Decimal Number System:-

In these Number system Digits are 0,1,2,3,4,5,6,7,8,9 base (or) Radix is 10.

$(d_n \times 10^n) + (d_{n-1} \times 10^{n-2}) + \cdots + (d_1 \times 10^1) + (d_0 \times 10^0) + (d_{-1} \times 10^{-1})$.

Consider the decimal number 9256.26 (MSD = 9, LSD = 6) using digits 0,5,6,9 hence

$$9256.26 = 9\times1000 + 2\times100 + 5\times10 + 6\times1 + 2\times(1/10) + 6\times(1/100)$$

$$= 9\times10^3 + 2\times10^2 + 5\times10^1 + 6\times10^0 + 2\times10^{-1} + 6\times10^{-2}$$

9's Compliment & 10's Compliment

(i) 3465

9999
3465
6534

(ii) 782.54

999.99
782.54
217.45

(iii) 4526.075

9999.999
4526.075
5473.924

* 10's Complement = 1 + 9's complement

(i) 9999
3465
6534
1
6535

(ii) 999.99
784.54
217.45
1
217.46

(iii) 9999.999
4526.075
5473.924
1
5473.925

(a) 4069

9999
4069
5930
1
5931

(b) 1054.074

9999.999
1054.074
8945.925
1
8945.926.

9's complement method of subtraction:-

to perform the decimal subtraction using 9's complement method.

1) Calculate the 9's Compliment of substrahand
2) And the add that answer to the minuent
3) If there is any carry the answer is positive add that carry to the LSD
4) If there is carry answer is Negative and final answer is 9's complement of result.

a) 745 - 436

745	999
436	436
309	563

745
563
308
1
309

4) Number system

Number system are 4 types they are

1) Decimal $r=10 \rightarrow [0,1,2,5,4,5,6,7,8,9]$
2) Binary $r=2 \rightarrow [0,1]$
3) octal $r=8 \rightarrow [0,1,2,3,4,5,6,7,8]$
4) Hexa decimal $r=16 \rightarrow [0,1,2,3,4,5,6,7,8$ A, B, C, D, E, F$]$

Positional weights

1) $[4\ 6\ 8\ 2]_{10}$

$3\ \ 2\ \ 1\ \ 0$

$10^3\ 10^2\ 10^1\ 10^0$

$4\times10^3+6\times10^2+8\times10+2\times1$

$4\times1000+6\times100+80+2$

$4000+600+80+2$

4682

2) $[A\ B\ 2\ 4\ 8]_{16}$

$4\ \ 3\ \ 2\ \ 1\ \ 0$

$16^4\ 16^3\ 16^2\ 16^1\ 16^0$

$A\times16^4+B\times16^3+2\times16^2+4\times16^1+8\times1$

$65536A+4096B+512+64+8$

$65536A+4096B+584$

6) $[425.62]_{10}$

$2\ 1\ 0\ -1\ -2$

$10^2\ 10^1\ 10^0\ 10^{-1}\ 10^{-2}$

3) $[0.6211]_{10}$

$-1\ -2\ -3$

$10^{-1}\ 10^{-2}\ 10^{-3}$

$10^{-1}6+10^{-2}2+10^{-3}11$

$=0.6+0.02+0.004$

$=0.624$

4) $[0.CDF]_{10}$

$-1\ -2\ -3$

$10^{-1}\ 10^{-2}\ 10^{-3}$

5) $[1011]_2$

$3\ 2\ 1\ 0$

$2^3\ 2^2\ 2^1\ 2^0$

Conversion from given n·s to (Binary, Decimal, Hexa decimal,

* Conversion from to decimal

Multiply the each digit of the given number system with position weight and final add all the product to get the final result (or) equivalent decimal number.

1) $[1\ 0\ 1\ 0]_2$

$3\ 2\ 1\ 0$

$2^3\ 2^2\ 2^1\ 2^0$

$8\times1+4\times0+2\times1+0$

$8+2+0$

$=[10]_{10}$

2) $[11.01]$

$2\ 1\ 0\ -1\ -2$

$2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}$

$4\times1+2\times1+1\times0+2^{-2}\times1$

$4+2+0.25$

$6.25,$

Number base conversions

→ Decimal to {Binary, octal & Hexa Decimal}
↓ consective multiplications/ divisions

- Decimal to Binary

Convert the following decimal numbers to its equivalent Binary numbers.

i) 625
ii) 1001
iii) 108

i)
2 | 625
2 | 312 → 1
2 | 156 → 0
2 | 78 → 0
2 | 39 → 01
2 | 19 → 1
2 | 9 → 1
2 | 4 → 1
2 | 2 → 0
1

100111001

②
2 | 108
2 | 54 → 0
2 | 27 → 0
2 | 13 → 1
2 | 6 → 1
2 | 3 → 0
2 | 1 → 1
0 1

1101100

③
2 | 1001
2 | 500 → 1
2 | 250 → 0
2 | 125 → 0
2 | 62 → 1
2 | 31 → 0
2 | 15 → 1
2 | 7 → 1
2 | 3 → 1
2 | 1 → 1
0 1

1111101001

→ Convert the following decimal numbers to Binary

i) 0.125
ii) 0.65
iii) 0.25

i) 0.125 × 2 = 0.25
0.25 × 2 = 0.50
0.5 × 2 = 1.0

$(0.125)_{10} = (0.001)_2$

ii) 0.65 × 2 = 1.30
1.30 × 2 = 2.60

convert the following decimal numbers

i) 625
ii) 108
iii) 11011

i)
8 | 625
8 | 78 → 1
8 | 9 → 6
8 | 1 → 1
0 → 1

$(625)_{10} = (1161)_8$

ii)
8 | 108
8 | 13 → 4
8 | 1 → 5 ↑
0 → 1

$(108)_{10} = 4 \Rightarrow (154)_8$

convert decimal into octa hedral

i) 0.15
ii) 0.25

0.15 × 8 = 1.2
0.2 × 8 = 1.6
0.6 × 8 = 4.8
0.8 × 8 = 6.4
0.4 × 8 = 3.2
0.2 × 8 = 1.6

$(0.15)_{10} = (0.114631)_8$

convert decimal into hexa decimal.

i) 108
ii) 1001

i)
16 | 108
16 | 6 → 0C
0 → 6 ↑

$(108)_{10} = (6C)_{16}$

ii)
16 | 1001
16 | 62 → 9
16 | 3 → E ↑
0 → 3

$(1001)_{10} = (3E9)_{16}$

convert decimal into hexa decimal.

i) 0.125

$0.125 \times 16 = 2.04$

$(0.125)_{10} = (0.2)_{16}$

Any other to decimal

$(\pm d_4 d_3 d_2 d_1 d_0 \cdot d_{-1} d_{-2} d_{-3} \cdots) = ($

$+ d_3 \times 2^2 + d_2 \times 2^1 + d_0 \times 2^0 \cdot d_{-1} \times 2^{-1} + d_{-2} \times 2^{-2} + d_{-3} \times 2^{-3}$

i) $(101101.12)_2$

ii) $(101101.11)_8$

iii) $(101101.11)_{16}$

i) 101101.11

5 4 3 2 1 0 -1 -2

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

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

$32 + 0 + 8 + 4 + 0 + 1 \cdot 0.5 + 0.25$

$= 45.75$

ii) $(101101.11)_8$

1 0 1 1 0 1 . 1 1

5 4 3 2 1 0 -1 -2

$8^5\ 8^4\ 8^3\ 8^2\ 8^1\ 8^0 \cdot 8^{-1}\ 8^{-2}$

$1 \times 8^5 + 0 \times 8^4 + 1 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 \cdot 1 \times 8^{-1} + 1 \times 8^{-2}$

$32768 + 0 + 512 + 0 + 64 + 1 \cdot 0.125 + 0.015625$

$(33345.140$

iii) $(10\,11\,01.11)_{16}$

1 0 11 0 1 . 11

5 4 3 2 1 0 -1 -2

$16^5\ 16^4\ 16^3\ 16^2\ 16^1\ 16^0 \cdot 16^{-1}\ 16^{-2}$

$1 \times 16^5 + 0 \times 16^4 + 1 \times 16^3 + 1 \times 16^2 + 0 \times 16^1 + 1 \times 16^0 \cdot 1 \times 16^{-1} + 1 \times 16^{-2}$

$16\ 1048576 + 4096 + 256 + 1 \cdot 0.065 + 0.003$

$(1052929.065)_{16}$

Decimal	Octa	hexadecimal	Binary (8421
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	01101
6	6	6	0110
7	7	7	0111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	11101
14	16	E	1110
15	17	F	1111
16	20	10	10000
17	21	11	10001
18	22	12	10010
19	23	13	10011

20	24	14	10100
21	25	15	10101
22	26	16	10110
23	27	17	10111
24	30	18	11000
25	31	19	11001
26	32	1A	11010
27	33	1B	11011
28	34	1C	11100
29	35	1D	11101
30	36	1E	11110
31	37	1F	11111
32	38 #0	20	100000
33	39 #1	21	100001
34	#2	22	100010
35	#3	23	100011
36	#4	24	100100
37	#5	25	100101
38	#6	26	100110
39	#7	27	100111
#0	50	28	101000
#1	51	29	101001
#2	52	2A	101010
#3	53	2B	101011
#4	54	2C	101100
#5	55	2D	101101
#6	56	2E	101110
#7	57	2F	101111
#8	60	30	110000
#9	61	31	110001
50	62	32	

Binary operations:-

1) Binary Addition
2) Binary subtraction
3) Binary Multiplication

① Binary Addition:-

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

② Binary subtraction

A	B	a-b diff	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

③ Binary multiplication

A	B	A×B
0	0	0
0	1	0
1	0	0
1	1	1

① 1000 × 0100 = 1100

② 111 + 011 = 1010

③ 101 − 110 = 1011

④ 100 × 011 = 001

⑤ 1000 ÷ 101 = 0011

Compliments of the numbers system:

	r-1	r
1) Decimal (r=10)	9	10
2) Binary (r=2)	1	2
3) Octal (r=8)	7	8
4) Hexadecimal (r=16)	15	16

Subtraction using 10's complement.

1) Calculate the 10's complement of subtrahend.
2) Add the 10's compliment to the minents.
3) If there is end around carry is generated the result is positive & it is obtained by neglecting the carry.
4) If there is no end carry generated result is negative and answer is obtained by 10's compliment of that value and put negative sign.

1) $(1204)_{10} - (982)_{10} = (222)_{10}$ (min, sub)

```
 999        1204
 982     (+)018
 ---        ---
 017        222
   1
 ---
 018
```

Here the carry is generated so answer '+ve' by neglecting carry.

2) $(982)_{10} - (1204)_{10} = (-222)_{10}$ (min, sub)

```
9999      982       9999
1204     8796       9778
----     ----       ----
8795     9778       0221
   1                   1
----                ----
8796 →10's           222
```

* $(2928.54)_{10} - (416.73)_{10}$

```
 999.99       2928.54
 416.73        583.27
 ------       -------
 583.26       2511.81
      1
 ------
 583.27
```

* $(416.73)_{10} - (2928.54)_{10} = -2511.81$

```
  9999.99      7071.46      9999.99
(-)2928.54   (+)416.73      7488.19
  -------      -------     --------
  7071.45      7488.19     -2511.80
        +1                       +1
  -------                  --------
  7071.46                  -2511.80
```

Binary number system:

* In this number system base (or) radix is '2'
* Here we have 2 symbols / 2 digits / 2 bits
* It is a positional weighted sum. Left most digit is called 'MSB'. Right most digit is called "LSB".

1011

1011.11 (MSB, LSB)

Integer part, fractional part

* Consider any general binary number.
* The decimal value for this number is given by

$$d_n 2^n + d_{n-1} 2^{n-1} + d_{n-2} 2^{n-2} + \dots + d_3 2^3 + d_2 2^2 + d_1 2^1 + d_0 2^0 + d_{-1} 2^{-1} + \dots d_{-k} 2^{-n}$$

Prob:- $(10100)_2 = (20)_{10}$

$= 2^4 \times 1 + 2^3 \times 0 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 0$

$= 16 + 0 + 4 + 0 + 0$

$= (20)_{10}$

Prob:- $(10110.110)_2 = (22.75)_{10}$

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

$= 16 + 0 + 4 + 2 + 0 \cdot 0.5 + 0.25 + 0$

$= 22.75$

Conversion of decimal to binary:

1) Integer part is successfully divided by '2' by keeping all remainder till Quotient becomes '0'.
2) Collecting all remainders from bottom to top gives the integer part of the binary number.
3) Fraction part is to be succisively multiply by '2' by all integer part of the product till fraction part becomes zero or atleast 5 digits.

1) $(105.15)_{10}$
I F

2 | 105		
2 | 52	→ 1	
2 | 26	→ 0	
2 | 13	→ 0	
2 | 6	→ 1	
2 | 3	→ 0	
2 | 1	→ 1	
0	→ 1	

$0.15 \times 2 = 0.30$
$0.3 \times 2 = 0.6$
$0.6 \times 2 = 1.2$
$0.2 \times 2 = 0.4$
$0.4 \times 2 = 0.8$
$(0.15)_{10} = (001001)$

$105 = (1101001)_2$ $(0.15)_{10}$

$(105.15)_{10} = (1101001.001001)$

2) $(625)_{10} = (\ )_2$

2 | 625	
2 | 312	1
2 | 156	0
2 | 78	0
2 | 39	0
2 | 19	1
2 | 9	1
2 | 4	1
2 | 2	0
2 | 1	→ 0

$625 = (1001110001)_2$

3) $(52.75)_{10}$

2 | 52	
2 | 26	0
2 | 13	0
2 | 6	1
2 | 3	0
1	1

$(52) = 110100$

$0.75 \times 2 = 1.5$
$0.5 \times 2 = 1.0$
$0.75 = 11$

$(52.75)_{10} = (110100.11)$

4) $(163.875)_{10} =$

2 | 163	
2 | 81	1
2 | 40	1
2 | 20	0
2 | 10	0
2 | 5	0
2 | 2	1
1	0

$0.875 \times 2 = 1.75$
$0.75 \times 2 = 1.50$
$0.5 \times 2 = 1.0$
$(0.875)_{10} = (111)_2$

$163 = 10100011$

$(163.875)_{10} = (10100011.111)$

Bi
-Applications of Binary system:

↳ Binary number system is used in digital computers because switching circuits used in this computers are always to state digit divisions only.
ex:- diodes, transistors.

—Here 0 and 1 belongs to closed or open switch and on or off condition.

Binary addition

$0+0=0$;
$0+1=1$
$1+0=1$
$1+1=0$ with Carry 1

ex:- $(1011)_2 + (1100)_2$

$$\begin{array}{r} 1011 \\ 1100 \\ \hline 10111 \end{array}$$

$$\begin{array}{r} 1111 \\ 1011 \\ \hline 11010 \end{array}$$

↳ 1001.110

$$\begin{array}{r} 1101.010 \\ \hline 10111.000 \end{array}$$

Binary subtraction:-

rules:

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 with borrow 1

Ex:- $(1101)_2 - (1000)_2$ → 1101 − 1000 = 0101

↳ 1101 − 0011 = 1010

↳ 1000 − 0001 = 0111

(or)

↳ 1011.110 − 0100.001 = 0011.101

Binary Multiplication:-

rules: 0×0=0, 0×1=0, 1×0=0, 1×1=1

↳ 1011 × 11

1011
1011 ×
1000 01 = 33

↳ $(1011.101)_2 \times (101.01)_2$

= 1011.101
000 0 000 ×
101 1.101 ××
0000.000 ×××
101 110 1 ××××
111101.00001

4 → 1000
100

→ 111101.00001

Binary division:-

Divide 1010 with 10

$1010_{10} \div 10_{2} = 101_{5}$

10) 1010 (101
10
10
10
0

↳ $101101_{45} \div 110_{6} = 111.1_{7.5}$

110) 101101 (111.1
110
1010
110
1001
110
0110
110
0

Octal number system:

↳ In this number system base or radix is '8'.
↳ symbols or digits are 0, 1, 2, 3, 4, 5, 6, 7
↳ This is a positional weighted system.

Ex:- (52) MSD → LSD

527.5 MSD → LSD

The decimal value for this number is given by

$d_n 8^n + d_{n-1} 8^{n-1} + d_{n-2} 8^{n-2} + \cdots + d_3 8^3 + d_2 8^2 + d_1 8^1 + d_0 8^0 + d_{-1} 8^{-1} + \cdots d_{-m} 8^{-m}$

Octal to decimal.

Ex:- $(2376)_8 = (1278)_{10}$

$2\times8^3 + 3\times8^2 + 7\times8^1 + 6\times8^0$
$= 1024 + 192 + 56 + 6$
$= (1278)_{10}$

Co ↳ $(77.84)_8$

$= 7\times8^1 + 7\times8^0 + 8\times8^{-1} + 4\times8^{-2}$

al $= 56 + 7 + 1 + 0.0625$

2) $= 63 + 1.0625$

$= 64.0625$

3) Decimal to Octal

↳ $(968)_{10} = (\quad)_8$

8	968	
8	121	→ 0
8	15	→ 1
8	1	→ 7
	0	→ 1

↳ $(1710)_8$

↳ $(69.625)_{10} = (\quad)_8$

8	69	
8	8	→ 5
8	1	→ 0
	0	→ 1

$0.625 \times 8 = 5.0$

$0.625 \times 8 = 1.250$

$0.25 \times 8 = 0.5$

$0.5 \times 8 = 1.0$

$(5)_8$

$69 = (105)_8$

$(69.625)_8 = (105.5)_8$

Counting in Octal number system:

Decimal	Octal	Binary
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111
8	10	001 000
9	11	001 001
10	12	001 010
11	13	001 011
12	14	001 100
13	15	001 101

↳ $(246)_8 = (\quad)_2$

$(010\ 100\ 110)_2$

Conversion of binary to octal:

$(101\ 101\ 101\ 111)_2 = (5557)_8$

$(01\ 101\ 110)_2 = (156)_8$

$(10\ 111\ 011\ 010.\ 001\ 1)_2 = (13552.14)_8$

Heradecimal Number System:

↳ In this number system Base (or) radix is 16.

↳ Symbols or digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

↳ 12.AF

$(12.AF)_{16} = (\quad)_{10}$

$1\times16^{1}+2\times16^{0}+A\times16^{-1}+F\times16^{-2}$

$16+2+0.683$

$(18.683)_{10}$

↳ $(2874)_{10} = (B3A)_{16}$

16	2874	
16	179	$-10-A$
16	11	3
	0	$-11 \rightarrow B$

↳ $(2598)_{10} = (A26)_{16}$

16	2598	
16	162	-6
16	10	-2
	0	$-10-A$

↳ $(0.675)_{10} = (0.ACC)_{16}$

$0.675\times16 = 10.28$

$0.8\times16 = 12.8$

$0.8\times16 = 12.8$

↳ counting in Hexadecimal

Decimal	Hexa	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

* $(0AF)_{16} \rightarrow (\quad)_2$

$(0010)\ 1010\ 1111)_2$

* $(12EE.EF)_{16}$

$(0001\ 0010\ 1110\ 1110.1110\ 1111)_2$

Conversion of binary to hexa.

↳ $(0\ 1101\ 1101\ 0111.1010\ 11)_2$

$= (0DD7.AC)_{16}$

→ $(ECE.EEE)_{16} = (1110\ 1100\ 1110.1110\ 1110)_2$

→ $(10\ 1110\ 1110\ 1111.1010)_2 = (2EEF.A8)_{16}$

↳ $(200103)_3 =$

$2\times3^{5}+0\times3^{4}+0\times3^{3}+1\times3^{2}+0\times3^{1}+1\times3^{0}$

$243\times2+9+1$

$486+10$

$(496)_{10}$

↳ $(496)_{10} = (\quad)_3$

3	496
3	165 → 1
3	55 → 0
3	18 → 1
3	6 → 0
3	2 → 0
	0 → 2

$(200101)_3$

↳ $(1023)_4 = (\quad)_{10}$

$1\times4^{3}+0\times4^{2}+2\times4^{1}+3\times4^{0}$

$= 64+16+8+3$

$= 64+8+3$

$= (75)_{10}$

↳ $(75)_{10} = (\quad)_4$

4	75
4	18 → 3
4	4 → 2
4	1 → 0
	0 → 1

$= (1023)_4$

* $(142)_5 = (\ )_{10}$

$1\times5^2+4\times5^1+2\times5^0$

$25+20+2$

$(47)_{10}$

* $(47)_{10} = (\ )_5$

5 | 47
5 | 9 - 2
5 | 1 - 4
0 - 1

$(142)_5$

* $(125)_6 = (\ )_{10}$

$1\times6^2+2\times6^1+5\times6^0$

$36+12+5$

$=(53)_{10}$

* $(53)_{10} = (\ )_6$

$=(125)_6$

6 | 53
6 | 8 - 5
6 | 1 - 2
0 - 1

* $(210)_7 = (\ )_{10}$

$2\times7^2+1\times7^1+0\times7^0$

$98+7$

$(105)_{10}$

* $(105)_{10} = (\ )_7$

$(210)_7$

7 | 105
7 | 15 - 0
7 | 2 - 1
0 - 2

* $(143)_5 = (\ )_{10}$

$1\times5^2+4\times5^1+3\times5^0$

$25+20+3$

$(48)_{10}$

* $(48)_{10} = (\ )_5$

$=(143)_5$

~~48~~
5 | 9 - 3
5 | 1 - 4
0 - 1

* $(2010\text{&}0)_3 = (\ )_5$

2×3^5
$+0\times3^4+1\times3^3+0\times3^2+0\times3^1+0\times3^0$

$2\times243+0+27+15+0$

$486+27+15+0$

$= (5\text{&}9)_5$

$2\times3^5+0\times3^4+1\times3^3+0\times3^2+2\times3^1+0\times3^0$

$486+87+6$

579

* $(579)_5 = (\ )_3$

$=(201020)_3$

3 | 579
3 | 193 - 0
3 | 64 - 1
3 | 21 - 1
3 | 7 - 0
3 | 2 - 1
0 - 2

3 | 532
3 | 177 - 1
3 | 59 - 0

counting in other number systems whose bases are 3, 4, 5, 6, 7

4	3	5	6	7
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	10	3	3	3
10	11	4	4	4
11	12	10	5	5
12	20	11	10	6
13	21	12	11	10
20	22	13	12	11
21	100	14	13	12
22	101	20	14	13
23	102	21	15	14
30	110	22	20	15
31	111	23	21	16
32	112	24	22	20
33	120	30	23	21
100	121	31	24	22
101	122	32	25	23
102	200	33	30	24
103	201	34	31	25
110	202	40	32	26
111	210	41	33	30
112	211	42	34	31
113	212	43	35	32
120	220	44	40	33
121	221	100	41	34
123	222	101	42	35
123	1000	102	43	36
130	1001	103	44	40

↳ $(312)_4 = (\ )_8$

$(312)_4 = (\ )_{10}$

$4^2\times3 + 4^1\times1\times4^0(2)$

$= 16\times3 + 4 + 2$

$= 48 + 4 + 2$

$= (54)_{10}$

$(54)_{10} = (\ )_8$

8 | 54
8 | 6 − 6
0 − 6

$= (066)_8$

↳ $(74A)_{16} = (\ )_8$

(0111 0100 1010)$_2$
3 2 1 2

$(3512)_8 = (74A)_{16}$

↳ $(456)_8 = (\ )_{16}$

(100 101 110)$_2$
1 2 3

$= (12E)_{16}$

other method:-

$8^2\times4 + 5\times8^1 + 6\times8^0$

$64\times4 + 40 + 6$

$256 + 40 + 6$

$(302)_{10} = (\ )_{16}$

16 | 302
16 | 18 − 14
16 | 1 − 2
0 − 1

$(12E)_{16}$

↳ $(1021)_3 + (2110)_3 = (\ )_{10}$

$1\times3^3 + 0\times3^2 + 2\times3^1 + 1\times3^0 + 2\times3^3 + 1\times3^2 + 1\times3^1 + 0\times3^0$

$= 27 + 0 + 6 + 1 + 54 + 9 + 3 + 0$

$= (100)_{10}$

↳ $(120)_4 + (312)_4 = (\ )_{10}$

$4^2\times1 + 4^1\times2 + 4^0\times0 + 4^2\times3 + 4^1\times1 + 4^0\times2$

$16 + 8 + 0 + 48 + 4 + 2$

$= (78)_{10}$

↳ $(341)_5 + (202)_5 = (\ )_{10}$

$5^2\times3 + 5^1\times4 + 5^0\times1 + 5^2\times2 + 5^1\times0 + 5^0\times2$

$75 + 20 + 1 + 50 + 0 + 2$

$= (148)_{10}$

↳ $(16)_{10} = (100)_b$

$16 = 1\times b^2 + 0\times b^1 + 0\times b^0$

$16 = b^2$

$\boxed{b=4}$

↳ $(292)_{10} = 1204$

$292 = 1\times b^3 + 2\times b^2 + 0\times b^1 + 4\times b^0$

$292 = b^3 + 2b^2 + 4$ here substitute b minimum is 4 so substitute more than 4.

Sub → b = 5

$292 = 5^3 + 2(5)^2 + 4$

$= 125 + 2(25) + 4$

$= 125 + 50 + 4$

$= 179$ ×

Sub → b = 6

$292 = 6^3 + 2(6)^2 + 4$

$= 216 + 2(36) + 4$

$= 216 + 72 + 4$

$= 292$

$\boxed{b=6}$

↳ $\frac{41}{3} = 13$

$41 = 39$

$4\times b^1 + 1\times b^0 = 3\times b^1 + 9\times b^0$

$4b + 1 = 3b + 9$

$4b - 3b = 9 - 1$

$\boxed{b=8}$

Subtraction of Binary numbers via addition using ones complument method.

* ones compliment of a binary number is obtained by subtracting each digit by one (or) by interchanging zeros to 1 and ones to zeros

↳ * Two's compliment of a binary number is obtained by adding one to its one's compliment.

Ex: 1011 → 1111

1's compliment 0100 — 1's com 0000

2's comp + 1 — 2nd + 1

0101 — 0001

↳ 010110.110

1's comp 101001.001

2's com + 1

101001.010

↳ 0000

1's compl 1111

2nd com + 1

10000

Subtraction.

↳ Calculate the ones compliment of subrahend & Add that one's compliment with the minuent.

↳ If end around carry is generated answer is positive and final answer is obtained by adding that carry to the suming answer.

↳ If end around carry is not generated then answer is negative and final answer is obtained by neglecting the carry and answer is one's complement of summing answer.

↳ $1100 - 1010 = (0010)_2$

↳ $1110 - 11 = (1011)_2$

1010 — 1100

1's comp 0101 → +0101

10001

+1

0010

0011 — 1110

1's com 1100 → (+) 1100

11010

+1

1011

↳ $11011.101 - 10010.001 = (01001100)$

10010.001 — 11011.101

1's comp 01101.110 → +01101.110

100100111

+1

01001100

Binary subtraction via addition by using 2's Compliment

* Calculate the 2's compliment of subprahend.
* Add that 2's compliment to to the minuent
* If Carry is generated answer is positive by neglecting the carry and it is final answer.
* If carry is not generated answer is negative and final answer is obtained by 2's compliment of summing answer.

Ex: $(1100)_2^m - (1010)_2^s = (0010)_2$

sub → 1010 — 1100

1's comp → 0101 — +0110

2's com +1 — 10010

0110 — neglect carry

↳ $(1110)_2 - (11)_2 = (1011)_2$

sub	0011	1110
1's comp	1100	1101
2's co	+1	
	1101	
		neglect carry

↳ $(1010)_2 - (1100)_2 = -(0010)_2$

sub -	1100	1010	
1's com	0011	0100	
	0011	1110	1110
	+1	← 1's complement	0001
	00000		+1
			0000

↳ $(11)_2 - (1110)_2 = (1011)_2$

sub -	1110	0011		0101
1's com	0001	0010	is	1010
2's co	+1	0101		+1
	0010	← no carry so 2's compliment again		1011

↳ $(11011.101)_2 - (10010.001)_2 = (01001.100)_2$

sub -	10010.001	11011.101	
1's	01101.110	01101.111	
2's	+1	11111 11	
	01101.111	101001.100	
		neglecting carry	

↳ $(10010.001)_2 - (11011.101)_2 = -(01001.100)_2$

sub	11011.101	10010.001	
1's	00100.010	00100.010	
2's	+1	1 1	
	00100.011	101 10.100	
		← no carry generated so negative.	
1's compl		01001.011	
2's		+ 1	
		01001.100	

Notes-

n bits in binary number can count from 0 to 2^{n-1} in decimal numbers. $2^4 - 1 = 16 - 1$, 0 to 15

Signed Binary Numbers.

* Sign magnitude representation
* Sign One's complement representation
* Sign two's complement representation.
* left most bit that is MSB is deserve for sign bit.
* plus ('+') is denoted by '0'
* minus (-) is denoted by '1'.

Decimal No Sign magnitude representation.

		sign 1's	sign 0's
1) +2	→	010	
2) -5	→	1101	
3) +12	→	01100	
4) -13	→	11101	

Decim No	sign 1's complement	sign 2's compliment
+2	010	010
-5	1010	1011
+12	01100	01100
-13	10010	10011

Signed Binary Number (+8 4 2 1)	Signed Magnitude (Decimal)	sign 1's complement	sign 2's complement
01101	+13	+13	+13
10110	-6	-9 (-1001)	-10 (-1010)
100101 (-16 8 4 2 1)	-5	-26 (-11010)	-27 (-11011)
10000 (-8 4 2 1)	-0	-15 (-1111)	-16 (-10000) (1111 +1 10000)

Decimal	Sign magnitu	Sign 1's compline	Sign 2's complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
0	0000	0000	0000
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

For an n bidth word which includes the sign width $2^{n-1}-1$ +ve integers and 2^{n-1} negative integers and one "zero" for a total of 2^n states.

Express -45 insign2's complement representation.

Number = -45

Binary (sign 32 16 8 4 2 1) 1 101101 → sign magcomplement

1's complm 1010010

2's comple +1

1010011

Subtract 14 from 46 wing 8 bit 2's complemen.

+46 - 14

+46 = (sign) 0101110 → 8bits = 00101110

-14 = (sign) 11110 ⟶ 8bits 10001110

-14 → 1's complement = 11110001

+1

2's comp 11110010

+46 + (-14) = 00101110

11110010

(1)00100000

ngluting Carry

↳ Subtract 27.50 from 68.75 wing 12 bit 1's complement method.

68.75 - 27.50

-27.50 = (8bits formation) 10011011.1000

1's complement = 11100100.0111

68.7 = 01000100.1100

Add 1's comple & 68.7:

```
  11100100.0111
  01000100.1100
 (1)00101001.0011 → ans
  ↓
 neglecting carry
```

→ Here carry is generated in 1's complement so add 1 to the answer

```
  00101001.0011
              +1
 ------------------
  00101001.0100
```

↳ Write the following Binary Numbers in Sign magnitude form and in sign 1's complement form and 2's complement form using 16 bit register.

(a) +1001010

Sign magnitude → 0000000001001010

Sign 1's comple → 0000000001001010

Sign 2's comple → 0000000001001010

(b) −11110000

Sign magnitu → 1000000011110000

1's complement → 1111111100001111

2's complement → ~~1111111100001111~~

```
  1111111100001111
                 +1
 ------------------
  1111111100010000
```

(c) −1100 1100.1

Sign magnitude → 1000000011001100.10

1's complement → 1111111100110011.11

2's complement →

```
  1111111100110011.11
                    +1
 ---------------------
  1111111100110100.00
```

d) +10000000 11.111

Sign magnitude

→ 0100000011.111000

1's complement → 0100000011.111000

2's complement → 0100000011.111000

Perform N_1+N_2, $N_1+(-N_2)$ for the following 8 bit number expressed in two's complement representation & verify your answer by using decimal addition.

(a) N_1 = 00110010

N_2 = 11111101

↳ N_1 = 00110010 (N_1+N_2)

N_2 = 11111101

```
 (1)00101111
  ↓
 neglect
```

($N_1+(-N_2)$)

N_1 = 00110010

$-N_2$ = 11111101

```
 (1)00100001
  ↓
 neglecting carry
```

(b) $N_1 = 10001110$

$N_2 = 00001101$

$N_1 = 10001110$

$N_2 = 00001101$

$N_1 + N_2$

10011011

$N_1 + (-N_2)$

(a) $N_1 = 10001110$

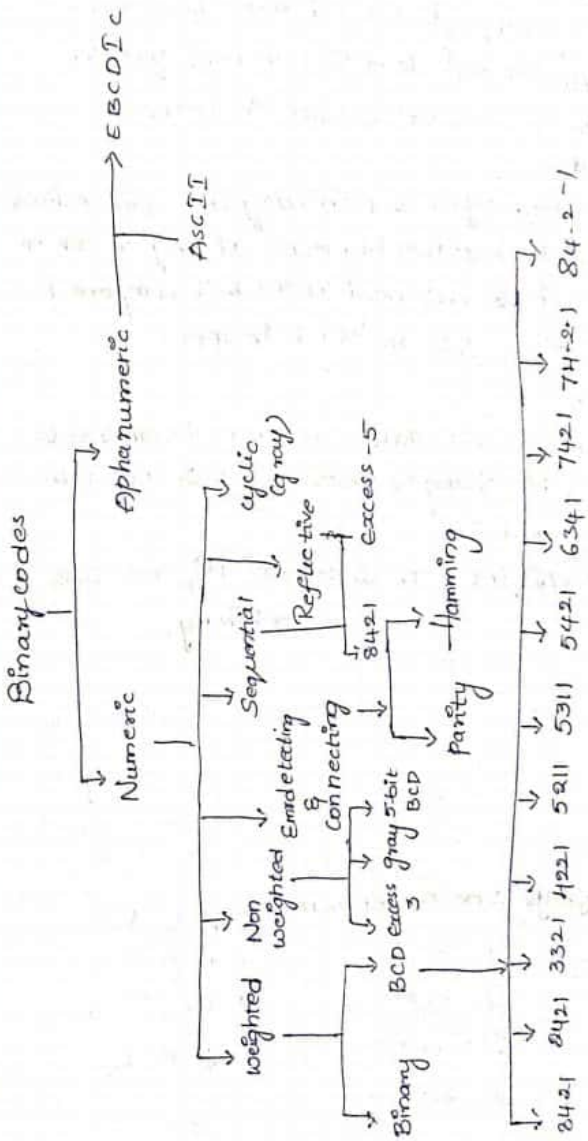
$-N_2 = 10001101$

00010001

neglecting carry

→ $(-6)_{10} - (-13)_{10}$ using 2's complement method.

$-6 =$



BCD - Binary coded decimal :.

Each digit in decimal number is represented by 4 bit binary code.

BCD is very simple but need more bits when compared with binary for decimal numbers greater than 9. 10 to 15 are unused states in BCD.

Excess 3 code:

It is obtained by 4-bit binary code for individual digit in decimal number after adding 3. Excess 3 code also simple but need more bits compared to binary code. 0,1,2 are unused in these.

Gray code :

Gray code is also name as unit distance code as only one bit changes from 0 to 1 or 1 to 0 in two successive numbers.

Convert $(76)_{10}$ into BCD & Convert $(14)_{10}$ into BCD & binary.

Binary to Gray code conversion.

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

$g_3 = b_3$

$g_2 = b_3 \oplus b_2$

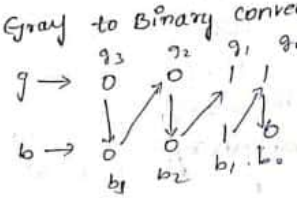
$g_1 = b_2 \oplus b_1$

$g_0 = b_1 \oplus b_0$

b_3	b_2	b_1	b_0
↓	↓	↓	↓
1	0	0	1
1	1	0	1
g_3	g_2	g_1	g_0

Decimal	8421	BCD	Excess - 3	Gray code
0	0000	0000	0011	0000
1	0001	0001	0100	0001
2	0010	0010	0101	0011
3	0011	0011	0110	0010
4	0100	0100	0111	0110
5	0101	0101	1000	0111
6	0110	0110	1001	0101
7	0111	0111	1010	0100
8	1000	1000	1011	1100
9	1001	1001	1100	1101

Gray to Binary conversion.



$b_3 = g_3$

$b_2 = b_3 \oplus g_2$

$b_1 = b_2 \oplus g_1$

$b_0 = b_1 \oplus g_0$

Convert $(146)_{10}$ into Binary BCD, Excess - 3 & Gray

2 | 146

2 | 73 → 0

2 | 36 → 1

2 | 18 → 0

2 | 9 → 0

2 | 4 → 1

2 | 2 → 0

2 | 1 → 0

0 → 1

1 4 6

↓ ↓ ↓

excess 3 → 4 7 9

↓

0100 0111 1001

Binary to gray

b → 1 0 0 1 0 0 1 0

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

g → 1 1 0 1 0 1 1

Binary = 10010010

BCD → (146) → 0001 0100 0110

Reflection of Gray code:

1 bit	2 bit	3 bit	4 bit
0	00	000	0000
1	01	001	0001
	11	011	0011
	10	010	0010
		110	0110
		111	0111
		101	0101
		100	0100
			1100
			1101
			1111 14-1001
			1110 15-1000
			1010
			1011

Error detecting and correcting codes:

BCD	even parity	odd parity
0000	0	1
0001	1	0
0010	1	0
0011	0	1
0100	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1

Block parity:- Even parity

0001110	0
0110001	0
1001110	1
0111110	0
1011101	0
1011010	0
1001110	

In an even parity scheme which of the word contain error.

a. 10111011 → even parity → no error

b. 11011010 → odd parity → error

c. 11111011111 → odd parity → error

d. 11110110111 → even parity → no error

Odd parity

a. 11011011 → error

b. 10110111 → error

c. 11011111 → no error

d. 10101110 → no error

Hamming codes:-

7 bit hamming code:

even parity

data = 1010

$P=0$; $2^P \geq n+p+1$

p → parity bits $2^0 \geq 4+0+1$

$1 \not\geq 4$

n → no. of data bits

$P=1$, $2^1 \geq 4+1+1$

$2 \not\geq 6$

P=3; $2^P \geq 4+3+1$

$4 \neq 7$

P=3 $\quad 2^3 \geq 4+3+1$

$8 \geq 8$

001	010	011	100	101	110	111
1	2	3	4	5	6	7
P_1	P_2	D_3	P_4	D_5	D_6	D_7
0	1	1	0	0	1	1

Parity bit position

$2^0\ 2^1\ 2^2\ 2^3\ 2^4 \ldots 2^P$

1, 2, 4, 8. ---

For P_1 = 1, 3, 5, 7

$P_1\ D_3\ D_5\ D_7$

0 1 0 1

$P_1 \rightarrow 0$

$P_2 \rightarrow$ 2 3, 6, 7

$P_2\ D_3\ D_6\ D_7$

1 1 1 1

$P_2 \rightarrow 1$

$P_4 \rightarrow$ 4, 5, 6, 7

$P_4\ D_5\ D_6\ D_7$

0 0 1 1

$P_4 \rightarrow 0$

0111011

For $C_1 \rightarrow$ 1, 3, 5, 7

$C_1 \rightarrow 0$ 0 1 0 1 → even → no error

For $C_2 \rightarrow$ 2, 3, 6, 7

$C_2 \rightarrow 0$ 1 1 1 1 → no error

000
001
010
011
100
101
110
111

For $C_3 \rightarrow$ 4, 5, 6, 7

$C_3 \rightarrow 1$ 1 0 1 1 → odd error

For error code $C_3\ C_2\ C_1$

1 0 0 = $(4)_{10}$

Answer: 0110011

② even parity

data 1 1 1 1

Parity position

$2^0\ 2^1\ 2^2\ 2^3 \ldots 2^P$

1, 2, 4, 8

001	010	011	100	101	110	111
1	2	3	4	5	6	7
P_1	P_2	D_3	P_4	D_5	D_6	D_7
1	1	1	1	1	1	1

For P_1 = 1, 3, 5, 7

$P_1\ D_3\ D_5\ D_7$

1 1 1 1

$P_1 \rightarrow 1$

P_2 = 2 3 6 7

$P_2\ D_3\ D_6\ D_7$

1 1 1 1

$P_2 = 0$

$P_4 \rightarrow$ 4, 5, 6, 7

$P_4\ D_5\ D_6\ D_7$

1 1 1 1

$P_4 = 1$

At receiver

data is: 1 2 3 4 5 6 7

1 1 1 1 1 0 1

For $C_1 \rightarrow$ 1, 3, 5 7

1 1 1 1

$C_1 \rightarrow 0$ no error

For $C_2 \rightarrow$ 2, 3, 6, 7

1 1 0 1

error

$C_2 \rightarrow 1$

For C_3 4, 5, 6, 7

1 1 0 1

$C_3 = 1$ error

For error code

$C_3\ C_2\ C_1$

1 1 0

= $(6)_{10}$

Answer

1111111

Encode data 0110 into the 7 bit hamming code in even parity.

even parity

data 0110

Parity pos'

1, 2, 4, 8

$P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7$

1 0 1 0 0 1 1 0

$P_1 \rightarrow 1, 3, 5, 7$

1 0 1 0

$P_1 = 0$

$P_2 \rightarrow 2, 3, 6, 7$

1 0 1 0

$P_2 = 1$

$P_4 = 4, 5, 6, 7$

0 1 1 0

$P_4 = 0$

2) The message below coded in the 7 bit hamming code is transmitted through a noisy channel. Decode the message assume that atmost a single error occurred

1) 1 2 3 4 5 6 7

1 0 0 1 0 0 1 → corrected code is 1 1 0 1 0 0 1

$c_1 \rightarrow 1, 3, 5, 7$

1 0 0 1

$c_1 \rightarrow 0$

$c_2 \rightarrow 2, 3, 6, 7$

0 0 0 1

$c_2 \rightarrow 1$

$c_3 \rightarrow 4, 5, 6, 7$

$c_3 \rightarrow 0$ 1 0 0 1 $\quad c_3\ c_2\ c_1 \rightarrow 010 = (2)_{10}$

2) 1 2 3 4 5 6 7

0 1 1 1 0 0 1

$c_1 \rightarrow 1, 3, 5, 7$

0 1 0 1

$c_1 \rightarrow 0$

$c_2 \rightarrow 2, 3, 6, 7$

1 1 0 1

$c_2 \rightarrow 1$

$c_3 \rightarrow 4, 5, 6, 7$

1 0 0 1

$c_3 \rightarrow 0$

$c_3\ c_2\ c_1 \rightarrow 0\ 1\ 0 = (2)_{10}$

corrected code

↳ 0 0 1 1 0 0 1

3) 1 2 3 4 5 6 7

1 1 1 0 1 1 0

$c_1 \rightarrow 1, 3, 5, 7$

1 1 1 0

$c_1 \rightarrow 1$

$c_2 \rightarrow 2, 3, 6, 7$

1 1 1 0

$c_2 \rightarrow 1$

$c_3 \rightarrow 4, 5, 6, 7$

0 1 1 0

$c_3 \rightarrow 0$

$c_3\ c_2\ c_1 \rightarrow 0\ 1\ 1 = (3)_{10}$

corrected code

↳ 1 1 0 0 1 1 0

4) 0 0 1 1 0 1 1

$c_1 \rightarrow 1, 3, 5, 7$

0 1 0 1

$c_1 \rightarrow 0$

$c_2 \rightarrow 2, 3, 6, 7$

0 1 1 1

$c_2 \rightarrow 1$

$c_3 \rightarrow 4, 5, 6, 7$

1 0 1 1

$c_3 \rightarrow 1$

$c_3\ c_2\ c_1 \rightarrow 1\ 1\ 0 = (6)_{10}$

corrected code

↳ 0 0 1 1 0 0 1

12-bit hamming code

Parity pos:

$2^0\ 2^1\ 2^2\ 2^4$

1 2 3 4 5 6 7 8 9 10 11 12

$P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7\ P_8\ D_9\ D_{10}\ D_{11}\ D_{12}$

$P_1 \rightarrow 1,3,5,7,9,11$

$P_2 \rightarrow 2,3,6,7,10,11$

$P_4 \rightarrow 4,5,6,7,12$

$P_8 \rightarrow 8,9,10,11,12$

0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100

Alpha-Numeric codes

	000	001	010	011	100	101	110	111
0000	NUL	DEL	Space	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC_2	"	2	B	R	b	r
0011	ETX	DC_3	#	3	C	S	c	s
0100	EOT	DC_4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

ASCII → American standard code for information interchange. 7bit → $2^7 = 128$

EBCDIC → Extended Binary coded decimal interchange code.

8-bit code

$2^8 = 256$

Logic Gates:

Basic gates → AND, OR, NOT

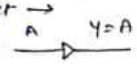
Universal Gates → NAND, NOR

Special Gates → Ex-OR, Ex-NOR.

↳ Inputs and outputs for any logic gates are alldays either zero 0 or 1 which represents on or off, True or false, close or open respectively. Hence Variables can also have one or zero, +ve or -ve, High or low.

Buffer →

Symbol:

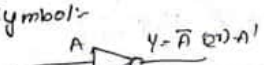


Truth table

I/p	O/p
A	Y
0	0
1	1

Not gate (Converter)

Symbol:-

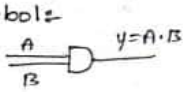


T.t

I/p	O/p
A	Y
0	1
1	0

~~And~~ gate

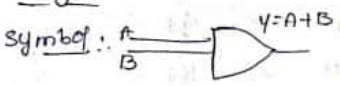
Symbol:-



I/p		O/p
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate

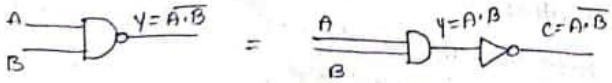
Symbol:-



Truth table

I/p		O/p
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

NAND Gate:



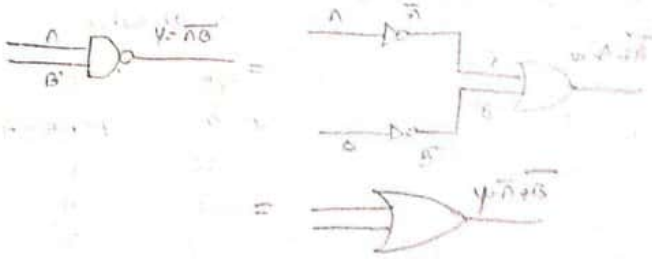
NAND Truth table

A	B	$Y=\overline{A\cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

By looking at the truth table of NAND gate output Y is 1 when A=0 or B=0 or when both A and B are equal to 0. If either $\overline{A}=1$ or $\overline{B}=1$ or Both $\overline{A}$ and $\overline{B}$ are equal to 1. Therefore NAND gate can perform the OR function.

Bubbled or gate = NAND gate

I/p		O/p	I/p		O/p
A	B	$Y=\overline{A\cdot B}$	$\overline{A}$	$\overline{B}$	$Y=\overline{A}+\overline{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0



NOR Gate:

Symbol:

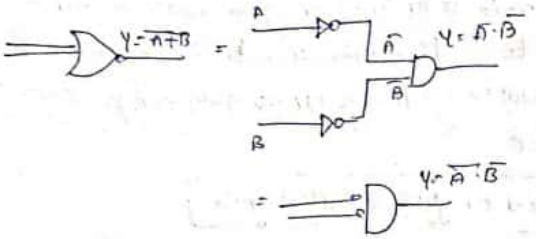
Truth table:-

I/P A	I/P B	O/P $Y=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

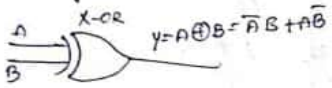
I/P A	I/P B	O/P $Y=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

I/P $\overline{A}$	I/P $\overline{B}$	O/P $Y=\overline{A}\cdot\overline{B}$
1	1	1
1	0	0
0	1	0
0	0	0

NOR = Bubbled AND gate



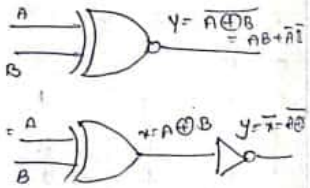
Ex-OR gates.



Truth table

I/P A	I/P B	O/P $Y=\overline{A}B+A\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0

Ex-NOR gates

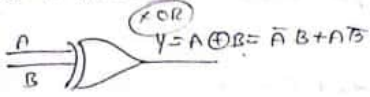


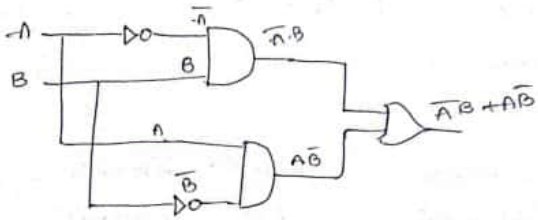
Truth table

I/P A	I/P B	O/P $Y=AB+\overline{A}\overline{B}$
0	0	1
0	1	0
1	0	0
1	1	1

Draw ex OR and ex NOR by using AOI gates.

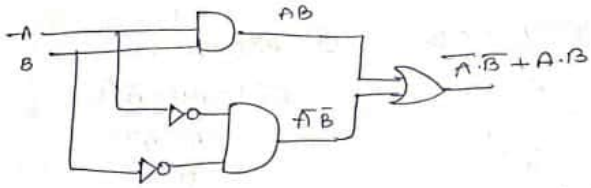
AOI → And, OR, Invert.





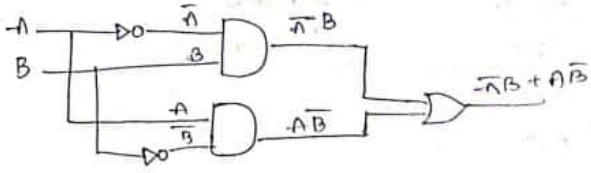
X-NOR

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



* Show that $A\oplus B=\overline{A}B+A\overline{B}$ and construct the corresponding logic diagrams.

A	B	$A\oplus B$	$\overline{A}$	$\overline{B}$	$\overline{A}B$	$A\overline{B}$	$\overline{A\overline{B}+A\overline{B}}$
0	0	0	1	1	0	0	0
0	1	1	1	0	1	0	1
1	0	1	0	1	0	1	1
1	1	0	0	0	0	0	0



* Show that $(A+B)\overline{AB} = A \oplus B$

* Find the logical equivalent of the following expression.

(a) $A \oplus 0 = A$

(b) $A \oplus 1 = \overline{A}$

(c) $A \odot 0 = \overline{A}$

(d) $A \odot 1 = A$

e. $1 \oplus A = \overline{A}$

f. $0 \oplus \overline{A} = \overline{A}$

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

$A \oplus 0 = \overline{A}0 + A\cdot\overline{0}$

$= 0 + A\cdot 1$

$= A$

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

$A \oplus 1 = \overline{A}\cdot 1 + A\cdot\overline{1}$

$= \overline{A} + A\cdot 0$

$= \overline{A} + 0$

$= \overline{A}$

(c) $A \odot B = AB + \overline{A}\,\overline{B}$

$A \odot 0 = A\cdot 0 + \overline{A}\cdot\overline{0}$

$= 0 + \overline{A}$

$= \overline{A}$

(d) $A \odot B = AB + \overline{A}\,\overline{B}$

$A \odot 1 = A\cdot 1 + \overline{A}\cdot\overline{1}$

$= A + \overline{A}\cdot 0$

$= A$

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

$1 \oplus A = \overline{1}\cdot A + \overline{1}\cdot\overline{A}$

$= 0\cdot A + \overline{A}$

$= \overline{A}$

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

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

$0 \oplus \overline{A} = \overline{0}\,\overline{A} + 0\cdot\overline{\overline{A}}$

$= 1\cdot\overline{A} + A\cdot 0$

$= \overline{A}$

→ Determine which of the following exp are equivalent to $A \oplus B$ & $A \odot B$.

a) $\overline{A} \oplus B$

b) $\overline{A} \odot B$

c) $\overline{A} \oplus \overline{B}$

d) $\overline{A} \odot \overline{B}$

e) $A \oplus \overline{B}$

f) $A \odot \overline{B}$

(a) → $A \oplus B = \overline{A}B + A\overline{B}$

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

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

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

(b) $A \odot B = \overline{A}\,\overline{B} + A\overline{B}$

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

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

(c) $\overline{A} \oplus \overline{B} = \overline{\overline{A}}\,\overline{B} + \overline{A}\,\overline{\overline{B}}$

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

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

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

$= A \oplus B$

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

(e) $A \oplus \overline{B} = A\overline{\overline{B}} + \overline{A}\,\overline{B}$

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

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

(f) $A \odot B = A\overline{B} + \overline{A}\,\overline{\overline{B}}$

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

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

Boolean Algebra.

Axioms (or) postulates of Boolean Algebra.

It is defined as set of logical expressions that we accept without proof and upon which we can build a set of useful rules those are called Axioms (or) postulates of Boolean Algebra.

→ Application as to reduce large Boolean expression into simple expression (or) small expression that has less number of terms (literals) hence less hardware is required to implement

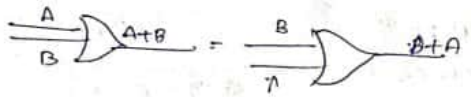
AND laws	OR laws	Complementary law
$A\cdot 0 = 0$ → Null law	$A + 0 = A$ → Null law	If $A=0$, then $\overline{A}=1$
$A\cdot 1 = A$ → Identity law	$A + 1 = 1$ Identity law	If $A=1$, then $\overline{A}=0$ → complementary law
$A\cdot\overline{A} = 0$ → Null law	$A + \overline{A} = 1$	$\overline{\overline{A}} = A$ → double complementary law
$A\cdot A = A$ → Idempotence law	$A + A = A$ → Idempotent law	

Commutative laws:-

Law 1. $A+B = B+A$

Proof:-

A	B	A+B	B+A
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1



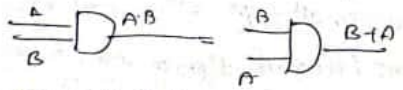
If 3 variables

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

Law 2:- $A \cdot B = B \cdot A$

Proof:-

A	B	A·B	B·A
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1



If three variables

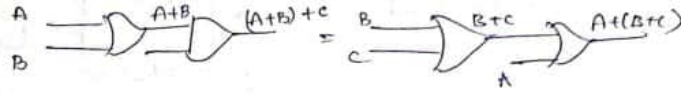
$A \cdot B \cdot C = B \cdot A \cdot C = C \cdot A \cdot B$

Associative laws:-

Law 1: $(A+B)+C = A+(B+C)$

Proof:-

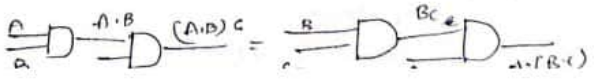
A	B	C	(A+B)	(A+B)+C	(B+C)	A+(B+C)
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1



Law 2:- $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

Proof:-

A	B	C	A·B	(A·B)·C	B·C	A·(B·C)
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

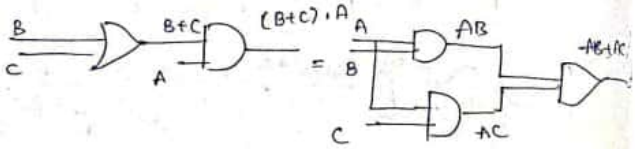


Distributive law:

Law 1:- $A \cdot (B+C) = AB + AC$.

Proof:-

A	B	C	B+C	A·(B+C)	AB	AC	AB+AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1



Law 2: $A + BC = (A+B) \cdot (A+C)$

Proof:- R.H.S $(A+B) \cdot (A+C)$

$= A \cdot A + AC + BA + BC$

$= A + AC + AB + BC$

$= A(1+B+C) + BC$

$= A(1) + BC$

$= A + BC = L.H.S$

Redundant law:

Law 1: $A + \bar{A}B = A + B$

Law 2: $A \cdot (\bar{A} + B) = AB$

Law 1. proof:- $A + \bar{A}B = L.H.S$

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

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

Law 2 proof: $A \cdot (\bar{A} + B) = AB$

$L.H.S = A \cdot (\bar{A} + B)$

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

$= 0 + AB$

$= AB$

$= RHS$

Absorption law:-

Law 1: $A + AB = A$

proof:- $A + AB = L.H.S$

$= A(1+B)$

$= A(1)$

$= A = RHS$

(A+A·Any variable = A)

Law 2: $A \cdot (A+B) = A$

Proof:- $L.H.S = A \cdot (A+B)$

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

$= A + AB$

$= A(1+B)$

$= A \cdot 1 = A = RHS$.

Consensus theorem:

Law 1: $AB + \bar{A}C + BC = AB + \bar{A}C$

Proof:- $L.H.S = AB + \bar{A}C + BC$

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

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

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

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

$= AB + \bar{A}C$

$= R.H.S$

Law 2: $(A+B)\cdot(\bar{A}+C)\cdot(B+C) = (A+B)\cdot(\bar{A}+C)$

L.H.S $= (A+B)\cdot(\bar{A}+C)\cdot(B+C)$

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

$= (0+AC+\bar{A}B+BC)\cdot(B+C)$

$= ABC+\bar{A}B\cdot B+BC\cdot B+AC\cdot C+\bar{A}B\cdot C+BC\cdot C$

$= ABC+\bar{A}B+BC+AC+\bar{A}BC+BC$

$= \bar{A}B(1+C)+AC(1+B)+BC$

$= \bar{A}B\cdot 1+AC\cdot 1+BC$

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

$=$ L.H.S

R.H.S $= (A+B)\cdot(\bar{A}+C)$

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

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

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

$=$ R.H.S

Transposition theorem.

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

R.H.S $= A\bar{A}+AB+\bar{A}C+BC$

$= 0+AB+\bar{A}C+BC(A+\bar{A})$

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

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

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

$= AB+\bar{A}C$

$=$ L.H.S

De-morgan's theorem

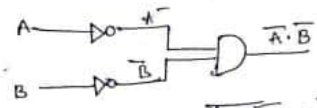
law 1: $\overline{A+B} = \bar{A}\cdot\bar{B}$

This law states that the complement of sum of variables is equal to the product of their individual compliments.

Truth table:

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

Nor gate = Bubbled AND Gate



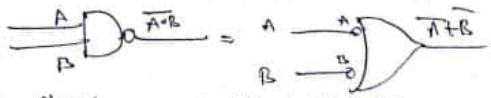
$\overline{A+B+C} = \bar{A}\cdot\bar{B}\cdot\bar{C}$ ---

law 2: $\overline{A\cdot B} = \bar{A}+\bar{B}$

This law states that complement of product of variable is equal to sum of their individual complements.

Truth table:

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



Nand Gate = Bubbled OR Gate

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

Duality theorem:

Dual of any logic expression can be obtained by changing AND operation to OR operation and OR operation to AND operation

↳ 0's → 1's, 1's → 0's without touching any variable.

Expression	Dual
1) $\overline{0} = 1$	$\overline{1} = 0$
2) $0 \cdot 1 = 0$	$1 + 0 = 1$
3) $0 \cdot 0 = 0$	$1 + 1 = 1$
4) $1 \cdot 1 = 1$	$0 + 0 = 0$
5) $A \cdot 0 = 0$	$A + 1 = 1$
6) $A \cdot 1 = A$	$A + 0 = A$
7) $A \cdot A = A$	$A + A = A$
8) $A \cdot \overline{A} = 0$	$A + \overline{A} = 1$
9) $A \cdot B = B \cdot A$	$A + B = B + A$
10) $A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$
11) $A(B+C) = AB + AC$	$A + BC = (A+B)(A+C)$
12) $A \cdot (A+B) = A$	$A + AB = A$
13) $A \cdot (\overline{A}B) = A \cdot B$	$A + (\overline{A}B) = A + B$
14) $\overline{AB} = \overline{A} + \overline{B}$	$\overline{A+B} = \overline{A} \cdot \overline{B}$

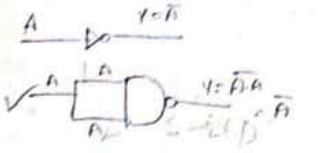
15) $(A+B)(\overline{A}+C)(B+C) = (A+B)(\overline{A}+C) \Rightarrow (A \cdot B) + (\overline{A} \cdot C) + (B \cdot C)$
$= (A \cdot B) + (\overline{A} \cdot C)$

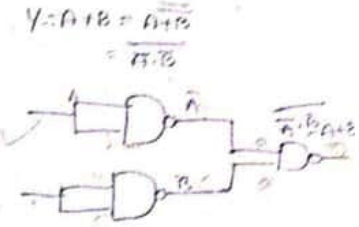
16) $(A+C)(\overline{A}+B) = AB + \overline{A}C \rightarrow (A \cdot C) + (\overline{A} \cdot B) = (A+B) \cdot (\overline{A}+C)$

UNIVERSAL gates

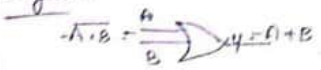
NAND and NOR are universal gates we can implement or realise any other gate output using NAND or NOR gate only.

Implementation or realisation of gates by using NAND gate as NOT.

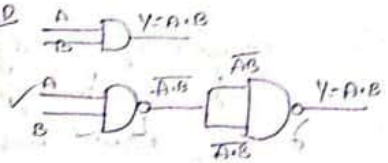




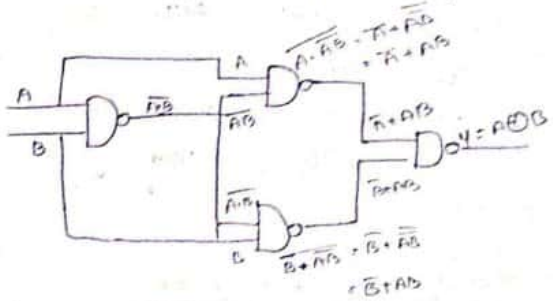
OR gate:



AND



Implementation of EX-OR gate using NAND gate.



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

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

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

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

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

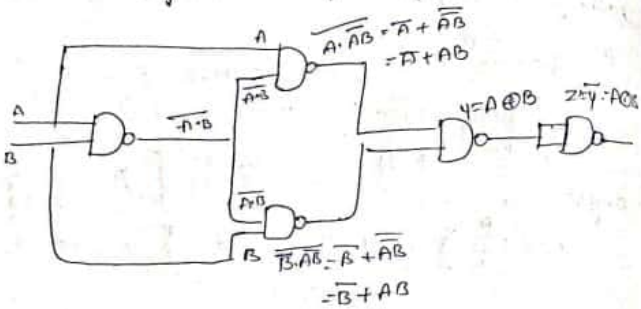
$= A\bar{A} + A\bar{B} + B\bar{A} + B\bar{B}$

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

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

$= A \oplus B$

Implementation of EX-NOR gate using NAND.



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

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

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

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

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

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

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

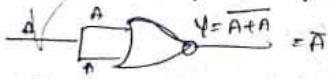
$= A\bar{B} + \bar{A}B = A \oplus B$

$Z = \bar{Y} = \overline{A \oplus B} = A \odot B$

Minimum No. of universal gate required for other gates

	NAND	NOR
NOT	1	1
AND	2	3
OR	3	2
X-OR	4	5
X-NOR	5	4

Implementation of NOT gate using NOR.



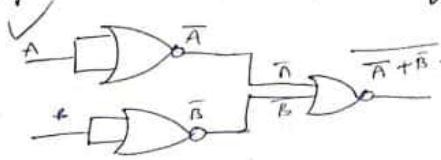
Implementation of OR by using NOR

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

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

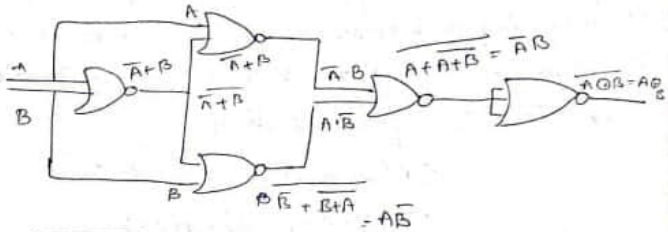
$= A+B$

Implementation of AND gate using NOR Gate



Realize output of EX-OR and EX-NOT Gate using universal gates:

EX-OR gate using NOR:- $Y = A \oplus B = \bar{A}B + A\bar{B}$



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

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

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

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

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

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

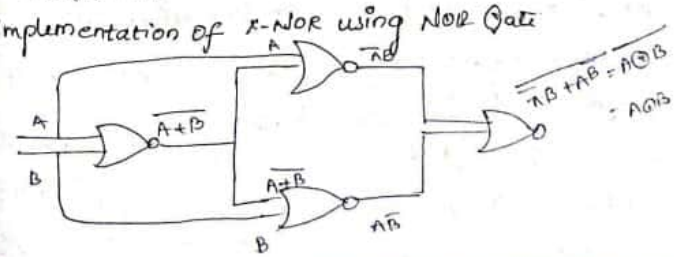
$= \bar{B} \cdot \overline{\overline{A+B}}$

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

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

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

Implementation of X-NOR using NOR Gate



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

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

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

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

$= 0 + \overline{A} B$

$= \overline{A} B$

↳ $\overline{B + \overline{A+B}}$

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

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

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

$= A\overline{B} + 0$

$= A\overline{B}$

Reduction of Boolean expression:

This reduction helps to reduce large Boolean expression into simple boolean expression so that it requires less hardware to implement and cost is also less.

↳ $f(A,B,C) = A[B + \overline{C}(\overline{AB + A\overline{C}})]$

$= A[B + \overline{C}[\overline{AB} \cdot \overline{A\overline{C}}]$ $[A+B = \overline{A} \cdot \overline{B}]$

$= A[B + \overline{C}(\overline{A}+\overline{B}) \cdot (\overline{A}+C)]$

$= A[B + \overline{C}[\overline{A} + \overline{A}C + A\overline{B} + \overline{B}C]]$

$= A[B + \overline{A}\overline{C} + \underbrace{\overline{A}C \cdot \overline{C}}_{0} + \overline{A}\overline{B}\overline{C} + \underbrace{\overline{B}C \cdot \overline{C}}_{0}]$

$= A[B + \overline{A}\overline{C} + \overline{A}B\overline{C}]$

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

$= AB + 0 + 0$

$= AB$

2. $A + B[AC + (B+\overline{C})D]$

$A + B[AC + (B+\overline{C})D]$

$= A + B[AC + BD + \overline{C}D]$

$= A + [ABC + BBD + B\overline{C}D]$

$= A + ABC + BD + B\overline{C}D]$

$= A[1 + BC] + BD[1 + \overline{C}]$

$= A[1] + BD[1]$

$= A + BD$

3) $\overline{A + \overline{BC}}(A\overline{B} + ABC)$

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

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

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

$= \overline{A}A\overline{B}B + A\overline{A}\overline{B}C + \overline{A}AB\overline{B}C + A\overline{A}BCC$

$= 0 + 0 + 0 \cdot C0$

$= 0$

4) $AB + A\overline{B}C + BC + AC + B\overline{C} = f(A,B,C)$ → show that

$AB + A\overline{B}C + BC$

$= A(B + BC) + BC$

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

$= A(1 \cdot (B+C)) + BC$

$= AB + AC + BC$

$= AB(1+\overline{C}) + AC + B\overline{C}$

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

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

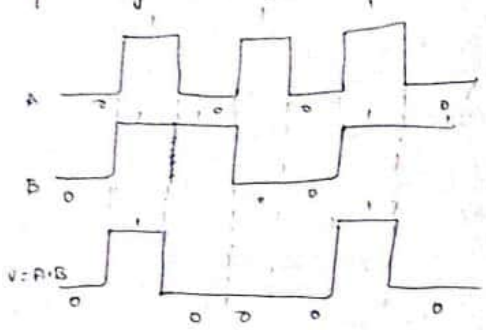
$= AC(1) + B\overline{C}(1)$

$= AC + B\overline{C}$

5) $f = (B+BC)(B+\overline{B}C)(B+D) = B$

6) show that $A\overline{B}C + B + BD + AB\overline{D} + \overline{A}C = B + C$

→ pulsed operations of logic gates.



Simplifying the following Boolean expression

1) $A'c' + ABC + Ac'$ into 3 literates

2) $(x'y'+z)' + z + xy + Wz$ to 3 literals

3) $A'B(B'+c'D) + B(A+A'CD)$ to 1 literal

4) $(A+c)(A'+c')(A+B+c'D)$ to 4 literals

i) $A'c' + ABC + Ac'$

$= \overline{A}\overline{c} + ABC + A\overline{c}$

$= \overline{c}(A+\overline{A}) + ABC$

$= \overline{c} + ABC$

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

$= (\overline{c}+A)\cdot(\overline{c}+B)\cdot 1$

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

$= \overline{c}(1+A+B) + AB$

$= \overline{c} + AB$

ii) $(x'y'+z)' + z + xy + Wz$

$= \overline{(\overline{x}\overline{y}+z)} + z + xy + Wz$

$= (\overline{\overline{x}\overline{y}} \cdot \overline{z}) + z + xy + Wz$

$= (xy+y)\cdot z + z + xy + Wz$

$= xz + yz + z + xy + Wz$

$= z(x+1$

Demorganise $f = \overline{(A+B)(c+D)}$

ii) $f = \overline{AB(cD+\overline{E}F)(A\overline{B}+\overline{c}D)}$

iii) $\overline{ABc + \overline{D}E + B\overline{c}E}$ iv) $ABc + \overline{D}\cdot E + B\overline{c}E$

① $\overline{(A+\overline{B})(c+\overline{D})}$

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

$= \overline{A}\cdot\overline{\overline{B}} + \overline{c}\cdot\overline{\overline{D}}$

$= \overline{A}B + \overline{c}\cdot D$

② $f = \overline{\overline{A}B(cD+\overline{E}F)(A\overline{B}+\overline{c}D)}$

$= (\overline{\overline{A}B}) + (\overline{\overline{c}D}\cdot\overline{\overline{E}F}) + (\overline{\overline{A}\overline{B}+\overline{\overline{c}}D})$

$= \overline{\overline{A}} + \overline{\overline{B}} + \overline{c} + \overline{D}\cdot E\overline{F} + \overline{A} + B\cdot c + \overline{D}$

$= A + B + \overline{c} + \overline{D}\cdot\overline{E}\,\overline{F} + \overline{A} + B\cdot c + \overline{D}$

③ $\overline{ABc + \overline{D}F + B\overline{c}E}$

$= (\overline{A}\,\overline{B}\,\overline{c})\cdot(\overline{\overline{D}}\,\overline{E})\cdot(\overline{B}\,\overline{\overline{c}}\,\overline{E})$

$= (\overline{A}+\overline{B}+\overline{c})\cdot(D+\overline{E})\cdot(\overline{B}+c+\overline{E})$

④ $ABc + \overline{D}\cdot E + B\overline{c}E$

$(A+B+c)\cdot(\overline{D}+E)\cdot(B+\overline{c}+E)$

prove the Dual of X-OR is its complument

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

Dual $\rightarrow (\overline{A}+B)\cdot(A+\overline{B})$

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

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

Complement of $\overline{A}B + A\overline{B} \rightarrow \overline{\overline{A}B + A\overline{B}}$

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

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

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

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

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

Hence dual of X-OR is its complement.

Find the dual of complement of $ABC + \bar{D}E + B\bar{C}\bar{E}$

$Y = ABC + \bar{D}E + B\bar{C}\bar{E}$

Dual → $\overline{(A+B+C)\cdot(\bar{D}+E)\cdot(B+\bar{C}+\bar{E})}$

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

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

dual of complement $= \bar{A}\bar{B}\bar{C} + D\bar{E} + \bar{B}CE$.

→ True the identity of the following equations

① $\bar{A}\bar{B} + \bar{A}B + AB = \bar{A} + B$

↳ $\bar{A}\bar{B} + \bar{A}B + AB$

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

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

$= \bar{A} + AB$

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

$= 1\cdot(\bar{A}+B)$

$= \bar{A} + B$

② $\bar{A}B + B\bar{C} + AB + \bar{B}C = 1$

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

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

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

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

$= (B+\bar{B})(1)$

$= 1$

Representation of Boolean expression:

A function of n Boolean variables denoted by $f(x_1, x_2, \ldots x_n)$ is a function of boolean algebra & its takes one of the two possible values 0 & 1.

Two ways are representing a given boolean function

i) Sum of products (sop)

This forms is also called as disjunctive normal form each product terms contains all variable of the function either in complemented or uncomplemented form.

Ex: $f(ABC) = \bar{A}B + \bar{B}C$

Product of Sum (pos)

This form is also called as conjunctive Normal form

Ex:- $f(A,B,C) = (\bar{A}+B)\cdot(\bar{B}+C)$

A B C	$\bar{A}B + \bar{B}C$	$\bar{A}B + \bar{B}C$
0 0 0	0	1.0 + 1.0
0 0 1	1	0+0 = 0
0 1 0	1	
0 1 1	1	
1 0 0	0	
1 0 1	1	
1 1 0	0	
1 1 1	0	

Sop = 0,01 + 010 + 011 + 101

$= \bar{A}\cdot\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC$

$= m_1 + m_2 + m_3 + m_5$

$= \Sigma m(1,2,3,5)$

Pos form, $f(A,B,C) = (A+B+C)\cdot(\bar{A}+B+C)\cdot(\bar{A}+\bar{B}+C)(\bar{A}+\bar{B}+\bar{C})$

$= M_0 M_4 M_6 M_7$

$= \Pi M(0,4,6,7)$

* each product term is called min term & can be obtained from truth table form which the function values is '1' (if value is 1).

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

	A	B	$\bar{A}B + A\bar{B}$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Sop $= m_1 + m_2$

$= \Sigma m(1,2)$

Pos form $(A+B)\cdot(\bar{A}+\bar{B})$

$= M_0 M_3 = \Pi M(0,3)$

* Min terms are denoted by $M_0, M_1, M_2 \ldots M_n$

* for a 3 variable function $f(A,B,C)$ minterms are

$M_0 = \bar{A}\cdot\bar{B}\cdot\bar{C}$ $\quad M_4 = A\bar{B}\bar{C}$

$M_1 = \bar{A}\bar{B}\cdot C$ $\quad M_5 = A\bar{B}C$

$M_2 = \bar{A}B\bar{C}$ $\quad M_6 = AB\bar{C}$

$M_3 = \bar{A}BC$ $\quad M_7 = ABC$

Standard sop form:

To get standard sop form from sop each product term is multiplied with sum of variable & its compliment which is not present in that term

Eg

find out standard sop for $f = \bar{A}B + \bar{B}C$.

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

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

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

* $f = \bar{A}BC + AB$

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

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

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

* $f = \bar{A}BC + AB + BC$

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

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

$= \bar{A}BC + ABC + AB\bar{C} + BA + B\bar{A}$

It is also called conjunctive canonical form (CCF) (or) expanded pos (or) canonical pos.

→ Each sum term contains all the variables of the function either in complimented or Uncomplimented form

* Each sum term is called maxterm and can be obtained from truth table for which function value is zero (f = 0)
* Max terms are denoted by $M_0, M_1, M_2 \ldots$.
* for 3 variable function f(a,b,c) Max terms are

$M_0 = A+B+C$

$M_1 = A+B+\bar{C}$

$M_2 = A+\bar{B}+C$

$M_3 = A+\bar{B}+\bar{C}$

$M_4 = \bar{A}+B+C$

$M_5 = \bar{A}+B+\bar{C}$

$M_6 = \bar{A}+\bar{B}+C$

$M_7 = \bar{A}+\bar{B}+\bar{C}$

$f(A,B,C) = M_0, M_2, M_6, M_7$

$= \Pi(M)(0,2,6,7)$

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

To get standard pos from Normal pos each sum term is added with product of variable and it compliment which is not present in that term.

* find the standard pos for $f = (\bar{A}+\bar{B}).(B+C)$

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

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

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

So this is standard pos form.

f(A, B, C, D)

Decimal NO	min terms	max terms
0	$\bar{A}\bar{B}\bar{C}\bar{D}$	$A+B+C+D$
1	$\bar{A}\bar{B}\bar{C}D$	$A+B+C+\bar{D}$
2	$\bar{A}\bar{B}C\bar{D}$	$A+B+\bar{C}+D$
3	$\bar{A}\bar{B}CD$	$A+B+\bar{C}+\bar{D}$
4	$\bar{A}B\bar{C}\bar{D}$	$A+\bar{B}+C+D$
5	$\bar{A}B\bar{C}D$	$A+\bar{B}+C+\bar{D}$
6	$\bar{A}BC\bar{D}$	$A+\bar{B}+\bar{C}+D$
7	$\bar{A}BCD$	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$
8	$A\bar{B}\bar{C}\bar{D}$	$\bar{A}+B+C+D$

9	$A\bar{B}\bar{C}D$	$\bar{A}+B+C+\bar{D}$
10	$A\bar{B}C\bar{D}$	$\bar{A}+B+\bar{C}+D$
11	$A\bar{B}CD$	$\bar{A}+B+\bar{C}+\bar{D}$
12	$AB\bar{C}\bar{D}$	$\bar{A}+\bar{B}+C+D$
13	$AB\bar{C}D$	$\bar{A}+\bar{B}+C+\bar{D}$
14	$ABC\bar{D}$	$\bar{A}+\bar{B}+\bar{C}+D$
15	$ABCD$	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$

Expand $\bar{A}+\bar{B}$ into minterms and max terms

↳ $\bar{A}+\bar{B}$ ↳ 1

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

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

$= 01+00+10+00$

$= m_1+m_0+m_2$

$= \Sigma m(0,1,2)$

Here m_3 is missing Max term $= M_3$

Min term $= m_0, m_1, m_2$

↳ $B+A$

$B+A$

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

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

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

$= 11+01+10$

$= m_3+m_1+m_2$

$= \Sigma m(1,2,3)$

Min terms m_1, m_2, m_3

Max terms M_0

↳ Expand $A(\bar{B}+A).B$ into max terms and min term

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

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

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

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

$= (0+0)(0+1)\cdot(1+0)$

$= M_0\cdot M_1\cdot M_2$

$= \pi M(0,1,2)$

Max terms are $0,1,2 = M_0, M_1, M_2$

Min terms are M_3

↳ $f = (AB+BC+AC)$

$= AB+BC+AC$

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

$= ABC+AB\bar{C}+BCA+\bar{A}BC+ABC+A\bar{B}C$

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

$= 111+110+101+011$

$= M_7+M_6+M_5+M_3$

$= \Sigma m(3,5,6,7)$

Min terms are M_3, M_5, M_6, M_7 (or) $\Sigma m(3,5,6,7)$

Max terms are M_0, M_1, M_2, M_4 (or) $\pi m(0,1,2,4)$

↳ expand $A+B\bar{C}+AB\bar{D}+ABCD$

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

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

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

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

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

$A\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + (AB\bar{C} + \bar{A}B\bar{C})(D+\bar{D})' + ABC\bar{D} + \bar{A}BC\bar{D}$

$+ \bar{A}BCD$

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

$+ A\bar{B}\bar{C}\bar{D} + AB\bar{C}D + AB\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}BCD$

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

$= ABCD + ABC\bar{D} + AB\bar{C}D + \bar{A}BCD + AB\bar{C}\bar{D} + A\bar{B}C\bar{D} +$

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

$= 1111 + 1110 + 1101 + 1011 + 1100 + 1010 + 1001 +$

$1000 + 0101 + 0101 + 0100$

$= m_{15} + m_{14} + m_{13} + m_{11} + m_{12} + m_{10} + m_9 + m_8 + m_5$

$+ m_5 + m_4$

Min term $= \Sigma m(4,5,8,9,10,11,12,13,14,15)$

Max term $= \Pi M(0,1,2,3,6,7)$

↳ $AB + BC + AC$

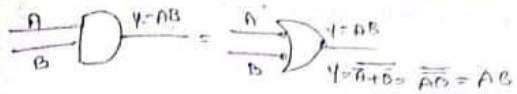
$ABX + XBC + AXC$

$110 + 011 \quad 101$

$111 +$

Alternative gate representation:

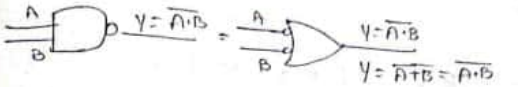
Alternative symbols are equivalent to standart symbols and their equivalence can be proved by using demorgeous law. standard symbols are AND, OR, INVERTOR, NAND. do get alternative symbols we can replace OR to AND, AND to OR, Active Low to Active high

1) 

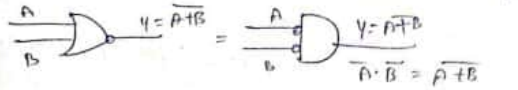
AND = Bubbled NOR gate

2) 

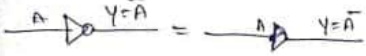
OR = Bubbled NAND gate

3) 

NAND = Bubbled OR gate

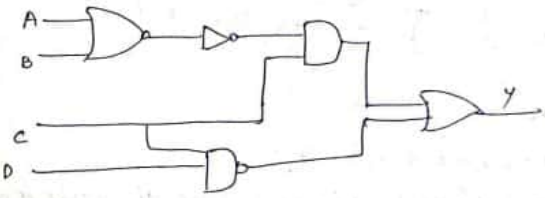
4) 

NOR = Bubbled AND gate

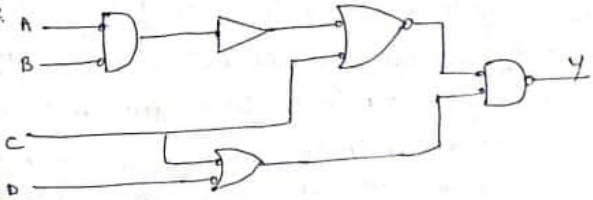
5) 

NOT = Bubbled Buffer

Draw the circuits using alternative symbols

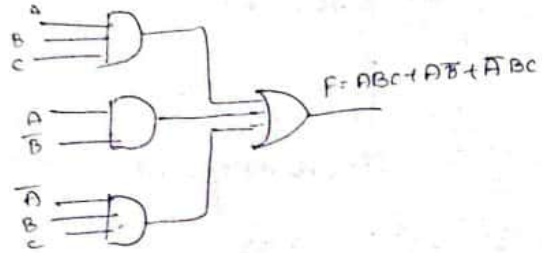


Ans:

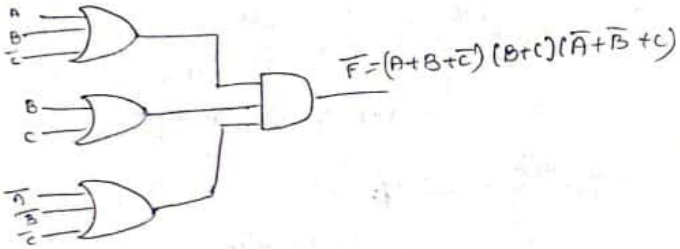


↳ Converting AOI into NAND (or) NOR logic

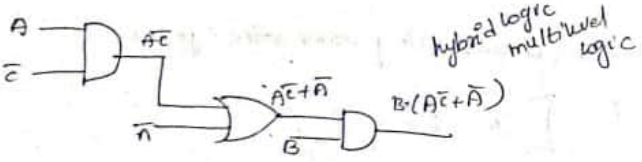
i) $F = ABC + A\bar{B} + \bar{A}BC$ → Sop form



ii) $\bar{F} = (A+B+\bar{C})(B+C)(\bar{A}+\bar{B}+C)$ → posform



↳



hybrid logic
multilevel logic

Procedure

↳ Draw the circuit in AOI logic

↳ If NAND is choosen add a circle at the output of each AND gate and inputs to all OR gate

↳ If NOR logic is choosen add a circle at the output of each OR gate and inputs to all AND gate.

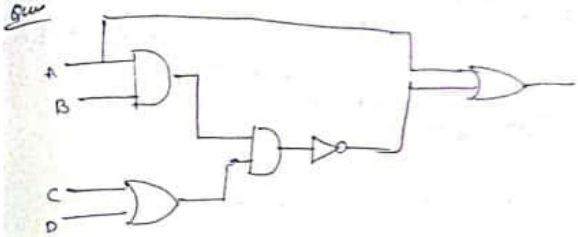
↳ Add or subtract an inverter on each line that recived a circle at step 2 (or) 3. So that polarity of signals on these lines remaining unchanged with the original diagram.

↳ Replace bubbled OR by NAND and Bubbled AND by NOR.

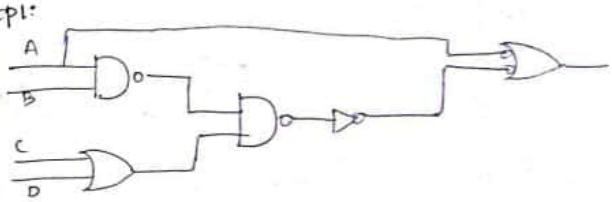
↳ Eliminate double inversions.

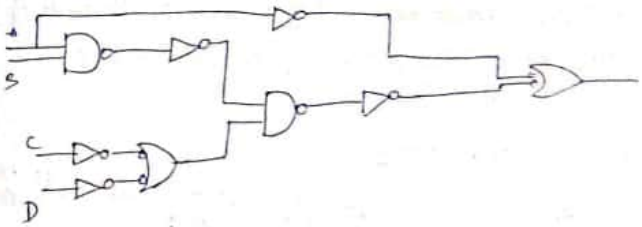
↳ convert the following AOI circuit to NAND logic and NOR logic.

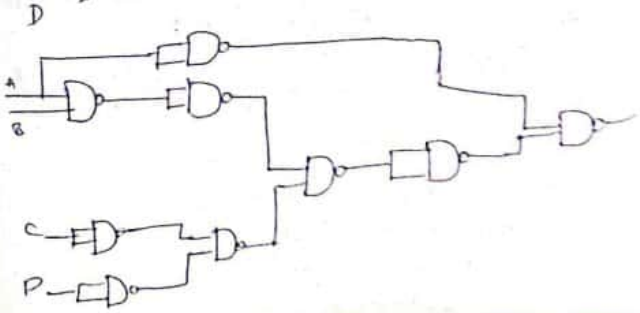
Sol



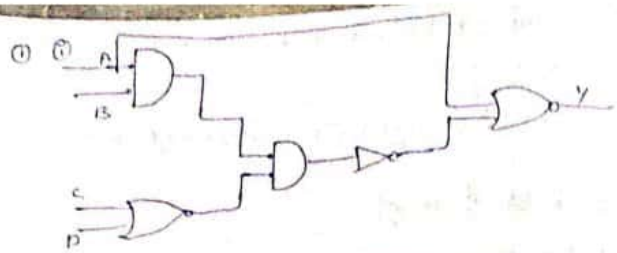
step1:



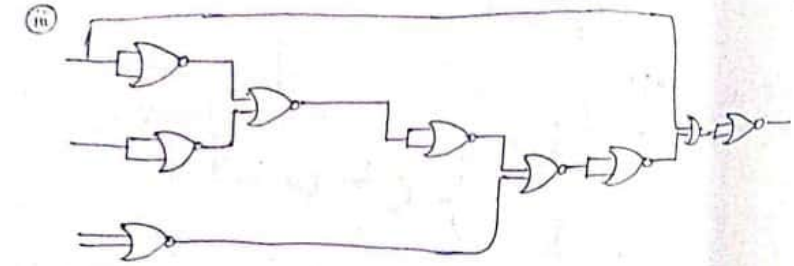




(i) (ii)

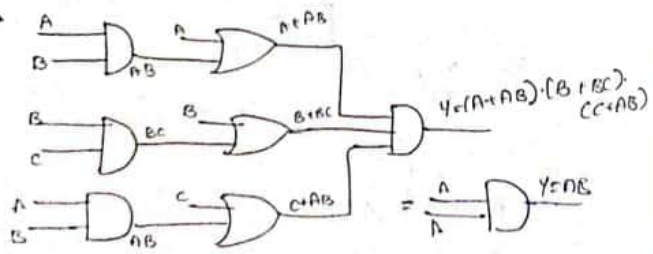


(iii)



Q Write the Boolean expression for below diagram & simplify as much as possible & draw the circuit.

→



$Y = (AB+A).(B+BC).(C+AB)$

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

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

$= AB(C+AB)$

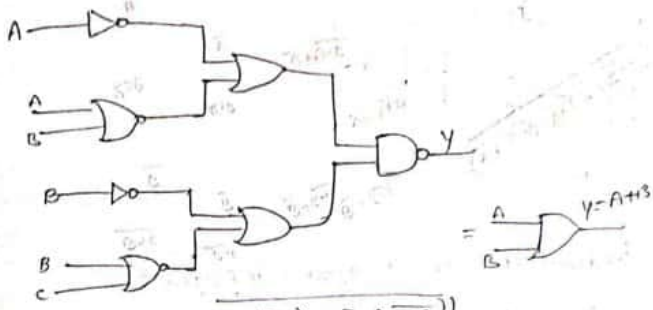
$= ABC + AB$

$= AB(1+C)$

$= AB$

↳

Redraw the circuit after simplification.



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

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

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

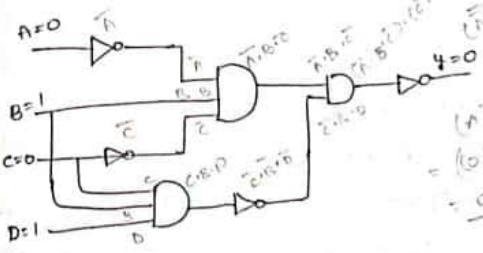
$= A + AB + B + BC$

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

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

$= A + B$

↳

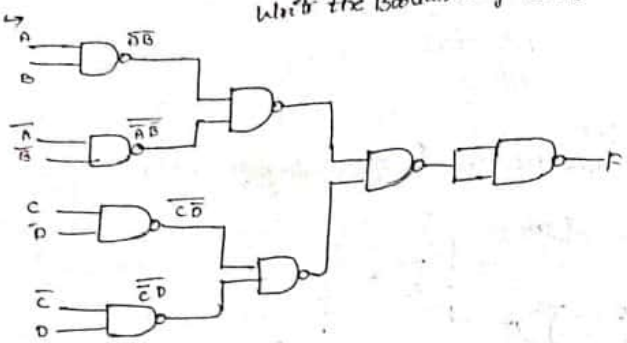


$(A\bar{B}C)(\bar{A}+B+\bar{C})$

$= (0)(0)(1)$

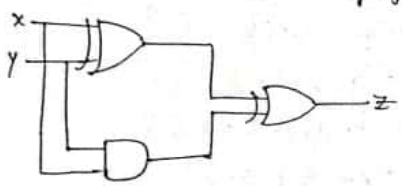
$= 0$

↳ Write the Boolean expression



F =

↳ Redraw the circuit after simplification



↳ Reduce the Boolean expressions:

① $\overline{\overline{x}\overline{y}z} + xy + wz$

② $\overline{\overline{A}C} + ABc + A\overline{C}$

③ $(\overline{x}\overline{y} + z) + z + xy + wz$

④ $\overline{A}B(\overline{D} + \overline{C}D) + B(A + \overline{A}CD)$

⑤ $xyz + x'y + xyz'$

⑥ $AB + A(B+C) + B(B+D)$

⑦ $A + B + AB'c$

⑧ $A(A+B)$

⑨ If $F = A + BC$ find $\overline{F}$ and show that

ⓐ $F + \overline{F} = 1$ ⓑ $F \cdot \overline{F} = 0$

⑩ $(A'Bc')' + (AB'c)'$

⑪ $(A+B)(A + (AB)'c') + (A'(B+c'))' + A'B + ABC$

⑫ $xy + x\overline{y} + x\overline{y}z\,(xy + z)$

⑬ $\overline{A}\overline{c} + ABC + A\overline{c}$

⑭ $(A'+c)(A'+c')(A+B+c'D)$

⑮ $\overline{\overline{A} \cdot \overline{AB}} \cdot \overline{B\,\overline{AB}}$

⑯ $(ABC)' + (AB)'c + A'BC + A(BC)' + AB'C$

→ ④ $\overline{A}\overline{c} + ABC + A\overline{c}$

Ans:- $= \overline{c}(\overline{A} + A) + ABC$

$= \overline{c} + ABC$

$= (\overline{c} + c)(\overline{c} + AB)$

$= \overline{c} + AB$

⑤ $(\overline{\overline{x}\overline{y} + z}) + z + xy + wz$

$= \overline{\overline{x}\overline{y}} \cdot \overline{z} + z(1 + w) + xy$

$= (x+y)\overline{z} + z + xy$

$= x + y + z + xy$

$= x + y + z$

⑥ $\overline{A}B(\overline{D} + \overline{C}D) + B(A + \overline{A}CD)$

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

$\overline{A}B + \overline{D}(C + c) + \overline{A}BCD + AB(C + \overline{C})(D + \overline{D}) + \overline{A}BCD$

$\overline{A}BC\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + (ABC + AB\overline{C})(D + \overline{D}) + \overline{A}BCD$

$\overline{A}BC\overline{D} + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + ABC\overline{D} + AB\overline{C}\overline{D} + \overline{A}BCD$

$\overline{A}B(C\overline{D} + \overline{C}\overline{D} + BC + CD) + ABC\overline{D} + AB\overline{C}\overline{D}$

$\overline{A}B(CD + \overline{D})(C + \overline{C}) + \overline{C}\overline{D}) + ABC\overline{D} + AB\overline{C}\overline{D}$

$\overline{A}B(1 + \overline{C}\overline{D}) + ABC\overline{D} + AB\overline{C}\overline{D}$

$(\overline{A}B + ABC\overline{D} + AB\overline{C}\overline{D}$

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

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

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

$AB\bar{D} + \bar{A}B$

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

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

$ABC\bar{D} + AB\bar{C}\bar{D} + \bar{A}BC\bar{D} + \bar{A}B\bar{C}D$

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

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

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

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

④ $xyz + x'y + xyz'$

$xyz + \bar{x}y + yz$

$xyz + \bar{x}y(z+\bar{z}) + (x+\bar{x})yz$

$xyz + \bar{x}yz + \bar{x}y\bar{z} + xyz + \bar{x}yz$

$xyz + \bar{x}yz + \bar{x}y\bar{z}$

$yz(x+\bar{y}) + \bar{x}y\bar{z}$

$yz + \bar{x}y\bar{z}$

$(z+\bar{z})(y+\bar{x}y)$

$y + \bar{x}y$

⑨ $A + B + A\bar{B}\bar{C} = A(B+\bar{B})(C+\bar{C}) + \bar{A}BC$

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

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

$= A\bar{B}C + ABC + AB\bar{C} + A\bar{B}\bar{C}$

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

$= A\bar{B} + AB$

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

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

⑩ $A(A+B) = AA + AB = A + AB = A(1+B) = A$

⑪ If $F = A+B+C$ then $\overline{A+B+C} = \bar{A}\bar{B}\bar{C}$

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

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

$= 0 + 0 + 0$

$F \cdot \bar{F} = 0$

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

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

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

$= 1$

⑫ $(A'BC')' + (AB'C)'$

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

$= \overline{\bar{A}\bar{C}B} + \overline{AC\bar{B}}$

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

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

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

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

$= 1 + 1 + B$

$= B$

⑬ $(A+B)(A+(AB)'C) + (A'(B+C')) + A'B + ABC$

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

$= AA + A\bar{A}\bar{B}C + AB + \bar{A}\bar{B}BC + \bar{A}B + \bar{A}\bar{C} + \bar{A}B + ABC$

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

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

$= A + 0 + A\bar{B}C + AB + \bar{A}\bar{B} + \bar{A}BC + 0 + \bar{A}\bar{C} + ABC$

$= A + ACCB + \bar{B} + AB + \bar{A}\bar{B} + \bar{A}\bar{C} + ABC$

$= A + AC + BC(\bar{A}+A) + \bar{A}\bar{C} + ABC$

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

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

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

$= 1 + 1 + ABC$

$= ABC$

(14) $(z+\bar{x})z\bar{y} + \bar{y}\bar{x} + \bar{x}$

$z\bar{y}\bar{x} + z\bar{y}\bar{\lambda}\bar{x} + \bar{\lambda}\bar{y} + \bar{\lambda}\bar{x}$

$= z\bar{y} + \bar{y} + \bar{x}$

$= z\bar{y}x + (z+z)\bar{y}(\bar{x}\bar{x}) + (\bar{z}+z)(\bar{y}+\bar{\lambda})\bar{x} + (\bar{z}+z)\bar{\lambda}x$

$= z\bar{y}x + \bar{z}\bar{\lambda}x + \bar{z}\bar{y}\bar{x} + z\bar{\lambda}\bar{x} + \bar{z}\bar{\lambda}x + z\bar{\lambda}x$

$z\bar{y}x + \bar{y}\bar{\lambda}(\bar{x}\bar{z} + \bar{z}x + z\bar{x}) + \bar{\lambda}(\bar{z}\bar{x} + \bar{z}\bar{\lambda} + z\bar{\lambda}\bar{x} + \bar{z}\bar{\lambda}x + z\bar{\lambda}x$

$z\bar{y}\bar{\lambda}x + z\bar{\lambda}\bar{x} + z\bar{\lambda}\bar{x} +$

$= z\bar{y}\bar{\lambda}x + z\bar{\lambda}\bar{x} + \bar{\lambda}z\bar{x} + \bar{z}\bar{\lambda}\bar{x} + z\bar{\lambda}\bar{x} + \bar{z}\bar{\lambda}x + z\bar{\lambda}x$

$z\bar{\lambda}x + (\bar{z}+x)z\bar{\lambda} + z\bar{\lambda}\bar{x} + \bar{z}\bar{\lambda}\bar{x} + z\bar{\lambda}\bar{x} + (\bar{z}+z)\bar{\lambda}x =$

$z\bar{\lambda}x + \bar{z}\bar{\lambda} + z\bar{\lambda}\bar{\lambda}\bar{x} + z\bar{\lambda}\bar{x} + (\bar{z}+z)\bar{\lambda}x =$

$\bar{z}\bar{\lambda} + \bar{z}\bar{\lambda}\bar{x} + z\bar{\lambda}\bar{x} + (\bar{z}+z)\bar{\lambda}x$

$= \bar{z}\bar{\lambda} + \bar{\lambda}\bar{x} + \bar{z}\bar{\lambda}\bar{x} + (\bar{z}+z)\bar{\lambda}x$

$= \bar{z}\bar{\lambda} + \bar{\lambda}\bar{x} + \bar{\lambda}x + (\bar{z}+z)\bar{\lambda}x$

$= \bar{z}\bar{\lambda} + \bar{\lambda}(\bar{x}+x) + (\bar{z}+z)\bar{\lambda}x$

$= \bar{\lambda}(\bar{z}+z) + \bar{\lambda}(x+\bar{x}) + \bar{\lambda}x$

$= \bar{\lambda} + \bar{\lambda}x$

$= 1$

(15) $\bar{A}\bar{C} + ABC + A\bar{C}$

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

$= \bar{A}B\bar{C} + \bar{A}\bar{B}\bar{C} + ABC + AB\bar{C} + A\bar{B}\bar{C}$

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

$= AC + \bar{A}B\bar{C} + \bar{A}\bar{B}\bar{C} + AB\bar{C}$

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

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

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

$= 1 + AB\bar{C}$

$= AB\bar{C}$

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

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

$= (A+\bar{A}+\bar{B}+AC+0)(A+B+\bar{C}D)$

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

$= A + AB + A\bar{C}D + 0 + \bar{A}B + \bar{A}\bar{C}D + A\bar{C} + \bar{A}BC + 0$

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

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

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

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

$= 1$

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

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

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

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

$= (0+AB)(\bar{A}B+0)$

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

$= AB(1)$

$= AB$

18) $(ABC)' + (AB)'C + A'BC + A(BC)' + AB'C$

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

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

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

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

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

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

$= \bar{B} + AB\bar{C}$

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

$= A\bar{C}$

3) $\overline{x\bar{y}z} + xy + wz$

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

$= x[1 + y] + y + \bar{z} + wz$

$= [x + y + \bar{z}] + wz$

$= (x + y + \bar{z} + w) \cdot (x + y + \bar{z} + z)$

$= (x + y + \bar{z} + w) \cdot (x + y + 1)$

$= x + y + \bar{z} + w$

Unit-2

Gate level minimization

Boolean laws are used to minimize logical expression. It is comfortable if the no. of variable used in the functions are 2 and 3.

So that we are going for gate level minimization techniques to reduce large Boolean-function.

1) K-map (Karnaugh map)

2) Tabular method.

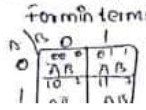
K-map:-

K maps are extensively used for the minimization of functions having Variables 3, 4, 5, 6.

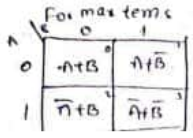
* K map is defined as map in which truth table is put in compact form by labeling the rows and columns.
* The rows and columns are assigned a gray code such that two adjacent rows (or) columns differ in one-bit only.
* Note that column on extreme left is adjacent of to column on extreme right. Ily Top row and bottom row are also adjacent.
* K map consists of no. of cells that represents min terms and max terms
* 1 represents min terms and 0 represents max terms.
* 2 cells are said to be adjacent to each other if they are physically adjacent or can be adjacent by wrapping the map from left to right or top to bottom.

Two variable K-maps:-

For min terms

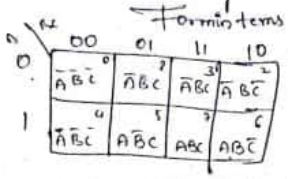


For max terms

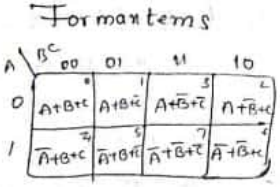


Three variable k-map

For min terms

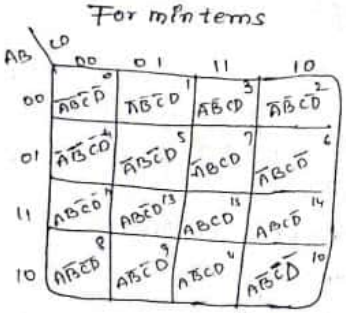


For max terms

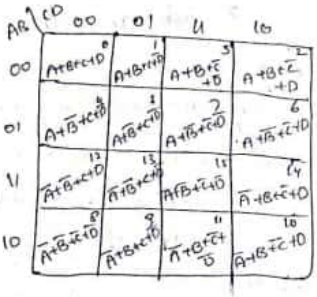


4 Variable k-map

For min terms



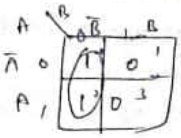
For max terms



Problem:-

* Map the following function over k-map write the standard form and also simplify by grouping procedure.

1) $f = \Sigma(0,2)$



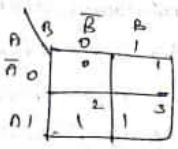
$f = \Sigma(0,2) \rightarrow SOP$

$= m_0 + m_2$

$= 00 + 10 \rightarrow \bar{A}\bar{B} + A\bar{B}$

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

ii) $f = \Sigma(2,3)$



$= m_2 + m_3 = A\bar{B} + AB$

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

$= A.$

3) $f = \Pi(0,1)$

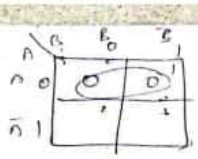
$= M_0 \cdot M_1 = (A+B)\cdot(A+\bar{B})$

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

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

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

$= A$



4. $f = \Pi(1,3)$

$= M_1 \cdot M_3 = (A+\bar{B}).(\bar{A}+\bar{B})$

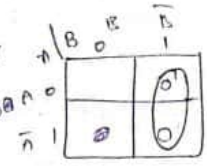
$= A\bar{A} + A\bar{B} + \bar{A}\bar{B} + \bar{B}\cdot\bar{B}$

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

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

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

$= \bar{B}(1) = \bar{B}$



5) $f = \Pi(1,2)$

$f = \Pi(1,2)$

$= M_1 \cdot M_2 = (A+\bar{B})\cdot(\bar{A}+B)$

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

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

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

c) $f = \Sigma(1,5,6,7)$

$f = \Sigma(1,5,6,7)$

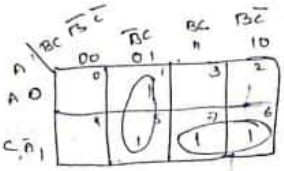
$= M_1 + M_5 + M_6 + M_7$

$= 001 + 101 + 110 + 111$

$= \bar{A}\bar{B}C + AB\bar{C} + ABC + A\bar{B}C$

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

$= AB + \bar{B}C$ //



7) $f = \Sigma(1,2,3,5,7)$

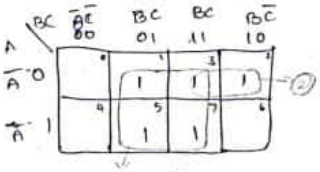
$f = \Sigma(1,2,3,5,7)$

$= M_1 + M_2 + M_3 + M_5 + M_7$

$= 001 + 010 + 011 + 101 + 111$

$= \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC$

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



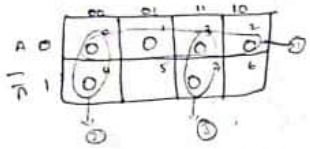
Quad ①

$f(A,B,C) = ① + ②$

$= C + \bar{A}B$

③ $f = \pi(0,1,2,3,4,7)$

$f = \pi(0,1,2,3,4,7)$



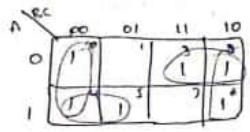
$F = ① \cdot ② \cdot ③$

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

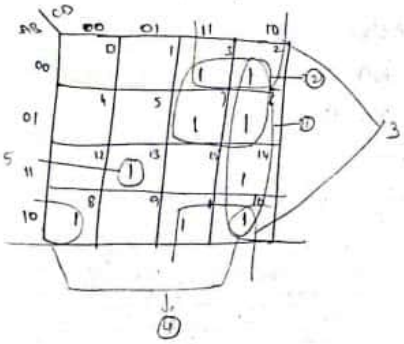
$= A \cdot (B\bar{B} + B\bar{C} + \bar{B}C + C\cdot\bar{C})$

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

④ $f = \Sigma(0,2,3,4,5,6)$



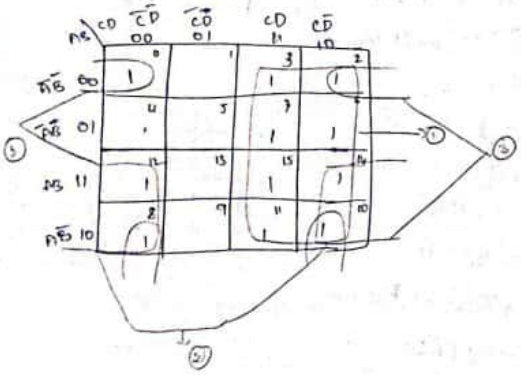
⑤ $f = \Sigma(2,3,5,7,8,10,11,13,14)$ → 0 → $\bar{A}$, 1 → A



① + ② + ③ + ④ + ⑤

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

⑥ $f = \Sigma(0,2,3,6,7,8,10,11,12,14,15)$

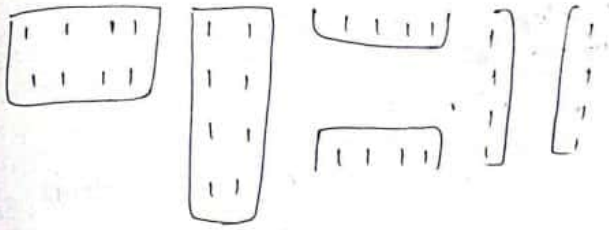


$f = ① + ② + ③$

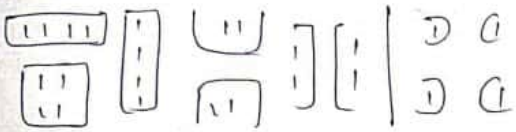
$F = C + A\bar{D} + \bar{B}\bar{D}$

Octants:

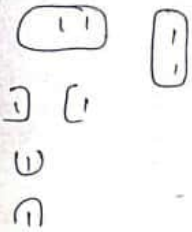
For minterms:-



Quad:-



Pair:-



Note:-

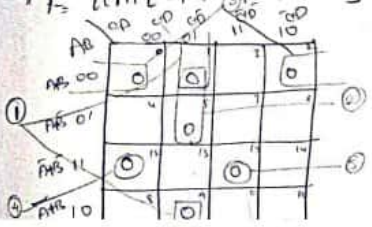
↳ Octant removes 3 variables

↳ Quard removes 2 variables

↳ pair removes 1 variables

} in SOP or POS form

* $F = \Pi m[0,1,2,5,9,15]$



$f(A,B,C,D) = ① + ② + ③ + ④ + ⑤$

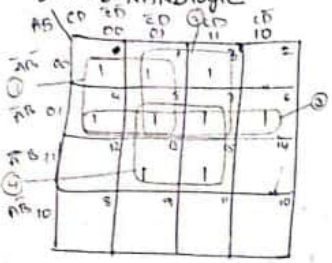
$\rightarrow (B+C+\bar{D})(A+C+\bar{D})(A+B+D)$

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

Simplify the following function $\Sigma m(0,1,3,4,5,6,7,13,15)$ by using k map also draw logic dia by using

1) AND-OR logic

2) NAND-NAND logic

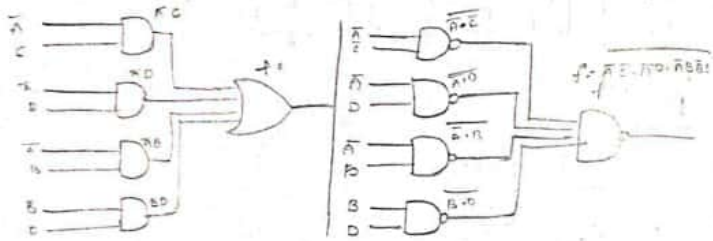


$f(A,B,C,D) = ① + ② + ③ + ④$

$\rightarrow \bar{A}\bar{C} + \bar{A}D + \bar{A}B + BD$

$\rightarrow$ sop-form

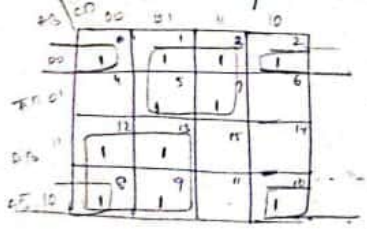
1) AND-OR logic



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

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

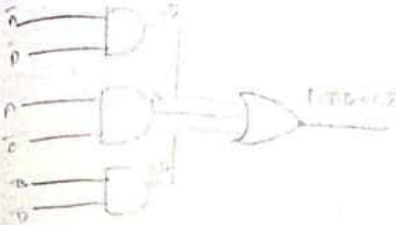
Reduce the following expression using k MAP $f = \Sigma(0,1,2,3,5,7,8,9,10,12,13)$ and implement the real minimal expression



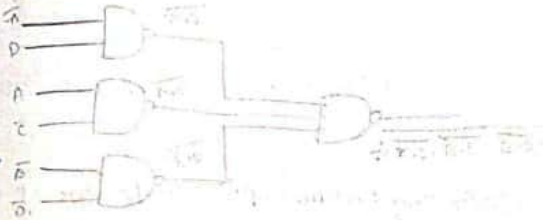
$f(A,B,C,D) = ① + ② + ③$

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

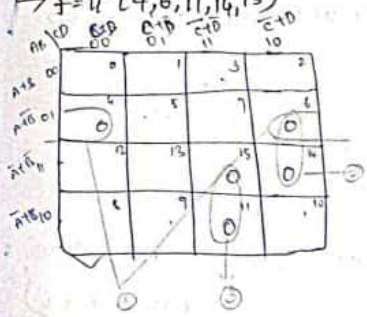
AND-OR logic



NAND-NAND logic:-

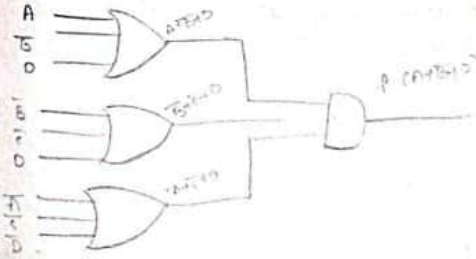


$\rightarrow f = \pi(4,6,11,14,15)$

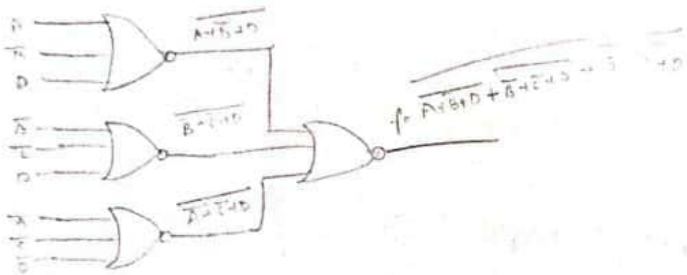


$f = (A+\bar{B}+D) \cdot (\bar{B}+\bar{C}+D) \cdot (\bar{A}+\bar{C}+\bar{D}) \rightarrow$ pos

OR-AND logic

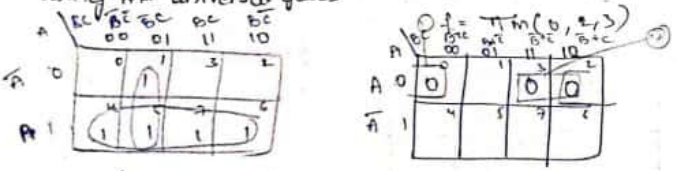


NOR-NOR logic



$f = \overline{AD} + A\overline{C} + \overline{B}\overline{D}$ is real minimal expression, so, universal logic is NAND-NAND logic.

→ Simplify the following Boolean expression to a min no. of minimal expression no. literals $f(A,B,C) = \Sigma m(1,4,5,6,7)$. Simplify & implement the real minimal expression using min universal gates

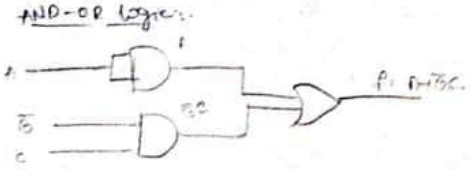


$f = A + \overline{B}C \rightarrow$ sop form $\quad f(A,B,C) = (0)(2) \rightarrow$ POS $\quad (A+\overline{B}).(A+C)$

Real minimal exp is sop form.

AND/OR logic Here in sop form we are getting 3 literals and in pos form we are getting 4 literals so real minimal exp is sop form.

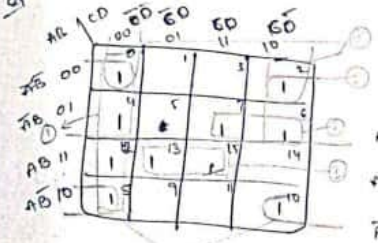
AND-OR logic:



NAND-NAND logic

Reduce the following expression using K-map Implement the minimal exp in universal gates $f = \Sigma m(0,2,4,6,7,8,10,12,13,15)$

$f = \Sigma m(0,2,4,6,7,8,10,12,13,15)$ Sop form



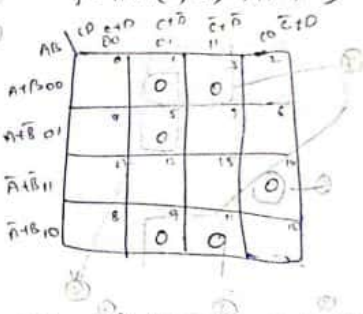
$f = \overline{A}\overline{D} + \overline{C}\overline{D} + \overline{B}\overline{D} + AB\overline{D} + \overline{A}BC$

↓

NAND-NAND logic

$f = \overline{\overline{A}\overline{D} \cdot \overline{C}\overline{D} \cdot \overline{B}\overline{D} \cdot \overline{AB\overline{D}} \cdot \overline{\overline{A}BC}}$

$f = \Pi m(1,3,5,9,11,14)$



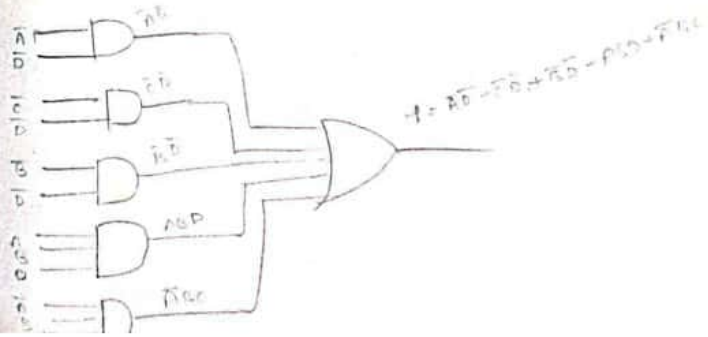
$f = (\overline{B}+\overline{D}).(\overline{A}+\overline{B}+\overline{C}+D).(A+\overline{C}+\overline{D})$

NOR-NOR-logic

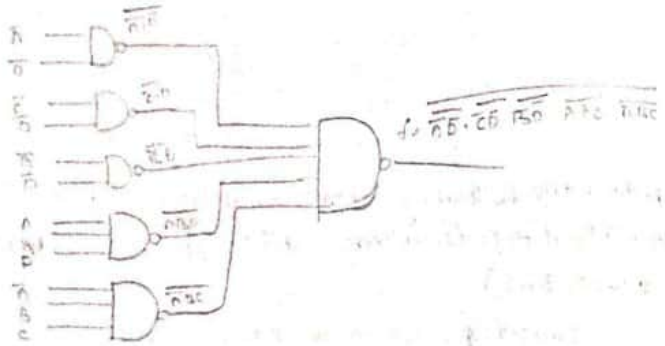
↓

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

AND-OR

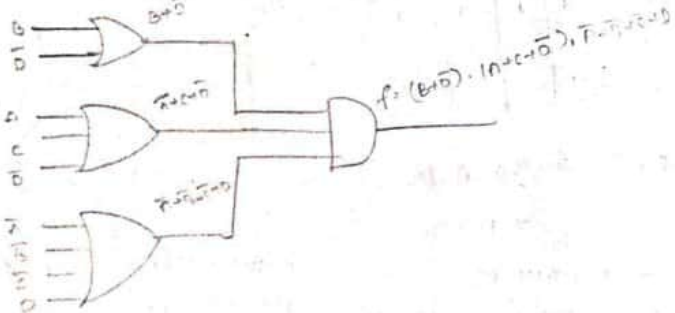


NAND-NAND

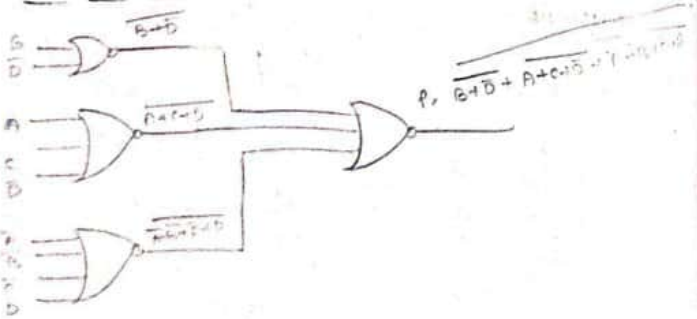


Pos form

OR AND logic

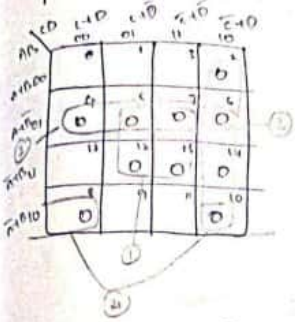


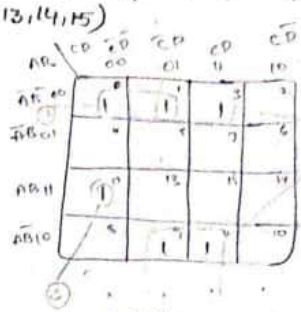
NOR-NOR logic



Reduce the following expression using k maps & implement the real minimum expression using logic gates.

$f = \pi M(0,4,5,6,7,8,10,13,14,15)$





sop form

$f = ① + ② + ③ + ④$

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

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

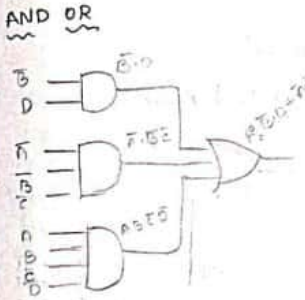
$f = ① + ② + ③$

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

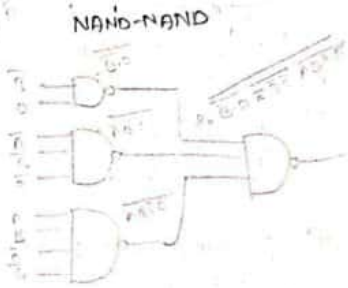
real logic logic is NAND-NAND & logic.

sop form

AND OR



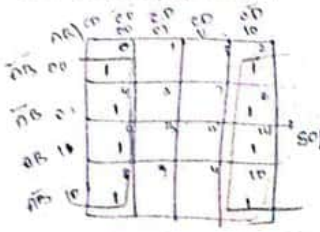
NAND-NAND



Hw

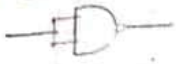
1) $f = \Sigma(0,2,4,6,8,10,12,14)$
2) $f = \Sigma(2,3,5,7,10,12,13)$
3) $f = \Sigma(0,2,4,6,9,13)$
4) $f = \pi(0,2,4,8,9,12,14)$
5) $f = \pi(0,2,3,8,9,12,13,15)$

1) $f = \Sigma(0,2,4,6,8,10,12,14)$

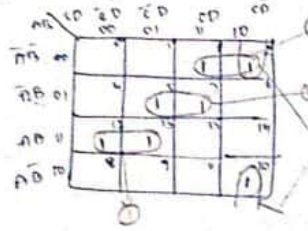


$f = \bar{C}\bar{D} + C\bar{D}$

sop form $f = \bar{D}$



2) $f = \Sigma(2,3,5,7,10,12,13)$

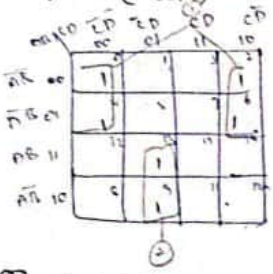


$f = ① + ② + ③ + ④$

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

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

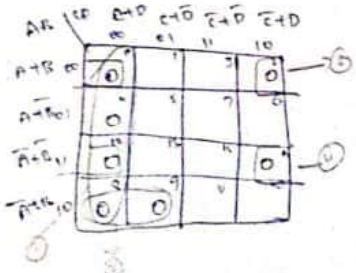
3) $f = \Sigma(0,2,4,6,9,13)$



$f = ① + ②$

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

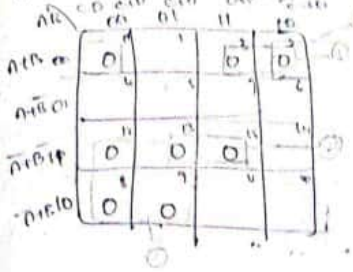
4) $f = \pi M(0,2,4,8,9,12,14)$ pos



$f = ① \cdot ② \cdot ③ \cdot ④$

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

5) $f = \pi M(0,2,3,8,9,12,13,15)$



$f = ① \cdot ② \cdot ③ \cdot ④$

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

Don't care conditions:-

Generally for any function the value can be expressed either 1 or 0 for all the input combinations

→ sometimes output is unspecified for some combinations of input. in that case it can be treated as either 0 or 1 and these conditions are called Don't care conditions.

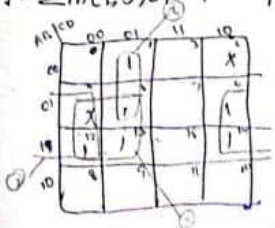
→ It is denoted by 'd' or 'x' or 'φ'

→ When we are doing rounding procedure in sop these don't care are treated as one's. and in Pos form these don't care can be treated as 0's.

→ All don't cares should not be rounded in sop and pos form.

→ When we are converting from sop form to pos form these don't cares should be retained as it is.

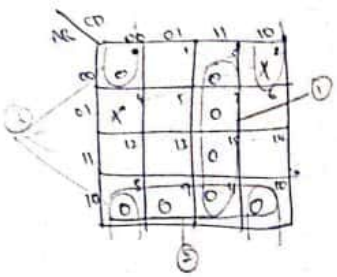
→ Simplify the following function by using K map $f = \Sigma m(1,5,6,12,13,14) + d(2,4)$ & implement the minimal expression using logic.



$f(A,B,C,D) = ① + ② + ③$

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

→ $\pi M(0,3,7,8,9,10,11,15) + d(2,4)$

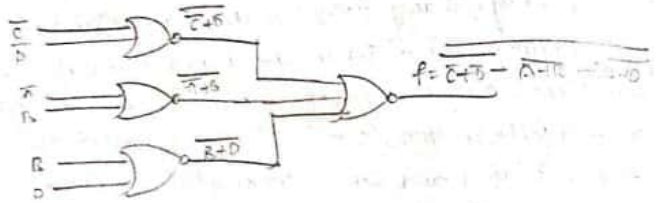


$f = ① \cdot ② \cdot ③$

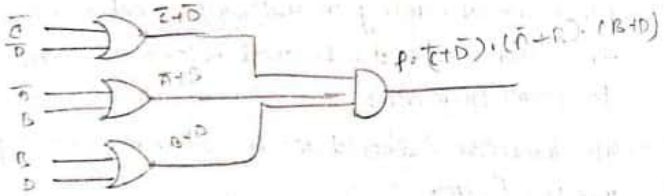
$= (\bar{C}+\bar{D}) \cdot (\bar{A}+B) \cdot (B+D)$

Here pos form is real minimal expression so universal logic for $f = \bar{z}$. is NOR - NOR logic

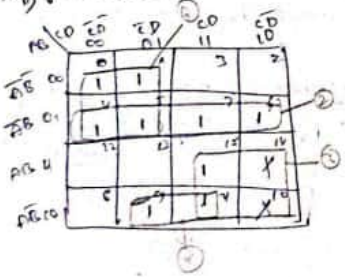
NOR - NOR logic



OR AND logic.



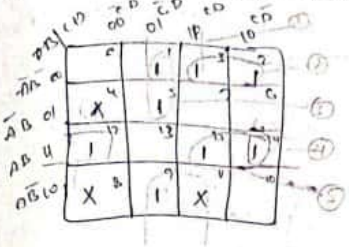
→ ii) $f = \Sigma(0,1,4,5,6,7,9,11,15) + d(10,14)$



$f = ① ② ③ ④$

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

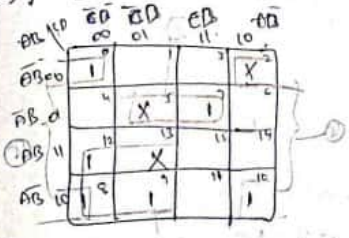
→ $f = \Sigma m(1,2,3,5,9,12,14,13) + d(4,8,11)$



$f = ① + ② + ③ + ④ + ⑤$

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

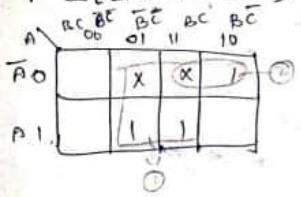
→ $f = \Sigma(0,7,8,9,10,12) + d(2,5,13)$



$f = ① + ② + ③$

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

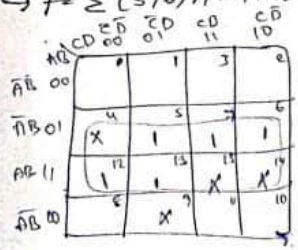
→ $f = \Sigma(2,5,9) + d(1,3)$



$f = ① + ②$

$= C + \bar{A}B$

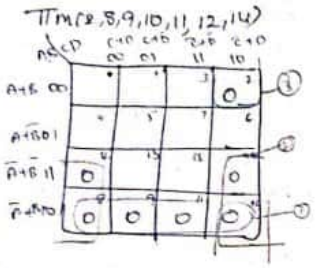
→ $f = \Sigma(5,6,7,12,13) + d(4,9,14,15)$



$f = B$

Reduce the following expression using k map and implement the minimal expression using Universal gates $f = \pi M(2,8,9,10,11,12,14)$

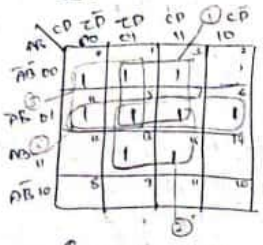
$\pi M(2,8,9,10,11,12,14)$



$f = ① \cdot ② \cdot ③$

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

$\Sigma(0,1,3,4,5,6,7,13,15)$

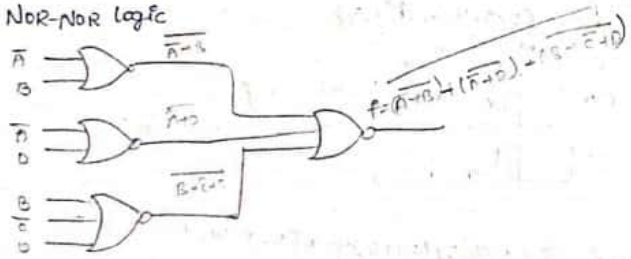


$f = ① + ② + ③ + ④$

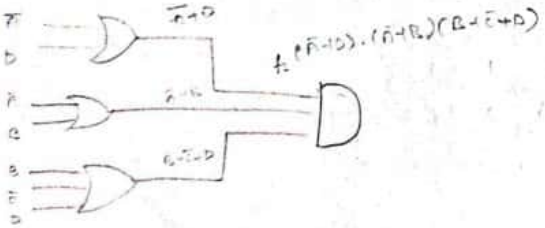
$= \bar{A}D + BD + \bar{A}\bar{C} + \bar{A}B$

∴ pos have the no. of minimum literals so pos is the real minimal expression.

NOR-NOR logic



OR-AND logic:-



→ For a 5-variable function $f(A,B,C,D,E)$ minterms are $2^5 = 32$

→ 32 minterms for SOP form, 32 maxterms for POS form.

→ Minterms are $m_0, m_1, m_2 \ldots m_{31}$ are represented as

$m_0 \rightarrow \bar{A}\bar{B}\bar{C}\bar{D}\bar{E}$

$m_1 - \bar{A}\bar{B}\bar{C}\bar{D}E$

$m_2 - \bar{A}\bar{B}\bar{C}D\bar{E}$

$m_3 - \bar{A}\bar{B}\bar{C}DE$

$m_4 - \bar{A}\bar{B}C\bar{D}\bar{E}$

$m_5 - \bar{A}\bar{B}C\bar{D}E$

$m_6 - \bar{A}\bar{B}CD\bar{E}$

$m_7 - \bar{A}\bar{B}CDE$

$m_8 - \bar{A}B\bar{C}\bar{D}\bar{E}$

$m_9 - \bar{A}B\bar{C}\bar{D}E$

$m_{10} - \bar{A}B\bar{C}D\bar{E}$

$m_{11} - \bar{A}B\bar{C}DE$

$m_{12} - \bar{A}BC\bar{D}\bar{E}$

$m_{13} - \bar{A}BC\bar{D}E$

$m_{14} - \bar{A}BCD\bar{E}$

$m_{15} - \bar{A}BCDE$

$m_{16} - A\bar{B}\bar{C}\bar{D}\bar{E}$

$m_{17} - A\bar{B}\bar{C}\bar{D}E$

$m_{18} - A\bar{B}\bar{C}D\bar{E}$

$m_{19} - A\bar{B}\bar{C}DE$

$m_{20} - A\bar{B}C\bar{D}\bar{E}$

$m_{21} - A$

$m_{22} -$

$m_{23} -$

* Maxterms are $M_0, M_1, M_2 \ldots M_{31}$ are

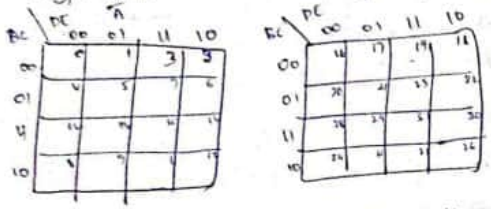
$M_0 = A+B+C+D+E$

$M_1 = A+B+C+D+\bar{E}$

$M_2 = A+B+C+\bar{D}+E$

$\vdots$

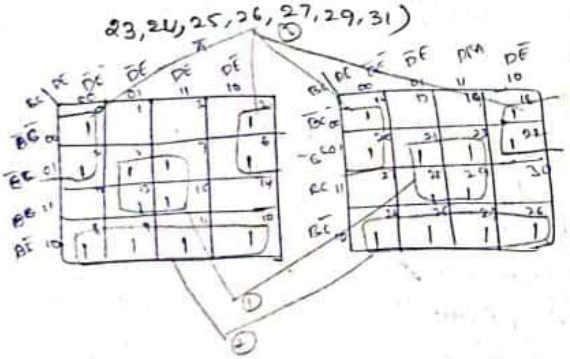
$M_{31} = \bar{A}+\bar{B}+\bar{C}+\bar{D}+\bar{E}$



* 32 cells of the K map are divided into 16 cells for 2 kmaps
* When we are super imposing the one block of the top of the other then the cells coincide with one another are said to be adjacent to each other.

→ Simplify the following expression by using K map

$f = \Sigma(0,2,4,5,6,7,8,9,10,11,13,15,16,18,20,21,22,23,24,25,26,27,29,31)$

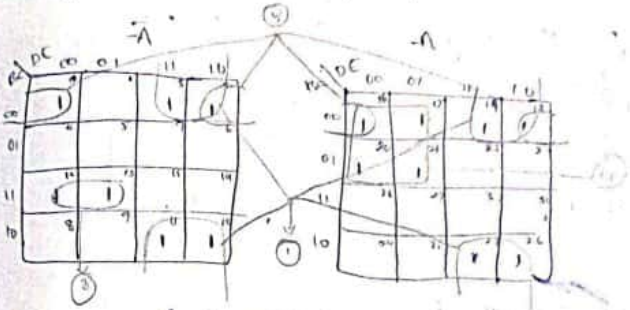


$f = ① + ② + ③$

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

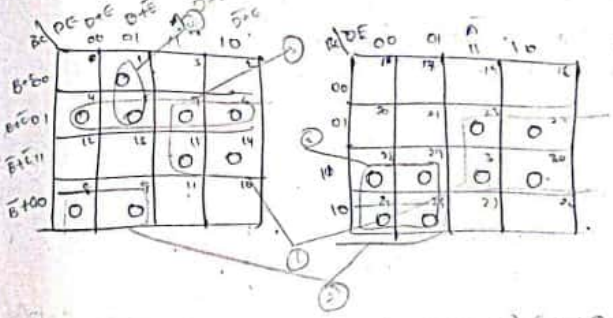
→ Simplify the following function by using K map and implement real minimal expression using universal gates.

$f = \Sigma(0,2,3,10,11,12,13,16,17,18,19,20,21,25,29)$



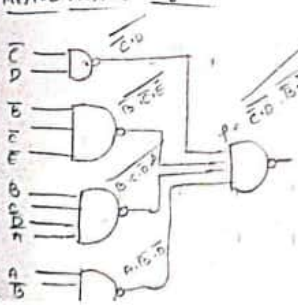
Sop :- $f = \bar{C}D + \bar{B}C\bar{E} + BC\bar{D}\bar{A} + A\bar{B}C\bar{D}$ $A\bar{B}\bar{D}$

$f = \Pi(1,4,5,6,7,8,9,14,15,22,23,24,25,28,29,30,31)$

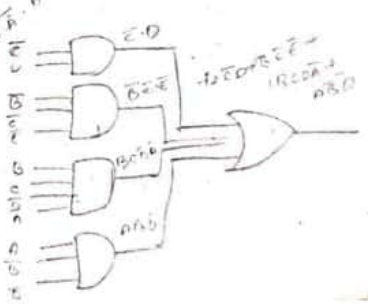


$f = (\bar{C}+\bar{D}).(\bar{B}+C+D).(\bar{A}+\bar{B}+\bar{D}).(A+B+D+\bar{E}).(A+B+C)$

NAND NAND logic



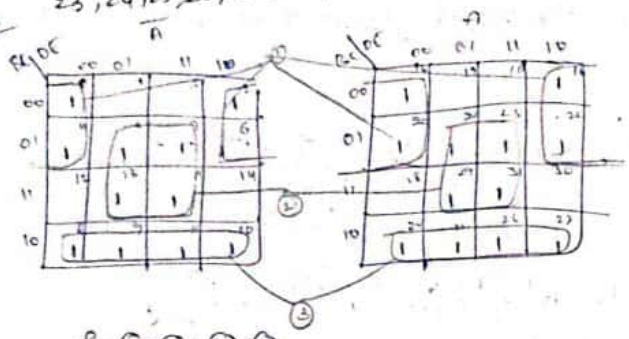
AND OR logic :



Simplify the following function using k map

$f = \Sigma m(0,1,4,5,6,7,8,9,10,11,13,15,16,18,20,21,22,23,24,25,26,27,29,31)$

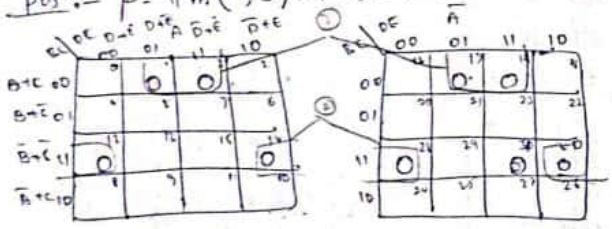
Sop



$f = ① + ② + ③ + ④$

$= \bar{B}\bar{E} + CE + \bar{B}\bar{C}$

pos :- $F = \Pi m(1,5,12,14,17,19,28,30)$



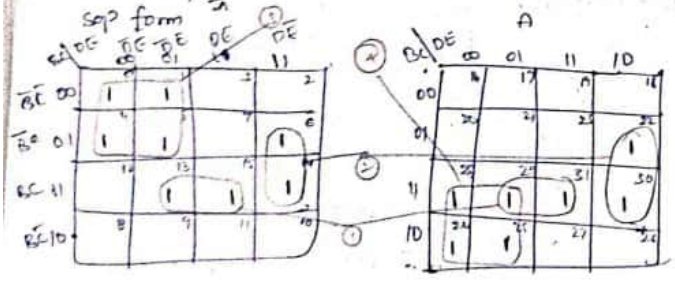
$f = ① \cdot ②$

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

Simplify the following function using kmap, pos, sop form

$f = \Sigma(0,1,4,5,6,13,14,15,22,24,25,28,29,30,31)$

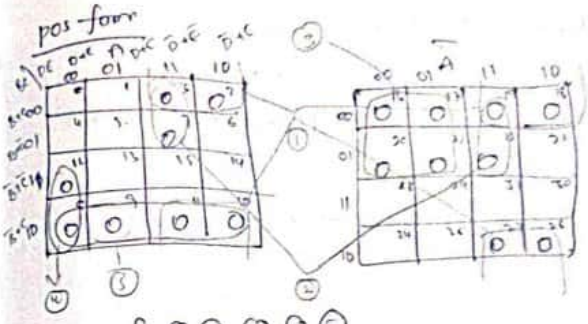
Sop form



$f = ① + ② + ③ + ④$

$f = \bar{B}\bar{E} + \bar{B}\bar{C} + \bar{C}\bar{E} + CDE$

pos form

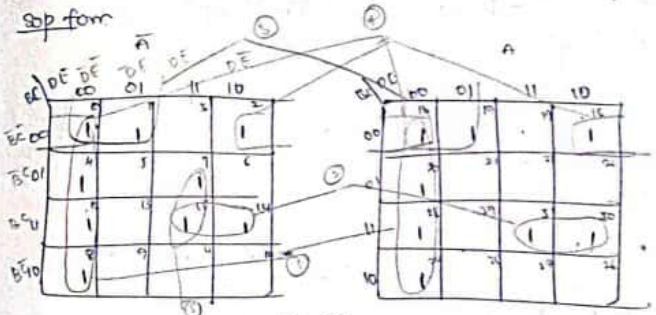


$f = ① \cdot ② \cdot ③ \cdot ④ \cdot ⑤$

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

→ ii) $f = \Sigma(0,1,2,4,7,8,12,14,15,16,17,18,20,24,28,30,31)$

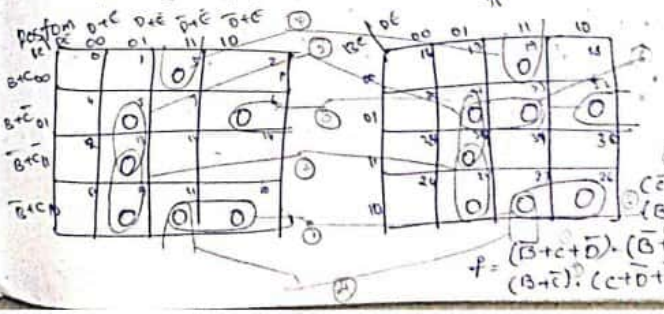
Sop form



$f = ① + ② + ③ + ④ + ⑤$

$= \bar{D}\bar{E} + BCD + \bar{B}\bar{C}\bar{D} + \bar{B}\bar{C}\bar{E} + \bar{A}CDE$

pos form



$f = (\bar{B}+C+\bar{D}) \cdot (\bar{B}+D+\bar{E}) \cdot (B+\bar{C}) \cdot (C+\bar{D}+\bar{E})$

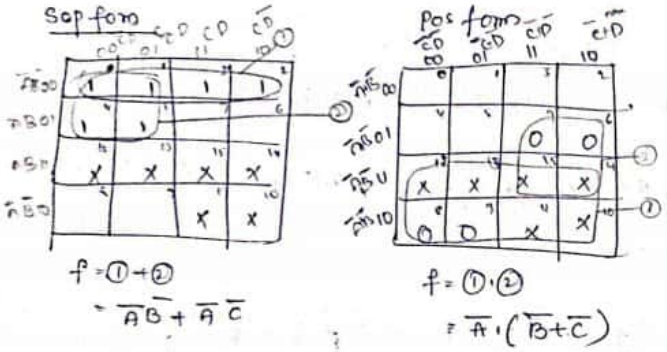
→ simplify the following function in pos form using pos form

i) $f = \Sigma(5,6,7,8,9,12,13,14), \Pi(0,1,2,3,4,10,11,15)$

ii) $f(w,x,y,z) = \Sigma(0,1,5,7,8,10,14,15)$

iii) $f(A,B,C,D) = \Sigma(0,1,2,3,4,5) + d(10,11,12,13,14,15)$
$\Pi(6,7,8,9) + d(10,11,12,13,14,15)$

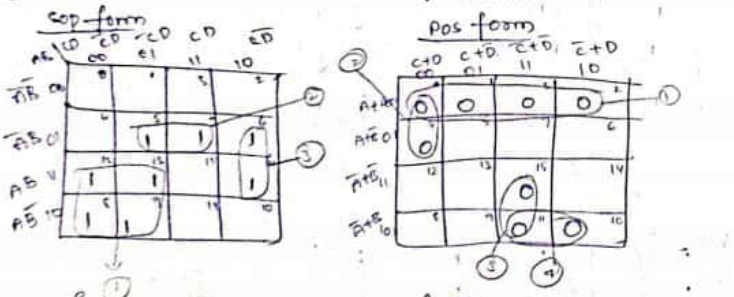
iii) $f = \Sigma(0,1,2,3,4,5) + d(10,11,12,13,14,15)$



$f = ① + ②$
$= \overline{A}\overline{B} + \overline{A}\overline{C}$

$f = ① \cdot ②$
$= \overline{A} \cdot (\overline{B} + \overline{C})$

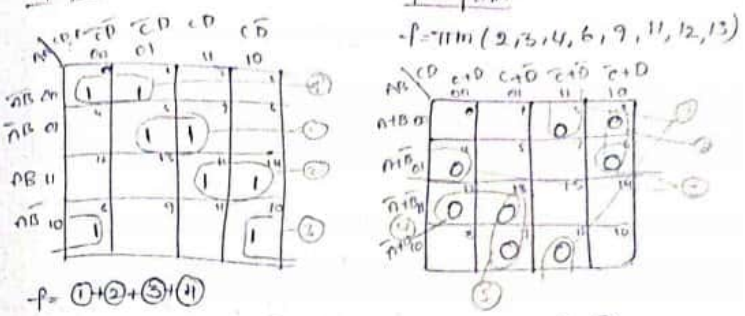
i) $f = \Sigma(5,6,7,8,9,12,13,14)$ $\quad f = \Pi(0,1,2,3,4,10,11,15)$



$f = ① + ② + ③$
$= A\overline{C} + \overline{A}BD + BC\overline{D}$

$f = ① \cdot ② \cdot ③ \cdot ④$
$= (A+B) \cdot (A+C+D) \cdot (\overline{A}+\overline{C}+\overline{D}) \cdot (\overline{A}+\overline{B}+\overline{C})$

ii) $f(w,x,y,z) = \Sigma(0,1,5,7,8,10,14,15)$



$f = ① + ② + ③ + ⑪$
$= A\overline{B}D + ABC + A\overline{B}\,\overline{D} + \overline{A}\overline{B}\overline{C}$

$f = ① \cdot ② \cdot ③ \cdot ④ \cdot ⑤$
$= (B+\overline{C}+\overline{D}) \cdot (A+\overline{C}+D) \cdot (\overline{A}+\overline{B}+D) \cdot (\overline{A}+\overline{B}+C) \cdot (\overline{A}+C+\overline{D})$

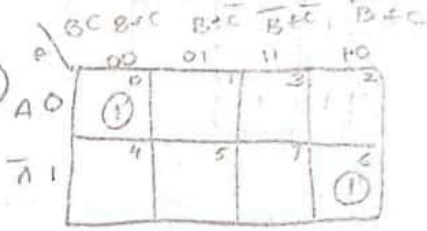
Other 2-level Implementations:

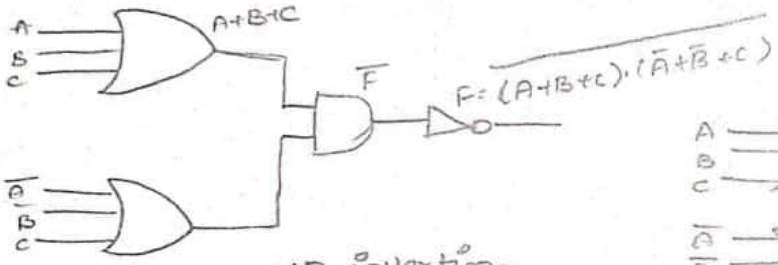
Complement the function $f = \Sigma(0,6)$ with the four 2-level forms listed below

a) AND-NOR b) NAND-NAND c) OR-NAND d) NOR-OR

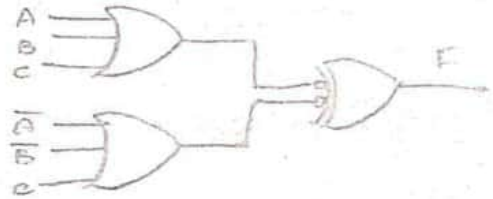
$\overline{F} = (A+B+C).(\overline{A}+\overline{B}+C)$ (one's Grouping by pos form)

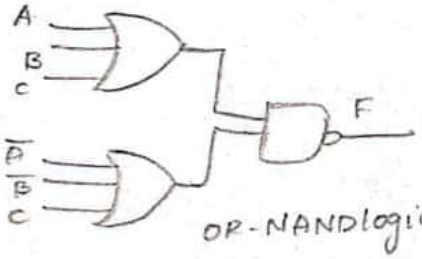
$F = \overline{(A+B+C).(\overline{A}+\overline{B}+C)}$



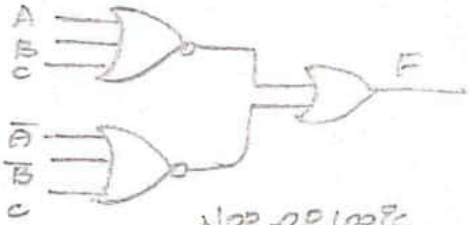


OR-AND-inverting

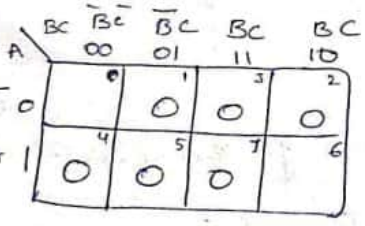




OR-NAND logic



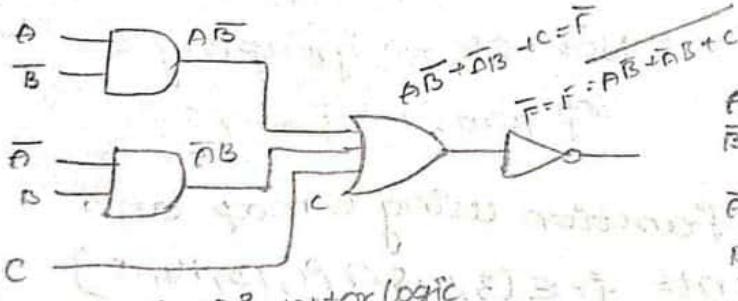
NOR-OR logic



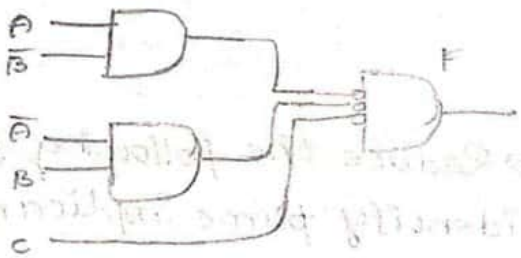
(0's grouping SOP form)

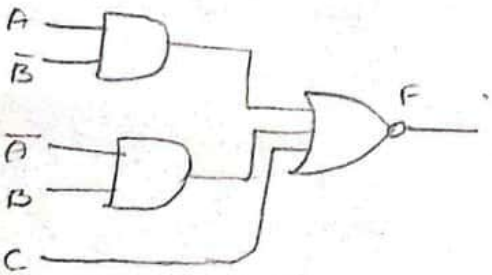
$\overline{F} = C + A\overline{B} + \overline{A}B$

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

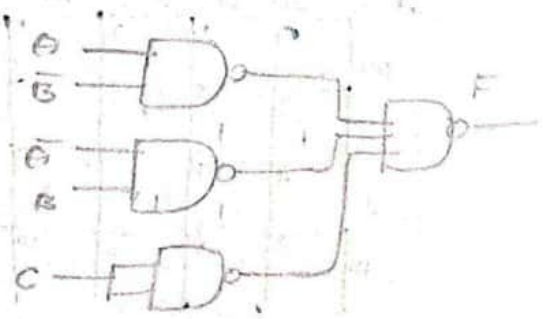


AND-OR-Inverter logic



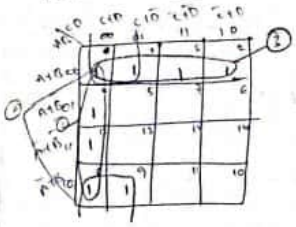


AND-OR logic



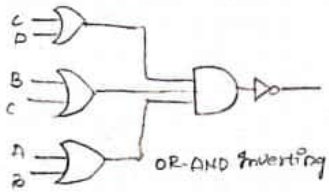
NAND-AND logic

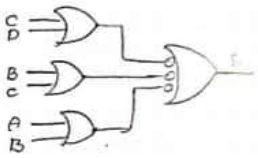
$f(A,B,C,D) = \sum m(0,1,2,3,4,8,9,12)$

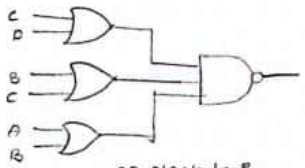


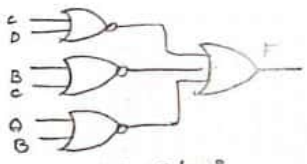
$\bar{F} = (C+D)(B+C)(A+B)$

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







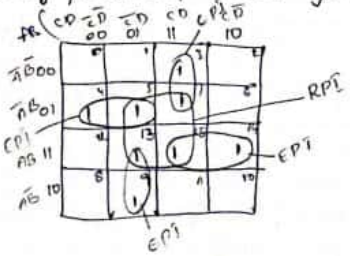


Note:-

SOP → AOI → AND-NOR → NAND-AND → grouping is in pos form → F → F

POS → OAI → OR-AND → NOR-OR → grouping 0's in sop form → $\bar{F}$ → F

→ Reduce the following function using k map and identify prime implicants $f = \Sigma(3,4,5,7,9,13,14,15)$



Prime Implicant:

Each group (pair, quad, octant) made up of bunch of adjacent minterms is called a prime implicant.

essential prime implicant: The prime implicant which contains at least one minterm which is not covered by any other prime implicant is called essential prime implicant

Redundant prime implicant:-

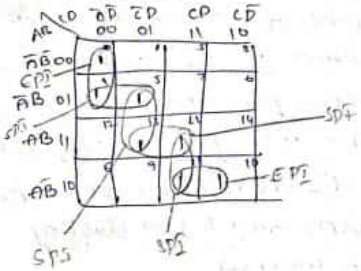
The prime implicant whose each (all) minterm is covered at least by one essential prime implicant is called Redundant prime implicant

Selective prime implicants:

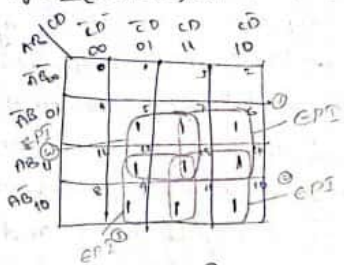
A prime implicant which is neither an essential prime implicant nor a redundant prime implicant is called selective prime implicant

→ Reduce the following function using Kmap and identify the prime implicants EPI's, EPI's, RPI's

for $f = \Sigma(0,4,5,10,11,13,15)$



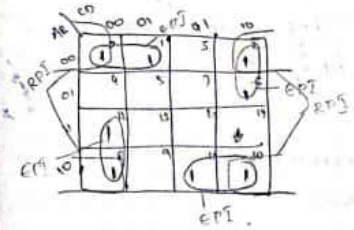
→ $f = \Sigma(5,6,7,9,10,11,13,14,15)$



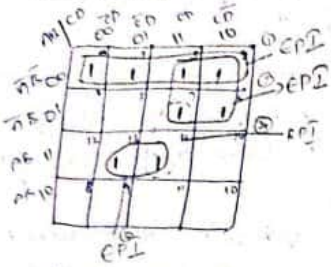
$f = ① + ② + ③ + ④$

$= BC + AC + BD + AD$

→ $\Sigma(0,1,2,6,12,8,10,11)$



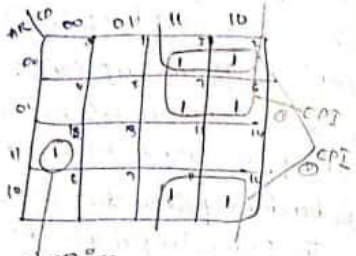
→ $\Sigma(0,1,2,3,6,7,13,15)$



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

EPI's

→ $\Sigma(2,3,6,7,10,11,12)$



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

Limitations / disadvantages of kmaps.

* K maps are extensively used for minimization of functions. It is comfortable with (3,4,5,6) variables only
* It is almost impossible to go with functions of equal to 7 or more than 7 variables.
* As the no. of variable increases it becomes difficult to make judgement about which combination forms all the minterms for minimization.
* K map is purely manual technique and it heavily depends on humans ability in simplification
* To overcome the above disadvantages, a method is developed W.V and Quine and E.J.M clusky and also called as Tabular method.
* It is used for simplification of 4,5,6,7.
* Tabular method is very much suitable for minimization of function by means of manual technique or machine technique.
* The Basic principle used in tabular method is combining law. $PA + P\bar{A} = P(A+\bar{A}) = P$

  where P is set of literals

Index:

* Index of any minterm is defined as no. of (ones)'s in its binary code for SOP form
* Index of any maxterm is defined as no. of 0's in its binary code for POS form.

Minterm	Binary code	Index
m_0	0000	0
m_1	0001	1
m_2	0010	1
m_3	0011	2
m_4	0100	1
m_5	0101	2
m_6	0110	2
m_7	0111	3
m_8	1000	1
m_9	1001	2
m_{10}	1010	2
m_{11}	1011	3
m_{12}	1100	2
m_{13}	1101	3
m_{14}	1110	3
m_{15}	1111	4

→ f(A,B,C,D)

max terms	Binary code	Index
M_0	0000	4
M_1	0001	3
M_2	0010	3
M_3	0011	2
M_4	0100	3
M_5	0101	2
M_6	0110	2
M_7	0111	1
M_8	1000	3
M_9	1001	2
M_{10}	1010	2
M_{11}	1011	1
M_{12}	1100	2
M_{13}	1101	1
M_{14}	1110	1
M_{15}	1111	0

- Simplify the following function by using tabular method $f(A,B,C,D) = \Sigma(1,2,3,5,6,7,8,9,12,13,15)$

Index	minterms	Pair (difference)	Quad (2 diff's)
Index 1	1	(1,3) (2)	1,3,5,7 (2,4) →
	2	(1,5) (4)	1,3,9,11 (4,8) →
	8	(1,9) (8)	(2,3,6,7) (1,4) → R
Index 2	3	(2,3) (1)	8,9,12,13 (1,4) → S
	5	(2,6) (4)	5,7,13,15 (2,8) → T
	6	(8,9) (1)	
	9	(8,12) (4)	
	12		
Index 3	7	(3,7) (4)	
	13	(5,7) (2)	
Index 4	15	(5,13) (8)	
		(6,7) (1)	
		(9,13) (4)	
		(12,13) (1)	
		(7,15) (8)	
		(13,15) (2)	

minimum number should be taken as A B C D binary code
1, 8 4 2 1

(2,6) P → 0 ø ø 1 → A B C D → 0 – – 1 → $\bar{A}D$

This should be eliminated

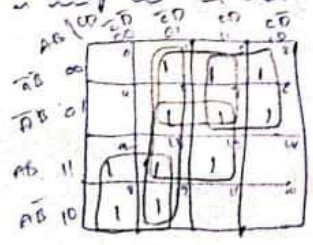
(4,8) → ø ø 0 1 → – – 0 1 → $\bar{C}D$

(4,6) R → 0 ø 1 ø → 0 – 1 – → $A\bar{C}$

(1,4) S → 1 ø 0 ø → 1 – 0 – → $A\bar{C}$

(2,8) T → ø 1 ø 1 → – 1 – 1 → BD

To verify draw k maps



$\bar{A}C + A\bar{C} + BD \rightarrow \bar{C}D + \bar{A}D$

PI's	PI	minterms	1	2	3	5	6	7	8	9	12	13	15
EPI's	P	(1,3,5,7)	x		x	x		x					
EPI's	S	(1,5,9,13)	x			x				x		x	
	R	(2,3,6,7)		x	x		x	x					
	Q	(8,9,12,13)						x	x	x	x	x	
	T	5,7,13,15				x		x				x	x

EPI's

$\therefore f = R + S + T + P = BC + A\bar{C} + BD + \bar{A}D$

(or)

$R + S + T + Q = \bar{A}C + A\bar{C} + BP + \bar{C}D$

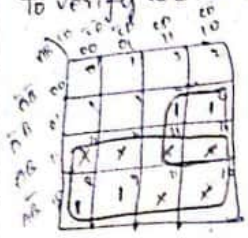
$\rightarrow f(A,B,C,D) = \sum m(6,7,8,9) + d(10,11,12,13,14,15)$

Index	min terms	Pairs (D)	Quad (dif)	octant (dif)
Index 1	8 ✓	(8,9) (1) ✓	8,9,10,11 (1,2) ✓	(8,9,10,11 12,13,14,15 (1,2,4) → P
		(8,10) (2) ✓	8,9,12,13 (1,4) ✓	
Index 2	6 ✓	(8,12) (4) ✓	8,10,12,14 (2,4) ✓	
	9 ✓	(6,7) (1) ✓	6,7,14,15 (1,8) → Q	
	10 ✓	(6,14) (8) ✓	9,11,13,15 (2,4)	
	12 ✓	(9,13) (4) ✓	12,14,13,15 (1,2)	
Index 3	7	(9,11) (2) ✓	10,11,14,15 (1,4)	
	13	(10,14) (4) ✓		
	11	(10,11) (1) ✓		
Index 4	15	(12,13) (1) ✓		
		(12,14) (2) ✓		
		(7,15) (8) ✓		
		(13,15) (2) ✓		
		(11,15) (4) ✓		

(8,9,10,11,12,13,14,15) (1,2,4) → P → 1 ø ø ø → 1 – – – → A

(6,7,14,15) (1,8) → Q → ø 1 1 ø → – 1 1 – → BC

To verify we use K maps



$f(A,B,C,D) = A + BC$

→ Minimize the following function using tabular method.

$f(A,B,C,D) = \Sigma m(2,3,5,7,8,10,12,13)$

Index	min terms	Pairs(D)	Quad → No Guards
Index 1	2	(2,3)(1) → P	0010 → $\bar{A}\bar{B}C$
	8	(2,10)(8) → Q	0010 → $\bar{B}C\bar{D}$
Index 2	3	(8,10)(2) → R	1000 → $A\bar{B}\bar{D}$
	5	(8,12)(4) → S	1000 → $A\bar{C}\bar{D}$
	10	(3,7)(4) → T	0011 → $\bar{A}CD$
	12	(5,7)(2) → U	0101 → $\bar{A}BD$
Index 3	7	(5,13)(8) → V	0101 → $B\bar{C}D$
	13	(12,13)(1) → W	1100 → $AB\bar{C}$

Prime Implicant chart

↓PI / min terms →	2	3	5	7	8	10	12	13
P(2,3)	X	X						
Q(2,10)	X					X		
R(8,10)					X	X		
S(8,12)					X		X	
T(3,7)		X		X				
U(5,7)			X	X				
V(5,13)			X					X
W(12,13)							X	X

$f(A,B,C,D) = P + R + U + W$

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

→ $f(A,B,C,D) = \Sigma m(0,1,2,3,6,7,13,15)$

Index	min terms	Pairs	Quads
Index 0	0	(0,1)(1)	(0,1,2,3)(1,2) → P → 0000 → $\bar{A}\bar{B}$
		(0,2)(2)	
Index 1	1	(1,3)(2)	(2,3,6,7)(1,4) → S → 0010 → $\bar{A}C$
	2	(2,3)(1)	
Index 2	3	(2,6)(4)	
	6		
Index 3	7	(3,7)(4)	
	13	(6,7)(1)	
Index 4	15	(7,15)(8) → R → 0111 → BCD	
		(13,15)(2) → S → 1101 → ABD	

PI Chart

PI / min terms	0	1	2	3	6	7	13	15
P(0,1,2,3)	X	X	X	X				
S(2,3,6,7)			X	X	X	X		
R(7,15)						X		X
S(13,15)							X	X

$f = P + S + R = \bar{A}\bar{B} + \bar{A}C + ABD$

→ $f(A,B,C,D) = \Sigma m(0,1,3,7,8,9,11,15)$

Index	min terms	Pairs (D)	Quad (D)
Index 0	0	(0,1)(1)	(0,1,8,9)(1,8) → P → 0000 → $\bar{B}\bar{C}$
		(0,8)(8)	
		(1,9)(8)	
Index 1	1	(1,3)(2)	(1,3,9,11)(2,8) → S → 0001 → $\bar{B}D$
	8	(1,9)(8)	
Index 2	3	(3,7)(4)	(3,7,11,15)(4,8) → R → 0011 → CD
	9	(3,11)(8)	
Index 3	7	(7,15)(8)	
	11	(11,15)(4)	
Index 4	15		

PI/minterms	0	1	3	7	8	9	11	15
P(0,1,8,9)	x	x			x	x		
Q (1,3,9,11)		x	x			x	x	
R (3,7,11,15)			x	x			x	x

$f = P + Q = \bar{B}\bar{C} + CD$

→ Simplify the following using tabular method

$f(A,B,C,D) = \Sigma m(0,4,8,10,12,13,15) + d(1,2)$

Index	min terms	Pairs	Quads
Index 0	0,	(0,1)(1)→R	(0,2,8,10)(2,8) → P → 0000 $\bar{B}\bar{D}$
Index 1	1,4, 8	(0,2)(2)✓	(0,4,8,12)(4,8) → Q → 0000 → $\bar{C}\bar{D}$
		(0,4)(4)✓	
		(0,8)(8)✓	
Index 2	10	(2,10)(8)✓	(0,1)(1) → R → 0000 → $A\bar{B}\bar{C}$
	12	(8,10)(2)✓	(12,13)(1) → 1100 → $AB\bar{C}$
		(4,12)(8)✓	(13,15)(2) → T → 110 1 → ABD
Index 3	13	(8,12)(4)✓	
Index 4	15	(12,13)(1) → S	
		(13,15)(2) → T	

PI chart → Q, R, S, T should not be involved in this PI chart

PI/min terms	0	4	8	10	12	13	15
P (0,2,8,10)	x		x	x			
Q (0,4,8,12)	x	x	x		x		
R (0,1)	x						
S (12,13)					x	x	
T (13,15)						x	x

$f(A,B,C,D) = P + Q + T$

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

→ $f(A,B,C,D) = \Sigma(1,2,3,9,12,14,15) + d(4,8,11)$

Index	min term	pair	Quads
Index 0		(1,3)(2)✓	(1,3,9,11)(2,8) → P → 0001 → $\bar{B}\bar{D}$
Index 1	1	(1,9)(8)	Q → 0010 → $\bar{A}\bar{B}C$
	2	(2,3)(1) → Q	R → 0100 → $B\bar{C}\bar{D}$
	4	(4,12)(8) → R	S → 1000 → $A\bar{B}\bar{C}$
	8	(8,9)(1) → S	T → 1000 → $A\bar{C}\bar{D}$
Index 2	3	(8,12)(4) → T	U → 1100 → $AB\bar{D}$
	9	(11,9)(2)✓	V → 1110 → ABC
	12	(11,3)(8)✓	W → 1011 → ACD
Index 3	11	(12,14)(2) → U	
	14	(11,15)(4)✓	
Index 4	15	(14,15)(1) → V	
		(11,15)(4) → W	

PI chart

PI/min terms	1	2	3	9	12	14	15
P (1,3,9,11)	x		x	x			
Q (2,3)		x	x				
R (4,12)				x	x		
S (8,9)				x			
T (8,12)					x		
U (12,14)					x	x	
V (14,15)						x	x
W (11,15)							x

$f = P + Q + U + V = \bar{B}D + \bar{A}\bar{B}C + AB\bar{D} + ABC$

(OR)

$f = P + Q + U + W = \bar{B}D + \bar{A}\bar{B}C + AB\bar{D} + ACD$

Simplify the following function using Tabular method

$f(A,B,C,D,E) = \Sigma(4,6,7,9,11,12,13,14,15,20,22,25,27,28,30) + d(1,5,29,31)$

Index	minterms
Index 1	4
Index 2	5, 6, 9, 12, 20
Index 3	7, 11, 13, 14, 22, 25, 28
Index 4	15, 27, 29, 30
Index 5	31

Pairs
(1,5)(4)
(1,9)(8)
(4,5)(1)
(4,6)(2)
(4,12)(8)
(4,20)(16)
(5,7)(2)
(6,7)(1)
(9,11)(2)
(5,13)(8)
(9,13)(4)
(12,13)(1)
(6,14)(8)
(12,14)(2)
(6,22)(16)
(9,25)(16)
(20,22)(2)
(12,28)(16)
(20,28)(8)
(7,15)(8)
(11,15)(4)
(13,15)(2)
(13,15)(1)
(11,27)(16)
(25,27)(2)
(13,29)(16)
(25,29)(4)
(28,29)(1)
(14,30)(16)
(22,30)(8)
(28,30)(2)
(29,31)(2)
(15,31)(16)
(30,31)(1)
(27,31)(4)

Quads
(1,5,9,13)(4,8)
(4,5,12,13)(1,8)
(4,6,12,14)(2,8)
(4,12,20,28)(8,16)
(4,5,6,7)(1,2)
(4,6,20,22)(2,16)
(6,7,14,15)(1,8)
(12,13,14,15)(1,2)
(12,13,28,29)(1,16)
(5,7,13,15)(2,8)
(9,11,13,15)(2,4)
(9,11,25,27)(2,16)
(12,14,28,30)(2,16)
(20,22,28,30)(2,8)
(9,13,25,29)(4,16)
(6,14,22,30)(8,16)
(20,22,28,30)
(14,15,30,31)(1,16)
(28,29,30,31)(1,2)
(13,15,29,31)(2,16)
~~(11,27,29,31)(2,4)~~
(11,15,27,31)(4,16)
(25,27,29,31)(2,4)

octanis:-

1,5,6,7,12,13,14,15 (1,2,8) → P → 0 0 1 0 0 → $\bar{A}C$

4,6,12,14,20,22,28,30 (2,8,16) → Q → 0 1 0 0 0 → $C\bar{E}$

9,11,13,15,25,27,29,31 (2,4,16) → R → 0 1 0 0 1 → BE

12,13,14,15,28,29,30,31 (1,2,16) → S → 0 1 1 0 0 → BC

(1,5,9,13)(4,8) → T → 0 0 0 0 1 → $\bar{A}\bar{D}E$

PI/prime terms: 4 / 6 / 7 / 9 / 11 / 12 / 13 / 14 / 15 / 20 / 22 / 25 / 27 / 28 / 30

PI/minterms	4	6	7	9	11	12	13	14	15	20	22	25	27	28	30
*P	x	x	x			x	x	x	x						
*Q	x	x				x		x		x	x			x	x
*R				x	x		x		x			x	x		
S						x	x	x	x					x	x
T				x			x								

$f(A,B,C,D,E) = P + Q + R$

$= \bar{A}C + C\bar{E} + BE$

→ f(A,B,C,D,E) = Σ(0,1,2,8,9,15,17,21,24,25,27,31)

Index	min terms	Pairs	Quads
Index 0	0 ✓	(0,1)(1) ✓	(0,1,8,9)(1,8) → P → 0000ϕ → $\bar{A}\bar{B}\bar{C}$
		(0,2)(2)	
Index 1	1 ✓	(0,8)(8) ✓	(8,9,24,25)(1,16) → Q → a000ϕ → BCD
	2 ✓		
	8 ✓	(1,9)(8) ✓	(1,9,17,25)(8,16) → R → 0000 1 → $\bar{C}DE$
		(1,17)(16) ✓	
Index 2	9 ✓	(8,9)(1) ✓	(0,2)(2) → S → 00000 → $\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}$
	17 ✓		
	24 ✓	(8,24)(16) ✓	(17,21)(4) → T → 10ϕ0 1 → $A\bar{B}\bar{C}DE$
			(25,27)(2) → U → 110ϕ 1 → $AB\bar{C}\bar{D}E$
Index 3	21 ✓	(9,25)(16) ✓	(15,31)(16) → V → 0111 1 → BCDE
	25 ✓	(17,21)(4)	(27,31)(4) → W → 11ϕ11 → ABDE
Index 4	15	(24,25)(1) ✓	
	27	(17,25)(8) ✓	
Index 5	31	(25,27)(2)	
		(15,31)(16)	
		(27,31)(4)	

PI / minterms : 0 1 2 8 9 15 17 21 24 25 27 31

P(0,1,8,9) x x x x

Q(1,9,17,25) x x x x

R(8,9,24,25) x x x x

S(0,2) x x

T(17,21) x x

U(25,27) x x

V(15,31) x x

W(27,31) x x

f = R + S + T + V + P + U

(or)

R + S + T + V + Q + W

(or)

R + S + T + V + Q + U

(or)

R + S + T + V + P + W

Tabular method for pos form:

→ Assume max terms = Min terms

→ write same terms for prime implicants

→ Get final expression in pos-form

Simplify the following function

f = Π m(2,3,8,12,13) into don't cares (10,14)

Index	max terms	Pairs	Quads
Index 1	2	(2,3)(1) → Q	(2,10,12,14)(2,4) → P → 1 0 0 0 → $\bar{A}+D$
	8	(2,10)(8) → R	Q → 0010 → $A+B+\bar{C}$
Index 2	3	(8,10)(2) ✓	R → 0010 → $B+\bar{C}+D$
	10	(8,12)(4) ✓	S → 1100 → $\bar{A}+\bar{B}+C$
	12	(10,14)(4) ✓	
Index 3	13	(12,13)(1) → S	
	14	(12,14)(2) ✓	

PI / maxterms	8	3	8	12	13
* P(8,10,12,14)			[X]	X	X
* Q(2,3)	X	[X]			
R(2,10)	X				
* S(12,13)				X	[X]

f = P · Q · S

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

Find the minimal expression in pos form

f = (0,1,4,5,9,11,13,15,16,17,25,27,28,29,31) + d(20,21,22,30).

COMBINATIONAL LOGIC CIRCUITS

→ When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuit is called "combinational logic."

→ A combinational logic circuit is a circuit whose output value depends upon the present input value.

→ The below figure shows the block diagram of a combinational circuit.

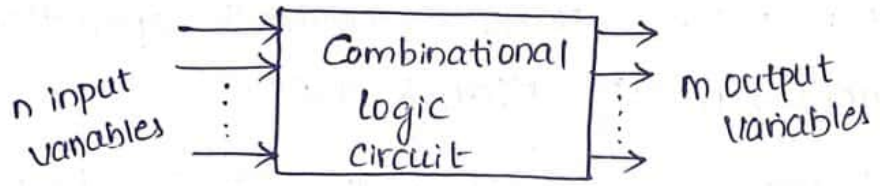
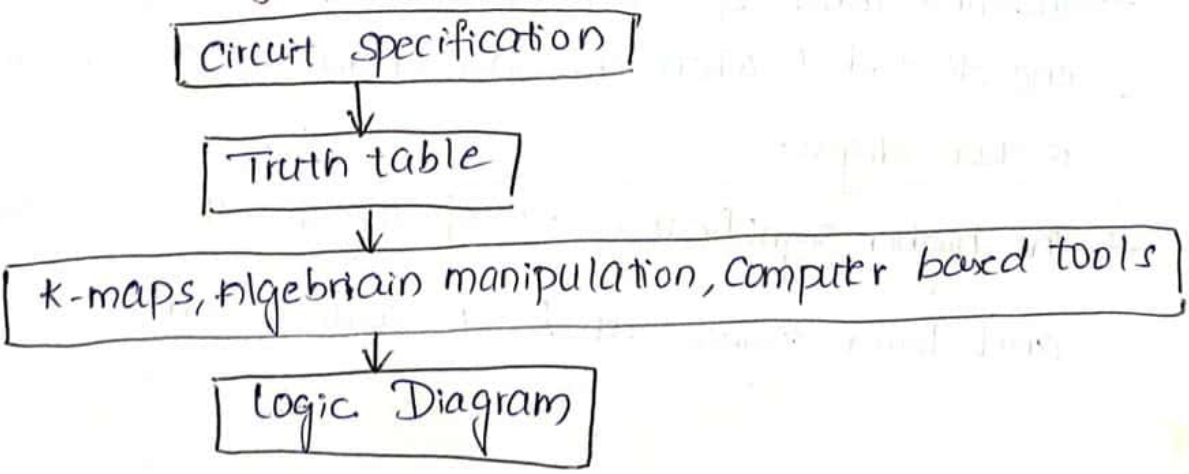


Figure : Block diagram of a combinational circuit

→ As shown in figure, the combinational circuit accepts n-input binary variables and generates m-output variables depending on the logical combination of gates.

Analysis and Design of Combinational Circuits:



UNIT-IV
Sequential circuits

A sequential circuit consists of a combinational circuit to which storage elements are connected to form feedback path. The block diagram of sequential circuit demonstrates that the outputs in a sequential circuit are a function not only of the inputs, but also of the present state of the storage elements.

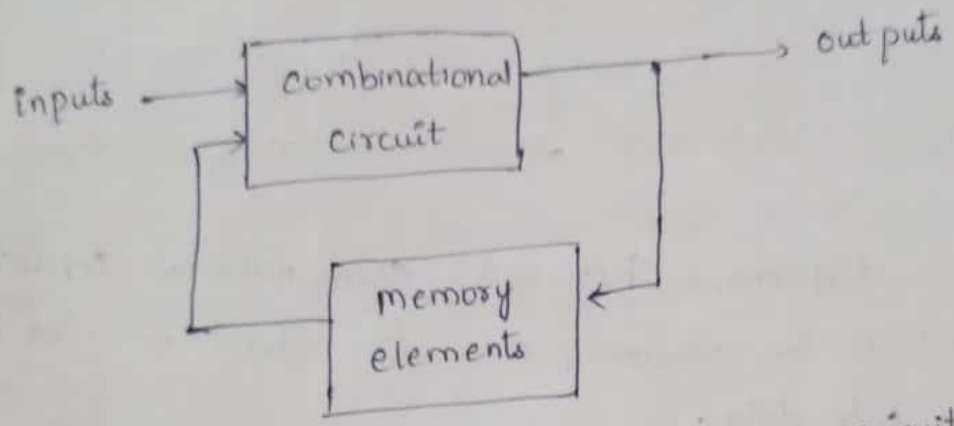


fig:- Block diagram of a sequential circuit.

Difference between combinational & sequential circuits:-

combinational circuits	Sequential circuits
1. In combinational circuits, the output variables are at all time dependent on the combination of input variables.	1. In sequential circuits, the output variables depend not only on the present input variables but they also depend upon the past history of these input variables.
2. Memory unit is not required in combinational circuits.	2. memory unit is required to store the past history of input variables in sequential circuit.

UNIT-IV
Sequential circuits

A sequential circuit consists of a combinational circuit to which storage elements are connected to form feedback path. The block diagram of sequential circuit demonstrates that the outputs in a sequential circuit are a function not only of the inputs, but also of the present state of the storage elements.

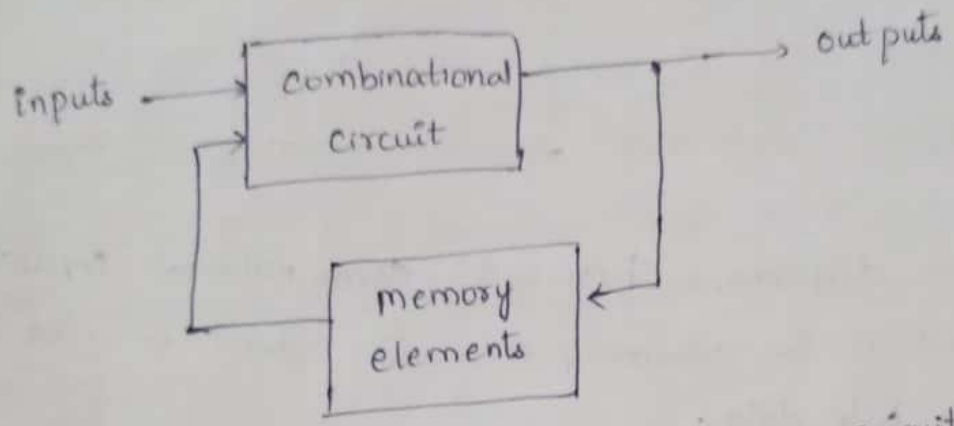


fig:- Block diagram of a sequential circuit.

Difference between combinational & sequential circuits:-

combinational circuits	Sequential circuits
1. In combinational circuits, the output variables are at all time dependent on the combination of input variables.	1. In sequential circuits, the output variables depend not only on the present input variables but they also depend upon the past history of these input variables.
2. Memory unit is not required in combinational circuits.	2. memory unit is required to store the past history of input variables in sequential circuit.

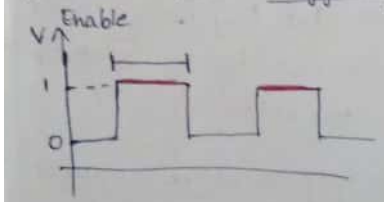
3. Combinational circuits are faster in speed.	3. Slower than the combi. circuits
4. Easy to design	4. Comparatively harder to design
5. parallel adder is a combinational circuit	5. Serial adder is a sequential circuit.

The difference between combinational circuit & sequential circuit is the memory elements which is used to store the binary bits data.

Latches & Flipflops (FF's) are the basic building blocks of the sequential circuits.

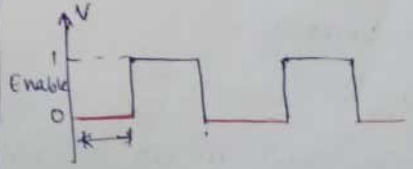
Latch:- Storage elements that operate with signal levels ie latches are controlled by enable signal and they are level triggered.

positive level triggering

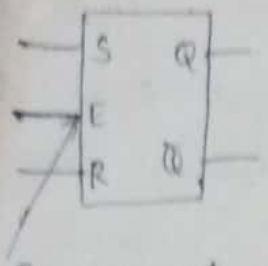


→ The output of latch responds to the input changes only when enable input is 1 (high).

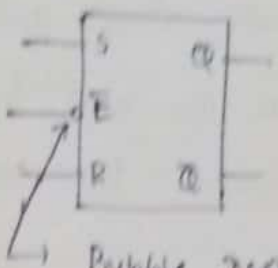
Negative level triggering



→ The output of latch responds to the input changes only when its enable input is 0 (low).



It represents positive level triggering



Bubble represents negative level triggering.

1 bit memory element :-

(S-R latch) :- SR latch is a circuit with two cross coupled NOR gate or two cross-coupled NAND gate, and two inputs labeled S for set & R for Reset state. Outputs Q & Q' are normally complement of each other.

When

S=0, R=1 then Q=0; $\bar{Q}$=1 (Reset state)

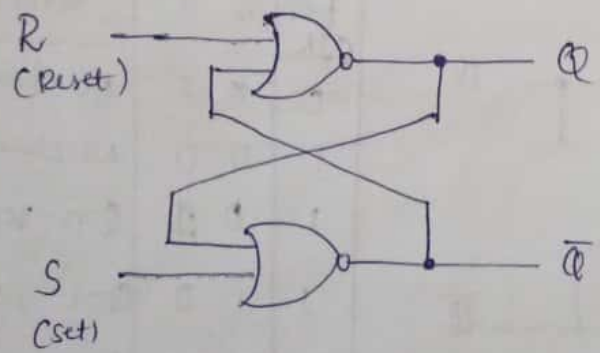
S=0, R=0 then Q=0; $\bar{Q}$=1 (previous same as previous outputs)

S=1, R=0 then Q=1; $\bar{Q}$=0 (Set state)

S=0, R=0 then Q=1; $\bar{Q}$=0 (same as previous outputs)

and hence S=0 & R=0 acts as memory state or NO change state.

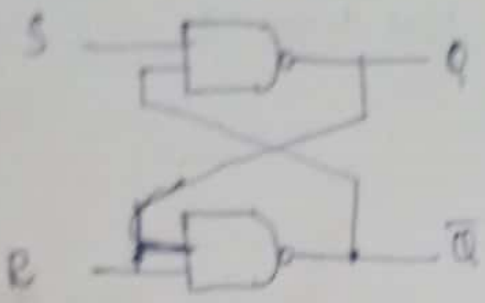
S=1, R=1 both Q=0 & $\bar{Q}$=0 (Indeterminate state)



a) Logic diagram.

S	R	Q
0	0	NO change/memory
0	1	0
1	0	1
1	1	Invalid

SR latch with NAND gates act as SR latch. In the NAND with the NOR latch, note that the input signals the NAND require the complement of those values used by NOR latch.



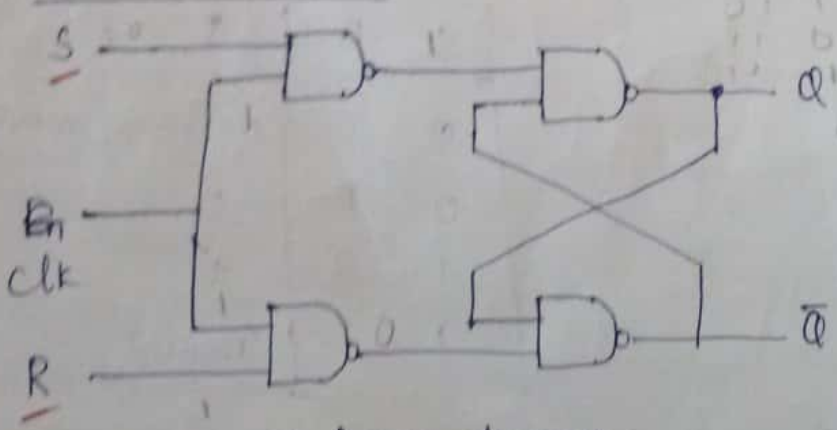
a) Logic diagram of SR latch.

S	R	Q
0	0	Invalid.
0	1	Set
1	0	Reset
1	1	No change/memory

The operation of the basic SR latch can be modified by providing an additional input signal that controls when the state of the latch can be changed by determining whether S & R can affect the circuit.

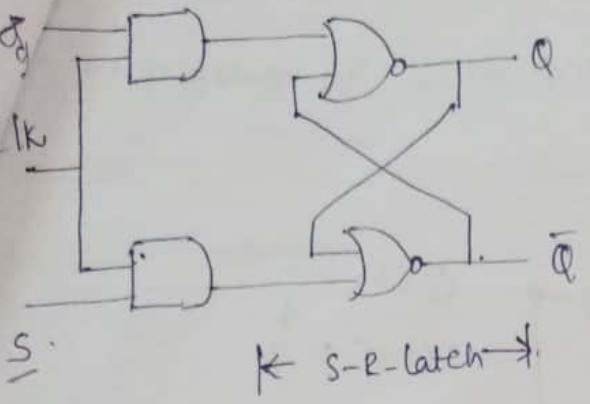
An SR latch with a control input is shown in fig below. It consists of the basic SR latch and two additional NAND gates. The SR latch with NAND gates a These two additional NAND gates converts the $\bar{S}\bar{R}$ latch as SR latch. and circuit will operate only when Enable input is high.

Clocked SR FF:-



a) Logic diagram

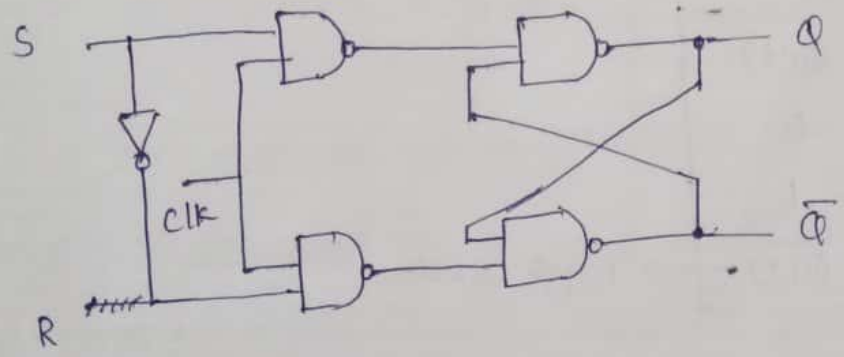
En Clk	S	R	Q_{n+1}
0	x	x	No change
1	0	0	No change.
1	0	1	Q=0 reset
1	1	0	Q=1 set
1	1	1	Invalid

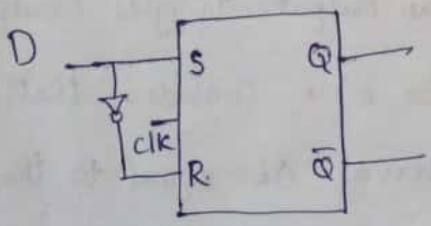


← S-R-latch →

Truth Instead of giving two seperate inputs for S & R by connecting an inverter to clocked S-R FF it acts as D-FF. whatever input is present at D is reflected at output. D-FF is also called as Delay FF/ Data FF.

D-FF:-



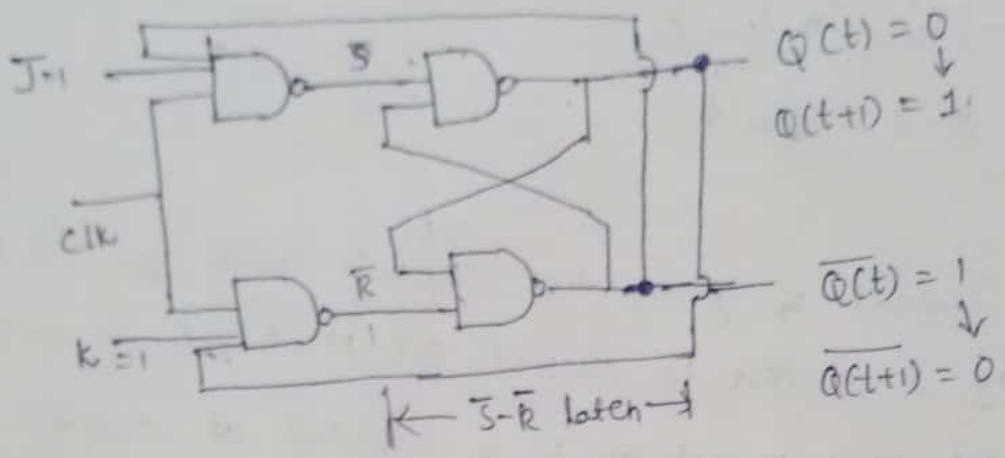


Truth table

D	Q(t+1)
0	0
1	1

Clocked JK flipflop:-

S=1 ; R=1 is the unused state. To make this useful state we go for JK-FF.



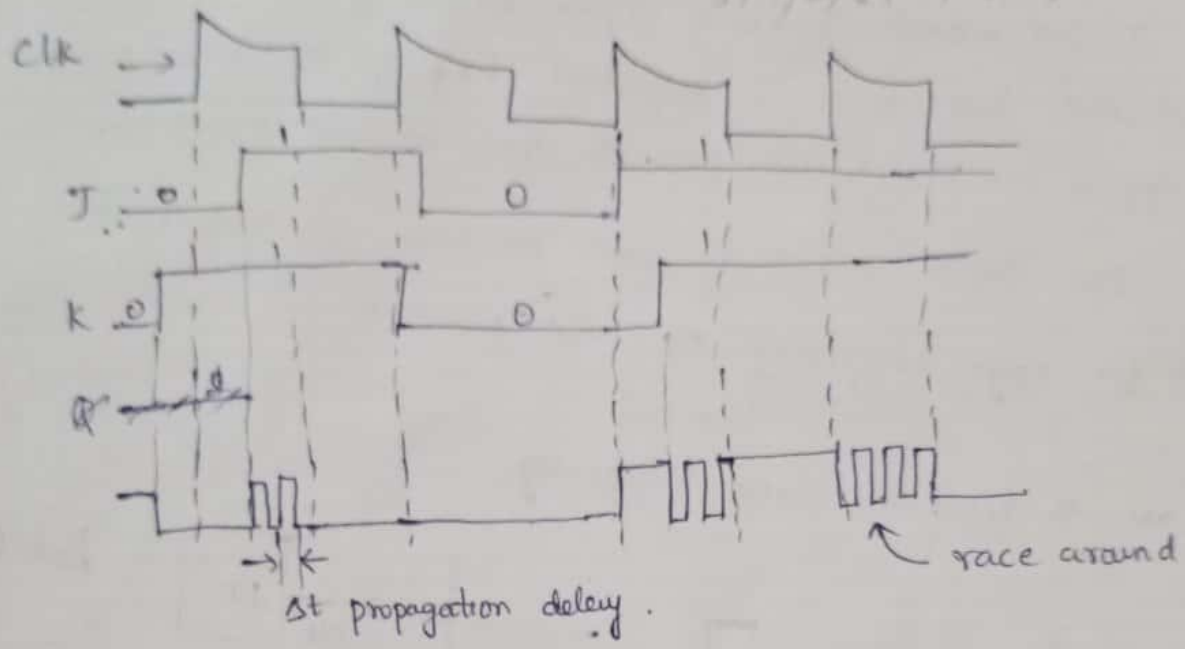
when J=1 & K=1 the output toggles from 0→1 or 1→0. The truth table for JK-FF is

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$ → Toggle mode.

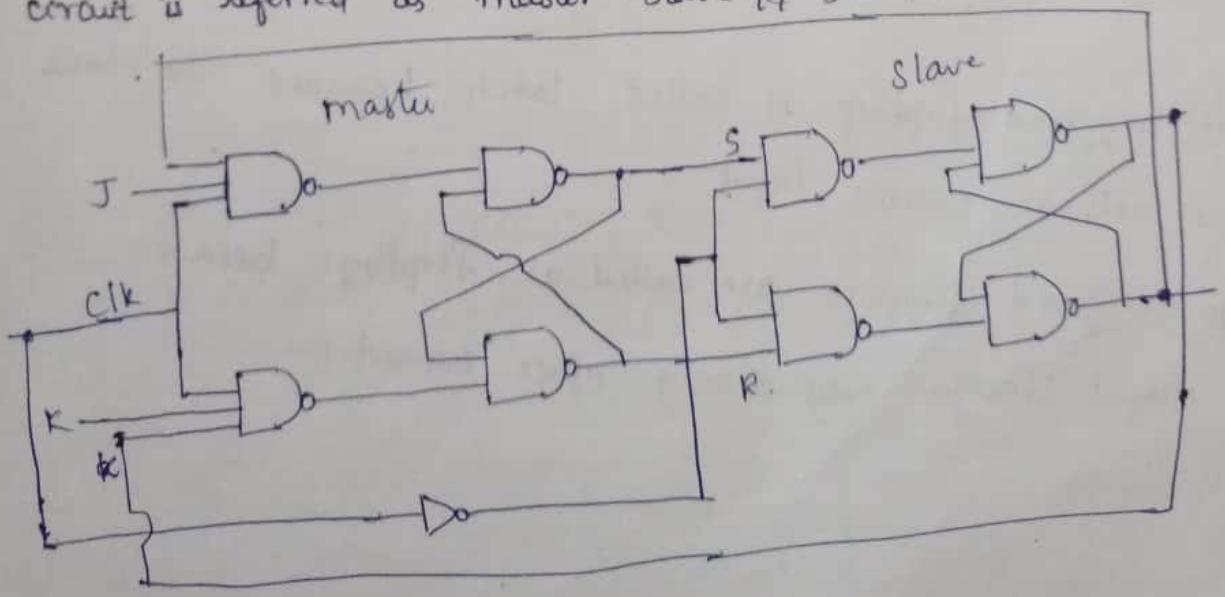
Race around condition:-

In J-K FF when J=1, K=1, the output toggles (output changes either from 0 to 1 or from 1 to 0). Consider that initially Q=0 & J=K=1. After a time interval Δt equal to the propagation delay through two NAND gate in series, the output will change to Q=1 and after another time interval Δt the output will change back to Q=0. This toggling

This toggling will continue until the FF is enabled and J=K=1. At the end of clock pulse the FF is disabled and the value of Q is uncertain. This situation is referred as race-around condition.



To eliminate racing in JK-FF we use master-slave JK-FF. master-slave FF is constructed from two FF's. One circuit serves as master and the other as slave and overall circuit is referred as master-slave JK FF



As shown in figure, clock signal is connected d.t to master flipflop, but it is connected through inverter to slave FF.

When J=1 & K=1 master toggles on the positive clock and slave then copies the output of master on the negative clock. At this instant, feedback inputs to the master FF are complemented but as it is negative half of the clock pulse master FF is inactive. This prevents race around condition.

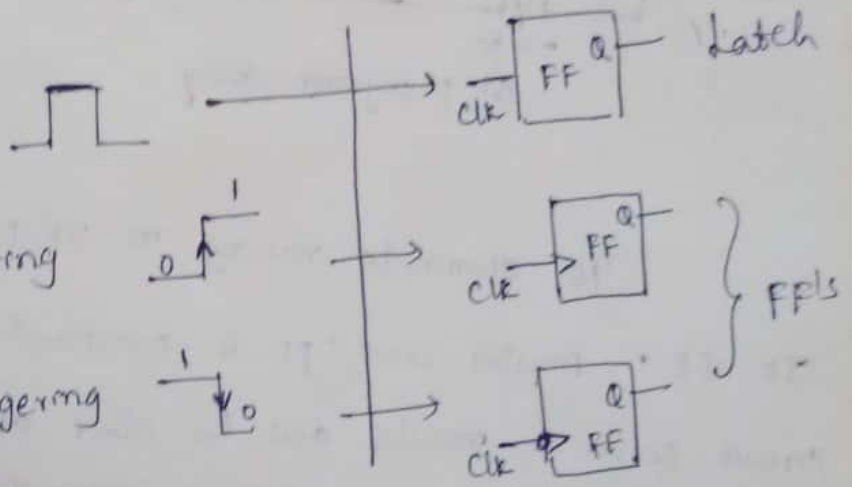
But circuit complexity will increase to reduce this we go for edge triggering of clock pulses.

There are 3 types of clock triggering

1) Level triggering

2) positive edge triggering

3) Negative edge triggering



* Level triggered flipflop is called latch because the clock input acts as enable input
* Edge triggered flipflops are called as flipflops because the output changes only during clock transitions.

R FF

S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Invalid

NO change

Truth table

Characteristic table

Inputs S	R	Present State Q(t)	Next State Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

Characteristic equation

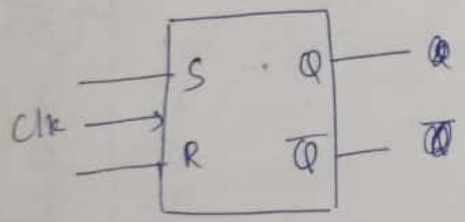
Q(t+1)

S \ R Q(t)	00	01	11	10
0	0	1	0	0
1	1	1	x	x

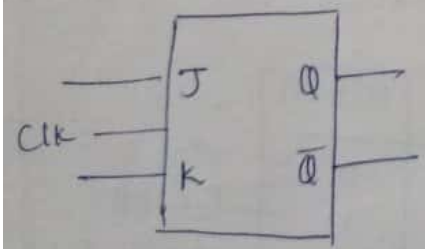
$Q(t+1) = S + \bar{R}\,Q(t)$

Excitation table

Present State Q(t)	Next State Q(t+1)	Inputs S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0



Clocked J-K FF



Truth table

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

Characteristic table

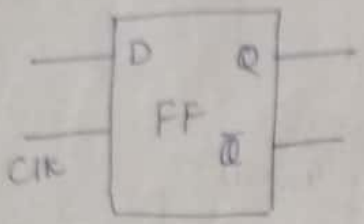
J	K	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

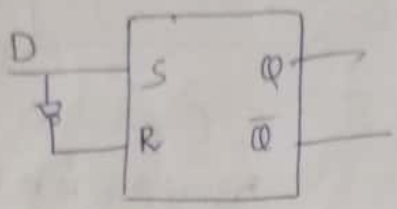
J \ Q(t)	00	01	11	10
0	0	1	0	0
1	1	1	0	1

$Q(t+1) = J.\overline{Q(t)} + \overline{K}Q(t)$

Q(t)	Q(t+1)	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Clocked D-FF





Truth table

D	Q(t+1)
0	0
1	1

Characterstic table

D	Q(t)	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

Characterstic equation

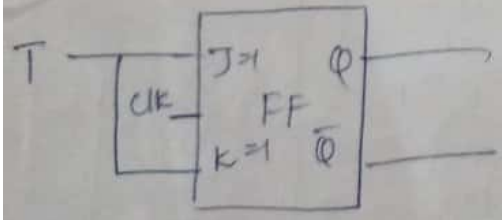
D \ Q(t)	0	1
0	0	0
1	1	1

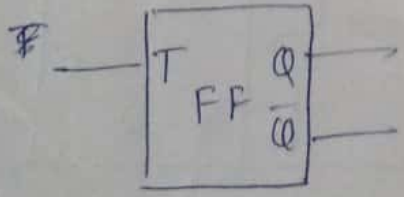
$Q(t+1) = D$

Excitation table

Q	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Clocked T-FF





Truth table

T	Q(t+1)
0	Q(t)
1	$\overline{Q(t)}$

Char. table

T	Q(t)	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

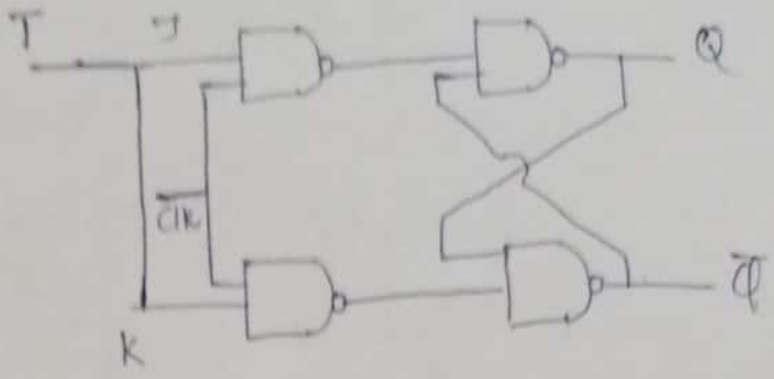
— Ex-OR truth table

$Q(t+1) = T\overline{Q(t)} + \overline{T}Q(t)$

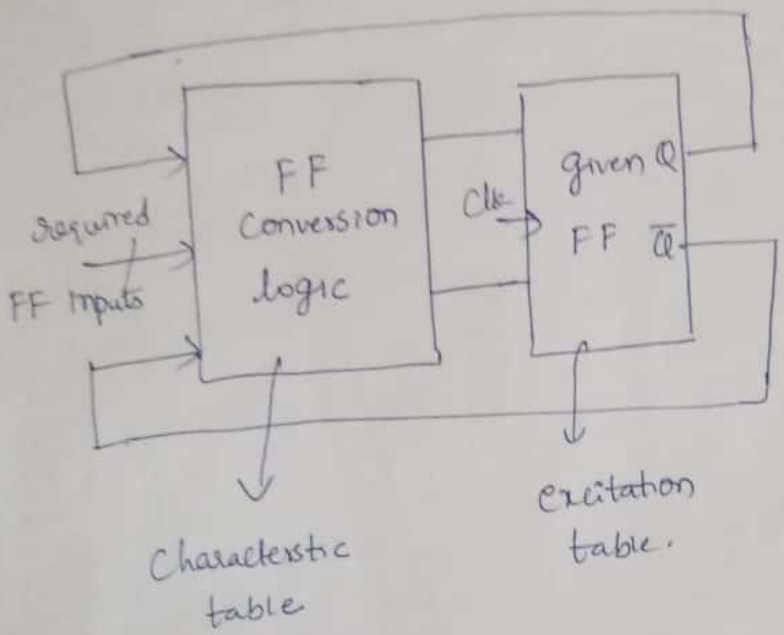
Excitation table

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Clocked T-FF



Conversion of Flipflops



S-R FF to T-FF $S = T \cdot \overline{Q(t)}$ $R = T \cdot Q(t)$

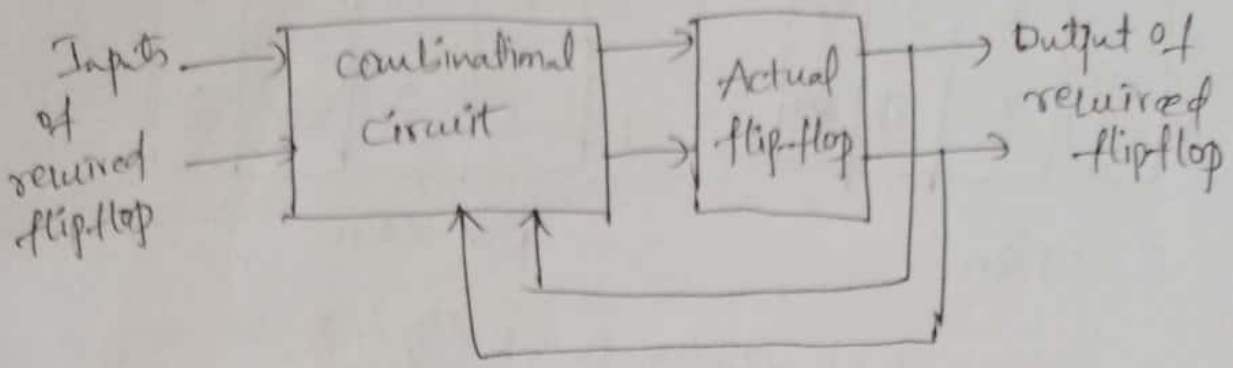
SR to Jk : $S = J\bar{Q}_n$ $R = KQ$

Jk to SR : $J = S$ $K = R$

Jk to D : $J = D$ $K = \bar{D}$

D to Jk : $D = J\bar{Q} + \bar{K}Q$.

Conversion of flip-flops:-



Block diagram for conversion of flip-flops

F/F	SR	D	JK	T
SR FF	—	✓	✓	✓
D_{FF}	✓	—	✓	✓
JK FF	✓	✓	—	✓
T FF	✓	✓	✓	—

Convert SR flip-flop into JK flip-flop:—

Conversion table

Characteristic table of required flip flop(JK)

J	K	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Excitation table of SR

S	R
0	X
X	0
0	X
0	1
1	0
X	0
1	0
0	1

S

J \ Qn	00	01	11	10
0	0	X	0	0
1	1	X	Q	Q

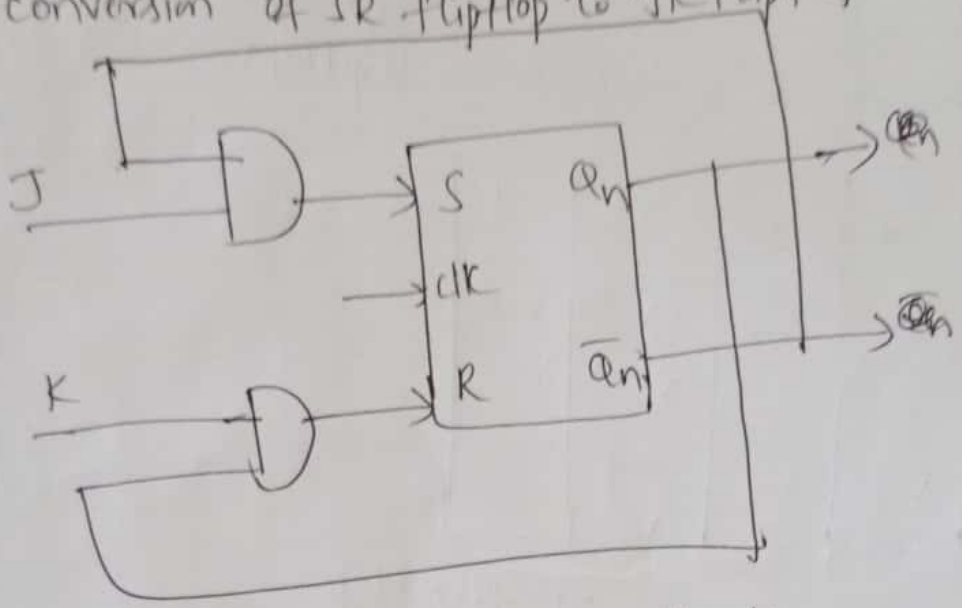
$S = J\overline{Q_n}$

R

J \ KQn	00	01	11	10
0	X	0	1	X
1	0	0	1	0

$R = KQ_n$

conversion of SR flipflop to JK flipflop



conversion of SR flipflop into D flipflop:-

D	Qn	Qn+1	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

S

D \ Qn	0	1
0	0	0
1	1	X

$S = D$

R

D \ Qn	0	1
0	X	1
1	0	0

$R = \overline{D}$

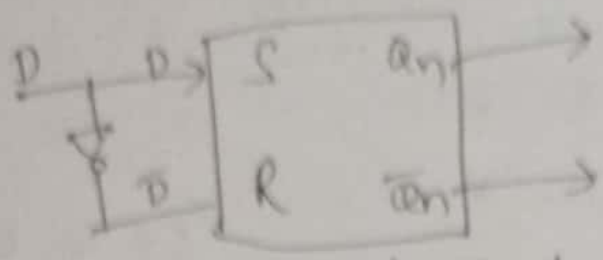
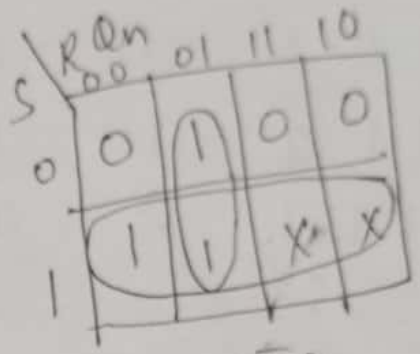


Fig: conversion of SR flip-flop to D-flip flop

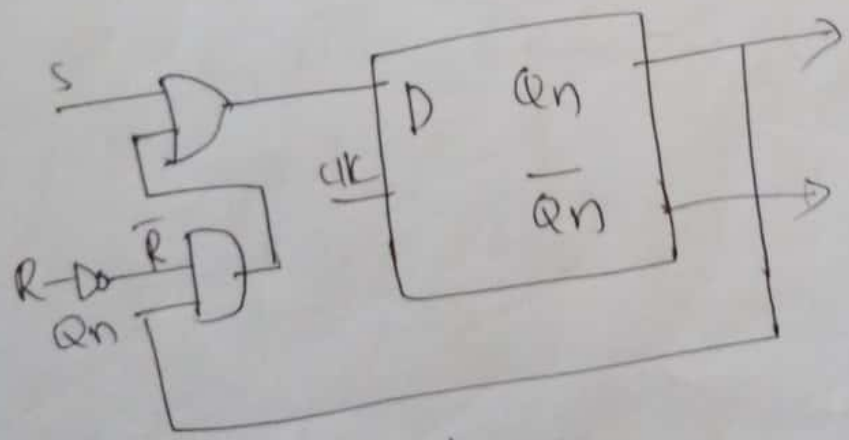
conversion of D-flip-flop to SR flip-flop :

I

S	R	Qn	Qn+1	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	X	X
1	1	1	X	X



$D = S + \bar{R} Qn$



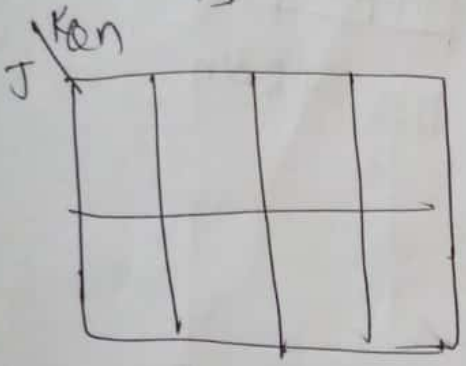
Logic diagram

Conversion of JK flipflop to SR flipflop:

JK flip	Qn	Qn+1	S	R
0 0	0	0	0	X
0 0	1	1	X	0
0 1	0	0	0	X
0 1	1	0	0	1
1 0	0	1	1	0
1 0	1	1	X	0
1 1	0	1	1	0
1 1	1	0	0	1

S R Qn Qn+1
0 0

S



Conversion of JK flipflop to SR flipflop:

S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	0	X	X	X
1	1	1	X	X	X

J

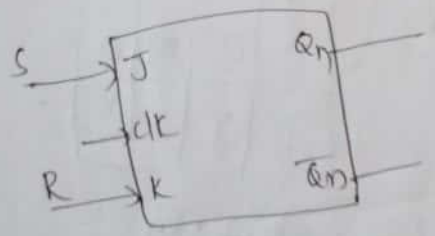
S \ RQ_n	00	01	11	10
0	0	X	X	0
1	1	X	X	X

K

S \ RQ_n	00	01	11	10
0	X	0	1	X
1	X	0	X	X

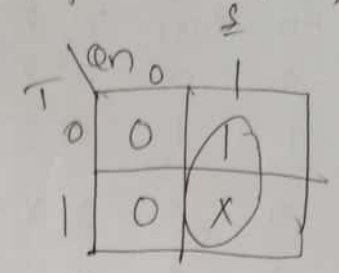
$K = R$

$J = S$

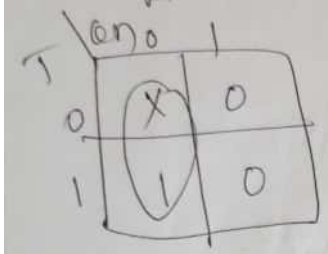


Conversion of SR flip flop to T flip-flop

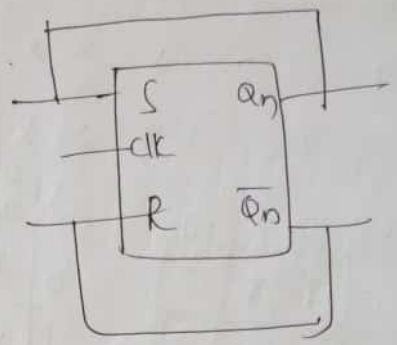
T	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	1	0
1	0	1	0	1
1	1	0	X	0



$S = Q_n$

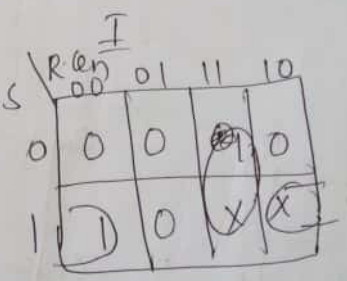


$R = \overline{Q_n}$

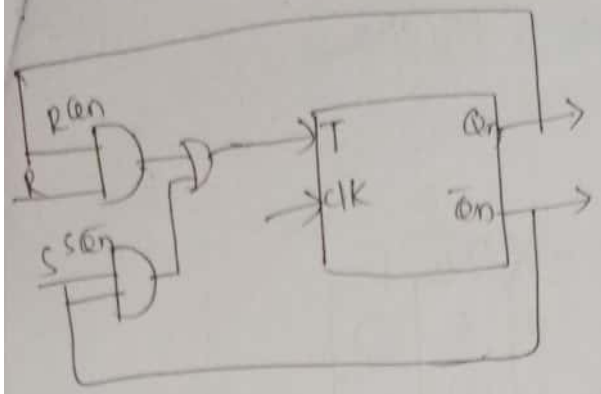


Conversion of T flip-flop to SR flip-flop

S	R	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	X	X
1	1	1	X	X



$T = RQ_n + S\overline{Q_n}$



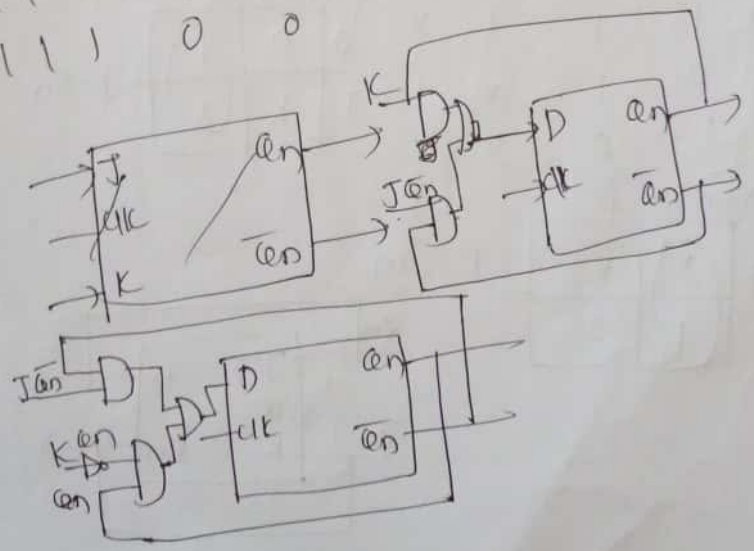
Convert D flip-flop to JK flip-flop:-

J	K	Qn	Qn+1	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

D

J \ KQn	00	01	11	10
0	0	1	0	0
1	1	1	0	1

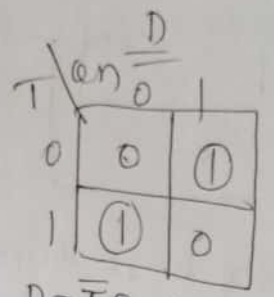
$D = J\bar{Q}n + \bar{K}Qn$



Convert D flip flop into T flip flop:-

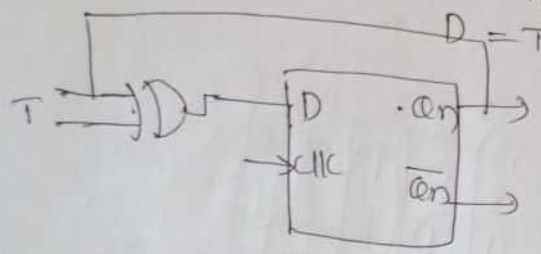
Conversion table:

T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0



$D = \bar{T}Q_n + T\bar{Q}_n$

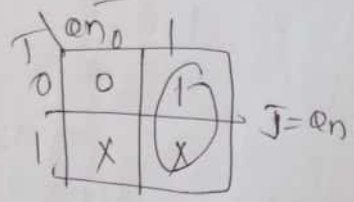
$D = T \oplus Q_n$



Conversion of JK flip flop to T flip flop:-

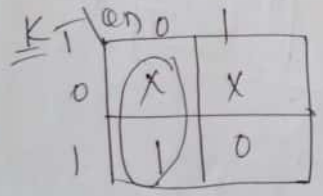
T	Q_n	Q_{n+1}	J	K
0	0	0	0	X
0	1	1	1	X
1	0	1	X	1
1	1	0	X	0

J

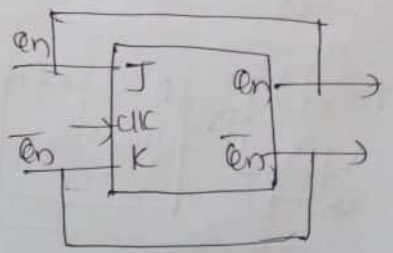


$J = Q_n$

K



$K = \bar{Q}_n$



Conversion of T flip-flop to SR flip-flop:

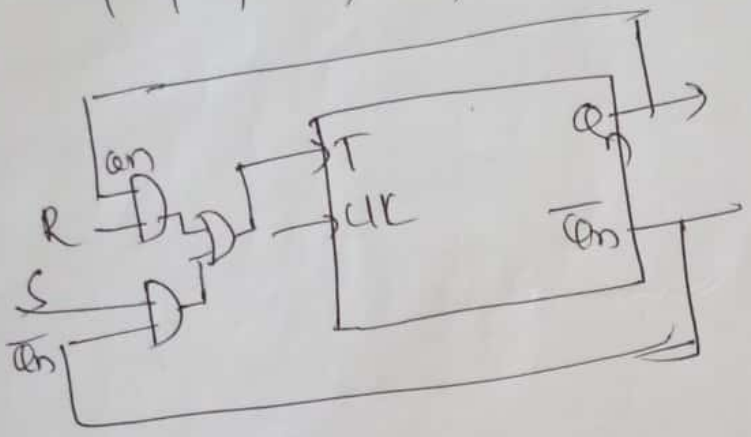
Conversion table

S	R	Qn	Qn+1	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	X	X
1	1	1	X	X

T

S \ RQn	00	01	11	10
0	0	0	1	0
1	1	0	X	X

$T = R Q_n + S \bar{Q}_n$



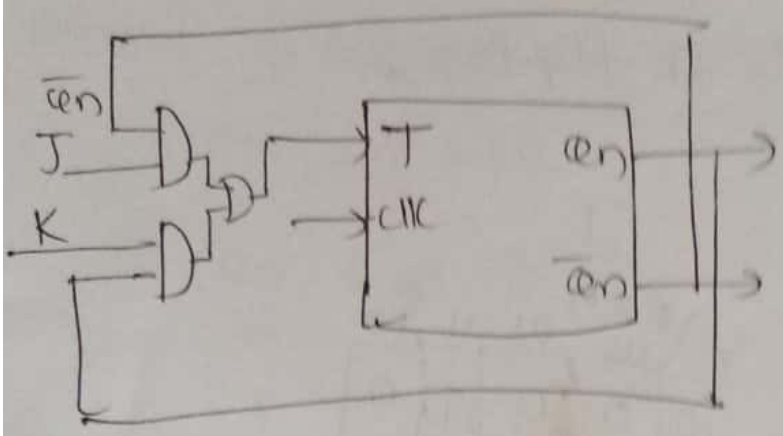
Conversion of T flip flop to JK flip flop:-

J	K	Qn	Qn+1	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

T

J \ KQn	00	01	11	10
0	0	0	1	0
1	1	0	1	1

$T = K Q_n + J \bar{Q}_n$



Propagation delay:-

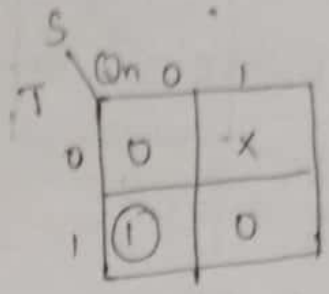
to T-FF conversion :-

SR given FF :- write excitation table of SR

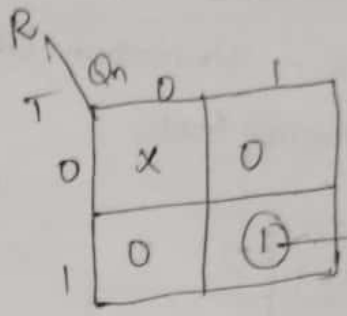
T-FF - Characterstic table.

T	Qn	Qn+1	S	R
0	0	0	0	x
0	1	1	x	0
1	0	1	1	0
1	1	0	0	1

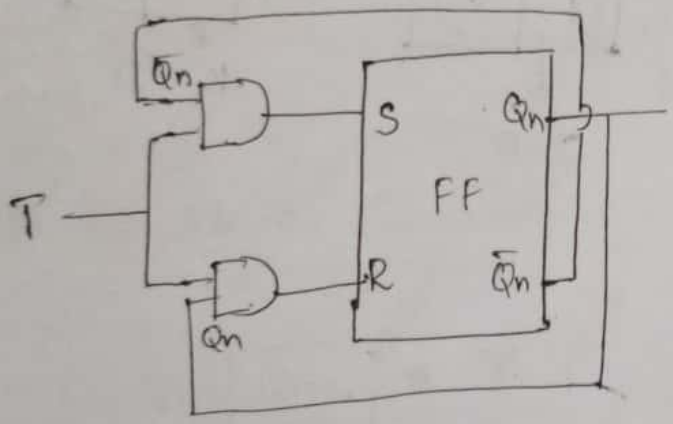
Characterstic table — excitation inputs



$S = T\bar{Q_n}$

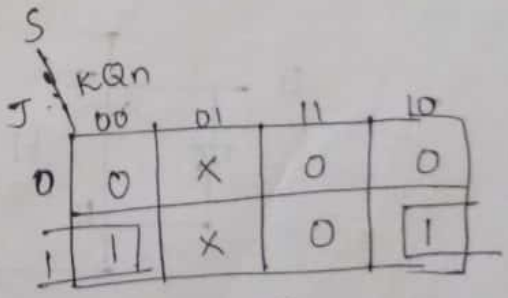


$R = TQ_n$

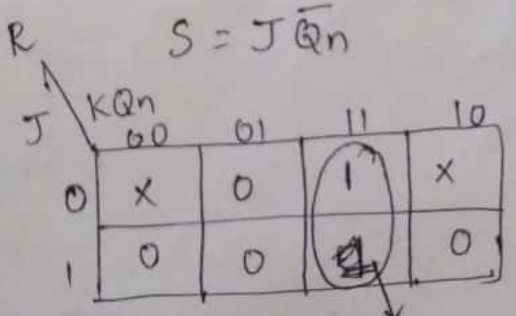


SR to JK FF Conversion :-

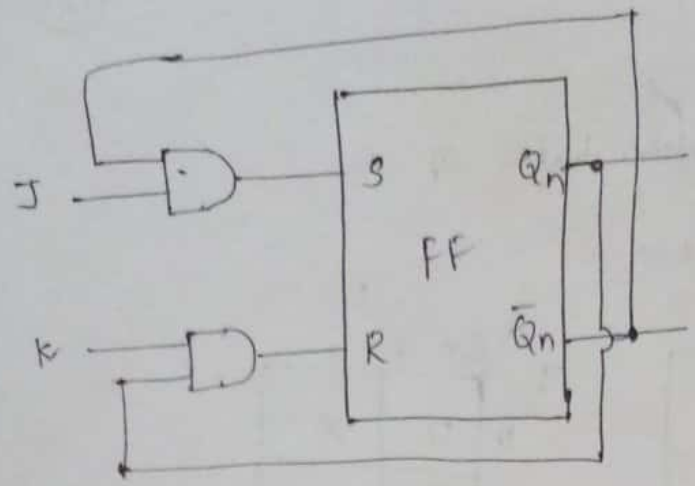
J	K	Qn	Qn+1	S	R
0	0	0	0	0	x
0	0	1	1	x	0
0	1	0	0	0	x
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	x	0
1	1	0	1	1	0
1	1	1	0	0	1



$S = J\bar{Q_n}$



$R = KQ_n$

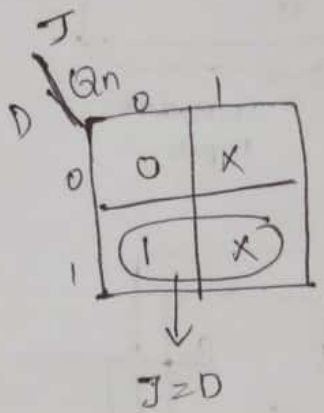


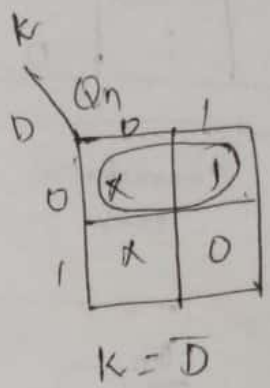
Jk FF to D-FF :-

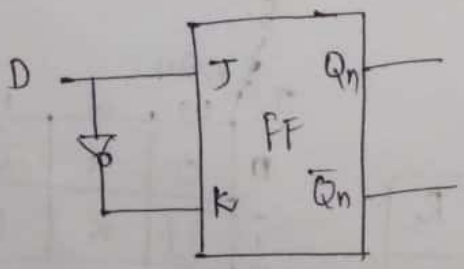
Jk given FF – Excitation table

D - Characteristic table

D	Qn	Qn+1	J	K
0	0	0	0	x
0	1	0	x	1
1	0	1	1	x
1	1	1	x	0





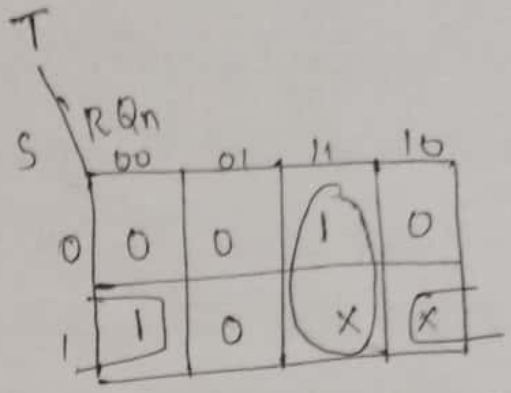


SR-FF Conversion:-

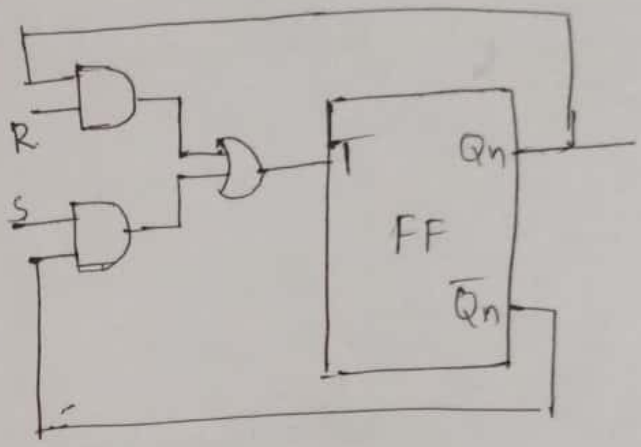
T - Excitation table

SR - Characteristic table.

S	R	Qn	Qn+1	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	x	x
1	1	1	x	x

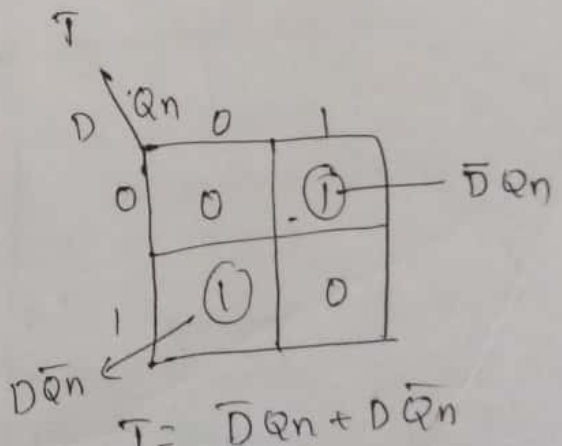


$T = RQ_n + S\bar{Q}_n$

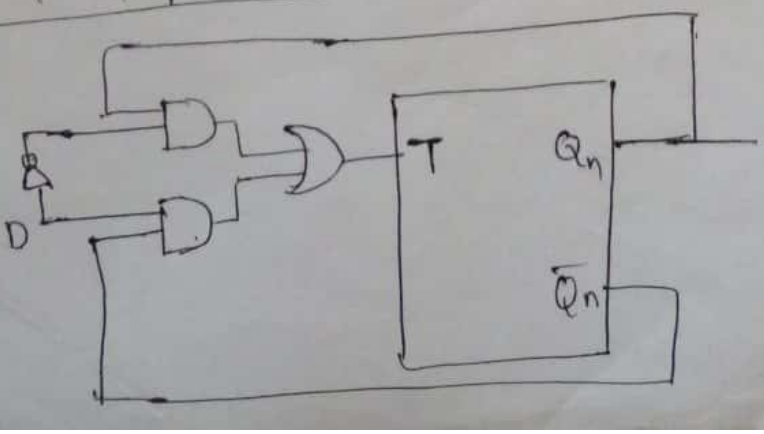


T to D Conversion:-

D	Qn	Qn+1	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0



$T = \bar{D}Q_n + D\bar{Q}_n$



Counting: A number of flip-flops may be connected in a particular fashion to count the pulses electronically. One flip-flop can count up to 2 pulses; two flip-flops can count up to $2^2 = 4$ pulses. In general, N flip-flops can count up to 2^N pulses. In a simple counter, all the flip-flops are connected in toggle mode. The clock pulses are applied to the first flip-flop and the clock terminal of each subsequent flip-flop is connected to the Q output of the previous flip-flop. Feedback may be provided if the maximum count required is not 2^N. Flip-flops may be used to count up or down or up/down. Figures 6.65, 6.66, 6.67, 6.68, 6.69, etc. illustrate the use of flip-flops for counting.

Frequency division: Flip-flops may be used to divide the input signal frequency by any number. A single flip-flop may be used to divide the input frequency by 2. Two flip-flops may be used to divide the input frequency by 4. In general, N flip-flops may be used to divide the input frequency by 2^N. If N flip-flops are connected as a ripple counter (a counter in which the external signal is applied to the clock terminal of the first flip-flop and the Q output of each flip-flop is connected to the clock input of next flip-flop) and if the input signal of frequency f is fed to the first flip-flop, the output of this flip-flop will be of frequency $f/2$, the output of the second flip-flop will be of frequency $f/4$, and so on.

Figure 6.65a and the waveforms in Figure 6.65b (also Figures 6.66a and 6.66b) illustrate the use of flip-flops for frequency division.

6.2 SHIFT REGISTERS

Data may be available in parallel form or in serial form. Multi-bit data is said to be in parallel form when all the bits are available (accessible) simultaneously. The data is said to be in serial form when the data bits appear sequentially (one after the other, in time) at a single terminal. Data may also be transferred in parallel form or in serial form. Parallel data transfer is the simultaneous transmission of all bits of data from one device to another. Serial data transfer is the transmission of one bit of data at a time from one device to another. Serial data must be transmitted under the synchronization of a clock, since the clock provides the means to specify the time at which each new bit is sampled.

As a flip-flop (FF) can store only one bit of data, a 0 or a 1, it is referred to as a single-bit register. When more bits of data are to be stored, a number of FFs are used. A register is a set of FFs used to store binary data. The storage capacity of a register is the number of bits (1s and 0s) of digital data it can retain. Loading a register means setting or resetting the individual FFs, i.e. inputting data into the register so that their states correspond to the bits of data to be stored. Loading may be serial or parallel. In serial loading, data is transferred into the register in serial form, i.e. one bit at a time, whereas in parallel loading, the data is transferred into the register in parallel form meaning that all the FFs are triggered into their new states at the same time. Parallel input requires that the SET and/or RESET controls of every FF be accessible.

A register may output data either in serial form or in parallel form. Serial output means that the data is transferred out of the register, one bit at a time serially. Parallel output means that the entire data stored in the register is available in parallel form, and can be transferred out at the same time.

Shift registers are a type of logic circuits closely related to counters. They are used basically for the storage and transfer of digital data. The basic difference between a shift register and a

counter is that, a shift register has no specified sequence of states except in certain very specialized applications, whereas a counter has a specified sequence of states.

A shift register is a very important digital building block. It has innumerable applications. Registers are often used to momentarily store binary information appearing at the output of an encoding matrix. A register might be used to accept input data from an alphanumeric keyboard and then present the data at the input of a microprocessor chip. Similarly, shift registers are often used to momentarily store binary data at the output of a decoder. A shift register also forms the basis for some very important arithmetic operations. For example, the operations of complementation, multiplication, and division are frequently implemented by means of a register. A shift register can also be connected to form a number of different types of counters. These counters offer some very distinct advantages.

6.2.1 Buffer Register

Some registers do nothing more than storing a binary word. The buffer register is the simplest of registers. It simply stores the binary word. The buffer may be a controlled buffer. Most of the buffer registers use D flip-flops.

Figure 6.51 shows a 4-bit buffer register. The binary word to be stored is applied to the data terminals. On the application of clock pulse, the output word becomes the same as the word applied at the input terminals, i.e. the input word is loaded into the register by the application of clock pulse.

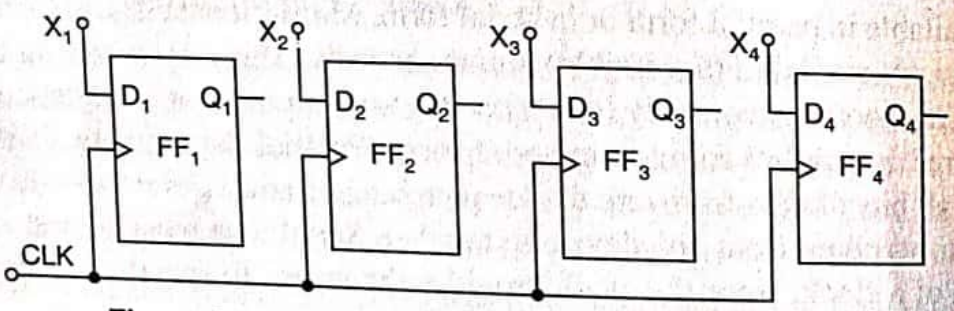


Figure 6.51 Logic diagram of a 4-bit buffer register.

When the positive clock edge arrives, the stored word becomes:

$$Q_4Q_3Q_2Q_1 = X_4X_3X_2X_1$$

or

$$Q = X$$

This circuit is too primitive to be of any use. What it needs is some control over the X bits, i.e. some way of holding them off until we are ready to store them.

6.2.2 Controlled Buffer Register

Figure 6.52 shows a controlled buffer register. If $\overline{CLR}$ goes LOW, all the FFs are RESET and the output becomes, Q = 0000.

When $\overline{CLR}$ is HIGH, the register is ready for action. LOAD is the control input. When LOAD is HIGH, the data bits X can reach the D inputs of FFs. At the positive-going edge of the next clock pulse, the register is loaded, i.e.

$$Q_4\ Q_3\ Q_2\ Q_1 = X_4\ X_3\ X_2\ X_1$$

$$Q = X$$

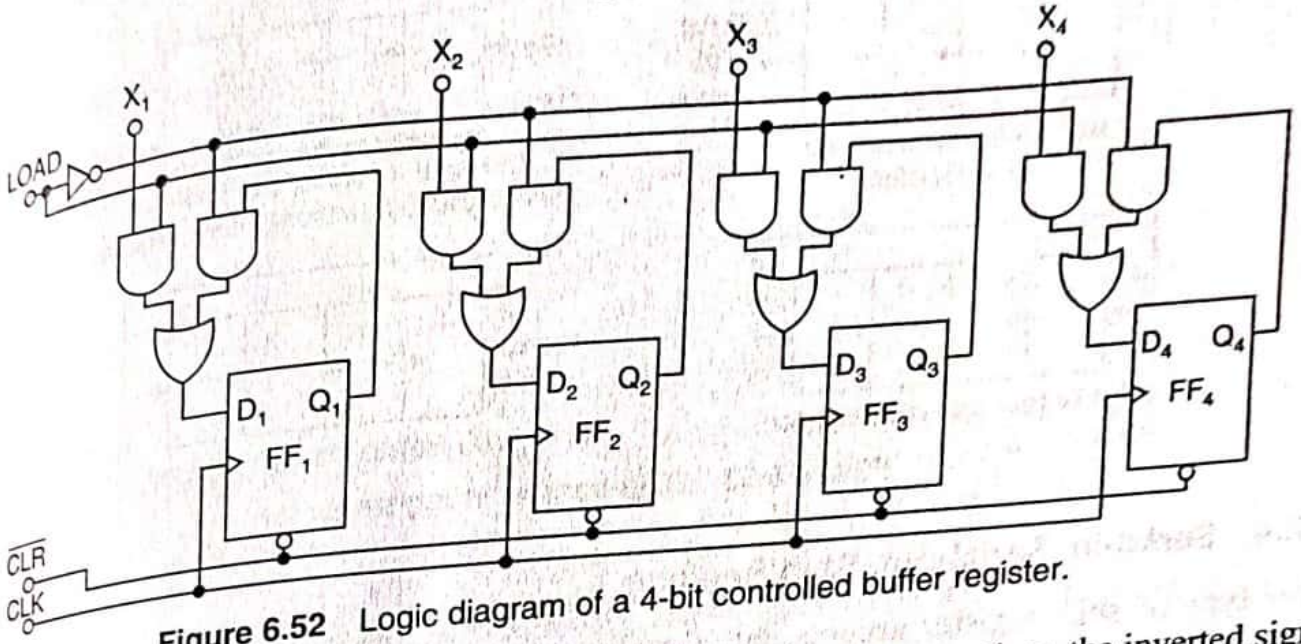


Figure 6.52 Logic diagram of a 4-bit controlled buffer register.

When LOAD is LOW, the X bits cannot reach the FFs. At the same time, the inverted signal $\overline{\text{LOAD}}$ is HIGH. This forces each flip-flop output to feed back to its data input. Therefore, data is circulated or retained as each clock pulse arrives. In other words, the contents of the register remain unchanged in spite of the clock pulses. Longer buffer registers can be built by adding more FFs.

6.2.3 Data Transmission in Shift Registers

A number of FFs connected together such that data may be shifted into and shifted out of them is called a *shift register*. Data may be shifted into or out of the register either in serial form or in parallel form. So, there are four basic types of shift registers: serial-in, serial-out; serial-in, parallel-out; parallel-in, serial-out; and parallel-in, parallel-out. The process of data shifting in these registers is illustrated in Figure 6.53. All of these configurations are commercially available as TTL MSI/LSI circuits. Data may be rotated left or right. Data may be shifted from left to right or right to left at will, i.e. in a bidirectional way. Also, data may be shifted in serially (in either way) or in parallel and shifted out serially (in either way) or in parallel.

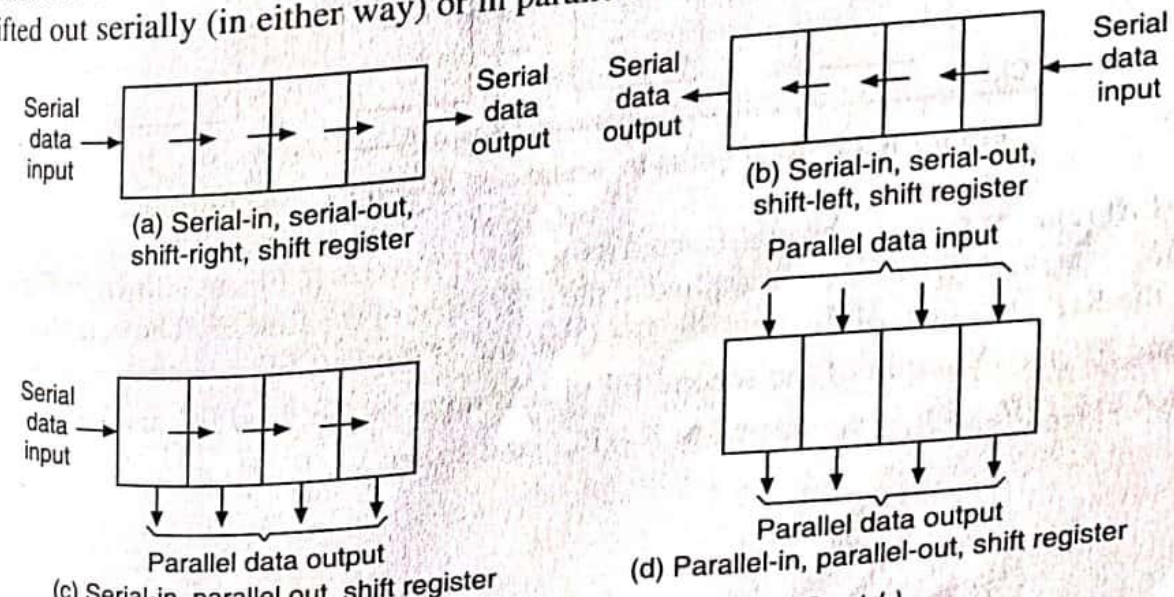


Figure 6.53 Data transfer in registers *(Contd.)*

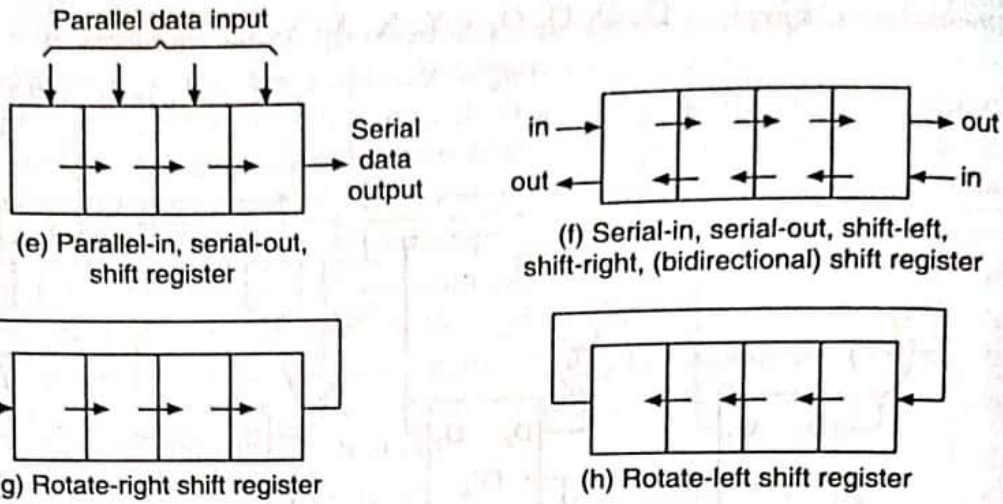


Figure 6.53 Data transfer in registers.

6.2.4 Serial-in, Serial-out, Shift Register

This type of shift register accepts data serially, i.e. one bit at a time, and also outputs data serially.

The logic diagram of a 4-bit serial-in, serial-out, shift-right, shift register is shown in Figure 6.54. With four stages, i.e. four FFs, the register can store upto four bits of data. Serial data is applied at the D input of the first FF. The Q output of the first FF is connected to the D input of the second FF, the Q output of the second FF is connected to the D input of the third FF and the Q output of the third FF is connected to the D input of the fourth FF. The data is outputted from the Q terminal of the last FF.

When serial data is transferred into a register, each new bit is clocked into the first FF at the positive-going edge of each clock pulse. The bit that was previously stored by the first FF is transferred to the second FF. The bit that was stored by the second FF is transferred to the third FF, and so on. The bit that was stored by the last FF is shifted out.

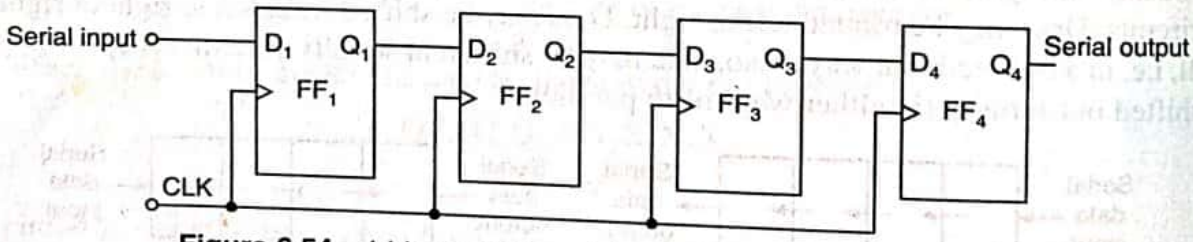
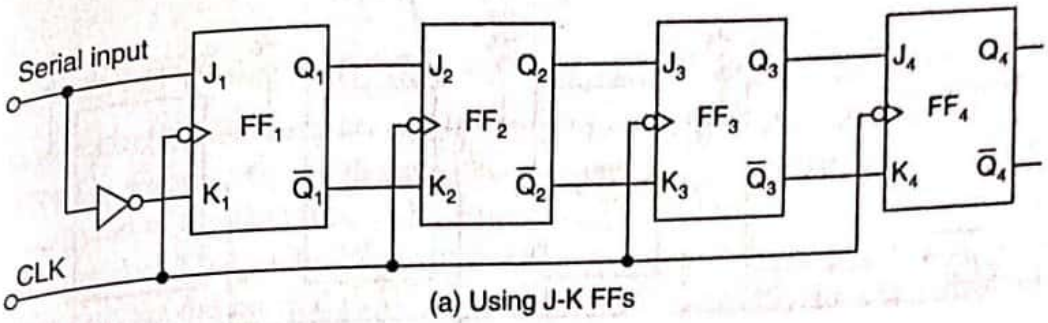
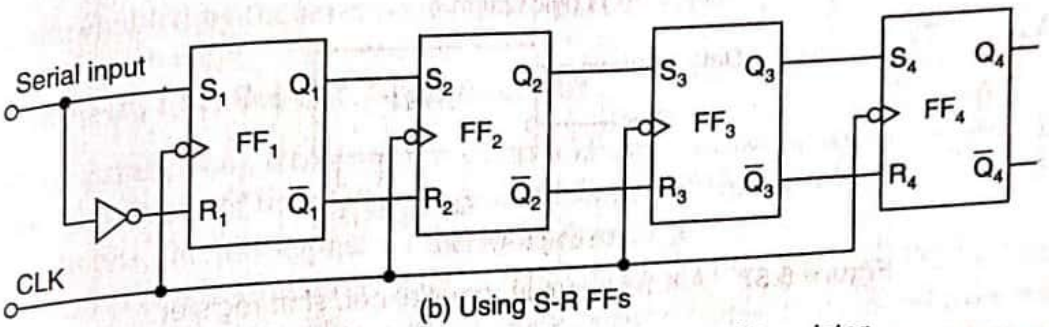


Figure 6.54 4-bit serial-in, serial-out, shift-right, shift register.

A shift register can also be constructed using J-K FFs or S-R FFs as shown in Figures 6.55a and 6.55b, respectively. The data is applied at the J(S) input of the first FF. The complement of this is fed to the K(R) terminal of the first FF. The Q output of the first FF is connected to J(S) input of the second FF, the Q output of the second FF to J(S) input of the third FF, and so on. Also, $\overline{Q}_1$ is connected to K_2 (R_2), $\overline{Q}_2$ is connected to K_3 (R_3), and so on.



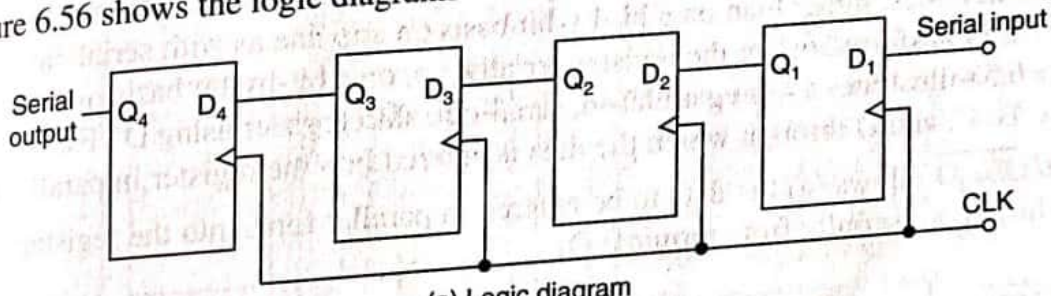
(a) Using J-K FFs



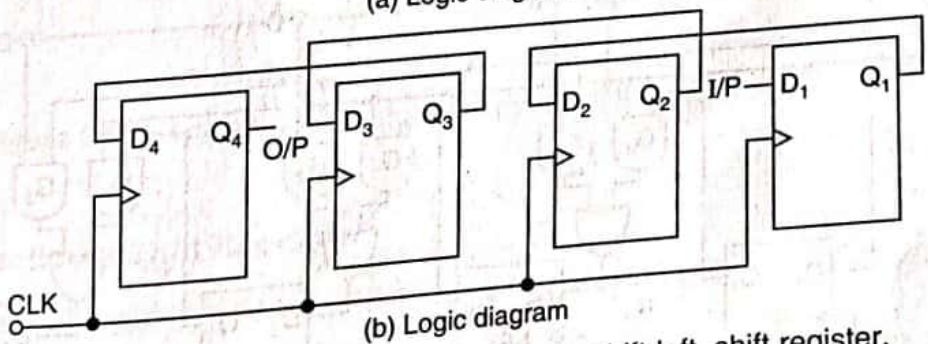
(b) Using S-R FFs

Figure 6.55 A 4-bit serial-in, serial-out, shift register.

Figure 6.56 shows the logic diagrams of a 4-bit serial-in, serial-out, shift-left, shift register.



(a) Logic diagram



(b) Logic diagram

Figure 6.56 A 4-bit serial-in, serial-out, shift-left, shift register.

6.2.5 Serial-in, Parallel-out, Shift Register

Figure 6.57 shows the logic diagram and the logic symbol of a 4-bit serial-in, parallel-out, shift register. In this type of register, the data bits are entered into the register serially, but the data stored in the register is shifted out in parallel form.

Once the data bits are stored, each bit appears on its respective output line and all bits are available simultaneously, rather than on a bit-by-bit basis as with the serial output. The serial-in, parallel-out, shift register can be used as a serial-in, serial-out, shift register if the output is taken from the Q terminal of the last FF.

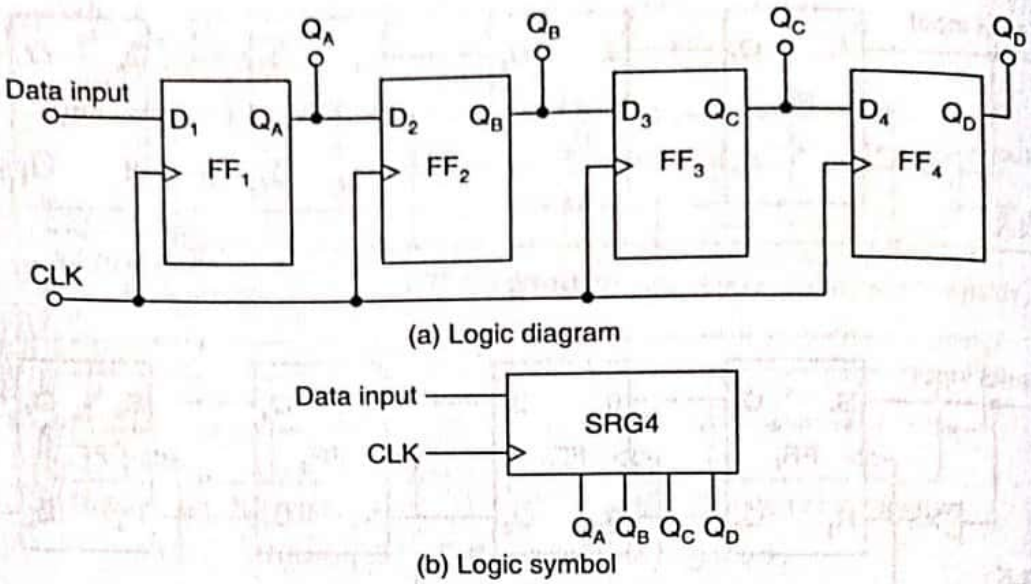


Figure 6.57 A 4-bit serial-in, parallel-out, shift register.

6.2.6 Parallel-in, Serial-out, Shift Register

For a parallel-in, serial-out, shift register, the data bits are entered simultaneously into their respective stages on parallel lines, rather than on a bit-by-bit basis on one line as with serial data inputs, but the data bits are transferred out of the register serially, i.e. on a bit-by-bit basis over a single line.

Figure 6.58 illustrates a 4-bit parallel-in, serial-out, shift register using D FFs. There are four data lines A, B, C, and D through which the data is entered into the register in parallel form. The signal Shift/$\overline{\text{LOAD}}$ allows (a) the data to be entered in parallel form into the register and (b) the data to be shifted out serially from terminal Q_4.

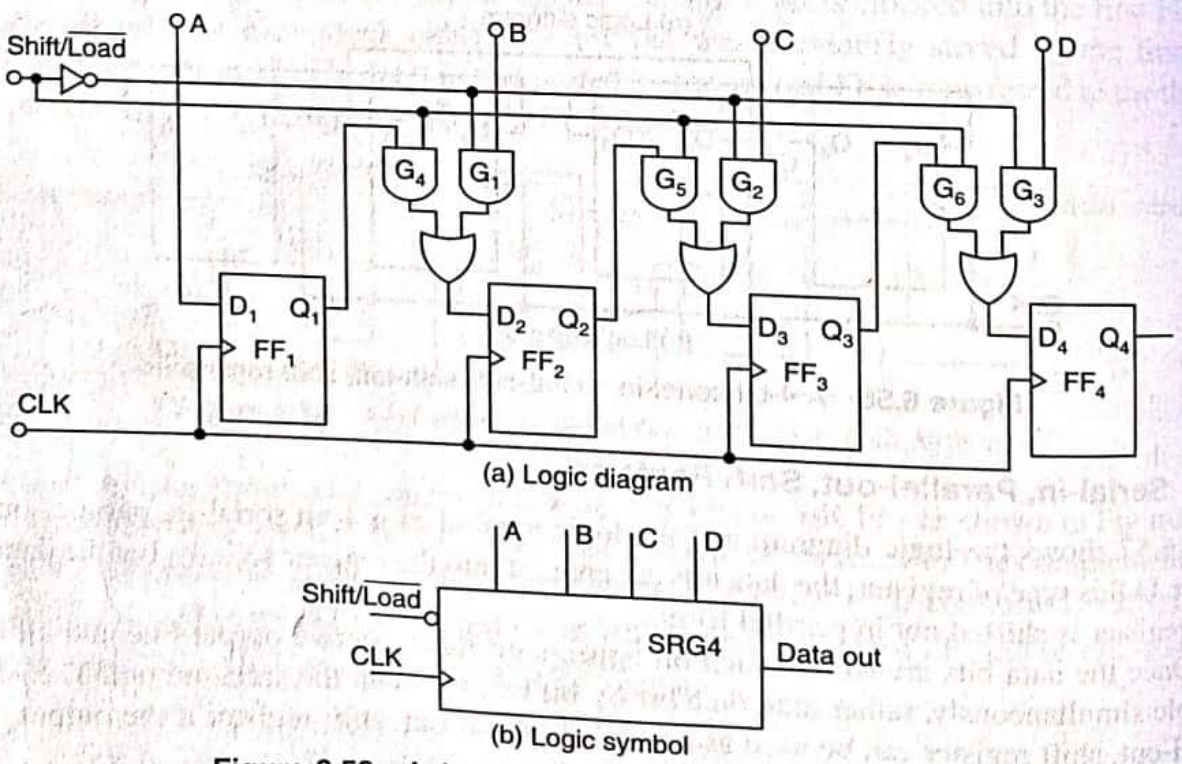


Figure 6.58 A 4-bit parallel-in, serial-out, shift register.

When Shift/$\overline{\text{LOAD}}$ line is HIGH, gates G_1, G_2, and G_3 are disabled, but gates G_4, G_5, and G_6 are enabled allowing the data bits to shift-right from one stage to the next. When Shift/$\overline{\text{LOAD}}$ line is LOW, gates G_4, G_5, and G_6 are disabled, whereas gates G_1, G_2, and G_3 are enabled allowing the data input to appear at the D inputs of the respective FFs. When a clock pulse is applied, these data bits are shifted to the Q output terminals of the FFs and, therefore, data is inputted in one step. The OR gate allows either the normal shifting operation or the parallel data entry depending on which AND gates are enabled by the level on the Shift/$\overline{\text{LOAD}}$ input.

6.2.7 Parallel-in, Parallel-out, Shift Register

In a parallel-in, parallel-out, shift register, the data is entered into the register in parallel form, and also the data is taken out of the register in parallel form. Immediately following the simultaneous entry of all data bits, the bits appear on the parallel outputs.

Figure 6.59 shows a 4-bit parallel-in, parallel-out, shift register using D FFs. Data is applied to the D input terminals of the FFs. When a clock pulse is applied, at the positive-going edge of that pulse, the D inputs are shifted into the Q outputs of the FFs. The register now stores the data. The stored data is available instantaneously for shifting out in parallel form.

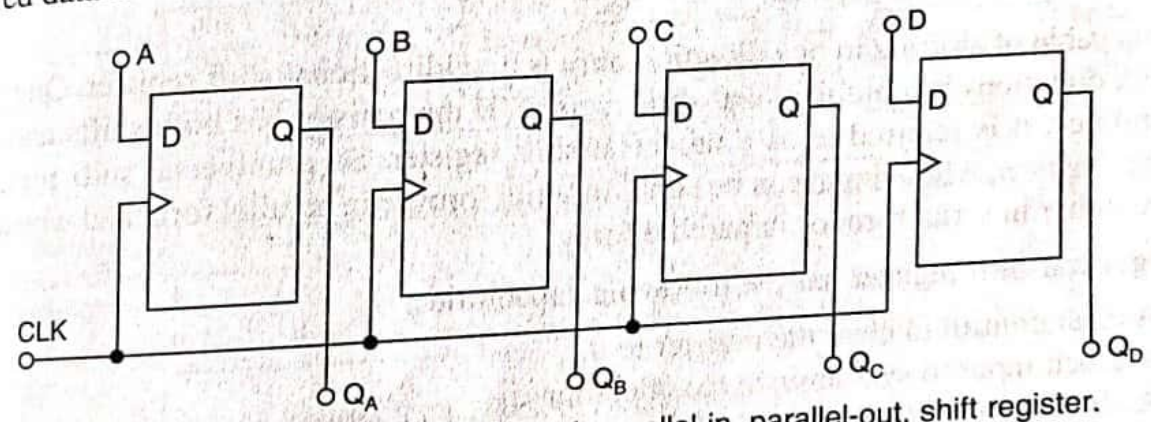


Figure 6.59 Logic diagram of a 4-bit parallel-in, parallel-out, shift register.

6.2.8 Bidirectional Shift Register

A bidirectional shift register is one in which the data bits can be shifted from left to right or from right to left.

Figure 6.60 shows the logic diagram of a 4-bit serial-in, serial-out, bidirectional (shift-left, shift-right) shift register. Right/$\overline{\text{Left}}$ is the mode signal. When Right/$\overline{\text{Left}}$ is a 1, the logic circuit works as a shift-right shift register. When Right/$\overline{\text{Left}}$ is a 0, it works as a shift-left shift register. The bidirectional operation is achieved by using the mode signal and two AND gates and one OR gate for each stage as shown in Figure 6.60.

A HIGH on the Right/$\overline{\text{Left}}$ control input enables the AND gates G_1, G_2, G_3, and G_4 and disables the AND gates G_5, G_6, G_7, and G_8, and the state of Q output of each FF is passed through the gate to the D input of the following FF. When a clock pulse occurs, the data bits are then effectively shifted one place to the right. A LOW on the Right/$\overline{\text{Left}}$ control input enables the AND

gates G_5, G_6, G_7, and G_8 and disables the AND gates G_1, G_2, G_3, and G_4, and the Q output of each FF is passed to the D input of the preceding FF. When a clock pulse occurs, the data bits are then effectively shifted one place to the left. Hence, the circuit works as a bidirectional shift register.

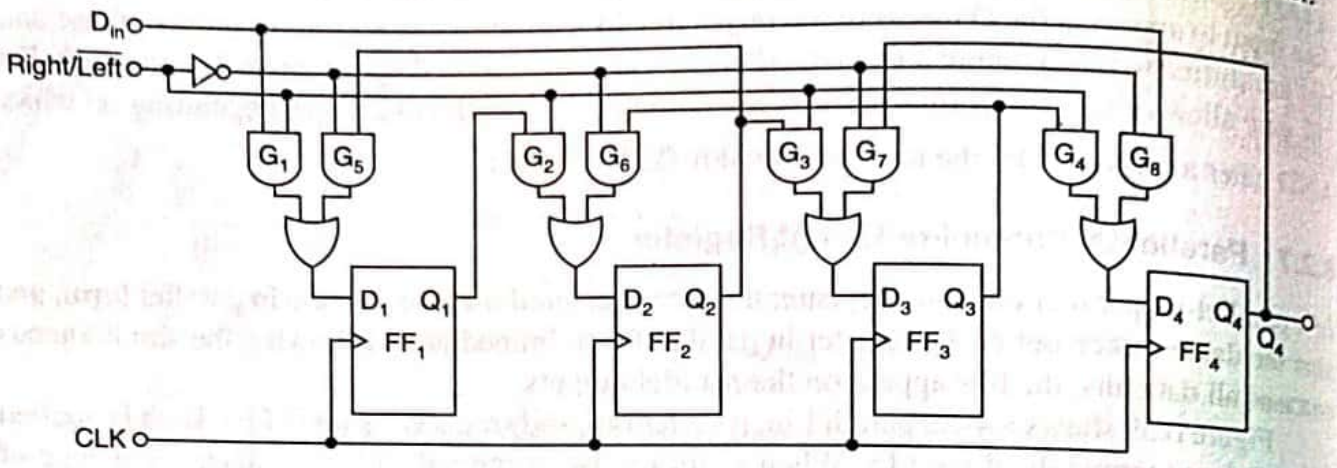


Figure 6.60 Logic diagram of a 4-bit bidirectional shift register.

6.2.9 Universal Shift Registers

A register capable of shifting in one direction only is a unidirectional shift register. One that can shift in both directions is a bidirectional shift register. If the register has both shifts and parallel load capabilities, it is referred to as a universal shift register. So a universal shift register is a bidirectional register, whose input can be either in serial form or in parallel form and whose output also can be either in serial form or in parallel form.

The most general shift register has the following capabilities:

1. A clear control to clear the register to 0.
2. A clock input to synchronize the operations.
3. A shift-right control to enable the shift-right operation and serial input and output lines associated with the shift-right.
4. A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift-left.
5. A parallel load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
6. n parallel output lines.
7. A control state that leaves the information in the register unchanged in the presence of the clock.

A universal shift register can be realized using multiplexers. Figure 6.61 shows the logic diagram of a 4-bit universal shift register that has all the capabilities listed above. It consists of four D flip-flops and four multiplexers. The four multiplexers have two common selection inputs S_1 and S_0. Input 0 in each multiplexer is selected when $S_1S_0 = 00$, input 1 is selected when $S_1S_0 = 01$, and input 2 is selected when $S_1S_0 = 10$ and input 3 is selected when $S_1S_0 = 11$. The selection inputs control the mode of operation of the register according to the function entries in Table 6.9. When $S_1S_0 = 0$, the present value of the register is applied to the D inputs of flip-flops. This condition

forms a path from the output of each flip-flop into the input of the same flip-flop. The next clock edge transfers into each flip-flop the binary value it held previously, and no change of state occurs. When $S_1S_0 = 01$, terminal 1 of the multiplexer inputs have a path to the D inputs of the flip-flops. This causes a shift-right operation, with the serial input transferred into flip-flop FF_4. When $S_1S_0 = 10$, a shift-left operation results with the other serial input going into flip-flop FF_1. Finally when $S_1S_0 = 11$, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock edge.

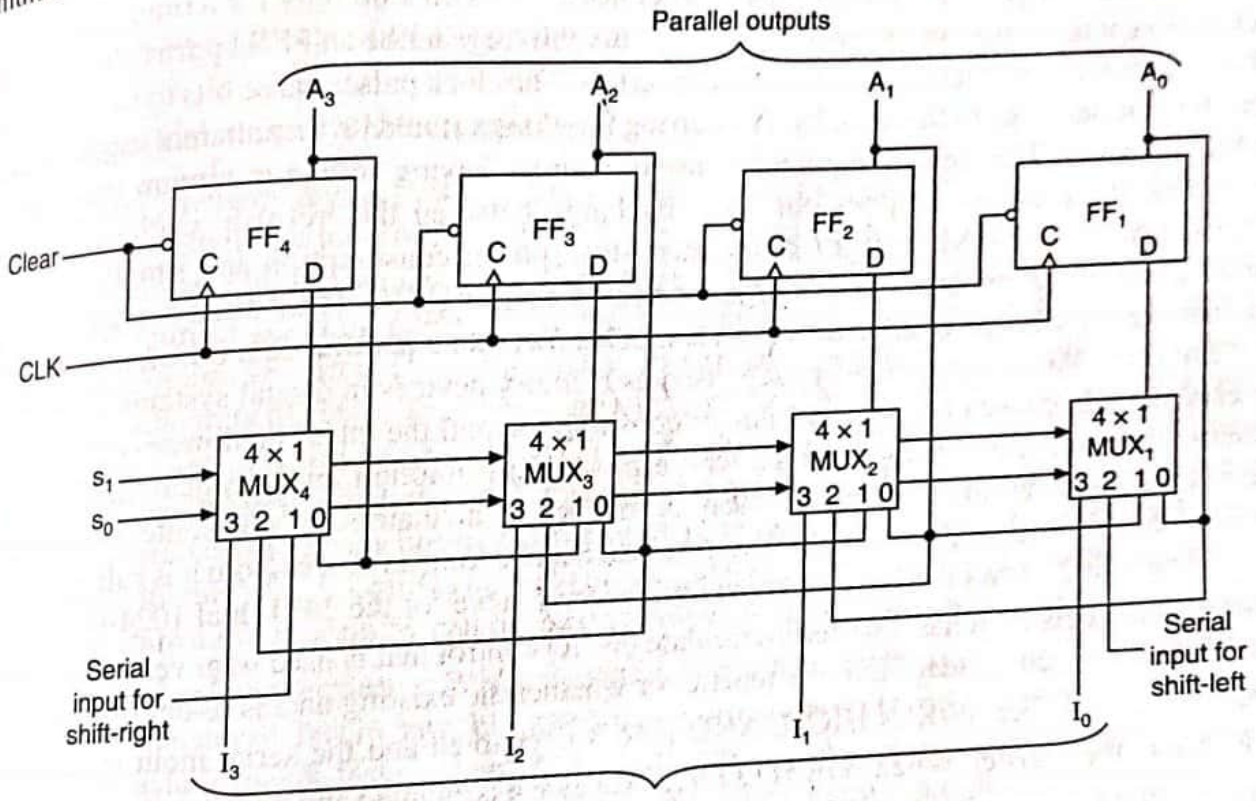


Figure 6.61 4-bit universal shift register.

Table 6.9 Function table for the register of Figure 6.61

Mode control		
S_1	S_0	Register operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

6.2.10 Dynamic Shift Registers

All the shift registers we have discussed till now are called *static shift registers*, because each one of the memory elements (i.e. FFs) used to build the register can retain the data bit indefinitely. So, once loaded, the contents of each element of the register remain the same. In a dynamic shift register, storage is accomplished by continually shifting the bits from one stage to the next and

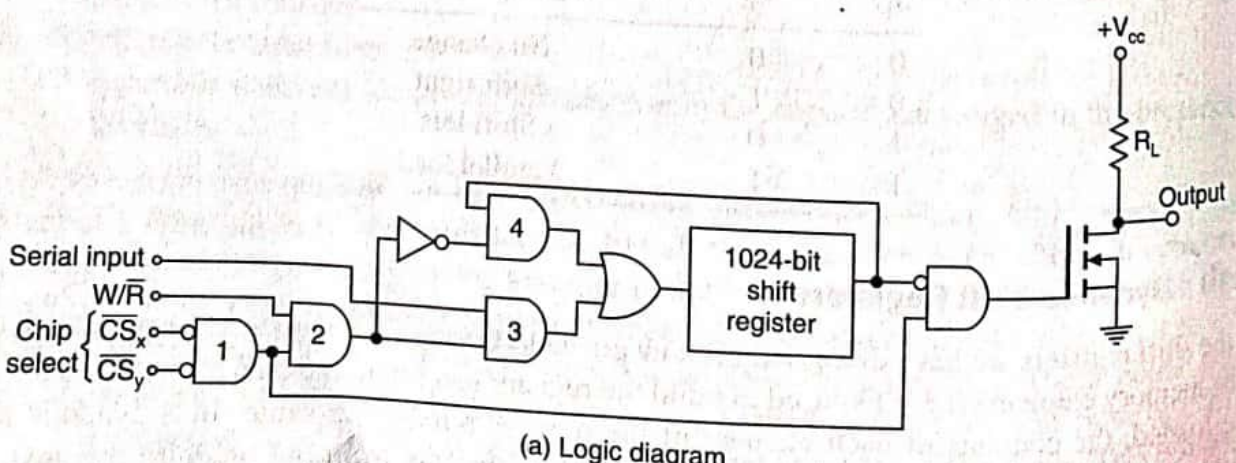
re-circulating the output of the last stage into the first stage. The data continually circulates through the register under the control of a clock. To obtain output, a serial output terminal must be accessed at a specific clock pulse; otherwise, the sequence of bits will not correspond to the data stored.

For example, if a 32-bit word is circulating through a 32-bit register; serial output must be given at multiples of 32 clock pulses. To store new data in such a register, the re-circulation path between the last stage and the first stage is intercepted and the new data is loaded serially into the first stage.

Since each stage of a dynamic shift register needs to retain a bit only for a time equal to one clock period, it is not necessary that each stage of the shift register be an FF. In particular, dynamic shift registers are constructed using dynamic inverters. The clock pulses cause bits to be transferred from one inverter stage to the next, by transferring the charge stored on the inherent capacitance of the MOS devices. This design requires the use of a clock, having certain minimum frequency to ensure that the capacitance does not fully discharge between the 'refresh' cycles. The main advantages of dynamic MOS registers are their small power consumption and simplicity, which permits a very large number of stages to be fabricated on a single IC. Their disadvantage is that all data transfer must be in serial form, which is much slower than parallel data transfer.

Dynamic MOS registers are widely used as memory devices in digital systems that operate on serial data. Because of their small power consumption and the inherent slowness of the serial systems, they are used in applications where power consumption and physical size are more important considerations than speed, such as in pocket calculators. In the context of memory applications, loading a register is called *writing* into it and taking an output from it is called *reading*.

Figure 6.62 shows the logic diagram and the truth table for the 2401 dual 1024-bit dynamic N-MOS shift register. It has a write/re-circulate ($W/\overline{R}$) control that is used to govern whether new serial data is written (loaded) into the register or whether the existing data is re-circulated (stored) by the register. When $W/\overline{R}$ is HIGH, AND gate 3 is enabled and the serial input is transferred through it to the register. When $W/\overline{R}$ is LOW, AND gate 4 is enabled and a re-circulation path from output to input is completed. Serial data appears at the output, regardless of the state of $W/\overline{R}$. The circuit also has two active-LOW chip-select inputs, labelled $\overline{CS}_x$ and $\overline{CS}_y$. Both $\overline{CS}_x$ and $\overline{CS}_y$ must be LOW in order to read or write data. Note, however, that re-circulation is independent of $\overline{CS}_x$ and $\overline{CS}_y$ and re-circulation is also independent of $W/\overline{R}$ if at least one of the chip-selects is HIGH.



(a) Logic diagram

Figure 6.62 One-half of the 2401 dual 1024-bit dynamic NMOS shift register *(Contd.)*

W/$\overline{R}$	$\overline{CS}_x$	$\overline{CS}_y$	Function
1	0	0	Write
0	×	×	Re-circulate
×	1	×	Re-circulate
×	×	1	Re-circulate
×	0	0	Read

(b) Truth table

Figure 6.62 One-half of the 2401 dual 1024-bit dynamic NMOS shift register.

6.2.11 Applications of Shift Registers

Time delays: In many digital systems, it is necessary to delay the transfer of data until such time as operations on other data have been completed, or to synchronize the arrival of data at a subsystem where it is processed with other data. A shift register can be used to delay the arrival of serial data by a specific number of clock pulses, since the number of stages corresponds to the number of clock pulses required to shift each bit completely through the register. The total time delay can be controlled by adjusting the clock frequency and by prescribing the number of stages in the register. In practice, the clock frequency is fixed and the total delay can be adjusted only by controlling the number of stages through which the data is passed. By using a serial-in, parallel-out register and by taking the serial output at any one of the intermediate stages, we have the flexibility to delay the output by any number of clock pulses equal to or less than the number of stages in the register. The arrangement shown in Figure 6.59 can be used to delay the data by 4 clock pulses.

Serial/Parallel data conversion: We know that data can be available either in serial form or in parallel form. Transfer of data in parallel form is much faster than that in serial form. Similarly, the processing of data is much faster when all the data bits are available simultaneously. For this reason, digital systems in which speed is an important consideration are designed to operate on data in parallel form. When large data is to be transmitted over long distances, transmitting data on parallel lines is costly and impracticable. It is convenient and economical to transmit data in serial form, since serial data transmission requires only one line. Shift registers are used for converting serial data to parallel form, so that a serial input can be processed by a parallel system and for converting parallel data to serial form, so that parallel data can be transmitted serially.

A serial-in, parallel-out, shift register can be used to perform serial-to-parallel conversion, and a parallel-in, serial-out, shift register can be used to perform parallel-to-serial conversion. A universal shift register can be used to perform both the serial-to-parallel and parallel-to-serial data conversions. A bidirectional shift register can be used to reverse the order of data. The arrangement shown in Figure 6.57 can be used for serial-to-parallel conversion of a 4-bit data. The arrangement shown in Figure 6.58 can be used for parallel-to-serial conversion of a 4-bit data.

Ring counters: Ring counters are constructed by modifying the serial-in, serial-out, shift registers. There are two types of ring counters—basic ring counter and Johnson counter. The basic ring counter can be obtained from a serial-in, serial-out, shift register by connecting the Q output of the last FF to the D input of the first FF. The Johnson counter can be obtained from a serial-in, serial-out, shift register by connecting the $\overline{Q}$ output of the last FF to the D input of the first FF. Ring counter outputs can be used as a sequence of synchronizing pulses. The ring counter is a decimal

counter. It is a divide-by-N counter, where N is the number of stages. The keyboard encoder is an example of the application of a shift register used as a ring counter in conjunction with other devices.

Ring counters are dealt with in detail later in this chapter only. Figures 6.108, and 6.109 (Section 6.3.13) illustrate the use of the shift register as a ring counter. Figures 6.112 and 6.113 (Section 6.3.13) illustrate the use of the shift register as a twisted ring counter.

Universal asynchronous receiver transmitter (UART): Computers and microprocessor-based systems often send and receive data in a parallel format. Frequently these systems must communicate with external devices that send and/or receive serial data. An interfacing device used to accomplish these conversions is the UART.

A UART is a specially designed integrated circuit that contains all the registers and synchronizing circuitry necessary to receive data in serial form and to convert and transmit it in parallel form and vice versa. Figure 6.63 shows the use of UART as an interfacing device.

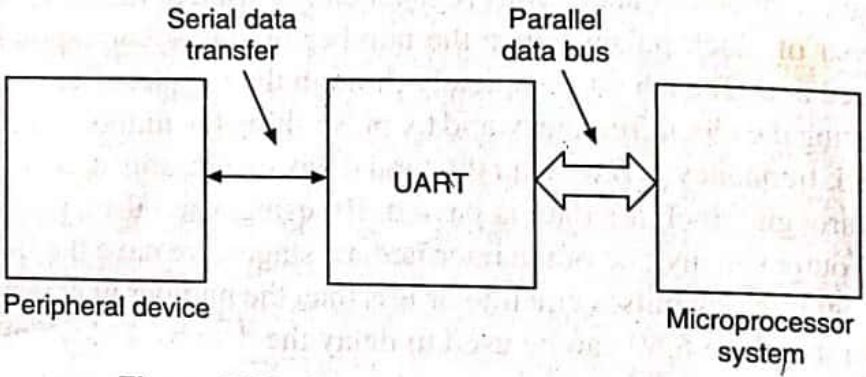


Figure 6.63 UART as an interfacing device.

6.3 COUNTERS

A digital counter is a set of flip-flops (FFs) whose states change in response to pulses applied at the input to the counter. The FFs are interconnected such that their combined state at any time is the binary equivalent of the total number of pulses that have occurred up to that time. Thus, as its name implies, a counter is used to count pulses. A counter can also be used as a frequency divider to obtain waveforms with frequencies that are specific fractions of the clock frequency. They are also used to perform the timing function as in digital watches, to create time delays, to produce non-sequential binary counts, to generate pulse trains, and to act as frequency counters, etc.

Counters may be *asynchronous* counters or *synchronous* counters. Asynchronous counters are also called *ripple counters*. The ripple counter is the simplest type of counter, the easiest to design and requires the least amount of hardware. In ripple counters, the FFs within the counter are not made to change the states at exactly the same time. This is because the FFs are not triggered simultaneously. The clock does not directly control the time at which every stage changes state. An asynchronous counter uses T FFs to perform a counting function. The actual hardware used is usually J-K FFs connected in toggle mode, i.e. with Js and Ks connected to logic 1. Even D FFs may be used here.

The asynchronous counter has a disadvantage, in so far as the unwanted spikes are concerned. This limitation is overcome in parallel counters. The asynchronous counter is called ripple counter

because when the counter, for example, goes from 1111 to 0000, the first stage causes the second to flip, the second causes the third to flip, and the third causes the fourth to flip, and so on. In other words, the transition of the first stage ripples through to the last stage. In doing so, many intermediate stages are briefly entered. If there is a gate that will AND during any state, a brief spike will be seen at the gate output every time the counter goes from 1111 to 0000. Ripple counters are also called *serial* or *series counters*. Synchronous counters are clocked such that each FF in the counter is triggered at the same time. This is accomplished by connecting the clock line to each stage of the counter. Synchronous counters are faster than asynchronous counters, because the propagation delay involved is less.

Comparison of synchronous and asynchronous counters is given in Table 6.10.

Table 6.10 Synchronous versus asynchronous counters

Asynchronous counters	Synchronous counters
1. In this type of counter FFs are connected in such a way that the output of first FF drives the clock for the second FF, the output of the second the clock of the third and so on.	1. In this type of counter there is no connection between the output of first FF and clock input of next FF and so on.
2. All the FFs are not clocked simultaneously.	2. All the FFs are clocked simultaneously.
3. Design and implementation is very simple even for more number of states.	3. Design and implementation becomes tedious and complex as the number of states increases.
4. Main drawback of these counters is their low speed as the clock is propagated through a number of FFs before it reaches the last FF.	4. Since clock is applied to all the FFs simultaneously the total propagation delay is equal to the propagation delay of only one FF. Hence they are faster.

A counter may be an *up-counter* or a *down-counter*. An up-counter is a counter which counts in the upward direction, i.e. 0, 1, 2, 3,..., N. A down-counter is a counter which counts in the downward direction, i.e. $N, N-1, N-2, N-3, \ldots, 1, 0$. Each of the counts of the counter is called the *state* of the counter. The number of states through which the counter passes before returning to the starting state is called the *modulus* of the counter. Hence, the modulus of a counter is equal to the total number of distinct states (counts) including zero that a counter can store. In other words, the number of input pulses that causes the counter to reset to its initial count is called the modulus of the counter. Since a 2-bit counter has 4 states, it is called a mod-4 counter. It divides the input clock signal frequency by 4, therefore, it is also called a divide-by-4 counter. It requires two FFs. Similarly, a 3-bit counter uses 3 FFs and has $2^3 = 8$ states. It divides the input clock frequency by 2^3, i.e. 8. In general, an n-bit counter will have n FFs and 2^n states, and divides the input frequency by 2^n. Hence, it is a divide-by-2^n counter.

A counter may have a shortened modulus. This type of counter does not utilize all the possible states. Some of the states are unutilized, i.e. invalid. The number of FFs required to construct a mod-N counter equals the smallest n for which $N \leq 2^n$. A mod-N counter divides the input frequency by N, hence, it is called a divide-by-N counter. In an asynchronous counter, the invalid states are bypassed by providing a suitable feedback. In a synchronous counter, the invalid states are taken care of by treating the corresponding excitations as don't cares. The least significant bit (LSB) of

any counter is that bit which changes most often. In ripple counters, the LSB is the Q output of the FF to which the external clock is applied.

A counter which goes through all the possible states before restarting is called the *full modulus counter*. A counter in which the maximum number of states can be changed is called the *variable modulus counter*. The final state of the counter sequence is called the *terminal count*.

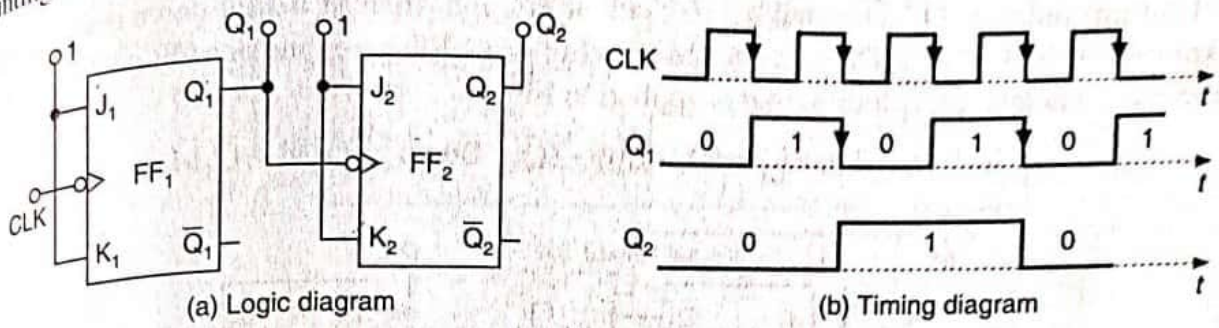
Lock-out: In shortened-modulus counters, there may occur the problem of *lock-out*. Sometimes when the counter is switched on, or any time during counting, because of noise spikes, the counter may find itself in some unused (invalid) state. Subsequent clock pulses may cause the counter to move from one unused state to another unused state and the counter may never come to a valid state. So, the counter becomes useless. A counter whose unused states have this feature is said to suffer from the problem of lock-out. To ensure that, at 'start-up' the counter is in its initial state, external logic circuitry is provided which properly resets each FF. The logic circuitry for presetting the counter to its initial state can be provided either by obtaining an expression for reset/preset for the FFs or by modifying the design such that the counter goes from each invalid state to the initial state after the clock pulse. So, no don't cares are permitted in this design.

Combination of modulo counters: A single FF is a mod-2 counter. We can have a counter of any modulus by choosing an appropriate number of FFs and providing proper feedback. Counters of different mods can be combined to get another mod counter. For example, a mod-2 counter and a mod-5 counter can be combined to get a mod-10 counter; a mod-5 counter and a mod-4 counter can be combined to get a mod-20 counter, and so on. The connection between the individual counters may be a ripple connection, or the counters may be operated in synchronism with one another independently of whether the individual counters are ripple or synchronous. Further, we are at liberty to choose the order of the individual counters in a chain of counters. Such permutations will not change the modulus of the composite counter but may well make a substantive difference in the code in which the counter state is to be read.

6.3.1 Asynchronous Counters

Two-bit ripple up-counter using negative edge-triggered flip-flops: The 2-bit up-counter counts in the order 0, 1, 2, 3, 0, 1,..., i.e. 00, 01, 10, 11, 00, 01,...,etc. Figure 6.64 shows a 2-bit ripple up-counter, using negative edge-triggered J-K FFs, and its timing diagram. The counter is initially reset to 00. When the first clock pulse is applied, FF_1 toggles at the negative-going edge of this pulse, therefore, Q_1 goes from LOW to HIGH. This becomes a positive-going signal at the clock input of FF_2. So, FF_2 is not affected, and hence, the state of the counter after one clock pulse is $Q_1 = 1$ and $Q_2 = 0$, i.e. 01. At the negative-going edge of the second clock pulse, FF_1 toggles. So, Q_1 changes from HIGH to LOW and this negative-going signal applied to CLK of FF_2 activates FF_2, and hence, Q_2 goes from LOW to HIGH. Therefore, $Q_1 = 0$ and $Q_2 = 1$, i.e. 10 is the state of the counter after the second clock pulse. At the negative-going edge of the third clock pulse, FF_1 toggles. So Q_1 changes from a 0 to a 1. This becomes a positive-going signal to FF_2, hence, FF_2 is not affected. Therefore, $Q_2 = 1$ and $Q_1 = 1$, i.e. 11 is the state of the counter after the third clock pulse. At the negative-going edge of the fourth clock pulse, FF_1 toggles. So, Q_1 goes from a 1 to a 0. This negative-going signal at Q_1 toggles FF_2, hence, Q_2 also changes from a 1 to a 0. Therefore, $Q_2 = 0$ and $Q_1 = 0$, i.e. 00 is the state of the counter after the fourth clock pulse. For subsequent clock pulses, the counter goes through the

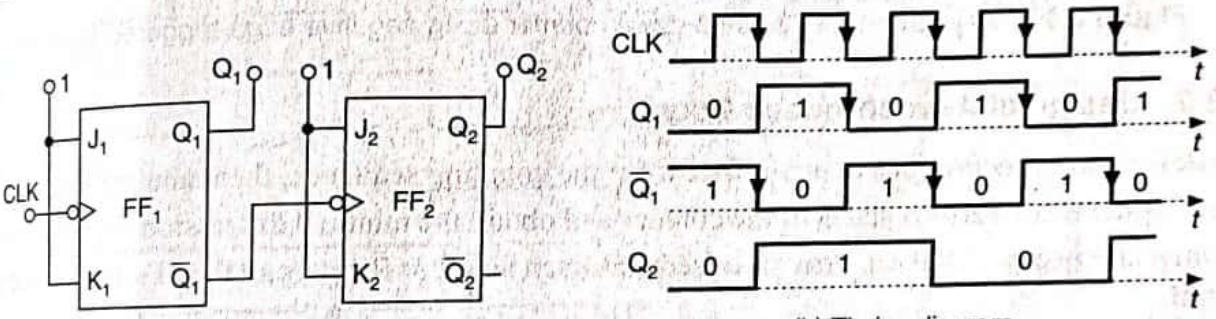
ame sequence of states. So, it acts as a mod-4 counter with Q_1 as the LSB and Q_2 as the MSB. The ounting sequence is thus 00, 01, 10, 11, 00, 01,..., etc.



(a) Logic diagram (b) Timing diagram

Figure 6.64 Asynchronous 2-bit up-counter using negative edge-triggered flip-flops.

Two-bit ripple down-counter using negative edge-triggered flip-flops: A 2-bit down-counter ounts in the order 0, 3, 2, 1, 0, 3,..., i.e. 00, 11, 10, 01, 00, 11, ..., etc. Figure 6.65 shows a 2-bit ipple down-counter, using negative-edge triggered J-K FFs, and its timing diagram.



(a) Logic diagram (b) Timing diagram

Figure 6.65 Asynchronous 2-bit down-counter using negative edge-triggered flip-flops.

For down counting, $\overline{Q}_1$ of FF_1 is connected to the clock of FF_2. Let initially all the FFs be eset, i.e. let the count be 00. At the negative-going edge of the first clock pulse, FF_1 toggles, so, Q_1 goes from a 0 to a 1 and $\overline{Q}_1$ goes from a 1 to a 0. This negative-going signal at $\overline{Q}_1$ applied to the lock input of FF_2, toggles FF_2 and, therefore, Q_2 goes from a 0 to a 1. So, after one clock pulse $Q_2 = 1$ and $Q_1 = 1$, i.e. the state of the counter is 11. At the negative-going edge of the second clock ulse, Q_1 changes from a 1 to a 0 and $\overline{Q}_1$ from a 0 to a 1. This positive-going signal at $\overline{Q}_1$ does not ffect FF_2 and, therefore, Q_2 remains at a 1. Hence, the state of the counter after the second clock ulse is 10. At the negative-going edge of the third clock pulse, FF_1 toggles. So, Q_1 goes from a 0 o a 1 and $\overline{Q}_1$ from a 1 to a 0. This negative-going signal at $\overline{Q}_1$ toggles FF_2 and, so, Q_2 changes from 1 to a 0. Hence, the state of the counter after the third clock pulse is 01. At the negative-going dge of the fourth clock pulse, FF_1 toggles. So, Q_1 goes from a 1 to a 0 and $\overline{Q}_1$ from a 0 to a 1. This ositive-going signal at $\overline{Q}_1$ does not affect FF_2. So, Q_2 remains at a 0. Hence, the state of the ounter after the fourth clock pulse is 00. For subsequent clock pulses the counter goes through the ame sequence of states, i.e. the counter counts in the order 00, 11, 10, 01, 00, and 11 ...

Two-bit ripple up-down counter using negative edge-triggered flip-flops: As the name ndicates an up-down counter is a counter which can count both in upward and downward directions

(Figure 6.66). An up-down counter is also called a forward/backward counter or a bidirectional counter. So, a control signal or a mode signal M is required to choose the direction of count. When M = 1 for up counting, Q_1 is transmitted to clock of FF_2 and when M = 0 for down counting, $\bar{Q}_1$ is transmitted to clock of FF_2. This is achieved by using two AND gates and one OR gate as shown in Figure 6.66. The external clock signal is applied to FF_1.

$$\text{Clock signal to } FF_2 = (Q_1 \cdot \text{Up}) + (\bar{Q}_1 \cdot \text{Down}) = Q_1M + \bar{Q}_1\bar{M}$$

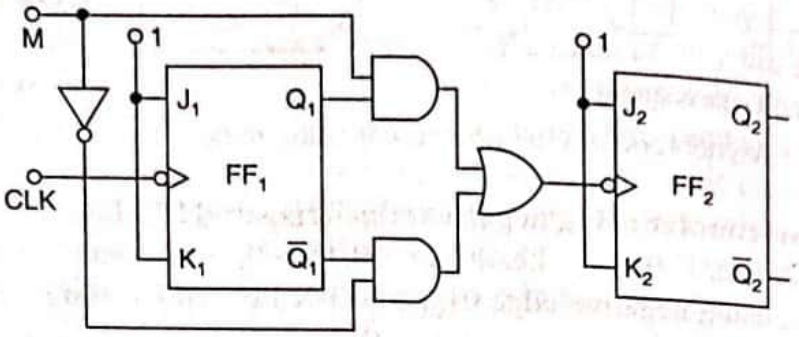


Figure 6.66 Asynchronous 2-bit up-down counter using negative edge-triggered flip-flops.

6.3.2 Design of Asynchronous Counters

To design an asynchronous counter, first write the counting sequence, then tabulate the values of reset signal R for various states of the counter and obtain the minimal expression for R or $\bar{R}$ using K-map or any other method. Provide a feedback such that R or $\bar{R}$ resets all the FFs after the desired count.

Design of a mod-6 asynchronous counter using T FFs: A mod-6 counter has six stable states 000, 001, 010, 011, 100, and 101. When the sixth clock pulse is applied, the counter temporarily goes to 110 state, but immediately resets to 000 because of the feedback provided. It is a 'divide-by-6 counter', in the sense that it divides the input clock frequency by 6. It requires three FFs, because the smallest value of n satisfying the condition $N \leq 2^n$ is $n = 3$; three FFs can have eight possible states, out of which only six are utilized and the remaining two states 110 and 111, are invalid. If initially the counter is in 000 state, then after the first clock pulse it goes to 001, after the second clock pulse, it goes to 010, and so on. After the sixth clock pulse, it goes to 000.

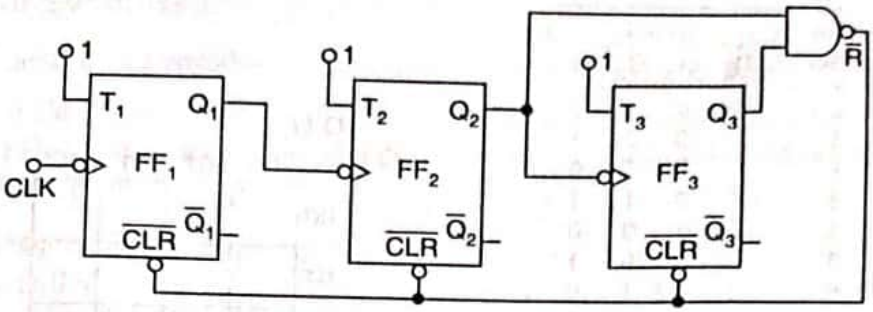
For the design, write a truth table (Figure 6.67c) with the present state outputs Q_3, Q_2 and Q_1 as the variables, and reset R as the output and obtain an expression for R in terms of Q_3, Q_2 and Q_1. That decides the feedback to be provided. From the truth table, $R = Q_3Q_2$. For active-LOW reset, $\bar{R}$ is used. The reset pulse is of very short duration, of the order of nanoseconds and it is equal to the propagation delay time of the NAND gate used. The expression for R can also be determined as follows.

$$R = 0 \text{ for } 000 \text{ to } 101,\ R = 1 \text{ for } 110, \text{ and } R = X \text{ for } 111$$

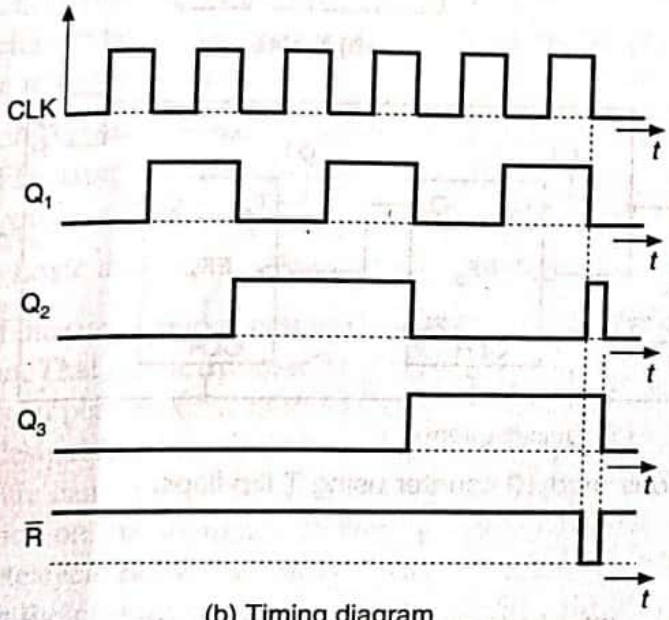
Therefore,

$$R = Q_3Q_2\bar{Q}_1 + Q_3Q_2Q_1 = Q_3Q_2$$

The logic diagram, the timing diagram, and the table for R of a mod-6 counter are all shown in Figure 6.67. From the timing diagram it is seen that a glitch appears in the waveform of Q_2.



(a) Logic diagram



(b) Timing diagram

After pulses	State Q_3	Q_2	Q_1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
	↓	↓	↓	
	0	0	0	0
7	0	0	1	0

(c) Table for R

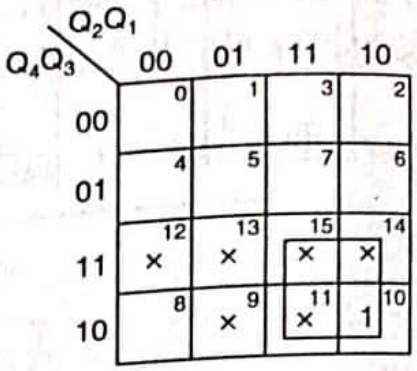
Figure 6.67 Asynchronous mod-6 counter using T flip-flops.

Design of a mod-10 asynchronous counter using T FFs: A mod-10 counter is a decade counter. It is also called a BCD counter or a divide-by-10 counter. It requires four FFs (the smallest value of n satisfying the condition $10 \leq 2^n$, is $n = 4$). So, there are 16 possible states, out of which ten are valid and the remaining six are invalid. The counter has ten stable states, 0000 through 1001, i.e. it counts from 0 to 9. The initial state is 0000 and after nine clock pulses it goes to 1001. When the tenth clock pulse is applied, the counter goes to state 1010 temporarily, but because of the feedback provided, it resets to initial state 0000. So, there will be a glitch in the waveform of Q_2. The state 1010 is a temporary state for which the reset signal R = 1, R = 0 for 0000 to 1001, and R = X (don't care) for 1011 to 1111.

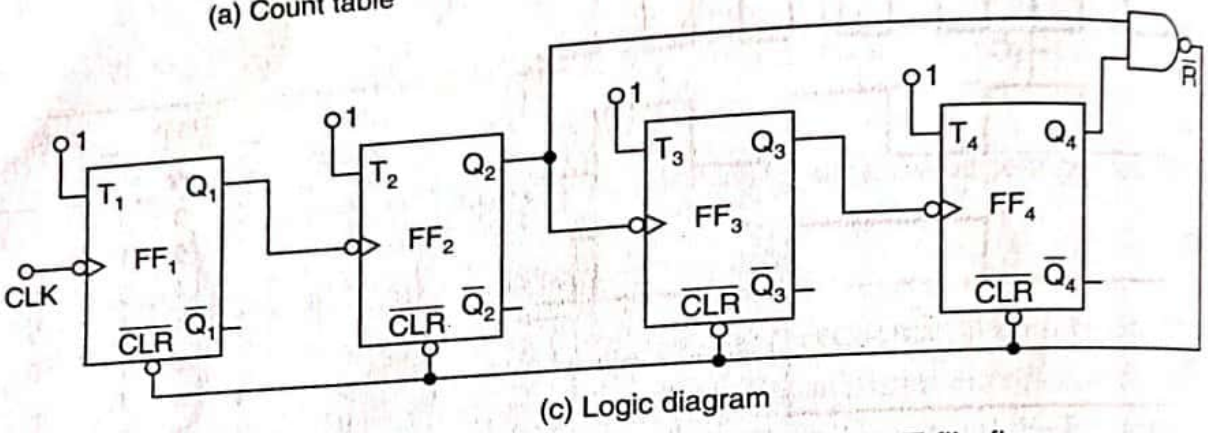
The count table and the K-map for reset are shown in Figures 6.68a and 6.68b, respectively. From the K-map, $R = Q_4Q_2$. So, feedback is provided from second and fourth FFs. For active-HIGH reset, Q_4Q_2 is applied to the CLEAR terminal. For active-LOW reset, $\overline{Q_4Q_2}$ is connected to $\overline{CLR}$ of all the FFs. The logic diagram of the decade counter is shown in Figure 6.68c.

After pulses	Count Q_4	Q_3	Q_2	Q_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

(a) Count table



(b) K-Map



(c) Logic diagram

Figure 6.68 Asynchronous mod-10 counter using T flip-flops.

6.3.3 Synchronous Counters

Asynchronous counters are serial counters. They are slow because each FF can change state only if all the preceding FFs have changed their state. The propagation delay thus gets accumulated, and so causes problems. If the clock frequency is very high, the asynchronous counter may skip some of the states and, therefore, malfunction. This problem is overcome in synchronous or parallel counters. Synchronous counters are counters in which all the FFs are triggered simultaneously (in parallel) by the clock-input pulses. Whether a FF toggles or not depends on the FF's inputs (J, K, or D, or T, or S, R). Since all the FFs change state simultaneously in synchronization with the clock pulse, the propagation delays of FFs do not add together (as in ripple counters) to produce the overall delay. In fact, the propagation delay of a synchronous counter is equal to the propagation delay of just one FF plus the propagation delay of the gates involved. So, the synchronous counters can operate at much higher frequencies than those that can be used in asynchronous counters.

Synchronous counters have the advantages of high speed and less severe decoding problems, but the disadvantage of having more circuitry than that of asynchronous counters. Many synchronous (parallel) counters that are available as ICs are designed to be presettable, i.e. they can be preset to any desired starting count either asynchronously or synchronously. This presetting operation is also referred to as *loading* the counter.

6.3.4 Design of Synchronous Counters

For a systematic design of synchronous counters, the following procedure is used.

Step 1. *Number of flip-flops:* Based on the description of the problem, determine the required number n of the FFs—the smallest value of n is such that the number of states $N \leq 2^n$—and the desired counting sequence.

Step 2. *State diagram:* Draw the state diagram showing all the possible states. A state diagram, which can also be called the transition diagram, is a graphical means of depicting the sequence of states through which the counter progresses. In case the counter goes to a particular state from the invalid states on the next clock pulse, the same can also be included in the state diagram.

Step 3. *Choice of flip-flops and excitation table:* Select the type of flip-flops to be used and write the excitation table. An excitation table is a table that lists the present state (PS), the next state (NS) and the required excitations.

Step 4. *Minimal expressions for excitations:* Obtain the minimal expressions for the excitations of the FFs using the K-maps drawn for the excitations of the flip-flops in terms of the present states and inputs.

Step 5. *Logic diagram:* Draw a logic diagram based on the minimal expressions.

If the synchronous counter is a shortened-modulus counter it may suffer from the problem of lock-out. That is, the counter may not self-start. A self-starting counter is one that will eventually enter its proper sequence of states regardless of its initial state. The counter can be made self-starting by so designing it that it goes to a particular state whenever it enters an invalid state. The same procedure can be used for counters of any number of bits and any arbitrary sequence. The only restriction on the sequence is that, it cannot contain the same state more than once within one complete cycle before repeating itself.

In the case of an up-down counter the state diagram shows the relationship between the present state, the input, and the next state of the counter. In the case of an up-counter or down-counter the state diagram shows only the relationship between the present state and the next state because only one type of input is given.

The excitation tables of various flip-flops used in the counters are shown in Table 6.11.

Table 6.11 Excitation tables

PS	NS	Required inputs	
Q_n	Q_{n+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

(a) S-R FF excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

(b) J-K FF excitation table

(Contd.)

Table 6.11 Excitation tables *(Contd.)*

PS	NS	Required input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

(c) D FF excitation table

PS	NS	Required input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

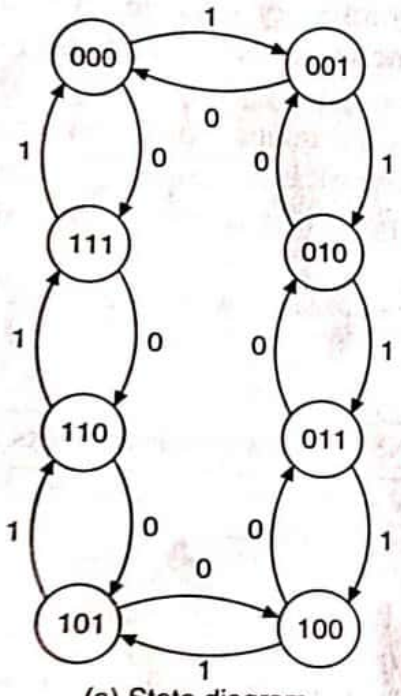
(d) T FF excitation table

6.3.5 Design of a Synchronous 3-bit Up-down Counter Using J-K FFs

Step 1. *Determine the number of flip-flops required:* A 3-bit counter requires three FFs. It has 8 states (000, 001, 010, 011, 100, 101, 110, 111) and all the states are valid. Hence no don't cares. For selecting up and down modes, a control or mode signal M is required. Let us say it counts up when the mode signal M = 1 and counts down when M = 0. The clock signal is applied to all the FFs simultaneously.

Step 2. *Draw the state diagram:* The state diagram of the 3-bit up-down counter is drawn as shown in Figure 6.69a.

Step 3. *Select the type of flip-flops and draw the excitation table:* JK flip-flops are selected and the excitation table of a 3-bit up-down counter using JK flip-flops is drawn as shown in Figure 6.69b.



(a) State diagram

PS			Mode	NS			Required excitations					
Q_3	Q_2	Q_1	M	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	1	1	1	1	×	1	×	1	×
0	0	0	1	0	0	1	0	×	0	×	1	×
0	0	1	0	0	0	0	0	×	0	×	×	1
0	0	1	1	0	1	0	0	×	1	×	×	1
0	1	0	0	0	0	1	0	×	×	1	1	×
0	1	0	1	0	1	1	0	×	×	0	1	×
0	1	1	0	0	1	0	0	×	×	0	×	1
0	1	1	1	1	0	0	1	×	×	1	×	1
1	0	0	0	0	1	1	×	1	1	×	1	×
1	0	0	1	1	0	1	×	0	0	×	1	×
1	0	1	0	1	0	0	×	0	0	×	×	1
1	0	1	1	1	1	0	×	0	1	×	×	1
1	1	0	0	1	0	1	×	0	×	1	1	×
1	1	0	1	1	1	1	×	0	×	0	1	×
1	1	1	0	1	1	0	×	0	×	0	×	1
1	1	1	1	0	0	0	×	1	×	1	×	1

(b) Excitation table

Figure 6.69 Synchronous 3-bit up-down counter.

Step 4. *Obtain the minimal expressions:* From the excitation table we can conclude that $J_1 = 1$ and $K_1 = 1$, because all the entries for J_1 and K_1 are either X or 1. The K-maps for J_3, K_3, J_2 and K_2

based on the excitation table and the minimal expressions obtained from them are shown in Figure 6.70.

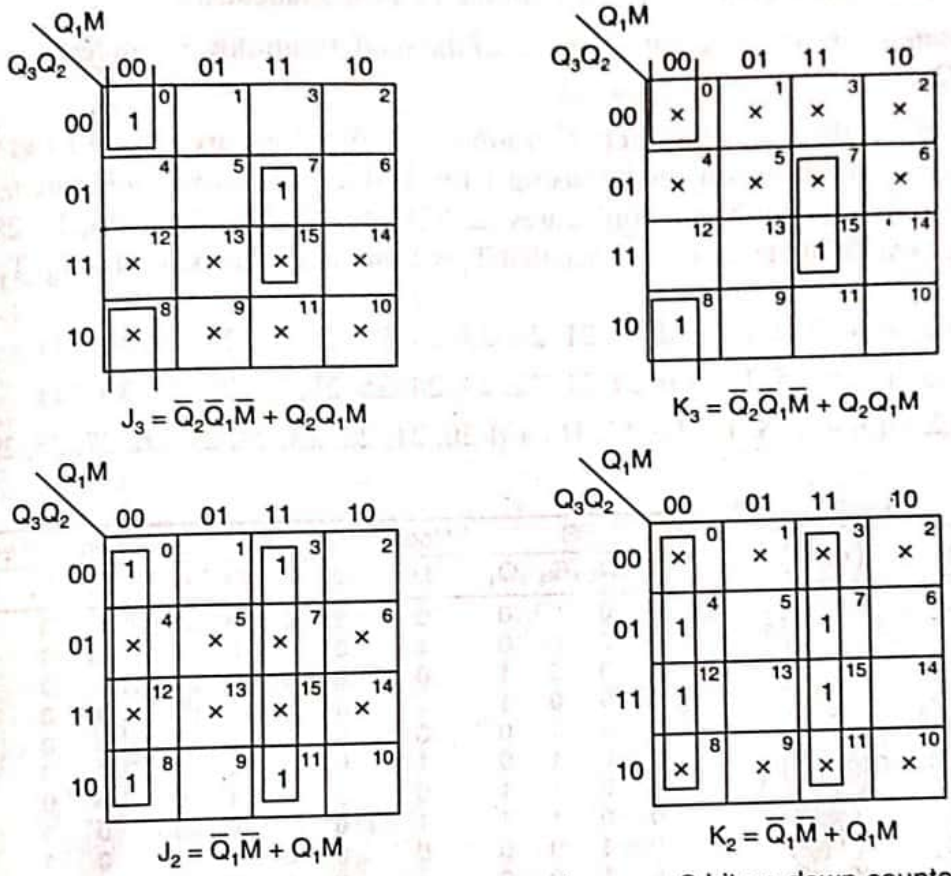


Figure 6.70 K-maps for excitations of synchronous 3-bit up-down counter.

Step 5. *Draw the logic diagram:* A logic diagram using those minimal expressions can be drawn as shown in Figure 6.71.

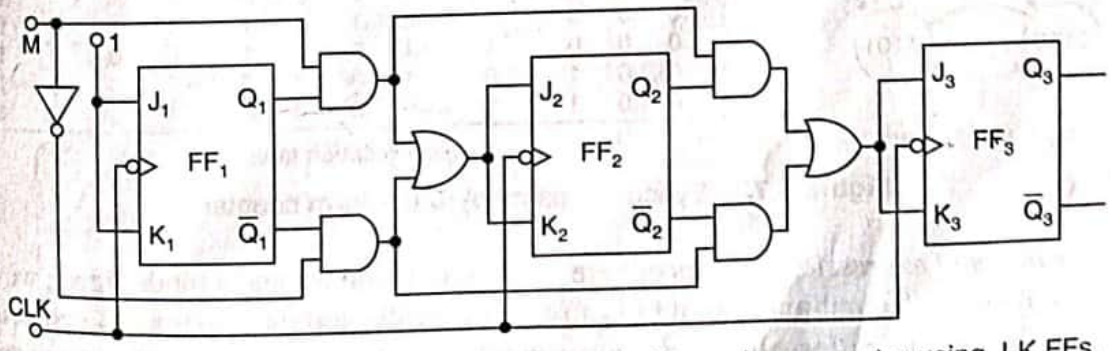


Figure 6.71 Logic diagram of the synchronous 3-bit up-down counter using J-K FFs.

6.3.6 Design of a Synchronous BCD Up-down Counter Using T FFs

Step 1. *The number of flip-flops:* A BCD counter is a mod-10 counter and has 10 states (0000 through 1001) and so it requires $n = 4$ FFs ($N \leq 2^n$, i.e. $10 \leq 2^4$). 4-FFs can have 16 states. So out of

16 states, six states (1010 through 1111) are invalid. For selecting up and down modes, a control or mode signal M is required. Let us say it counts up when the mode signal M = 1 and counts down when M = 0. The clock signal is applied to all the FFs simultaneously.

Step 2. *The state diagram:* The state diagram of the mod-10 up-down counter is drawn as shown in Figure 6.72a.

Step 3. *Type of flip-flops and the excitation table:* T flip-flops are selected and the excitation table of the modulo10 up-down counter using T FFs is drawn as shown in Figure 6.72b.

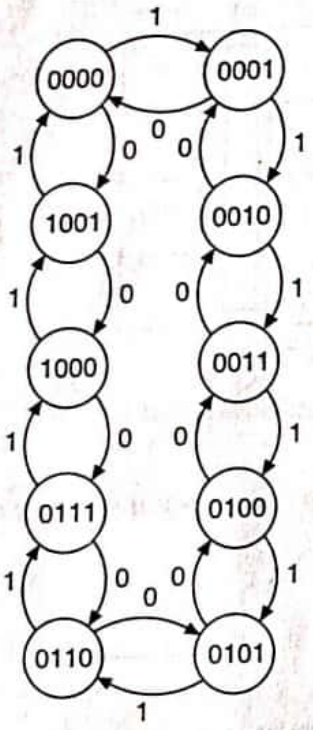
The remaining minterms are don't cares (Σ d(20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31))

From the excitation table we can see that $T_1 = 1$ and the expressions for T_4, T_3 and T_2 are as follows:

$$T_4 = \Sigma\, m(0,15,16,19) + d(20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)$$

$$T_3 = \Sigma\, m(7, 8, 15, 16) + d(20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)$$

$$T_2 = \Sigma\, m(3, 4, 7, 8, 11, 12, 15, 16) + d(20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)$$



(a) State diagram

PS				Mode	NS				Required excitations			
Q_4	Q_3	Q_2	Q_1	M	Q_4	Q_3	Q_2	Q_1	T_4	T_3	T_2	T_1
0	0	0	0	0	1	0	0	1	1	0	0	1
0	0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	0	1	0	0	1	1
0	0	1	0	1	0	0	1	1	0	0	0	1
0	0	1	1	0	0	0	1	0	0	0	0	1
0	0	1	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	0	0	1	1	0	1	1	1
0	1	0	0	1	0	1	0	1	0	0	0	1
0	1	0	1	0	0	1	0	0	0	0	0	1
0	1	0	1	1	0	1	1	0	0	0	1	1
0	1	1	0	0	0	1	0	1	0	0	1	1
0	1	1	0	1	0	1	1	1	0	0	0	1
0	1	1	1	0	0	1	1	0	0	0	0	1
0	1	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	0	0	1	0	0	1

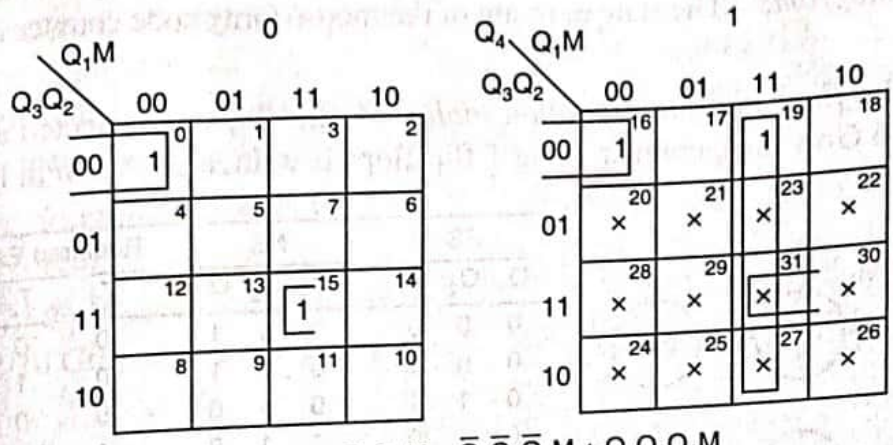
(b) Excitation table

Figure 6.72 Synchronous mod-10 up-down counter.

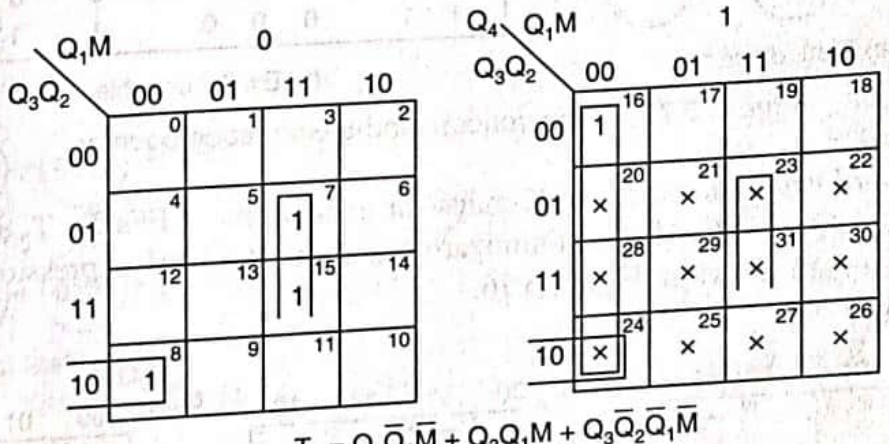
Step 4. *The minimal expressions:* Since there are 4 state variables and a mode signal, we require a 5-variable K-map. 20 combinations of $Q_4Q_3Q_2Q_1M$ are valid and the remaining 12 combinations are invalid. So the entries for excitations corresponding to those invalid combinations are don't cares. From the excitation table we observe that $T_1 = 1$ because all the entries for T_1 are 1. The K-maps for T_4 and T_3 based on the excitation table and the minimal expressions obtained from them are shown in Figure 6.73. Also drawing and minimizing the K-maps for T_2, we get

$$T_2 = Q_4\bar{Q}_1\bar{M} + \bar{Q}_4Q_1M + Q_2\bar{Q}_1\bar{M} + Q_3\bar{Q}_1\bar{M}$$

Step 5. The logic diagram: The logic diagram based on the above equations is shown in Figure 6.74.



$$T_4 = Q_4Q_1M + \bar{Q}_3\bar{Q}_2\bar{Q}_1M + Q_3Q_2Q_1M$$



$$T_3 = Q_4\bar{Q}_1\bar{M} + Q_2Q_1M + Q_3\bar{Q}_2\bar{Q}_1\bar{M}$$

Figure 6.73 K-maps for excitations T_4 and T_3 of a mod-10 up-down counter.

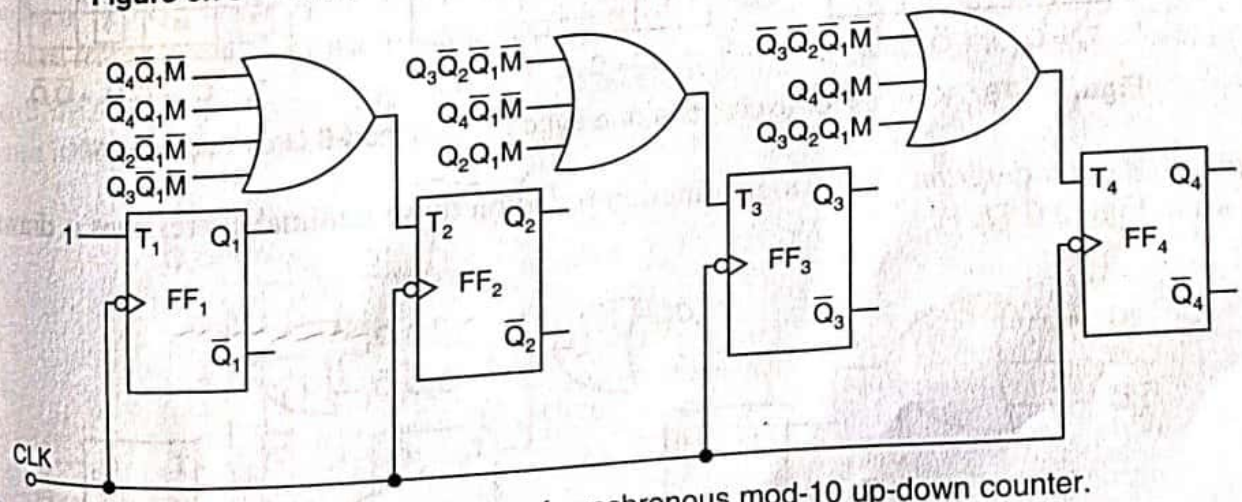


Figure 6.74 Logic diagram of synchronous mod-10 up-down counter.

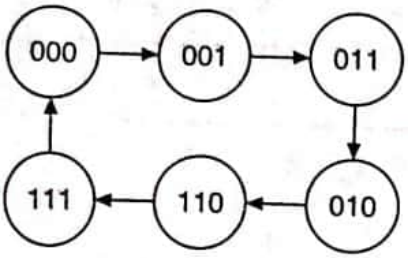
6.3.7 Design of a Synchronous Modulo-6 Gray Code Counter

Step 1. The number of flip-flops: We know that the counting sequence for a modulo-6 Gray code counter is 000, 001, 011, 010, 110 and 111. It requires $n = 3$ FFs ($N \leq 2^n$, i.e. $6 \leq 2^3$). 3 FFs can have

8 states. So the remaining two states 101 and 100 are invalid. The entries for excitations corresponding to invalid states are don't cares.

Step 2. *The state diagram:* The state diagram of the mod-6 Gray code counter is drawn as shown in Figure 6.75a.

Step 3. *Type of flip-flops and the excitation table:* T flip-flops are selected and the excitation table of the mod-6 Gray code counter using T flip-flops is written as shown in Figure 6.75b.



(a) State diagram

PS			NS			Required Excitations		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	T_3	T_2	T_1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

(b) Excitation table

Figure 6.75 Synchronous mod-6 Gray code counter.

Step 4. *The minimal expressions:* The K-maps for excitations of FFs T_3, T_2 and T_1 in terms of outputs of FFs Q_3, Q_2 and Q_1, their minimization and the minimal expressions for excitations obtained from them are shown in Figure 6.76.

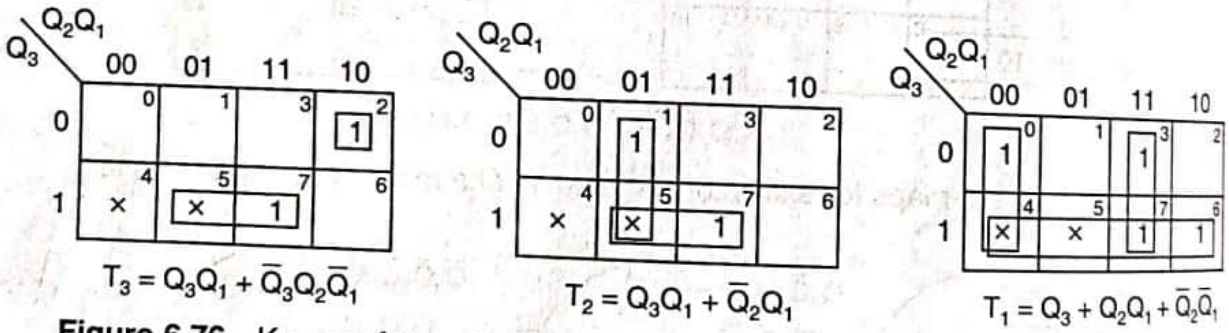


Figure 6.76 K-maps for excitations of a synchronous mod-6 Gray code counter.

Step 5. *The logic diagram:* The logic diagram based on those minimal expressions is drawn as shown in Figure 6.77.

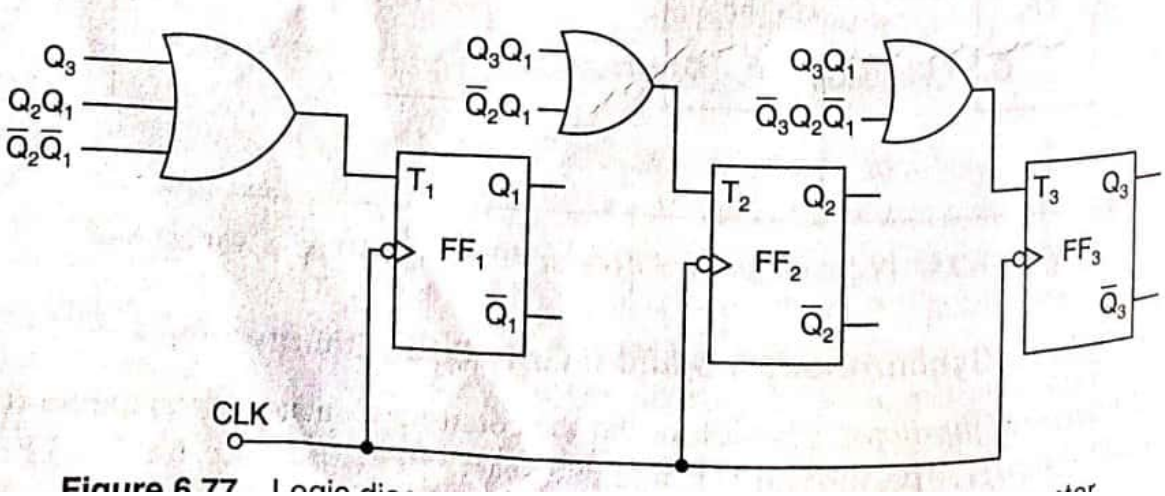


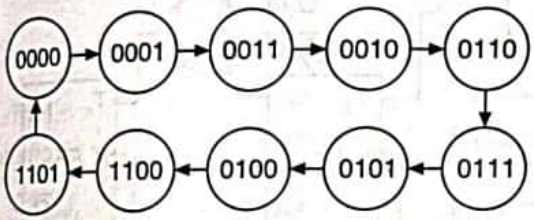
Figure 6.77 Logic dia... ...ter

6.3.8 Design of a Synchronous Modulo-10 Gray Code Counter

Step 1. The number of flip-flops: We know that the counting sequence for a modulo-10 Gray code counter is 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 0000.... It has 10 states and so it requires $n = 4$ FFs ($N \leq 2^n$, i.e. $10 \leq 2^4$). Four FFs can have 16 states. So the remaining six states 1111, 1110, 1010, 1011, 1001 and 1000 are invalid. The entries for excitations corresponding to these invalid states are don't cares.

Step 2. The state diagram: The state diagram of the mod-10 Gray code counter is drawn as shown in Figure 6.78a.

Step 3. The type of flip-flops and the excitation table: T flip-flops are selected and the excitation table of the mod-10 Gray code counter using T FFs is written as shown in Figure 6.78b.



(a) State diagram

PS				NS				Required Excitations			
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	T_4	T_3	T_2	T_1
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	1	0	0	1	0
0	0	1	1	0	0	1	0	0	0	0	1
0	0	1	0	0	1	1	0	0	1	0	0
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	0	1	0	1	0	0	1	0
0	1	0	1	0	1	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1	0	0	0
1	1	0	0	1	1	0	1	0	0	0	1
1	1	0	1	0	0	0	0	1	1	0	1

(b) Excitation table

Figure 6.78 Synchronous mod-10 Gray code counter.

Step 4. The minimal expressions: The K-maps for excitations of FFs T_4, T_3, T_2 and T_1 in terms of outputs of FFs Q_4, Q_3, Q_2 and Q_1, their minimization, and the minimal expressions for excitations obtained from them are shown in Figure 6.79.

Step 5. The logic diagram: The logic diagram based on those minimal expressions is drawn as shown in Figure 6.80.

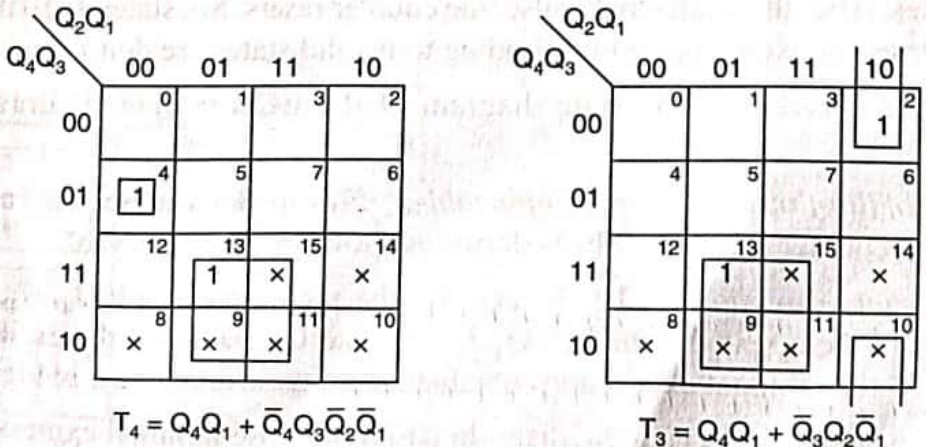


Figure 6.79 K-maps for excitations of a synchronous mod-10 Gray code counter *(Contd.)*

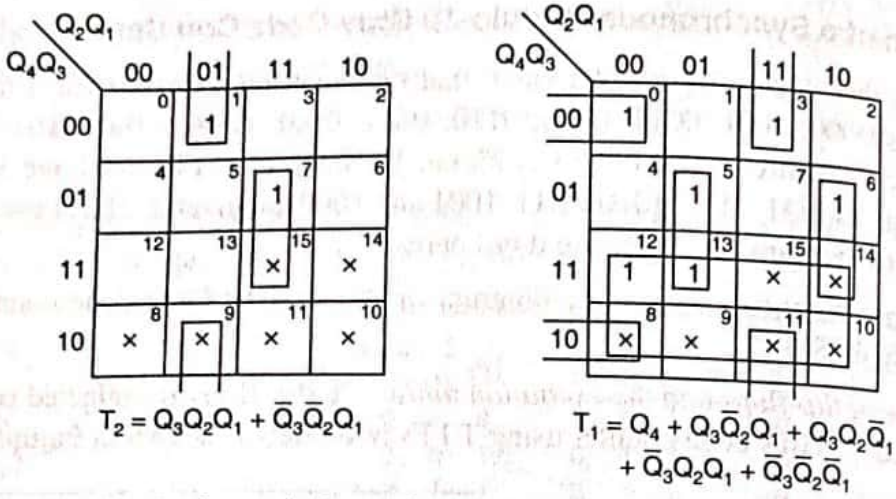


Figure 6.79 K-maps for excitations of a synchronous mod-10 Gray code counter.

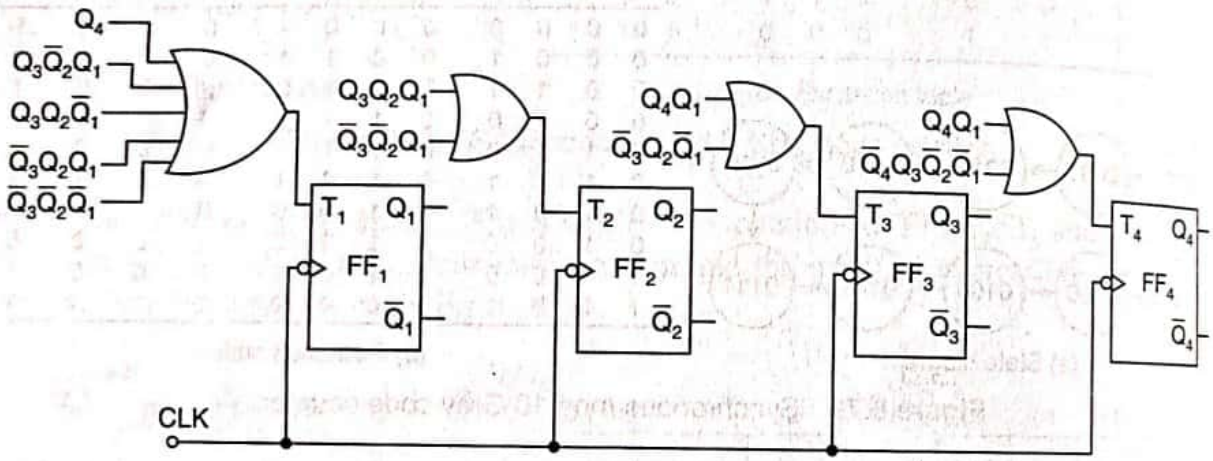


Figure 6.80 Logic diagram of synchronous mod-10 Gray code counter using T FFs.

6.3.9 Design of a Synchronous BCD Counter Using J-K FFs

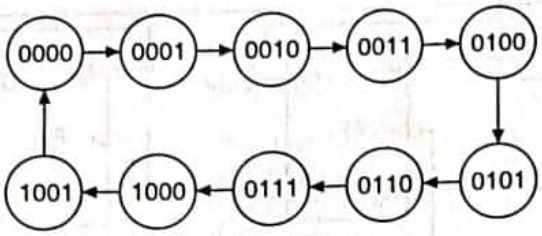
Step 1. *The number of flip-flops:* A BCD counter is nothing but a mod-10 counter. It is a decade counter. It has 10 states (0000 through 1001). It requires $n = 4$ FFs ($N \leq 2^n$, i.e. $10 \leq 2^4$). Four FFs can have 16 states. After the tenth clock pulse, the counter resets. So, states 1010 through, 1111 are invalid. The entries for excitations corresponding to invalid states are don't cares.

Step 2. *The state diagram:* The state diagram of the BCD counter is drawn as shown in Figure 6.81a.

Step 3. *The type of flip-flops and the excitation table:* JK flip-flops are selected and the excitation table of the BCD counter using J-K FFs is drawn as shown in Figure 6.81b.

Step 4. *The minimal expressions:* The K-maps for the excitations of FFs J_4, K_4, J_3, K_3, J_2, K_2, J_1 and K_1 in terms of the outputs of the FFs Q_4, Q_3, Q_2 and Q_1, based on the excitation table, their minimization and the minimal expressions obtained from them are shown in Figure 6.82.

Step 5. *The logic diagram:* The logic diagram based on those minimal expressions is drawn as shown in Figure 6.83.



(a) State diagram

PS				NS				Required excitations							
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	1	0	×	0	×	0	×	1	×
0	0	0	1	0	0	1	0	0	×	0	×	1	×	×	1
0	0	1	0	0	0	1	1	0	×	0	×	×	0	1	×
0	0	1	1	0	1	0	0	0	×	1	×	×	1	×	1
0	1	0	0	0	1	0	1	0	×	×	0	0	×	1	×
0	1	0	1	0	1	1	0	0	×	×	0	1	×	×	1
0	1	1	0	0	1	1	1	0	×	×	0	×	0	1	×
0	1	1	1	1	0	0	0	1	×	×	1	×	1	×	1
1	0	0	0	1	0	0	1	×	0	0	×	0	×	1	×
1	0	0	1	0	0	0	0	×	1	0	×	0	×	×	1

(b) Excitation table

Figure 6.81 Synchronous BCD (mod-10) counter.

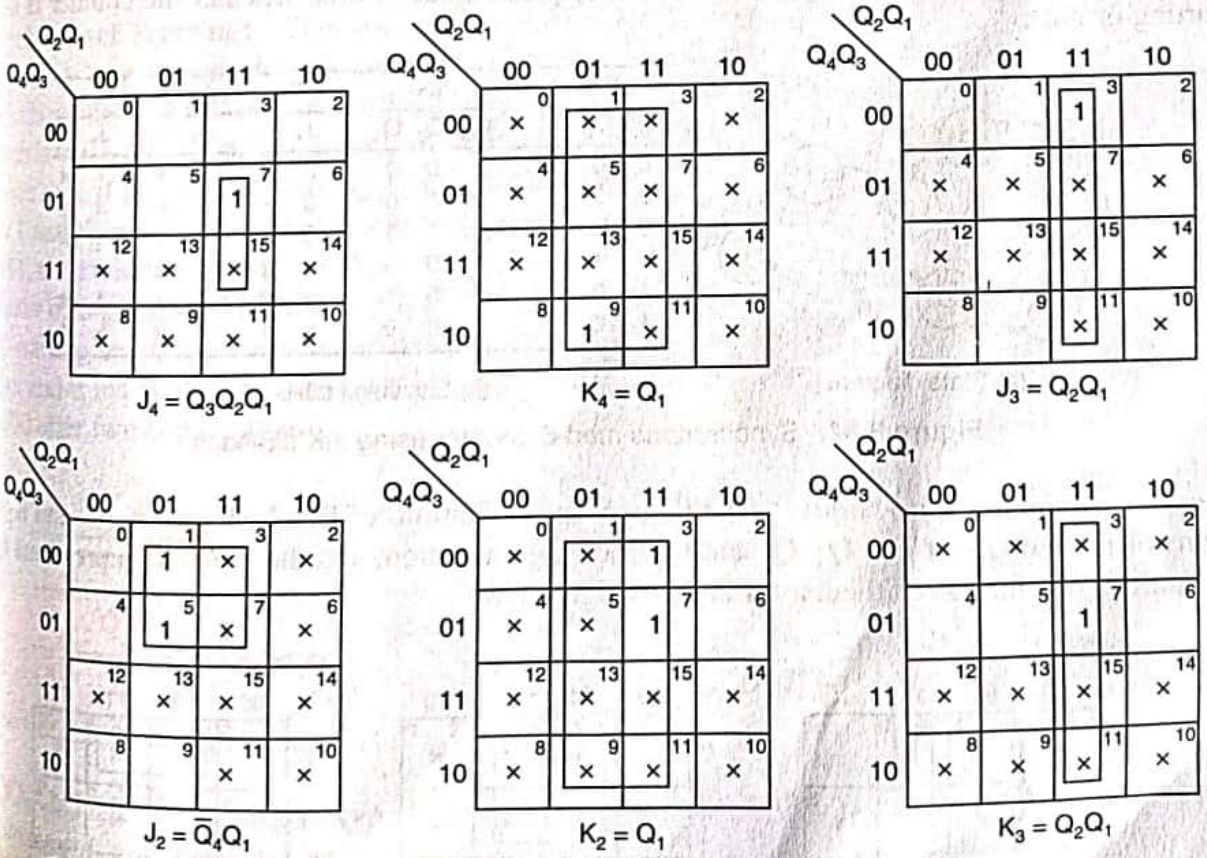


Figure 6.82 K-maps for excitations of synchronous BCD counter using J-K flip-flops.

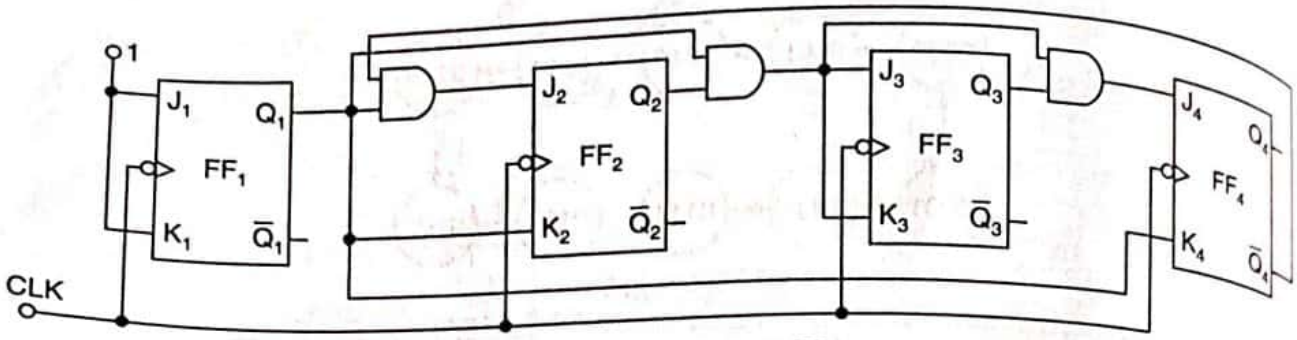


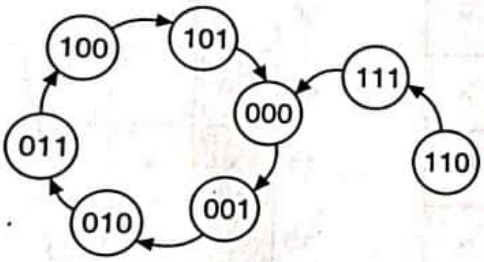
Figure 6.83 Logic diagram of synchronous BCD counter using J-K flip-flops.

6.3.10 Design of a Synchronous Mod-6 Counter Using J-K FFs

Step 1. *The number of flip-flops:* We know that the counting sequence for a mod-6 counter is 000, 001, 010, 011, 100, 101, 000, … . It has six states. So it requires $n = 3$ FFs ($N \leq 2^n$, i.e. $6 \leq 2^3$). Three FFs can have eight states. So the remaining two states 110 and 111 are invalid. The entries for excitations corresponding to invalid states are don't cares.

Step 2. *The state diagram:* The state diagram for the mod-6 counter is drawn as shown in Figure 6.84a.

Step 3. *The type of flip-flops and the excitation table:* JK flip-flops are selected and the excitation table of a mod-6 counter using J-K FFs is drawn as shown in Figure 6.84b. (States 110 and 111 can be removed from the state diagram if it is not required to determine whether the counter is self starting or not).



(a) State diagram

PS			NS			Required excitations					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
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	1	0	1	×	0	0	×	1	×
1	0	1	0	0	0	×	1	0	×	×	1

(b) Excitation table

Figure 6.84 Synchronous mod-6 counter using J-K flip-flops.

Step 4. *The minimal expressions:* The K-maps for excitations of FFs J_3, K_3, J_2, K_2, J_1 and K_1 in terms of the outputs of FFs Q_3, Q_2 and Q_1, their minimization, and the minimal expressions for excitations obtained from them are shown in Figure 6.85.

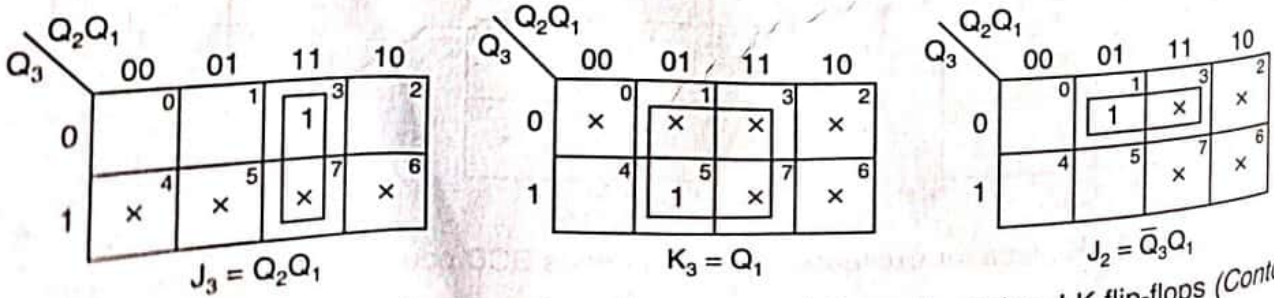


Figure 6.85 K-maps for excitations of synchronous mod-6 counter using J-K flip-flops *(Contd.)*

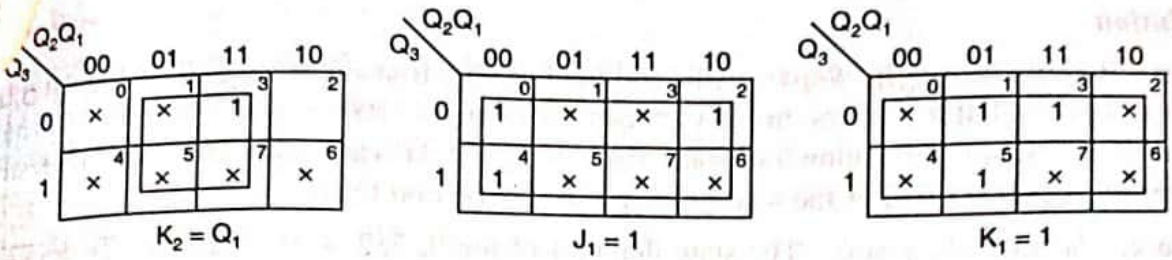


Figure 6.85 K-maps for excitations of synchronous mod-6 counter using J-K flip-flops.

Step 5. The logic diagram: The logic diagram based on those minimal expressions is drawn as shown in Figure 6.86.

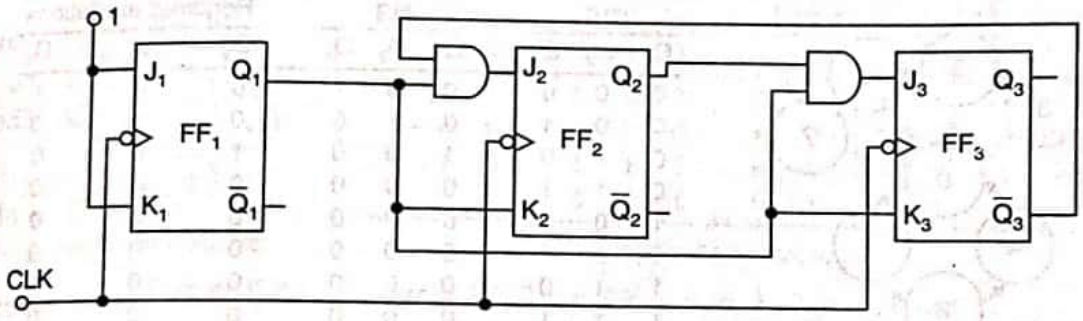


Figure 6.86 Logic diagram of synchronous mod-6 counter using J-K flip-flops.

Check for self starting: For this counter, 110 and 111 are the invalid states. We can determine the counter's sequence when its initial present state is 110 or 111. Given each present state, we can determine the J and K inputs from the logic diagram. With these inputs applied to the counter, which is in the present state, we can determine each of the next states of the counter. From Table 6.12 we see that, if the present state ($Q_3Q_2Q_1$) is 110, the excitations are $J_3 = 0$, $K_3 = 0$, $J_2 = 0$, $K_2 = 0$, $J_1 = 1$ and $K_1 = 1$. With these excitations, the counter changes from 110 to 111. If the present state is 111, the excitations are $J_3 = 1$, $K_3 = 1$, $J_2 = 0$, $K_2 = 1$, $J_1 = 1$ and $K_1 = 1$. With these excitations the counter changes from 111 to 000. From the table, to check for lock-out, we see that the counter is self-starting because if the counter initially finds itself in state 111, it goes to 000 after one clock pulse and if the counter is initially in state 110, then it goes to 111 after one clock pulse and goes to 000 after two clock pulses.

Table 6.12 Check for lock-out

PS			Present inputs						NS		
Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1	Q_3	Q_2	Q_1
1	1	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	0	0	0

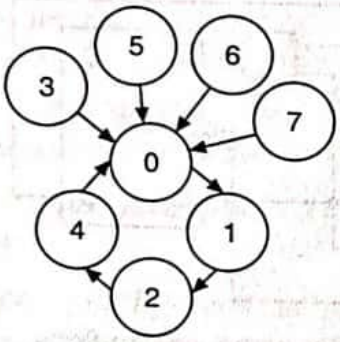
EXAMPLE 6.7 Design a type D counter that goes through states 0, 1, 2, 4, 0, … . The undesired (unused) states must always go to zero (000) on the next clock pulse.

Solution

Step 1. *The number of flip-flops:* This counter has only four stable states 0, 1, 2 and 4 (000, 001, 010, 100), but it requires three FFs because it counts 4 (100) as well. Three FFs can have eight states. So, the remaining four states (011, 101, 110, 111) are undesired. These undesired states must go to 000 after the next clock pulse. So, no don't cares.

Step 2. *The state diagram:* The state diagram of the 0, 1, 2, 4, 0, ... counter is drawn as shown in Figure 6.87a.

Step 3. *The type of flip-flops and the excitation table:* D flip-flops are selected and the excitation table of the counter using D FFs is written as shown in Figure 6.87b.



(a) State diagram

PS			NS			Required excitations		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	D_3	D_2	D_1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	1	0	0	1	0	0
0	1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

(b) Excitation table

Figure 6.87 Example 6.7: 0, 1, 2, 4, 0, ... counter.

Step 4. *The minimal expressions:* From the excitation table we can see that no minimization is possible. So the expressions for excitations read from the excitation table are:

$$D_3 = \bar{Q}_3 Q_2 \bar{Q}_1; \quad D_2 = \bar{Q}_3 \bar{Q}_2 Q_1; \quad D_1 = \bar{Q}_3 \bar{Q}_2 \bar{Q}_1$$

Step 5. *The logic diagram:* The logic diagram based on these expressions is drawn as shown in Figure 6.88. The counter is self-starting, i.e. it does not suffer from the problem of lock-out as may be seen from the state diagram.

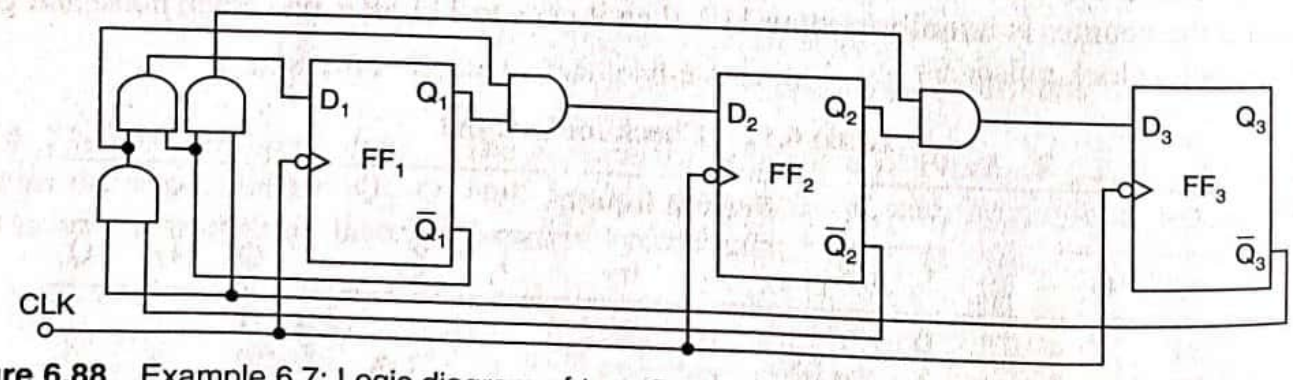


Figure 6.88 Example 6.7: Logic diagram of type D counter that goes through states 0, 1, 2, 4, 0, ...

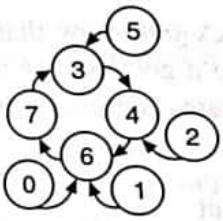
EXAMPLE 6.8 Design a J-K counter that goes through states 3, 4, 6, 7 and 3... . Is the counter self-starting? Modify the circuit such that whenever it goes to an invalid state it comes back to state 3.

Solution

Step 1. *The number of flip-flops:* The counter has only four states: 3, 4, 6, 7 (011, 100, 110, 111), but the maximum count is 7. So this counter requires three FFs ($7 \leq 2^3$). A counter using three flip-flops can have eight states. So the remaining four states (000, 001, 010, 101) are invalid. The entries for excitations corresponding to these invalid states are don't cares.

Step 2. *The state diagram:* The state diagram of the 3, 4, 6, 7, 3, ... counter is shown in Figure 6.89a.

Step 3. *The type of flip-flops and the excitation table:* JK flip-flops are selected and the excitation table of the counter using JK FFs is drawn as shown in Figure 6.89b.



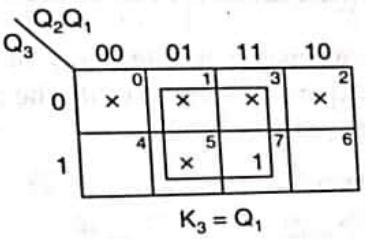
(a) State diagram

PS			NS			Required excitations					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	1	1	1	0	0	1	×	×	1	×	1
1	0	0	1	1	0	×	0	1	×	0	×
1	1	0	1	1	1	×	0	×	0	1	×
1	1	1	0	1	1	×	1	×	0	×	0

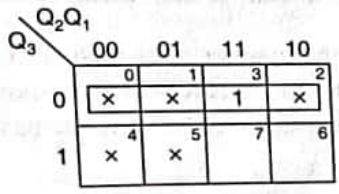
(b) Excitation table

Figure 6.89 Example 6.8: Type J-K 3, 4, 6, 7, 3, ... counter.

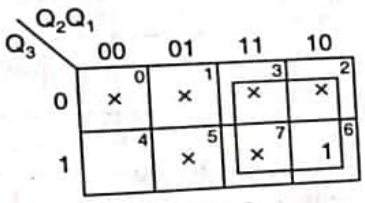
Step 4. *The minimal expressions:* From the excitation table, $J_3 = 1$ and $J_2 = 1$, since all the entries for J_3 and J_2 are either a 1 or a X. The K-maps for K_3, K_2, J_1 and K_1 based on the excitation table, their minimization, and the minimal expressions obtained from them are shown in Figure 6.90.



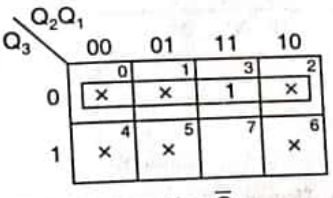
$K_3 = Q_1$



$K_2 = \overline{Q}_3$



$J_1 = Q_2$



$K_1 = \overline{Q}_3$

Figure 6.90 Example 6.8: K-maps for excitations of type J-K 3, 4, 6, 7, 3, ... counter.

Step 5. *The logic diagram:* The logic diagram for the counter based on those minimal expressions is drawn as shown in Figure 6.91.

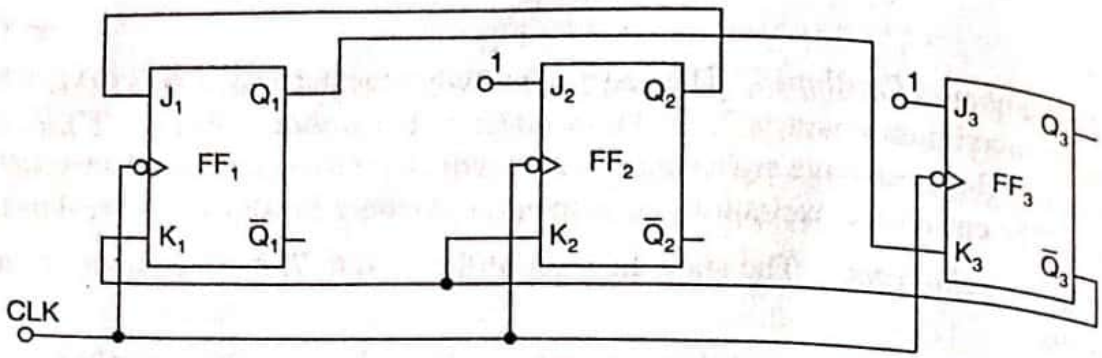


Figure 6.91 Example 6.8: Logic diagram of the J-K counter that goes through states 3, 4, 6, 7, 3, ...

Test for lock-out: The NS entries of Table 6.13 to check for lock-out show that there is no problem of lock-out and the counter is self-starting, because any time it goes into an invalid state, it comes out and goes into a useful state after one clock pulse. The state diagram of the counter is shown in Figure 6.89a.

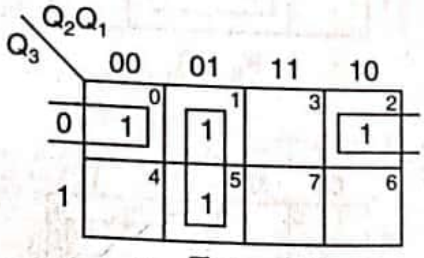
Table 6.13 Example 6.8: Check for lock-out

PS			Present inputs						NS		
Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1	Q_3	Q_2	Q_1
0	0	0	1	0	1	1	0	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	0
0	1	0	1	0	1	1	0	1	1	0	0
1	0	1	1	1	1	0	1	0	0	1	1

To see that the counter goes to state 3 (011) whenever it enters any of the invalid states, obtain an expression for reset R or set S as shown in Figure 6.92 and modify the circuit accordingly such that the counter goes to 011 after the next clock pulse.

PS			Reset
Q_3	Q_2	Q_1	R
0	0	0	1
0	0	1	1
0	1	0	1
1	0	1	1

(a) Truth table



$R = \bar{Q}_2 Q_1 + \bar{Q}_3 \bar{Q}_1$

(b) K-map

Figure 6.92 Example 6.8: Type J-K 3, 4, 6, 7, 3, ... counter.

Since we do not want the counter to reset to 000, and instead we want it to go to state 011, we apply the output to the clear terminal of FF_3 and to the preset terminals of FF_2 and FF_1 as shown in the modified circuit of Figure 6.93.

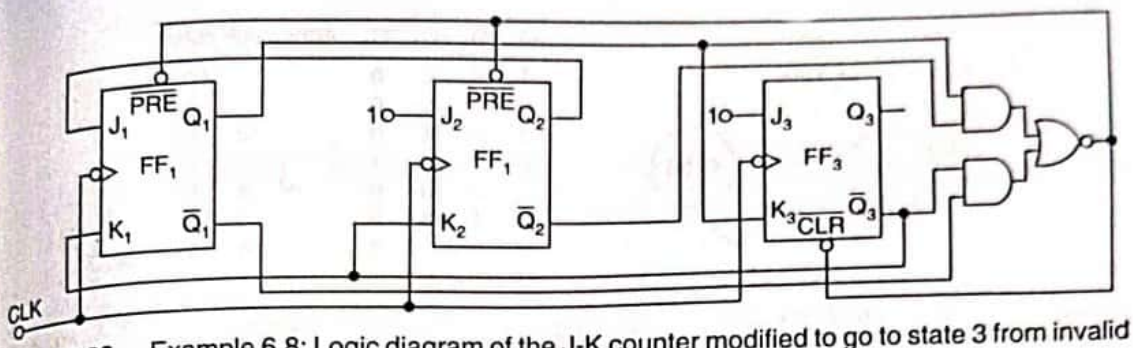


Figure 6.93 Example 6.8: Logic diagram of the J-K counter modified to go to state 3 from invalid states.

6.3.11 Shift Register Counters

One of the applications of shift registers is that they can be arranged to form several types of counters. Shift register counters are obtained from serial-in, serial-out shift registers by providing feedback from the output of the last FF to the input of the first FF. These devices are called counters because they exhibit a specified sequence of states. The most widely used shift register counter is the ring counter (also called the basic ring counter or the simple ring counter) as well as the twisted ring counter (also called the Johnson counter or the switch-tail ring counter).

Ring counter: This is the simplest shift register counter. The basic ring counter using D FFs is shown in Figure 6.94. The realization of this counter using J-K FFs is shown in Figure 6.95. Its state diagram and the sequence table are shown in Figure 6.96. Its timing diagram is shown in Figure 6.97. The FFs are arranged as in a normal shift register, i.e. the Q output of each stage is connected to the D input of the next stage, but the Q output of the last FF is connected back to the D input of the first FF such that the array of FFs is arranged in a ring and, therefore, the name *ring counter.*

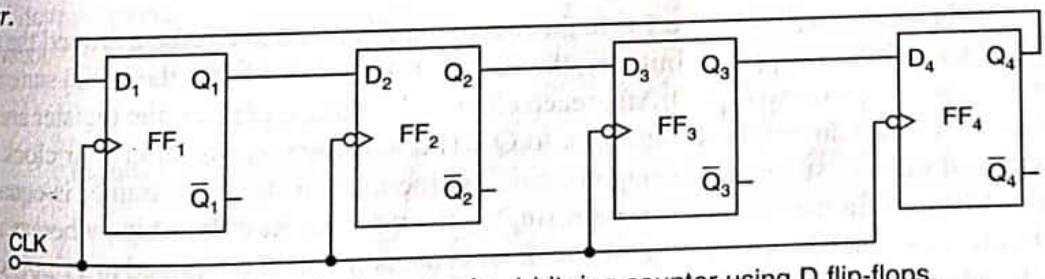


Figure 6.94 Logic diagram of a 4-bit ring counter using D flip-flops.

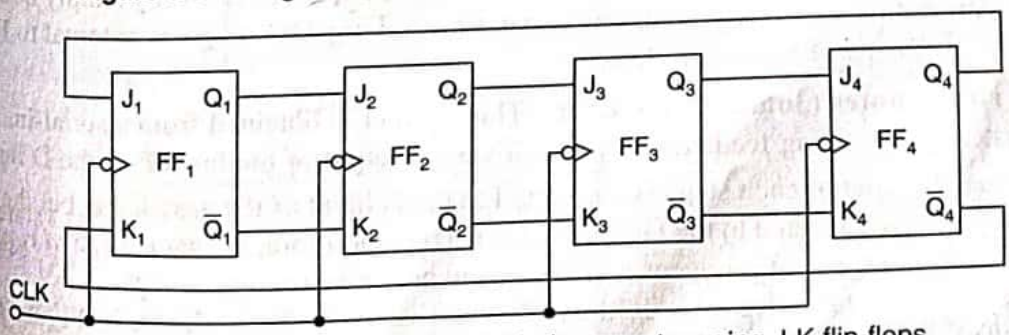
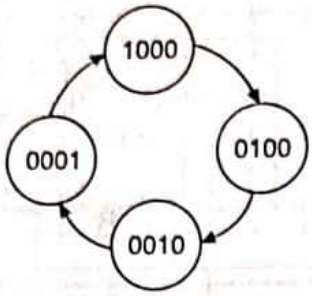


Figure 6.95 Logic diagram of a 4-bit ring counter using J-K flip-flops.



(a) State diagram

Q_1	Q_2	Q_3	Q_4	After clock pulse
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7

(b) Sequence table

Figure 6.96 State diagram and sequence table of a 4-bit ring counter.

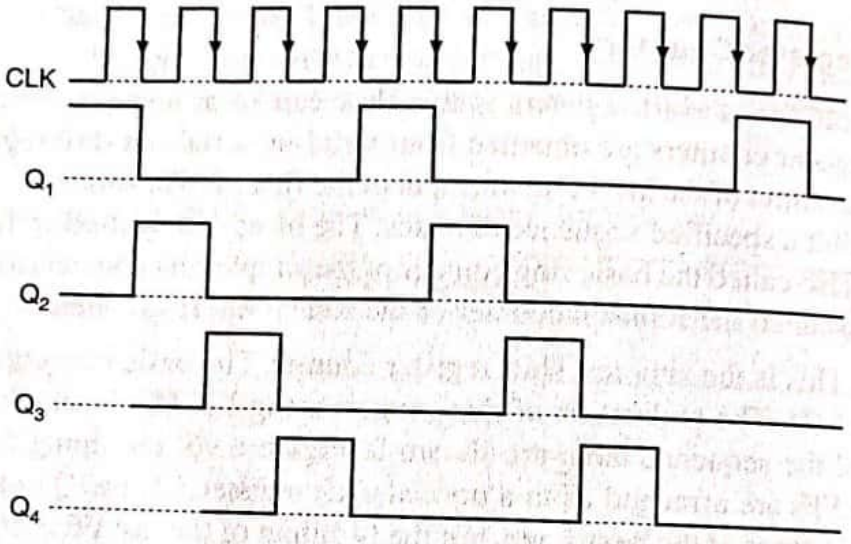


Figure 6.97 Timing diagram of a 4-bit ring counter.

In most instances, only a single 1 is in the register and is made to circulate around the register as long as clock pulses are applied. Initially, the first FF is preset to a 1. So, the initial state is 1000, i.e. $Q_1 = 1$, $Q_2 = 0$, $Q_3 = 0$ and $Q_4 = 0$. After each clock pulse, the contents of the register are shifted to the right by one bit and Q_4 is shifted back to Q_1. The sequence repeats after four clock pulses. The number of distinct states in the ring counter, i.e. the mod of the ring counter is equal to the number of FFs used in the counter. An n-bit ring counter can count only n bits, whereas an n-bit ripple counter can count 2^n bits. So, the ring counter is uneconomical compared to a ripple counter, but has the advantage of requiring no decoder, since we can read the count by simply noting which FF is set. Since it is entirely a synchronous operation and requires no gates external to FFs, it has the further advantage of being very fast.

Twisted ring counter (Johnson counter): This counter is obtained from a serial-in, serial-out shift register by providing feedback from the inverted output of the last FF to the D input of the first FF. The Q output of each stage is connected to the D input of the next stage, but the $\overline{Q}$ output of the last stage is connected to the D input of first stage, therefore, the name *twisted ring counter*. This feedback arrangement produces a unique sequence of states.

The logic diagram of a 4-bit Johnson counter using D FFs is shown in Figure 6.98. The realization of the same using J-K FFs is shown in Figure 6.99. The state diagram and the sequence table are shown in Figure 6.100. The timing diagram of a Johnson counter is shown in Figure 6.101.

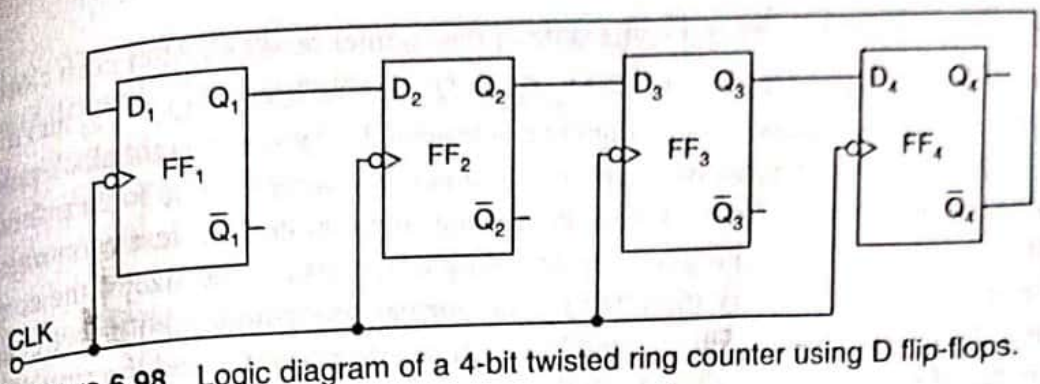


Figure 6.98 Logic diagram of a 4-bit twisted ring counter using D flip-flops.

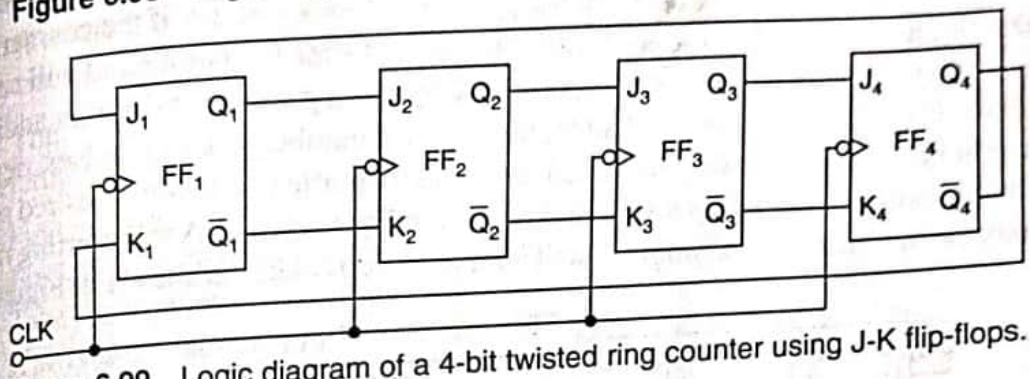
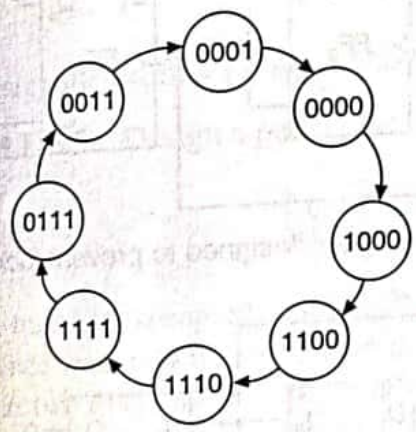


Figure 6.99 Logic diagram of a 4-bit twisted ring counter using J-K flip-flops.



(a) State diagram

Q_1	Q_2	Q_3	Q_4	After clock pulse
0	0	0	0	0
1	0	0	0	1
1	1	0	0	2
1	1	1	0	3
1	1	1	1	4
0	1	1	1	5
0	0	1	1	6
0	0	0	1	7
0	0	0	0	8
1	0	0	0	9

(b) Sequence table

Figure 6.100 State diagram and sequence table of a twisted ring counter.

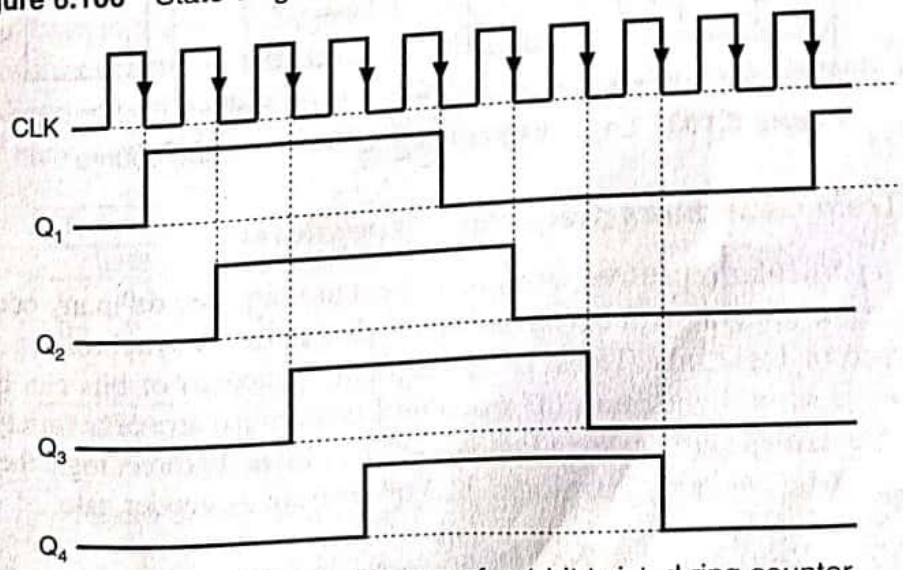


Figure 6.101 Timing diagram of a 4-bit twisted ring counter.

CHAPTER

5

PROGRAMMABLE LOGIC DEVICES AND THRESHOLD LOGIC

5.1 PROGRAMMABLE LOGIC DEVICES

Logic designers have a wide range of standard ICs available to them with numerous logic functions and logic circuit arrangements on a chip. In addition, these ICs are available from many manufacturers and at a reasonably low cost. For these reasons, designers have been interconnecting standard ICs to form an almost endless variety of different circuits and systems. Standard ICs (for example, Multiplexers, demultiplexers, decoders, encoders, adders, code converters, comparators, parity generators/checkers, ALUs, etc.) are also called fixed function ICs because each one of them performs a fixed digital operation.

However, there are some problems with circuit and system designs that use only standard ICs. Some system designs might require hundreds or thousands of these ICs.

Some of the disadvantages of this method are as follows:

1. Large board space requirements
2. Large power requirements
3. Lack of security
4. Additional cost, space, power requirements, etc. required to modify the design or to introduce more features

Usually the designs are too complex to be implemented using fixed function ICs.

To overcome the disadvantages of fixed-function ICs, *application specific integrated circuits* (ASICs) have been developed. The ASICs are designed by the users to meet the specific requirements of a circuit and are produced by an IC manufacturer as per the specifications supplied by the user.