Digital Logic Design and Analysis

Chapter 1 : Number Systems

Q. 1 Convert $(126)_{10}$ to Octal, Hexcode. May 15

Ans. :

Step 1 : Convert decimal to octal :

8	126	6	↑ LSB
8	15	7	
8	1	1	
	0		MSB

(C-5189)

$\therefore (126)_{10} = (176)_8$

Step 2 : Convert decimal to hexadecimal :

16	126	14 → E	↑ LSB
16	7	7 → 7	
	0		MSB

(C-5190)

$\therefore (126)_{10} = (7E)_{16}$ ...Ans.

Q. 2 Convert $(214.32)_{10}$ to binary. May 15

Ans. :

Step 1 : Convert integer and fractional parts :

214 · 32 (C-5191)

Integer Fractional part

Step 2 : Convert integer :

Step 3 : Convert fractional part :

2	214	0	↑ LSB
2	107	1	
2	53	1	
2	26	0	
2	13	1	
2	6	0	
2	3	1	
2	1	1	
	0		MSB

Decimal fraction	Base	Product	Carry	
0.32	× 2 =	0.64	0	MSB
0.64	× 2 =	1.28	1	
0.28	× 2 =	0.56	0	
0.56	× 2 =	1.12	1	
0.12	× 2 =	0.24	0	LSB

(C-5192)

$\therefore (214)_{10} = (11010110)_2$ $(0.32)_{10} = (0.01010)_2$

Step 4 : Combine results of steps 2 and 3 :

$(214.32)_{10} = (11010110.01010)_2$

Q. 3 Convert decimal number 199.375 into binary, octal, hexadecimal system. Dec. 15

Ans. :

$(199.375)_{10} = (\ ?\ )_2 = (\ ?\ )_8 = (\ ?\ )_{16}$

Step 1 : Convert decimal to binary :

Separate integer and fractional part :

Convert the integer : **Convert the fractional part :**

2	199	1	↑ LSB
2	99	1	
2	49	1	
2	24	0	
2	12	0	
2	6	0	
2	3	1	
2	1	1	
	0		MSB

Decimal	Base	Product	Carry	
0.375	× 2 =	0.75	0	MSB
0.75	× 2 =	1.5	1	
0.5	× 2 =	1.0	1	
0.0	× 2 =	0.0	0	LSB

$\therefore (199)_{10} = (11000111)_2$ $\therefore (0.375)_{10} = (0.0110)_2$ (C-5263)

$\therefore (199.375)_{10} = (11000111.0110)_2$

Step 2 : Convert decimal to octal :

Convert the integer : Convert the fractional part :

8	199	7	↑ LSB
8	24	0	
8	3	3	
	0		MSB

Decimal	Base	Product	Carry	
0.375	× 8 =	3.0	3	MSB
0.0	× 8 =	0.0	0	LSB

(C-5264)

$\therefore (199)_{10} = (307)_8$

$\therefore (0.375)_{10} = (0.30)_8$

$\therefore (199.375)_{10} = (307.30)_8$ ...Ans.

Step 3 : Convert decimal to hexadecimal :

Convert the integer : Convert the fractional part :

		Hex		
16	199	7 → 7	↑ LSB	
16	12	12 → C		
	0		MSB	

Decimal	Base	Product	Carry	
0.375	× 16 =	6.0	6	MSB
0.0	× 16 =	0.0	0	LSB

(C-5265)

$\therefore (199)_{10} = (C7)_{16}$

$\therefore (0.375)_{10} = (0.60)_{16}$

$\therefore (199.375)_{10} = (C7.60)_{16}$ ...Ans.

Q. 4 Convert decimal number 151.33 into binary, base-4, octal, hexadecimal system. Dec. 16

Ans. : 1. Decimal to binary : ...Ans.

Step 1 : Separate integer and fractional parts :

151 · 33 (C-5714)

Integer Fractional

Step 2 : Convert the integer :

2	151	1 ← LSB
2	75	1
2	37	1
2	18	0
2	9	1
2	4	0
2	2	0
2	1	1
	0	← MSB

$\therefore (151)_{10} = (10010111)_2$

Step 3 : Convert the fractional part :

	Base	Product	Carry	
0.33 × 2 =		0.66	0	MSB
0.66 × 2 =		1.32	1	
0.32 × 2 =		0.64	0	
0.64 × 2 =		1.28	1	
0.28 × 2 =		0.56	0	LSB

$\therefore (0.33)_{10} = (0.01010)_2$

Step 4 : Combine the results of steps 2 and 3 :

$\therefore (151.33)_{10} = (10010111.01010)_2$

2. Decimal to base-4 :

Convert the integer part : Convert the fractional part :

4	151	3 ← LSD
4	37	1
4	9	1
2	2	0
2	1	1
	0	← MSD

$\therefore (151)_{10} = (10113)_4$ (C-5716)

	Base	Product	Carry	
0.33 × 4 =		1.32	1	MSD
0.32 × 4 =		1.28	1	
0.28 × 4 =		1.12	1	
0.12 × 4 =		0.48	1	
0.48 × 4 =		1.92	1	LSD

$\therefore (0.33)_{10} = (0.11111)_2$ ...Ans.

$\therefore (151.33)_{10} = (10113.11111)_4$

3. Decimal to octal

Convert the obtained binary number to octal :

Binary : 010 010 111 . 010 100 — Group of 3 bits

Octal : 2 2 7 . 2 4 (C-5717)

...Ans.

$\therefore (151.33)_{10} = (227.24)_8$

4. Decimal to hexadecimal

Binary : 1001 0111 . 0101 0000 — Group of 4 bits

Hexadecimal : 9 7 . 5 0 (C-5718)

$\therefore (151.33)_{10} = (97.5)_{16}$ ...Ans.

Q. 5 Convert $(243.63)_8$ to decimal, binary and hexadecimal. May 11, Dec. 12

Ans. :

1. Conversion to decimal :

N = 2 4 3 . 6 3 Octal (C-2164)

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

$= 128 + 32 + 3 + 0.75 + 0.0468 = 163.7968$

$\therefore (243.63)_8 = (163.7968)_{10}$ ...Ans.

2. Conversion to binary

N = 2 4 3 . 6 3 Octal

010 100 011 . 110 011 Binary

$\therefore (243.63)_8 = (010100011.110011)_2$ ...Ans.

3. Conversion to hex :

Obtained binary number : $(010100011.110011)_2$

Add two zero's to extreme right (on LSB side to get (10100011.11001100).

Binary number : 1010 0011 . 1100 1100 — Group of 4 bits

Hex number : A 3 . C C

$\therefore (243.63)_8 = (A3.CC)_{16}$ ...Ans.

Q. 6 Convert $(650.17)_8$ into decimal, binary and hex. Dec. 11

Ans. :

1. Conversion to decimal :

Given octal number : 6 5 0 . 1 7 (C-2165)

$(6 \times 8^2) + (5 \times 8^1) + (0 \times 8^0) + (1 \times 8^{-1}) + (7 \times 8^{-2})$

$= 384 + 40 + 0 + 0.125 + 0.109$

$\therefore N = (424.234)_{10}$

2. Conversion to binary :

N = 6 5 0 . 1 7

110 101 000 . 001 111

$\therefore (650.17)_8 = (110101000.001111)_2$

3. Conversion to hex :

Obtained binary number : 110101000.001111

Add three zero's to extreme left (MSB side) and two zero's to extreme right (LSB side) to get (0001 1010 1000.0011 1100)$_2$

Binary number :	000 1	101 0	100 0	·	001 1	110 0	Group of 4 bits
Hex number :	1	A	8	·	3	C	

$\therefore (650.17)_8 = (1A8.3C)_H$

Q. 7 Convert the $(761.514)_8$ to binary and hexadecimal. Dec. 13

Ans. :

$(761.514)_8 = (?)_2 = (?)_H$:

Step 1 : Convert octal to binary :

Given octal number	7	6	1	·	5	1	4
	111	110	001	·	101	001	100

$\therefore (761.514)_8 = (111\ 110\ 001 \cdot 101\ 001\ 100)_2$ ...Ans.

Step 2 : Convert octal to hex :

$(761.514)_8 = (111\ 110\ 001 \cdot 101\ 001\ 100)_2$

Make group of 4 bits.

Binary number :	0001	1111	0001	•	1010	0110	0000
Hex number :	1	F	1		A	6	0

(C-4199)

$\therefore (761.514)_8 = (1F1.A60)_H$ ...Ans.

Q. 8 Convert the hex number $(67.4A)_{16}$ into equivalent octal number. Dec. 14

Ans. :

1. Given hex number :

	6	7	·	4	A
2. Convert to binary :	0110	0111	·	0100	1010

3. Now put additional zeros to extreme left and right and make groups of 3 bits.

001	100	111	·	010	010	100
1	4	7	·	1	2	4

(C-4957)

$\therefore (67.4A)_{16} = (147.124)_8$ ...Ans.

Q. 9 Convert $(532.125)_8$ into decimal, binary and hexadecimal. May 16

Ans. :

1. Conversion to decimal :

N = 5 3 2 · 1 2 5

$(5 \times 8^2) + (3 \times 8^1) + (2 \times 8^0) \cdot (1 \times 8^{-1}) + (2 \times 8^{-2}) + (5 \times 8^{-3})$ (C-5511)

$= 320 + 24 + 2 + 0.125 + 0.03125 + 0.009765$

$= 346.166015$

$\therefore (532.125)_8 = (346.166015)_{10}$ ...Ans.

2. Conversion to binary

N = 5 3 2 · 1 2 5

101 011 010 · 001 010 101

$\therefore (532.125)_8 = (101011010 \cdot 001010101)_2$ ...Ans.

3. Conversion to hexadecimal :

Obtained binary number : 101011010 · 001010101

Add 3 zeros to extreme left and right side.

Group of 4 bits :	0001	0101	1010	·	0010	1010	1000
Hex number :	1	5	A	·	2	A	8

$\therefore (532.125)_8 = (15A \cdot 2A8)_{16}$ ...Ans.

Q. 10 Convert decimal (215.32) into base '7'. Dec. 14

Ans. :

N = 215 • 32

Integer Fractional (C-4958)

Step 1 : Convert the integer part

Division	Quotient	Remainder	
215 / 7	30	5	LSD
30 / 7	4	2	
4 / 7	0	4	MSD

Step 2 : Convert the fractional part :

Fraction	Base	Product	Carry	
0.32	× 7 =	2.24	2	MSD
0.24	× 7 =	1.68	1	
0.68	× 7 =	4.76	4	LSD

$\therefore (215)_{10} = (425)_7$

$\therefore (0.32)_{10} = (0.214)_7$

Step 3 : Combine :

$(215 \cdot 32)_{10} = (425 \cdot 214)_7$ ...Ans.

Q. 11 Convert $(121.2)_3$ into base 10. May 15

Ans. :

Step 1 : Convert $(121.2)_3$ into decimal :

Given number : 1 2 1 · 2

Decimal : $(1 \times 3^2) + (2 \times 3^1) + (1 \times 3^0) + (2 \times 3^{-1})$ (C-5186)

$\therefore N = 9 + 6 + 1 + 0.66 = 16.66$

$\therefore (121.2)_3 = (16.66)_{10}$ ...Ans.

Q. 1 Find the one's complement and two's complement of $(57)_{10}$. May 15

Ans. :

Step 1 : Convert decimal to binary :

$(57)_{10} \longrightarrow (111001)_2$

Step 2 : 1's complement of $(57)_{10}$:

$(111001)_2 \xrightarrow{\text{1's complement}} (000110)_2$

Step 3 : 2's complement of $(57)_{10}$:

$(111001)_2 \xrightarrow{\text{1's complement}} (000110)_2 \xrightarrow{\text{add 1}} (000111)_2$

$\therefore \quad (9)_{10} - (5)_{10} = (4)_{10}$ ...Ans.

Q. 2 Perform binary subtraction using 2's complement for $(62)_{10}$ and $(99)_{10}$. May 15

Ans. : Step 1 : Convert both the numbers to binary :

$(62)_{10} = (0111110)_2$

$(99)_{10} = (1100011)_2$

Step 2 : Obtain 2's complement of $(99)_{10}$:

Decimal Binary 1's complement 2's complement

$(99)_1 \longrightarrow (1100011)_2 \longrightarrow (0011100)_2 \longrightarrow (0011101)_2$

Step 3 : Add $(62)_{10}$ and 2's complement of $(99)_{10}$:

$(62)_{10}$: 0 1 1 1 1 1 0

2's complement of $(99)_{10}$: + 0 0 1 1 1 0 1

Carry : 1 1 1 1

0 1 0 1 1 0 1 1 (C-5193)

"0" carry indicates that the result is negative and in its 2's complement form

Step 4 : Convert the answer into its true form :

Answer : 1 0 1 1 0 1 1

Subtract 1 : − 1

1 0 1 1 0 1 0 (C-5194)

Invert all bits

0 1 0 0 1 0 1

$\therefore (62)_{10} - (99)_{10} = -(0100101)_2 = -(37)_{10}$ ...Ans.

$\therefore (73)_{10} - (49)_{10} = (24)_{10}$ ...Ans.

Q. 3 Subtract using 1's and 2's complement $(15)_{10} - (21)_{10}$. May 12

Ans. :

Using 1's complement method :

Step 1 : Obtain 1's complement of $(21)_{10}$:

$(21)_{10} \xrightarrow{\text{Binary}} (10101)_2 \xrightarrow{\text{1's complement}} (01010)_2$

Step 2 : Add $(15)_{10}$ and 1's complement of $(21)_{10}$:

$(15)_{10}$: 0 1 1 1 1

$(21)_{10}$: 0 1 0 1 0

Carry 1 1 1

Final carry → 0 1 1 0 0 1 (C-3707)

As the final carry is 0, the answer is negative and in it's 1's complement form.

So convert the answer into its true form as follows :

invert

1 1 0 0 1 ⟶ 00110

$\therefore \quad (15)_{10} - (21)_{10} = (-6)_{10} = -(110)_2$ ...Ans.

Using 2's complement form :

Step 1 : Obtain 2's complement of $(21)_{10}$:

$(21)_{10} \xrightarrow{\text{Binary}} (10101)_2 \xrightarrow{\text{1's complement}} (01010)_2 \xrightarrow{\text{add 1}} (01011)_2$

Step 2 : Add $(15)_{10}$ and 2's complement of $(21)_{10}$:

$(15)_{10}$: 0 1 1 1 1

$(21)_{10}$: 0 1 0 1 1

1 1 1 1

Final carry → 0 1 1 0 1 0 (C-3708)

As the final carry is 0, the answer is negative and in it's 2's complement form.

Step 3 : Convert the answer into its true form :

Answer : 1 1 0 1 0

Subtract 1 : − 1

1 1 0 0 1

Invert all bits

00110 (C-3709)

$\therefore \quad (15)_{10} - (21)_{10} = -(6)_{10} = (110)_2$ ...Ans.

Q. 4 Subtract the following using method given below : Dec. 13

1. $(11)_{10} - (22)_{10}$ **using 2's complement.**
2. $(33)_{10} - (44)_{10}$ **using one's complement.**

Ans. :

1. $(11)_{10} - (22)_{10}$ using 2's complement

Step 1 : Obtain 2's complement of $(22)_{10}$:

$(22)_{10} \xrightarrow{\text{Binary}} (10110)_2 \xrightarrow{\text{1's complement}} (01001)_2 \xrightarrow{\text{add 1}} (01010)_2$

Step 2 : Add $(11)_2$ and 2's complement of $(22)_{10}$:

$(11)_2$: 0 1 0 1 1

2's complement of $(22)_{10}$: 0 1 0 1 0 (C-4200)

1 1

Final carry → 0 1 0 1 0 1

As final carry is not generated the answer is negative and in 2's complement form.

Step 3 : Convert the answer into its true form :

$$10101 \xrightarrow[\text{complement}]{2's} 1011$$

$\therefore (11)_{10} - (22)_{10} = -(1011)_2 = -(11)_2$ ...Ans.

2. $(33)_{10} - (44)_{10}$ using one's complement :

Step 1 : Obtain 1's complement of $(44)_{10}$:

$$(44)_{10} \xrightarrow{\text{Binary}} (101100)_2 \xrightarrow{\text{1's complement}} (010011)_2$$

Step 2 : Add $(33)_{10}$ and 1's complement of $(44)_{10}$:

$(33)_{10}$	:	1	0	0	0	0	1
1's complement of $(44)_{10}$	:	0	1	0	0	1	1
Carry	:				1	1	
Final carry →		0	1	1	0	1	0	0

(C-4201)

As final carry is not generated the answer is negative and in its 1's complement form.

Invert all bits $\therefore (110100)_2 \longrightarrow (001011)_2$

$\therefore (33)_{10} - (44)_{10} = -(1011)_2 = -(11)_2$ ...Ans.

Q. 5 Subtract using 1's and 2's complement method $(56)_{10} - (76)_{10}$. May 16

Ans. :

Using 1's complement method :

Step 1 : Obtain 1's complement of $(76)_{10}$:

$$(76)_{10} \xrightarrow{\text{Binary}} (1001100)_2 \xrightarrow{\text{1's complement}} (0110011)_2$$

Step 2 : Add $(56)_{10}$ and 1's complement of $(76)_{10}$:

$(56)_{10}$	:	0	1	1	1	0	0	0
1's complement of $(76)_{10}$	:	0	1	1	0	0	1	1
Carry	:	1	1					
Final carry →		0	1	1	0	1	0	1	1

(C-5517)

As the final carry is 0, the answer is negative and in its 1's complement form. So convert answer into its true form as follows :

$$\text{Answer : } (1101011)_2 \xrightarrow{\text{Invert}} (0010100)_2 = (20)_{10}$$

$\therefore (56)_{10} - (76)_{10} = -(20)_{10}$ ...Ans.

Using 2's complement form :

Step 1 : Obtain 2's complement of $(76)_{10}$:

$$(76)_{10} \xrightarrow{\text{Binary}} (1001100)_2 \xrightarrow{\text{1's complement}} (0110011)_2 \xrightarrow{\text{add 1}} (0110100)_2$$

Step 2 : Add $(56)_{10}$ and 2's complement of $(76)_{10}$:

$(56)_{10}$	:	0	1	1	1	0	0	0
2's complement of $(76)_{10}$	:	0	1	1	0	1	0	0
Carry	:	1	1					
Final carry →		0	1	1	0	1	1	0	0

(C-5518)

As the final carry is 0, the answer is negative and in its 2's complement form.

Step 3 : Convert the answer into its true form :

Answer :	1	1	0	1	1	0	0
Subtract 1 :							1
	1	1	0	1	0	1	1

(C-5519)

Invert

$0\,0\,1\,0\,1\,0\,0 = (20)_{10}$

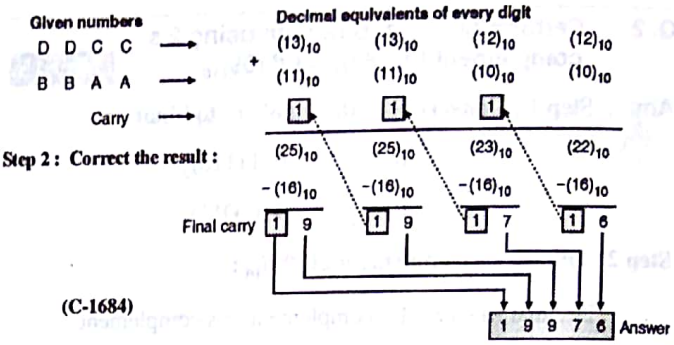
$\therefore (56)_{10} - (76)_{10} = -(20)_{10}$ ...Ans.

Q. 6 Add $(DDCC)_{16}$ and $(BBAA)_{16}$. Dec. 14

Ans. :

The addition takes place as shown in Fig. 2.1.

Step 1 : Add the digits by assuming them to be digital :



(C-1684)

$\therefore (DDCC)_{16} + (BBAA)_{16} = (19976)_{16}$ ...Ans.

Fig.2.1: Addition of $(DDCC)_{16}$ and $(BBAA)_{16}$

Q. 7 Perform the following directly without converting to any other base :

1. $(63)_8 \times (21)_8$ 2. $(D9)_H - (80)_H$ Dec. 13

Ans. :

1. $(63)_8 \times (21)_8$:

Step 1 : Convert octal number to binary :

$(63)_8 = (110011)_2$

$(21)_8 = (010001)_2$

Step 2 : Perform binary multiplication :

$$\begin{array}{r} 1\,1\,0\,0\,1\,1 \\ \times\ 0\,1\,0\,0\,0\,1 \\ \hline 1\,1\,0\,0\,1\,1 \\ 0\,0\,0\,0\,0\,0- \\ 0\,0\,0\,0\,0\,0-- \\ 0\,0\,0\,0\,0\,0--- \\ 1\,1\,0\,0\,1\,1---- \\ 0\,0\,0\,0\,0\,0----- \\ \text{Carry}\quad 1\ 1 \\ \hline 0\,0\,1\,1\,0\,1\,1\,0\,0\,0\,1\,1 \end{array}$$

(C-4202)

Binary

1 5 4 3 Octal

$\therefore (63)_8 \times (21)_8 = (1543)_8$ ...Ans.

2. $(D9)_H - (80)_H$:

Borrow	16		
A :	6		3
B :	8		0
	14		3

(C-4203)

Q. 8 **Perform following without converting into other bases :** May 14

1. $(57)_8 \times (24)_8$ 2. $(312.0)_4 + (213.2)_4$

Ans. :

1. $(57)_8 \times (24)_8$:

Step 1 : Convert both octal numbers into binary :

$(57)_8 = (101\ 111)_2$

$(24)_8 = (010\ 100)_2$

Step 2 : Carry out the binary multiplication :

```
      1 0 1 1 1 1
    × 0 1 0 1 0 0          (C-4960)
    -------------
      0 0 0 0 0 0
    + 0 0 0 0 0 0 X
    + 1 0 1 1 1 1 X X
    + 0 0 0 0 0 0 X X X
    + 1 0 1 1 1 1 X X X X
    + 0 0 0 0 0 0 X X X X X
Carry   1 1 1 1
    -------------------------
    0 1 1 1 0 1 0 1 1 0 0
```

Step 3 : Convert the answer into octal :

(C-4961)

```
001  110  101  100
 ↓    ↓    ↓    ↓
 1    6    5    4
```

$\therefore\ (57)_8 \times (24)_8 = (1654)_8$

2. $(312.0)_4 + (213.2)_4$:

```
      3   1   2   .   0
      2   1   3   .   2
          1
    -------------------
      5   3   5   .   2          (C-4962)
     -4      -4
    -------------------
     (1)1    (1)1
      ↓       ↓       ↓
      1   1   3   1   .   2
```

$\therefore\ (312.0)_4 + (213.2)_4 = (1131.2)_4$ ...Ans.

Q. 9 **Perform hexadecimal arithmetic operation : DADA + BABA.** Dec. 15

Ans. :

Step 1 : Add the digits by assuming them to be decimal :

Given numbers				Decimal equivalents			
+ D	A	D	A	$(13)_{10}$	$(10)_{10}$	$(13)_{10}$	$(10)_{10}$
B	A	B	A	$(11)_{10}$	$(10)_{10}$	$(11)_{10}$	$(10)_{10}$
Carry				1	1	1	
1	9	5	9	$(25)_{10}$	$(21)_{10}$	$(25)_{10}$	$(20)_{10}$
4				$-(16)_{10}$	$-(16)_{10}$	$-(16)_{10}$	$-(16)_{10}$
				1 9	1 5	1 9	1 4

Final carry

1 9 5 9 4

$\therefore\ (DADA)_{16} + (BABA)_{16} = (19594)_{16}$ ...Ans.

Q. 10 **Perform the following operations without changing the base :** May 16

1. $(314)_8 + (737)_8$ 2. $(312.40)_5 + (214.33)_5$

Ans. :

1. $(314)_8 + (737)_8$:

```
      3   1   4
    + 7   3   7
    -------------
Carry  1   1
      10   5  11
     -8      -8
    -------------
      2   3   3          (C-5515)
      ↓   ↓   ↓
      1   2   5   3
```

$\therefore\ (314)_8 + (737)_8 = (1253)_8$ ...Ans.

2. $(312.40)_5 + (214.33)_5$:

```
      3   1   2   .   4   0
    + 2   1   4   .   3   3
    -----------------------
Carry 1   1   1
      5   3   7   .   7   3
     -5      -5      -5
    -----------------------
     (1)0    (1)2    (1)2
      ↓       ↓       ↓   ↓
      1   0   3   2   .   2   3
```

$\therefore\ (312.40)_5 + (214.33)_5 = (1032.23)_5$ ...Ans.

Chapter 3 : Codes

Q. 1 **Explain in brief weighted and non-weighted codes with one example each.** Dec. 15

Ans. : Weighted binary codes are those codes which are based on the principle of positional weight. Each position of a number represents a specific weight. Several systems of codes are used to express the decimal digits 0 through 9. These codes have been listed in Fig. 3.1. The codes 8421, 2421, 3321 .... all are the weighted codes. In these codes each decimal digit is represented by a group of four bits.

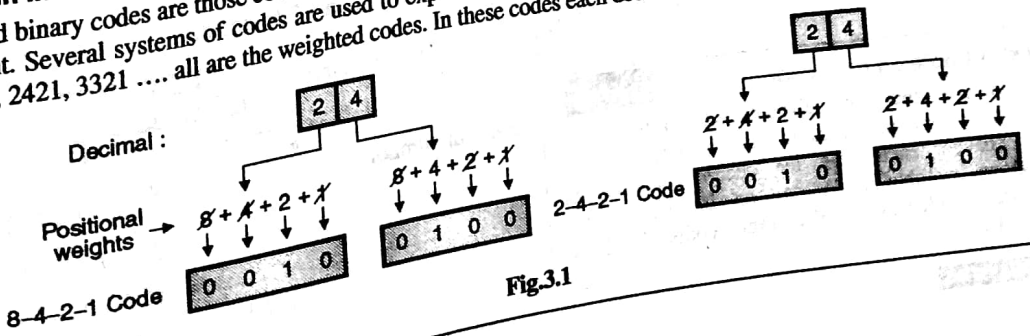
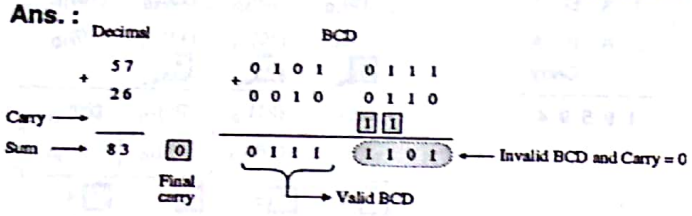


Fig. 3.1

Q. 2 Add $(57)_{10}$ and $(26)_{10}$ in BCD. Dec. 14

Ans. :



(C-80)

We have to add 6 to the sum to correct it.

	0111 1101	Incorrect sum
+	0000 0110	Add 6 for correction
Carry	1111 1	
	1000 0011	Correct result
	8 3	

$\therefore (57)_{10} + (26)_{10} = (83)_{10}$ ...Ans.

Q. 3 Write short note on Excess-3 code. May 15

Ans. : Excess – 3 is also called as XS - 3 code. It is a nonweighted code used to express decimal numbers as shown in Table 3.1. The Excess-3 code words are derived from the 8421 BCD code words by adding $(0011)_2$ or $(3)_{10}$ to each code word in 8421. The excess - 3 codes are obtained as follows :

$$\text{Decimal Number} \rightarrow \text{8421 BCD} \xrightarrow[0011]{\text{Add}} \text{Excess} - 3 \text{ code}$$

Excess – 3 codes for the single digit decimal numbers are listed in Table 3.1.

Table 3.1 : Excess – 3 codes

Decimal	BCD 8 4 2 1	Excess - 3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Note : Excess – 3 is a sequential code, because each succeeding code is one binary number greater than its preceeding code.

Excess – 3 is a **sequential code** because we get any code word by adding binary 1 to its previous code word as illustrated in Table 3.1. Excess – 3 is a **self complementing code.** This is because in Excess – 3 we get the 9's complement of a number by just complementing each bit that means by replacing a 0 by 1 and 1 by 0.

Q. 4 Write short note on gray code. May 15

Ans. : Gray code is another non-weighted code. It is not an arithmetic code. It has a very special feature that only one bit in the gray code will change, each time the decimal number is incremented as shown in Table 3.2. As only one bit changes at a time, the Gray code is called as a **unit distance code.** The Gray code is a **Cyclic code.**

(C-91) **Table 3.2**

Decimal	Binary	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0
16	1 0 0 0 0	0 0 0 0

Note :
1. In gray code only the encircled bit changes when the decimal number is incremented by 1.
2. In binary one, two, three or sometimes all the 4 bits change when the decimal number is incremented by 1.

Q. 5 Represent $(52)_{10}$ into Excess - 3 code and Gray code. May 15

Ans. :

Step 1 : Write the BCD equivalent of $(52)_{10}$:

$$(52)_{10} = (01010010)_{BCD}$$

Step 2 : Convert decimal to excess-3 code :

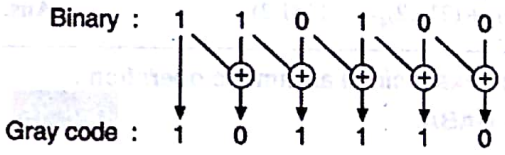
$(52)_{10}$: 0 1 0 1 0 0 1 0

Excess 3 : 1 0 0 0 0 1 0 1

$\therefore (52)_{10} = (1000\ 0101)_{Excess\text{-}3}$

Step 3 : Convert $(52)_{10}$ to gray code :

$$(52)_{10} = (110100)_2$$



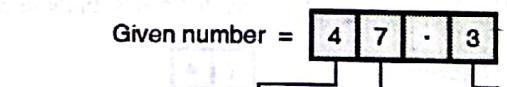
(C-5187)

$\therefore (52)_{10} = (101110)_{Gray}$ ...Ans.

Q. 6 Convert $(47.3)_7$ into BCD, excess-3 and gray code. May 16

Ans. :

Step 1 : Convert base 7 to decimal :



(C-5512)

$$\text{Decimal number} = (4 \times 7^1) + (7 \times 7^0) \cdot (3 \times 7^{-1})$$

$$= 28 + 7 + 0.4285$$

$$= 35.4285$$

$\therefore (47.3)7 = (35.4285)_{10}$

Step 2 : Conversion to BCD :

Decimal	:	3	5	.	4	2	8	5
BCD	:	0011	0101	.	0100	0010	1000	0101

$\therefore (47.3)_7 = (0011\ 0101 \cdot 0100\ 0010\ 1000\ 0101)_{BCD}$

Step 3 : Conversion to Excess – 3 : ...Ans.

Decimal	:	3	5	.	4	2	8	5
Excess-3 equivalent	:	6	8	.	7	5	11	8
		0110	1000	.	0111	0101	1011	1000

$\therefore (47.3)_7 = (0110\ 1000 \cdot 0111\ 0101\ 1011\ 1000)_{Excess-3}$

...Ans.

Step 4 : Conversion to binary :

Convert the integer :

2	35	1	↑ LSB
2	17	1	
2	8	0	
2	4	0	
2	2	0	
2	1	1	
	0		MSB

$\therefore (35)_{10} = (100011)_2$

Convert the fractional part :

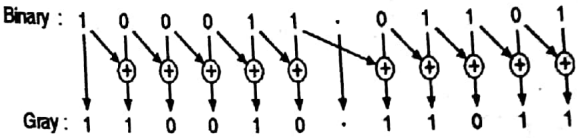
			Base	Product	Carry	
0.4285	×	2	=	0.857	0	MSB
0.857	×	2	=	1.714	1	
0.714	×	2	=	1.428	1	
0.428	×	2	=	0.856	0	
0.856	×	2	=	1.712	1	LSB

$\therefore (0.4285)_{10} = (0.01101)_2$

(C-5513)

$\therefore (35.4285)_{10} = (100011.01101)_2$

Step 5 : Conversion to gray :



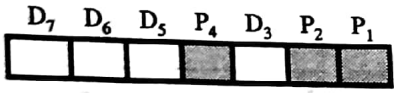
(C-5514)

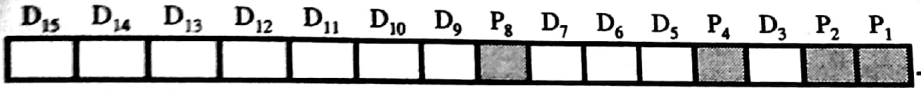
$\therefore (47.3)_7 = (100011 \cdot 01101)_2 = (110010 \cdot 11011)_{gray}$

...Ans.

Q.7 Prove that hamming code is an error detecting and correcting code. May 11, Dec. 12, May 14

Ans. : Hamming code is basically a linear block code named after its inventor. It is an error correcting code. The parity bits are inserted in between the data bits as shown in Fig. 3.2. The 7-bit Hamming code is used commonly, but the concept can be extended to any number of bits.





D → Data bits

P → Parity bits.

Fig. 3.2 : Hamming code words

Note that the parity bits are inserted at each 2^n bit where n = 0, 1, 2, 3 ..... 3.2.

1 7-Bit Hamming Code

A scientist named R.W. Hamming developed a coding system which was easy to implement. Assuming that four data bits are to be transmitted, he suggested a code word pattern shown in Fig. 3.3.

$D_7\ D_6\ D_5\ D_3$ → Data bits

$P_4\ P_2\ P_1$ → Parity bits.

Fig. 3.3 : Code word pattern for Hamming code

The D bits in Fig. 3.3 are data bits, whereas P bits are parity bits. The parity bits P_1, P_2, P_4 are adjusted in a particular way as explained below.

2 Minimum Number of Parity Bits

Table 3.3 gives a listing of minimum number of parity bits needed for various ranges of "m" information bits.

Table 3.3 : Number of parity bits to be used

Number of information bits	Number of parity bits
2 to 4	3
5 to 11	4
12 to 26	5
27 to 57	6
58 to 120	7

3 Deciding the Values of Parity Bits : Table 3.3(a) indicates which bit positions are associated with each parity bit in order to establish required parity (even or odd) over the selected bits positions.

Table 3.3(a)

Parity Bit	Bits to be checked
P_1	1,3,5,7,9,11,13,15,.....
P_2	2,3,6,7,10,11,14,15, ...
P_4	4,5,6,7,12,13,14,15, ...
P_8	8,9,10,11,12,13,14,15, ...

4 Deciding the Parity Bits for a 7 Bit Code :

Selection of P_1 : P_1 is adjusted to 0 or 1 so as to establish even parity over bits 1,3,5 and 7 i.e. P_1, D_3, D_5 and D_7.

→ Consider bits 1,3,5,7 for P_1

→ Consider bits 2,3,6,7 for P_2

→ Consider bits 4,5,6,7 for P_4

Selection of P_2 :

P_2 is adjusted to 0 or 1 so as to set even parity over bits 2,3,6 and 7 (P_2, D_3, D_6 and D_7).

Selection of P_4 :

P_4 is adjusted to 0 or 1 so as to set even parity over bits 4, 5, 6 and 7 (P_4, D_5, D_6 and D_7).

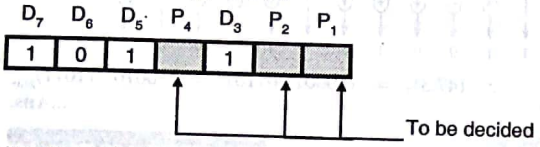
Q. 8 A bit word 1 0 1 1 is to be transmitted. Construct the even parity seven-bit Hamming code for this data. May 15

Ans. :

Step 1 : The code word format :

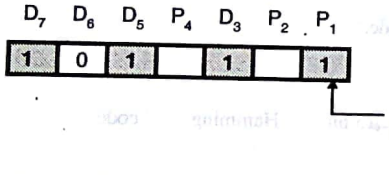
The seven bit Hamming code format is shown below

Given bit word = 1 0 1 1



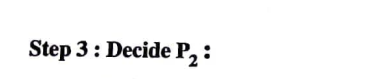
Step 2 : Decide P_1 :

P_1 sets the parity of bits P_1, D_3, D_5 and D_7. As D_7 D_5 D_3 = 1 1 1 we have to set P_1 = 1 in order to have the even parity.



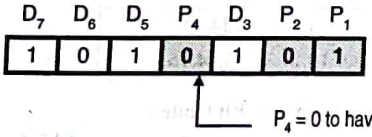
Step 3 : Decide P_2 :

P_2 is set to have the even parity of P_2, D_3, D_6 and D_7. But D_3, D_6, D_7 = 1 0 1 hence set P_2 = 0.



Step 4 : Decide P_4 :

P_4 is set to have the even parity of P_4, D_5, D_6 and D_7. But D_5 D_6 D_7 = 1 0 1, hence set P_4 = 0.

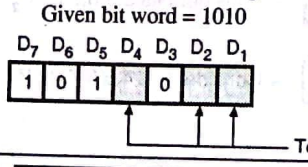


Q. 9 Write the hamming code for 1010. May 10, May 11, May 14, Dec. 15

Ans. : Step 1 : The code word format :

The seven bit Hamming code format is given :

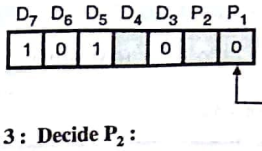
Given bit word = 1010



Step 2 : Decide P_1 :

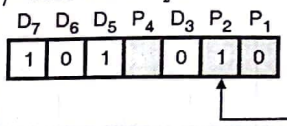
P_1 is set to have even parity of P_1 D_3 D_5 D_7

But D_7 D_5 D_3 = 110 hence set P_1 = 0.



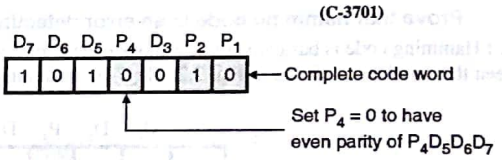
Step 3 : Decide P_2 :

P_2 is set to have even parity of P_2 D_3 D_6 and D_7. But D_3 D_6 D_7 = 001 hence set P_2 = 1



Step 4 : Decide P_4 :

P_4 is set to have even parity of P_4 D_5 D_6 D_7. But D_5 D_6 D_7 = 101. Hence set P_4 = 0.

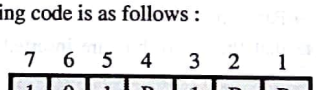


Q. 10 Obtain odd parity Hamming code for 1011. May 16

Ans. :

Data word : 1011

Format of 7-bit hamming code is as follows :



Step 1 : Decide P_1 :

Consider bits 1, 3, 5 and 7. They are 111 P_1

So for odd parity set P_1 = 0 ∴ P_1 = 0

Step 2 : Decide P_2 :

Consider bits 2, 3, 6 and 7. They are 101 P_2

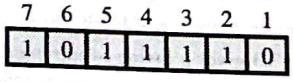
So for odd parity set P_2 = 1 ∴ P_2 = 1

Step 3 : Decide P_4 :

Consider bits 4, 5, 6 and 7. They are 101 P_4

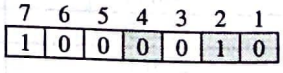
So for odd parity set P_4 = 1 ∴ P_4 = 1

Obtained codeword is as follows :



Q. 11 A 7 bit even parity hamming code is received as 1000010. Correct it for any errors and extract 4 bit data. Dec. 16

Ans. :



Step 1 : Check bits 4, 5, 6, 7 = 0001, odd parity hence error ∴ $P_4 = 1$

Step 2 : Check bits 2, 3, 6, 7 = 1001, even parity hence no error ∴ $P_2 = 0$

Step 3 : Check bits 1, 3, 5, 7 = 0001, odd parity hence error ∴ $P_1 = 1$

∴ Error word E = | 1 | 0 | 1 |
P_4 P_2 P_1

Decimal equivalent of error word is $(5)_{10}$. Hence the fifth bit in the received code word is incorrect. Hence invert the fifth bit to get the correct codeword as follows :

Correct codeword : | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
↑
Fifth bit inverted

Chapter 4 : Logic Gates and Boolean Algebra

Q. 1 State the Boolean algebra laws used in K-map simplification. Dec. 14

Ans. : The most important postulates used in Boolean system are as follows :

1. Closure
2. Associative law
3. Commutative law
4. Identity element
5. Inverse
6. Distributive law.

1. **Closure :** A set S is said to be closed for a particular binary operator if for every pair of elements of S, that binary operator specifies a rule to obtain a unique element of S.
2. **Associative law :** For a given set "S" a binary operator * can be said to be associative if the following equation is satisfied.

$$(A * B) * C = A * (B * C) \quad \text{for all } (A, B, C) \in S \quad ...(1)$$

Note that * can be an operator such that · (product) or + (addition) etc.

3. **Commutative law :** For a given set "S" a binary operator * is said to be commutative, if the following equation is satisfied.

$$A * B = B * A \quad \text{for all } A, B \in S \quad ...(2)$$

4. **Identity element :** A set S is said to be having an identity element with respect to a binary operation * on S, if there is an element x in set S that satisfies the following property.

$$x * A = A * x = A \quad \text{for every } A \in S \quad ...(3)$$

The example of identity element is 0 for the + operation on the set of integers (I).

$$I = \{ ....., -2, -1, 0, 1, 2, ... \}$$

$$\therefore \quad A + 0 = 0 + A = A \quad \text{for every } A \in I.$$

Thus for the set of integers (I) the element "0" acts as an identity element.

5. **Inverse :** A set S having an identity element x with respect to a binary operator * is said to have an inverse if for every A which belongs to S, there exists another element B which also belongs to S such that,

$$A * B = x$$

Take the example of set of integers (I) for which 0 is the identity element with respect to + operation that means x = 0 and * = +. Hence the inverse of an element A would be – A because only then the condition A * B = x is satisfied as follows : (C-128)

$A + (-A) = 0$ → Identity element
Given element ← A
→ Another element in the set
→ Operator

6. **Distributive law :** If * and · are two binary operators working on a set S, then * is said to be distributive over · if the following condition is satisfied.

$$A * (B \cdot C) = (A * B) \cdot (A * C) \quad ...(4)$$

Q. 2 State and prove De-Morgan's theorems. Dec. 11, May 12, Dec. 12, May 16

Ans. : The two theorems suggested by De-Morgan and which are extremely useful in Boolean algebra are as follows :

Theorem 1 : $\overline{AB} = \bar{A} + \bar{B}$: NAND = Bubbled OR :

This theorem states that the, complement of a product is equal to addition of the complements. This rule is illustrated in Fig. 4.1. The Left Hand Side (LHS) of this theorem represents a NAND gate with inputs A and B whereas the Right Hand Side (RHS) of the theorem represents an OR gate with inverted inputs. Such an OR gate is called as "Bubbled OR". Thus we can state De-Morgan's first theorem as a NAND operation is equivalent to a bubbled OR operation.

NAND ≡ Bubbled OR

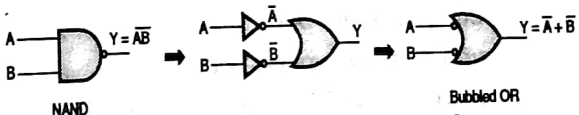


Fig. 4.1 : Illustration of De-Morgan's first theorem

This theorem can be verified by writing a truth table as shown in Fig. 4.2.

A	B	$\overline{AB}$	$\bar{A}$	$\bar{B}$	$\bar{A} + \bar{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

LHS $\overline{AB} = \bar{A} + \bar{B}$ RHS

Fig. 4.2 : Verification of the theorem $\overline{AB} = \bar{A} + \bar{B}$

Theorem 2 : $\overline{A + B} = \bar{A} \cdot \bar{B}$: NOR = Bubbled AND :

This theorem is illustrated in Fig. 4.3. The LHS of this theorem represents a NOR gate with inputs A and B whereas the RHS represents an AND gate with inverted inputs. Such an AND gate is called as "Bubbled AND". Thus we can state De-Morgan's second theorem as a NOR function is equivalent to a bubbled AND function.

NOR = Bubbled AND

Fig. 4.3 : Illustration of De-Morgan's second theorem

This theorem can be verified by writing a truth table for both the sides of the theorem statement. This truth table is shown in Fig. 4.4, which shows that LHS = RHS.

A	B	$\overline{A+B}$	$\bar{A}$	$\bar{B}$	$\bar{A}\cdot\bar{B}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

LHS $\overline{A+B} = \bar{A}\cdot\bar{B}$ RHS

Fig. 4.4: Truth table to verify De-Morgan's theorem

Q. 3 Simplify y = AB $(\overline{CD})$ + $\bar{B}$CD + $(\bar{A}+\bar{C})$ (B + D) May 04, Dec. 14

Ans. : y = AB $(\overline{CD})$ + $\bar{B}$CD + $(\bar{A}+\bar{C})$ (B + D)

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

[$\because \overline{AB} = \bar{A}+\bar{B}$]

= AB$\bar{C}$ + AB$\bar{D}$ + $\bar{B}$CD + $\bar{A}$B + $\bar{A}$D + B$\bar{C}$ + $\bar{C}$D

= B$\bar{C}$ (A + 1) + AB$\bar{D}$ + D ($\bar{B}$C + $\bar{C}$) + $\bar{A}$B + $\bar{A}$D

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

[$\because$ 1 + A = 1 and A + $\bar{A}$B = A + B]

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

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

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

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

= B $\overline{ACD}$ + D $\overline{ABC}$ [$\because \bar{A}+\bar{B} = \overline{A\cdot B}$]

Q. 4 A misguided mathematician would like to subtract term A$\bar{C}$ form both sides of equality.

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

Would they still be equal if he did so. Justify.

Simplify the expression : F = (X + $\bar{Z}$) ($\overline{Z+WY}$) + (VZ + W$\bar{X}$) ($\overline{Y+Z}$) Dec. 04, Dec. 14

Ans. : BC + ABD + A$\bar{C}$ = BC + A$\bar{C}$

Subtracting A$\bar{C}$ from both sides

BC + ABD + A$\bar{C}$ − A$\bar{C}$ = BC + A$\bar{C}$ − A$\bar{C}$

BC + ABD = BC

LHS ≠ RHS

Given : F = (X + $\bar{Z}$) ($\overline{Z+WY}$) + (VZ + W$\bar{X}$) ($\overline{Y+Z}$)

= (X + $\bar{Z}$) ($\bar{Z}$ · $\overline{WY}$) + (VZ + W$\bar{X}$) ($\bar{Y}$ $\bar{Z}$)

[$\because \overline{A+B} = \bar{A}\cdot\bar{B}$]

= (X + $\bar{Z}$) [$\bar{Z}$ · ($\bar{W}$ + $\bar{Y}$)] + VZ$\bar{Y}\bar{Z}$ + W$\bar{X}\bar{Y}\bar{Z}$

= (X + $\bar{Z}$) [$\bar{W}\bar{Z}$ + $\bar{Y}\bar{Z}$] + 0 + W$\bar{X}\bar{Y}\bar{Z}$ = $\bar{Z}$

[X$\bar{W}$ + $\bar{W}$ + X$\bar{Y}$ + $\bar{Y}$ + W$\bar{X}$ $\bar{Y}$]

= $\bar{Z}$ [$\bar{W}$ (X + 1) + $\bar{Y}$ (X + 1) + W$\bar{X}$ $\bar{Y}$]

= $\bar{Z}$ [$\bar{W}$ + $\bar{Y}$ + W$\bar{X}$ $\bar{Y}$] [$\because$ 1 + A = 1]

= $\bar{Z}$ [$\bar{W}$ + $\bar{Y}$ (1 + W$\bar{X}$)] = $\bar{Z}$ [$\bar{W}$ + $\bar{Y}$]

Q. 5 Simplify Y = $\overline{(A+\bar{A}B)(C+\bar{D})}$. May 15

Ans. :

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

= (A + $\bar{A}$B) + (C + $\bar{D}$) ...(Using De-morgan's theorem)

= [$\bar{A}\cdot\overline{\bar{A}B}$] + [$\bar{C}\cdot\bar{\bar{D}}$] ...(Using De-morgan's theorem)

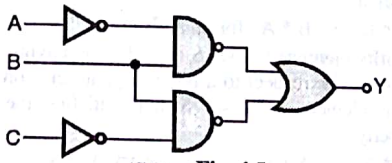
= $\bar{A}\cdot[\bar{\bar{A}}\cdot\bar{B}]$ + [$\bar{C}\cdot$ D] ...(Using De-morgan's theorem)

= $\bar{A}\cdot$[A$\bar{B}$] + $\bar{C}$D ... ($\because \bar{\bar{A}}$ = A)

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

Y = $\bar{C}$D ... ($\because$ A$\bar{A}$ = 0)

Q. 6 Determine the truth table for the circuit given in Fig. 4.5. Dec. 11



(C-2197) Fig. 4.5

Ans. :

Inputs			Output
A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Q. 7 Simply using Boolean theorems and implement using AOI gates only. Dec. 07, Dec. 14

1. **(C + $\bar{C}$ D) (C + $\bar{C}$ D) [(AB + $\bar{A}$ $\bar{B}$) + (A ⊕ B)]**
2. **$\overline{AB + \bar{A}\,\bar{B}}$ + $\overline{(A+B)\cdot\bar{A}+B}$**

Ans. :

1. $[(C + \bar{C}D)(C + \bar{C}\bar{D})]\ [(AB + \bar{A}\bar{B}) + (A \oplus B)]$:

$= [CC + C\bar{C}\bar{D} + C\bar{C}D + \bar{C}\bar{C}D\bar{D}]$
$[(AB + \bar{A}\bar{B})(\bar{A}B + A\bar{B})]$

$= [C + 0 + 0 + 0]$
$[A\bar{A}BB + AAB\bar{B} + \bar{A}\bar{B}\bar{A}B + \bar{A}\bar{B}A\bar{B}]$

$= C[0 + 0 + 0 + 0]\ [\because A\bar{A} = 0]$

$= 0 \qquad [\because C \cdot 0 = 0]$

2. $\overline{\overline{AB + \bar{A}\bar{B}} + (A + B) \cdot (\bar{A} + \bar{B})}$:

$= \overline{\overline{AB} \cdot \overline{\bar{A}\bar{B}} + \overline{A + B} + \bar{A} + \bar{B}}$

$\left[\because \overline{A + B} = \bar{A} \cdot \bar{B} \text{ and } \overline{A \cdot B} = \bar{A} + \bar{B}\right]$

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

$\left[\because \overline{A \cdot B} = \bar{A} + \bar{B} \text{ and } \overline{A + B} = \bar{A} \cdot \bar{B}\right]$

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

$[\because \bar{\bar{A}} = A]$

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

$= 0 + \bar{A}B + A\bar{B} + 0 + \bar{A}B$

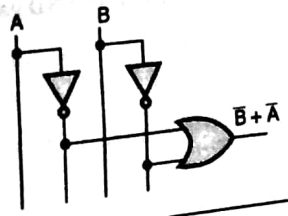
$[\because A\bar{A} = 0,\ A + A = A]$

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

$= \bar{A}B + \bar{B} \qquad [\because A + \bar{A} = 1]$

$= \bar{B} + \bar{A} \qquad [\because \bar{A} + A\bar{B} = \bar{A} + \bar{B}]$

or



Q. 8 Prove using Boolean law prove NAND as universal gate. Dec. 08, May 09, Dec. 10, Dec. 16

Ans. :

1. NOT gate (Inverter) using NAND gate : Fig. 4.6(a) shows the realization of a NOT gate (inverter) using a two input NAND gate. As both the inputs of the NAND are connected together, we can write that,

Input = $A = B = A$

So output is given by,

$Y = \overline{A \cdot B} = \overline{A \cdot A}$ ...since $A = B = A$

But $A \cdot A = A$ $\therefore Y = \bar{A}$

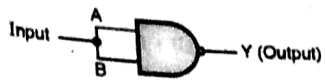


Fig. 4.6(a) : NOT using NAND

This expression is the Boolean expression for an inverter. Hence Fig. 4.6(a) is an inverter.

2. **AND gate using NAND :**

$Y = A \cdot B$ ...AND gate

$Y = \overline{\overline{A \cdot B}}$ ...Double inversion

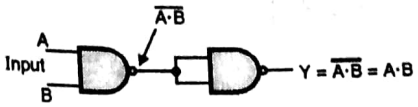


Fig. 4.6(b) : AND gate using NAND

$\therefore Y = A \cdot B = \overline{\overline{A \cdot B}}$

Equation (4.4) can be realized using only NAND gates as shown in Fig. 4.6(b).

3. **OR gate using NAND**

$Y = A + B$ ... OR gate

$Y = \overline{\overline{A + B}}$ ... Double inversion

$\therefore Y = \overline{\bar{A} \cdot \bar{B}}$ ... De Morgan's law

This is the required expression.

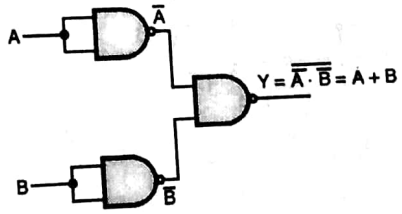


Fig. 4.6(c) : OR gate using NAND gates

So $Y = A + B = \overline{\bar{A} \cdot \bar{B}}$ and Fig. 4.6(c) shows the realization.

4. **NOR gate using NAND**

$Y = \overline{A + B}$ ... NOR gate

$= \bar{A} \cdot \bar{B}$ ...as per De Morgan's theorem

$Y = \overline{\overline{\bar{A} \cdot \bar{B}}}$ ...Double inversion

This is the required expression. Fig. 4.6(d) shows the realization of NOR gate using NAND gates.

$\therefore \overline{A + B} = \overline{\overline{\bar{A} \cdot \bar{B}}}$

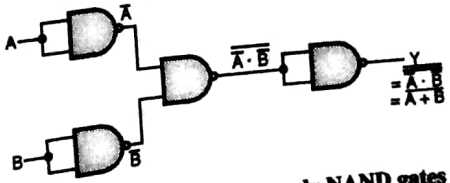


Fig. 4.6(d) : NOR gate using only NAND gates

5. **EX-OR gate using NAND :**

$Y = A \oplus B = \bar{A}B + A\bar{B}$ ... EX-OR gate

$Y = \overline{\overline{\bar{A}B + A\bar{B}}}$ ...Double inversion

Let $X = \bar{A}B$ and $Z = A\bar{B}$. $\therefore Y = \overline{\overline{X + Z}}$

Using De Morgan's theorem $\overline{X + Z} = \bar{X} \cdot \bar{Z}$

$$\therefore Y = \overline{\bar{X} \cdot \bar{Z}} = (\overline{\bar{A}B}) \cdot (\overline{A\bar{B}})$$

This is the required expression for EX-OR using only NAND. Fig. 4.6 (e) shows the realization.

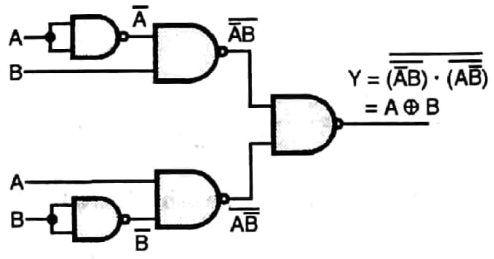


Fig. 4.6(e) : EX-OR using NAND gates

6. **EX-NOR using NAND :**

$Y = A \odot B = \bar{A}\bar{B} + AB = X + Z$...EX-NOR gate

Taking double inversion of RHS,

$Y = \overline{\overline{X + Z}} = \overline{\bar{X} \cdot \bar{Z}}$ ...De Morgan's theorem

$\therefore Y = (\overline{A\bar{B}}) \cdot (\overline{\bar{A}B})$

This is the required expression for EX-NOR in terms of only NAND gates. Fig. 4.6(f) shows the realization.

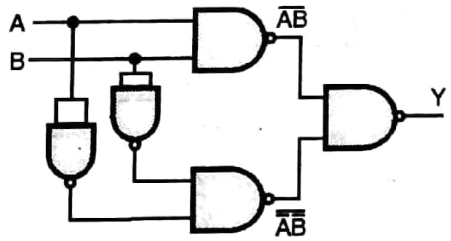


Fig. 4.6(f) : EX-NOR using only NAND gates

Q. 9 Implement Ex-OR gate using NOR gate only.

May 16

Ans. : The implementation of every logic operation using only NOR gates. In order to do so, the Boolean expression of the given logic circuit must be first converted into the NOR format which is as follows,

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

1. **NOT gate (inverter) using NOR :** Fig. 4.7 (a) shows the realization of a NOT gate using only NOR gate. As both the inputs of the NOR gate are connected together, we can write that,

$$A = B = A$$

So output of NOR is given by,

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

But $A + A = A$

$$\therefore Y = \bar{A}$$

Fig. 4.7(a) : NOT gate using NOR gate

This is the Boolean expression of an inverter. So Fig. 4.7(a) indeed represents an inverter.

2. **OR using NOR**

$Y = A + B$ ... OR gate

$Y = \overline{\overline{A + B}}$ ... Double inversion

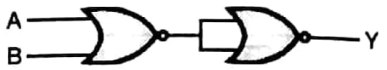


Fig.4.7(b) : OR using only NOR gates

This is the required expression.

Fig. 4.7 (b) shows the realization of OR gate using only NOR gates.

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

3. AND using only NOR gates :

$Y = A \cdot B$ ... AND gate

$Y = \overline{\overline{A \cdot B}}$ ...Double inversion

$\therefore Y = \overline{\bar{A} + \bar{B}}$ ...De Morgan's theorem

This is the required expression, Fig. 4.7(c) shows the realization of AND gate using NOR gates only.

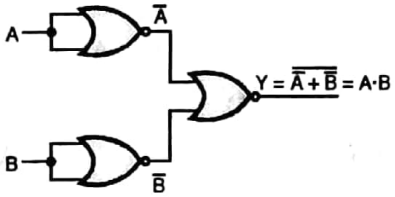


Fig. 4.7(c) : AND gate using only NOR gates

4. **NAND gate using only NOR gates**

Boolean expression for a NAND gate is,

$Y = \overline{A \cdot B} = \bar{A} + \bar{B}$ ...using De Morgan's theorem

Take double inversion of RHS to get,

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

This is the required expression. Fig. 4.7(d) shows the realization of NAND gate using only NOR gates.

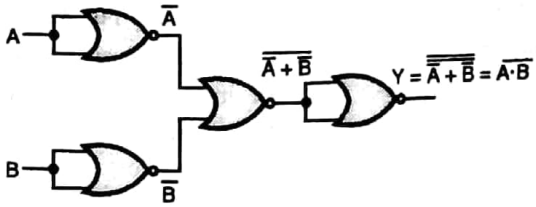


Fig. 4.7(d) : NAND gate using only NOR gates

5. EX-OR using only NOR :

$$Y = A \oplus B = \bar{A}B + A\bar{B}$$

Let $X = \bar{A}B$ and $Z = A\bar{B}$ ...EX-OR gate

$$\therefore Y = X + Z$$

$$Y = \overline{\overline{X+Z}}$$

$$\therefore Y = \overline{\overline{X}\cdot\overline{Z}}$$ ...Double inversion.

$$= \overline{(\overline{\bar{A}B})\cdot(\overline{A\bar{B}})}$$ ...De Morgan's theorem

But $\overline{\bar{A}B} = (A + \bar{B})$ and $\overline{A\bar{B}} = (\bar{A} + B)$

$$\therefore Y = \overline{(A+\bar{B})\cdot(\bar{A}+B)}$$ ...De Morgan's theorem

$$\therefore Y = \overline{(A+\bar{B})} + \overline{(\bar{A}+B)}$$

Take double inversion of RHS we get, ...De Morgan's theorem

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

This is the required expression. Fig. 4.7(e) shows the realization of EX-OR gate using only NOR gates.

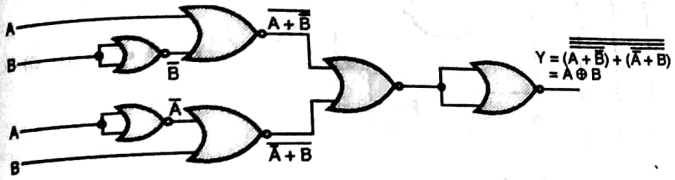


Fig. 4.7(e) : EX-OR using only NOR gates

6. EX-NOR using only NOR gates :

Boolean expression for an EX-NOR gate is,

$$Y = \bar{A}\bar{B} + AB = X + Z$$

Take double inversion of RHS to get,

$$Y = \overline{\overline{\bar{A}\bar{B} + AB}} = \overline{\overline{(\bar{A}\bar{B})}\cdot\overline{(AB)}}$$ ...De Morgan's theorem

$$\therefore Y = \overline{(A+B)\cdot(\bar{A}+\bar{B})}$$ ...Using De Morgan's theorem

$$\therefore Y = \overline{(A+B)} + \overline{(\bar{A}+\bar{B})}$$

...Using De Morgan's theorem

Take double inversion of RHS to get,

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

This is the required expression. Fig. 4.7(f) shows the realization of EX-NOR gate using only NOR gates.

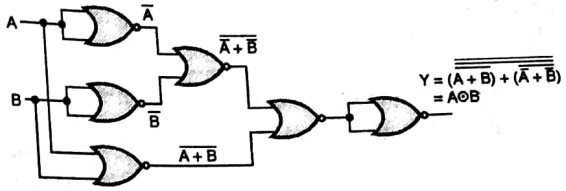


Fig. 4.7(f) : EX-NOR using only NOR gates

Q. 10 Realize $y = AB + \overline{AB}$ using NAND gates only. May 15

Ans. :

$$Y = A\cdot B + \overline{AB}$$

Taking double inversion of RHS,

$$= \overline{\overline{AB}\cdot\overline{\overline{AB}}}$$

Implementation using NAND gates is as shown in Fig. 4.8

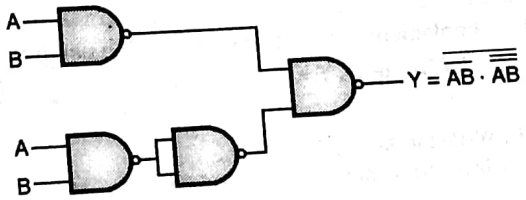


Fig. 4.8 : Implementation using NAND gates

Chapter 5 : Logic Minimization and Reduction Techniques

Q. 1 Convert the expression $Y = (A + B)(A + C)(B + \bar{C})$ into standard POS form. Dec. 13

Ans. : Step 1 : Find the missing literal for each term :

$$Y = \underbrace{(A+B)}_{C}\ \underbrace{(A+C)}_{B}\ \underbrace{(B+\bar{C})}_{A}$$ (C-2384)

Missing literal →

Step 2 : OR each term with (Missing literal. Its complement) :

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

Missing literal ANDed with its complement

This term is ORed with the term formed by ANDing the missing literal with its complement.

Step 3 : Simplify the expression to get standard POS

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

But $A + BC = (A + B)(A + C)$

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

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

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

$\therefore Y = \underbrace{(A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+\bar{C})}_{\text{Each term contains all the literals}}$ Standard POS form

Q. 2 Express the equation in standard SOP form : $F(A, B, C) = \Pi M(0, 2, 5, 7)$. Dec. 16

Ans. : $F(A, B, C) = \Pi M(0, 2, 5, 7)$

$\therefore$ Equation in standard SOP form is $\Sigma m(1, 3, 4, 6)$

Q. 3 Simplify following function using K-map.

$F(A, B, C, D) = \Sigma m(1, 2, 3, 4, 6, 8, 10, 14, 15)$ May 14

Ans. : $F(A, B, C, D) = \sum m(1, 2, 3, 4, 6, 8, 10, 14, 15)$

Simplification using K-map is as shown in Fig.5.1.

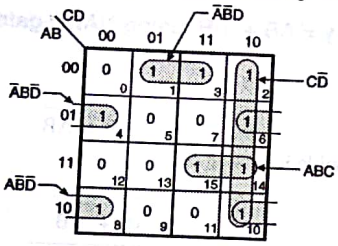


Fig. 5.1 : K-map

$\therefore F(A, B, C, D) = \bar{A}\bar{B}D + C\bar{D} + A\bar{B}\bar{D} + ABC + \bar{A}B\bar{D}$

...Ans.

Q. 4 Implement the following expression using NAND - NAND logic, $Y = \Sigma m(0, 1, 5)$ Dec. 14

Ans. :

Step 1 : Write the K-map :

Obtain the simplified expression for Y.

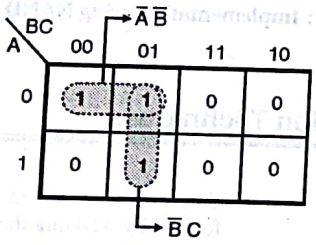


Fig.5.2 : K-map for given expression

$\therefore Y = \bar{A}\bar{B} + \bar{B}C$

Step 2 : Implement using AND–OR–NOT logic :

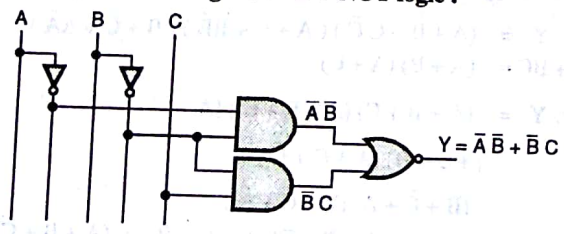


Fig. 5.2(a) : AND – OR - NOT implementation

Step 3 : Convert AND–OR–NOT into NAND–NAND logic :

Replace every AND by NAND, every OR by a bubbled OR and every inverter by a NAND inverter to get the NAND - NAND logic shown in Fig. 5.2(b).

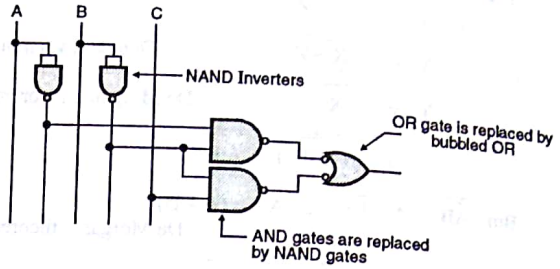


Fig. 5.2(b)

Step 4 : Draw the circuit using only NAND gates

The circuit using only NAND gates is shown in Fig. 5.2(c).

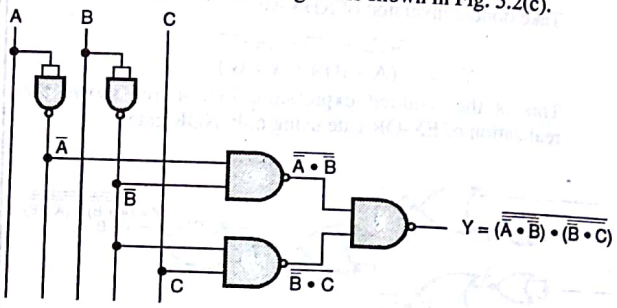


Fig.5.2(c) : Implementation using NAND-NAND logic

Q. 5 Simplify using K-map, obtain SOP equation and realize using only NAND gates.

$f(A, B, C, D) = \Pi M(1, 2, 3, 8, 9, 10, 11, 14) + d(7, 15)$.

May 09, Dec. 11, Dec. 12, Dec. 14

Ans. :

$f(A, B, C, D) = \Pi M(1, 2, 3, 8, 9, 10, 11, 14) + d(7, 15)$

i.e. $f(A, B, C, D) = \Sigma m(0, 4, 5, 6, 12, 13) + d(7, 15)$

Step 1 : Simplification using K-map :

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

In SOP form :

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

$= \overline{\bar{A}B + B\bar{C} + \bar{A}\bar{C}\bar{D}} = \overline{\bar{A}B \cdot B\bar{C} \cdot \bar{A}\bar{C}\bar{D}}$

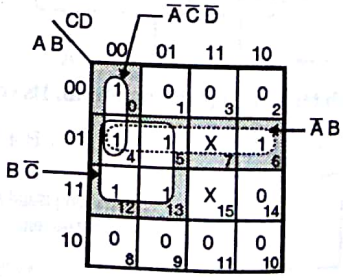


Fig. 5.3(a)

Step 2 : Implementation using NAND gates only :

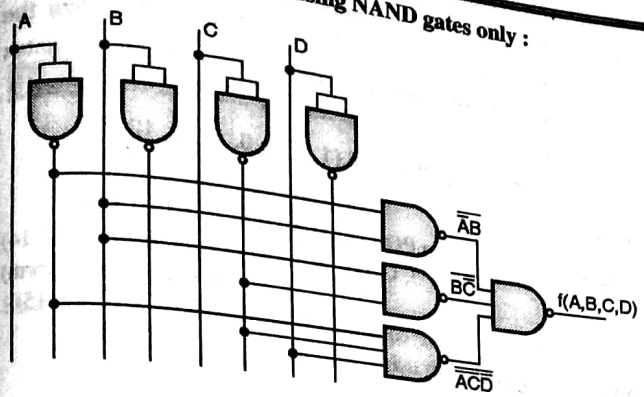


Fig. 5.3(b)

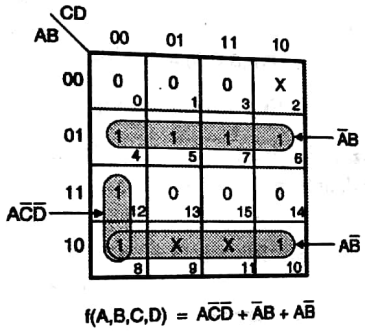
Q. 6 Simplify the logic function using K-map :

f (A, B, C, D) = Σm (4, 5, 6, 7, 8, 10, 12) + d (2, 9, 11)

Draw the logic diagram using NAND gates only.

Dec. 13

Ans. : Refer Fig. 5.4(a) for required K-map.



$f(A,B,C,D) = A\overline{CD} + \overline{A}B + A\overline{B}$

Fig. 5.4(a)

Implementation using NAND gates is as shown in Fig. 5.4(b). Taking double inversion of RHS,

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

$$= \overline{(\overline{A\overline{CD}}) \cdot (\overline{\overline{A}B}) \cdot (\overline{A\overline{B}})}$$

...(using De-Morgan's theorem)

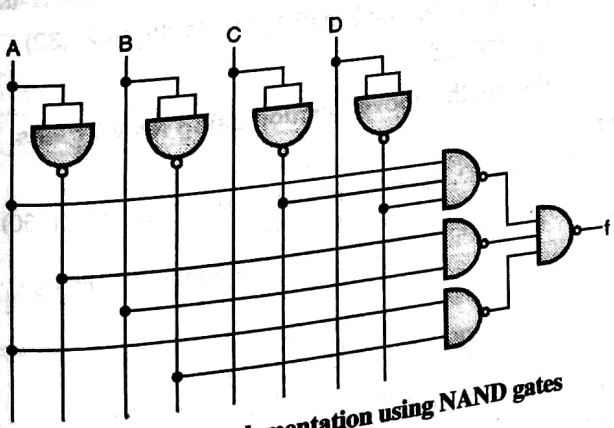


Fig. 5.4(b) : Implementation using NAND gates

Q. 7 Given the logic expression :

$A + \overline{B}\overline{C} + AB\overline{D} + ABCD$

May 06, May 15

1. **Express it in standard SOP form.**
2. **Draw K-map and simplify.**
3. **Draw logic diagram using NOR gates only.**
4. **Draw logic diagram using NAND gates only.**
5. **Express it in standard POS form.**

Ans. :

1. $Y = A + \overline{B}\,\overline{C} + AB\overline{D} + AB\overline{D} + ABCD$:

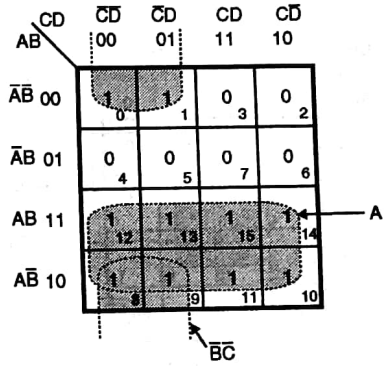
Standard SOP form

$$Y = A(B+\overline{B})(C+\overline{C})(D+\overline{D}) + \overline{B}\,\overline{C}(A+\overline{A})(D+\overline{D}) + AB\overline{D}(C+\overline{C}) + ABCD$$

$$Y = ABCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + A\overline{B}\overline{C}D + AB\overline{C}\overline{D} + A\overline{B}C\overline{D} + A\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}\overline{D} + ABC\overline{D} + AB\overline{C}\overline{D} + ABCD$$

$$Y = ABCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + A\overline{B}\overline{C}D + AB\overline{C}\overline{D} + A\overline{B}C\overline{D} + A\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D$$

2. K-map :



$\therefore Y = A + \overline{B}\,\overline{C}$ ...Ans.

Fig. 5.5(a)

3. Logic diagram using NOR gates only :

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

...According to De-Morgan's second theorem

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

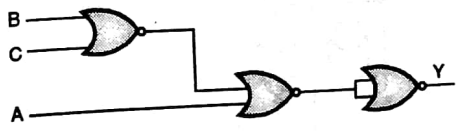


Fig. 5.5(b)

4. Logic diagram using NAND gates only :

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

According to De-Morgan's theorem

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

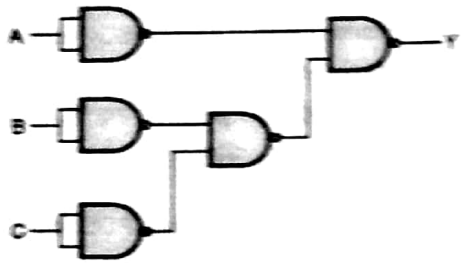


Fig. 5.5(c)

5. Expression in standard POS form

$$Y = (A + B + \bar{C} + \bar{D})(A + B + \bar{C} + D)$$
$$(A + \bar{B} + C + D)(A + \bar{B} + C + \bar{D})$$
$$(A + \bar{B} + \bar{C} + \bar{D})(A + \bar{B} + \bar{C} + D) \quad \text{...Ans.}$$

Q. 8 Simplify the following equation using K-map to obtain minimum POS equation and realize the minimum equation using only NOR gates.

$F(A, B, C, D) = \sum m (1, 3, 4, 6, 9, 11, 12, 14)$

Dec. 13

Ans. :

Step 1 : Simplification using K-map :

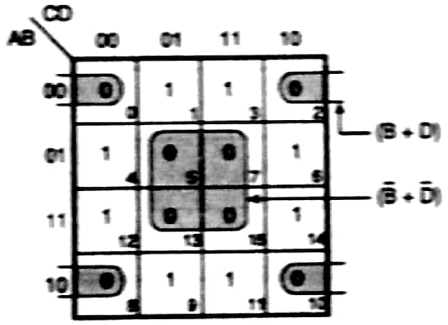


Fig. 5.6(a) : K-map

$\therefore f(A, B, C, D) = (B + D) + (\bar{B} + \bar{D})$ ...(1)

This is required expression in POS form.

Step 2 : Implementation using NOR gates :

Taking double inversion of R.H.S.

$$f(A, B, C, D) = \overline{\overline{(B + D)(\bar{B} + \bar{D})}} = \overline{B + D \cdot \bar{B} + \bar{D}}$$

...(Using De-Morgan's theorem)

Implementation using NOR gates is as shown in Fig. 5.6(b).

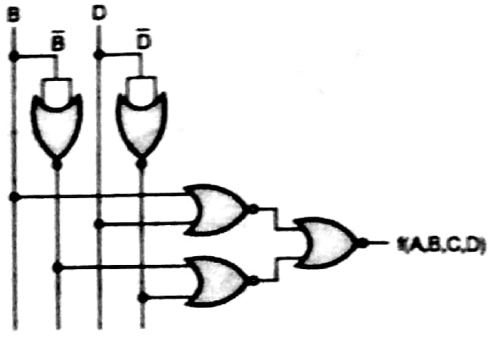


Fig. 5.6(b) : Implementation using NOR gates

Q. 9 Simplify the following equation using K-map to obtain minimum SOP equation and realize the minimum equation using two level NAND gates only.

Dec. 15

$F(A, B, C, D) = \Pi M (1, 3, 4, 6, 9, 11, 12, 14)$

Ans. :

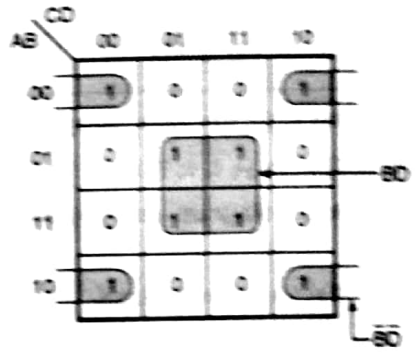
Step 1 : Convert given POS equation into standard SOP :

$F(A, B, C, D) = \Pi M (1, 3, 4, 6, 9, 11, 12, 14)$

...(Given standard POS form)

$\therefore$ Standard SOP equation is $\sum m (0, 2, 5, 7, 8, 10, 13, 15)$

Step 2 : Simplification using K-map :



$\therefore F(A,B,C,D) = BD + B\bar{D}$

Fig. 5.7(a) : K-map

Step 3 : Realization using two level NAND gates :

Taking double inversion of R.H.S.

$$= \overline{\overline{BD} + \overline{B}\,\overline{D}} = \overline{\overline{BD} \cdot \overline{\overline{B}\,\overline{D}}}$$

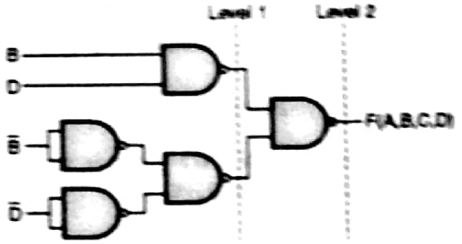


Fig. 5.7(b) : Realization using two level NAND gates

Q. 10 Use Quine Mc-Cluskey method to simplify the logic function as given below.

Dec. 18

$f(A, B, C, D, E) = \sum m (0,1,8,10,11,12,20,21,30) + d (14, 19)$

Realize the above function using NAND gates.

Ans. :

$f(A, B, C, D, E) = \sum m (0, 1, 8, 10, 11, 12, 20, 21, 30) + d (14, 19)$

Step 1 : Group the minterms according to number of 1's :

Group	Min term	Binary representation A	B	C	D	E	
0	0	0	0	0	0	0	✓
1	1	0	0	0	0	1	✓
	8	0	1	0	0	0	✓
	10	0	1	0	1	0	✓

Q. 9 **Simplify the following equation using K-map to obtain minimum SOP equation and realize the minimum equation using two level NAND gates only.** Dec. 16

$F(A, B, C, D) = \Pi M (1, 3, 4, 6, 9, 11, 12, 14)$

Ans. :

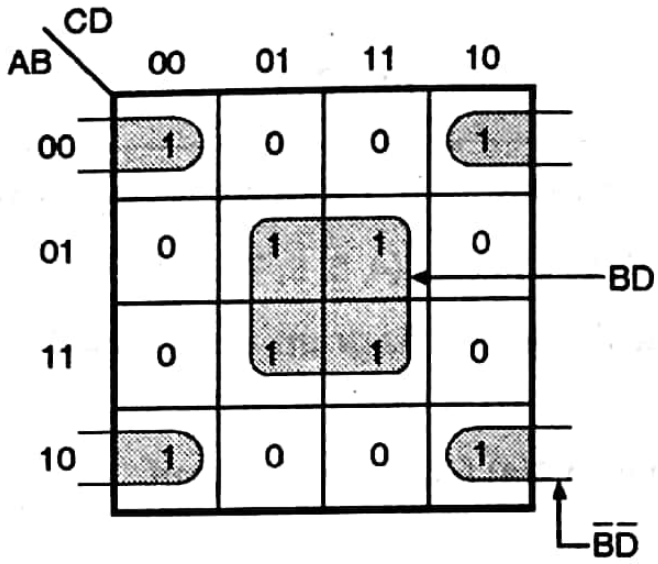
Step 1 : Convert given POS equation into standard SOP :

$$F(A, B, C, D) = \Pi M (1, 3, 4, 6, 9, 11, 12, 14)$$

...(Given standard POS form)

$\therefore$ Standard SOP equation is $\Sigma m (0, 2, 5, 7, 8, 10, 13, 15)$

Step 2 : Simplification using K-map :



$\therefore F(A,B,C,D) = BD + \bar{B}\bar{D}$

Fig. 5.7(a) : K-map

Step 3 : Realization using two level NAND gates :

Taking double inversion of R.H.S.,

$$= \overline{\overline{BD + \bar{B}\,\bar{D}}} = \overline{\overline{BD} \cdot \overline{\bar{B}\,\bar{D}}}$$

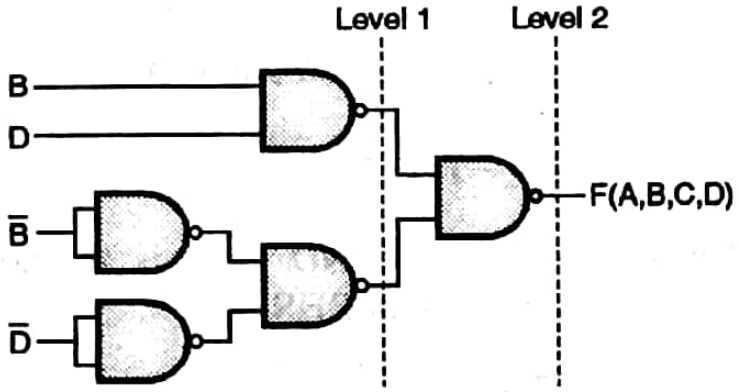


Fig. 5.7(b) : Realization using two level NAND gates

Group	Min term	Binary representation A	B	C	D	E	
2	12	0	1	1	0	0	✓
	20	1	0	1	0	0	✓
3	11	0	1	0	1	1	✓
	14	0	1	1	1	0	✓
	19	1	0	0	1	1	$A\bar{B}\bar{C}DE$
	21	1	0	1	0	1	✓
4	30	1	1	1	1	0	$ABCD\bar{E}$

Step 2 : Group the minterms to form pairs :

Group	Min term	Binary representation A	B	C	D	E	
0	0–1	0	0	0	0	–	$\bar{A}\bar{B}\bar{C}\bar{D}$
	0–8	0	–	0	0	0	$\bar{A}\bar{C}\bar{D}\bar{E}$
1	8–10	0	1	0	–	0	✓
	8–12	0	1	–	0	0	✓
2	10–11	0	1	0	1	–	$\bar{A}B\bar{C}D$
	10–14	0	1	–	1	0	✓
	20–21	1	0	1	0	–	$A\bar{B}\bar{C}D$
	12–14	0	1	1	–	0	✓
3	14–30	–	1	1	1	0	$BCD\bar{E}$

Step 3 : Group the minterms to form groups of four :

Group	Minterm quad	Binary representation A	B	C	D	E	
1	8–10–12–14	0	1	–	–	0	$\bar{A}B\bar{E}$
	8–12–10–14	0	1	–	–	0	

Step 4 : Prepare the table prime implicants :

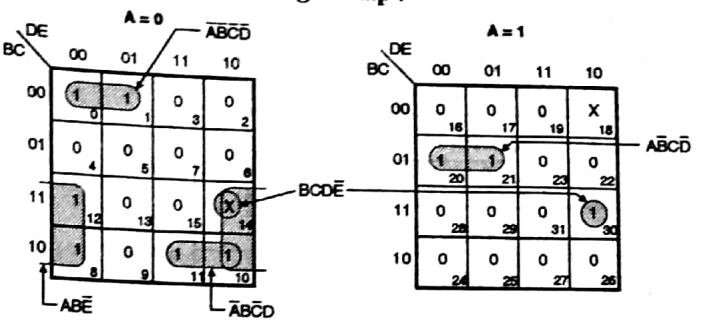
Prime implicants	Decimal numbers	Given minterms 0	1	8	10	11	12	20	21	30	14	19
$A\bar{B}\bar{C}DE$	19											(X)
$ABCD\bar{E}$	30									X		
$\bar{A}\bar{B}\bar{C}\bar{D}$	0, 1	X	(X)									
$\bar{A}\bar{C}\bar{D}\bar{E}$	0, 8	X		X								
$\bar{A}B\bar{C}D$	10, 11				X	(X)						
$A\bar{B}\bar{C}D$	20, 21							(X)	(X)			
$BCD\bar{E}$	14, 30									X	X	
$\bar{A}B\bar{E}$	8, 10, 12, 14			X	X		(X)					

(C-4211)

The encircled crosses represent the essential prime implicants. They are $\bar{A}\bar{B}\bar{C}\bar{D}$, $\bar{A}B\bar{C}D$, $A\bar{B}\bar{C}D$, $\bar{A}B\bar{E}$ which cover minterms. 0, 1, 8, 10, 11, 12, 20, 21. $BCD\bar{E}$ cover minterm 30.

$$\therefore \quad f(A, B, C, D, E) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + A\bar{B}\bar{C}D + \bar{A}B\bar{E} + BCD\bar{E}$$

Step 5 : Cross check using K-map :



$\therefore$ $f(A,B,C,D,E) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + A\bar{B}\bar{C}D + \bar{A}B\bar{E} + BCD\bar{E}$

Fig.5.8

Q. 11 Using Quine Mc Cluskey method determine minimal SOP form for May 14

F(A, B, C, D) = $\sum$ m (1, 2, 3, 6, 7, 10, 12, 14)

Ans. :

(A, B, C, D) = $\sum$ m (1, 2, 3, 6, 7, 10, 12, 14)

Step 1 : Group the minterms according to number of 1's :

Group	Minterm	Binary representation A	B	C	D	
1	1	0	0	0	1	✓
	2	0	0	1	0	✓
2	3	0	0	1	1	✓
	6	0	1	1	0	✓
	10	1	0	1	0	✓
	12	1	1	0	0	✓
3	7	0	1	1	1	✓
	14	1	1	1	0	✓

Step 2 : Group the minterms to form pairs :

Group	Minterm pair	Binary representation A	B	C	D	
1	1-3	0	0	–	1	$\bar{A}\bar{B}$
	2-3	0	0	1	–	$\bar{A}\bar{B}$
	2-6	0	–	1	0	✓
	2-10	–	0	1	0	✓
2	3-7	0	–	1	1	$\bar{A}CD$
	6-14	–	1	1	0	✓
	10-14	1	–	1	0	✓
	12-14	1	1	–	0	$AB\bar{D}$
	6-7	0	1	1	–	

Step 3 : Group the minterms to form groups of four :

Group	Minterm quad	Binary representation				
		A	B	C	D	
1	2-6-10-14	–	–	1	0	$C\bar{D}$
	2-10-6-14	–	–	1	0	
	2-6-3-7	0	–	1	–	$\bar{A}C$
	2-3-6-7	0	–	1	–	

Step 4 : Prepare table of prime implicants :

Prime implicants	Decimal numbers	Given minterms							
		1	2	3	6	7	10	12	14
$\bar{A}\bar{B}D$	1, 3	(X)		X					
$\bar{A}\bar{B}C$	2, 3		X	X					
$\bar{A}CD$	3, 7			X		X			
$AB\bar{D}$	12, 14							(X)	X
$C\bar{D}$	2, 6, 10, 14		X		X		(X)		X
$\bar{A}C$	2, 3, 6, 7		X	X	X	X			

(C-4963)

The encircled crosses represent the essential prime implicants. They are $\bar{A}\bar{B}D$, $C\bar{D}$, $AB\bar{D}$ which cover the minterms 1, 2, 3, 6, 10, 12, 14. Now check the remaining minterms. i.e. 7. The PI $\bar{A}C$ cover minterm 7. So they should be included.

$\therefore$ $F(A, B, C, D) = \bar{A}\bar{B}D + C\bar{D} + AB\bar{D} + \bar{A}C$

Step 5 : Crosscheck using K-map :

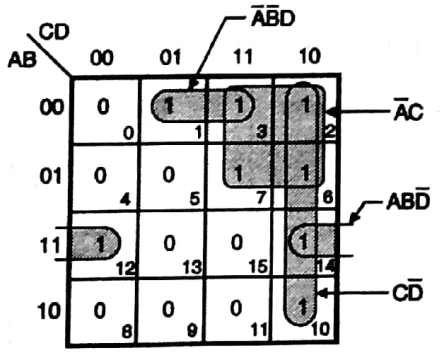


Fig. 5.9 : K-map

$\therefore$ $f(A, B, C, D) = \bar{A}\bar{B}D + C\bar{D} + AB\bar{D} + \bar{A}C$

Q. 12 Minimize the logic function using Quine-Mc Cluskey method.

$f(A, B, C, D) = \sum m(1, 3, 7, 9, 10, 11, 13, 15)$

May 15, May 16

Ans. :

Step 1 : Group the minterms according to number of 1's :

Table 5.1

Group	Minterm	Binary representation				
		A	B	C	D	
1	1	0	0	0	1	One 1
2	3	0	0	1	1	Two 1's
	9	1	0	0	1	
	10	1	0	1	0	
3	7	0	1	1	1	Three 1's
	11	1	0	1	1	
	13	1	1	0	1	
4	15	1	1	1	1	Four 1's

Step 2 : Group the minterms to form the pairs :

Table 5.1(a)

Group	Minterm pair	Binary representation				
		A	B	C	D	
1	1 – 3	0	0	–	1	
	1 – 9	–	0	0	1	
2	3 – 7	0	–	1	1	
	9 – 11	1	0	–	1	
	9 – 13	1	–	0	1	
	10 – 11	1	0	1	–	$A\bar{B}C$
	3 – 11	–	0	1	1	
3	7 – 15	–	1	1	1	
	11 – 15	1	–	1	1	
	13 – 15	1	1	–	1	

Step 3 : Group the minterms to form groups of four :

Table 5.1(b)

Group	Minterm quad	Binary representation				Prime implicants
		A	B	C	D	
1	1, 3, 9, 11	–	0	–	1	$\bar{B}D$
	1, 9, 3, 11	–	0	–	1	
2	3, 7, 11, 15	–	–	1	1	CD
	3, 11, 7, 15	–	–	1	1	
	9, 11, 13, 15	1	–	–	1	AD
	9, 13, 11, 15	1	–	–	1	

Step 4 : Prepare the table of prime implicants :

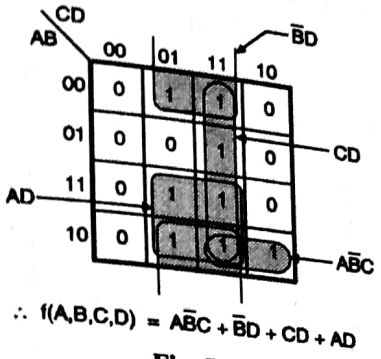
Table 5.1(c)

Prime implicants	Decimal numbers	Given minterms							
		1	3	7	9	10	11	13	15
$A\bar{B}C$	10, 11					(X)	X		
$\bar{B}D$	1, 3, 9, 11	(X)	X		X		X		
CD	3, 7, 11, 15		X	(X)			X		X
AD	9, 11, 13, 15				X		X	(X)	X

The encircled crosses represent the EPIs. They cover minterms 1, 3, 7, 9, 10, 11, 13 and 15.

$\therefore f(A, B, C, D) = A\bar{B}C + \bar{B}D + CD + AD$

Step 5 : Crosscheck using K-map :



$\therefore f(A,B,C,D) = A\bar{B}C + \bar{B}D + CD + AD$

Fig. 5.10

Q. 13 Reduce using Quine Mc-Cluskey method and realize the equation using only NAND gates.

$F(P, Q, R, S) = \Sigma m (0, 1, 2, 8, 10, 11, 14, 15)$

Dec. 15, Dec. 16

Ans. :

Step 1 : Group the minterms according to number of 1's :

Table 5.2(a)

Group	Minterm	Binary representation				
		P	Q	R	S	
0	0	0	0	0	0	No 1's
1	1	0	0	0	1	
	2	0	0	1	0	One 1
	8	1	0	0	0	
2	10	1	0	1	0	Two 1's
3	11	1	0	1	1	
4	14	1	1	1	0	Three 1's
	15	1	1	1	1	Four 1's

Step 2 : Group the minterms to form pairs :

Table 5.2(b)

Group	Minterm pair	Binary representation				
		P	Q	R	S	
0	0, 1	0	0	0	–	$\bar{P}\bar{Q}\bar{R}$
	0, 2	0	0	–	0	✓
	0, 8	–	0	0	0	✓
1	2, 10	–	0	1	0	✓
	8, 10	1	0	–	0	✓
2	10, 11	1	0	1	–	✓
	10, 14	1	–	1	0	✓
3	11, 15	1	–	1	1	...Ans. ✓
	14, 15	1	1	1	–	✓

Step 3 : Group the minterms to form quads :

Table 5.2(c)

Group	Minterm Quad	Binary representation				
		P	Q	R	S	
0	0, 2, 8, 10	–	0	–	0	$\bar{Q}\bar{S}$
	0, 8, 2, 10	–	0	–	0	
2	10, 11, 14, 15	1	–	1	–	PR
	10, 14, 11, 15	1	–	1	–	

Step 4 : Prepare the table of prime implicants :

Table 5.2(d)

Prime implicant	Decimal numbers	Given minterms							
		0	1	2	8	10	11	14	15
$\bar{P}\bar{Q}\bar{R}$	0, 1	X	(X)						
$\bar{Q}\bar{S}$	0, 2, 8, 10	X		(X)	(X)	X			
PR	10, 11, 14, 15					X	(X)	(X)	(X)

The encircled crosses represent the EPs. So the terms $\bar{P}\bar{Q}\bar{R}$, $\bar{Q}\bar{S}$ and PR are EPs. They cover all the minterms 0, 1, 2, 8, 10, 11, 14 and 15.

$\therefore f(P, Q, R, S) = \bar{P}\bar{Q}\bar{R} + \bar{Q}\bar{S} + PR$ ...Ans.

Step 5 : Crosscheck using K-map :

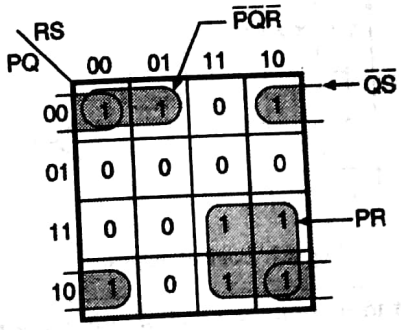


Fig. 5.11 : K-map

$\therefore f = (P, Q, R, S) = \bar{P}\bar{Q}\bar{R} + \bar{Q}\bar{S} + PR$

This is same as the result obtained with Quine Mc-Cluskey method.

Q. 1 **Design a 4-input (A, B, C, D) digital circuit that will give at its output (X) a logic 1 only if the binary number formed at the input is between 2 and 9 (including).** May 15

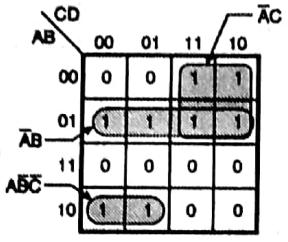
Ans. : **Step 1 : Write the truth table :**

Table 6.1

Decimal	Inputs				Output X
	A	B	C	D	
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Step 2 : K-map and simplification :

For X



$\therefore X = \bar{A}C + \bar{A}B + A\bar{B}\bar{C}$

Step 3 : Draw the logic diagram :

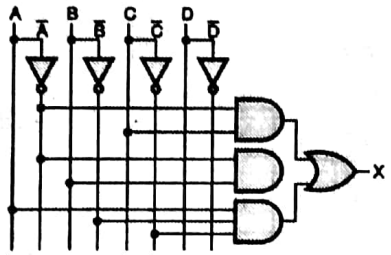


Fig. 6.1: Logical diagram

Q. 2 **Input to a combinational circuit is a 4-bit binary number. Design the circuit with minimum hardware for the following :** Dec 15

1. **Output P = 1 if the number is prime.**
2. **Output Q = 1 if the number is divisible by 3.**

Ans. : **Step 1 : Write the truth table :**

Prime numbers are : 1, 2, 3, 5, 7, 11 and 13.

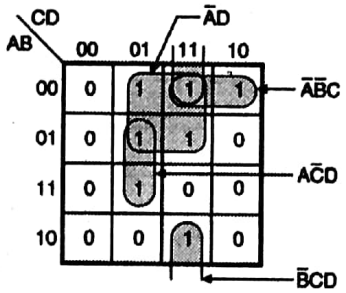
Numbers divisible by 3 are : 0, 3, 6, 9, 12, 15.

Table 6.2

Decimal	Inputs				Outputs	
	A	B	C	D	P	Q
0	0	0	0	0	0	0
1	0	0	0	1	1	0
2	0	0	1	0	1	0
3	0	0	1	1	1	1
4	0	1	0	0	0	0
5	0	1	0	1	1	0
6	0	1	1	0	0	1
7	0	1	1	1	1	0
8	1	0	0	0	0	0
9	1	0	0	1	0	1
10	1	0	1	0	0	0
11	1	0	1	1	1	0
12	1	1	0	0	0	1
13	1	1	0	1	1	0
14	1	1	1	0	0	0
15	1	1	1	1	0	1

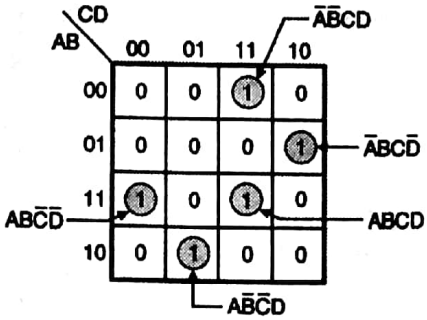
Step 2 : **K-maps and simplification :**

For output P



$\therefore P = \bar{A}\bar{B}C + \bar{A}D + A\bar{C}D + \bar{B}CD$

For output Q



$\therefore Q = \bar{A}\bar{B}CD + \bar{A}BC\bar{D} + ABCD + A\bar{B}\bar{C}D + AB\bar{C}\bar{D}$

Step 3 : Draw the logic diagram :

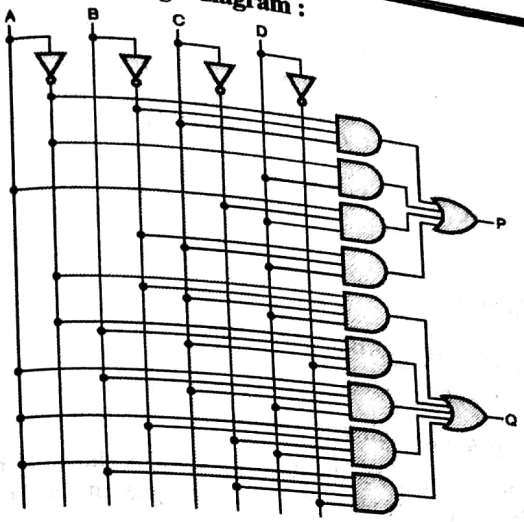


Fig. 6.2 : Logic diagram

Q. 3 Convert 4-bit gray code into corresponding BCD code. Show truth table and implement using gates. May 08. May 16

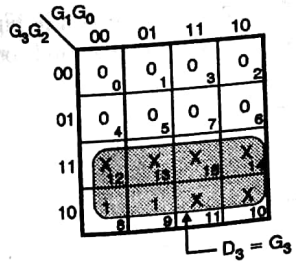
Ans. :

Step 1 : Write the truth table relating gray and BCD codes :

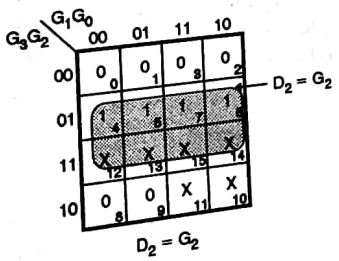
Decimal	Gray inputs				BCD outputs			
	G_3	G_2	G_1	G_0	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1

Step 2 : Write K-maps for each output and get simplified equation :

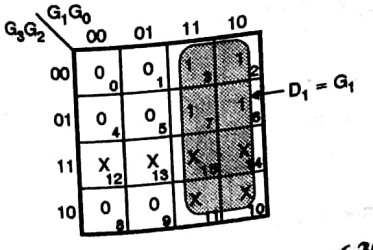
For Output D_3



For Output D_2



For Output D_1



For Output D_0

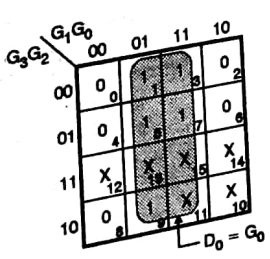
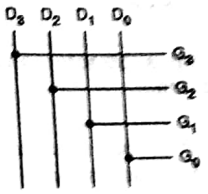


Fig. 6.3(a)

Step 3 : Realization :

The gray to BCD converter is shown in Fig. 6.3(b).



(C-1710) **Fig.6.3(b)**

Q. 4 Design a full adder circuit using half adders and some gates. May 11, Dec. 12, May 14, May 15

Ans. :

The full adder circuit can be constructed using two half adders as shown in Fig. 6.4(a) and the detail circuit is shown in Fig. 6.4(b).

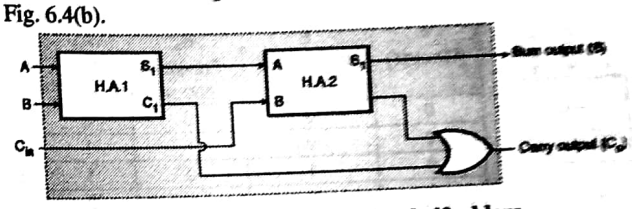


Fig. 6.4(a) : Full adder using half adders

A full adder can be implemented using two half adders and an OR gate as shown in Fig. 6.4(b).

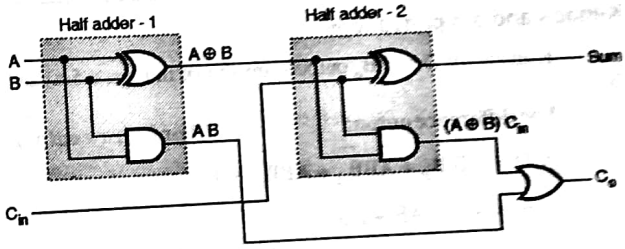


Fig. 6.4(b) : Full adder using two half adders

Now let us prove that this circuit acts as a full adder.

Proof :

Refer Fig. 6.4(b) and write the expression for sum output as,

$$S = (A \oplus B) \oplus C_{in} = A \oplus B \oplus C_{in}$$

This expression is same as that obtained for the full adder. Thus the sum output has been successfully implemented by the circuit shown in Fig. 6.4(b).

Now write the expression for carry output C_o as,

$$C_o = (A \oplus B)\, C_{in} + AB$$

$$C_o = (\bar{A}B + A\bar{B})\, C_{in} + AB = \bar{A}BC_{in} + A\bar{B}C_{in} + AB$$

$$= \bar{A}BC_{in} + A\bar{B}C_{in} + AB\,(1 + C_{in}) = \bar{A}BC_{in} + A\bar{B}C_{in} + AB + ABC_{in}$$

$$= BC_{in}(\bar{A} + A) + A\bar{B}C_{in} + AB = BC_{in} + A\bar{B}C_{in} + AB$$

$$= BC_{in} + A\bar{B}C_{in} + AB\,(1 + C_{in}) = BC_{in} + A\bar{B}C_{in} + AB + ABC_{in}$$

$$= BC_{in} + AB + AC_{in}\,(\bar{B} + B)$$

$\therefore C_o = BC_{in} + AB + AC_{in}$ ...Proved.

This expression is same as that for a full adder. Thus we have proved that circuit shown in Fig. 6.4(b) really behaves like a full adder.

Q. 5 Design a full subtractor and implement using basic logic gates. May 12

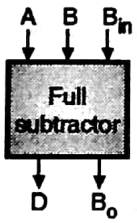
Ans. :

The disadvantage of a half subtractor is overcome if we use the full subtractor. The full subtractor is a combinational circuit with three inputs A, B and B_{in} and two outputs D and B_o. A is the minuend, B is subtrahend, B_{in} is the borrow produced by the previous stage, D is the difference output and B_o is the borrow output.

Truth table :

The truth table for full subtractor is shown in Table 6.3.

Table 6.3 : Truth table for a full subtractor



Inputs			Outputs	
A (Minuend)	B (Subtrahend)	B_{in} Previous borrow	(A – B – B_{in})	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

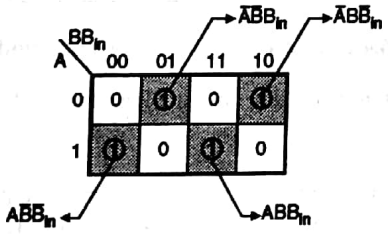
K-maps and simplifications :

K-maps for D and B_o outputs are shown in Figs. 6.5(a) and (b).

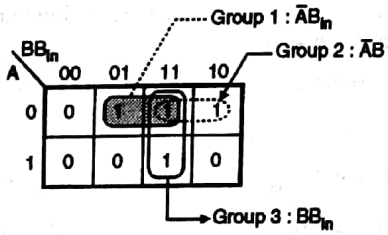
For difference output **For borrow output**

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

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



(a) K-map for D



(b) K-map for B_o

Fig. 6.5

Simplification for difference output :

From Fig. 6.5(a),

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

$$= B_{in}\underbrace{(\bar{A}\bar{B} + AB)}_{\text{EX-NOR}} + \bar{B}_{in}\underbrace{(A\bar{B} + \bar{A}B)}_{\text{EX-OR}}$$

$$\therefore D = B_{in}\overline{(A \oplus B)} + \bar{B}_{in}(A \oplus B)$$

Let $A \oplus B = C$, $\therefore D = B_{in}\bar{C} + \bar{B}_{in}C = B_{in} \oplus C$

$$\therefore D = B_{in} \oplus A \oplus B \quad ...(1)$$

Simplification for borrow output :

From Fig.6.6, $B_o = \bar{A}B_{in} + \bar{A}B + BB_{in}$ ...(2)

No further simplification is possible.

Logic diagram for full subtractor :

Logic diagram for the full subtractor is shown in Fig. 6.6. This has been drawn by using the Boolean equations of (1) and (2).

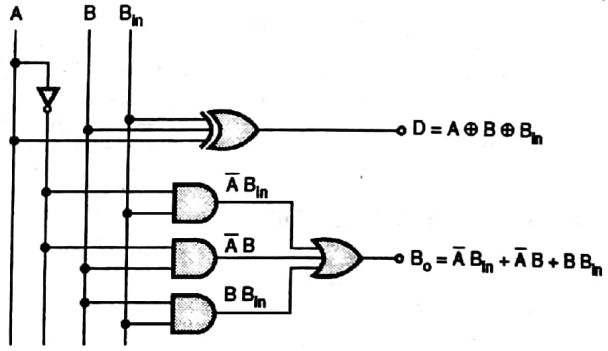


Fig.6.6 : Logic diagram for a full subtractor

Q. 6 Implement single digit BCD adder using 4-bit binary adder IC 7483. Show the design procedure and explain its operation. Dec. 15

Ans.: The 4-bit, BCD adder should consist of the following blocks

1. A 4-bit binary adder to add the given two 4-bit BCD numbers A and B.
2. A combinational circuit to check if sum is greater than 9 or carry = 1.
3. Another 4-bit binary adder to add six (0110) to the incorrect sum if sum > 9 or carry = 1.

The block diagram of such a BCD adder is shown in Fig. 6.7. So we have to design the combinational circuit that finds out whether the sum is greater than 9 or Carry = 1.

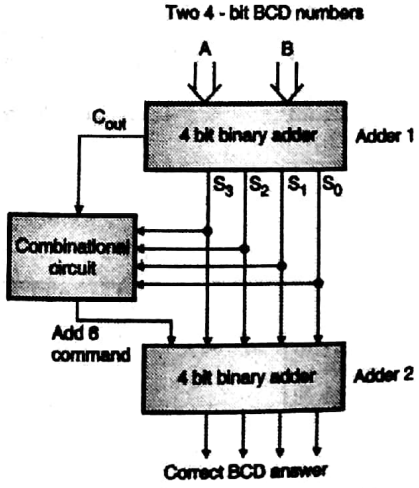


Fig.6.7 : Block diagram of BCD adder

Q. 7 Implement single digit BCD adder using 4-bit binary adder IC 7483. Show the design procedure and explain its operation. Dec. 15

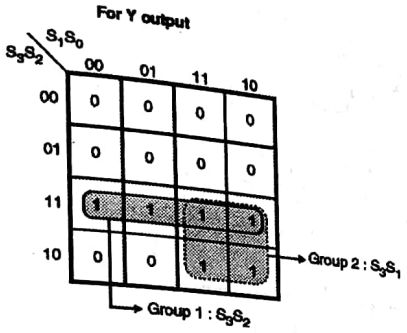
Ans. : The output of combinational circuit should be 1 if the sum produced by adder 1 is greater than 9 i.e. 1001. The truth table is as follows :

Table 6.4 : Truth table for combinational circuit design

Inputs				Output	
S_3	S_2	S_1	S_0	Y	
0	0	0	0	0	Sum is valid BCD number. ∴ Y = 0
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	1	Sum is invalid BCD number. ∴ Y = 1
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	1	

Write K-map :

Sum bits of adder-1 →



(C-398) **Fig. 6.8(a) : K-map for Y output**

The Boolean expression is,

$$Y = S_3S_2 + S_3S_1$$

The complete BCD adder is shown in Fig. 6.8(b). The output of the combinational circuit should be 1 if C_{out} of adder-1 is high or if the output of adder-1 is greater than 9. Therefore Y is ORed with C_{out} of adder 1 as shown in Fig 6.8(b). The output of combinational circuit is connected to B_1B_2 inputs of adder-2 and $B_3 = B_1 = 0$ as they are connected to ground permanently. This makes $B_3\ B_2\ B_1\ B_0 = 0\ 1\ 1\ 0$ if Y′ = 1. The sum outputs of adder-1 are applied to $A_3A_2A_1A_0$ of adder-2. The output of combinational circuit is to be used as final output carry and the carry output of adder-2 is to be ignored.

Operation :

Case I : Sum ≤ 9 and carry = 0

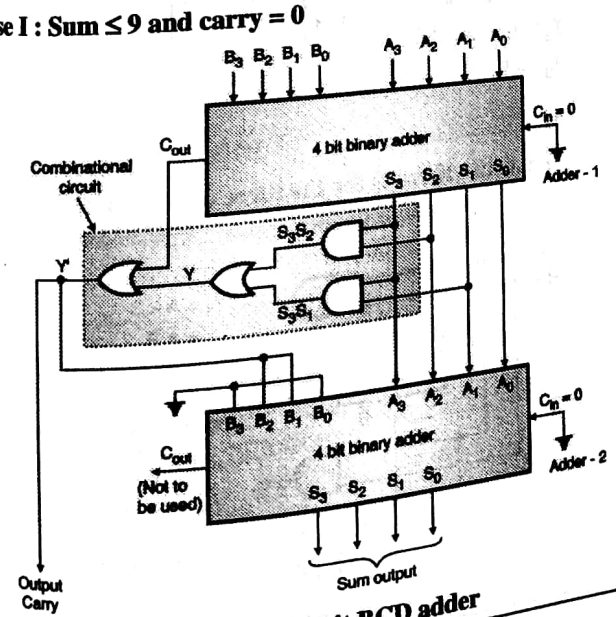


Fig. 6.8(b) : 4-bit BCD adder

The output of combinational circuit Y′ = 0. Hence $B_3\ B_2\ B_1\ B_0 = 0000$ for adder-2. Hence output of adder-2 is same as that of adder-1.

Case II : Sum > 9 and carry = 0

If $S_3\ S_2\ S_1\ S_0$ of adder-1 is greater than 9, then output Y′ of combinational circuit becomes 1.

∴ $B_3\ B_2\ B_1\ B_0 = 0110$ (of adder-2)

Hence six (0110) will be added to the sum output of adder-1.

We get the corrected BCD result at the sum output of adder-2.

Case III : Sum ≤ 9 but carry = 1

As carry output of adder-1 is high, Y′ = 1.

∴ $B_3\ B_2\ B_1\ B_0 = 0\ 1\ 1\ 0$ (of adder-2)

∴ 0 1 1 0 will be added to the sum output of adder-1.

We get the corrected BCD result at the sum output of adder-2.

Thus the four bit BCD addition can be carried out using the binary adder.

Q. 8 Design 8 bit BCD adder.

Ans. : The 8-bit BCD adder using the 4-bit BCD adder 74283 is shown in Fig. 6.9.

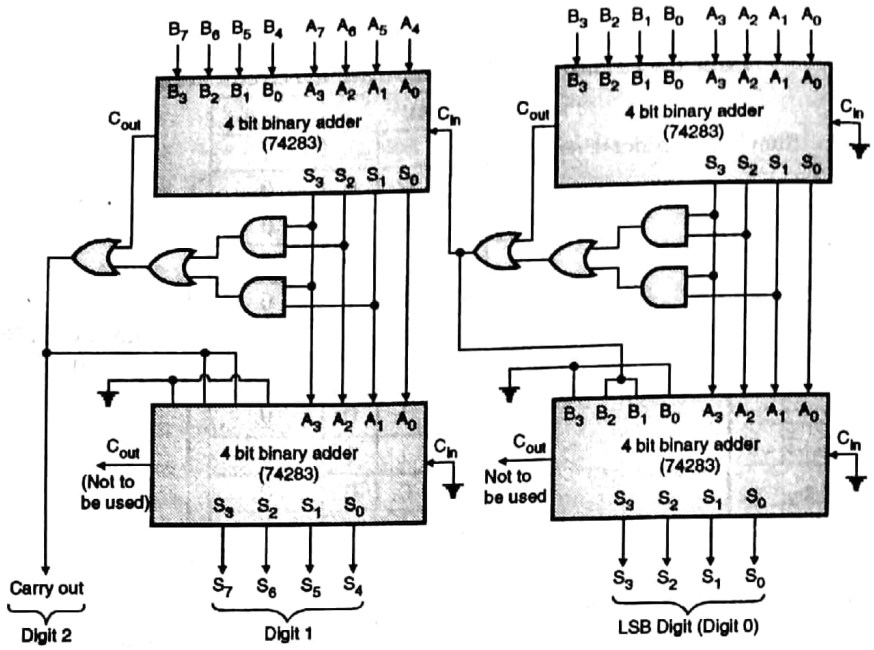


Fig. 6.9 : 8-bit BCD adder using 74283

Q. 9 Design 1 bit comparator using logic gates.

May 15

Ans. :

Truth table of one-bit comparator :

The one-bit comparator is a combinational logic circuit with two inputs A and B and three outputs namely A < B, A = B and A > B. A one bit comparator compares the two single bit numbers A and B and produces an output that indicates the result of the comparison. This is clear from the truth table as given in Table 6.5.

Table 6.5 : Truth table of a one-bit comparator

Inputs		Outputs		
A	B	$Y_1 = A < B$	$Y_2 = A = B$	$Y_3 = A > B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

K-maps for each output :

The K-maps for the three outputs Y_1, Y_2 and Y_3 are as shown in Fig. 6.10.

From Fig.6.10, the expressions for the three outputs as,

$$Y_1 = (A < B) = \bar{A}B,$$

$$Y_2 = (A = B) = \bar{A}\bar{B} + AB = \overline{A \oplus B}$$

$$Y_3 = A\bar{B}$$

The expression for Y_2 is nothing but the expression for an EX-NOR gate. Hence the single bit comparator can be realized using basic gates as shown in Fig. 6.11.

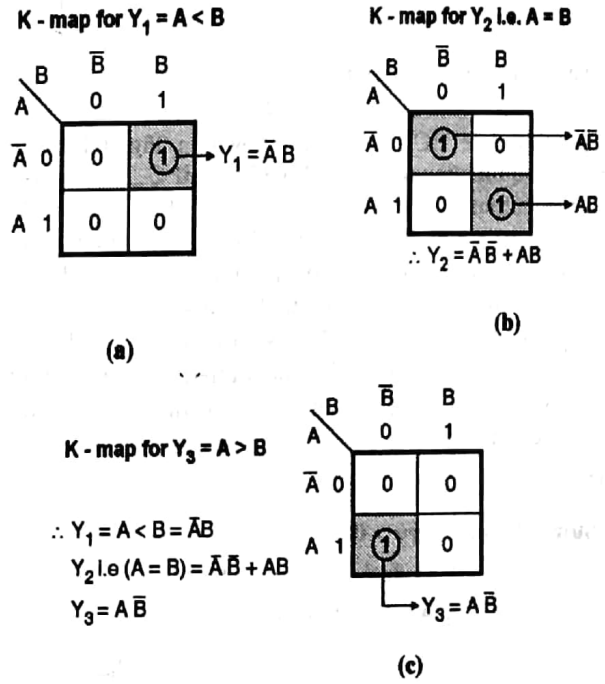
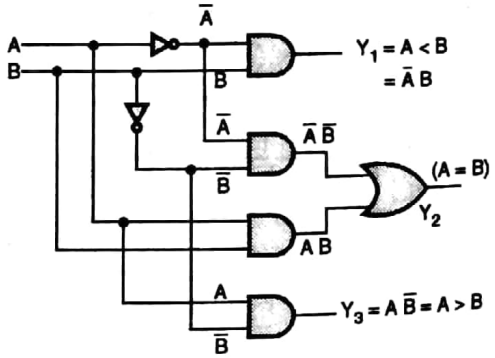
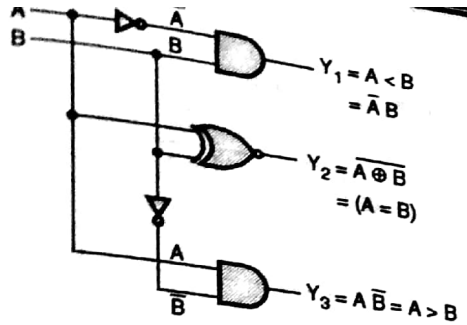


Fig.6.10 : K-maps for the three outputs of a one-bit comparator



(a) Using basic gates



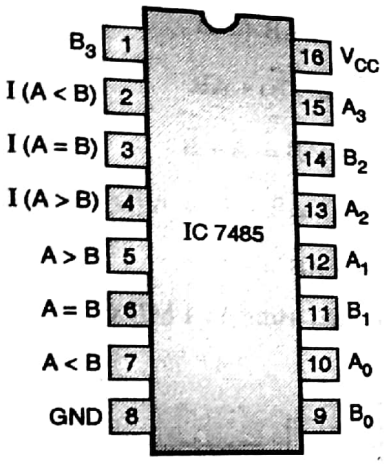
(b) Using AND and EX-NOR gates

Fig.6.11: Realization of one bit comparator

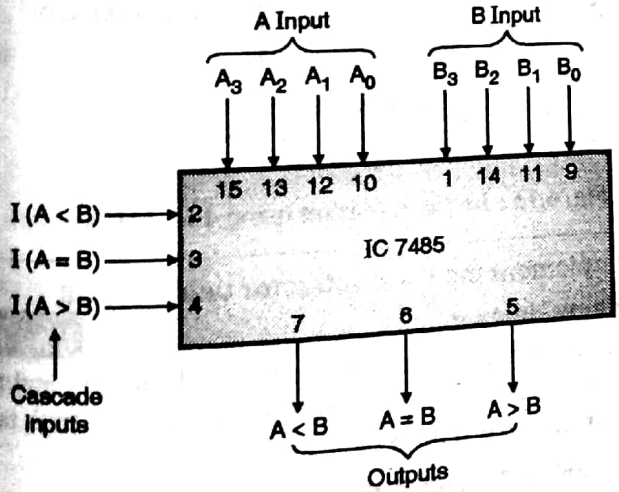
10 Write short note on :4-bit magnitude comparator

May 16

Ans. : IC 7485 is a four bit comparator in the integrated circuit form. It compares two 4-bit words A ($A_3 - A_0$) and B ($B_3 - B_0$). It is possible to cascade more than one IC 7485 to compare words of almost any length. Fig. 6.12 shows the pin configuration and logic symbol of IC 7485.



(a) Pin configuration



(b) Logic diagram of IC 7485

Fig.6.12

Pin names and their functions :

The pin names and their functions are as listed in Table 6.6(a).

Table 6.6(a)

Pin name	Pin number	Function
A_0 to A_3 B_0 to B_3	10, 12, 13, 15 9, 11, 14, 1	Binary input (operand 1) Active high Binary input (operand 2) Active high
I (A < B) I (A = B) I (A > B)	2 3 4	These lines are used for cascading a number of IC 7485 outputs of the previous stage are fed as inputs to this stage.
A < B A = B A > B	7 6 5	These are the outputs. When ICs 7485 are cascaded, these outputs are applied to cascading inputs of the next stage.

Truth table of IC 7485

Table 6.6(b)

Comparing Inputs				Cascading Inputs			Outputs		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$Y_{A>B}$	$Y_{A<B}$	$Y_{A=B}$
$A_3 > B_3$	X	X	X	X	X	X	H	L	L
$A_3 < B_3$	X	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	L	L	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	H	L	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	L	H	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	X	X	H	L	L	H
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	H	L	L	L	L

Function table of IC 7485 :

Table 6.6(c) shows the function table for IC 7485.

Table 6.6(c)

Comparing inputs	Cascading inputs			Outputs		
	$I_{A<B}$	$I_{A=B}$	$I_{A>B}$	$Y_{A<B}$	$Y_{A=B}$	$Y_{A>B}$
A < B	X	X	X	1	0	0
A = B	0	0	0	1	0	1
	0	0	1	0	0	1
	1	0	0	1	0	0
	1	0	1	0	0	0
	X	1	X	0	1	0
A > B	X	X	X	0	0	1

The function table shown in Table 6.6(c) tells us that if I (A < B) or I (A > B) is equal to 1 and if the other two cascading inputs are at logic 0 then the corresponding output (A < B or A > B) will be active. But if A = B, then the corresponding output will be

active if and only if I (A = B) is at logic 1 irrespective of the other two inputs.

General description : As shown in the truth table (Table 6.6(c)); the MSB are compared first i.e. A_3 is compared with B_3. Depending on whether $A_3 > B_3$ or $A_3 < B_3$ the outputs Y (A > B) or Y (A < B) are activated. But if $A_3 = B_3$ then the next MSB bits B_2 and A_2 are compared. If $A_2 = B_2$ then comparison of A_1 and B_1 is performed and so on. If $A_3\ A_2\ A_1\ A_0 = B_3\ B_2\ B_1\ B_0$ then the IC 7485 will check the cascading inputs. The decision making will then take place as follows :

Case 1 : If I (A > B) = 1 and I (A = B) = I (A < B) = 0 :

This indicates that at the previous stage LSB of A is greater than LSB of B.

Case 2 : If I (A = B) = 1 :

If I (A = B) = 1 then the chip will not check the status of I (A < B) and I (A > B) inputs and will understand that the LSB of A is equal to LSB of B.

Q. 11 Implement the following Boolean function using 4:1 MUX May 10, May 15

F(A, B, C, D) = Σm(0, 1, 2, 4, 6, 9, 12, 14)

Ans. :

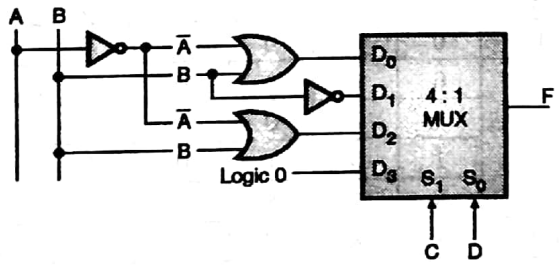
F (A, B, C, D) = Σm(0, 1, 2, 4, 6, 9, 12, 14)

Step 1 : Write the design table :

$$D_0 = \bar{A}\bar{B} + \bar{A}B + AB = \bar{A}(\bar{B} + B) + AB$$
$$= \bar{A} + AB = \bar{A} + B$$
$$D_1 = \bar{A}\bar{B} + A\bar{B} = \bar{B}(\bar{A} + A) = \bar{B}$$
$$D_2 = \bar{A}\bar{B} + \bar{A}B + AB$$
$$= \bar{A}(\bar{B} + B) + AB = \bar{A} + AB = \bar{A} + B = D_3 = 0$$

Inputs	D_0	D_1	D_2	D_3
$\bar{A}\bar{B}$	(0)	(1)	(2)	3
$\bar{A}B$	(4)	5	(6)	7
$A\bar{B}$	8	(9)	10	11
AB	(12)	13	(14)	15
				0

Step 2 : Implementation using 4 : 1 MUX : (C-3268)



Q. 12 Implement the following function using 4 : 1 multiplexer and few gates. Dec. 16

F (A, B, C, D) = Σ m (0, 1, 2, 3, 6, 7, 9, 10, 13, 15)

Ans. :

Step 1 : Write the design table :

	D_0	D_1	D_2	D_3
$\bar{A}\bar{B}$	(0)	(1)	(2)	(3)
$\bar{A}B$	4	5	(6)	(7)
$A\bar{B}$	8	(9)	(10)	11
AB	12	(13)	14	(15)
	$\bar{A}\bar{B}$			

$$D_0 = \bar{A}\bar{B}$$
$$D_1 = \bar{A}\bar{B} + A\bar{B} + AB = \bar{B}(A + \bar{A}) + AB$$
$$= \bar{B} + AB \quad \ldots (\because A + \bar{A} = 1)$$
$$= \bar{B} + A$$
$$D_2 = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$$
$$= \bar{A}(B + \bar{B}) + A\bar{B} \quad (\because B + \bar{B} = 1)$$
$$= \bar{A} + A\bar{B} = \bar{A} + \bar{B}$$
$$D_3 = \bar{A}\bar{B} + \bar{A}B + AB = \bar{A}(B + \bar{B}) + AB$$
$$= \bar{A} + AB = \bar{A} + B$$

Step 2 : Implementation using 4 : 1 MUX :

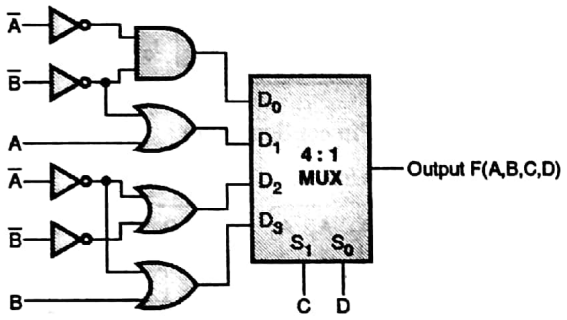


Fig. 6.13 : Implementation using 4 : 1 MUX

Q. 13 Implement the full subtractor using a 1 : 8 demultiplexer. May 16

Ans. : The full subtractor has three inputs A, B and B_{in} and two outputs namely difference D and borrow out (B_{out}). The truth table of a full subtractor is as follows :

Table 6.7: Truth table of a full subtractor

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

1	1	0	0	0
1	1	1	1	1

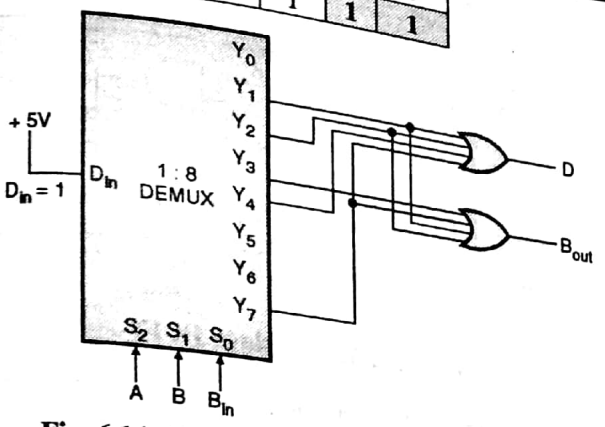


Fig. 6.14 : Full subtractor using 1 : 8 demux

From the truth table we can express the difference and borrow out outputs in the standard SOP forms as,

$$D = f(A, B, B_{in}) = \Sigma m (1, 2, 4, 7)$$

and $$B_{out} = f(A, B, B_{in}) = \Sigma m (1, 2, 3, 7)$$

Connect D_{in} to logic 1 permanently and connect A, B and B_{in} to the select inputs S_2 , S_1 , S_0 respectively as shown in Fig. 6.14. As D_{in} is connected to 1, we get the required minterms at the Demux outputs. We have to OR the required minterms to obtain the D and B_{out} outputs as shown in Fig. 6.14.

Chapter 7 : Flip Flops

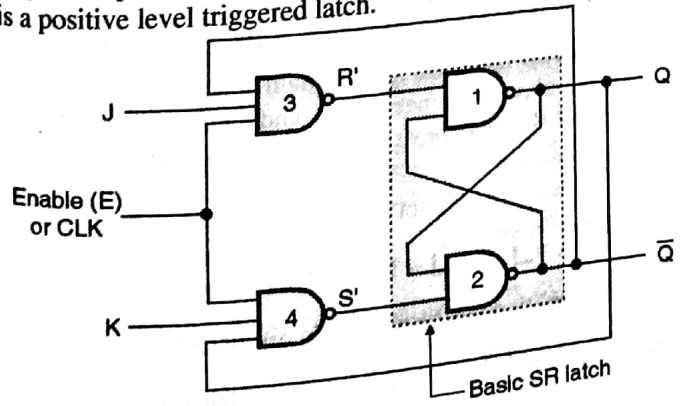
Q. 1 Differentiate in brief between combinational and sequential circuits. Dec. 15

Ans. :

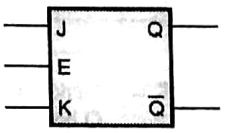
Sr. No.	Parameter	Combinational circuits	Sequential circuits
1.	Output depends on	Inputs present at that instant of time.	Present inputs and past inputs / outputs.
2.	Memory	Not necessary	Necessary
3.	Clock input	Not necessary	Necessary
4.	Examples	Adders, subtractors, code converters	Flip flops, shift registers, counters

Q. 2 Draw JK flip-flop using SR flip-flop and additional gates. Explain briefly the race around condition in JK flip-flop. Dec. 16

Ans. : The JK latch using NAND gates is shown in Fig. 7.1(a). It consists of the basic SR latch and an enable input. It is also called as level triggered JK flip-flop. Note the outputs Q and $\bar{Q}$ have been fed back and connected to the inputs of NAND gates 4 and 3 respectively. The JK latch of Fig. 7.1(a) responds, to the input changes if a positive level is applied at the enable (E) input. Hence it is a positive level triggered latch.



(a) A level triggered JK flip-flop



(b) Symbol of the gated JK latch

Fig. 7.1

Operation :

The operation of NAND JK latch is exactly identical to that of the positive edge triggered JK flip flop. The only difference between them is that, this circuit is level triggered. The operation of NAND JK latch has been summarized in Table 7.1.

Table 7.1 : Truth table of positive level triggered JK latch

Case No.	Inputs			Outputs		State
	E	J	K	Q_{n+1}	$\bar{Q}_{n+1}$	
Case I	0	×	×	Q_n	$\bar{Q}_n$	No change
Case II	1	0	0	Q_n	$\bar{Q}_n$	No change
Case III	1	0	1	0	1	Reset
Case IV	1	1	0	1	0	Set
Case V	1	1	1	$\bar{Q}_n$	Q_n	Toggle

Q. 3 Draw JK flip-flop using SR flip-flop and additional gates. Explain briefly the race around condition In JK flip-flop. Dec. 12, Dec. 15, Dec. 16

Ans. : The "Race Around Condition" that we are going to explain occurs when J = K = 1 i.e. when the latch is in the toggle mode. Refer Fig.7.2 which shows the waveforms for the various modes, when a rectangular waveform is applied to the "Enable" input.

Interval t_0 - t_1

During this interval J = 1, K = 0 and E = 0. Hence the latch is disabled and there is no change in Q.

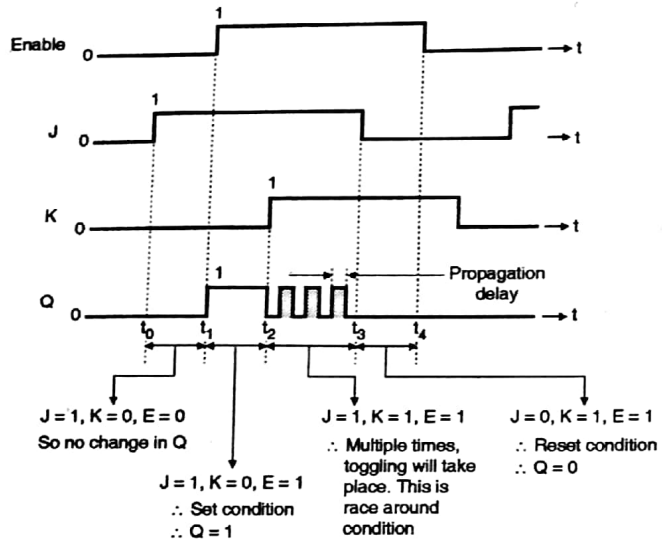


Fig. 7.2 : Waveforms for various modes of a JK latch

Interval t_1 - t_2

During this interval J = 1, K = 0 and E = 1. Hence this is a set condition and Q becomes 1.

Interval t_2 - t_3 : Race around

At instant t_2, J = K = 1 and E = 1 Hence the JK latch is in the toggle mode and Q becomes low (0) and $\bar{Q}$ = 1. These changed outputs get applied at the inputs of NAND gates 3 and 4 of the JK latch. Thus the new inputs to Gates 3 and 4 are :

NAND - 3 : J = 1, E = 1, $\bar{Q}$ = 1.
NAND - 4 : K = 1, E = 1, Q = 0.

Hence R′ will become 0 and S′ will become 1. Therefore after a time period corresponding to the propagation delay, the Q and $\bar{Q}$ outputs will change to, Q = 1 and $\bar{Q}$ = 0. These changed output again get applied to the inputs of NAND-3 and 4 and the outputs will toggle again. Thus as long as J = K = 1 and E = 1, the outputs will keep toggling indefinitely as shown in Fig. 7.2. This multiple toggling in the J-K latch is called as Race Around condition. It must be avoided.

Interval t_3 - t_4

During this interval J = 0, K = 1 and E = 1. Hence it is the reset condition. So Q becomes zero.

How to avoid race around condition ?

The race around condition in JK latch can be avoided by :

Using the edge triggered JK flip flop.

Using the master slave JK flip flop.

Q. 4 Explain working of Master-Slave J-K flip flop.

May 14

Ans. : Fig. 7.3 shows the master slave JK flip flop.

It is a combination of a clocked JK latch (level triggered JK FF) and clocked SR latch (level triggered SRFF). The clocked JK latch acts as the master and the clocked SR latch acts as the slave. Master is positive level triggered. But due to the presence of the inverter in the clock line, the slave will respond to the negative level.

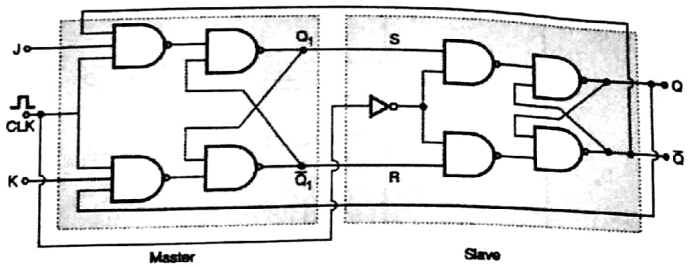


Fig. 7.3 : Master slave JK FF

Thus both master and slave circuits are level triggered circuits. Hence when the clock = 1 (positive level) the master is active and the slave is inactive. Whereas when clock = 0 (low level) the slave is active and the master is inactive. Table 7. 2 gives truth table of master slave JK flip flop.

Table 7. 2 : Truth table of master slave JK FF

Case	Inputs			Outputs		Remark
	CLK	J	K	Q_{n+1}	$\bar{Q}_{n+1}$	
I	×	0	0	Q_n	$\bar{Q}_n$	No change
II	⊓(1)	0	0	Q_n	$\bar{Q}_n$	No change
III	⊓(1)	0	1	0	1	Reset
IV	⊓(1)	1	0	1	0	Set
V	⊓(1)	1	1	$\bar{Q}_n$	Q_n	Toggle

Operation :

We will discuss the operation of the master slave JK FF with reference to its truth table. We must always remember one important thing that in the positive half cycle of the clock, the master is active and in the negative half cycle, the slave is active. This is shown in Fig. 7.4.

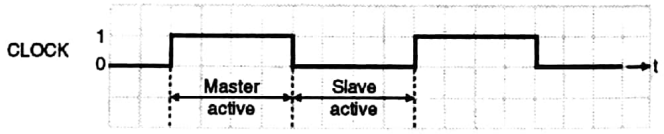


Fig. 7.4

Case I : Clock = ×, J = K = 0 (No change)

For clock = 1, the master is active, slave inactive. As J = K = 0. ∴ Outputs of master i.e. Q_1 and $\bar{Q}_1$ will not change. Hence the S and R inputs to the slave will remain unchanged. As soon as clock = 0, the slave becomes active and master is inactive. But since the S and R inputs have not changed, the slave outputs will also remain unchanged.

∴ The outputs will not change if J = K = 0.

Case II : Clock = ⊓, J = K = 0 (No change)

This condition has been already discussed in case I.

Case III : Clock = ⊓, J = 0 and K = 1 (Reset)

Clock = 1 : Master active, slave inactive.

∴ Outputs of the master become $Q_1 = 0$ and $\bar{Q}_1 = 1$. That means S = 0 and R = 1.

Clock = 0 : Slave active, master inactive

∴ Outputs of the slave become Q = 0 and $\bar{Q} = 1$. This is the RESET operation.

Again if clock = 1 : Master active, slave inactive.

∴ Even with the changed outputs Q = 0 and $\bar{Q} = 1$ fed back to master, its outputs will be $Q_1 = 0$ and $\bar{Q}_1 = 1$. That means S = 0 and R = 1.

Hence with clock = 0 and slave becoming active, the outputs of slave will remain Q = 0 and $\bar{Q} = 1$.

Thus we get a stable output from the Master slave.

Case IV : CLK = ⊓, J = 1, K = 0 (Set)

Clock = 1 : Master active, slave inactive.

∴ Outputs of master become $Q_1 = 1$ and $\bar{Q}_1 = 0$ i.e. S = 1, R = 0.

Clock = 0 : Master inactive, slave active.

∴ Outputs of slave become Q = 1 and $\bar{Q} = 0$.

Again if clock = 1 then it can be shown that the outputs of the slave are stabilized to Q = 1 and $\bar{Q} = 0$.

Case V : CLK = ⊓, J = 1, K = 1 (Toggle without Race)

Clock = 1 : Master ⊓ ve, slave inactive.

∴ Outputs of master will toggle. So S and R also will be inverted.

Clock = 0 : Master inactive, slave active.

∴ Outputs of the slave will toggle.

These changed output are returned back to the master inputs. But since clock = 0, the master is still inactive. So it does not respond to these changed outputs. This avoids the multiple toggling which leads to the race around condition. Thus the master slave flip flop will avoid the race around condition. The waveforms for the master slave flip flop are shown in Fig. 7.5.

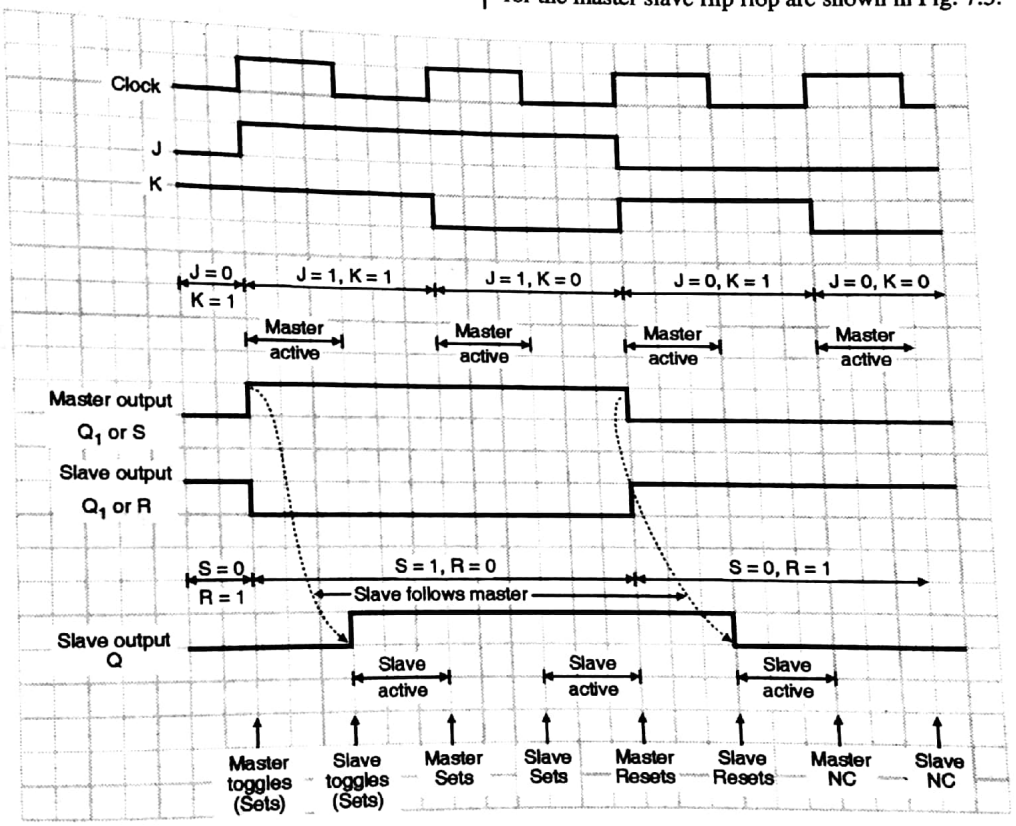


Fig. 7.5 : Waveforms of master slave JK flip flop

Observations from the waveforms :

We can make the following important observations from the waveforms of the master slave JK FF :

1. The slave always follows the master, after a delay of half clock cycle period.
2. The multiple toggling or the race around condition is successfully avoided.

Q. 5 Convert SR-flip-flop to D flip-flop.

Dec 04. May 08. May 10. Dec. 10. May 12. Dec. 12. May 14. May 16

Ans. :

Refer Fig. 7. 6. Here the given FF is SR FF and the required FF is D FF. The truth table for the conversion logic is shown in Table 7.3(a). The inputs are D and Q whereas outputs are S and R.

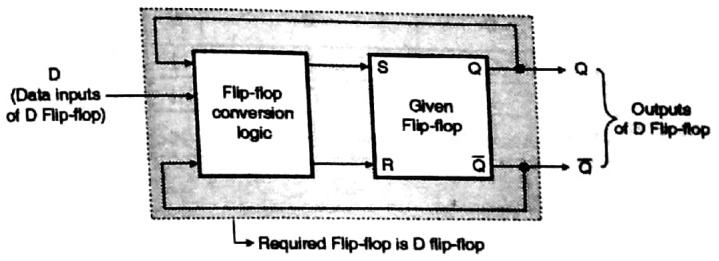


Fig. 7. 6.

The truth table is prepared by combining the excitation tables of D F/F and SR FF.

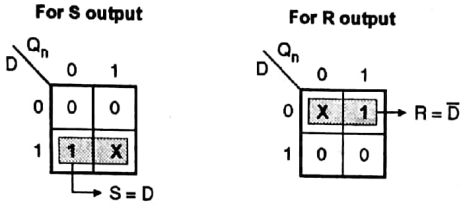
Table 7.3(a) : Truth table for SR to D FF conversion

Inputs			Outputs	
D	Present state Q_n	Next state Q_{n+1}	S	R
0	0	0	0	×
1	0	1	1	0
0	1	0	0	1
1	1	1	×	0

← Entries from excitation table of D FF →|

|← Entries from excitation table of SR FF →|

Now write the K maps for the S and R outputs as shown in Figs. 7.7(a) and (b).



(a) K map for S (b) K map for R

Fig. 7.7

Logic diagram : The logic diagram for SR FF to D FF is shown in Fig. 7.8.

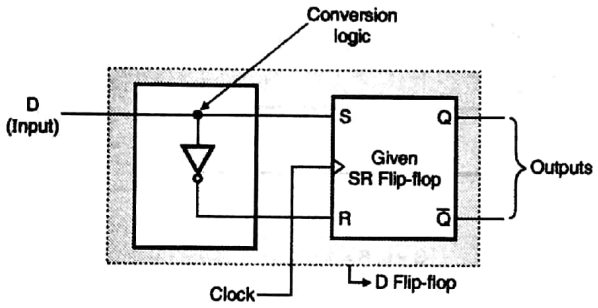


Fig. 7.8 : SR flip flop to D flip flop

Q. 6 Convert SR FF to T FF and draw waveform.

Dec 04, Dec. 05, May 10, Dec. 10, May 15

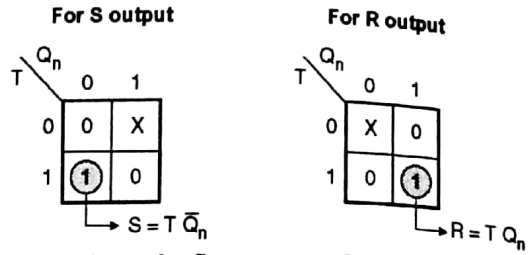
Ans. : The stepwise conversion process is as follows :

Step 1 : Write the truth table :

Table 7.3(b) : Truth table for SR FF to T FF

Inputs			Outputs	
T	Present state Q_n	Next state Q_{n+1}	S	R
0	0	0	0	×
1	0	1	1	0
1	1	0	0	1
0	1	1	×	0

Step 2 : Write the K maps and obtain the expressions for S and R :



(a) K map for S (b) K map for R

Fig. 7.9

Step 3 : Draw the logic diagram :

The logic diagram is shown in Fig. 7.10.

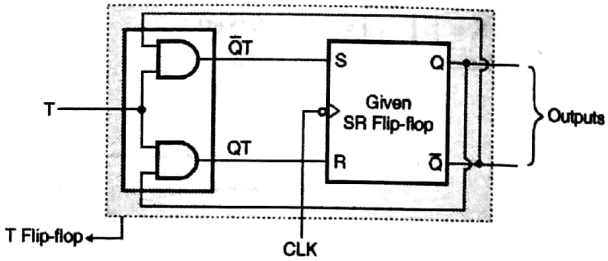


Fig. 7.10 : Conversion from SR flip flop to T flip flop

Q. 7 Convert : SR FF to JK FF.

Dec. 04, May 12, Dec. 12, May 14, May 15, May 16

Ans. :

Step 1 : Write the truth table for SR to JK :

The truth table for SR to JK flip flop conversion is shown in Table 7. 4.

Table 7.4 : Truth table for SR to JK FF conversion

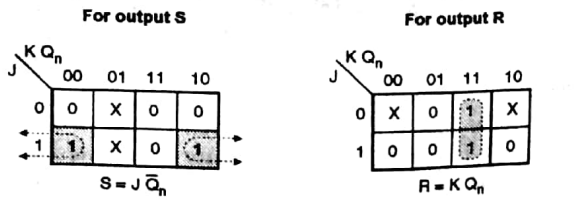
Inputs				Outputs	
J	K	Present state Q_n	Next state Q_{n+1}	S	R
0	0	0	0	0	×
0	1	0	0	0	×
1	0	0	1	1	0
1	1	0	1	1	0
0	1	1	0	0	1
1	1	1	0	0	1
0	0	1	1	×	0
1	0	1	1	×	0

|← Excitation table of JKFF →|

|← Excitation table of SR FF →

Step 2 : K maps and simplification :

K maps for S and R outputs are shown in Fig. 7.11.



(a) K-map for S output (b) K-map for R output

Fig. 7.11

Step 3 : Logic diagram : The logic diagram of SR to JK flip flop is given in Fig. 7.12.

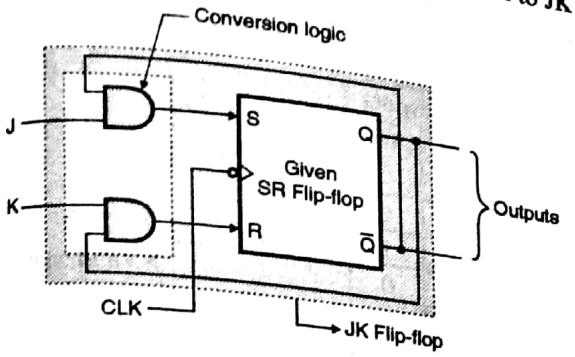


Fig. 7.12 : SR to JK flip flop conversion

Q. 8 Convert JK FF to SR and D FF.

Dec. 05, May 11, May 14, Dec. 14, Dec. 15

Ans. : Step 1 : Write the truth table for JK to D conversion :

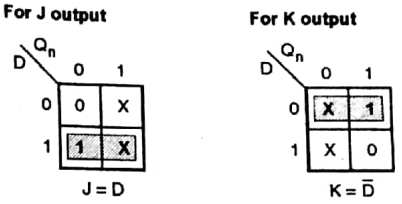
Table 7.5 : Truth table for JK to D flip flop conversion

D	Inputs: Previous state Q_n	Inputs: Next state Q_{n+1}	Outputs: J	Outputs: K
0	0	0	0	×
1	0	1	1	×
0	1	0	×	1
1	1	1	×	0

← Excitation table of D FF →

← Excitation table of JK FF →

Step 2 : K maps and simplification



(a) K map for J output **(b) K map for K output**

Fig. 7.13

Step 3 : Draw the logic diagram :

The logic diagram for JK flip flop to D flip flop is shown in Fig. 7.14.

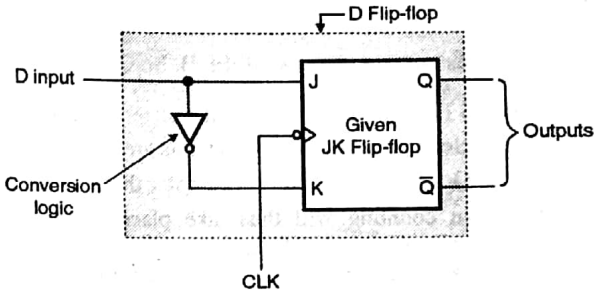
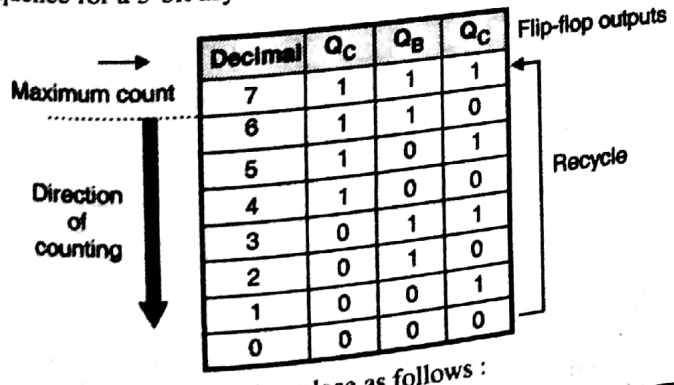


Fig. 7.14 : Logic diagram for conversion from JK FF to D FF

Chapter 8 : Counters

Q. 1 Draw a circuit diagram for 3-bit asynchronous binary down counter using master-slave JK flip-flops. Show the output of each flip-flop with reference to the clock and justify that the down counting action. Also prove from the timing diagram that the counter is "divide by 8" counter. Dec. 15

Ans. : All the counters discussed so far have counted upwards from zero. So they can be called as **up counters.** But the counters which can count in the **downward** direction i.e. from the maximum count to zero are called **down counters.** The countdown sequence for a 3-bit asynchronous down counter is as follows :



Thus counting takes place as follows :

$Q_C\,Q_B\,Q_A$ = 111, 110, 101, 100, 011, 010, 001, 000.

From this sequence it is evident that FF-A should toggle at every negative going clock edge but FF-B should change its state only at those instants when Q_A changes from LOW (0) to HIGH (1) and Q_C should changes only when Q_B changes from LOW to HIGH. Thus in a down counter, each FF except the first one (FF-A) should toggle when the output of its preceding flip-flop changes from LOW to HIGH. If all the FFs are negative edge triggered i.e. responding to the negative CLK edge, then we can place an inverter in front of every CLK input or we can drive the CLK input of next FF from the $\overline{Q}$ output of the preceding FF and not from the Q outputs as shown in Fig. 8.1. A 3-bit asynchronous down counter is shown in Fig. 8.1. The clock input is applied directly to the clock input of FF-A. But $\overline{Q_A}$ is connected to clock of FF-B, $\overline{Q_B}$ to clock of FF-C and so on.

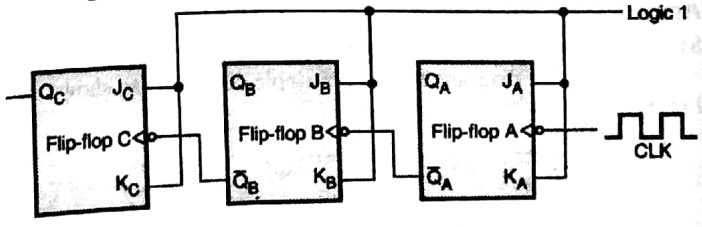


Fig. 8.1 : A 3-bit asynchronous down counter

Operation : Initially let all the flip-flops be in the reset condition.

$$\therefore \quad Q_C\, Q_B\, Q_A = 0\,0\,0$$

As soon as the first falling clock pulse arrives, FF-A toggles. So Q_A becomes 1 and $\overline{Q_A}$ changes from 1 to 0. The negative going change in $\overline{Q_A}$ acts as a clock to FF-B. Hence FF-B will change its state. So Q_B becomes 1 and $\overline{Q_B}$ changes from 1 to 0. This negative going change in $\overline{Q_B}$ acts as a clock to FF-C. Hence FF-C will change its state. So Q_C becomes 1 and $\overline{Q_C}$ becomes 0. Thus after the first clock pulse the output of counter are,

$$Q_C\, Q_B\, Q_A = 1\,1\,1 \quad \text{..... After the } 1^{st} \text{ CLK pulse}$$

Corresponding to the second falling clock edge, FF-A toggles. Q_A becomes 0 and $\overline{Q_A}$ becomes 1. This positive going change in $\overline{Q_A}$ does not alter the state of FF-B. So Q_B remains 1 and $\overline{Q_B}$ remains 0. So there is no change in the state of FF-C. Hence after the second clock pulse the counter outputs are,

$$Q_C\, Q_B\, Q_A = 1\,1\,0 \quad \text{..... After the } 2^{nd} \text{ CLK pulse}$$

The down counting will thus take place. Similarly the counter will count down to pass through the states 101, 100, 011, 010, 001 and 000. The operation repeats itself thereafter. The timing diagram for the down counter is shown in Fig. 8.2.

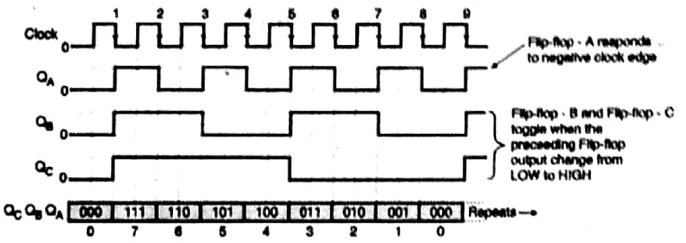


Fig. 8.2 : Timing diagram of a 3-bit down counter

State diagram : The state diagram of the 3-bit down counter is shown in Fig. 8.3.

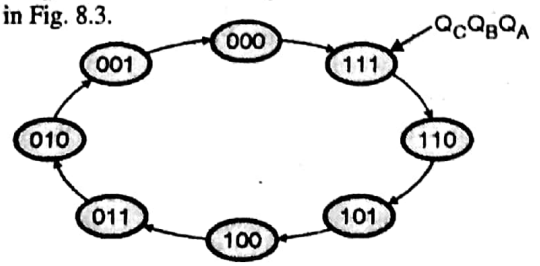


Fig. 8.3 : State diagram of a 3 bit ripple down counter

Q. 2 Design a Modulo-5 ripple counter using a 3-bit ripple counter.

OR Design MOD-5 asynchronous counter, and also draw the waveforms. Dec 02, May 16

Ans. :

Step 1 : Draw the state diagram :

The state diagram of MOD-5 ripple counter is as shown in Fig. 8.4(a).

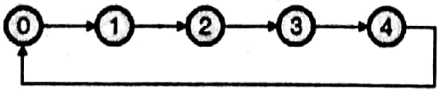


Fig. 8.4(a) : State diagram of a MOD-5 ripple counter

Step 2 : Write truth table for the reset logic :

Table . 8.1 shows the truth table for the reset logic

Table 8.1 : Truth table for the reset logic

State	Flip-flop outputs Q_C	Q_B	Q_A	Output Y of reset logic	
0	0	0	0	1	Valid states
1	0	0	1	1	
2	0	1	0	1	
3	0	1	1	1	
4	1	0	0	1	
5	1	0	1	0	Invalid states
6	1	1	0	0	
7	1	1	1	0	

Step 3 : K-map :

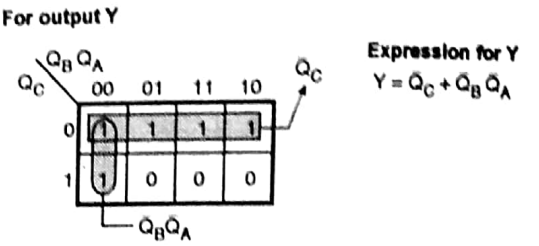


Fig. 8.4(b) : K-map and simplification

The K map is as shown in Fig. P. 8.4(b). The states 0 through 4 are valid states and the output Y of reset logic (Y) is inactive (1) for them. The states 5, 6 and 7 are invalid states. If counter enters into any one of these states that Y = 0 (active) and will reset all the flip-flops.

Step 4 : Logic diagram : The logic diagram of a MOD-5 ripple counter is shown in Fig. 8.4(c).

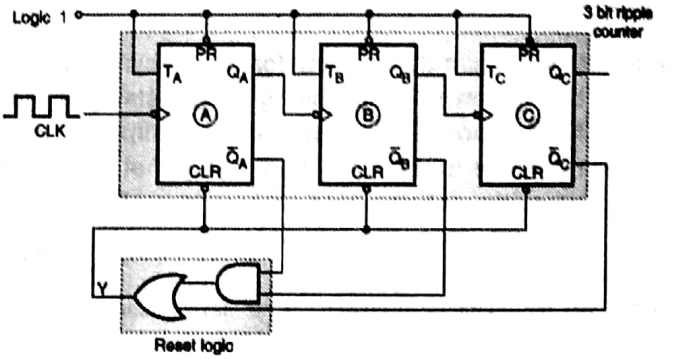


Fig. 8.4(c) : Logic diagram of MOD-5 ripple counter

Step 5 : Timing diagram : The timing diagram is as shown in Fig. 8.4(d).

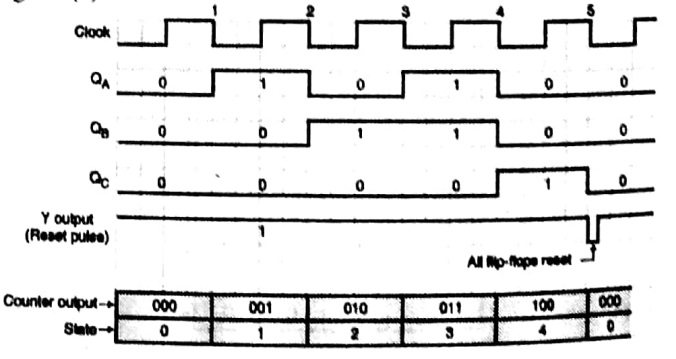


Fig. 8.4(d) : Timing diagram of MOD-5 ripple counter

Q.3 Design a synchronous counter for the sequence shown in Fig. 8.5(a). **May 15, Dec. 15**

Ans. :

Table 8.2(a) : Desired sequence

Q_C	Q_B	Q_A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0

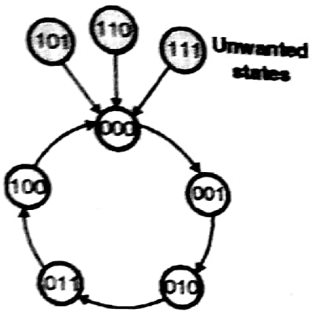


Fig. 8.5(a) : State diagram

Ans. :

Step 1 : Determine the desired number of FFs :

From the given sequence the number of FFs is equal to 3. This is a MOD-5 synchronous counter since the number of states is 5. The state diagram is shown in Fig. 8.5(a) which shows that 101, 110, 111 are unwanted states.

Excitation table of JK FF :

Table 8.2(b) exhibits the excitation table of a JK FF.

Circuit excitation table :

The circuit excitation table is as shown in Table 8.2(c).

Step 2 : Write the excitation table and state table :

The type of FF used is JK flip-flop. The excitation table for a JK FF is as shown in Table 8.2(b). We have already seen how to write the excitation table for JK FF.

Table 8.2(b) : Excitation table of JK FF

Present state Q_n	Next state Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table 8.2(c) : Circuit excitation table

Present state			Next state			Flip-flop inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	×	0	×	1	×
0	0	1	0	1	0	0	×	1	×	×	1
0	1	0	0	1	1	0	×	×	0	1	×
0	1	1	1	0	0	1	×	×	1	×	1
1	0	0	0	0	0	×	1	0	×	0	×
1	0	1	0	0	0	×	1	0	×	×	1
1	1	0	0	0	0	×	1	×	1	0	×
1	1	1	0	0	0	×	1	×	1	×	1

Refer to the shaded portion of the circuit excitation table. This is nothing but the excitation table of FF-C. The J_C and K_C values have been decided based on Q_C and Q_{C+1}. Similarly the entries for J_B and K_B are based on Q_B and Q_{B+1} whereas those for J_A and K_A are based on Q_A and Q_{A+1}.

Step 3 : K-maps and simplifications :

K-maps for the J and K inputs of all the FFs and the corresponding simplified equations are shown in Fig. 8.5(b). Note that the inputs to this combinational circuit are Q_A , Q_B , Q_C and outputs are J_A , K_A through J_C, K_C.

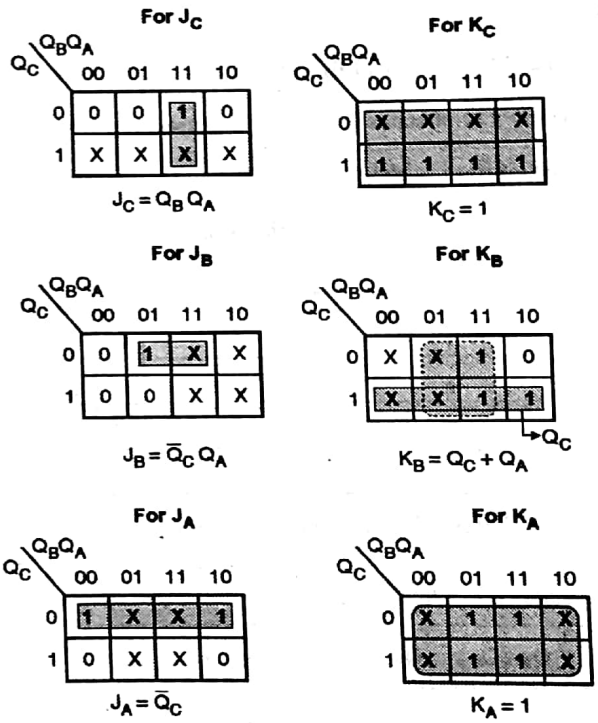


Fig. 8.5(b) : K-maps and simplifications

Step 4 : Logic diagram :

The logic diagram of MOD-5 synchronous counter is shown in Fig. 8.5(c).

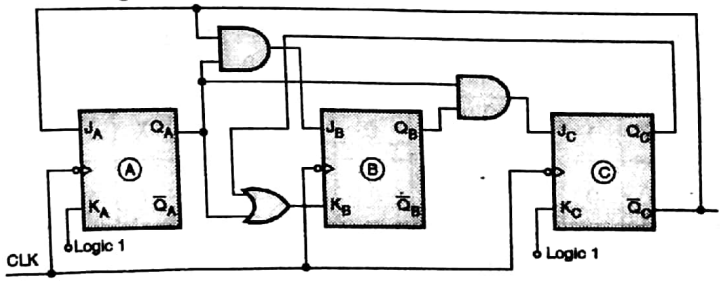


Fig. 8.5(c) : Logic diagram of MOD-5 synchronous counter

Step 5 : Draw the timing diagram :

The timing diagram of MOD-5 synchronous counter is shown in Fig. 8.5(d).

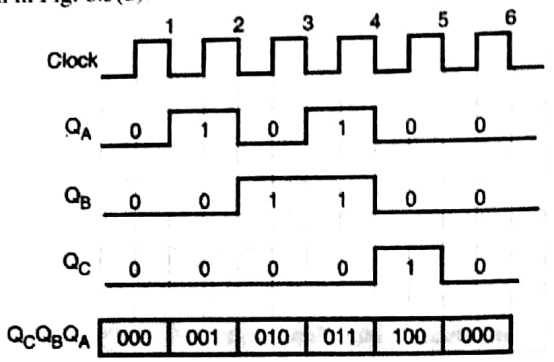


Fig. 8.5(d) : Timing diagram of MOD-5 counter

Q. 4 Design a mod-6 synchronous counter using JK FF. May 12

Ans. :

Step 1 : Decide number of flip flops :

Since the number of states is 6 we need to use 3 flip-flops

Step 2 : Excitation tables :

Table 8.3(a) : Excitation table of JK FF

Present state Q_n	Next state Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table 8.3(b) : Circuit excitation table

Present state			Next state			Flip-Flop inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	0	X	1	0	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X
1	1	1	0	0	0	X	1	X	1	X	1

Step 3 : K-map simplification :

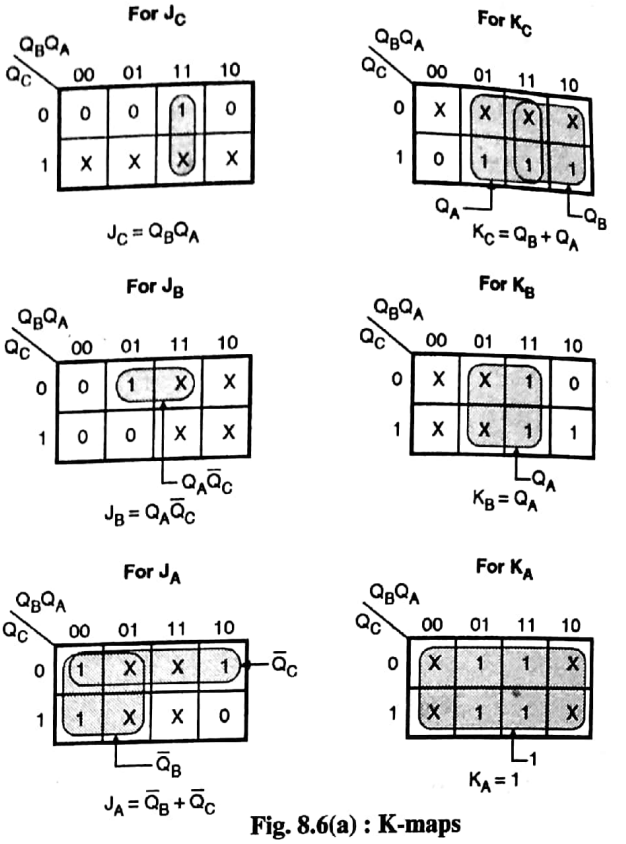


Fig. 8.6(a) : K-maps

Step 4 : Logic diagram :

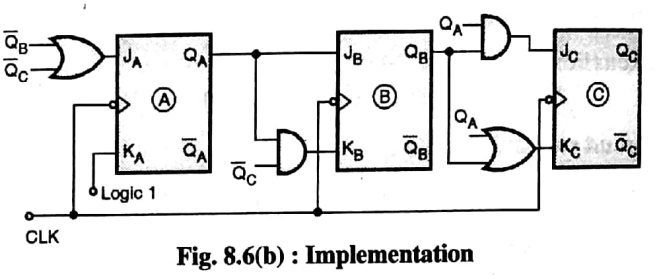


Fig. 8.6(b) : Implementation

Q. 5 Design a sequence generator using T flip-flop for the given sequence. Check for lock-out conditions. 0 → 2 → 4 → 5 → 0 May 10, Dec. 12

Ans. :

Step 1 : Draw the state diagram :

The state diagram is as shown in Fig. 8.7(a).

Step 2 : Number of FFs :

Since the highest state is 5, we have to use three T flip-flops.

Step 3 : Write the excitation table :Excitation table for a T FF

Present state	Next state	T input
	0	0
0	1	1
1	0	1
1	1	0

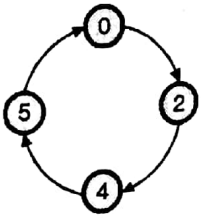


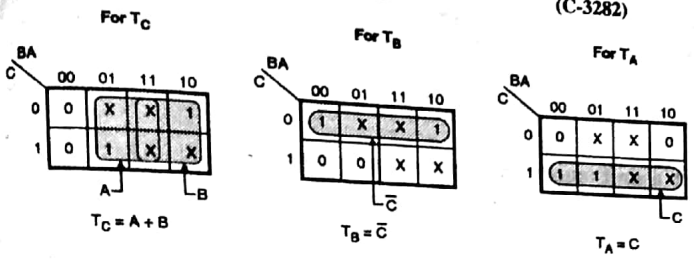
Fig. 8.7.(a)

Circuit excitation table

Present state			Next state			Flip-flop inputs		
C	B	A	C_{n+1}	B_{n+1}	A_{n+1}	T_C	T_B	T_A
0	0	0	0	1	0	0	1	0
0	0	1	×	×	×	×	×	×
0	1	0	1	0	0	1	1	0
0	1	1	×	×	×	×	×	×
1	0	0	1	0	1	0	0	1
1	0	1	0	0	0	1	0	1
1	1	0	×	×	×	×	×	×
1	1	1	×	×	×	×	×	×

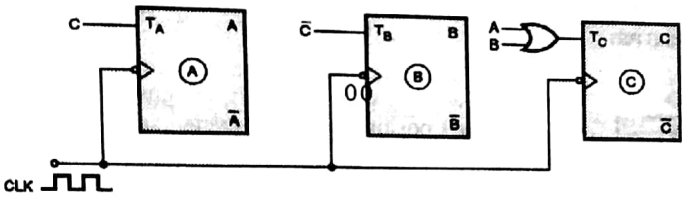
Step 4 : K-map and simplification

(C-3282)



Step 5 : Logic diagram :

(C-3283)



Step 6 : Examine the counter action for the unused states :

The unused states are 1, 3, 6 and 7.

Present state			FF inputs			Next state			
C	B	A	$T_C = A + B$	$T_B = \bar{C}$	$T_A = C$	C_{n+1}	B_{n+1}	A_{n+1}	
0	0	1	1	1	0	1	1	1	
0	1	1	1	1	0	1	0	1	
1	1	0	1	0	1	0	1	1	
1	1	1	1	0	1	0	1	0	Valid state

Thus if counter goes to state 7 it returns to state 2.
Similarly states 1, 3 and 6 gets terminated to 2.

Q. 6 Design a sequence generator to generate the sequence using 'D' FF 1101001 and repeat. Draw neat state diagram and circuit diagram. Dec. 13

Ans. :

Step 1 : Decide the number of flip flops :

Number of 1's = 4 and number of 0's = 3.
So select higher one of them i.e. 4. So N = 4.
Hence $N = 2^{n-1}$, so $4 \le 2^{n-1}$
∴ Number of flip flop n = 3

Step 2 : Write the state table and state diagram :

Assume that the required sequence is obtained at the output of flip flop C. The state assignment for other outputs is as shown in Table 8.8(a).

Table 8.4(a)

Flip flop outputs			State
C	B	A	
1	0	0	4
1	0	1	5
0	1	0	2
1	1	1	7
0	0	0	0
0	0	1	1
1	1	0	6

Given sequence

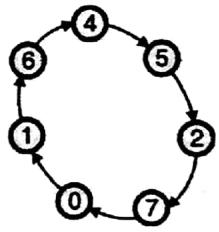


Fig. 8.8(a) : State diagram

There is only one unused state i.e. 3. We assume that next state of 3 is don't care.

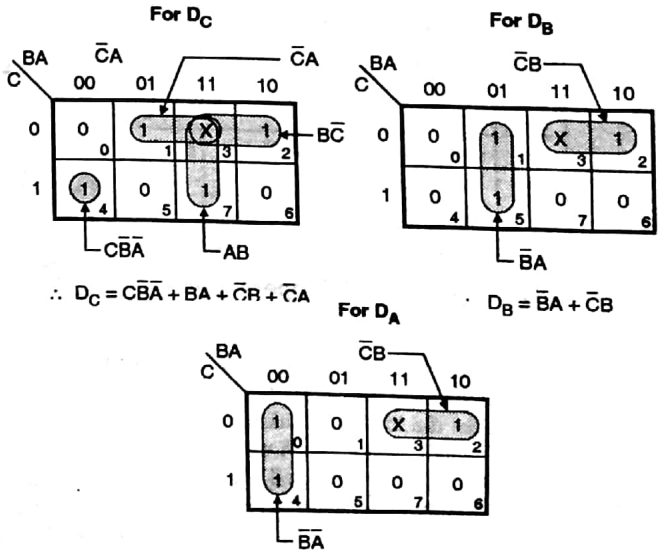
Step 3 : Write the excitation table :

Circuit excitation table is as shown in Table 8.4(b).

Table 8.4(b)

Present state			Next state			Flip flop inputs		
C	B	A	C_{n+1}	B_{n+1}	A_{n+1}	D_C	D_B	D_A
0	0	0	0	0	1	0	0	1
0	0	1	1	1	0	1	1	0
0	1	0	1	1	1	1	1	1
0	1	1	×	×	×	×	×	×
1	0	0	1	0	1	1	0	1
1	0	1	0	1	0	0	1	0
1	1	0	1	0	0	1	0	0
1	1	1	0	0	0	0	0	0

Step 4 : K-maps and simplification :



Step 5 : Draw the logic diagram :

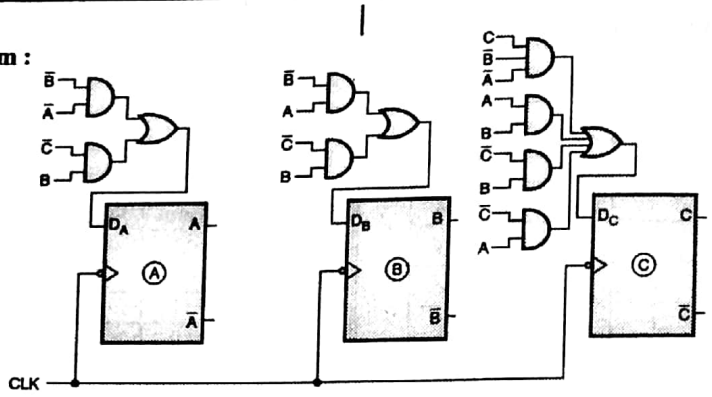


Fig. 8. 8(b) : Logic diagram

Chapter 9 : Shift Registers

Q. 1 What is a shift register ? May 04, Dec. 05, Dec. 07, Dec. 10, Dec. 11, Dec. 14, Dec. 15, Dec. 16

Ans. : The binary data in a register can be moved within the register from one flip-flop to the other or outside it with application of clock pulses. The registers that allow such data transfers are called as shift registers. Shift registers are used for data storage, data transfer and certain arithmetic and logic operations.

Modes of operation of a shift register : The various modes in which a shift register can operate are as follows :

1. Serial Input Serial Output. (SISO)
2. Serial Input Parallel Output. (SIPO)
3. Parallel In Serial Out. (PISO)
4. Parallel In Parallel Out. (PIPO)

These modes are explained in brief in Table 9.1.

Table 9.1 : Brief explanation of various modes of shift register

Sr. No.	Mode	Illustrative diagram	Comments
1.	Serial input serial output (serial shift right)	(C-731) I/P → FF3 FF2 FF1 FF0 → O/P	Data bits shift from left to right by 1 position per clock cycle.
2.	Serial input serial output (serial shift left)	(C-731) O/P ← FF3 FF2 FF1 FF0 ← I/P	Data bits shift from right to left by 1 position per clock.
3.	Serial input parallel output	I/P → FF3 FF2 FF1 FF0 Outputs (C-731)	All output bits are made available simultaneously after 4-clock pulses.
4.	Parallel input serial output	Inputs (C-731) FF3 FF2 FF1 FF0 → O/P	All inputs are loaded simultaneously but output bit by bit.
5.	Parallel input parallel output	Inputs FF3 FF2 FF1 FF0 Outputs (C-3167)	All inputs are loaded simultaneously and are available at the output simultaneously.

Explain PISO and SIPO operations. Dec. 16

his operation the data is entered serially and taken out in
it means first the data is loaded bit by bit. The outputs
as long as the loading is taking place. As soon as the
omplete, and all the flip-flops contain their required
utputs are enabled so that all the loaded data is made
ver all the output lines simultaneously. Number of clock
uired to load a four bit word is 4. Hence the speed of
of SIPO mode is same as that of SISO mode.

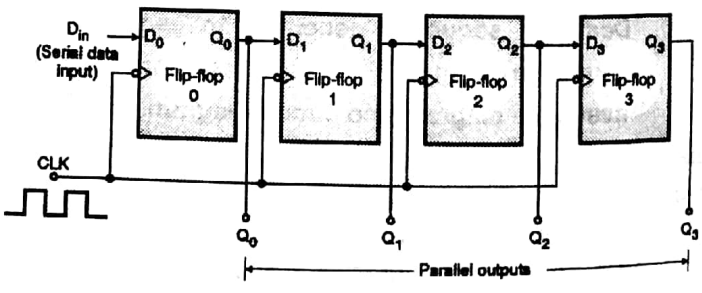


Fig. 9.1 : Serial input parallel output mode

Q. 3 Explain PISO and SIPO operations. Dec. 16

Ans. : In this mode, the bits are entered in parallel i.e. simultaneously into a shift register as shown in Fig. 9 .2. The circuit shown in Fig. 9.2 is a four bit parallel input serial output register. Output of previous FF is connected to the input of the next one via a combinational circuit. The binary input word B_0, B_1, B_2, B_3 is applied through the same combinational circuit. There are two modes in which this circuit can work namely shift mode or load mode.

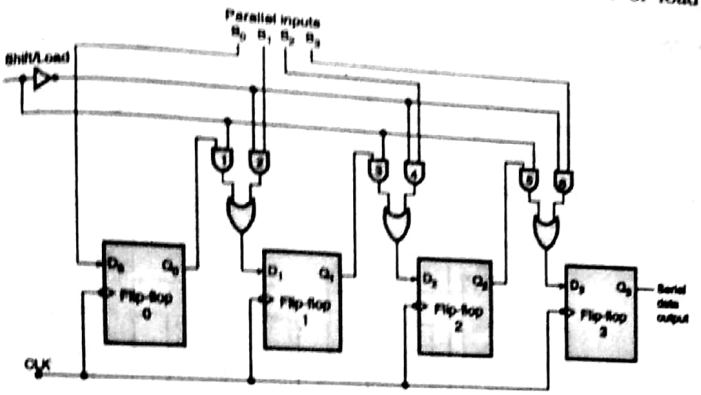


Fig. 9.2 : Parallel in serial out shift register

Load mode : In order to load the word B_3 B_2 B_1 B_0 into the shift register we have to select the load mode by setting shift/ $\overline{\text{load}}$ input to 0. When the shift / $\overline{\text{load}}$ line is low (0), the AND gates 2, 4 and 6 become active. They will pass B_1, B_2 and B_3 bits to the inputs D_1, D_2 and D_3 of the corresponding flip-flops. D_0 is directly connected to B_0. On the low going edge of clock, the binary inputs B_0 B_1 B_2 B_3 will get loaded into the corresponding flip-flops. Thus parallel loading takes place.

Shift mode : In order to operate the shift register in the shift mode we have to select the shift mode by applying a logic 1 to the shift/ $\overline{\text{load}}$ input. When the shift / $\overline{\text{load}}$ line is high (1), the AND gates 2, 4, 6 become inactive. Hence the parallel loading of the data becomes impossible. But the AND gates 1, 3 and 5 become active. Therefore the shifting of data from left to right bit by bit on application of clock pulses becomes possible. D_0 acts as the data input terminal and at Q_3 we get the serial data output. Thus the parallel in serial out operation takes place.

Q. 4 With the help of a neat diagram, explain the functioning of a 4 bit bidirectional shift register. May 04, Dec. 04, Dec. 05, Dec. 07, Dec. 10, Dec. 11, Dec. 14, Dec. 15

Ans. :

If a binary number is shifted left by one position then it is equivalent to multiplying the original number by 2. On the other hand if a binary number is shifted right by one position then it is equivalent to dividing the original number by 2. This is illustrated below. Let a four bit number Q_3 Q_2 Q_1 Q_0 = 0010 = $(2)_{10}$ is existing in a shift register. Now with a 0 applied at the input, if we shift these contents by one position to left then we get Q_3 Q_2 Q_1 Q_0 = 0100 = $(4)_{10}$. Thus the shift left is equivalent to multiplying by 2. Now shift it right by one position to get Q_3 Q_2 Q_1 Q_0 = 0001 = $(1)_{10}$. Thus shifting right is equivalent to dividing by 2. Hence if we want to use the shift register to multiply and divide the given binary number, then we should be able to move the data in either left or right direction as that register is called as a bi-directional register. A four bit bi-directional shift register is shown in Fig. 9.3. There are two serial inputs namely the serial right shift data input D_R and the serial left shift data input D_L alongwith a Mode control input (M).

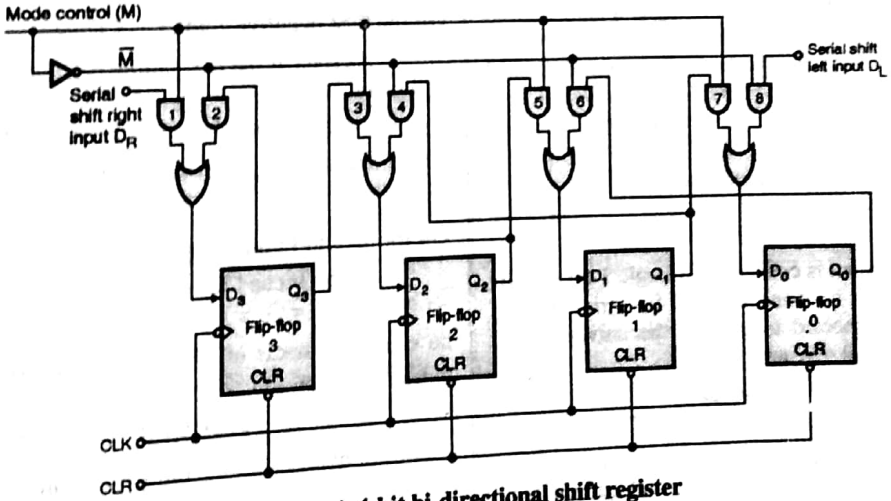


Fig. 9.3 : A 4-bit bi-directional shift register

Operation :

With M = 1 : Shift right operation : If M = 1, then the AND gates 1, 3, 5 and 7 are enabled whereas the remaining AND gates 2, 4, 6 and 8 will be disabled. Hence the data at D_R (shift right input) is shifted right bit by bit from FF-3 to FF-0 on the application of clock pulses. Thus with M = 1 we get the serial right shift operation.

With M = 0 : Shift left operation : When the mode control M is connected to "0" then the AND gates 2, 4, 6 and 8 are enabled while 1, 3, 5 and 7 are disabled. Therefore the data at D_L (shift left input) is shifted left bit by bit from FF-0 to FF-3 on application of the clock pulses. Thus with M = 0 we get the serial left shift operation. Note that M should be changed only when CLK = 0, otherwise the data stored in the register may get changed in an undesirable manner.

Q. 5 Explain 3 bit bidirectional shift register using JK flip flop. Draw the neat waveforms. Dec. 13

Ans. : A 3-bit bi-directional shift register using JK flip flops is shown in Fig. 9.4.

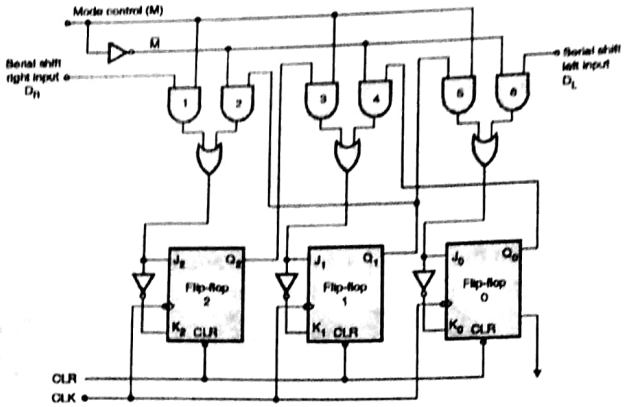


Fig.9.4 : A 3 bit bi-directional shift register using JK flip flops

This circuit is same as the 4 bit bi-directional shift register discussed in the previous section. The only modification is the inclusion of an inverter between the J and K inputs of each flip flops. With M = 1 the shift right operation will take place as discussed in the previous section and with M = 0 the shift left operation is going to take place.

Q. 6 Draw the diagram of a 4 bit universal shift register and explain its operation.

Dec. 02, Dec. 08, May 11, May 12, Dec. 12, May 14, May 16, Dec. 16

Ans. :

A shift register which can shift the data in only one direction is called as a unidirectional shift register. A shift register which can shift the data in both the directions is called as a bi-directional shift register. Applying the same logic, a shift register which can shift the data in both the directions (shift right or left) as well as load it parallely, then it is called as a **universal shift register.** Fig. 9.5 shows the logic diagram of a universal shift register. This shift register is capable of performing the following operations :

1. Parallel loading (parallel input parallel output)
2. Left shifting
3. Right shifting

The mode control input is connected to Logic 1 for parallel loading operation whereas it is connected to 0 for serial shifting. With mode control pin connected to ground, the universal shift register acts as a bi-directional register.

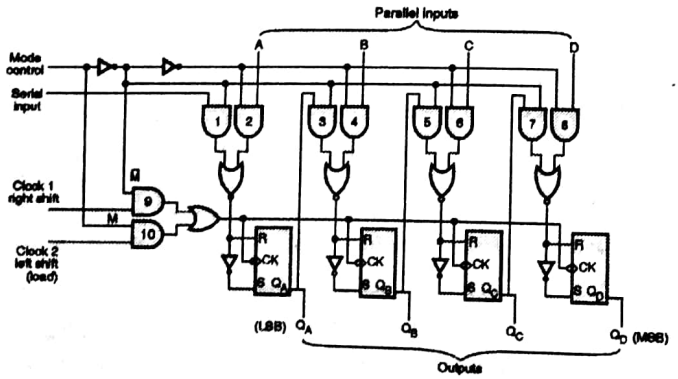


Fig.9.5 : Logic diagram of a universal shift register

For serial left operation, the input is applied to the serial input which goes to AND gate-1 in Fig. 9.5. Whereas for the shift right operation, the serial input is applied to D input (input of AND gate 8). The well known example of universal shift register in the IC form is IC7495.

Q. 7 Draw and explain the working of 4-bit ring counter with timing diagram.

May 12, Dec. 13, Dec. 12, Dec. 15

Ans. :

Fig. 9.6 shows a typical application of shift registers called Ring Counter. The connections reveal that they are similar to the connections for shift right operation, except for one change. Output of FF-3 is connected to data input D_0 of FF-0. Ring counter is a special type of shift register.

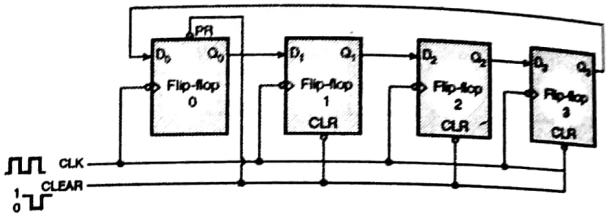


Fig. 9.6 : A four bit ring counter

Operation : Initially a low clear (CLR) pulse is applied to all the flip-flops. Hence FF-3, FF-2 and FF-1 will reset but FF-0 will be preset. So the outputs of the shift register are :

$$Q_3\ Q_2\ Q_1\ Q_0 = 0\ 0\ 0\ 1.$$

Now the clear terminal is made inactive by applying a high level to it. The clock signal is then applied to all the flip-flops simultaneously. Note that all the flip-flops are negative edge triggered.

On the first negative going CLK edge :

As soon as the first falling edge of the clock hits, only FF-1 will be set because $Q_0 = D_1 = 1$. The FF-0 will reset because $D_0 = Q_3 = 0$ and there is no change in the status of FF-2 and FF-3. Hence after the first clock pulse the outputs are

$$Q_3\ Q_2\ Q_1\ Q_0 = 0\ 0\ 1\ 0.$$

On the second falling edge of clock :

At the second falling edge of the clock, only FF-2 will be set because $D_2 = Q_1 = 1$. FF-1 will reset since $D_1 = Q_0 = 0$. There is no change in status of FF-3 and FF-0. So after the second clock pulse the outputs are,

$$Q_3\ Q_2\ Q_1\ Q_0 = 0\ 1\ 0\ 0.$$

Similarly after the third clock pulse the outputs are,

$$Q_3\ Q_2\ Q_1\ Q_0 = 1\ 0\ 0\ 0.$$

And after the fourth one the outputs are,

$$Q_3\ Q_2\ Q_1\ Q_0 = 0\ 0\ 0\ 1.$$

These are the outputs from where we started. Hence the operation repeats from this point onwards.

Number of output states :

The number of output states for a ring counter will always be equal to the number of flip-flops. So for a 4-bit ring counter the number of states is equal to 4. The operation of a four bit ring counter is summarized in Table 9.27.

Table 9.2 : Summary of operation of a ring counter

CLR	CLK	Q_0	Q_1	Q_2	Q_3	
⊓	×	1	0	0	0	← FF-0 Preset, others cleared
1	↓	0	1	0	0	
1	↓	0	0	1	0	
1	↓	0	0	0	1	
1	↓	1	0	0	0	

The presetted 1 follows a circular path to form a ring

Waveforms for the ring counter :

The waveforms for the 4-bit ring counter are as shown in Fig. 9.7. These waveforms clearly show that the presetted "1" shifts one bit per clock cycle and forms a ring. Hence the name ring counter.

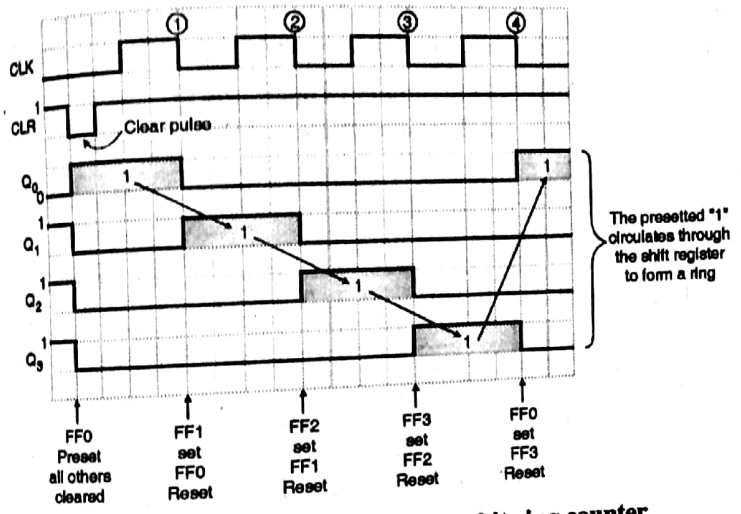


Fig. 9.7 : Waveforms of a four bit ring counter

Applications of ring counter :

Ring counters are used in those applications in which several operations are to be controlled in a sequential manner. For example in resistance welding the operations called squeeze, hold, weld and off are to be performed sequentially. We can use a ring counter to initiate these operations.

Ring counter using JK Flip Flop :

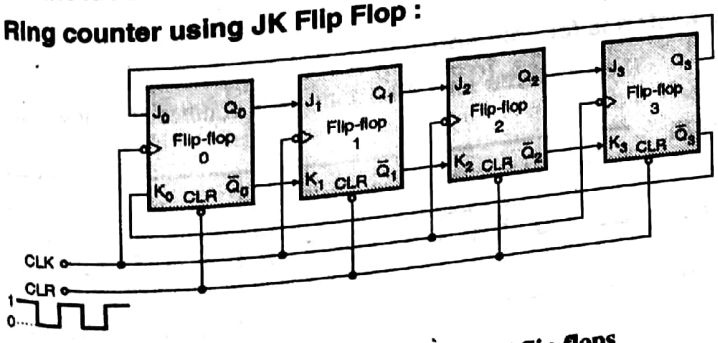


Fig. 9.7(a) : Ring counter using JK flip flops

A ring counter can also be constructed using the JK flip flops, as shown in Fig. 9.7(a).

Q. 8 : Draw a 3 bit ring counter and draw timing wave form. Prove that it is a divide by 3 network. May 11

Ans. :

The 3 bit ring counter is as shown in Fig. 9.8(a).

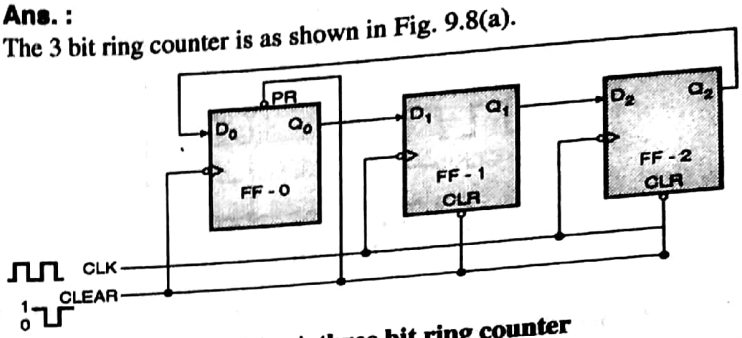


Fig.9.8(a) : A three bit ring counter

The operation of a three bit ring counter is summarized in Table 9.3

Table 9.3 : Summary of operation of a ring counter

CLR	CLK	Q_0	Q_1	Q_2
⊓	×	1	0	0
1	↓	0	1	0
1	↓	0	0	1
1	↓	1	0	0

Waveforms :

The waveforms of a 3-bit ring counter are as shown in Fig. 9.8(b).

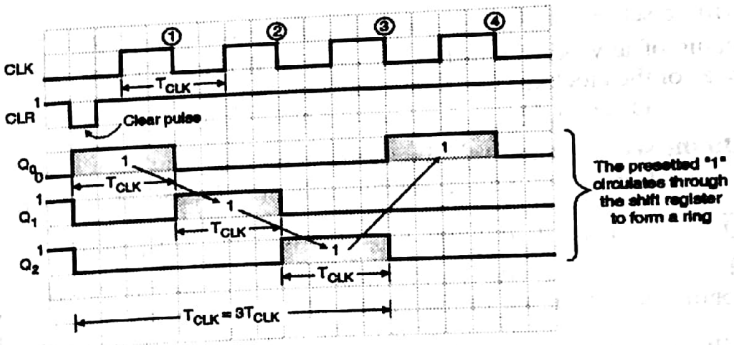


Fig.9.8(b) : Waveforms of a 3-bit ring counter

Compare the time periods of waveforms at Q_2 and CLK.

$$T_{out} = 3\,T_{CLK}$$

$$\therefore \quad f_{out} = \frac{1}{3} f_{CLK} \qquad \textbf{...Proved.}$$

Q. 9 Draw and explain the working of 4-bit twisted ring counter with timing diagram. May 14. May 15. Dec. 16

Ans. : In the ring counter the outputs of FF-3 were connected directly to the inputs of FF-0 i.e. Q_3 to J_0 $\bar{Q}_3$ to K_0. Instead if the outputs are cross coupled to the inputs i.e. if Q_3 is connected to K_0 and $\overline{Q}_3$ is connected to J_0 then the circuit is called as twisted ring counter or Johnson's counter. The Johnson's counter is shown in Fig. 9.9.

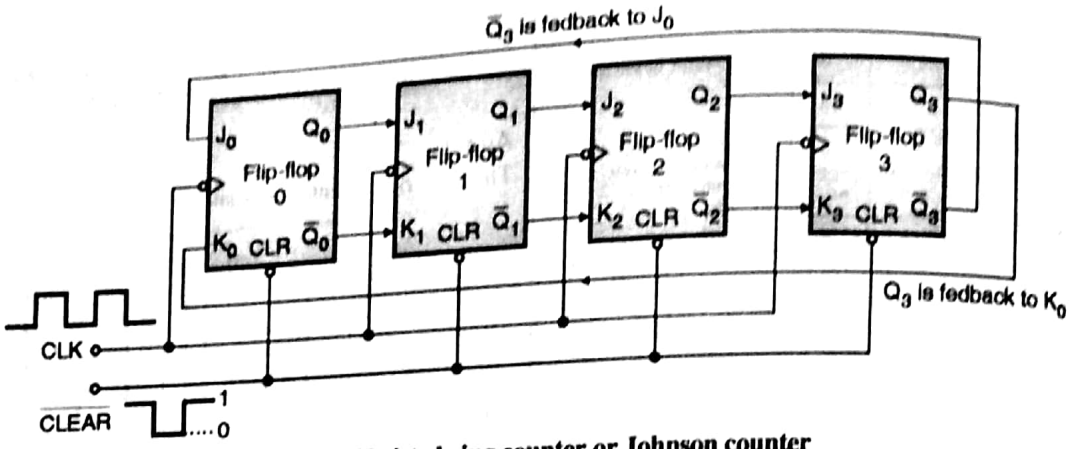


Fig. 9.9 : Twisted ring counter or Johnson counter

All the flip-flops are negative edge triggered, and clock pulses are applied to all of them simultaneously. The clear inputs of all the flip-flops are connected together and connected to an external clear signal. Note that all these clear inputs are active low inputs.

Operation :

Initially a short negative going pulse is applied to the $\overline{\text{clear}}$ input of all the flip-flops. This will reset all the flip-flops. Hence initially the outputs are,

$$Q_3\,Q_2\,Q_1\,Q_0 = 0000$$

But $\overline{Q}_3 = 1$ and since it is coupled to J_0 it is also equal to 1.

$$\therefore \quad J_0 = 1 \text{ and } K_0 = 0 \quad \ldots \text{Initially}$$

On the first falling edge of clock pulse :

As soon as the first negative edge of clock arrives, FF-0 will be set. Hence Q_0 will become 1. But there is no change in the status of any other flip-flop. Hence after the first negative going edge of the clock the flip-flop outputs are,

$$Q_3\,Q_2\,Q_1\,Q_0 = 0001$$

On the second negative going clock edge :

Before the second negative going clock edge, $Q_3 = 0$ and $\overline{Q}_3 = 1$. Hence $J_0 = 1$ and $K_0 = 1$. Also $Q_0 = 1$. Hence $J_1 = 1$. Hence as soon as the second falling clock edge arrives, FF-0 continues to be in the set mode and FF-1 will now set. Hence Q_1 will become 1 and $\overline{Q}_1 = 0$. There is no change in the status of any other flip-flop. Hence after the second clock edge the outputs are,

$$Q_3\,Q_2\,Q_1\,Q_0 = 0011$$

Similarly after the third clock pulse, the outputs are,

$$Q_3\,Q_2\,Q_1\,Q_0 = 0111$$

And after the fourth clock pulse, the outputs are,

$$Q_3\,Q_2\,Q_1\,Q_0 = 1111$$

Note that now $\overline{Q}_3 = 0$ i.e. $J_0 = 0$ and $K_0 = 1$.

Hence as soon as the fifth negative going clock pulse strikes, FF-0 will reset. But the outputs of the other flip-flops will remain unchanged. So after the fifth clock pulse, the outputs are,

$$Q_3\,Q_2\,Q_1\,Q_0 = 1110.$$

.. after the 5th clock pulse

This operation will continue till we reach the all zero output state. (i.e. $Q_3\,Q_2\,Q_1\,Q_0 = 0\,0\,0\,0$). The operation of Johnson's counter is summarised in Table 9.4.

Table 9.4 : Summary of operation of Johnson's counter

Error!	CLK	Q_3	Q_2	Q_1	Q_0	State number	Decimal equivalent
⊓	Initially	0	0	0	0	1	0
1	↓	0	0	0	1	2	1
1	↓	0	0	1	1	3	3
1	↓	0	1	1	1	4	7
1	↓	1	1	1	1	5	15
1	↓	1	1	1	0	6	14
1	↓	1	1	0	0	7	12
1	↓	1	0	0	0	8	8
1	↓	0	0	0	0	1	0

Note that there are 8 distinct states of output.

In general we can say that the number of states of a Johnson's counter is twice the number of flip-flops used. Therefore for a 4-flip-flop Johnson's counter, there are 8-distinct output states.

Waveforms for Johnson's counter :

The waveforms for a 4-bit Johnson's counter are shown in Fig. 9.10..

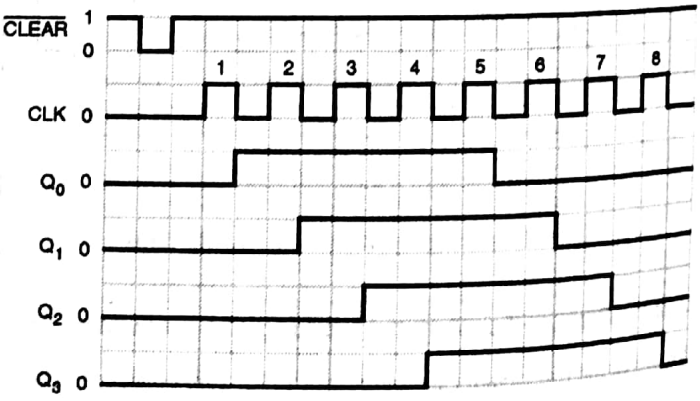


Fig. 9.10 : Waveforms of Johnson counter

Q. 1 Write short note on : VHDL.

Dec. 14. May 15. May 16

Ans. : The long form of VHDL is Very High Speed Integrated Circuit (VHSIC) hardware description language. **VHDL** is hardware description language used to form a digital system at many levels of ideas ranging from the algorithmic to the gate level. VHDL language defines the syntax as well as simulation semantics for each language. It is strong typed language which frequently contains too many words to write. VHDL is difficult to understand because it provides wide range of modeling capabilities but without learning the more complex features it is possible to incorporate a core subset of language which is simple and easy to understand. VHDL is an official IEEE standard which has become the industry standard language for explaining digital circuits. The original standard for VHDL called IEEE 1076 was accepted in 1987.

Q. 2 Write short note on : VHDL

Dec. 14. May 15. May 16

Ans. : Design units of VHDL code are independent components which are separately compiled and stored in library. VHDL program is composed of following design units :

1. Package (Optional)
2. Entity
3. Architecture
4. Configuration (Optional)

From the above design units, entity and architecture are necessary and others are optional. Design units of VHDL code is as shown in Fig. 10.1.

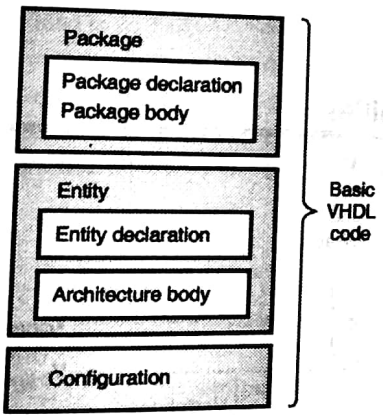


Fig. 10.1: Design units of VHDL code

In the following subsections we will discuss each design unit in detail.

1. **Package :** A VHDL package is a method to store and share some declarations which are common across many design units. Package is optional design unit. A set of declarations included in a package declaration can be shared by many design units. A set of useful type declarations, functions and procedures are declared by using the package and package body units. A package is represented by using package declaration and package body.

```
Package      package_name is
                  (declarations)
end     package _name ;
```

Fig. 10.2 : Package declaration

```
Package body package_name is
                  [Function and procedure descriptions]
end     package_name ;
```

Fig. 10.2(a): Package body

In a package declaration all declarations (type, subtype, subprogram) are encapsulated and package body contains behavior and definitions of subprograms declared in package declaration. A package declaration is necessary part while package body is mandatory when one or more subprograms are declared in package declaration. The name of package and package body should be same. As the entity configuration and package declarations can be compiled independently they are called as **'primary design units'**, while architecture and package bodies are compiled with reference to primary design unit. They are **"Secondary design units".**

2. **Entity :** The basic unit of VHDL hardware design is **entity.** The entity describes the interface between design and external environment. A VHDL entity states the parameters such as name and port of entity. All VHDL designs are generated using one or more entities. An entity is represented by using an **entity declaration** and an associated architecture body. An entity described input and output ports of a design.

Entity Declaration :

The entity declaration defines the entity name and lists the input and output ports. Fig. 10.2(b) shows syntax of entity.

```
entity entity_name is
        port ( port_name : mode port_type;
               port_name : mode port_type);
end [ entity_name ] ;
```

Fig. 10.2(b) : The general form of an entity declaration

In entity declaration, an entity starts with keyword **entity** which is followed by **name of entity** and keyword **is.** In entity declaration, names and identifiers can include letters, numbers and underscore (_) character, but it must starts with alphabet characters. In the next line port declarations are reserved with keyword **port.** The keyword **end** is always used to end an entity declaration which is followed by entity name. Here we can use statement end entity. VHDL language is not sensitive to white spaces, due to this alignment shown in entity syntax is not necessary. It is used for documentation. The communication between the entity and other models which occurs in its external environment takes place through signal is known as **ports.**

Port Modes in VHDL : In VHDL program, there are four modes available for ports. Table 10.1 shows port modes and its purpose in VHDL program.

Table 10.1 : Port modes

Sr. No.	Mode	Purpose
1.	In	Used for variable / signal i.e. an input to entity. This mode can read value, but they can't allow assignment of value to it. **In port** is

Sr. No.	Mode	Purpose
		unidirectional.
2.	Out	Used for signal i.e. an output from an entity. This mode can only write the value, they can't read a value from it. **Out port** is unidirectional.
3.	Inout	Bidirectional port. Both assignments to ports and read from port is allowed in **inout.**
4.	Buffer	Used to indicate that signal is an output of entity whose value can be read inside the entity architecture.

Q. 3 Write entity and architecture of D flip-flop with clear input using behavioural modeling. Dec 15

Ans. : Logic diagram :

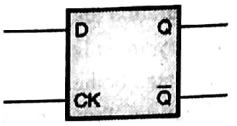


Fig. 10.3 : Logic symbol

Truth table of D flip flop :

Table 10.2: Truth table of a D flip flop

Q output		Input
Present state	Next state	D_n
0	0	0
0	1	1
1	0	0
1	1	1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
entity dff is
   Port ( D,clk : in std_logic;  Q: out std_logic);
end dff;
architecture Behavioral of dff is
begin
process(clk)
begin
if(clk'event and clk='1')then
Q<=D;

end if;
end process;

end Behavioral;
```

Chapter 11 : Digital Logic Families

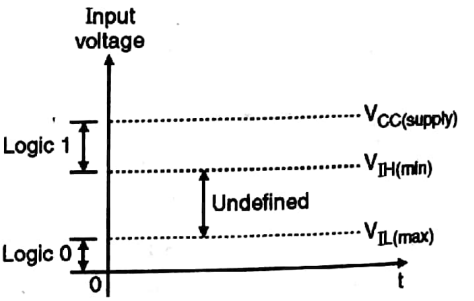
Q. 1 Write short note on current and voltage parameters of logic gates. Dec. 09

Ans. :

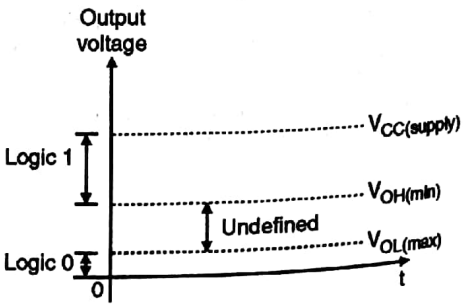
Voltage parameters (Threshold levels) :

Ideally the input voltage levels of 0 V and + 5 V (for TTL) are called as logic 0 and 1 levels respectively. However practically we won't always observe or obtain the voltage levels matching exactly to these values. Therefore it is necessary to define the worst case input voltages.

1. $V_{IL\ (max)}$ - **Worst case low level input voltage :** This is the maximum value of input voltage which is to be considered as a logic 0 level. If the input voltage is higher than $V_{IL\ (max)}$, then it will not be treated as a low (0) input level.
2. $V_{IH\ (min)}$ - **Worst case high level input voltage :** This is the minimum value of the input voltage which is to be considered as a logic 1 level. If the input voltage is lower than $V_{IH\ (min)}$, then it will not be treated as a High (1) input.
3. $V_{OH\ (min)}$ - **Worst case high level output voltage :** This is the minimum value of the output voltage which will be considered as a logic HIGH (1) level. If the output voltage is lower than this level then it won't be treated as a HIGH (1) output.
4. $V_{OL\ (max)}$ - **Worst case low level output voltage**



(a) Input voltage parameters



(b) Output voltage parameters

Fig. 11.1 : Voltage parameters

This is the maximum value of the output voltage which will be considered as a logic LOW (0) level. If the output voltage is higher than this value then it won't be treated as a LOW (0) output.

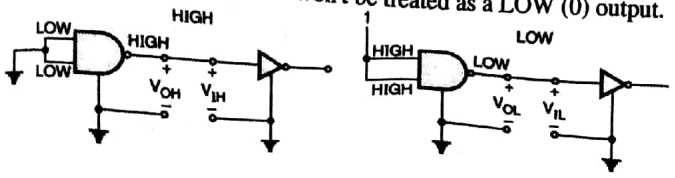


Fig. 11.2 : Voltage parameters on a logic circuit

All the voltage parameters are shown in Fig. 11.1. The voltage parameters can be shown on the digital circuit consisting of gates as shown in Fig. 11.2. Note that, the NAND and NOT gates shown, can be of TTL, ECL, CMOS or any other type.

Current parameters :

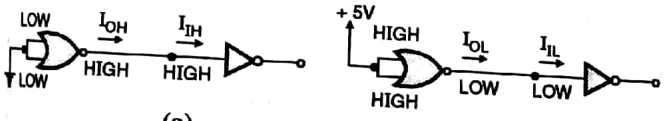
(a) **I_{IL} - Low level input current :** It is the current that flows into the input terminals when a low level input voltage in the specified range is applied.

(b) **I_{IH} - High level input current :** It is the current that flows into the input terminals when a high level input voltage in the specified range is applied.

(c) **I_{OL} - Low level output current :** This is the current that flows out of the output when the output voltage happens to be in the specified low (0) voltage range and a specified load is applied.

(d) **I_{OH} - High level output current :** This is the current flowing from the output when the output voltage happens to be in the specified HIGH (1) voltage range and a specified load is applied.

If the output current flows into the output terminal then it is called as a **sinking current** and if the output current flows out of the output terminal then it is called as a **sourcing current.** The current parameters are displayed on the logic circuit shown in Fig. 11.3.



(a) (b)

Fig. 11. 3 : Current parameters

Note that the actual current directions can be opposite to those shown in Fig. 11.3; depending on the logic family. Note that current flowing into a node or device is considered positive and current flowing out of node or device is considered negative.

Q. 2 Define parameters for CMOS family :
1. Fan out 2. Fan in. May 14

Ans. :

Fan-in : Fan-in is defined as the number of inputs a gate has. For example a two inpyut gate will have a fan-in equal to 2.

Fan-out : Fan-out is defined as the maximum number of inputs of the same IC family that a gate can drive without falling outside the specified output voltage limits. Higher fan out indicates higher the current supplying capacity of a gate. For example, a fan out of 5 indicates that the gate can drive (supply current to) at the most 5 inputs of the same IC family.

Q. 3 Explain with a neat diagram 2 input TTL NAND gate in detail. List important characteristics of TTL family. Dec. 11, Dec. 14

Ans. : A two input TTL-NAND gate is shown in Fig. 11.4(a). A and B are the two inputs while Y is the output terminal of this NAND gate.

Operation : In order to understand the operation of this circuit, let us replace transistor Q_1 by its equivalent circuit as shown in Fig. 11.4(b)

1. A and B are the input terminals. The input voltages A and B can be either LOW (zero volt ideally) or HIGH (+ V_{CC} ideally).

2. **A and B both LOW :** If A and B both are connected to ground, then both the B-E junctions of transistor Q_1 are forward biased.

Hence diodes D_1 and D_2 in Fig. 11.4(a) will conduct to force the voltage at point C in Fig. 11.4(a) to 0.7 V. This voltage is insufficient to forward bias base-emitter junction of Q_2 due to the presence of D_3. Hence Q_2 will remain OFF. Therefore its collector voltage V_X rises to V_{CC}. As transistor Q_3 is operating in the emitter follower mode, output Y will be pulled up to high voltage.

$\therefore$ Y = 1 (HIGH) ...For A = B = 0 (LOW)

The equivalent circuit for this input condition is shown in Fig. 11.5(a).

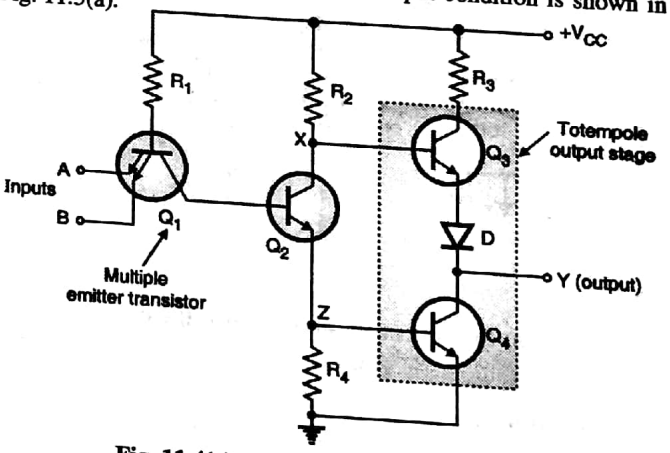


Fig. 11.4(a) : Two input TTL NAND gate

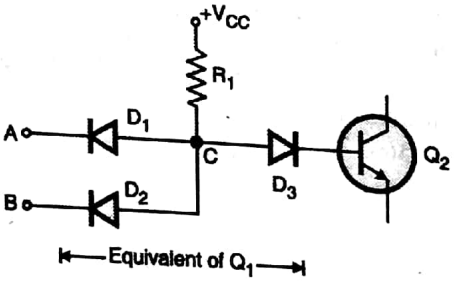


Fig. 11.4(b) : Transistor Q_1 is replaced by its equivalent

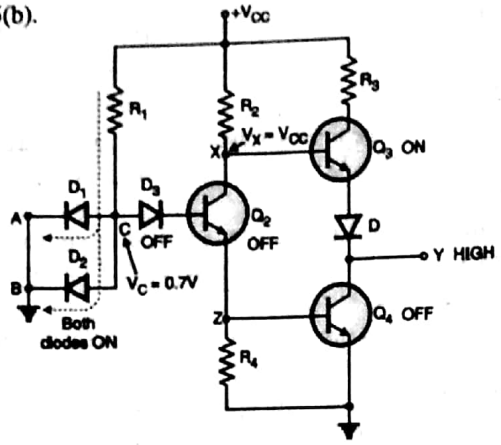
3. **Either A or B LOW :** If any one input (A or B) is connected to ground with the other terminal left open or connected to + V_{CC}, then the corresponding diode (D_1 or D_2) will conduct.

This will pull down the voltage at "C" to 0.7 V. (Fig. 11.4(a)) This voltage is insufficient to turn ON D_3 and Q_2. So it remains OFF. So collector voltage V_X of Q_2 will

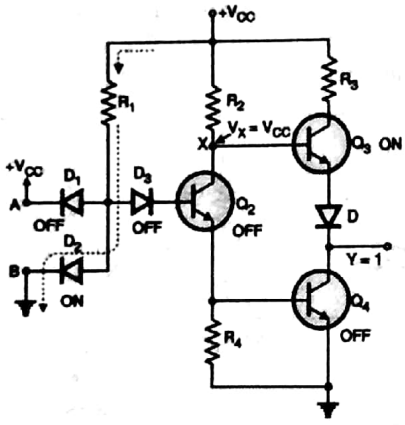
be equal to V_{CC}. This voltage acts as base voltage for Q_3. As Q_3 acts as an emitter follower, output Y will be pulled to V_{CC}.

$$\therefore Y = 1 \begin{cases} \text{if } A = 0 \text{ and } B = 1 \\ \text{if } A = 1 \text{ and } B = 0 \end{cases}$$

The equivalent circuit for this mode is shown in Fig. 11.5(b).



(a) Equivalent circuit for A = B = 0



(b) Equivalent circuit for A = 1, B = 0

Fig. 11.5

4. A and B both HIGH

If A and B both are connected to + V_{CC}, then both the diodes D_1 and D_2 will be reverse biased and do not conduct. Therefore voltage at point "C" i.e. at the anode of D_3 increases to a sufficiently high value.

Therefore diode D_3 is forward biased and base current is supplied to transistor Q_2 via R_1 and D_3 as shown in Fig. 11.5(c).

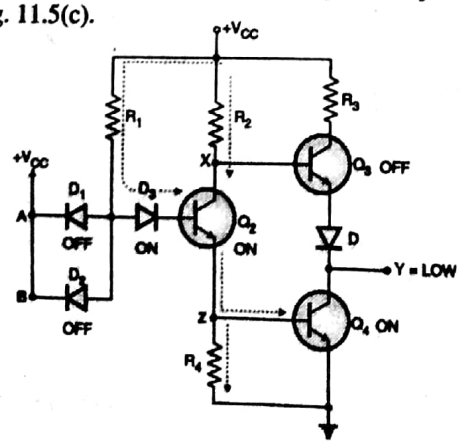


Fig. 11.5(c) : Equivalent circuit for A = B = 1

As Q_2 conducts, the voltage at X will drop down and Q_3 will be OFF, whereas voltage at Z (across R_3) will increase to a sufficient level to turn ON Q_4. As Q_4 goes into saturation, the output voltage Y will be pulled down to a low voltage.

$$\therefore Y = 0 \quad \text{...For } A = B = 1$$

The equivalent circuit for this mode of operation is shown in Fig. 11.5(c).

Q. 4 Define the various typical characteristics of a TTL gate.

Dec. 02, May 03, Dec. 03, Dec. 05, May 09

Ans. :

The Texas instrument first introduced two TTL series namely 54 series and 74 series.

Supply voltage and temperature ranges :

Table 11.1 gives the supply voltage and temperature ranges for two TTL families.

Table 11.1

TTL series	Supply voltage range	Temperature range
74 series	4.75 V to 5.25 V	0° C to 70° C
54 series	4.5 V to 5.5 V	– 55° C to 125° C

Voltage levels :

Table 11.2 shows the input and output logic voltage levels for the TTL 74 series.

Table 11.2 : Voltage levels for TTL 74 series

Voltages	Minimum	Typical	Maximum
V_{OL}	–	0.2 V	0.4 V
V_{OH}	2.4 V	3.4 V	–
V_{IL}	–	–	0.8 V
V_{IH}	2.0 V	–	–

Noise margin :

We have already defined the noise margins as,

High level noise margin, $V_{NH} = V_{OH\,(min)} - V_{IH\,(min)}$

and Low level noise margin, $V_{NL} = V_{IL\,(max)} - V_{OL\,(max)}$

Substituting the values from Table 11.9.2, the noise margin for the TTL logic families can be calculated.

$$\therefore V_{NH} = 2.4 - 2 = 0.4 \text{ V}. V_{NL} = 0.8 - 0.4 = 0.4 \text{ V}.$$

Thus noise margin for the TTL family is 0.4 V. This means that as long as the induced noise voltage is less than 0.4 V, the operation of the TTL ICs will be unaffected.

Power dissipation :

The average power dissipation for the standard TTL 74 series is approximately 10 mW. It is dependent on the parameters such as tolerances, signal level etc.

Propagation delay :

The propagation delay of a standard TTL gate is approximately 10 nanoseconds (1 nanosecond = 10^{-9} seconds).

A standard TTL gate is capable of driving at the most 10 other TTL gates simultaneously, maintaining its output voltage within the specified limits. Hence fan out of a TTL gate is 10. All the important characteristics of TTL logic family are listed in Table 11.3.

Table 11.3 : Important parameters of TTL logic family

Sr. No.	Parameter	Values
1.	Supply voltage	74 series : (4.75 to 5.25 V) 54 series : (4.5 to 5.5 V)
2.	Temperature range	74 series : 0 to 70° C 54 series : – 55 to 125° C
3.	Voltage levels	$V_{IL\ (max)} = 0.8$ V, $V_{OL\ (max)} = 0.4$ V $V_{IH\ (min)} = 2$ V, $V_{OH\ (min)} = 2.4$ V
4.	Noise margin	0.4 V
5.	Power dissipation	10 mW
6.	Propagation delay	10 nanosec.
7.	Fan out	10
8.	Figure of merit	100

Q.5 Explain the interfacing of CMOS and TTL gates. May 05, Dec. 05, Dec. 06

Ans. :

When we interface different types of ICs, it is necessary to check whether the driving IC is capable of meeting the current and voltage requirements of the load IC or not. For example Table 11.4 reveals that the output current capability of TTL ICs is much higher than the input current values of CMOS ICs. Therefore there is no problem for a TTL IC to drive CMOS as far as current is concerned. But there is a problem when we compare the voltage levels of TTL and CMOS. Because $V_{OH\ (min)}$ of a TTL series is very low as compared with $V_{IH\ (min)}$ required for the CMOS series like 400B, 74 HC or 74 AC. In such situations, something has to done to increase the level of TTL output voltage to an acceptable voltage level for CMOS.

Table 11.4 : Input output currents for standard devices with a supply voltage of 5 V

Parameter	CMOS		TTL		
	4000 B	74 HC/HCT	74	74 LS	74 AS
$I_{IH\ (max)}$	1 µA	1 µA	40 µA	20 µA	20 µA
$I_{IL\ (max)}$	1 µA	1 µA	1.6 mA	0.4 mA	0.5 mA
$I_{OH\ (max)}$	0.4 mA	4 mA	0.4 mA	0.4 mA	2 mA
$I_{OL\ (max)}$	0.4 mA	4 mA	16 mA	8 mA	20 mA

The situation in which a TTL IC is driving CMOS is shown in Fig. 11.16. Note the presence of pull up resistor at the output of the TTL gate. Due to this resistor the TTL output will rise to approximately +5 V in its HIGH state. This will provide the sufficient voltage level at the input of the CMOS gate.

Such a pull up resistance is not required if the CMOS gate belongs to 74 HCT or 74 ACT family, because these families can accept the TTL outputs directly. In other words they are TTL compatible CMOS families.

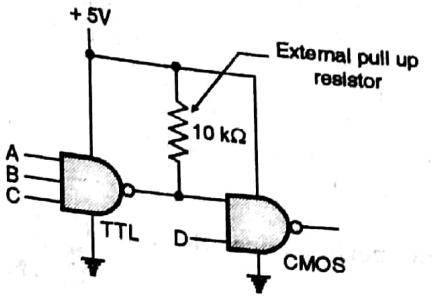


Fig. 11.6 : TTL driving CMOS

Q. 6 Compare TTL and CMOS logic. May 16

Ans. :

Sr. No.	Parameter	CMOS	TTL
1.	Device used	N-channel MOSFET and P-channel MOSFET	Bipolar junction transistor
2.	$V_{IH\ (min)}$	3.5 V (V_{DD} = 5 V)	2 V
3.	$V_{IL\ (max)}$	1.5 V	0.8 V
4.	$V_{OH\ (min)}$	4.95 V	2.7 V
5.	$V_{OL\ (max)}$	0.05 V	0.4 V
6.	High level noise margin	V_{NH} = 1.45 V	0.4 V
7.	Low level noise margin	V_{NL} = 1.45 V	0.4 V
8.	Noise immunity	Better than TTL	Less than CMOS
9.	Propagation delay	105 nS (Metal gate CMOS)	10 nS. (Standard TTL)
10.	Switching speed	Less than TTL.	Faster than CMOS
11.	Power dissipation per gate.	P_D = 0.1 mW. Hence used for battery backup applications	10 mW
12.	Speed power product.	10.5 pJ	100 pJ
13.	Dependence of P_D on frequency	P_D increases with increase in frequency.	P_D does not depend on frequency.
14.	Fan out	Typically 50.	10
15.	Unconnected inputs	Unused inputs should be returned to GND or V_{DD}. They should never be left floating.	Inputs can remain floating. The floating inputs are treated as logic 1s.
16.	Component density	More than TTL since MOSFETs	Less than CMOS since

Sr. No.	Parameter	CMOS	TTL
		need smaller space while fabricating an IC.	BJT needs more space.
17.	Operating areas	MOSFETs are operated as switches. i.e. in the ohmic region or cut off region.	Transistors are operated in saturation or cut off regions.
18.	Power supply voltage	Flexible from 3 V to 15 V.	Fixed equal to 5 V.

Q. 7 Compare the performance of TTL, CMOS and ECL logic. Dec. 14

Ans. :

Sr. No.	Parameter	TTL	CMOS	ECL
1.	Components used.	Transistors diodes and resistors	MOSFETs	Resistors and transistors
2.	Fan out.	Moderate	Highest	High
3.	Propagation delay.	10 nS	70 nS	2 nS
4.	Noise margin.	Moderate	High	Low
5.	Power dissipation per gate	10 mW	0.1 mW	40-50 mW
6.	Speed power product	100 pJ	0.7 pJ	40-100 pJ
7.	Circuit complexity	Complex	Moderately complex	Complex
8.	Basic gate	NAND	NAND/NOR	OR/NOR
9.	Applications	Lab and demonstration equipment	Portable equipments as they consume less power	High speed switching applications due to high speed.

□□□