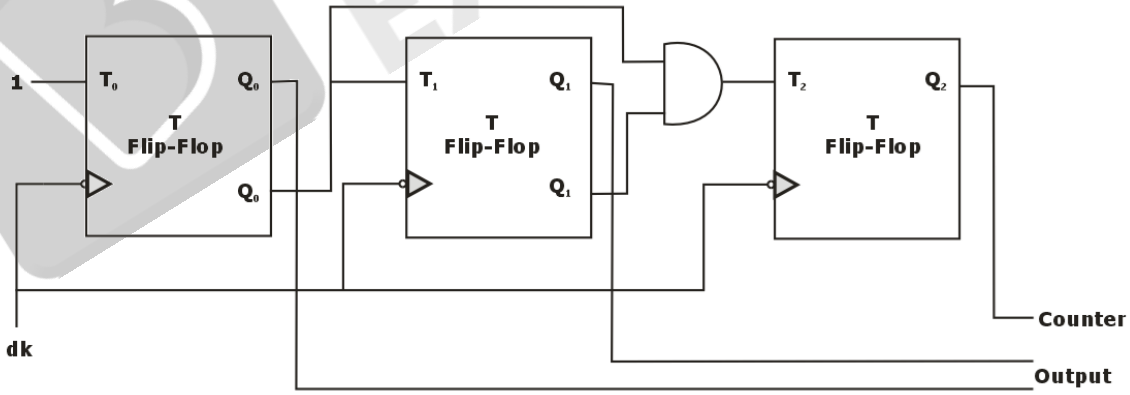
9.4. SYNCHRONOUS COUNTERS

There are two types of Synchronous counters:

a. Synchronous Binary Up Counter



b. Synchronous Binary Down Counter



9.5. Synchronous Counter Design:

Synchronous counters for any given count sequence and modulus can be designed in the following way:

1. Find the number of FLIP-FLOPs required.
2. Write the count sequence in the tabular form.
3. Determine the FLIP-FLOP inputs which must be present for the desired next state from the present state using the excitation table of the FLIP-FLOPS.

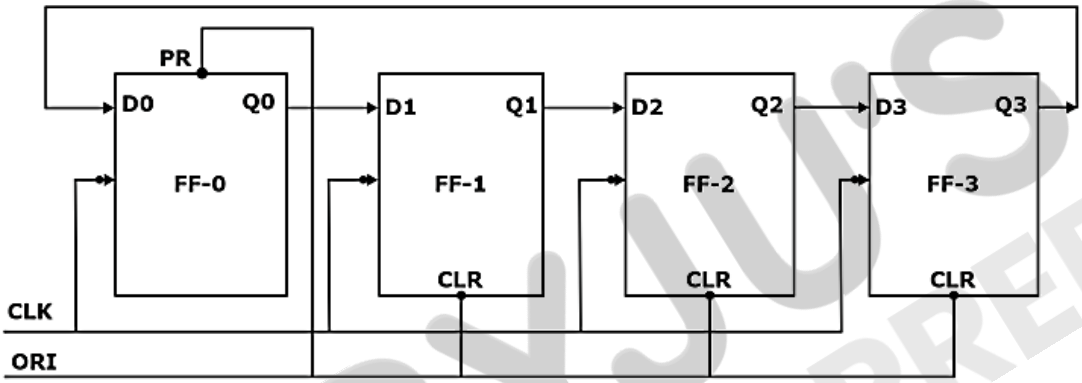
4. Prepare K-map for each FLIP-FLOP input in terms of FLIP-FLOP outputs as the input variables.
5. Simplify the K-maps and obtain the minimized expressions.
6. Connect the circuit using FLIP-FLOPS and other gates corresponding to the minimized expressions.

10. RING AND JOHNSON COUNTERS

10.1 Ring Counters:

No. of states in Ring counter = No. of flip-flop used.

With 'n' flip-flops, maximum count possible in ring counter is 2^{n-1}.

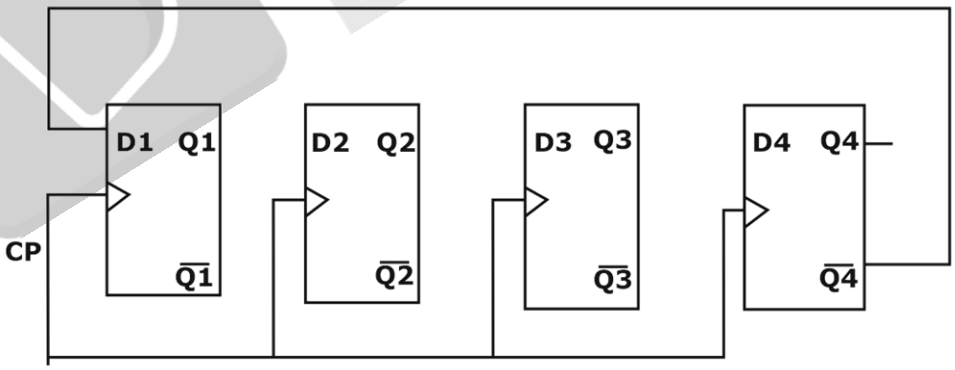


10.2 Johnson Counters:

Total number of used and unused states in n-bit Johnson counter:

number of used states=2n

number of unused states=2^n – 2n



11. SYNCHRONOUS SEQUENTIAL CIRCUIT MODELS

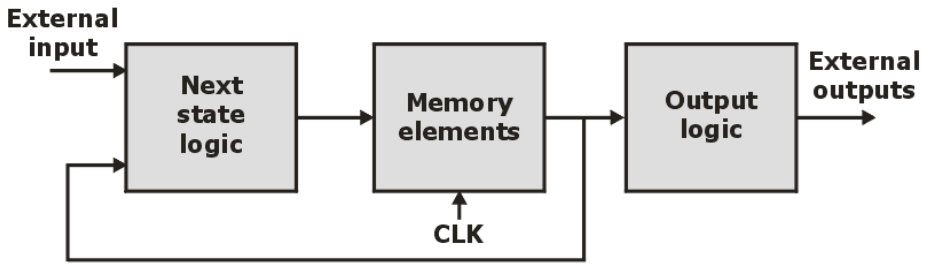
11.1. Mealy model:

In mealy model, the next of the function depends on present state as well as present inputs.

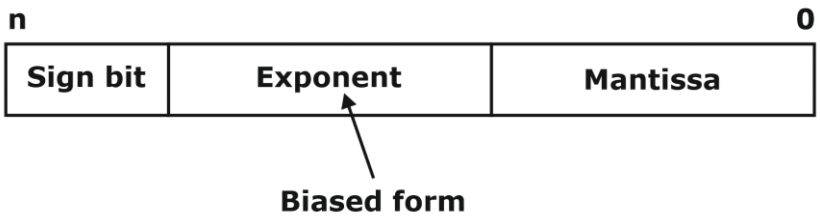
11.2. Moore Model:

In more model, the next state depends on the present state and present inputs but also on the outputs of more model.

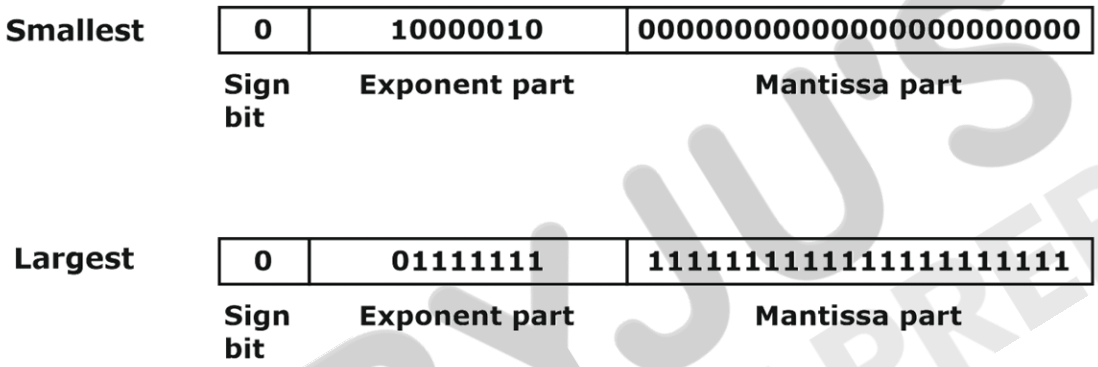
The block diagram of a Moore model is given as



Chapter 6: Floating Point Representation



So, actual number is $(-1)^s(1+m)x2^{(e-Bias)}$, where *s* is the sign bit, *m* is the mantissa, *e* is the exponent value, and *Bias* is the bias number.



Example: 01000001110100000000000000000000

10000011 = $(131)_{10}$

131-127 = 4

Hence the exponent of 2 will be 4 i.e. $2^4 = 16$.

****