out of 32 combinations two code gone for FS and LS. Remaining 30, can be given 'double meaning'. Thus we have total 60 (30 × 2) combinations, which are sufficient to represent 26 character, 10 digits and some special character. The Table 14 shows the ***Baudot Code***, which implements same fundamental of FS and LS (32 bit code).

1.12 Error Detecting and Correcting Code :

[ASKED IN EXAM - MAY 97, DEC. 98 III]

In digital system, the movement of binary data and codes from one location to another is most common operation performed. Examples are as follows :

(1) Communication between Computers.

(2) Storage and retrieval of data from floppy, hard disk and tape.

We normally pass data Information from one device (transmitter) to another device (Receiver). We expect that receiver should receive identical Information that was transmitted by transmitter. But the ideal situation doesn't occur because of ***electrical noise***, in the environment, normally EMI or RFI Interference. The noise is unwanted signal which, gets mixed up with the data and may change bit from 1 to 0 or 0 to 1. The change in single bit can change the meaning of Information / data. Fig. 1.18 shows the problem.

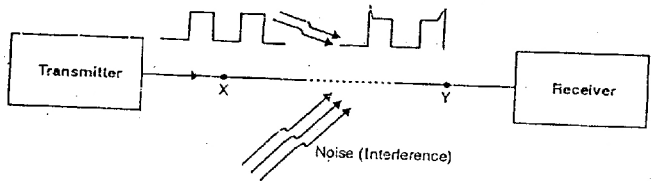


Fig. 1.18 : Transmission data corrupted by noise

As shown in Fig. 1.18, at transmitting end data is relatively noise free. While passing through cable (wire) certain degree of noise gets superimposed. The amplitude of noise is large enough to change the ***binary bits***, or the data bits will be wrongly interpreted by receiver end.

In modern digital equipment you will find that communication is relatively error free. But if data is corrupted we should have some means to ***Detect*** it. Simplest and most widely used scheme for error detection is parity.

1.13.1 Parity Bit :

[ASKED IN EXAM - DEC. 98 III]

A parity bit is an extra bit that is attached to a code group that is being transferred from one location to another. The parity bit can be either 0 or 1 depending upon number of 1's contained in code group. We have two types of parity.

(i) Even.

(ii) Odd.

Even Parity : In this case value of extra parity bit is chosen so that group of bits produces even number of 1's (Including parity bit), is called even parity. For example suppose data is 1 0 0 0 0 1 1 0. In this stream no. of ones are odd. ∴ Final data with parity will give us.

1 0 0 0 0 1 1 0, 1 } Data with parity (Total even number of 1's)
Data Parity

But if data is 1 0 0 0 0 1 0 0 i.e. already it has even number of 1's, then we have.

1 0 0 0 0 1 0 0, 0 } Data with parity (Total even number of 1's)
Data Parity

Odd Parity : In this case value of extra parity bit is chosen so that group of bits produces odd number of 1's (Including parity bit), is called odd parity. We will take same example.

1 0 0 0 0 1 1 0 ⟶ has odd number 1's, ∴

Finally we have 1 0 0 0 0 1 1 0, 0 } Total odd number of 1's.
Data Parity

For 1 0 0 0 0 1 0 0 ⟶ even number of 1's, ∴

Finally 1 0 0 0 0 1 1 0, 1 } Total odd number of 1's.
Data Parity

Both the parity scheme are commonly used, there is no any strong argument in favour of either type. When this method is used, transmitter and receiver both the station should be tuned to same parity i.e. either even or odd. The disadvantage of the system is if 2 or more number of bits are in error, the system fails.

Ex. 61 : Attach even parity bit to following ASCII codes :

(1) 'H' 1 0 0 1 0 0 0
(2) 'L' 1 0 0 1 1 0 0
(3) 'O' 1 0 0 1 1 1 1
(4) 'A' 1 0 0 0 0 0 1
(5) 'a' 1 1 0 0 0 0 1

Soln. :

ASCII	Parity bit	Code
'H'	0	1 0 0 1 0 0 0
'L'	1	1 0 0 1 1 0 0
'O'	1	1 0 0 1 1 1 1
'A'	0	1 0 0 0 0 0 1
'a'	1	1 1 0 0 0 0 1

To set even parity

1.13.2 Double Parity :

To improve the performance for detecting erroneous data, one may prefer ***double parity***. In previous case we have added parity bit horizontally only. But in double parity, parity bit is added horizontally as well as vertically. The example is as follows :

		Data	
	0	001	0010
horizontally	1	000	1101
calculated	1	111	0110
parity bit	0	101	0110
		011	1111

→ Vertically calculated parity bit

Parity bits added (presently it is set for even parity).

Here information is passed in blocks.

1.13.3 Checksum :

This method is most widely used in computer system. Checksum means successive words are added to generate sum. The final sum generated is called checksum.

A_1 ⟶ data byte/ word
+ A_2
+ A_3 (Total 'n' number of data bytes / words)
+ .
.
.
+ A_n

Checksum

For example take A_1 and A_2 bytes

A_1	0100	1000
+ A_2	0011	1100
Checksum	1000	0100

After transmitting block of data, checksum is transmitted, to receiver end. On the receiver side after receiving the data, it will add up and find out checksum, this checksum should exactly match with checksum transmitted by transmitter.

1.13.4 Five Bit Codes :

Five bit Codes are also available. Actually 4 bits are needed to encode any decimal digit from 0 to 9. An extra bit is required to allow us to decode the number more easily, as well as to detect errors more readily.

The Table 15 shows 5 bit codes such as 8 6 4 2 1, 6 3 2 1 0, shift counter and 5 1 1 1 1 code.

Decimal	86421	63210	Shift Counter	51111
0	00000	00110	00000	00000
1	00001	00011	00001	00001
2	00010	00101	00011	00011
3	00011	01001	00111	00111
4	00100	01010	01111	01111
5	00101	01100	11111	10000
6	01000	10001	11110	11000
7	01001	10010	11100	11100
8	10000	10100	11000	11110
9	10001	11000	10000	11111

Table 15

63210 : This code is weighted code except for decimal value equal to zero. There are exactly two, 1's in each code, allowing reliable error detection. This code is used for storing digital data on magnetic drum.

Shift Counter : This code is also called as Johnson Code. This code is non weighted code. Normally used in electronic counter.

51111 : This code is weighted code and self complementing. The code is approximately similar to shift counter.

1.13.5 Code with More than 5 Bits or Equal to 5 Bits :

The codes like 9876543210, 543210, 50–43210 or bi–quinary, 2 out of 5 are having more than or equal to 5 bits. Let's first see the Table for this codes.

Decimal	Ring Counter Code 9876543210	543210	50–43210	4 out of 5 code 74210
0	0000000001	000001	01–00001	11000
1	0000000010	000010	01–00010	00011
2	0000000100	000100	01–00100	00101
3	0000001000	001000	01–01000	00110
4	0000010000	010000	01–10000	01001
5	0000100000	100001	10–00001	01010
6	0001000000	100010	10–00010	01100
7	0010000000	100100	10–00100	10001
8	0100000000	101000	10–01000	10010
9	1000000000	110000	10–10000	10100

Table 16

9876543210 Code : This code is sometimes called ring counter code. This uses only single 1 in each group. This makes it easy to decode and detect error. The code has disadvantage that it requires more electronic circuitry than simple 4 and 5 bit code.

50 – 43210 : It is called bi–quinary code, occasionally used in electronic counter circuit. Each code contain group of 2 bits and group of 5 bits. The group of 2 bits Indicates, whether the number is more or less than 5. The group of 5 denotes count, Reliable error detection is possible because a single 1 in group of 2 bits and in 5 bits.

543210 : Same as biquinary code except 6 bits are used.

2 out of 5 : This code uses 5 bit and each code contain two 1s.

1.13.6 Error Correcting Codes :

➥ [ASKED IN EXAM - DEC. 96, MAY 96, 97, DEC. 97 !!!]

An efficient method would be not only to detect the code but also to correct it. The most popular error correcting code is the *Hamming Code*. It was discovered by R. W. Hamming in 1950.

The basic of hamming code is, it uses *Parity* bit, set for even parity over a selected bits. The parity bits are placed at bit positions which are powers of 2 i.e. 2^0, 2^1, 2^2, ... 2^n. The 7 bit hamming code format is given as follows :

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

Where D = Data bits ⟶ D_7 = MSB, D_3 = LSB. Total 4 data bits are present.

P = Parity bits ⟶ P_4, P_2, P_1 total 3 bits.

This is also called (7, 4) hamming code. Normally hamming code is represented by (n, k), where n = total number of bits and k = Total number of data bit(s).

m = n – k = Number of Parity bits.

In our case n = 7, k = 4, m = n – k = 7 – 4 = 3.

Conditions for parity bit :

(a) **P_1 bit :** This bit should be such that it establishes even parity over P_1, D_3, D_5 and D_7 bits.

P_1 ⟶ 1, 3, 5, 7.

(b) **P_2 bit :** This bit should be such that it establishes even parity ever P_2, D_3, D_6, D_7.

P_2 ⟶ 2, 3, 6, 7.

(c) **P_4 bit :** This bit should be such that it establishes even parity ever P_4, D_5, D_6, D_7 bits.

P_4 ⟶ 4, 5, 6, 7.

Let's see example to understand the thing more clearly. Say for example, generate 7 bit hamming code for given data bits 1 0 1 1 = $B_3\ B_2\ B_1\ B_0$, B_3 = MSB, B_0 = LSB.

7 bit hamming format ⟶

D_7	D_6	D_5	P_4	D_3	P_2	P_1
↓	↓	↓	↓	↓	↓	↓
B_3	B_2	B_1	P_4	B_0	P_2	P_1
↓	↓	↓	↓	↓	↓	↓
1	0	1	P_4	1	P_2	P_1

Now our job is to find out parity bits P_1, P_2 and P_4.

For P_1 bit

P_1	D_3	D_5	D_7	
–	1	1	1	As $D_3 = D_5 = D_7 = 1$ forms odd parity. P_1 = 1, to make it even.

For P_2 bit

P_2	D_3	D_6	D_7	
–	1	0	1	As D_3, D_6, D_7 already gives us even parity, we set P_2 = 0.

For P_4 bit

P_4	D_5	D_6	D_7	
–	1	0	1	As D_5, D_6, D_7 already forms even parity we set P_4 = 0.

∴ Finally we have 7 bit hamming code.

ns. :

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	0	1	0	1

We saw how to generate hamming code. Now we will see how hamming code *detects* error. Let's solve ample for the same.

Let's say on receiver end we received data bits 1011011. Which is hamming code. Find out the error d correct it.

Hamming Code Format $D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

1 0 1 1 0 1 1

(1) 1, 3, 5, 7 should have even parity.

$P_1\ D_3\ D_5\ D_7$

1 0 1 1 ⟶ these bits doesnot provide even parity ∴ there is error ∴ set error bit.

(2) 2, 3, 6, 7 should give even parity.

$P_2\ D_3\ D_6\ D_7$

1 0 0 1 ⟶ Yes, these bits gives even parties ∴ reset error bit.

(3) 4, 5, 6, 7 should give even parity.

$P_4\ D_5\ D_6\ D_7$

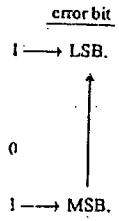
1 1 0 1 ⟶ No, this provides odd parity ∴ there is an error. ∴ set error bit

∴ error code is $(101)_2 = (5)_{10}$ ∴ 5th bit is in error. Complement it.

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

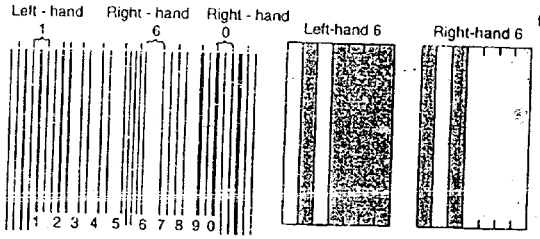
1 0 0 1 0 1 1 ⟶ Corrected Code.

Thus hamming provides you direct answer that which bit is in error.



1.14 Universal Product Code :

Universal product code is basically based on optical technique. Let's refer Fig. 1.19 to understand UPC.



(a) A 10 digit UPC symbol

(b) Representation of digit 6

Decimal digit	Left hand code	Right hand code
0	0001101	1110010
1	0011001	1100110
2	0010011	1101100
3	0111101	1000010
4	0100011	0011100
5	0110001	0001110
6	0101111	1010000
7	0111011	1000100
8	0110111	1001000
9	0001011	1110100

(c) Digit codes

Fig. 1.19

UPC 'symbol' contains decimal number uniquely identifying a product type. Fig. 1.19 (a) shows symbol that encodes 123456790. As shown decimal digit are represented in UPC symbol by light (0) and dark (1). UPC is composed of seven binary elements. UPC digits has two form, i.e. depending upon whether used in left half (first five digits) or the right half of the symbol, the code changes Fig. 1.19 (c) represents code for left and right. Certain other information is also provided in UPC code called sets of 'guard' bar, which appear at both ends and in the middle of the symbol.

Ex. 57 : Express decimal number 0 – 9 in following BCD codes : (i) 7421 (ii) 8421

Soln. :

Decimal	7421	8421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	1000	0111
8	1001	1000
9	1010	1001

Ex. 58 : Encode following four bit binary words into a seven bit even parity Hamming code format :
(i) 1001
(ii) 1010 **(June 99, 2 Marks)**

Soln. : Hamming code 7 bit data format is given as

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

(i) Given $D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$
$1\ 0\ 0\ P_4\ 1\ P_2\ P_1$

Selection should be such that we get even parity for

(a) $P_4\ D_5\ D_6\ D_7$
$-\ 0\ 0\ 1 \Rightarrow P_4 = 1$

(b) $P_2\ D_3\ D_6\ D_7$
$-\ 1\ 0\ 1 \Rightarrow P_2 = 0$

(c) $P_1\ D_3\ D_5\ D_7$
$-\ 1\ 0\ 1 \Rightarrow P_1 = 0$

$\therefore$ The 7 bit even parity Hamming Code is
$7\ 0\ 0\ 1\ 1\ 0\ 0$
$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

(ii) Given $D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$
$1\ 0\ 1\ P_4\ 0\ P_2\ P_1$

Selection should be such that we get even parity for

(a) $P_4\ D_5\ D_6\ D_7$
$-\ 1\ 0\ 1 \Rightarrow P_4 = 0$

(b) $P_2\ D_3\ D_6\ D_7$
$-\ 0\ 0\ 1 \Rightarrow P_2 = 1$

(c) $P_1\ D_3\ D_5\ D_7$
$-\ 0\ 1\ 1 \Rightarrow P_1 = 0$

$\therefore$ 7 bit hamming code is.
$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$
$1\ 0\ 1\ 0\ 0\ 1\ 0$

Ex. 59 : A 7 bit Hamming Code is received as 1111101. What is the correct data ?

Soln. : Given $D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$
$1\ 1\ 1\ 1\ 1\ 0\ 1$

The parity can be checked as follows, considering even parity.

(a) $P_4\ D_5\ D_6\ D_7$
$1\ 1\ 1\ 1$ No error 0 — For even parity P_4 must be 1, as $D_5 = D_6 = D_7 = 1$ results in odd number of 1's.

(b) $P_2\ D_3\ D_6\ D_7$
$0\ 1\ 1\ 1$ error 1 — For even parity P_2 must be 1 as $D_3 = D_6 = D_7 = 1$, results in odd number of 1's.

(c) $P_1\ D_3\ D_5\ D_7$
$1\ 1\ 1\ 1$ No error 0 — For even parity P_1 must be 1 as $D_3 = D_5 = D_7 = 1$, results in odd numbers of 1's.

$\therefore$ Resulting binary word is
$(010)_2 = (2)_{10}$ $\therefore$ 2nd bit is in error.

$\therefore$ Correct Code is 1111 111.

Ex. 60 : A seven bit even parity hamming code is received as 1110101. What is the correct code ?

Soln. : Given $D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$
$1\ 1\ 1\ 0\ 1\ 0\ 1$

Bits $P_4\ D_5\ D_6\ D_7$
$0\ 1\ 1\ 1$ error 1 — For even parity P_4 must be 1 as $D_5 = D_6 = D_7 = 1$, results in odd parity.

Bits $P_2\ D_3\ D_6\ D_7$
$0\ 1\ 1\ 1$ error 1 — For even parity P_2 must be 1 as $D_3 = D_6 = D_7 = 1$, results in odd parity.

Bits $P_1\ D_3\ D_5\ D_7$
$1\ 1\ 1\ 1$ No error 0 — $P_1 = 1$, as $D_3 = D_5 = D_7 = 1$ forms odd parity.

Resulting binary error word is $(110)_2$ $\therefore$ 6th bit is in error. $\therefore$ correct code is 1010101.

Ex. 61 : Encode data bits 1100 into seven bits even parity hamming code.

Soln. : The hamming code is,

$D_7\ D_6\ D_5\ D_4\ D_3\ P_2\ P_1$
$1\ 1\ 0\ P_4\ 0\ P_2\ P_1$ → data bits introduced

(a) P_4 set for even parity for bits 4, 5, 6, 7.

$P_4\ D_5\ D_6\ D_7$
$-\ 0\ 1\ 1 \Rightarrow P_4 = 0$

(b) P_2 set for even parity for bits 2, 3, 6, 7.

$P_4\ D_3\ D_5\ D_7$
$-\ 0\ 1\ 1 \Rightarrow P_2 = 0$

(c) P_1 set for even parity for bits 1, 3, 5, 7.

P_1	D_3	D_5	D_7	
–	0	0	1	$\Rightarrow P_1 = 1$

∴ Final code used is,

$D_7\ D_6\ D_5\ P_4\ P_3\ P_2\ P_1 = 1\ 1\ 0\ 0\ 0\ 0\ 1$

Ex. 62 : Determine the value of base x if,

(i) $(211)_x = (152)_8$. (ii) $(193)_x = (623)_8$. (iii) $(225)_x = (341)_8$.

Soln. : (i) $(211)_x = (152)_8$

This can be written as

$2 \times x^2 + 1 \times x + 1 \times x^0 = 1 \times 8^2 + 5 \times 8 + 2 \times 8^0$

$2x^2 + x + 1 = 64 + 40 + 2$

∴ $2x^2 + x + 1 = 106$.

∴ $2x^2 + x - 105 = 0$

Now solve the equation $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Where a = 2 b = 1 c = 105 Take +ve root.

Ans. $\boxed{x = 7}$

(ii) $(193)_x = (623)_8$

$1 \times x^2 + 9 \times x + 3 \times x^0 = 6 \times 8^2 + 2 \times 8^1 + 3 \times 8^0$

∴ $x^2 + 9x + 3 = 403$.

∴ $x^2 + 9x - 400 = 0$

Solving this we get $\boxed{x = 16}$

(iii) $(225)_x = (341)_8$.

$2 \times x^2 + 2 \times x^1 + 5 \times x^0 = 3 \times 8^2 + 4 \times 8^1 + 1 \times 8^0$

$2x^2 + 2x + 5 = 225$

∴ $2x^2 + 2x - 200 = 0$

$x^2 + x - 100 = 0$

Solve this using equation given.

∴ $\boxed{x = 10}$

Ex. 63 : Represent 8620 decimal number in (a) BCD 8421 (b) XS–3 (c) 2421 code.

Soln. : (a) BCD 8421

8	6	2	0	⟶ decimal
1000	0110	0010	0000	⟶ 8421 BCD.

(b) XS–3

8	6	2	0	⟶ decimal
+ 3	3	3	3	⟶ Add 3
11	9	5	3	
↓	↓	↓	↓	
1011	1000	0101	0011	⟶ XS–3

(c) 2421 Code.

8	6	2	0	⟶ Decimal
1110	1100	0010	0000	⟶ 2421 Code.

1.16 Examples from University Papers :

1. Determine equivalent decimal numbers for the following binary numbers :

(i) 1001.1101 (ii) 1100.0110 **(May 96, 2 Marks)**

Ans. :

(i)

Write binary number	→	1	0	0	1	·	1	1	0	1
Weights of number	→	2^3	2^2	2^1	2^0	·	2^{-1}	2^{-2}	2^{-3}	2^{-4}

$\therefore (1001 \cdot 1101)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$

$= 8 + 0 + 0 + 1 + 0.5 + 0.250 + 0 + 0.0625$

$= (9.8125)_{10}$

∴ $\boxed{(1001.1101)_2 = (9.8125)_{10}}$ ...Ans.

(ii) $(1100.0110)_2$

Write binary number	→	1	1	0	0	·	0	1	1	0
Weights of number	→	2^3	2^2	2^1	2^0	·	2^{-1}	2^{-2}	2^{-3}	2^{-4}

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

$= 8 + 4 + 0 + 0 + 0 + 0.250 + 0.125 + 0$

$= \boxed{(12.375)_{10}}$ ...Ans.

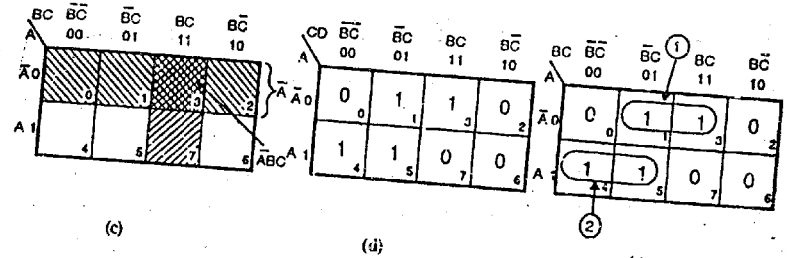
2. Convert following base 10 numbers to base 2 numbers : **(May 96, 2 Marks)**

(i) 524 (ii) 98

Ans. : (i) $(524)_{10}$

	524	→ Original number
	Quotient	Remainder
2	262	0 → LSB
2	131	0
2	65	1
2	32	1
2	16	0
2	8	0
2	4	0
2	2	0
2	1	0
		1 → MSB

$\boxed{(524)_{10} = (1000001100)_2}$ ...Ans.



(c) (d) (e)

Fig. Ex. 4.24 (iv)

Fig. Ex. 4.24 (iv)

(a) → for term $A\bar{B}$ → covers minterms 4,5

(b) → for term $\bar{A}\,\bar{B}\,C$ → covers minterms 1

(c) → for term $\bar{A}BC$ → covers minterms 3

(d) → Total of Fig. (a), (b) and (c) → covers 1, 3, 4, 5.

(e) → New K-map showing regrouping so the output function is minimised.

∴ Ouput $Y = A\bar{B} + \bar{A}C$

Draw K-map for following expression :

$Y = AB\bar{C}\bar{D} + \bar{B}\bar{C} + \bar{B}D + AB\bar{C}\bar{D} + \bar{B}C$ (June 99, 6 Marks)

. : Now the minterms for all terms are directly shown in Fig. Ex. 4.24 (v)(a). Fig. Ex. 4.24 (v)(b) s remapped K-map.

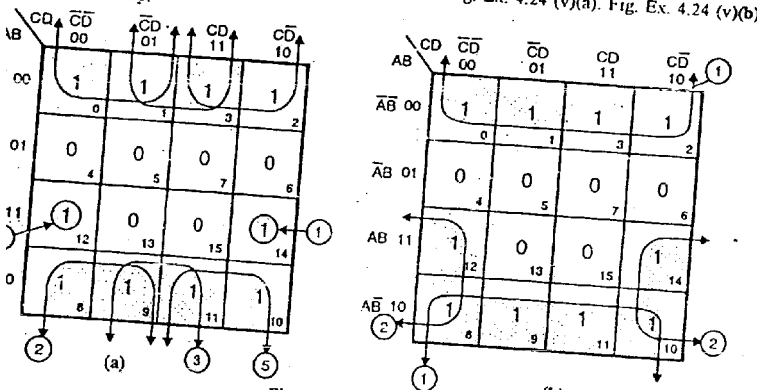


Fig. Ex. 4.24 (v)

Group 1 → $AB\bar{C}\bar{D}$

2 → $\bar{B}\,\bar{C}$

3 → $\bar{B}\,D$

4 → $AB\bar{C}\bar{D}$

5 → $\bar{B}\,C$

Newly mapped K-map,

Group 1 → $\bar{B}$

2 → $A\bar{D}$

∴ (Ouput) $Y = \bar{B} + A\bar{D}$

Conclusions about K-map :

At this stage, you must be feeling that K-map is easy method in comparison with, reduction (minimisation) using Boolean expression (hopefully). But the problems involved with K-map are :

(1) There is no guarantee that the K-map will give the best realisation. By mistake you can form redundant group.

(2) To deal with K-map for variables more than or equal to 6 is very difficult.

(3) Now-a-days it is trend that we expect that everything should be done by computer. The same way we expect computer should *reduce or minimise* the given expression, either by Boolean or K-map. But, as no unique recognitable pattern is generated, while solving a K-map, it is not possible to write an algorithm (Algorithm is nothing but systematic approach to solve any given problem).

Because of these problems associated with K-map, we use new method i.e. *Quine Mc Cluskey Method.*

4.9 Quine Mc Cluskey Method or Tabular Method :

It is an efficient tabular method employed to minimise an expression for six or more variables. Just like K-maps it searches for terms that cancel out the terms like $A + \bar{A}$, $x + \bar{x}$ and so on. Before I start explaining you steps involved, first we will learn *Prime Implicants.*

Prime Implicant :

Let P (x) be a product term i.e. Boolean function that can be written as a product of literals of X_0. If for the function F (x), the relationship for all A such that P (A) = 1, F (A) = 1. holds, then P is called an implicant of F. means F (A) = 1 implies F (A) = 1. it means that every minterm of P is also a minterm of F.

Let's take one example,

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

Remaining minterms not appearing are,

$ABC,\ \bar{A}BC,\ AB\bar{C},\ \bar{A}\bar{B}C$

An implicant P of F is called prime implicant if any product term obtained by deleting a literal from P is not an implicant of F. Prime implicant is an *indivisible* implicant in that it ceases to be an implicant if any of its literals are removed.

Code Word :

In the tabular (Quine Mc cluskey) method, a product term $x_1\ x_2\ \ldots\ x_n$ when n is equal to number of input variables is represented by word of length n. In that 1 in the position denotes uncomplemented literal

'0' denotes complemented literal ($\bar{x}_i$), and (X) dont care symbol indicates neither x_i nor $\bar{x}_i$. These words are called code words.

Say for example,

$1 \times 1 \times 0 \rightarrow x_1\, x_3\, \bar{x}_5$

$0\ 1\ 1\ 1\ 0 \rightarrow \bar{x}_1\, x_2\, x_3\, x_4\, \bar{x}_5$ and so on. Steps to be performed for tabular method.

Step I : List out all the minterms of given function f (x).

Step II : Group the minterms in Table format, based on number of logical '1's in the standard SOP term,

Step III : Start from top and go on comparing minterms, with remaining minterms, from each successive group.

Step IV : Combine the minterms, such that they differ by only one bit. The bit that differs is replaced by 'X'.

Step V : Using Step III again, reduce first reduction Table to second reduction Table and so on until no further reduction is possible.

Step VI : **Prime Implicant Selection :**

Set up a cover Table T whose rows are all the prime implicants and whose columns are all the minterms. Enter an 'X' at intersection of row i and column j if the j^{th} prime implicant P_i covers the j^{th} minterm of m_j.

Step VII : Now identify ***Essential prime implicants***. The essential prime implicants are easily identified by scanning cover Table for columns containing exactly one X. If such a column m_j is found and its sole X is in row P_i, then P_i is an essential prime implicant. If P_i is omitted from solution, m_j minterms will not be covered.

Step VIII : Identify all the essential rows of cover Table and add to the solution set S. Reduce cover Table by removing essential rows and all columns covered by these rows.

Step IX : Continue applying step VIII till no further reduction of cover Table is possible.

Let's solve one example to understand it better.

Find minimal SOP expression for the function :

$F(A, B, C, D, E) = \Sigma m\,(0, 6, 8, 10, 12, 14, 17, 19, 21, 25, 27, 28, 29)$.

Step I : Listing of all the minterm.

Decimal (minterms)	Equivalent Binary				
	B_4	B_3	B_2	B_1	B_0
m_0	0	0	0	0	0
m_6	0	0	1	1	0
m_8	0	1	0	0	0
m_{10}	0	1	0	1	0
m_{12}	0	1	1	0	0
m_{14}	0	1	1	1	0
m_{17}	1	0	0	0	1
m_{19}	1	0	0	1	1
m_{21}	1	0	1	0	1
m_{25}	1	1	0	0	1
m_{27}	1	1	0	1	1
m_{28}	1	1	1	0	0
m_{29}	1	1	1	0	1

Table 4.7

Step II : Grouping minterms in Table format, depending upon number of 1's. The Table 4.8 shows grouping.

Group	Minterm	Binary Equivalent				
		B_4	B_3	B_2	B_1	B_0
Group 1	m_0	0	0	0	0	0
Group 2	m_8	0	1	0	0	0
Group 3	m_6	0	0	1	1	0
	m_{10}	0	1	0	1	0
	m_{12}	0	1	1	0	0
	m_{17}	1	0	0	0	1
Group 4	m_{14}	0	1	1	1	0
	m_{19}	1	0	0	1	1
	m_{21}	1	0	1	0	1
	m_{25}	1	1	0	0	1
	m_{28}	1	1	1	0	0
Group 5	m_{27}	1	1	0	1	1
	m_{29}	1	1	1	0	1

Table 4.8

Step III : Comparing the minterms and prepare first reduction Table we will start comparing m_0 with remaining minterms. Initially compare m_0 and m_8.

		B_4	B_3	B_2	B_1	B_0
m_0	=	0	0	0	0	0
m_8	=	1	0	0	0	0

So if you observe B_0 to B_3 bits are same only B_4 changes. Therefore we can make entry of these both minterms in first reduction Table as,

$m_0, m_8 \rightarrow X\,0\,0\,0\,0$ (X means Not '1' or '0')

Same way go on comparing with 6, 10, 12, 17, 14, 19, 21, 25, 28, 27, 29. You won't find any match. So in group – 1 we have only one entry of m_0, m_8. Now start comparing m_8. Here you have to take precaution that you should compare present minterm with next minterms only. It should not be compared with previous minterm. Presently for us m_0 is previous and 8, 10, 12, ..., 29 are next. So don't compare m_8 with m_0.

While comparing m_8 with minterms you will find only two match

m_8, m_{10}	→	m_8	=	0 1 0 0 0		
		m_{10}	=	0 1 0 1 0	→	0 1 0 X 0
m_8, m_{12}	→	m_8	=	0 1 0 0 0		
		m_{12}	=	0 1 1 0 0	→	0 1 X 0 0

These two terms should be written under group 2.

Now group 3 will start.

Go comparing m_6, m_{10}, m_{12}, m_{17} and entries of all there will be covered under group 3. Table 4.9 shows full first reduction Table.

First Reduction Table

Group	Minterms	Binary Equivalent				
		B_4 (A)	B_3 (B)	B_2 (C)	B_1 (D)	B_0 (E)
Group – 1	m_0, m_8	0	X	0	0	0
Group – 2	m_8, m_{10}	0	1	0	X	0
	m_8, m_{12}	0	1	X	0	0
Group – 3	m_6, m_{14}	0	X	1	1	0
	m_{10}, m_{14}	0	1	X	1	0
	m_{12}, m_{14}	0	1	1	X	0
	m_{12}, m_{28}	X	1	1	0	0
	m_{17}, m_{19}	1	0	0	X	1
	m_{17}, m_{21}	1	0	X	0	1
	m_{17}, m_{25}	1	X	0	0	1
Group – 4	m_{19}, m_{27}	1	X	0	1	1
	m_{21}, m_{29}	1	X	1	0	1
	m_{25}, m_{27}	1	1	0	X	1
	m_{25}, m_{29}	1	1	X	0	1
	m_{28}, m_{29}	1	1	1	0	X

Table 4.9

tep IV : Now from ***First Reduction Table*** we are going to derive ***Second Reduction Table.***

First Reduction Table

Group	Minterm	Binary Equivalent					
		B_4 A	B_3 B	B_2 C	B_1 D	B_0 E	
1	m_0, m_8	0	X	0	0	0	→ P_A
2	m_8, m_{10}	0	1	0	X	0	√
	m_8, m_{12}	0	1	X	0	0	√
3	m_6, m_{14}	0	X	1	1	0	→ P_B
	m_{10}, m_{14}	0	1	X	1	0	√
	m_{12}, m_{14}	0	1	1	X	0	√
	m_{12}, m_{28}	X	1	1	0	0	→ P_C
	m_{17}, m_{19}	1	0	0	X	1	√
	m_{17}, m_{21}	1	0	X	0	1	√
	m_{17}, m_{25}	1	X	0	0	1	√
4	m_{19}, m_{27}	1	X	0	1	1	√
	m_{21}, m_{29}	1	X	1	0	1	√
	m_{25}, m_{27}	1	1	0	X	1	√
	m_{25}, m_{29}	1	1	X	0	1	√
	m_{28}, m_{29}	1	1	1	0	X	→ P_D

Table 4.10 (a)

Second Reduction Table

Group	Minterm	Binary Equivalent				
		B_4 A	B_3 B	B_2 C	B_1 D	B_0 E
2	m_6, m_{10}, m_{12}, m_{14}	0	1	X	X	0
	m_6, m_{12}, m_{10}, m_{14}	0	1	X	X	0
3	m_{17}, m_{19}, m_{25}, m_{27}	1	X	0	X	1
	m_{17}, m_{21}, m_{25}, m_{29}	1	X	X	0	1
	m_{17}, m_{25}, m_{19}, m_{27}	1	X	0	X	1
	m_{17}, m_{25}, m_{21}, m_{29}	1	X	X	0	1

Table 4.10 (b)

Here also the procedure is same, group i should be compared with group i + 1. In this comparing task is bit simple, first check that 'X' matches, if it matches then only go for comparing remaining bits. The term which is covered, simply put a tick. Table 4.10 shows both the Tables.

Final reduction from Table 4.10 (b) is,

Minterm	Binary Equivalent	
m_8, m_{10}, m_{12}, m_{14} m_8, m_{12}, m_{10}, m_{14}	0 1 X X 0	→ P_E
m_{17}, m_{19}, m_{25}, m_{27}, m_{17}, m_{25}, m_{19}, m_{27}	1 X 0 X 1	→ P_F
m_{17}, m_{21}, m_{25}, m_{23}, m_{17}, m_{25}, m_{21}, m_{29}	1 X X 0 1	→ P_G

Step V : Now take remaining terms from Table 4.10 (b) (unticked terms) and final terms from Table 4.11 and make ***Cover Table*** or ***Prime implicant Table.*** This Table has a row for every prime implicant and a column for every minterm of Z. An 'X' (cross) is entered at the intersection of row i and column i, if prime implicant P_i of row i covers minterm m_j of row j.

Cover Table :

P. I.	m_0	m_6	m_8	m_{10}	m_{12}	m_{14}	m_{17}	m_{19}	m_{21}	m_{25}	m_{27}	m_{28}	m_{29}
P_A	X		X										
P_B		X				X							
P_C					X							X	
P_D												X	X
P_E			X	X	X	X							
P_F							X	X		X	X		
P_G							X		X	X			X
			✔	✔	✔	✔	✔	✔	✔	✔	✔		✔

Table 4.11

$P_A = 0\,X\,0\,0\,0 \rightarrow (00000)_2 = (0)_{10}\ (01000)_2 = (6)_{10}$

$P_A = 0\,X\,0\,0\,0$ (X can be 0 or 1)

↓

$0\,0\,0\,0\,0 \rightarrow (0\,0\,0\,0\,0)_2$

$0\,1\,0\,0\,0 \rightarrow (0\,1\,0\,0\,0)_2$

$P_B = 0\,X\,1\,1\,0 \rightarrow (00110)_2 = (6)_{10}, \quad (01110)_2 = (14)_{10}$

$P_C = X\,1\,1\,0\,0 \rightarrow (01100)_2 = (12)_{10}, \quad (11100)_2 = (28)_{10}$

$P_D = 1\,1\,1\,0\,X \rightarrow (11100)_2 = (28)_{10}, \quad (11101)_2 = (29)_{10}$

$P_E = 0\,1\,X\,X\,0 \rightarrow (01000)_2, (01010)_2, (01100)_2, (01110)_2$

$P_F = 1\,X\,0\,X\,1 \rightarrow (10001)_2, (10011)_2, (11001)_2, (11011)_2$

$P_G = 1\,X\,X\,0\,1 \rightarrow (10001)_2, (10101)_2, (11001)_2, (11101)_2$

As shown in Table 4.11, the moment we select P_E, P_F and P_G all ticked minterms are covered. Therefore P_E, P_F, $P_G \rightarrow$ essential implicant. Now as we select P_E, P_F and P_G, the minterms covered by P_C are already covered therefore P_C is not required. Now the remaining minterms are, m_0, m_6 and m_{28}.

To cover these three minterms we have to select P_A, P_B and P_D.

∴ Final solution is

$\{P_A, P_B, P_D, P_E, P_F, P_G\} = \{0X000, 0X110, 1110X, 01XX0, 1X0X1, 1XX01\}$

Consider we have A, B, C, D, E variables where A = MSB and E = LSB.

∴ Ouput $Y = \bar{A}\bar{C}\bar{D}\bar{E} + \bar{A}CD\bar{E} + AB\bar{C}D + \bar{A}B\bar{E} + A\bar{C}E + A\bar{D}E$

('X' terms are not considered).

Ex. 25 : Minimise expression using Quine Mc Cluskey method. **(June 99)**

(i) $f(A, B, C, D) = \Sigma (1, 3, 5, 10, 11, 12, 13, 14, 15)$.

Soln. :

Step I : List all minterms :

Decimal value or minterm	Binary equivalent			
	A	B	C	D
m_1	0	0	0	1
m_3	0	0	1	1
m_5	0	1	0	1
m_{10}	1	0	1	0
m_{11}	1	0	1	1
m_{12}	1	1	0	0
m_{13}	1	1	0	1
m_{14}	1	1	1	0
m_{15}	1	1	1	1

Table 4.12

Step II : Group according to number of 1's.

Group	Minterms	Binary Equivalent			
		A	B	C	D
1	m_1	0	0	0	1
2	m_3	0	0	1	1
	m_5	0	1	0	1
	m_{10}	1	0	1	0
	m_{12}	1	1	0	0
3	m_{11}	1	0	1	1
	m_{13}	1	1	0	1
	m_{14}	1	1	1	0
4	m_{15}	1	1	1	1

Table 4.13

Step III : Form first reduction Table.

Group	Minterms	Binary Equivalent				
		A	B	C	D	
1	m_1, m_3	0	0	X	1	
	m_1, m_5	0	X	0	1	
2	m_3, m_{11}	X	0	1	1	
	m_5, m_{13}	X	1	0	1	
	m_{10}, m_{11}	1	0	1	X	√
	m_{12}, m_{14}	1	1	X	0	√
	m_{12}, m_{13}	1	1	0	X	√
	m_{10}, m_{14}	1	X	1	0	√
3	m_{11}, m_{15}	1	X	1	1	√
	m_{13}, m_{15}	1	1	X	1	√
	m_{14}, m_{15}	1	1	1	X	√

Table 4.14

Step IV : Now list second reduction Table :

Group	Minterms	Binary Equivalent			
		A	B	C	D
2	$m_{10}, m_{11}, m_{14}, m_{15}$	1	X	1	X
	$m_{12}, m_{14}, m_{13}, m_{15}$	1	1	X	X
	$m_{12}, m_{13}, m_{14}, m_{15}$	1	1	X	X
	$m_{10}, m_{14}, m_{11}, m_{15}$	1	X	1	X

Table 4.15

Finally reducing this Table 4.15, we get,

Minterms								A	B	C	D
m_{10},	m_{11},	m_{12},	m_{15},	m_{10},	m_{14},	m_{11},	m_{15},	1	X	1	X
m_{12},	m_{14},	m_{13},	m_{15},	m_{12},	m_{13},	m_{14},	m_{15},	1	1	X	X

Table 4.16

Now we have to consider remaining terms from Table 4.14 and final term from Table 4.16 and make cover Table.

Step V : Cover Table

P. I.	m_1	m_3	m_5	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}
	Minterms								
P_A = 00X1	X	X							
P_B = 0X01	X		X						
P_C = X011		X			X				
P_D = X101			X				X		
P_E = 1X1X				X	X			X	X
P_F = 11XX						X	X	X	X
				✓	✓	✓	✓	✓	✓

Table 4.17

P_E and P_F both are essential implicants. If we cover P_E and P_F, these terms will cover minterm 10, 11, 12, 13, 14, 15. Now only remaining terms are P_A, P_B, P_C and P_D. These covers minterms 1, 3, 5, 11 and 13. But as minterms 11 and 13 are already covered by P_E and P_F, therefore consider only minterms 1, 3 and 5.

So, *reduced cover Table.*

Prime Implicants	Minterms m_1	m_3	m_5
P_A = 00X1	X	X	
P_B = 0X01	X		X
P_C = X011		X	
P_D = X101			X

Table 4.18

Observing Table 4.18, we can conclude that if prime implicants P_A and P_B, both are selected, it will cover minterms 1, 3, 5. Therefore one need not consider P_C and P_D.

∴ Final reduction,

$F(P_A, P_B, P_E, P_F) = \{00X1, 0X01, 1X1X, 11XX\}$

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

(ii) Minimise expression using Quine–Mc Cluskey method.

$f(A, B, C, D) = \Sigma m(1, 2, 4, 8, 10, 11, 12, 13, 15)$

Soln. : First list out all the minterms.

Step I :

Minterms	Binary Equivalent A	B	C	D
m_1	0	0	0	1
m_2	0	0	1	0
m_4	0	1	0	0
m_8	1	0	0	0
m_{10}	1	0	1	0
m_{11}	1	0	1	1
m_{12}	1	1	0	0
m_{13}	1	1	0	1
m_{15}	1	1	1	1

Table 4.19

Step II : Arrange minterms depending upon number of 1's.

Group	Minterms	Binary Equivalent A	B	C	D
1	m_1	0	0	0	1
	m_2	0	0	1	0
	m_4	0	1	0	0
	m_8	1	0	0	0
2	m_{10}	1	1	1	1
	m_{12}	1	1	0	0
3	m_{11}	1	0	1	1
	m_{13}	1	1	0	1
	m_{15}	1	1	1	1

Table 4.20

Step III : Now prepare 1st reduction Table.

First Reduction Table

Group	Minterms	Binary Equivalent A	B	C	D		
Group 1	m_2, m_{10}	X	0	1	0	→	P_A
	m_4, m_{12}	X	1	0	0	→	P_B
	m_8, m_{10}	1	0	X	0	→	P_C
	m_8, m_{12}	1	X	0	0	→	P_D
Group 2	m_{10}, m_{11}	1	0	1	X	→	P_E
	m_{12}, m_{13}	1	1	0	X	→	P_F
Group 3	m_{11}, m_{15}	1	X	1	1	→	P_G
	m_{13}, m_{15}	1	1	X	1	→	P_H

Table 4.21

Further reduction is not possible.

Step IV : Cover Table :

	m_1	m_2	m_4	m_8	m_{10}	m_{11}	m_{12}	m_{13}	m_{15}
P_A = X010		X			X				
P_B = X100			X				X		
P_C = 10X0				X	X				
P_D = 1X00				X			X		
P_E = 101X					X	X			
P_F = 110X							X	X	
P_G = 1X11						X			X
P_H = 11X1								X	X
P_I = 0001	X								

Table 4.22

Here if we select P_A, P_B, P_C, P_E, P_H, P_I all minterms are covered.

Therefore final function is,

$F \{ P_A, P_B, P_C, P_E, P_H, P_I \} = \{ X010, X100, 10X0, 101X, 11X1, 0001 \}$

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

(iii) Simplify following using tabular method :

$F(A, B, C, D, E, F) = \sum m(20, 28, 52, 60)$.

Soln. :

Step I :

Minterms	Binary
m_{20}	0 1 0 1 0 0
m_{28}	0 1 1 1 0 0
m_{52}	1 1 0 1 0 0
m_{60}	1 1 1 1 0 0

Step II : Arrange them with respect to number of bits.

Group	Minterm	Binary
1	m_{20}	0 1 0 1 0 0
2	m_{28}	0 1 1 1 0 0
	m_{52}	1 1 0 1 0 0
3	m_{60}	1 1 1 1 0 0

Step III : First Reduction Method.

Minterms	Binary	
$m_{20}, m_{28},$	0 1 X 1 0 0	√
$m_{20}, m_{52},$	X 1 0 1 0 0	√
$m_{28}, m_{60},$	X 1 1 1 0 0	√
$m_{52}, m_{60},$	1 1 X 1 0 0	√

Step IV : Second Reduction Table.

Minterms	Binary
$m_{20}, m_{28}, m_{52}, m_{60}$	X 1 X 1 0 0
$m_{20}, m_{52}, m_{28}, m_{60}$	X 1 X 1 0 0

Step V : Final reduction gives us,

$m_{20}, m_{28}, m_{52}, m_{60}$ X 1 X 1 0 0

$\therefore F(A, B, C, D, E, F) = B\ D\ \bar{E}\ \bar{F} \rightarrow$ Answer

---

(iv) Reduce using Quine Mc Cluskey method :

$f(A, B, C, D) = \sum m(2, 3, 6, 7, 8, 9, 13, 15) + d(4, 10, 12)$.

Soln. : List all the minterms.

Minterms	Binary			
	A	B	C	D
m_2	0	0	1	0
m_3	0	0	1	1
m_6	0	1	1	0
m_7	0	1	1	1
m_8	1	0	0	0
m_9	1	0	0	1
m_{13}	1	1	0	1
m_{15}	1	1	1	1
m_4	0	1	0	0
m_{10}	1	0	1	0
m_{12}	1	1	0	0

Now list all minterms with respect to number of 1's.

Group	Minterm	Binary				Number of 1's
		A	B	C	D	
group 1	m_2	0	0	1	0	1
	m_4	0	1	0	0	
	m_8	1	0	0	0	
group 2	m_3	0	0	1	1	2
	m_6	0	1	1	0	
	m_9	1	0	0	1	
	m_{10}	1	0	1	0	
	m_{12}	1	1	0	0	
group 3	m_7	0	1	1	1	3
	m_{13}	1	1	0	1	
group 4	m_{15}	1	1	1	1	4

First level implicants (Reduction Table)

Minterm	Binary					
	A	B	C	D		
$m_2, m_3,$	0	0	1	X	√	
$m_2, m_6,$	0	X	1	0	√	
$m_2, m_{10},$	X	0	1	0	→	P_A
$m_4, m_5,$	0	1	X	0	→	P_B
$m_4, m_{12},$	X	1	0	0	→	P_C
$m_8, m_9,$	1	0	0	X	√	
$m_8, m_{10},$	1	0	X	0	→	P_D
$m_8, m_{12},$	1	X	0	0	√	
$m_3, m_7,$	0	X	1	1	√	
$m_6, m_7,$	0	1	1	X	√	
$m_9, m_{13},$	1	X	0	1	√	
$m_{12}, m_{13},$	1	1	0	X	√	
$m_7, m_{15},$	X	1	1	1	→	P_E
$m_{13}, m_{15},$	1	1	X	1	→	P_F

Second Reduction Table

Minterms				Binary Equivalent			
$m_2,$	$m_3,$	$m_6,$	$m_7,$	0	X	1	X
$m_2,$	$m_6,$	$m_3,$	$m_7,$	0	X	1	X
$m_8,$	$m_9,$	$m_{12},$	$m_{13},$	1	X	0	X
$m_8,$	$m_{12},$	$m_9,$	$m_{13},$	1	X	0	X

Now list out unticked item in first level implicant Table and rule prime implicant i.e. form cover le.

Prime Implicants	m_2	m_3	m_4	m_5	m_7	m_8	m_9	m_{10}	m_{12}	m_{13}	m_{15}
P_A = X010	X							X			
P_B = 01X0			X	X							
P_C = X100			X						X		
P_D = 10X0						X		X			
P_E = X111					X						X
P_F = 11X1										X	X
P_G = 0X1X	X	X		X	X						
P_H = 1X0X						X	X		X	X	
	✔	✔		✔	✔	✔	✔		✔	✔	

If we select P_G and P_H we almost cover m_2, m_3, m_6, m_7, m_8, m_9, m_{12}, m_{13}.

Therefore reduced cover Table is,

	m_4	m_{10}	m_{15}
P_A = X010		X	
P_B = 01X0	X		
P_C = X100	X		
P_D = 10X0		X	
P_E = X111			X
P_F = 11X1			X

Therefore we select P_A, P_B and P_E.

$F\{P_A, P_B, P_E, P_G, P_H\} = \{X010, 01X0, X111, 0X1X, 1X0X\}$

But as m_4 and m_{10} are ***don't care conditions*** we need not cover it, therefore from reduced cover Table we select ONLY P_F.

Therefore finally, $F\{P_F, P_G, P_H\} = \{11X1, 0X1X, 1X0X\}$

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

Note : If we select PE instead of PF ; it will also do.

4.10 Examples from University Papers :

1. Simplify the following expression using K-map and realise using AND-OR realisation :

$f(A, B, C, D) = \sum m(1, 3, 7, 8, 10, 12, 13, 15)$ **(May 96, 6 Marks)**

Ans.: Refer Section 4.6.3, Ex. 11.

2. Obtain K–map for :

$f(A, B, C, D) = A\bar{B} + BC + C\bar{D}$ **(May 96, 6 Marks)**

Ans.: Refer Section 4.8, Ex. 24 (ii).

3. Minimize the following expression using Quine McCluskey tabular method.

$f(A, B, C, D) = \sum m(2, 9, 10, 11, 13, 15)$ **(May 96, 10 Marks)**

Ans. : First list out all minterms.

Step I :

Minterms	Binary equivalent			
	A	B	C	D
m_2	0	0	1	0
m_9	1	0	0	1
m_{10}	1	0	1	0
m_{11}	1	0	1	1
m_{13}	1	1	0	1
m_{15}	1	1	1	1

Table 1

Step II : Arrange minterms depending upon number of 1's : From Table 1

Group	Minterms	Binary equivalent			
		A	B	C	D
1	m_2	0	0	1	0
2	m_9	1	0	0	1
	m_{10}	1	0	1	0
3	m_{11}	1	0	1	1
	m_{13}	1	1	0	1
4	m_{15}	1	1	1	1

Table 2

Step III : Now prepare first reduction table

First Reduction Table

Group	Minterms	Binary equivalent				
		A	B	C	D	
1	m_2, m_{10}	X	0	1	0	$\rightarrow P_A$
2	m_9, m_{11}	1	0	X	1	√
	m_9, m_{13}	1	X	0	1	√
	m_{10}, m_{11}	1	0	1	X	$\rightarrow P_B$
3	m_{11}, m_{15}	1	X	1	1	√
	m_{13}, m_{15}	1	1	X	1	√

Table 3

Step IV : Now list second reduction table from Table 3

Second Reduction Table

Group	Minterms	Binary equivalent				
		A	B	C	D	
2	$m_9, m_{11}, m_{13}, m_{15}$	1	X	X	1	$\rightarrow P_C$
	$m_9, m_{13}, m_{11}, m_{15}$	1	X	X	1	

Table 4

Step V : Now we have to consider remaining terms from Table 3 (unticked marks and final term from Table 4 and make cover table or prime implicant table.

Prime Implicants						Minterms m_2	m_9	m_{10}	m_{11}	m_{13}	m_{15}
P_A	X	0	1	0		X		X			
P_B	1	0	1	X					X		
P_C	1	X	X	1			X		X	X	X
							✓		✓	✓	✓

P_C is the essential implicant. If we cover P_C, it will cover minterms 9, 11, 13, 15. Now remaining terms are P_A and P_B. These covers minterms 2, 10, 11. But minterm 11 is already covered by P_C therefore consider only minterm 2 and 10 by selecting P_A.

$F(P_A, P_C) = \{X\ 0\ 1\ 0,\ 1\ X\ X\ 1\}$

$\therefore$ $\boxed{F(A, B, C, D) = \bar{B}\bar{C}\bar{D} + AD}$ **...Ans.**

4. Simplify the following expression using K–map and realise using minimum number of 2–input gates. **(Dec. 96, 6 Marks)**

$f(A, B, C, D) = \sum m(1, 2, 9, 10, 11, 14, 15)$

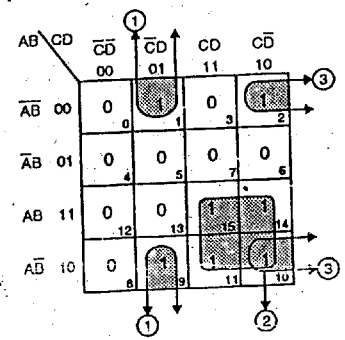
Ans. : As highest state = 15

Therefore K–map required will be of 4 variables (A, B, C, D).

Step 1 : First draw basic skeleton of K–map.

Step 2 : Presently box is empty. Write down '1' in the box whose number is specified in the example i.e. 1, 2, 9, 10, 11, 14,15.

Remaining boxes of K–map should be filled up with zeros.



After making different groups of ones. Write down output equation

$$Y = \underset{(1)}{\overline{B}\overline{C}D} + \underset{(2)}{AC} + \underset{(3)}{\overline{B}C\overline{D}} \quad \ldots(1)$$

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

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

Implementing above using Equation (1)

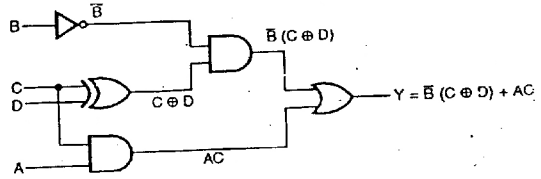


Fig. Q. 4.4

5. Obtain K–map for the following function : **(Dec. 96, 6 Marks)**

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

Ans.: Refer Section 4.8, Ex. 24 (iii).

6. Minimise the following expression using Quine McCluskey tabular method : **(Dec. 96, 10 Marks)**

$$f(A, B, C, D) = \sum m(1, 3, 4, 5, 6, 8, 12, 14, 15)$$

Ans. :

Step I : List all the minterms

Minterms	Binary equivalent A	B	C	D
m_1	0	0	0	1
m_3	0	0	1	1
m_4	0	1	0	0
m_5	0	1	0	1
m_6	0	1	1	0
m_8	1	0	0	0
m_{12}	1	1	0	0
m_{14}	1	1	1	0
m_{15}	1	1	1	1

Table 5

Step II : From Table 5 arrange minterms with respect to number of 1's.

Group	Minterms	Binary equivalent A	B	C	D
Group 1	m_1	0	0	0	1
	m_4	0	1	0	0
	m_8	1	0	0	0
Group 2	m_3	0	0	1	1
	m_5	0	1	0	1
	m_6	0	1	1	0
	m_{12}	1	1	0	0
Group 3	m_{14}	1	1	1	0
Group 4	m_{15}	1	1	1	1

Table 6

Step III : Using Table 6 prepare first reduction table

Group	Minterms	Binary equivalent A	B	C	D	
Group 1	m_1, m_3	0	0	X	1	$\rightarrow P_A$
	m_1, m_5	0	X	0	1	$\rightarrow P_B$
	m_4, m_5	0	1	0	X	$\rightarrow P_C$
	m_4, m_6	0	1	X	0	√
	m_4, m_{12}	X	1	0	0	√
	m_8, m_{12}	1	X	0	0	$\rightarrow P_D$
Group 2	m_6, m_{14}	X	1	1	0	√
	m_{12}, m_{14}	1	1	X	0	√
Group 3	m_{14}, m_{15}	1	1	1	X	$\rightarrow P_E$

Table 7

Step IV : From Table 7 from second reduction table.

Group	Minterms	Binary equivalent A	B	C	D	
Group 1	m_4, m_6, m_{12}, m_{14}	X	1	X	0	$\rightarrow P_F$
	m_4, m_{12}, m_6, m_{14}	X	1	X	0	

Table 8

Now we have to consider remaining terms from Table 7 and final term from Table 8 and make cover table or prime implicant table.

Step V : Cover table

Prime Implicants					m_1	m_3	m_4	m_5	m_6	m_8	m_{12}	m_{14}	m_{15}
P_A =	0	0	X	1	X	X							
P_B =	0	X	0	1	X			X					
P_C =	0	1	0	X			X	X					
P_D =	1	X	0	0						X	X		
P_E =	1	1	1	X								X	X
P_F =	X	1	X	0			X		X		X	X	
							✓		✓		✓	✓	

Minterms

Table 9

P_F is the essential implicant. If we cover P_F, it will cover minterm 4, 6, 12, 14. Now remaining terms are P_A, P_B, P_C, P_D and P_E. These cover minterms 1, 3, 5, 4, 8, 12, 14, 15. But as minterms 4, 12 and 14 are already covered by P_F, consider only minterms 1, 3, 5, 8 and 15.

So reduced cover table

Prime Implicants					m_1	m_3	m_5	m_8	m_{15}
P_A =	0	0	X	1	X	X			
P_B =	0	X	0	1	X		X		
P_C =	0	1	0	X			X		
P_D =	1	X	0	0				X	
P_E =	1	1	1	X					X

Minterms

Table 10

Observing Table 10, we can conclude that if prime implicants, P_A, P_B, P_D and P_E are selected, it will cover minterms 1, 3, 5, 8 and 15. Therefore we need not consider P_C.

∴ Final reduction,

$$F(P_A, P_B, P_D, P_E, P_F) = (0\,0\,X\,1, 0\,X\,0\,1, 1\,X\,0\,0, 1\,1\,X\,X, X\,1\,X\,0)$$

∴ $$\boxed{F = \bar{A}\,\bar{B}\,D + \bar{A}\,\bar{C}\,D + A\bar{C}\,\bar{D} + ABC + B\bar{D}}$$ ...Ans.

7. Simplify the following expression using K–map and realize using any two input gates :

$$f(A, B, C, D) = \sum m\,(4, 5, 8, 9, 11, 12, 13, 15)$$ **(May 97, 6 Marks)**

Ans.: Refer Section 4.6.3, Ex. 13.

8. Draw K–map for the following function and simplify : **(May 97, 6 Marks)**

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

Ans.: Refer Section 4.8, Ex. 24 (iv).

9. Minimise the following expression using Quine McCluskey tabular method:

$$f(A, B, C, D) = \sum m(1, 4, 6, 7, 9, 12, 13, 14, 15)$$ **(May 97, 10 Marks)**

Ans. :

Step I : List all the minterms.

Minterms	Binary equivalent A	B	C	D
m_1	0	0	0	1
m_4	0	1	0	0
m_6	0	1	1	0
m_7	0	1	1	1
m_9	1	0	0	1
m_{12}	1	1	0	0
m_{13}	1	1	0	1
m_{14}	1	1	1	0
m_{15}	1	1	1	1

Table 11

p II : From Table 11 arrange minterms with respect to 1.

Group	Minterms	Binary equivalent			
		A	B	C	D
Group 1	m_1	0	0	0	1
	m_4	0	1	0	0
Group 2	m_6	0	1	1	0
	m_9	1	0	0	1
	m_{12}	1	1	0	0
Group 3	m_7	0	1	1	1
	m_{13}	1	1	0	1
	m_{14}	1	1	1	0
Group 4	m_{15}	1	1	1	1

Table 12

p III : Using Table 12 prepare first reduction table.

Group	Minterms	Binary equivalent				
		A	B	C	D	
Group 1	m_1, m_9	X	0	0	1	$\rightarrow P_A$
	m_4, m_6	0	1	X	0	√
	m_4, m_{12}	X	1	0	0	√
Group 2	m_6, m_7	0	1	1	X	√
	m_6, m_{14}	X	1	1	0	√
	m_9, m_{13}	1	X	0	1	$\rightarrow P_B$
	m_{12}, m_{13}	1	1	0	X	√
	m_{12}, m_{14}	1	1	X	0	√
Group 3	m_7, m_{15}	X	1	1	1	√
	m_{13}, m_{15}	1	1	X	1	√
	m_{14}, m_{15}	1	1	1	X	√

Table 13

ep IV : Using Table 13 list second reduction

Group	Minterms	Binary equivalent			
		A	B	C	D
Group 1	m_4, m_6, m_{12}, m_{14}	X	1	X	0
	m_4, m_{12}, m_6, m_{14}	X	1	X	0
Group 2	m_6, m_7, m_{14}, m_{15}	X	1	1	X
	m_6, m_{14}, m_7, m_{15}	X	1	1	X
	$m_{12}, m_{13}, m_{14}, m_{15}$	1	1	X	X
	$m_{12}, m_{14}, m_{13}, m_{15}$	1	1	X	X

Table 14

Finally reducing this Table 14, we get

Group	Minterms	A	B	C	D	
1	$m_4, m_6, m_{12}, m_{14}, m_4, m_{12}, m_6, m_{14}$	X	1	X	0	$\rightarrow P_C$
2	$m_6, m_7, m_{14}, m_{15}, m_6, m_{14}, m_7, m_{15}$	X	1	1	X	$\rightarrow P_D$
3	$m_{12}, m_{13}, m_{14}, m_{15}, m_{12}, m_{14}, m_{13}, m_{15}$	1	1	X	X	$\rightarrow P_E$

Table 15

Now we have to consider remaining terms from Table 13 and final terms from Table 15 and make cover table.

Step V : Cover table

Prime Implicants	m_1	m_4	m_6	m_7	m_9	m_{12}	m_{13}	m_{14}	m_{15}
P_A = X 0 0 1	X				X				
P_B = 1 X 0 1					X		X		
P_C = X 1 X 0		X	X			X		X	
P_D = X 1 1 X			X	X				X	X
P_E = 1 1 X X						X	X	X	X
		✓	✓	✓		✓	✓	✓	✓

Table 16

P_C, P_D and P_E are essential implicants. If we cover P_C, P_D and P_E ; these terms will cover minterms 4, 6, 7, 12, 13, 14, 15. Now only remaining terms are P_A and P_B. These cover minterms 1, 9 and 13. But minterm 13 is already covered by P_E. Therefore consider only minterms 1 and 9.

So reduced cover table.

Prime Implicants	m_1	m_9
P_A = X 0 0 1	X	X
P_B = 1 X 0 1		X

Table 17

Observing Table 17, we can conclude that if prime implicant P_A is selected, it will cover minterms 1 and 9. Therefore we need not consider P_B.

∴ Final reduction

$F(P_A, P_C, P_D, P_E) = \{X\,0\,0\,1, X\,1\,X\,0, X\,1\,1\,X, 1\,1\,X\,X\}$

∴ $\boxed{F(A, B, C, D) = \bar{B}\bar{C}D + B\bar{D} + BC + AB}$ ...Ans.

10. Simplify the following expression using K–map and realise using minimum number of 2–input gates. (Dec. 97, 6 Marks)

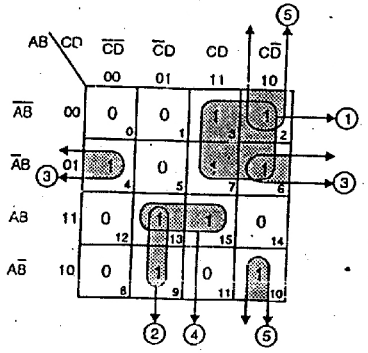
$f(A, B, C, D) = \sum m\,(2, 3, 4, 6, 7, 9, 10, 13, 15)$

Ans. : As highest state = 15, therefore K–map required will be of 4 variables (A, B, C, D).

Step 1 : Now first draw basic skeleton of K-map.

Step 2 : Presently box is empty. Write down 1 in the box whose number is specified in the example. 1, 3, 4, 6, 7, 9, 10, 13,15.

Remaining boxes of K–map should be filled up with zeros.



fter making different groups of ones write down output equation.

$$Y = \underset{(1)}{\bar{A}C} + \underset{(2)}{A\bar{C}D} + \underset{(3)}{\bar{A}B\bar{D}} + \underset{(4)}{ABD} + \underset{(5)}{\bar{B}C\bar{D}}$$

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

$$= C(\bar{A} + \bar{B}\bar{D}) + A\bar{C}D + B(\bar{A}\bar{B} + AB) \quad \ldots(2)$$

$$\ldots(3)$$

plementing above using Equation (2)

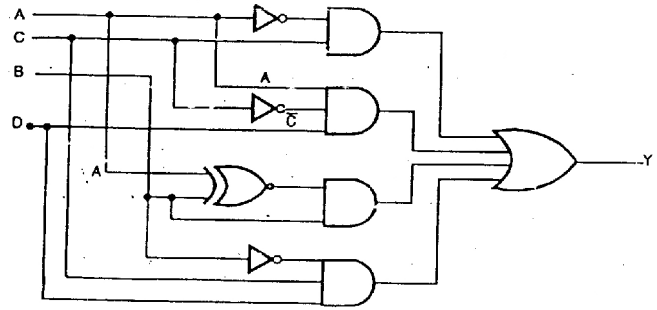


Fig. Q. 4.10 (a)

NOT gate is equal to single input NAND gate

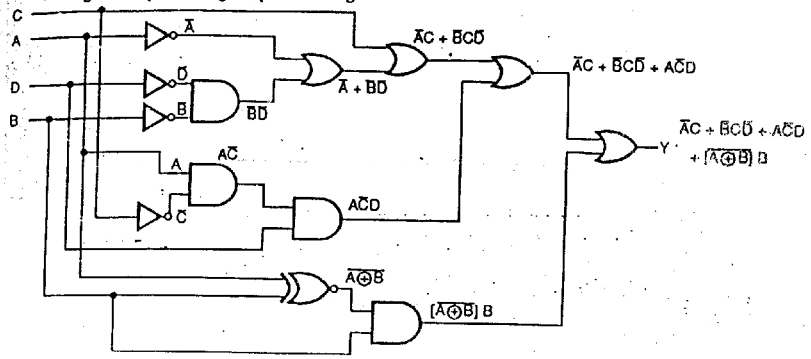


Fig. Q. 4.10 (b)

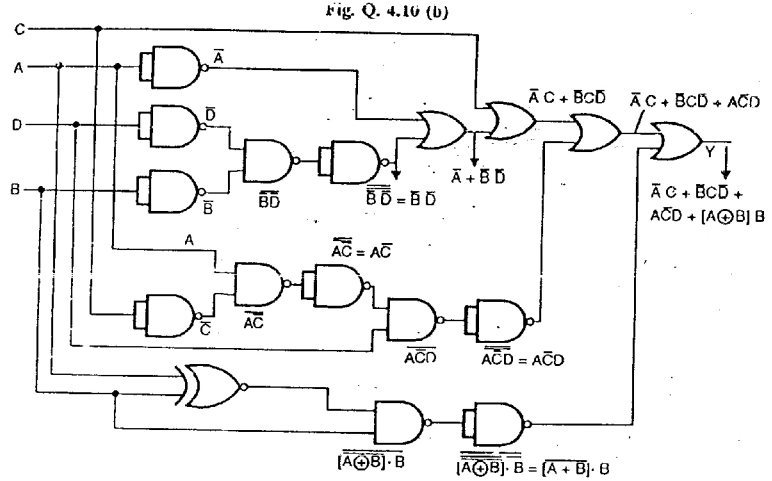


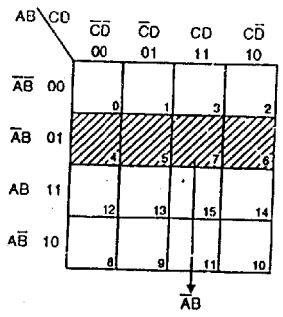
Fig. Q. 4.10 (c)

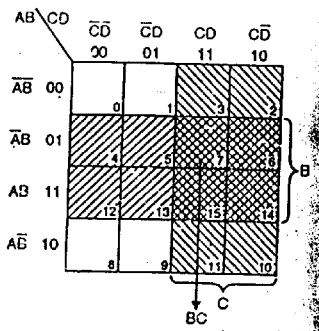
11. Obtain the K-map for the following function : **(Dec. 97, 6 Marks)**

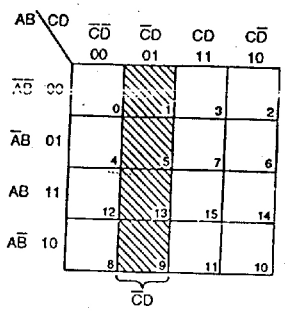
$$f(A, B, C, D) = \bar{A}B + BC + \bar{C}D$$

Ans. : Observing the expression we conclude that it has 4 variables,

Draw basic skeleton for 4 variable K-map.

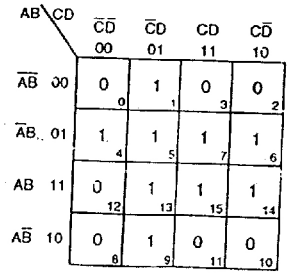






Now put 1 in the shaded blocks for every term in the equation and rest of the blocks are filled by zeros.

∴ K-map for the function will be



12. Minimise the following expression using Quine McCluskey tabular method:

$f(A, B, C, D) = \Sigma m (1, 3, 7, 9, 10, 11, 13, 15)$ **(Dec. 97, 10 Marks)**

Ans. :

Step I : List all the minterms.

Minterms	Binary equivalent			
	A	B	C	D
m_1	0	0	0	1
m_3	0	0	1	1
m_7	0	1	1	1
m_9	1	0	0	1
m_{10}	1	0	1	0
m_{11}	1	0	1	1
m_{13}	1	1	0	1
m_{15}	1	1	1	1

Table 18

Step II : From Table 18 arrange minterms with respect to 1.

Group	Minterms	Binary equivalent			
		A	B	C	D
Group 1	m_1	0	0	0	1
Group 2	m_3	0	0	1	1
	m_9	1	0	0	1
	m_{10}	1	0	1	0
Group 3	m_7	0	1	1	1
	m_{11}	1	0	1	1
	m_{13}	1	1	0	1
Group 4	m_{15}	1	1	1	1

Table 19

Step III : Using Table 19 prepare first reduction table.

Group	Minterms	Binary equivalent				
		A	B	C	D	
Group 1	m_1, m_3	0	0	X	1	√
	m_1, m_9	X	0	0	1	√
Group 2	m_3, m_7	0	X	1	1	√
	m_3, m_{11}	X	0	1	1	√
	m_9, m_{11}	1	0	X	1	√
	m_9, m_{13}	1	X	0	1	√
	m_{10}, m_{11}	1	0	1	X	$\rightarrow P_A$
Group 3	m_7, m_{15}	X	1	1	1	√
	m_{11}, m_{15}	1	X	1	1	√
	m_{13}, m_{15}	1	1	X	1	√

Table 20

Step IV : Using Table 20 list second reduction table

Group	Minterms	Binary equivalent			
		A	B	C	D
1	m_1, m_3, m_9, m_{11}	X	0	X	1
	m_1, m_9, m_3, m_{11}	X	0	X	1
2	m_3, m_7, m_{11}, m_{15}	X	X	1	1
	m_3, m_{11}, m_7, m_{15}	X	X	1	1
	$m_9, m_{11}, m_{13}, m_{15}$	1	X	X	1
	$m_9, m_{13}, m_{11}, m_{15}$	1	X	X	1

Table 21

Step V : Now we have to consider Table 21 and prepare third reduction table

Minterms	Binary equivalent				
	A	B	C	D	
$m_1, m_3, m_9, m_{11}, m_1, m_9, m_3, m_{11}$	X	0	X	1	$\rightarrow P_B$
$m_3, m_7, m_{11}, m_{15}, m_3, m_{11}, m_7, m_{15}$	X	X	1	1	$\rightarrow P_C$
$m_9, m_{11}, m_{13}, m_{15}, m_9, m_{13}, m_{11}, m_{15}$	1	X	X	1	$\rightarrow P_D$

Table 22

Step VI : Now we have to consider remaining terms from Table 20 and final term from Table 22 and make cover table.

Prime Implicants	m_1	m_3	m_7	m_9	m_{10}	m_{11}	m_{13}	m_{15}
P_A = 1 0 1 X					X	X		
P_B = X 0 X 1	X	X		X		X		
P_C = X X 1 1		X	X			X		X
P_D = 1 X X 1				X		X	X	X
	✓	✓	✓			✓	✓	✓

Table 23

All implicants are prime.

$F(P_A, P_B, P_C, P_D) = \{1\,0\,1\,X, X\,0\,X\,1, X\,X\,1\,1, 1\,X\,X\,1\}$

$\therefore$ $\boxed{F = A\bar{B}C + \bar{B}D + CD + AD}$ ...Ans.

13. Minimise the following expression using Quine McCluskey tabular method.

(May 98, 10 Marks)

$f(A, B, C, D) = \pi M(0, 2, 3, 5, 6, 7, 9, 13, 14, 16, 18, 19, 21, 22, 23, 25, 29, 30)$

Ans.: The max-terms are,

$f(A, B, C, D) = \Sigma m(1, 4, 8, 10, 11, 12, 15, 17, 20, 24, 26, 27, 28, 31)$

Step I : List all the minterms

Minterms	Binary equivalent				
	A	B	C	D	E
m_1	0	0	0	0	1
m_4	0	0	1	0	0
m_8	0	1	0	0	0
m_{10}	0	1	0	1	0
m_{11}	0	1	0	1	1
m_{12}	0	1	1	0	0
m_{15}	0	1	1	1	1
m_{17}	1	0	0	0	1
m_{20}	1	0	1	0	0
m_{24}	1	1	0	0	0
m_{26}	1	1	0	1	0
m_{27}	1	1	0	1	1
m_{28}	1	1	1	0	0
m_{31}	1	1	1	1	1

Table 24

Step II : From Table 24 arrange minterms with respect to 1.

Group	Minterms	Binary equivalent				
		A	B	C	D	E
Group 1	m_1	0	0	0	0	1
	m_4	0	0	1	0	0
	m_8	0	1	0	0	0
Group 2	m_{10}	0	1	0	1	0
	m_{12}	0	1	1	0	0
	m_{17}	1	0	0	0	1
	m_{20}	1	0	1	0	0
	m_{24}	1	1	0	0	0
Group 3	m_{11}	0	1	0	1	1
	m_{26}	1	1	0	1	0
	m_{28}	1	1	1	0	0
Group 4	m_{15}	0	1	1	1	1
	m_{27}	1	1	0	1	1
Group 5	m_{31}	1	1	1	1	1

Table 25

Step III : Using Table 25 prepare first reduction table.

Group	Minterms	Binary equivalent					
		A	B	C	D	E	
Group 1	m_1, m_{17}	X	0	0	0	1	$\rightarrow P_A$
	m_4, m_{12}	0	X	1	0	0	√
	m_4, m_{20}	X	0	1	0	0	√
	m_8, m_{10}	0	1	0	X	0	√
	m_8, m_{12}	0	1	X	0	0	√
	m_8, m_{24}	X	1	0	0	0	√
Group 2	m_{10}, m_{11}	0	1	0	1	X	√
	m_{10}, m_{26}	X	1	0	1	0	√
	m_{12}, m_{28}	X	1	1	0	0	√
	m_{20}, m_{28}	1	X	1	0	0	√
	m_{24}, m_{26}	1	1	0	X	0	√
	m_{24}, m_{28}	1	1	X	0	0	√
Group 3	m_{11}, m_{15}	0	1	X	1	1	√
	m_{11}, m_{27}	X	1	0	1	1	√
	m_{26}, m_{27}	1	1	0	1	X	√
	m_{15}, m_{31}	X	1	1	1	1	√
Group 4	m_{27}, m_{31}	1	1	X	1	1	√

Table 26

Step IV : From Table 26 prepare second reduction table :

Group	Minterms	Binary equivalent					
		A	B	C	D	E	
1	$m_4, m_{12}, m_{20}, m_{28}$	X	X	1	0	0	√
	$m_4, m_{20}, m_{12}, m_{28}$	X	X	1	0	0	√
	$m_8, m_{10}, m_{24}, m_{26}$	X	1	0	X	0	$\rightarrow P_B$
	$m_8, m_{12}, m_{24}, m_{28}$	X	1	X	0	0	√
	$m_8, m_{24}, m_{12}, m_{28}$	X	1	X	0	0	√
2	$m_{10}, m_{11}, m_{26}, m_{27}$	X	1	0	1	X	√
	$m_{10}, m_{26}, m_{11}, m_{27}$	X	1	0	1	X	√
3	$m_{11}, m_{15}, m_{27}, m_{31}$	X	1	X	1	1	√
	$m_{11}, m_{27}, m_{15}, m_{31}$	X	1	X	1	1	√

Table 27

Step V : From Table 27 prepare final reduction table.

Minterms	Binary equivalent					
	A	B	C	D	E	
$m_4, m_{12}, m_{20}, m_{28}, m_4, m_{20}, m_{12}, m_{28}$	X	X	1	0	0	$\rightarrow P_C$
$m_8, m_{12}, m_{24}, m_{28}, m_8, m_{24}, m_{12}, m_{28}$	X	1	X	0	0	$\rightarrow P_D$
$m_{10}, m_{11}, m_{26}, m_{27}, m_{10}, m_{26}, m_{11}, m_{27}$	X	1	0	1	X	$\rightarrow P_E$
$m_{11}, m_{27}, m_{15}, m_{31}, m_{11}, m_{15}, m_{27}, m_{31}$	X	1	X	1	1	$\rightarrow P_F$

Table 28

Step VI : Now we have to consider some terms from Table 26, Table 27 and Table 28 and make cover table.

Prime Implicants	m_1	m_4	m_8	m_{10}	m_{11}	m_{12}	m_{15}	m_{17}	m_{20}	m_{24}	m_{26}	m_{27}	m_{28}	m_{31}
	Minterms													
P_A = X 0 0 0 1	X							X						
P_B = X 1 0 X 0			X	X						X	X			
P_C = X X 1 0 0		X				X			X				X	
P_D = X 1 X 0 0			X			X				X			X	
P_E = X 1 0 1 X				X	X						X	X		
P_F = X 1 X 1 1					X		X					X		X

Table 29

Step VII : We conclude that if prime implicants P_A, P_C, P_D, P_E and P_F are selected. It will cover all the minterms. Therefore we need not consider P_B.

∴ Final reduction

$F(P_A, P_C, P_D, P_E, P_F) = \{X\,0\,0\,0\,1, X\,X\,1\,0\,0, X\,1\,X\,0\,0, X\,1\,0\,1\,X, X\,1\,X\,1\,1\}$

∴ $\boxed{F(A, B, C, D, E) = \overline{B}\overline{C}\overline{D}E + C\overline{D}\overline{E} + B\overline{D}\overline{E} + B\overline{C}D + BDE}$ ...Ans.

14. Consider following boolean expression :

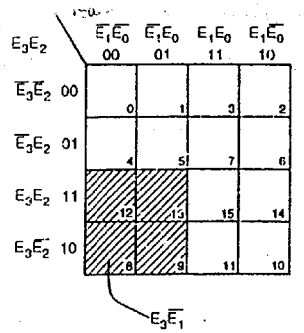
$$Y = E_3\overline{E}_1 + \overline{E}_3\overline{E}_0 + E_2\overline{E}_1E_0$$

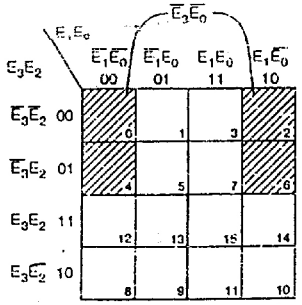
(i) Show K–map for this function.

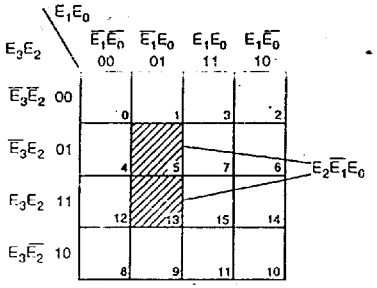
(ii) Reduce it by using K–map. **(May 98, 3 Marks)**

Ans. : (i) Observing the expression we conclude that it has 4 variables.

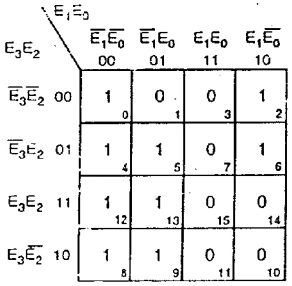
Draw basic skeleton of 4 variable K-map.

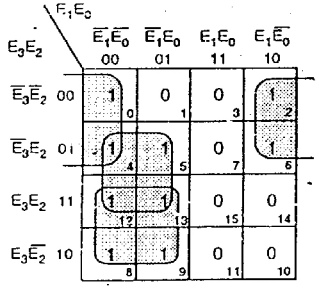






(ii) The K-map for the equation is as shown below :



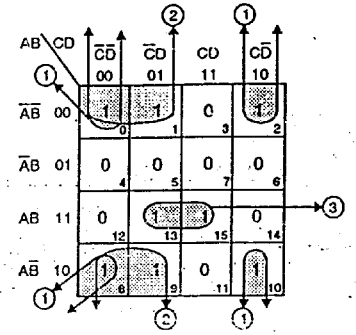


$$Y = \bar{E}_0\bar{E}_3 + \bar{E}_1E_3 + E_2\bar{E}_1$$ ...Ans.

15. Draw Veitch diagram for the following function :

$$f(A_3, A_2, A_1, A_0) = \sum m(0, 1, 2, 8, 9, 10, 13, 15)$$ **(May 98, 5 Marks)**

Ans. : Veitch Diagram is also known as K-map . Draw basic skeleton of K-map . Write 1 for the given known minterms and write zero for remaining terms.

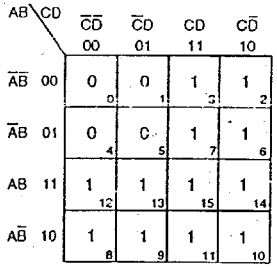


$$Y = \underset{(1)}{\bar{B}\bar{D}} + \underset{(3)}{ABD} + \underset{(2)}{\bar{B}\bar{C}}$$ ...Ans.

16. (a) Make a K-map for the function

$$Y = A\bar{B} + A\bar{C} + C + AD + A\bar{B}C + ABC$$ **(May 98, 2 Marks)**

Ans.:



(b) Express Y in standard SOP form. **(May 98, 3 Marks)**

Ans. : $Y = A\bar{B} + A\bar{C} + C + AD + A\bar{B}C + ABC$

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

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

$$= A\bar{B}CD + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{C}BD + A\bar{C}B\bar{D} + A\bar{C}\bar{B}D$$
$$+ A\bar{C}\bar{B}\bar{D} + (AC + A\bar{C})(B + \bar{B})(D + \bar{D}) + ABCD + AB\bar{C}D + A\bar{B}CD + A\bar{B}\bar{C}D$$
$$+ A\bar{B}CD + A\bar{B}C\bar{D} + ABCD + ABC\bar{D}$$
$$= A\bar{B}CD + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{C}BD + A\bar{C}B\bar{D} + A\bar{C}\bar{B}D + A\bar{C}\bar{B}\bar{D}$$
$$+ ABCD + AB\bar{C}D + ABC\bar{D} + AB\bar{C}\bar{D} + \bar{A}BCD + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}B\bar{C}\bar{D}$$
$$+ ABCD + AB\bar{C}D + A\bar{B}CD + A\bar{B}\bar{C}D + A\bar{B}CD + A\bar{B}C\bar{D} + ABCD + ABC\bar{D}$$

$$\boxed{Y = A\bar{B}CD + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + AB\bar{C}D + AB\bar{C}\bar{D} + ABCD + ABC\bar{D} + \bar{A}BCD + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}B\bar{C}\bar{D}}$$

...Ans.

(c) Minimise it and realise the minimized expression using NAND gate only. (May 98, 4 marks)

Ans. :
$$Y = A\bar{B} + A\bar{C} + C + AD + A\bar{B}C + ABC$$
$$= A\bar{B} + A\bar{C} + C + AD + AC(B + \bar{B})$$

As $A + \bar{A} = 1$
$$= A\bar{B} + A\bar{C} + C + AD + AC$$
$$= A\bar{B} + A(C + \bar{C}) + C + AD$$
$$= A\bar{B} + A + C + AD$$
$$= A(\bar{B} + 1) + C + AD$$
$$= A + C + AD \quad (\because 1 + \bar{B} = 1)$$
$$= A(1 + D) + C$$
$$= A + C$$
$$= \overline{\overline{A + C}}$$

As $\bar{\bar{A}} = A$ and using Demorgan theorem

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

$$\boxed{Y = \bar{A} \cdot \bar{C}}$$

...Ans.

Implement this using NAND gate

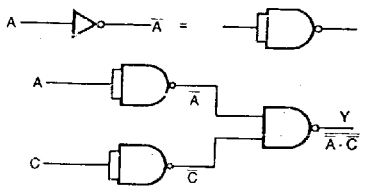


Fig. Q.4.16

...Ans.

(d) Express Y in standard POS form. (May 98, 3 Marks)

Ans. : The minimised expression from last answer is,

$$Y = \overline{\overline{\bar{A}\,\bar{C}}}$$

$\therefore$ In POS form, $Y = \overline{\bar{A}\,\bar{C}\,(B + \bar{B})(D + \bar{D})}$

Simplify this, we get,

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

...Ans.

(e) Minimise it and implement it using NOR gates only. (May 98, 4 Marks)

Ans. :
$$Y = (A + B + C + D)(A + B + C + \bar{D})(A + \bar{B} + C + D)(A + \bar{B} + C + \bar{D})$$
$$= (A + B + C)(D + \bar{D})(A + \bar{B} + C)(D + \bar{D}) \quad \{\text{As } A + \bar{A} = 1\}$$
$$Y = (A + B + C)(A + \bar{B} + C)$$
$$= (A + C)(B + \bar{B})$$
$$Y = A + C$$

Now using Boolean law

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

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

...Ans.

17. Simiplify the Veitch diagram and obtain reduced SOP form expression.

$$f(A_3, A_2, A_1, A_0) = \sum m\,(0, 2, 3, 7, 9, 10, 11)$$

(i) Find reduced expression using Boolean Algebra. (Dec. 98, 4 Marks)

Ans.: According to minterms write expression for each term.

$$Y = \bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0 + \bar{A}_3\bar{A}_2A_1\bar{A}_0 + \bar{A}_3\bar{A}_2A_1A_0 + \bar{A}_3A_2A_1A_0$$
$$+ A_3\bar{A}_2\bar{A}_1A_0 + A_3\bar{A}_2A_1\bar{A}_0 + A_3A_2\bar{A}_1A_0$$

As $A + A = A$

So adding some terms to the equation we get

$$Y = \bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0 + \bar{A}_3\bar{A}_2A_1\bar{A}_0 + \bar{A}_3\bar{A}_2A_1A_0 + \bar{A}_3A_2A_1A_0 + \bar{A}_3\bar{A}_2A_1A_0 + \bar{A}_3\bar{A}_2A_1\bar{A}_0$$
$$+ A_3\bar{A}_2A_1A_0 + A_3\bar{A}_2A_1\bar{A}_0 + A_3\bar{A}_2\bar{A}_1A_0 + A_3\bar{A}_2A_1A_0$$
$$= \bar{A}_3\bar{A}_2\bar{A}_0(\bar{A}_1 + A_1) + \bar{A}_3A_1A_0(\bar{A}_2 + A_2) + \bar{A}_3\bar{A}_2\bar{A}_1(A_0 + \bar{A}_0) + A_3\bar{A}_2A_0(\bar{A}_1 + A_1)$$
$$+ A_3\bar{A}_2A_1(\bar{A}_0 + A_0)$$

$$= \bar{A}_3\bar{A}_2\bar{A}_0 + \bar{A}_3A_1A_0 + \bar{A}_3\bar{A}_2A_1 + A_3A_2A_0 + A_3\bar{A}_2A_1$$

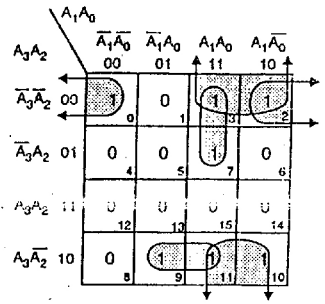
$$= \bar{A}_3\bar{A}_2\bar{A}_0 + \bar{A}_1A_1A_0 + \bar{A}_2A_1(\bar{A}_3 + A_3) + A_3\bar{A}_2A_0$$

$$= \bar{A}_3\bar{A}_2\bar{A}_0 + \bar{A}_3A_1A_0 + \bar{A}_2A_1 + A_3\bar{A}_2A_0$$

$$\boxed{Y = \bar{A}_3\bar{A}_2\bar{A}_0 + \bar{A}_3A_1A_0 + \bar{A}_2A_1 + A_3\bar{A}_2A_0}$$ ...Ans.

(ii) Find reduced expression using K-map. **(Dec. 98, 4 Marks)**

Ans. :



$$\therefore \quad \boxed{Y = \bar{A}_3\bar{A}_2\bar{A}_0 + \bar{A}_3A_1A_0 + A_3\bar{A}_2A_0 + \bar{A}_2A_1}$$ ...Ans.

18. Minimise following expression using Quine McCluskey technique. **(May 98, 10 Marks)**

$f(A_3, A_2, A_1, A_0) = \Sigma m\,(0, 1, 4, 6, 13, 15, 16, 17, 29, 31)$

Ans. :

Step I : List all the minterms.

Minterms	A	B	C	D	E	
m_0	0	0	0	0	0	√
m_1	0	0	0	0	1	√
m_4	0	0	1	0	0	√
m_6	0	0	1	1	0	√
m_{13}	0	1	1	0	1	√
m_{15}	0	1	1	1	1	√
m_{16}	1	0	0	0	0	√
m_{17}	1	0	0	0	1	√
m_{29}	1	1	1	0	1	√
m_{31}	1	1	1	1	1	√

Table 30

Step II : Arrange minterms according to number of 1's.

Group	Minterms	A	B	C	D	E	
1	m_0	0	0	0	0	0	√
2	m_1	0	0	0	0	1	√
	m_4	0	0	1	0	0	√
	m_{16}	1	0	0	0	0	√
3	m_6	0	0	1	1	0	√
	m_{17}	1	0	0	0	1	√
4	m_{13}	0	1	1	0	1	√
5	m_{15}	0	1	1	1	1	√
	m_{29}	1	1	1	0	1	√
6	m_{31}	1	1	1	1	1	√

Table 31

Step III : Using Table 31 prepare first reduction Table :

Group	Minterms	A	B	C	D	E	
1	m_0, m_1	0	0	0	0	X	√
	m_0, m_4	0	0	X	0	0	P_A
	m_0, m_{16}	X	0	0	0	0	√
2	m_1, m_{17}	X	0	0	0	1	√
	m_4, m_6	0	0	1	X	0	P_B
	m_{16}, m_{17}	1	0	0	0	X	√
4	m_{13}, m_{15}	0	1	1	X	1	√
	m_{13}, m_{29}	X	1	1	0	1	√
5	m_{15}, m_{31}	X	1	1	1	1	√
	m_{29}, m_{31}	1	1	1	X	1	√

Table 32

Step IV : Using Table 32 prepare second reduction table.

Group	Minterms	A	B	C	D	E
1	m_0, m_1, m_{16}, m_{17}	X	0	0	0	X
	m_0, m_{16}, m_1, m_{17}	X	0	0	0	X
4	$m_{13}, m_{15}, m_{29}, m_{31}$	X	1	1	X	1
	$m_{13}, m_{29}, m_{15}, m_{31}$	X	1	1	X	1

Table 33

Step V :

Minterms	A	B	C	D	E	
$m_0, m_1, m_{16}, m_{17}, m_0, m_{16}, m_1, m_{17}$	X	0	0	0	X	$\rightarrow P_C$
$m_{13}, m_{15}, m_{29}, m_{31}, m_{13}, m_{29}, m_{15}, m_{31}$	X	1	1	X	1	$\rightarrow P_D$

Table 34

Step VI : Now we have to consider terms from Table 33 and some terms from fig. 18.5 and make cover table.

Prime Implicants	m_0	m_1	m_4	m_6	m_{13}	m_{15}	m_{16}	m_{17}	m_{29}	m_{31}
P_A = 0 0 X 0 0	X		X							
P_B = 0 0 1 X 0			X	X						
P_C = X 0 0 0 X	X	X					X	X		
P_D = X 1 1 X 1					X	X			X	X
	√	√			√	√	√	√	√	√

Table 35

P_B P_C and P_D are essential implicants. If we cover P_C and P_D, these terms will cover minterms 0, 1, 4, 6, 13, 15, 16, 17, 29, 31. By taking these three terms, P_A is covered.

∴ Final reduction

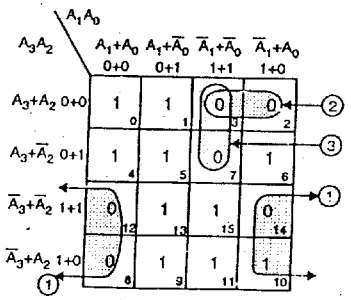
$$F(P_B, P_C, P_D) = \{0\,0\,1\,X\,0, X\,0\,0\,0\,X, X\,1\,1\,X\,1\}$$

∴ $\boxed{F(A, B, C, D, E) = \bar{A}B\bar{C}E + \bar{B}\bar{C}\bar{D} + BCE}$ ...Ans.

19. Draw a Veitch diagram for the following function and find the reduced expression.

$f(A_3, A_2, A_1, A_0) = \pi M\,(2, 3, 7, 8, 10, 12, 14)$ **(May 98, 6 Marks)**

Ans. :



As it is product of maxterms, make groups of zeros.

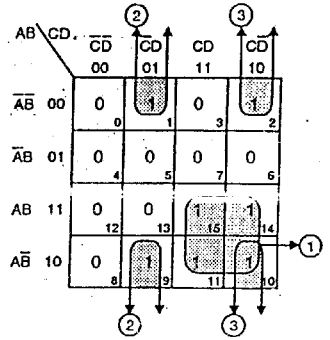
∴ Output equation is

$$\boxed{Y = \underset{(1)}{(A_0 + \bar{A}_3)}\,\underset{(2)}{(A_3 + A_2 + \bar{A}_1)}\,\underset{(3)}{(A_3 + \bar{A}_1 + \bar{A}_0)}}$$ ...Ans.

20. Simplify the following expression using K-map and realise using any 2–input gates.

$f(A, B, C, D) = \Sigma\, m\,(1, 2, 9, 10, 11, 14, 15)$ **(June 99, 6 Marks)**

Ans. :



∴ Output equation, $Y = AC + \bar{B}\bar{C}D + B\bar{C}\bar{D}$

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

$\boxed{Y = AC + B(C \oplus D)}$ ...Ans.

Implementation of above equation using any 2 input gates.

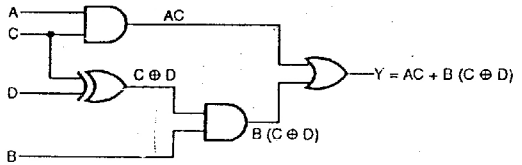


Fig. Q. 4.20 ...Ans.

21. Draw K–map for the following expression : **(June 99, 6 Marks)**

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

Ans.: Refer Section 4.8, Ex. 24 (v).

22. Minimize the following expression using Quine McCluskey tabular method :

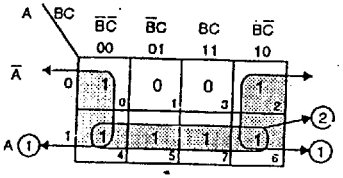
$$f(A, B, C, D) = \sum m (1, 3, 5, 10, 11, 12, 13, 14, 15)$$ **(June 99, 10 Marks)**

Ans.: Refer Section 4.9, Ex. 25.

23. (a) Make a K-map for the function

$$Y = A\bar{B} + AC + \bar{C} + A\bar{B}C + ABC$$ **(Dec. 99, 2 Marks)**

Ans. :



$$\boxed{Y = \bar{C} + A = A + \bar{C}}$$ **...Ans.**

(b) Express Y in standard SOP form. **(Dec. 99, 3 Marks)**

Ans. : $Y = A\bar{B} + AC + \bar{C} + A\bar{B}C + ABC$

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

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

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

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

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

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

But Y in standard SOP form is,

$$\boxed{Y = A\bar{B}C + A\bar{B}\bar{C} + ABC + AB\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}\bar{C}}$$ **...Ans.**

(c) Minimise it and realize using only NAND gates. (Use K–map) **(Dec. 99, 4 Marks)**

Ans. : From K–map

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

$$\therefore \quad Y = \overline{\overline{A + \bar{C}}}$$

(Now using Demorgan theorem $\overline{A + B} = \bar{A} \cdot \bar{B}$)

$$\therefore \quad Y = \overline{\bar{A} \cdot \bar{\bar{C}}}$$

$$\therefore \quad \boxed{Y = \overline{\bar{A} \cdot C}}$$ **...Ans.**

Now implement above equation using NAND gates only.

As

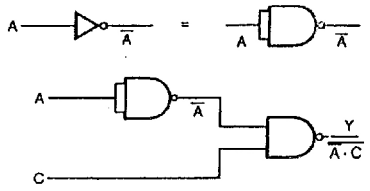
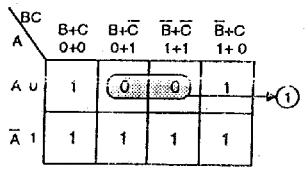


Fig. Q.4.23 (a)

(d) Express Y in standard POS form. **(Dec. 99, 4 Marks)**

Ans. : The K-map for Y is,



Making groups of zeros we get,

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

In standard POS form,

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

$$\boxed{Y = (A + B + \bar{C})(A + \bar{B} + \bar{C})}$$ **...Ans.**

(e) Minimise it and realize it using only NOR gates (use K–map) **(Dec. 99, 4 Marks)**

Ans. : From K–map, $Y = A + \bar{C}$

As $\bar{\bar{A}} = A$

$$\therefore \quad \boxed{Y = \overline{\overline{A + \bar{C}}}}$$ **...Ans.**

Implementation of above equation using NOR gates only.

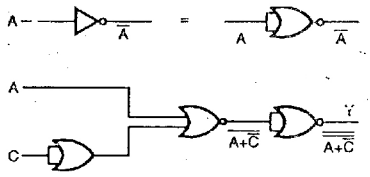


Fig. Q.4.23 (b)

24. Minimise the following function using Quine McClusky method : (Dec. 99, 10 Marks)

f(A, B, C, D, E) = $\sum$ m (8, 9, 10, 11, 13, 15, 16, 18, 21, 24, 25, 26, 27, 30, 31)

Ans.:

Step I : List all the minterms.

Minterms	A	B	C	D	E	
m_8	0	1	0	0	0	√
m_9	0	1	0	0	1	√
m_{10}	0	1	0	1	0	√
m_{11}	0	1	0	1	1	√
m_{13}	0	1	1	0	1	√
m_{15}	0	1	1	1	1	√
m_{16}	1	0	0	0	0	√
m_{18}	1	0	0	1	0	√
m_{21}	1	0	1	0	1	√
m_{24}	1	1	0	0	0	√
m_{25}	1	1	0	0	1	√
m_{26}	1	1	0	1	0	√
m_{27}	1	1	0	1	1	√
m_{30}	1	1	1	1	0	√
m_{31}	1	1	1	1	1	√

Table 36

Step II : Arrange all the minterms according to number of one's

Group	Minterms	A	B	C	D	E	
1	m_8	0	1	0	0	0	√
	m_{16}	1	0	0	0	0	√
2	m_9	0	1	0	0	1	√
	m_{10}	0	1	0	1	0	√
	m_{18}	1	0	0	1	0	√
	m_{24}	1	1	0	0	0	√
3	m_{11}	0	1	0	1	1	√
	m_{13}	0	1	1	0	1	√
	m_{21}	1	0	1	0	1	→ P_A
	m_{25}	1	1	0	0	1	√
	m_{26}	1	1	0	1	0	√
4	m_{15}	0	1	1	1	1	√
	m_{27}	1	1	0	1	1	√
	m_{30}	1	1	1	1	0	√
5	m_{31}	1	1	1	1	1	√

Table 37

Step III : Using Table 37 prepare first reduction table

Group	Minterms	A	B	C	D	E	
1	m_8, m_9	0	1	0	0	X	√
	m_8, m_{10}	0	1	0	X	0	√
	m_8, m_{24}	X	1	0	0	0	√
	m_{16}, m_{18}	1	0	0	X	0	√
	m_{16}, m_{24}	1	X	0	0	0	√
2	m_9, m_{11}	0	1	0	X	1	√
	m_9, m_{13}	0	1	X	0	1	√
	m_9, m_{25}	X	1	0	0	1	√
	m_{10}, m_{11}	0	1	0	1	X	√
	m_{10}, m_{26}	X	1	0	1	0	√
	m_{18}, m_{26}	1	X	0	1	0	√
	m_{24}, m_{25}	1	1	0	0	X	√
	m_{24}, m_{26}	1	1	0	X	0	√
3	m_{11}, m_{15}	0	1	X	1	1	√
	m_{11}, m_{27}	X	1	0	1	1	√
	m_{13}, m_{15}	0	1	1	X	1	√
	m_{25}, m_{27}	1	1	0	X	1	√
	m_{26}, m_{27}	1	1	0	1	X	√
	m_{26}, m_{30}	1	1	X	1	0	√
4	m_{15}, m_{31}	X	1	1	1	1	√
	m_{27}, m_{31}	1	1	X	1	1	√
	m_{30}, m_{31}	1	1	1	1	X	√

Table 38

Step IV : From Table 38 prepare second reduction table.

Group	Minterms	A	B	C	D	E	
1	m_8, m_9, m_{10}, m_{11}	0	1	0	X	X	→ P_B
	m_8, m_9, m_{24}, m_{25}	X	1	0	0	X	√
	m_8, m_{10}, m_9, m_{11}	0	1	0	X	X	√
	m_8, m_{10}, m_9, m_{13}	0	1	0	X	X	√
	$m_8, m_{10}, m_{24}, m_{26}$	X	1	0	X	0	√
	m_8, m_{24}, m_9, m_{25}	X	1	0	0	X	√
	$m_8, m_{24}, m_{10}, m_{26}$	X	1	0	X	0	√
	$m_{16}, m_{18}, m_{24}, m_{26}$	1	X	0	X	0	√
	$m_{16}, m_{24}, m_{18}, m_{26}$	1	X	0	X	0	√
2	$m_9, m_{11}, m_{13}, m_{15}$	0	1	X	X	1	√
	$m_9, m_{11}, m_{25}, m_{27}$	X	1	0	X	1	√
	$m_9, m_{13}, m_{11}, m_{15}$	0	1	X	X	1	√
	$m_9, m_{25}, m_{11}, m_{27}$	X	1	0	X	1	→ P_C
	$m_{10}, m_{11}, m_{26}, m_{27}$	X	1	0	1	X	√
	$m_{10}, m_{26}, m_{11}, m_{27}$	X	1	0	1	X	√
	$m_{24}, m_{25}, m_{26}, m_{27}$	1	1	0	X	X	→ P_D
3	$m_{24}, m_{26}, m_{25}, m_{27}$	1	1	0	X	X	√
	$m_{11}, m_{15}, m_{27}, m_{31}$	X	1	X	1	1	√
	$m_{11}, m_{27}, m_{15}, m_{31}$	X	1	X	1	1	√
	$m_{26}, m_{27}, m_{30}, m_{31}$	1	1	X	1	X	√
	$m_{26}, m_{30}, m_{27}, m_{31}$	1	1	X	1	X	√

Table 39

Step V : From Table 39 prepare third reduction table

Minterms	A	B	C	D	E	
$m_8, m_9, m_{10}, m_{11}, m_8, m_{10}, m_9, m_{11}, m_8, m_{10}, m_9, m_{13}$	0	1	0	X	X	→ P_A
$m_8, m_9, m_{24}, m_{25}, m_8, m_{24}, m_9, m_{25}$	X	1	0	0	X	→ P_B
$m_8, m_{10}, m_{24}, m_{26}, m_8, m_{24}, m_{10}, m_{26}$	X	1	0	X	0	→ P_C
$m_{16}, m_{18}, m_{24}, m_{26}, m_{16}, m_{24}, m_{10}, m_{26}$	1	X	0	X	0	→ P_D
$m_9, m_{11}, m_{13}, m_{15}, m_9, m_{13}, m_{13}, m_{15}$	0	1	X	X	1	→ P_E
$m_9, m_{11}, m_{25}, m_{27}, m_9, m_{25}, m_{11}, m_{27}$	X	1	0	X	1	→ P_F
$m_{10}, m_{11}, m_{26}, m_{27}, m_{10}, m_{26}, m_{11}, m_{27}$	X	1	0	1	X	→ P_G
$m_{24}, m_{25}, m_{26}, m_{27}, m_{24}, m_{26}, m_{25}, m_{27}$	1	1	0	X	X	→ P_H
$m_{11}, m_{15}, m_{27}, m_{31}, m_{11}, m_{27}, m_{15}, m_{31}$	X	1	X	1	1	→ P_I
$m_{26}, m_{27}, m_{30}, m_{31}, m_{26}, m_{30}, m_{27}, m_{31}$	1	1	X	1	X	→ P_J

Table 40

Prime Implicant table,

Prime implicants	m_8	m_9	m_{10}	m_{11}	m_{13}	m_{15}	m_{16}	m_{18}	m_{21}	m_{24}	m_{25}	m_{26}	m_{27}	m_{30}	m_{31}
P_A = 010XX	X	X	X	X						X	X				
P_B = X100X	X	X								X	X				
P_C = X10X0	X		X							X		X			
P_D = 1X0X0							X	X		X		X			
P_E = 01XX1		X		X	X	X									
P_F = X10X1		X		X							X		X		
P_G = X101X			X	X								X	X		
P_H = 110XX										X	X	X	X		
P_I = X1X11				X		X							X		X
P_J = 11X1X												X	X	X	X

8, 9, 10, 11, 24 and 25 these terms are covered by P_A. 16, 18 are covered by prime implicant P_D. 13 and 15 are covered by P_E. 26 27, 30 and 31 are covered by P_J. Minterm 21 is not covered so we will write it separately,

∴ Final reduction

$$F(P_A, P_D, P_E, P_J, P_{21}) = \{ 010XX, 1X0X0, 01XX1, 11X1X, 10101 \}$$

$$f(A, B, C, D, E) = \bar{A}B\bar{C} + A\bar{C}\bar{E} + \bar{A}BE + ABD + A\bar{B}\bar{C}DE$$

...Ans.

❑❑❑

Circuit diagram is as shown in Fig. 6.4.

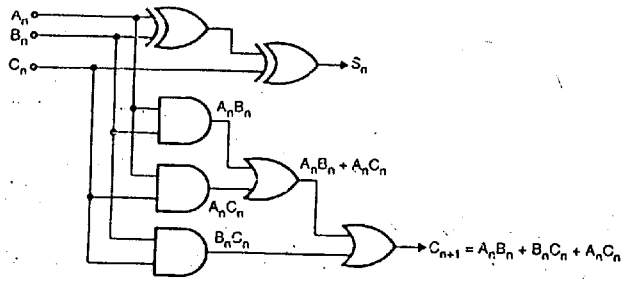


Fig. 6.4

Normally we deal with multibit operand (4 bit, 8 bit, 16 bit and so on); we can design multibit adder as basic building block as shown in Fig. 6.5.

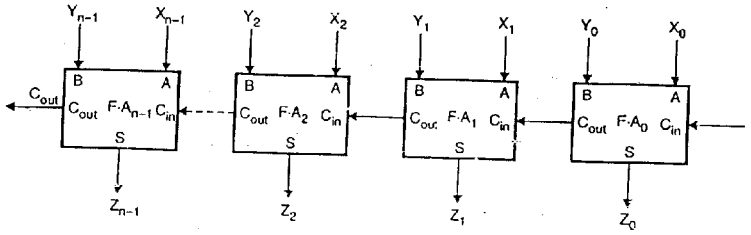


Fig. 6.5

X_{n-1} to X_0 = Operand 1 Z_{n-1} to Z_0 = Final output
Y_{n-1} to Y_0 = Operand 2

As shown carry out of previous stage is connected or propagated to carry in of next stage, it is also called ***ripple carry propagation***. When we apply operands, adder circuit will take some time to perform addition because carry should be propagated to next stage. If the number of blocks are more, propagation of carry will be slower and more time will be consumed to perform addition job. But if number of blocks are less time required will be less

At this stage I would like to say that we concentrate on improving performance of adder circuit, i.e. we should design ***Fast adders***. Here you may ask, why you concentrate on simple addition to be done very fast ? The answer is any mathematical equation can be written in terms of addition, therefore basic operation is addition For example :

A – B can be written as A + (– B)

A × B is nothing but adding A, B times. Thus adder is the basic building block therefore we w concentrate on it.

6.2.3 Fast Adders or Carry Look ahead Generator : (May 96, Dec. 96, Dec. 97)

Now we are going to design fast adder which employs ***Carry look ahead technique***. In Fig. 6.4 v have designed full adder circuit. Full adder can also be built using two half adders. Fig. 6.6 shows bloc schematic representation.

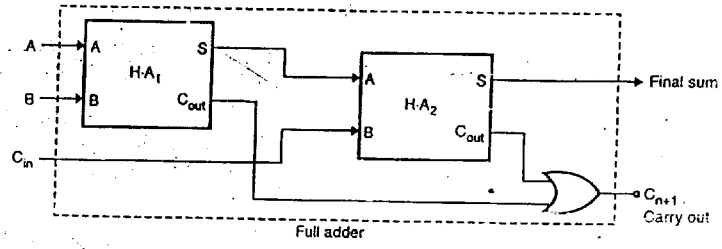


Fig. 6.6

Fig. 6.6 full adder using two half adders. Fig. 6.6 shows full adder circuit.

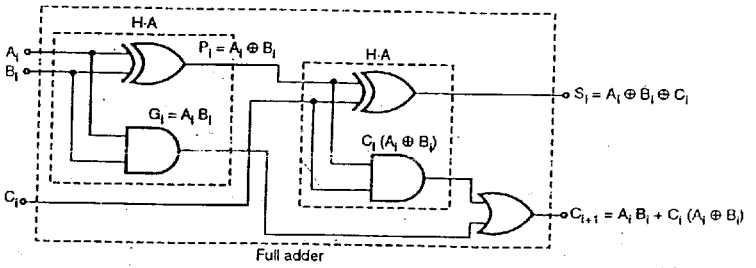


Fig. 6.7

where, $S_i = A_i + B_i + C_i$ ...(1)

$C_{i+1} = (A_i \oplus B_i) C_i + A_i B_i$

This can be written as,

$C_{i+1} = G_i + P_i C_i$ ...(2)

where, $G_i = A_i B_i$

and $P_i = A_i \oplus B_i$

The expressions G_i and P_i are called the generate and propagate functions. The propagate function causes an output carry if there is an input carry and either A_i or B_i is at logic 1. The generate function produces an output carry when both x_i and y_i are at logic 1.

Now let's see how the circuit will look like. Take i = 0 from Equation (2).

$\therefore C_1 = G_0 + P_0C_0$ Here we are going to consider $C_0 = 0$

$\therefore C_1 = G_0 = A_0B_0$

this gives us $S_0 = A_0 \oplus B_0$ (means simple half adder circuit).

$i = 1 \quad \therefore C_2 = G_1 + P_1C_1$

$i = 2 \quad \therefore C_2 = G_2 + P_2C_2$ ...(3)

Place Equation (3)

$\therefore C_3 = G_2 + P_2 (G_1 + P_1C_1)$

$\therefore C_3 = G_2 + P_2G_1 + P_1P_2C_1$

$i = 3 \quad \therefore C_4 = G_3 + P_3C_3$ ...(4)

Place Equation (4)

$= G_3 + P_3 (G_2 + P_2G_1 + P_1P_2C_1)$

$= G_3 + G_2P_3 + G_1P_2P_3 + P_1P_2P_3C_1$

$i = 4 \quad \therefore C_5 = G_4 + P_4C_4$ ...(5)

Place Equation (3)

$= G_4 + P_4 [G_3 + G_2P_3 + G_1P_2P_3 + P_1P_2P_3C_1]$

$= G_4 + G_3P_4 + G_2P_3P_4 + G_1P_2P_3P_4 + P_1P_2P_3P_4C_1$ ...(6)

Now, draw circuit for C_2, C_3 and C_4

As illustrated in Fig. 6.8 total delay is of 2 gates. Even though number of carry bit increases delay is going to remain that of 2 gates. This particular circuit is named as ***Look ahead carry generator. Look ahead*** means forecasting, carry bit. As we have seen in previous adder circuit, carry was propagating and the delay in generating final carry is a function of number of adder stages.

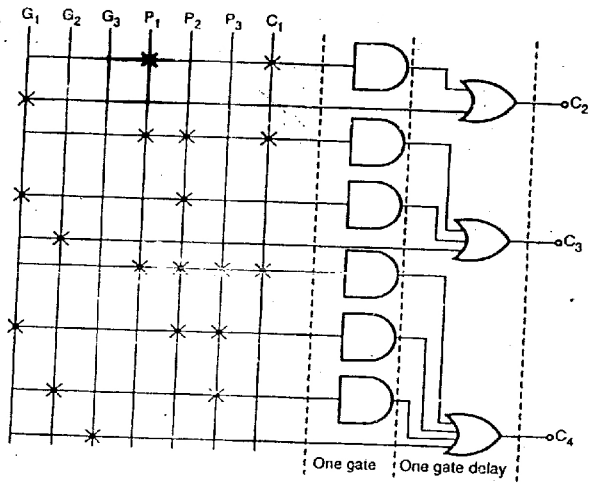


Fig. 6.8

Before we implement full fast adder circuit, we will generate 2 to 3 expressions for S_i [Equation number 1].

$S_i = A_i \oplus B_i \oplus C_i$

For $i = 0 \quad S_0 = A_0 \oplus B_0 = P_0$

$i = 1 \quad S_1 = A_1 \oplus B_1 \oplus C_1 = P_1 \oplus C_1$

$i = 2 \quad S_2 = A_2 \oplus B_2 \oplus C_2 = P_2 \oplus C_2$

$i = 3 \quad S_3 = A_3 \oplus B_3 \oplus C_3 = P_3 \oplus C_1$

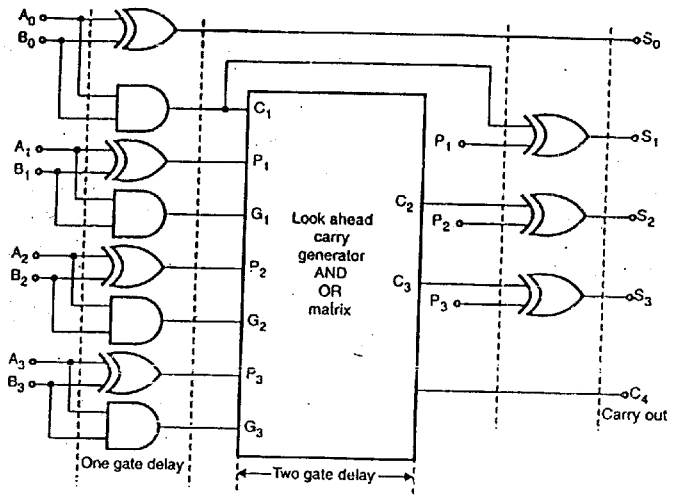


Fig. 6.9

Ex. 1 : Design 4 bit adder using 3 F. A. (full adder) and 1 (Half Adder).

Soln. : (1) From above statement we conclude that :

(a) Numbers given to 4 bit adder will be $A_3 - A_0$ and $B_3 - B_0$.

(b) Output will be 4 bit.

(c) We have to cascade all the stage.

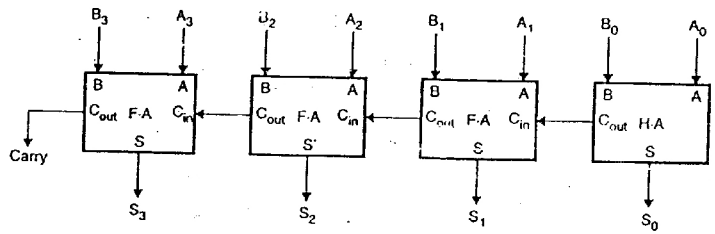


Fig. Ex. 6.1(a)

As shown in Fig. Ex. 6.1 (a) we have first block H.A. (half adder) and next three stages are full adder. H.A. doesn't cater for carry, so if we want to 'Cascade' such 4 bit adders it will not be possible because

actually carry of 1st 4 bit stage should propagate to next stage 4 bit adder, bit as H. A. doesn't have that terminal the same is not implementable. Output is $S_0 - S_3$ (Sum) with final *Carry*.

Ex. 2 : Design 4 bit adder using 4 full adders.

Soln. : In previous problem we saw that cascade is not implementable because of non availability of carry in to H.A. therefore we replace H.A. by F.A., which caters for carry generated from previous stage.

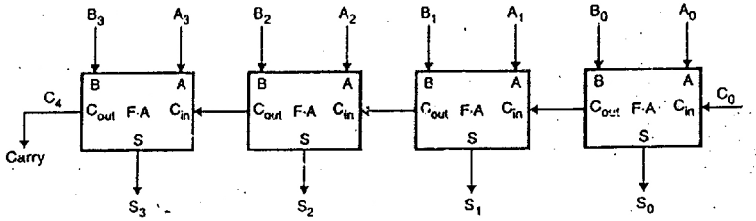


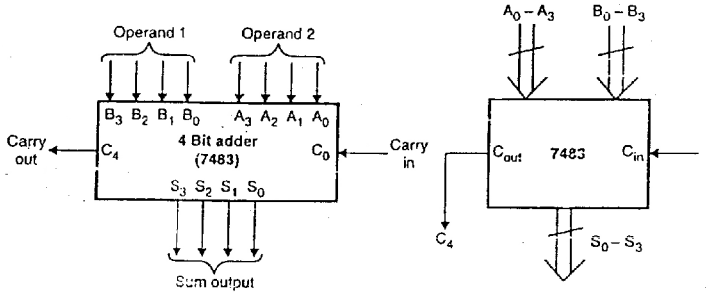
Fig. Ex. 6.2(a)

As shown in Fig. Ex. 6.2(a) we have cascaded 4 full adder blocks. The 1st F.A. (right most) is inputs A_0, B_0 and C_0 i.e. 2 inputs of operand and carry of previous stage (if at all it is present). This circuit is going to perform :

	A_3	A_2	A_1	A_0
+	B_3	B_2	B_1	B_0
+	C_3	C_2	C_1	C_0
	C_4	S_3	S_2	S_1	S_0

Final carry

Fig. Ex. 6.2(b) shows, 4 full adder circuit combined in one package, and named *4 bit full adder circuit*.



(b) (c)

Fig. Ex. 6.2

Input Operand 1 – A_3 A_2 A_1 A_0
Input Operand 2 – B_3 B_2 B_1 B_0
Carry in – C_0
Carry out – C_4
Sum output – S_3 S_2 S_1 S_0

This circuit is also called as *Binary Parallel Adder*

Pin Description :

Pin Names	Description
$A_0 - A_3$	A operand input (4 bits)
$B_0 - B_3$	B operand input (4 bits)
C_0	Carry input
C_4	Carry output
$S_0 - S_3$	Sum outputs (4 bits)

Table 1

Ex. 3 : Implement 8 bit adder using 4 bit full adder. (June 99, 6 Marks)

Soln. :

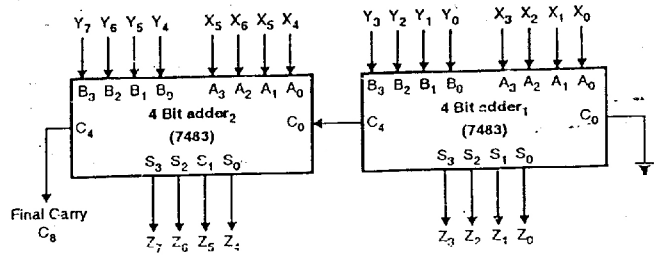


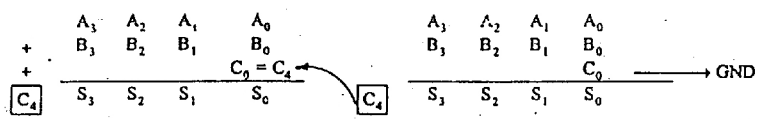
Fig. Ex. 6.3(a)

For 4 bit adder 1

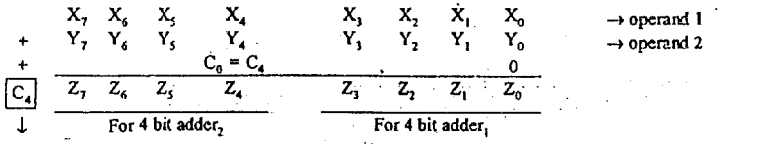
(1) A_3 A_2 A_1 A_0 = X_3 X_2 X_1 X_0
B_3 B_2 B_1 B_0 = Y_3 Y_2 Y_1 Y_0
S_3 S_2 S_1 S_0 = Z_3 Z_2 Z_1 Z_0
$C_0 = 0$ (Because this stage is the starting stage and assumes NO CARRY GENERATED FROM PREVIOUS STAGE.)
$C_4 \Rightarrow$ Carry from 3rd bit *propagated* or *Passed on to* next stage.

(2) For 4 bit adder 2
A_3 A_2 A_1 A_0 = X_7 X_6 X_5 X_4
B_3 B_2 B_1 B_0 = Y_7 Y_6 Y_5 Y_4
S_3 S_2 S_1 S_0 = Z_7 Z_6 Z_5 Z_4
C_0 = Connected to output C_4 of 1st stage
$C_4 \Rightarrow$ final carry

Operation performed is,



Is equivalent to,



Let's implement some circuits using 4 bit adder block.

Note : Here, in between carry i.e. C_1, C_2 and C_3 are not shown. But it will be taken into account.

6.3 BCD Adder :

4 bit adder block can be used to perform BCD addition. The rules for BCD addition are as follows :

Rules : (1) When answer is > 9, add $(6)_{10} = (0110)_2$
(2) If a carry is generated, add $(6)_{10}$

The above steps are performed to get VALID BCD Number

Reference for the same is Chapter 2.

Let's refresh the concept

(i) Add BCD numbers $(6)_{10}$ and $(5)_{10}$

(6) → 0 1 1 0 (A_3 A_2 A_1 A_0)
(5) → 0 1 0 1 (B_3 B_2 B_1 B_0)
1 0 1 1 (S_3 S_2 S_1 S_0)

C_{in} is grounded (for 4 bit adder)

∴ Output S_3 S_2 S_1 S_0 = 1 0 1 1. But this code is not valid BCD. To get correct answer, we should add 6 into it.

∴ 1 0 1 1 (S_3 S_2 S_1 S_0)
\+ 0 1 1 0
Carry 1 | 0 0 0 1
1 1

∴ Answer is ⟶ 11 (6 + 5 = 11) valid BCD

Take one more example,
Add BCD 9 and 8

(9) ⇒ 1 0 0 1
(8) ⇒ 1 0 0 0
arry 1 | 0 0 0 1 — First level of addition

If we observe answer is 11 but we want answer (9 + 8 = 17).

0 1 1 0
1 | 0 1 1 1

∴ add again 6, EVENTHOUGH answer is 0001 i.e. is valid BCD

1 7

∴ Answer is 17 (valid answer)

Conclusion from above two examples :

(1) We require digital circuit which senses that answer is > 9 i.e. invalid BCD. After sensing, it should add 6 to the answer.
(2) We have to also check 'Carry' generated because when carry generates answer is normally valid BCD. Therefore, circuit designed in point 1 will not help us to add 6, therefore sense 'Carry' and add six.
(3) To achieve correct BCD answer we require two level of adder circuit. First adder will add BCD inputs. Second will add 6 to the answer only and only if numebr is invalid BCD **OR** carry generated from FIRST STAGE.

So let's first design combinational logic circuit which will sense that number is greater than 9. The truth table is as shown in Table 6.3.

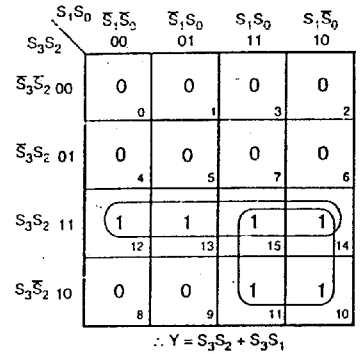
Decimal	Inputs					Output
		S_3	S_2	S_1	S_0	Y
0		0	0	0	0	0
1		0	0	0	1	0
2		0	0	1	0	0
3		0	0	1	1	0
4		0	1	0	0	0
5		0	1	0	1	0
6		0	1	1	0	0
7		0	1	1	1	0
8		1	0	0	0	0
9		1	0	0	1	0
10		1	0	1	0	1
11		1	0	1	1	1
12		1	1	0	0	1
13		1	1	0	1	1
14		1	1	1	0	1
15		1	1	1	1	1

Valid BCD

Invalid BCD

Table 6.3

K-Map :



∴ $Y = S_3S_2 + S_3S_1$

Fig. 6.10

Design :

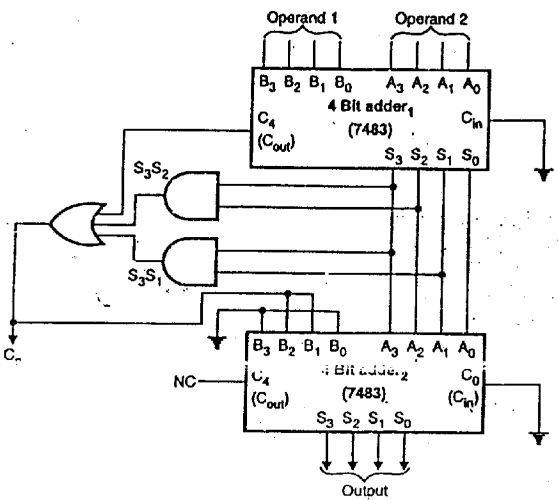


Fig. 6.11

(1) Two adder circuits (4 bit binary adders) are required. The first 4 bit binary adder performs actual addition. The second 4 bit binary adder adds six or zero depending on whether result is greater or less than 9, respectively.

(2) Adding zero to the number, does not affect the result. But just passes it through the 4 bit binary adder.

Refer Fig. 6.11.

3.1 Packed BCD Addition :

A packed BCD number is nothing but combination of two BCD digits i.e. 34, 59, 60 and so on. hat we designed under BCD adder was for single BCD digit (4 bit number), now we require to add two :D digits (8 bit), ∴ logically the answer is, we have to cascade BCD adder circuit. So total we require ır 7483. Refer Fig. 6.12.

Consider example :

```
)   19    0 0 0 1    1 0 0 1   [X7 - X0]
+   16    0 0 0 1    0 1 1 0   [Y7 - Y0]
    35    0 0 1 0    1 1 1 1
                   + 0 1 1 0
          0 0 1 1    0 1 0 1
             3          5
```

```
  90          1 0 0 1   0 0 0 0   [X7 - X0]
+ 91          1 0 0 1   0 0 0 1   [Y7 - Y0]
 181    [1]   0 0 1 0   0 0 0 1
          +   0 1 1 0   0 0 0 0
        1     1 0 0 0   0 0 0 1
        1        8         1
```

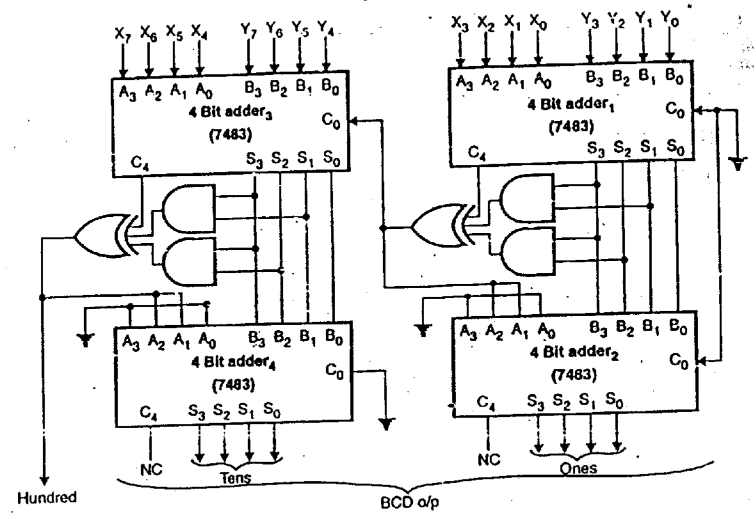



Fig. 6.12

6.4 Excess – 3 Adder :

Basic for excess – 3 addition :

(1) Convert given numbers in their XS-3 format. This is achieved by adding 3 to it.

(2) Now add both the XS-3 numbers.

(3) If carry generated ⟶ add 3 to the sum of two digits

else ⟶ subtract 3 from sum of two digits.

(4) For subtraction one can go for 2's complement method.

∴ $(3)_{10} = 0011$

```
1's complement  =   1 1 0 0
+            1    +       1
2's complement      1 1 0 1   ∴ add (1101) to the answer and neglect carry.
```

Take an example, add 8 and 4 in XS-3.

$(8)_{10} \Rightarrow (1000)_2$ ∴ XS-3 = 1 0 0 0 ; $(4)_{10} \Rightarrow (0100)_2$ ∴ XS-3 = 0 1 0 0

```
  1 0 0 0        0 1 0 0
+ 0 0 1 1  ;   + 0 0 1 1
  1 0 1 1        0 1 1 1
```

Now add both :

1 0 1 1 ⟶ XS – 3 (8)
0 1 1 1 ⟶ XS – 3 (4)
carry → [1] 0 0 1 0 As carry is 1, add 3

\+ \+
0 0 1 1 0 0 1 1
0 1 0 0 0 1 0 1
1 2 ⟶ in XS–3

Take one more example, add 4 and 3
4 in XS – 3 is 0111
3 in XS – 3 is 0110

∴ 4 0 1 1 1
\+ 3 0 1 1 0
7 [0] 1 1 0 1 As carry is not generated subtract 3. Subtract 3 is same as
\+ 1 1 0 1 adding 1101.
[1]
1 0 1 0
Neglect carry 7 in XS – 3

Now here the problem is how you will come to know when to add 0011 or 1101 (i.e. subtract 3). The answer is 'carry' bit. So here you have to design the circuit which will add 0011 or 1101 depending upon carry bit. Therefore let's compare 0011 and 1101.

B_3 B_2 B_1 B_0
0 0 1 1 ⟶ when carry = 1
1 1 0 1 ⟶ when carry = 0

If you observe B_0 bit in both situation, it is high. Now compare B_1, B_2 and B_3, you will find that one is complement of another.

∴ when Carry = 1 $B_1 = 1$ $B_2 = B_3 = 0$
Carry = 0 $B_1 = 0$ $B_2 = B_3 = 1$
follows carry complement of carry

Circuit is,

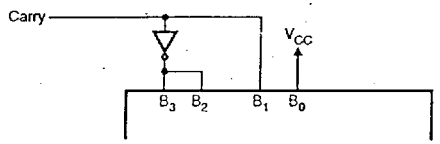


Fig. 6.13 (a)

Finally, again we required two level of adder circuit. Refer Fig. 6.13 (b).

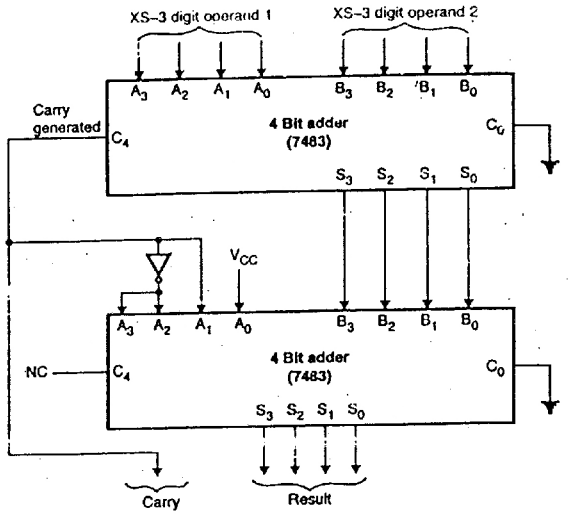


Fig. 6.13 (b)

6.5 Binary Subtractor :

Before we start designing binary subtractor using 7483. Let's have basics refreshed.

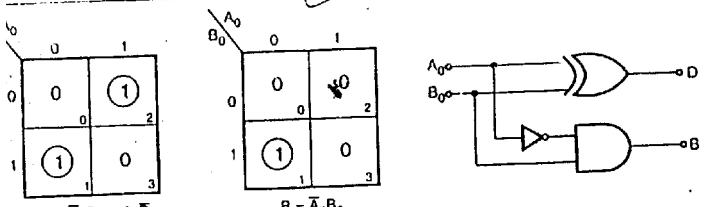
6.5.1 Half Subtractor :

This is basic building block for subtraction of two, one bit number. The truth Table is as shown below in Truth Table 6.4.

Inputs		Outputs	
A_0	B_0	Difference (D)	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Table 6.4

The rules for subtraction has been already covered in chapter 2.
K-Map : The circuit is as shown in Fig. 6.14.



$D = \overline{A}_0B_0 + A_0\overline{B}_0$
$= A_0 \oplus B_0$

$B = \overline{A}_0B_0$

(a) (b) (c)

Fig. 6.14

isadvantages of half subtractor is, it doesn't cater for previous borrow, if any.

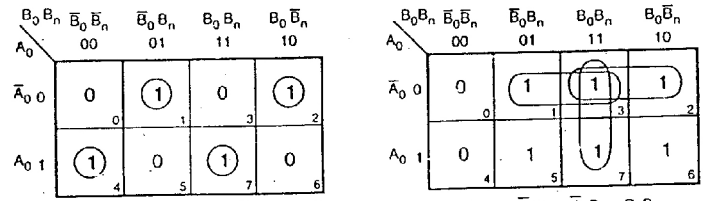
Full Subtractor :

o overcome problem of half subtractor, we design full subtractor. Assume A_0 and B_0 as one bit and B_n as previous borrow input, also output difference is D_n and borrow is B_{n+1}.

Inputs			Outputs	
A_0	B_0	B_n	D_n	B_{n+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table 6.5

ap :



$D = A_n(\overline{B}_0\overline{B}_n + B_0B_n) + \overline{A}_0(\overline{B}_0B_n + B_0\overline{B}_n)$
$= A_0(\overline{B}_0 \oplus \overline{B}_n) + \overline{A}_0(B_0 \oplus B_n)$
$= A_0 \oplus B_0 \oplus B_n$

$B_{n+1} = \overline{A}_0B_n + \overline{A}_0B_0 + B_0B_n$

Fig. 6.15 (a)

The circuit diagram is as shown in Fig. 6.15 (b).

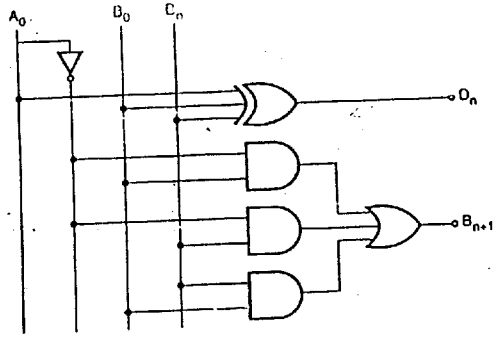


Fig. 6.15 (b)

Basically we never go for direct subtraction method. We use negative number representation i.e. 1's complement and 2's complement number system, because A – B = A + (–B).

So basic operation we perform is add A with negative B number. Secondly, we normally work with multidigit number, so we should go for multibit subtractor.

6.5.3 Binary Parallel Subtractor using 1's Complement Method :

We have already seen 1's complement method in chapter 2. We normally complement the operand which is to be represented in –ve number. Secondly after addition A and (–B), if carry generated, it is added in the answer therefore also called as *end around carry*.

The circuit is as shown in Fig. 6.16.

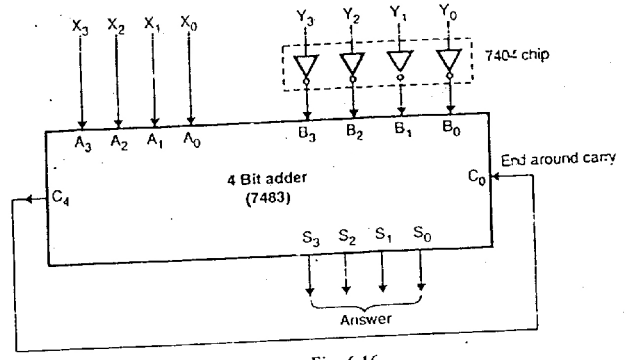


Fig. 6.16

Fig. 6.16 uses 4 bit adder. The same circuit can be implemented using full adder circuit, refer Fig. 6.17.

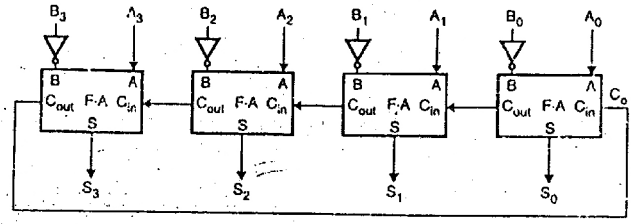


Fig. 6.17

Basically working of both the circuit is same. We will see working of Fig. 6.16 which is also valid for Fig. 6.17.

Case – I : Perform 4 – 2, ∴ operation is 4 + (–2)

∴ $(4)_{10} = 0\ 1\ 0\ 0 = X_3\ X_2\ X_1\ X_0 = A_3\ A_2\ A_1\ A_0$

$(2)_{10} = 0\ 0\ 1\ 0 = Y_3\ Y_2\ Y_1\ Y_0$

1's complement of $(2)_{10} = 1\ 1\ 0\ 1\ (B_3\ B_2\ B_1\ B_0)$

∴	A_3	A_2	A_1	A_0		0	1	0	0
+	B_3	B_2	B_1	B_0	+	1	1	0	1
					[1]	0	0	0	1
						↓			1
					C_4	0	0	1	0

$\longrightarrow S_3\ S_2\ S_1\ S_0$

Carry (Indicates answer in true form)

Case – II : Perform (2) – (4), ∴ operation is (2) + (–4)

∴ $(2)_{10} = 0\ 0\ 1\ 0 = X_3\ X_2\ X_1\ X_0 = A_3\ A_2\ A_1\ A_0$

$(4)_{10} = 0\ 1\ 0\ 0 = Y_3\ Y_2\ Y_1\ Y_0$

1's complement of $(4)_{10} = 1\ 0\ 1\ 1 = B_3\ B_2\ B_1\ B_0$

	A_3	A_2	A_1	A_0		0	0	1	0
∴ +	B_3	B_2	B_1	B_0	+	1	0	1	1
					[0]	1	1	0	1
						↓			0
					C_4	1	1	0	1

$\longrightarrow S_3\ S_2\ S_1\ S_0$

$C_4 = 0$ carry indicates answer is in 1's complement

So if we want in true form again invert it

∴ $S_3\ S_2\ S_1\ S_0 = 1\ 1\ 0\ 1 = (-2)$

∴ Complement of $(-2)_{10} = 0\ 0\ 1\ 0 = (+2)$ (True form)

6.5.4 Binary Parallel Subtractor using 2's Complement Method :

Normally in practical condition we use 2's complement method of subtraction :

2's complement = 1's complement + 1

∴ To perform A – B, A + (–B) is performed, and B represented in 2's complement form. After addition if carry is generated, it is not taken in the final answer. It is used just to know that answer is in true form or not.

Refer Fig. 6.16, Output of inverter (7400) gives 1's complement. In 2's complement, end around carry is not required. Therefore C_4 and C_0 are not connected. Now we can use C_0 input to get 2's complement, therefore C_0 we will tie to logic 1.

So finally inverter gives 1's complement and C_0 adds 1, so we get 2's complement number. Refer Fig. 6.18.

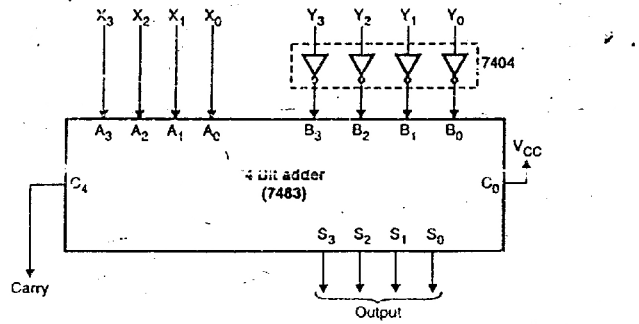


Fig. 6.18

Fig. 6.18 uses 4 bit adder block. Same circuit can be implemented using full adder. Refer Fig. 6.19.

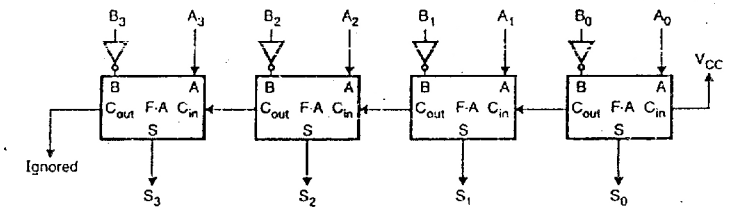


Fig. 6.19

The working of the circuit in Figs. 6.18 and 6.19 is same. We will learn how Fig. 6.18 works.

Working of circuit :

Case I : Perform $(4)_{10} - (2)_{10}$, $(4)_{10} + (-2)_{10}$

∴ $(+4)_{10} = 0\ 1\ 0\ 0 = X_3\ X_2\ X_1\ X_0 = A_3\ A_2\ A_1\ A_0$

$(+2)_{10} = 0\ 0\ 1\ 0 = Y_3\ Y_2\ Y_1\ Y_0$

1's complement = $1\ 1\ 0\ 1 = B_3\ B_2\ B_1\ B_0$, $C_0 = 1 = V_{CC}$

$\therefore$ $A_3\ A_2\ A_1\ A_0$ + $\overline{B_3}\ \overline{B_2}\ \overline{B_1}\ \overline{B_0}$ + C_0 = 0 1 0 0 + 1 1 0 1 + 1 = 0 0 1 0 $\longrightarrow S_2 S_2 S_1 S_0$

This gives 2's complement of $Y_3 Y_2 Y_1 Y_0$ given number

carry C_4 = 1

Answer = 0 0 1 0

Case II : Perform $(2)_{10} - (4)_{10}$, $(2) + (-4)_{10}$

$(+2) = 0\ 0\ 1\ 0 = X_3 X_2 X_1 X_0 = A_3 A_2 A_1 A_0$.

$(+4)_{10} = 0\ 1\ 0\ 0 = Y_3 Y_2 Y_1 Y_0$

1's complement = $1\ 0\ 1\ 1 = \overline{B_3}\ \overline{B_2}\ \overline{B_1}\ \overline{B_0}$, $C_0 = 1 = V_{CC}$

$\therefore$ Operation :

$A_3\ A_2\ A_1\ A_0$ + $\overline{B_3}\ \overline{B_2}\ \overline{B_1}\ \overline{B_0}$ + C_0 = 0 0 1 0 + 1 0 1 1 + 1 = 0 1 1 1 0 $\longrightarrow S_3 S_2 S_1 S_0$

This gives 2's complement of given input number $Y_3 Y_2 Y_1 Y_0$

C_4 (Carry) = 0

answer = 1 1 1 0

Carry indicates, to get answer find out 2's complement of $S_3 S_2 S_1 S_0$.

$\therefore$ $S_3 S_2 S_1 S_0$ 1 1 1 0 $\longrightarrow (-2)_{10}$

1's complement 0 0 0 1

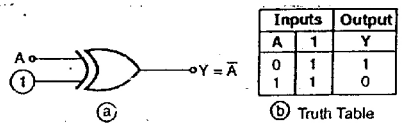
\+ 1

2's complement 0 0 1 0 $\longrightarrow (2)_{10}$

6.5.5 Controlled Addition / Subtraction :

Under this section we would like to design a circuit which will function as adder as well as subtractor. The operation should be performed according to the ***Control Input*** provided to the circuit. To achieve this we use EXOR gate with 4 bit parallel adder (7483).

Case I : As shown in Fig. 6.20, one terminal of EX–OR gate is tied to V_{CC} (logic 1) and other is given with variable A.



ⓐ

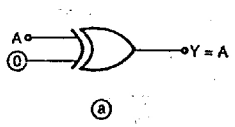
Inputs		Output
A	1	Y
0	1	1
1	1	0

ⓑ Truth Table

Fig. 6.20

As shown in truth table Output y is 'complement' of input variable A.

Case II : As shown in Fig. 6.21, one terminal of EX–OR gate is tied to GND (logic zero) and other is tied to variable A.



ⓐ

Inputs		Output
A	0	Y
0	0	0
1	0	1

ⓑ Truth Table

Fig. 6.21

As shown in truth table, Output Y is same as that of variable A. It means that EXOR with one terminal tied to 'zero' acts as ***transparent*** gate i.e. input is 'zero', output is 'zero', input is '1', output is '1'. Let's design circuit which will perform addition as well as subtraction. Refer Fig. 6.22.

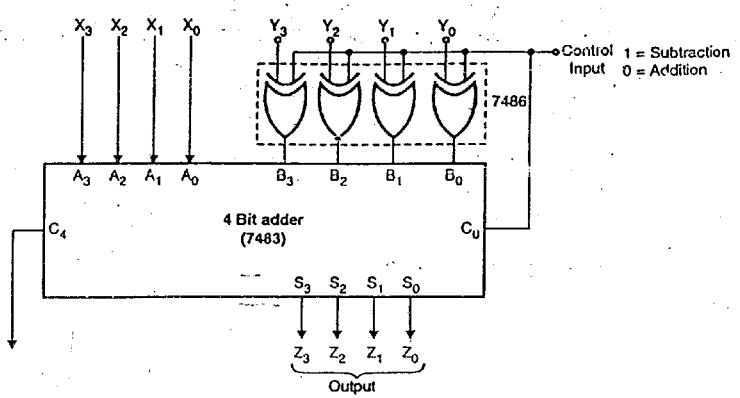


Fig. 6.22

Fig. 6.22 uses 4 bit adder. The same circuit can be implemented using full adder circuit. The working of the circuit will be identical. The circuit using full adder is shown in Fig. 6.23.

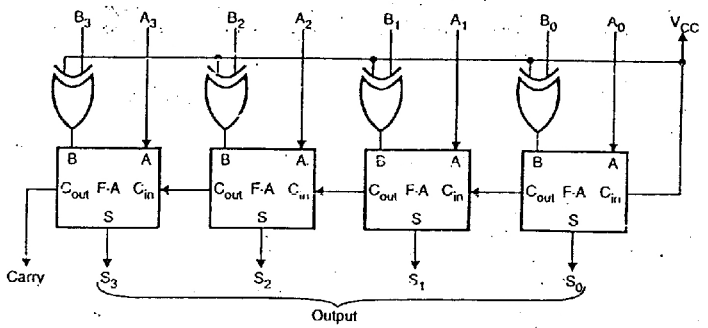


Fig. 6.23

Operation :

Case – I : Control Input = 0

When Control Input = 0, EX–OR will act as ***transparent*** gate i.e.

$B_3\ B_2\ B_1\ B_0 = Y_3\ Y_2\ Y_1\ Y_0$; $A_3\ A_2\ A_1\ A_0 = X_3\ X_2\ X_1\ X_0$ and $C_0 = 0$

∴ 7483 performs,

$$\begin{array}{rcccccccc} & A_3 & A_2 & A_1 & A_0 & = & X_3 & X_2 & X_1 & X_0 \\ + & B_3 & B_2 & B_1 & B_0 & = & Y_3 & Y_2 & Y_1 & Y_0 \\ + & & & & C_0 & = & & & & 0 \\ \hline & S_3 & S_2 & S_1 & S_0 & & Z_3 & Z_2 & Z_1 & Z_0 \end{array}$$

Adding '0' to $X_3\ X_2\ X_1\ X_0$ and $Y_3\ Y_2\ Y_1\ Y_0$ will not affect answer.

Case – II : Control Input = 1

When Control Input = 1, EX–OR will act as an *Inverter* gate

i.e. $B_3\ B_2\ B_1\ B_0 = \bar{Y}_3\ \bar{Y}_2\ \bar{Y}_1\ \bar{Y}_0$

Here $B_3\ B_2\ B_1\ B_0$ 1's complement $\bar{Y}_3\ \bar{Y}_2\ \bar{Y}_1\ \bar{Y}_0$

$+ \quad C_0 = + \quad 1 = + \quad C_0$

2's complement — 2's complement of given number $Y_3\ Y_2\ Y_1\ Y_0$

Final operation can be represented mathematically,

$$\begin{array}{rcccc} & A_3 & A_2 & A_1 & A_0 \\ + & B_3 & B_2 & B_1 & B_0 \\ + & & & & C_0 \\ \hline & S_3 & S_2 & S_1 & S_0 \end{array} = \begin{array}{rcccc} & X_3 & X_2 & X_1 & X_0 \\ + & \bar{Y}_3 & \bar{Y}_2 & \bar{Y}_1 & \bar{Y}_0 \\ + & & & & C_0 \\ \hline & Z_3 & Z_2 & Z_1 & Z_0 \end{array}$$

6.6 BCD Subtractor :

The 7483 is used as a basic, building block to implement BCD subtractor. For BCD subtraction nine's complement of the subtrahend is added to the minuend. The nine's complement of a number can be found by subtracting the given number from nine (9), or by adding 10 to 1's complement of given number.

For example : Find 9's complement of $(2)_{10}$.

$(2)_{10} = (0010)_2$

$\therefore (9)_{10} \quad 1\ 0\ 0\ 1$

$- (2)_{10} \quad 0\ 0\ 1\ 0$

$0\ 1\ 1\ 1 \longrightarrow$ Nine's complement of $(2)_{10}$

The same you achieve by ***Adding 10 to 1's complement of given number.***

$(2)_{10} \quad (0\ 0\ 1\ 0)_2$

1's complement of $(2)_{10}$ $(1\ 1\ 0\ 1)_2 \longrightarrow (-2)_{10}$

$(10)_{10} \longrightarrow 1\ 0\ 1\ 0$

$+ (-2)_{10} \longrightarrow 1\ 1\ 0\ 1$

carry → $\boxed{1}\ 0\ 1\ 1\ 1 \longrightarrow$ Nine's complement of $(2)_{10}$

(don't take carry into account)

This method is more easier to implement therefore most widely used. The circuit using the same is shown in Fig. 6.24.

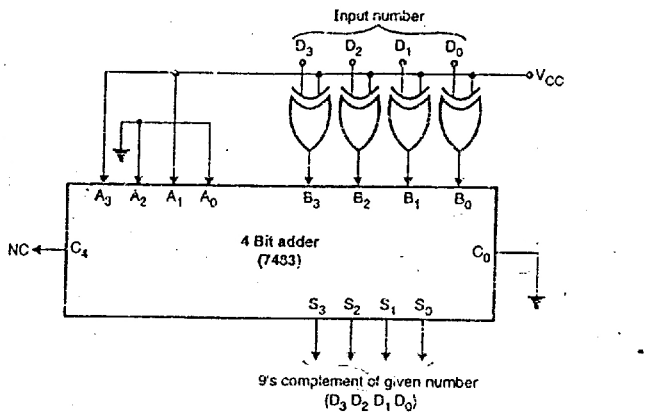


Fig. 6.24

6.6.1 Working of Circuit :

Take $D_3\ D_2\ D_1\ D_0 = (0\ 1\ 0\ 0)_2 = (4)_{10}$

$\therefore \quad B_3\ B_2\ B_1\ B_0 = \bar{D}_3\ \bar{D}_2\ \bar{D}_1\ \bar{D}_0 = 1\ 0\ 1\ 1$, $C_0 = 0$ (GND)

$\therefore \quad A_3\ A_2\ A_1\ A_0 \longrightarrow \quad 1\ 0\ 1\ 0$

$+ \quad B_3\ B_2\ B_1\ B_0 \longrightarrow \quad 1\ 0\ 1\ 1$

Ignore carry → $\boxed{1} \qquad 0\ 1\ 0\ 1 \rightarrow$ 9's complement of $(4)_{10}$ i.e. $(5)_{10}$

As stated earlier to perform BCD subtraction, the 9's complement of subtrahend (operand 2) is added to the minuend (operand 1). Let's first see the BCD subtractor circuit. Refer Fig. 6.25.

As shown, 4 bit adder$_1$ used to find 9's complement of operand 2.

4 bit adder$_2$, 4 bit adder$_3$ provides simple ***BCD adder*** circuit.

4 bit adder$_4$ is used to get answer in true form.

Let's study two examples for the same :

Case – I : Perform BCD subtraction $(8)_{10} - (3)_{10}$

Here $(8)_{10}$ = operand 1 = $(1000)_2$ = $X_3\ X_2\ X_1\ X_0$

$(3)_{10}$ = operand 2 = $(0011)_2$ = $Y_3\ Y_2\ Y_1\ Y_0$

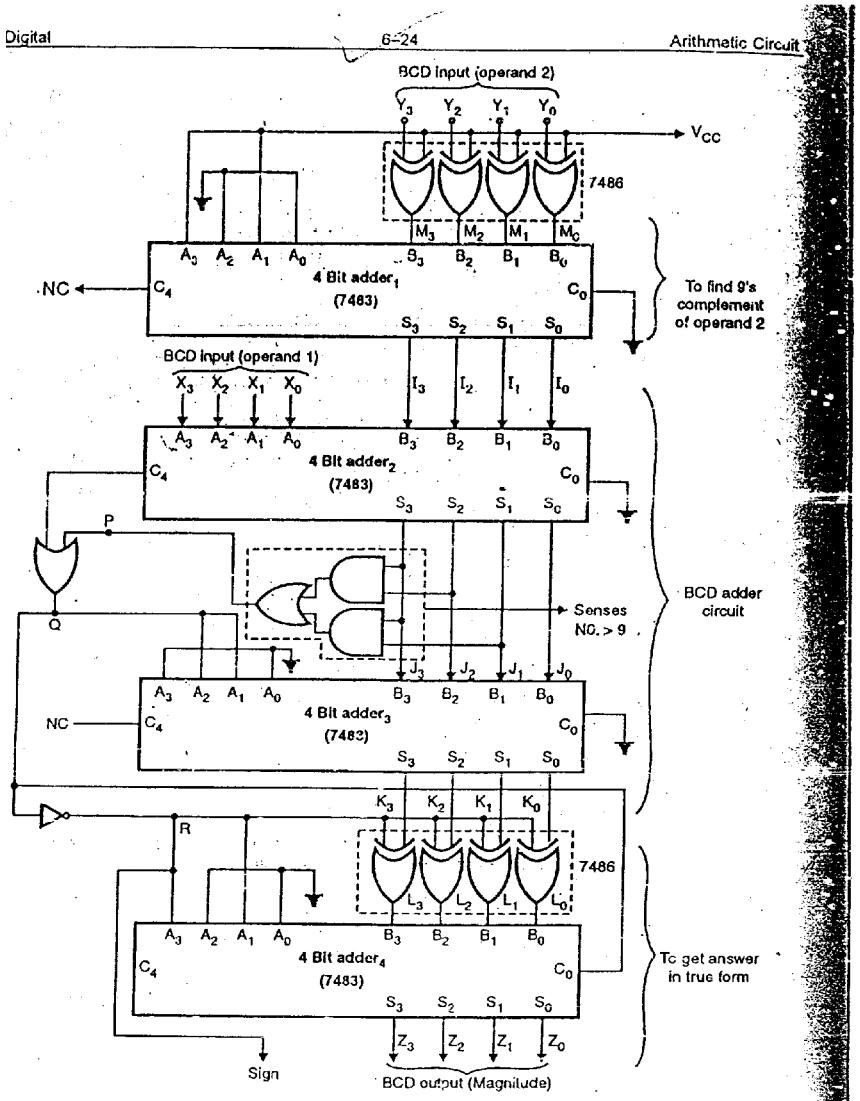


Fig. 6.25

Operation performed by 4 bit adder$_1$: This is to find 9's complement of $(3)_{10}$

$A_3\ A_2\ A_1\ A_0 \longrightarrow 1\ 0\ 1\ 0 \longrightarrow 1\ 0\ 1\ 0 \longrightarrow 1\ 0\ 1\ 0$

$+\ B_3\ B_2\ B_1\ B_0 \longrightarrow M_3\ M_2\ M_1\ M_0 \longrightarrow \bar{Y}_3\ \bar{Y}_2\ \bar{Y}_1\ \bar{Y}_0 \longrightarrow 1\ 1\ 0\ 0$

$S_3\ S_2\ S_1\ S_0 \qquad I_3\ I_2\ I_1\ I_0 \qquad I_3\ I_2\ I_1\ I_0 \quad \boxed{1}\ 0\ 1\ 1\ 0$

↑ carry ignored

$\therefore$ 9's complement of $(3)_{10} = (0110)_2 \longrightarrow (6)_{10}$

Operation performed by 4 bit adder$_2$:

$A_3\ A_2\ A_1\ A_0 \longrightarrow X_3\ X_2\ X_1\ X_0 \longrightarrow 1\ 0\ 0\ 0$

$+\ B_3\ B_2\ B_1\ B_0 \longrightarrow I_3\ I_2\ I_1\ I_0 \longrightarrow 1\ 1\ 1\ 0$

$S_3\ S_2\ S_1\ S_0 \qquad J_3\ J_2\ J_1\ J_0 \quad \boxed{0}\ 0\ 1\ 1\ 0$

↑ carry

Operation performed by 4 bit adder$_3$:

As $J_3\ J_2\ J_1\ J_0 = (1110)_2$ which is greater than 9, logic level at point 'P' is '1'. $\therefore$ Q = 1

$\therefore$ 7483_1 will add $(6)_{10} = (0110)_2$ to $J_3\ J_2\ J_1\ J_0$.

$A_3\ A_2\ A_1\ A_0 \longrightarrow 0\ 1\ 1\ 0 \longrightarrow 0\ 1\ 1\ 0$

$+\ B_3\ B_2\ B_1\ B_0 \longrightarrow J_3\ J_2\ J_1\ J_0 \longrightarrow 1\ 1\ 1\ 0$

$S_3\ S_2\ S_1\ S_0 \qquad K_3\ K_2\ K_1\ K_0 \quad \boxed{1}\ 0\ 1\ 0\ 0 \rightarrow K_3\ K_2\ K_1\ K_0$

↑ carry

Operation performed by 4 bit adder$_4$:

As point Q = 1, point R = 0. This will make EX–OR gate transparent.

$\therefore$ $K_3\ K_2\ K_1\ K_0 = L_3\ L_2\ L_1\ L_0$

Finally operation is :

$A_3\ A_2\ A_1\ A_0 \longrightarrow 0\ 0\ 0\ 0 \longrightarrow 0\ 0\ 0\ 0$

$+\ B_3\ B_2\ B_1\ B_0 \quad +\ J_3\ J_2\ J_1\ J_0 \quad +\ 0\ 1\ 0\ 0$

$+\ C_0 \qquad +\ C_0 \qquad +\ 1$

$S_3\ S_2\ S_1\ S_0 \qquad Z_3\ Z_2\ Z_1\ Z_0 \qquad 0\ 1\ 0\ 1 \rightarrow (5)_{10}$

and sign = 0

$\therefore$ Final answer for $(8)_{10} - (3)_{10} = (5)_{10}$

Case II : Perform BCD subtraction for $(3)_{10} - (8)_{10}$

Here $(3)_{10}$ = Operand 1 = $(0\ 0\ 1\ 1)_2$ = $X_3\ X_2\ X_1\ X_0$

$(8)_{10}$ = Operand 1 = $(1\ 0\ 0\ 0)_2$ = $Y_3\ Y_2\ Y_1\ Y_0$

Operation performed by 4 bit adder$_1$:

$A_3\ A_2\ A_1\ A_0 \rightarrow 1\ 0\ 1\ 0 \rightarrow 1\ 0\ 1\ 0 \rightarrow 1\ 0\ 1\ 0$

$+\ B_3\ B_2\ B_1\ B_0 \rightarrow M_3\ M_2\ M_1\ M_0 \rightarrow \bar{Y}_3\ \bar{Y}_2\ \bar{Y}_1\ \bar{Y}_0 \rightarrow 0\ 1\ 1\ 1$

$S_3\ S_2\ S_1\ S_0 \qquad I_3\ I_2\ I_1\ I_0 \qquad I_3\ I_2\ I_1\ I_0 \quad \boxed{1}\ 0\ 0\ 0\ 1$

↑ ignored

$\therefore$ 9s complement of $(8)_{10} = 0\ 0\ 0\ 1 \longrightarrow (1)_{10}$

Operation performed by 4 bit adder$_2$:

$A_3\ A_2\ A_1\ A_0 \rightarrow X_3\ X_2\ X_1\ X_0 \rightarrow 0\ 0\ 1\ 1$

$+\ B_3\ B_2\ B_1\ B_0 \rightarrow I_3\ I_2\ I_1\ I_0 \rightarrow 0\ 0\ 0\ 1$

$S_3\ S_2\ S_1\ S_0 \qquad J_3\ J_2\ J_1\ J_0\ \boxed{0}\ 0\ 1\ 0\ 0$

carry

Operation performed by 4 bit adder$_3$:

As $J_3\ J_2\ J_1\ J_0 = 0\ 1\ 0\ 0$ and No carry generated $P = Q = 0$

$\therefore\ A_3\ A_2\ A_1\ A_0 \longrightarrow 0\ 0\ 0\ 0 \longrightarrow 0\ 0\ 0\ 0$

$B_3\ B_2\ B_1\ B_0 \longrightarrow J_3\ J_2\ J_1\ J_0 \longrightarrow 0\ 1\ 0\ 0$

$K_3\ K_2\ K_1\ K_0 \qquad 0\ 1\ 0\ 0$

Operation performed by 4 bit adder$_4$:

As point Q = 0, point R = 1. Therefore now EXOR gate will act as an *Inverter* i.e.

$L_3\ L_2\ L_1\ L_0 = \bar{K}_3\ \bar{K}_2\ \bar{K}_1\ \bar{K}_0$ and $C_0 = 0$

$A_3\ A_2\ A_1\ A_0 \longrightarrow 1\ 0\ 1\ 0 \rightarrow 1\ 0\ 1\ 0$

$+\ B_3\ B_2\ B_1\ B_0 + L_3\ L_2\ L_1\ L_0 + 1\ 0\ 1\ 1$

$+\ C_0 + C_0 + 0$

$S_3\ S_2\ S_1\ S_0 \qquad Z_3\ Z_2\ Z_1\ Z_0\ \boxed{1} \qquad 0\ 1\ 0\ 1 \rightarrow (5)_{10}$

ignore

$\therefore$ Answer is sign = 1 $\quad Z_3\ Z_2\ Z_1\ Z_0 = 0\ 1\ 0\ 1 = (5)_{10}$

So answer is – 5 as sign = 1

6.7 XS – 3 Subtractor :

Steps involved in XS – 3 subtraction are as follows :

Step I	:	Convert both number in their XS–3 format
Step II	:	Complement the subtrahend
Step III	:	Add complemented subtrahend to minuend
Step IV	:	After addition of these numbers
		If carry generated then
		(i) Result is positive
		(ii) Add carry to the result therefore called *end around carry*.
		else
		(i) Result is negative
		(ii) add $(13)_{10}$
		(iii) Take 1's complement of the result.

Now we will see the circuit for the same and analyse the circuit.

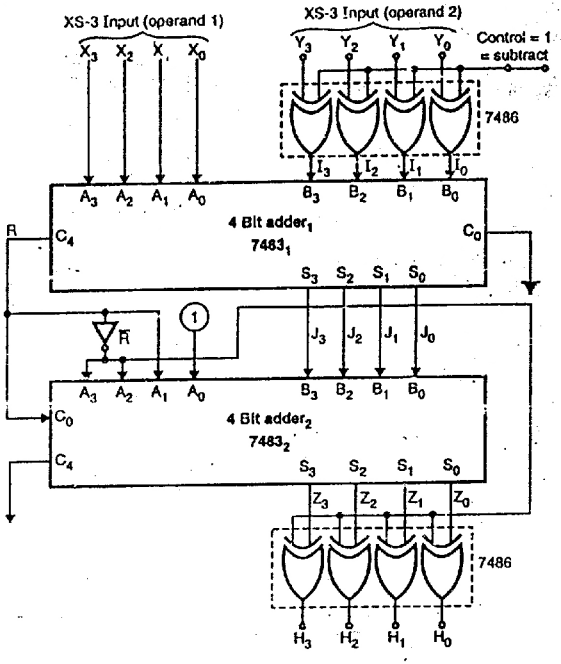


Fig. 6.26

Operation of the circuit :

Case I : Perform $(8)_{10} - (3)_{10}$ in XS – 3 code, $(8)_{10} + (-3)_{10}$

XS – 3 for $(8)_{10} = 1\ 0\ 1\ 1 = X_3\ X_2\ X_1\ X_0$

XS – 3 for $(3)_{10} = 0\ 1\ 1\ 0 = Y_3\ Y_2\ Y_1\ Y_0$

$\therefore\ I_3\ I_2\ I_1 I_0 = 1\ 0\ 0\ 1$ i.e. complement of $(0\ 1\ 1\ 0)_2$

Operation performed by 4 bit adder$_1$ is :

$\therefore\ A_3\ A_2\ A_1\ A_0 \rightarrow 1\ 0\ 1\ 1 \rightarrow 1\ 0\ 1\ 1$

$B_3\ B_2\ B_1\ B_0 \rightarrow I_3\ I_2\ I_1\ I_0 \rightarrow 1\ 0\ 0\ 1$

$S_3\ S_2\ S_1\ S_0 \qquad J_3\ J_2\ J_1\ J_0\ \boxed{1} \qquad 0\ 1\ 0\ 0$

C_4

ition performed by 4 bit adder$_2$ is :

A_3	A_2	A_1	A_0	→	0	0	1	1	→	0	0	1	1
B_3	B_2	B_1	B_0	+	J_3	J_2	J_1	J_0	+	0	1	0	0
			C_0	+				1	+	0	0	0	1
S_3	S_2	S_1	S_0		Z_3	Z_2	Z_1	Z_0	0	1	0	0	0	→ Z_3 Z_2 Z_1 Z_0

Ignore

as C_4 of 7483_1 is equal to '1', R = 1, ∴ $\bar{R}$ = 0. ∴ $Z_3\ Z_2\ Z_1\ Z_0 = H_3\ H_2\ H_1\ H_0$ as EXOR gates are parent.

∴ $H_3\ H_2\ H_1\ H_0 = (1\ 0\ 0\ 0)_2$ = EX–3 code of $(5)_{10}$ with R = 1

II : Perform $(3)_{10} - (8)_{10}$ in XS–3

XS–3 for $(3)_{10}$ = 0 1 1 0 ⟶ $X_3\ X_2\ X_1\ X_0$

XS–3 for $(8)_{10}$ = 1 0 1 1 ⟶ $Y_3\ Y_2\ Y_1\ Y_0$

$B_3\ B_2\ B_1\ B_0$ = $I_3\ I_2\ I_1\ I_0$ = $\bar{Y}_3\ \bar{Y}_2\ \bar{Y}_1\ \bar{Y}_0$ = 0 1 0 0

ration performed by 4 bit adder$_1$:

A_3	A_2	A_1	A_0	→	X_3	X_2	X_1	X_0	→	0	1	1	0
B_3	B_2	B_1	B_0	→	I_3	I_2	I_1	I_0	→	0	1	0	0
S_3	S_2	S_1	S_0		J_3	J_2	J_1	J_0	0	1	0	1	0

↑ C_4 of 4 bit adder

As C_4 = 0, R = 0 ∴ $\bar{R}$ = 1

ration performed by 4 bit adder$_2$:

A_3	A_2	A_1	A_0	→	0	0	1	1	→	1	1	0	1
B_3	B_2	B_1	B_0	+	J_3	J_2	J_1	J_0	+	1	0	1	0
			C_0	+				R	+				0
S_3	S_2	S_1	S_0		Z_3	Z_2	Z_1	Z_0	1	0	1	1	1

low as R = 1, EX–OR gates connected at final output stage will act as complement or inverter gate,

$H_3\ H_2\ H_1\ H_0 = \bar{Z}_3\ \bar{Z}_2\ \bar{Z}_1\ \bar{Z}_0 = \bar{0}\ \bar{1}\ \bar{1}\ \bar{1} = 1\ 0\ 0\ 0$

where 1 0 0 0 is XS–3 code for $(5)_{10}$ with R = 0

So R will tell you about positive or negative sign.

4 : Using three half adders (HA), implement the following four Boolean functions. Draw a neat diagram of the arrangement using minimum hardware.

(i) $PQR + (\bar{P} + \bar{Q}\ \bar{R})$

(ii) $P \oplus Q \oplus R$

(iii) $PQ\bar{R}$

(iv) $\bar{P}QR + P\bar{Q}R$

Soln. : Let's take 1st half adder,

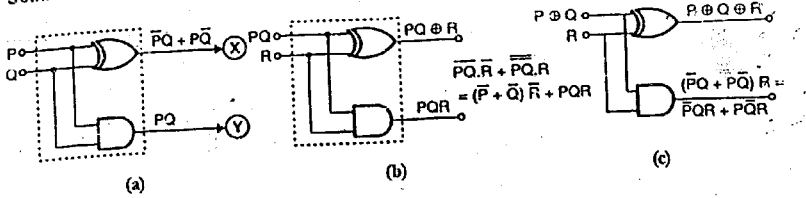


Fig. Ex. 6.4

Ex. 5 : Show that a full adder circuit consists of a three input EX-OR and a three input majority functions.

Soln. : For a full adder we have,

$$S = A \oplus B \oplus C$$

$$C_{n+1} = AB + BC_n + C_nA = AB\,(C_n + \bar{C}_n) + BC_n\,(A + \bar{A}) + AC_n\,(B + \bar{B})$$

$$= ABC_n + AB\bar{C}_n + \bar{A}BC_n + A\bar{B}\,C_n = \Sigma m\,(7, 6, 3, 5)$$

Since there are three variables, the minterms possible are 0 through 7 out of which minterms 3, 5, 6 and 7 has a majority of logical 1's in their binary equivalent. Thus, C_{n+1} is a three input majority function.

Ex. 6 : Using half adder and additional gates, design a controlled half adder/half subtractor such that when control signal is logic 0, the entire circuit behaves as a H.S and when it is 1, if behaves as a half adder (H.A.)

Soln. : Table 2 shows the truth table for the above mentioned problem where M(mode control) decides HA or HS operation.

Inputs					Outputs	
Decimal	A	B	M		(D/S) x	(B/C) y
0	0	0	0	HS	0	0
2	0	1	0		1	1
4	1	0	0		1	0
6	1	1	0		0	0
1	0	0	1	HA	0	0
3	0	1	1		1	0
5	1	0	1		1	0
7	1	1	1		0	1
					Difference/Sum	Borrow/Carry

Table 2

The K map is drawn in Fig. Ex. 6.6(a).

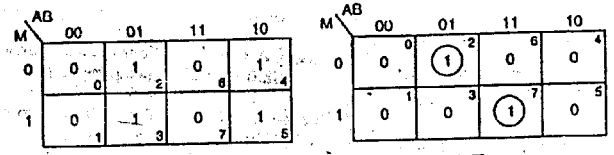


Fig. Ex. 6.6(a)

Circuit : Refer Fig. Ex. 6.6(b)

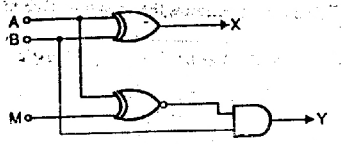


Fig. Ex. 6.6(b)

Ex. 7 : Implement a full subtractor using two half subtractors and OR gate.

Soln. : We know that,

For half subtractor

$D = A_0 \oplus B_0$

$B = \bar{A}_0 B_0$

Let's say $A_0 = X$, $B_0 = Y$

$\therefore$ For half subtractor

$D = x \oplus y$

$B = \bar{x} y$

For full subtractor

$D_n = A_0 \oplus B_0 \oplus B_n$

$B_{n+1} = \bar{A}_0 B_n + \bar{A}_0 B_0 + B_0 B_n$

$D_n = x + y + B_n$

$B_{n+1} = \bar{x} B_n + \bar{x} y + y B_n$

The circuit is as shown in Fig. Ex. 6.7

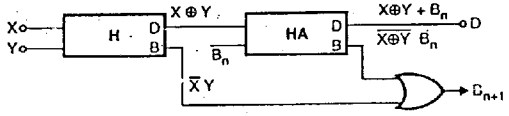


Fig. Ex. 6.7

$$B_{n+1} = \overline{x \oplus y} \cdot B_n + \bar{x} y$$

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

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

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

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

$$= \bar{x}y + \bar{x}B_n + xyB_n$$

$$= y (\bar{x} + xB_n) + \bar{x}B_n$$

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

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

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

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

6.8 Multiplier :

Multiplication is accomplished by successive addition of shifted partial products. Multiplying a multiplicand by 1/0 is equivalent to ANDing the multiplier bit by all the multiplicand bits, and then adding the shifted partial products A basic building block is illustrated in Fig. 6.27.

Since a n bit by n bit multiplication results in a 2n bit solution, we generally work with registers as shown in Fig. 6.28.

Fig. 6.28 shows a register configuration to perform multiplication, where a register is an electronic circuit to hold multiple bit data. Every time a multiplier bit (LSB onwards) is multiplied with the multiplicand, it has no further use. Also, since the partial product is shifted by one bit, an additional bit space is generated. To reduce hardware, the partial product is shifted into the multiplier register from MSB side, while LSB bit is lost and does not affect the circuit performance as its purpose has been served.

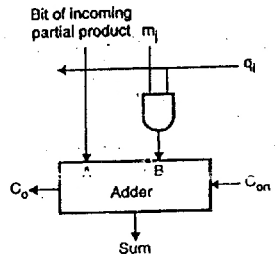


Fig. 6.27

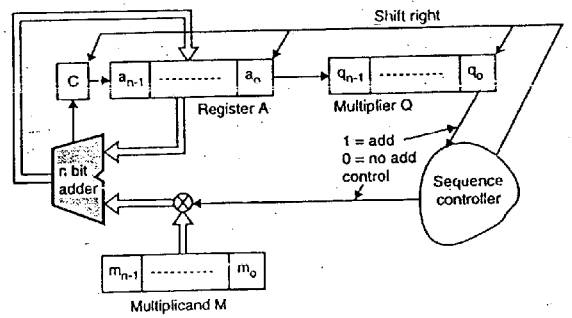


Fig. 6.28

The register A is initially loaded with zero, and is used to save partial result subsequently. Consider an example for above hardware where M = 1100 and Q = 1001. The multiplication steps are shown as follows i.e.

C	A	Q		
0	0 0 0 0	1 0 0	1	
	+ 1 1 0 0			add M
0	1 1 0 0	1 0 0	1	shift right
0	0 1 1 0	0 1 0	0	shift right
0	0 0 1 1	0 0 1	0	shift right
0	0 0 0 1	1 0 0	1	
0	0 0 0 1	1 0 0	1	
	+ 1 1 0 0			add M
0	1 1 0 1	1 0 0	1	shift right
0	0 1 1 0	1 1 0	0	
	64 32	8 4	=	108

The steps involved are to check the LSB bit of multiplier Q and take the following steps :

(i) If $Q_0 = 0$ shift CAQ one bit right

(ii) If $Q_0 = 1$ add M to A keeping new result in A and then shift CAQ one bit right.

In both cases, every time a shift is performed, carry bit C is copied into itself.

8.1 Fast Multiplier :

As seen in previous topic every multiplier bit is ultiplied with the multiplicand. Also the previous chnique fails if the multiplicand or multiplier is gative. This is because partial product for a negative ultiplicand must be negative. One can avoid this oblem by converting both, multiplier and multiplicand positive numbers, perform the multiplication and entually take two's complement of the result only, if sign of the two original numbers differed. This can achieved using a technique called BOOTHS GORITHM which also speeds up multiplication. The wchart for Booths algorithm is depicted in Fig. 6.29.

Consider an example using Booths algorithm as own, 6 × 5 where,

M = 6 = 0110

Q = 5 = 0101

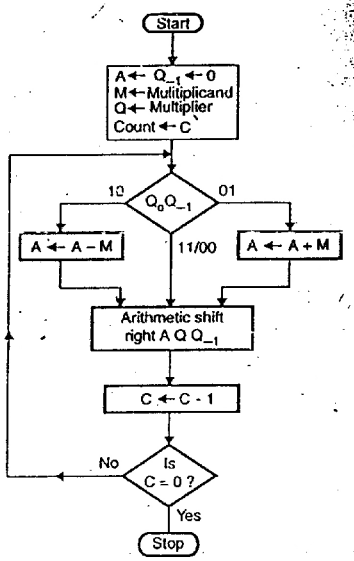


Fig. 6.29

i.e.,

	A	Q	Q_{-1}	
	0 0 0 0	0 1 0 1	0	Initially
−	0 1 1 0			A ← A − M
	1 0 1 0	0 1 0 1	0	Shift
	1 1 0 1	0 0 1 0	1	A ← A + M
+	0 1 1 0			
	0 0 1 1	0 0 1 0	1	Shift
	0 0 0 1	1 0 0 1	0	A ← A − M
−	0 1 1 0			
	1 0 1 1	1 0 0 1	0	Shift
	1 1 0 1	1 1 0 0	1	A ← A + M
+	0 1 1 0			
	0 0 1 1	1 1 0 0	1	Shift
	0 0 0 1	1 1 1 0	0	
	16 8 4 2		= 30	

Four shifts as number of bits in multiplier are four.

The MSB bit is copied onto itself and shifted Q_{-1} bit is always zero initially.

This is how the electronic circuit performs the operation. One may however use the Booth multiplier recoding table as shown in :

Multiplier bit		Multiplicand selected by bit i
i	i − 1	
0	0	0 X M
0	1	+1 X M
1	0	−1 X M
1	1	0 X M

(i) Multiplicand and multiplier both positive 7 × 3 = 21

				0	1	1	1	
				×	0	0	1	1 (0)
1	1	1	1	1	0	0	1	1 − 0
0	0	0	0	0	0	0	X	1 − 1
0	0	0	1	1	1	X	X	0 − 1
0	0	0	1	0	1	0	1	= 21

(TWOS COMPLEMENT FORM)
← sign extend in logical 1 as number is negative

(ii) Multiplicand positive and multiplier negative 7 × − 3 = 21

				0	1	1	1	
				×	1	1	0	1 (0)
1	1	1	1	1	0	0	1	1 − 0
0	0	0	0	1	1	1	X	0 − 1
1	1	1	0	0	1	X	X	1 − 0
1	1	1	0	1	0	1	1	= Two's complement of 21

(iii) Multiplicand negative and multiplier positive − 7 × 3 = − 21

				1	0	0	1	
				×	0	0	1	1 (0)
0	0	0	0	0	1	1	1	1 − 0
0	0	0	0	0	0	0	X	1 − 1
1	1	1	0	0	1	X	X	0 − 1
1	1	1	0	1	0	1	1	= Two's complement of 21

← sign extend to 0 as the original number is negative

$\therefore\ Q_{n+1} = 0,\ \bar{Q}_{n+1} = 1$

Conclusion : When S = 0, R = 1, irrespective of previous state, $Q_{n+1} = 0$, $\bar{Q}_{n+1} = 1$. This state is *'RESET'* state of circuit.

Case III :

(i) S = 1, R = 0, $Q_n = 0$, $\bar{Q}_n = 1$

$\therefore$ Input to NOR1 is R = 0, $\bar{Q} = 1$; produces Q = 0.

$\therefore$ Input to NOR2 is S = 1, Q = 0.

$\therefore$ Q will change from 1 to 0, $\therefore \bar{Q}_{n+1} = 0 = \bar{Q}$.

$\therefore$ Input to NOR1 is R = 0, $\bar{Q} = 0$.

$\therefore$ Q will change from 0 to 1, $\therefore Q_{n+1} = 1 = Q$.

Now input to NOR2 is S = 1 and Q = 1. $\therefore \bar{Q} = 0$.

$\therefore$ Input to NOR1 is R = 0, $\bar{Q} = 0$. $\therefore$ Q = 1.

Means circuit reached to stable state, $Q_{n+1} = 1$, $\bar{Q}_{n+1} = 0$.

(ii) S = 1, R = 0, $Q_n = 1$, $\bar{Q}_n = 0$.

As seen in above case $Q_n = 1$, $\bar{Q}_n = 0$ is stable state when S = 1, R = 0.

Conclusion : When S = 1, R = 0, irrespective of previous condition $Q_{n+1} = 1$, $\bar{Q}_{n+1} = 0$. This state is *'SET'* state of the circuit.

Case IV : When S = 1, R = 1, we know the fact that when either of the input to NOR gate is 1, output of gate is 0. This condition leads us to 'RACE' condition of the circuit. It means both the outputs will try to reach logic1; and therefore, this state is ambiguous or undefined. Normally referred as *RACE*.

State table

For SR FF, we have two inputs, i.e. S and R. Q output represents, state of FF. As Q is single output, here will be two states i.e. 0 and 1. Note that $\bar{Q}$ is simply inverted output of Q. Therefore not treated as eparate output. Thus state table will be as follows :

Present State	Next state Q (n + 1)			
Q(n)	SR = 00	SR = 01	SR = 10	SR = 11
0	0	0	1	Race
1	1	0	1	Race

Table 8.1 : State table for SR FF

Race condition cannot be shown or predicted.

State Diagram

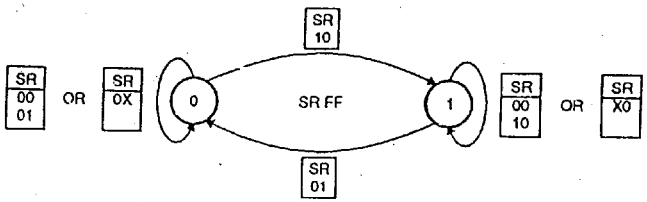


Fig. 8.9 : State diagram of SR FF

Thus if we observe we have following state transitions, with different combinations of SR input.

Previous state (Q(n))	to	Next state (Q(n + 1)	If S	R
0		0	0	X
0		1	1	0
1		0	0	1
1		1	X	0

Table 8.2 : Excitation table

Read the table as given below,

If previous state is Q(n) then to switch over to next state i.e. Q (n + 1) input S and R should be Q(n + 1).

For example, if previous state is 0, then to switch over to next state 1, input S and R should be 10 [S = 1, R = 0].

The above Table 8.2 is referred as excitation table.

State Equation

For SR FF, the state equation is,

$$Q(n+1) = S + \bar{R}\,Q(n)$$

8.3.1 Nand Latch :

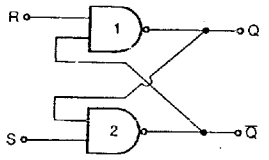


Fig. 8.10(a) : Nand latch

S	R	Q_{n+1}	$\bar{Q}_{n+1}$	
0	0	RACE	RACE	
0	1	0	1	→ Reset
1	0	1	0	→ Set
1	1	NC (Q_n)	NC ($\bar{Q}_n$)	→ Inactive

Fig. 8.10(b) : Truth Table

Input		Output	
CLK	D	Q_{n+1}	$\bar{Q}_{n+1}$
0	X	NC	NC
1	X	NC	NC
↓	X	NC	NC
↑	0	0	1
↑	1	1	0

Fig. 8.19(c) : Truth table

State table

Present State	Next State	
Q(n)	D = 0	D = 1
0	0	1
1	0	1

Table 8.8 : State table for D FF

State diagram

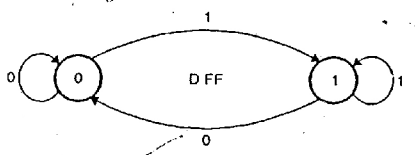


Fig. 8.20 : D FF state diagram

Previous state	to	Next state	If	D
0		0		0
0		1		1
1		0		0
1		1		1

Table 8.9 : Excitation table for D FF

State equation

$$Q(n+1) = D$$

x. 3 : Draw output waveform for given D input, for positive edge triggered D FF.

Soln. :

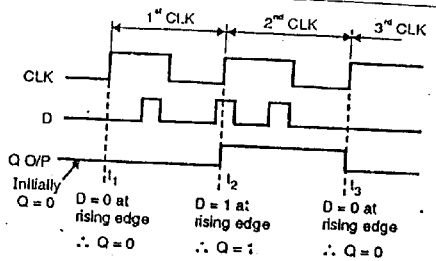


Fig. Ex. 8.3

Available D FF chips are :

(1) Positive edge triggered D type FF – IC 7474
(2) Positive level triggered D type FF – IC 7475 (transparent latch)
(3) Quad D flip-flop with clear – IC 74175
(4) Octal D latch – IC 74373

8.5 Flip-Flop Timing Consideration :

Manufacturer of flip-flop IC will specify several important timing parameters and characteristics that must be considered before FF is used in any circuit.

For explaining the same we will use D FF.

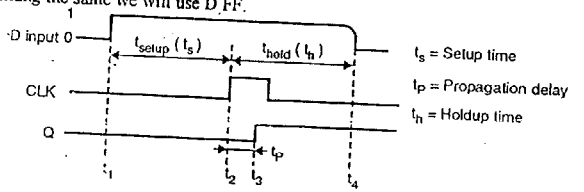


Fig. 8.21

Propagation Delay (Refer Fig. 8.21)

In IC's we use diodes and transistors. Transistor is operated either in saturation or cut off. Transistor takes time to switch from saturation to cutoff or V.V. Same way diode also takes small time to turn ON/OFF. This switching time is in nanosecond. But is main cause for propagation delay (t_p).

Definition : It is the amount of time it takes for output of gate or FF to change state after the input changes or CLK edge hits. Typical value = 10 – 20 η sec.

This time should be considered when digital circuit used in high speed circuit. Propagation delay is measured for both, LOW to HIGH and HIGH to LOW transition.

$t_p = t_3 - t_2$

t_{PLH} → From CLK to Q (LH → LOW to HIGH)
t_{PHL} → From CLK to Q (HL → HIGH to LOW)
t_{PLH} → $\overline{PRE}$ to Q } $\overline{PRE}$ – Preset
t_{PHL} → $\overline{CLR}$ to Q } $\overline{CLR}$ – Clear

Setup Time (t_s) (t_{setup}) ➥ [ASKED IN EXAM - DEC. 99, DEC. 98 !!!]

The setup time ($t_2 - t_1$) is the minimum length of time input should be present before CLK edge rives. The main cause of setup time is stray capacitance at the input side. If this time requirement is not t, the FF may not respond reliably when CLK edge arrives. Typical timing is from 5 to 40 η sec.

Holdup Time (t_h) (t_{hold}) ➥ [ASKED IN EXAM - DEC. 99, DEC. 98 !!!]

The holdup time ($t_4 - t_2$) is minimum length of time, input must be present after CLK edge has uck. The main cause of holding input is because of switching time of internal transistors and diodes. If s time is not met, one cannot assure about switching of internal circuit and FF will not work properly. Typical holdup time is ranging from 0 to 10 η sec.

Maximum CLK Frequency (f_{max})

This is the maximum CLK frequency that can be applied to CLK input of FF, for proper and stable eration.

$$f_{max} = 1/t_c = \frac{1}{t_{setup} + t_{pd(FF)} + t_{ns}}$$

where, t_c = Clock period
t_{ns} =Propagation delay of next state decoder
t_{setup} =Setup time
$t_{pd\,(FF)}$ =Flip flop propagation delay

Refer Fig. 8.22.

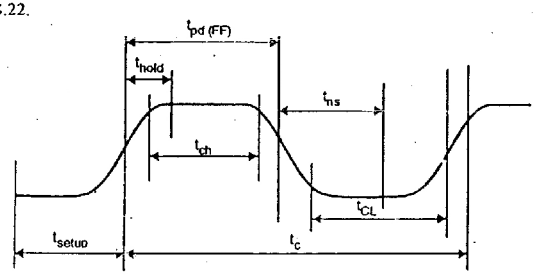


Fig. 8.22

Clock Pulse HIGH and LOW Time

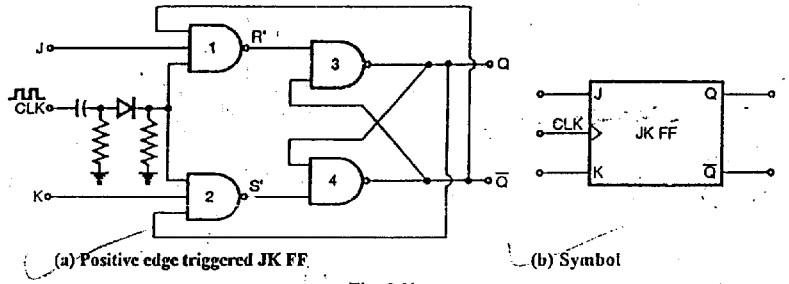
Refer Fig. 8.22. The manufacturer specifies minimum time duration that the CLK signal must remain W before it goes HIGH, it is called as CLK pulse low time (t_{CL} or t_{WCL}) and minimum time duration

that the CLK signal must remain HIGH before it returns LOW, is called CLK pulse high time (t_{CH} or t_{WCH}) typical timing for $t_{W\,(H)}$ and $t_{W\,(L)}$ are 15 η sec. to 100 η sec.

Asynchronous Active Pulse Width

As we have seen in section 8.3.5 preset and clear are asynchronous inputs. For the inputs manufacturer specify minimum time duration, $t_{W\,(L)}$ ⟶ Preset or clear (For active low asynchronous input.).

8.6 JK Flip-Flop :



(a) Positive edge triggered JK FF (b) Symbol

Fig. 8.23

As shown in Fig. 8.23(a) NAND 3/4 forms simple SR FF. Inputs to the NAND1 are J, $\bar{Q}$ and CLK (edge). Inputs to NAND2 are K, Q and CLK (edge). Now let's see how JK FF works and derive truth table.

Case I : When CLK is at *HIGH* or *LOW* level, output Q and $\bar{Q}$ remains in previous state. Because NAND1/2 produces '1' output, irrespective of Q, $\bar{Q}$, J and K inputs, as input to NAND1/2 is '0', after differentiator circuit.

Case II : When CLK = ↓ i.e. falling edge, output Q and $\bar{Q}$ are unchanged.

Case III : When CLK = ↑ and J = 0, K = 0,
S' = R' = 1. ∴ output does not change.
Conclusion : Above three cases are called *Inactive* condition of JK FF.

Case IV : CLK = ↑, J = 0 and K = 1.
Let's consider previous state of Q = 1 and $\bar{Q}$ = 0

$$R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{0 \cdot 0 \cdot 1} = 1.$$
$$S' = \overline{K \cdot Q \cdot CLK} = \overline{1 \cdot 1 \cdot 1} = 0.$$

∴ Q changes from 1 to 0. Q = 0, $\bar{Q}$ = 1 (Refer Table 8.8(b) of SR FF). We will again start analysis because Q and $\bar{Q}$ has been changed.

$$\text{Now, } R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{0 \cdot 1 \cdot 1} = 1$$
$$S' = \overline{K \cdot Q \cdot CLK} = \overline{1 \cdot 0 \cdot 1} = 1$$

Q and $\bar{Q}$ will remain unchanged unless and until next JK input is given with CLK. This state is stable *'RESET'* condition of JK FF.
Conclusion : When J = 0, K = 1 and CLK hits, irrespective of previous condition, Q = 0 and $\bar{Q}$ = 1.

Case V : CLK = ↑, J = 1, K = 0

Let's consider previous state of Q = 0 and $\bar{Q}$ = 1

$R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{1 \cdot 1 \cdot 1} = 0$

$S' = \overline{K \cdot Q \cdot CLK} = \overline{0 \cdot 0 \cdot 1} = 1$

∴ Q changes from 0 to 1, Q = 1, $\bar{Q}$ = 0.

Now $R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{1 \cdot 0 \cdot 1} = 1$

$S' = \overline{K \cdot Q \cdot CLK} = \overline{0 \cdot 1 \cdot 1} = 1$

∴ Q and $\bar{Q}$ will remain unchanged. This state is stable *'SET'* condition of JK FF.

Conclusion : When J = 1, K = 0 and CLK hits, irrespective of previous condition Q = 1 and $\bar{Q}$ = 0.

Case VI : When CLK = ↑, J = K = 1, this condition is more interesting. Here we are going to consider 2 CLK pulses.

(i) Let's say previous condition of Q = 1, $\bar{Q}$ = 0

∴ $R' = \overline{J.\bar{Q}.CLK} = \overline{1 \cdot 0 \cdot 1} = 1$

∴ $S' = \overline{K.Q.CLK} = \overline{1 \cdot 1 \cdot 1} = 0$

∴ $Q_{n+1} = 0$ and $\overline{Q_{n+1}} = 1$ $(Q_{n+1} = \bar{Q}_n, Q_{n+1} = \bar{1} = 0)$

(ii) Now let's say previous condition of Q = 0, $\bar{Q}$ = 1 (If you note Q_{n+1} of (i) = Q_n).

∴ $R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{1 \cdot 1 \cdot 1} = 0$

∴ $S' = \overline{K \cdot Q \cdot CLK} = \overline{1 \cdot 0 \cdot 1} = 1$

∴ $Q_{n+1} = 1$, $\overline{Q_{n+1}} = 0$ $(Q_{n+1} = \bar{Q}_n = \bar{0} = 1)$

From above two cases (i) and (ii) we conclude that when J = K = 1 and CLK hits $Q_{n+1} = \bar{Q}_n$ (means tput toggles).

Conclusion : This condition of JK FF gives us *TOGGLING* mode. Now let's summarize everything d make truth table :

Inputs			Outputs		
CLK	J	K	Q_{n+1}	$\bar{Q}_{n+1}$	
0	X	X	NC	NC	Case I
1	X	X	NC	NC	Case I
↓	X	X	NC	NC	Case II
↑	0	0	NC	NC	Case III
↑	0	1	0	1	Case IV
↑	1	0	1	0	Case V
↑	1	1	$\bar{Q}_n$	Q_n	Case VI

State table

Present state Q (n)	Next state Q (n + 1)			
	JK = 00	JK = 01	JK = 10	JK = 11
0	0	0	1	1
1	1	0	1	0

Table 8.10 : State table for JK flip-flop

State diagram

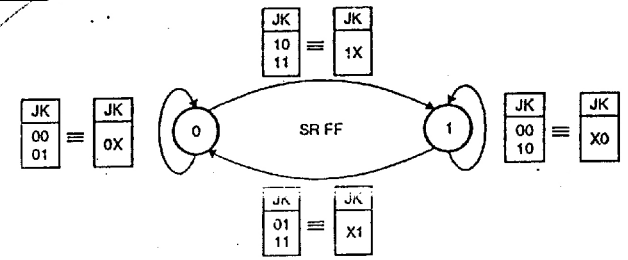


Fig. 8.24 : State diagram for JK-FF

Previous state	to	Next state	If	J	K
0		0		0	X
0		1		1	X
1		0		X	1
1		1		X	0

Table 8.11 : Excitation table for JK–FF

State Equation

$$Q(n+1) = J\,\overline{Q(n)} + \bar{K}\,Q(n)$$

Ex. 4 : Determine output Q for Fig. Ex. 8.4 :

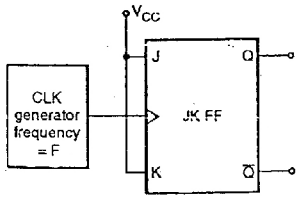


Fig. Ex. 8.4

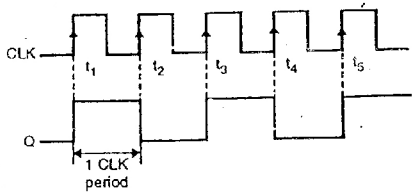


Fig. Ex. 8.4(a)

Soln. : When J = K = 1, output Q toggles.

Initially Q = 0.

When, 1st CLK edge hits $Q_{n+1} = \bar{Q}_n$. $Q_{n+1} = \bar{0} = 1$ (t_1)

2nd CLK edge hits $Q_{n+1} = \bar{Q}_n$ $Q_{n+1} = \bar{1} = 0$ (t_2)

3rd CLK edge hits $Q_{n+1} = \bar{Q}_n$ $Q_{n+1} = \bar{0} = 1$ (t_3)

and the chain will repeat.

The frequency of the Q output is F/2. Therefore this circuit is called as ÷ 2 circuit.

8.6.1 Race Condition in JK FF (Race Around Condition) :

➥ [ASKED IN EXAM - DEC. 96, MAY 97, DEC. 97 III]

We have started our discussion on JK FF, directly with positive edge triggered type. But how JK is going to behave if FF is level triggered ? Here we are going to see interesting RACE AROUND phenomena.

Remember three important points :

(1) When J = K = 1, JK FF output Q toggles

(2) When FF is level triggered, FF is active till *active level* is present

(3) Output Q and $\bar{Q}$ is fed back to input NAND gate in JK FF.

Refer Fig. 8.23(a) (Remove differentiator circuit, then analyse). Draw output Q when J = K = 1 for level triggered JK FF.

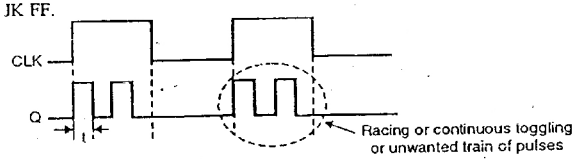


Fig. 8.25

Let's consider initially Q = 0. So when CLK = 1, Q changes state, Q = 1. Now new state of Q propagate through NAND1, NAND2 and will get settled and satisfy setup/hold up time for NAND2/3. The total time of propagation is represented as t in Fig. 8.25. After time t, as CLK = J = K = 1, Q again toggles. ∴ Q = 0. This chain will continue till CLK = 0.

We are going to get ***unwanted train of pulses*** when CLK = J = K = 1. This is drawback of level triggered JK FF.

But if user wants level triggered JK FF without racing, then the answer is MASTER SLAVE JK FF.

8.7 Master Slave JK FF (M/S JK FF) :

➥ [ASKED IN EXAM - DEC. 96, MAY 98, III]

Fig. 8.26 shows M/S JK FF which is combination of two clocked latches; first is called as ***Master*** and second is called as ***Slave***. Master is positively clocked and slave is negatively clocked i.e. :

(1) When CLK = 1 Master is active, slave is inactive

(2) When CLK = 0 Master is inactive, slave is active

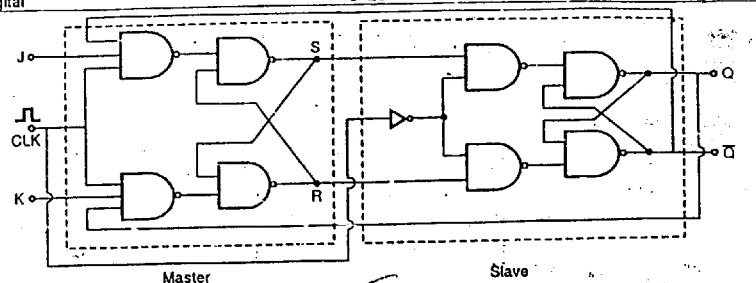


Fig. 8.26 : Master slave JK FF

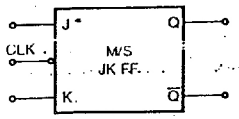


Fig. 8.27 : Symbol

Inputs			Outputs	
CLK	J	K	Q_{n+1}	$\bar{Q}_{n+1}$
X	0	0	Q_n (NC)	$\bar{Q}_n$ (NC)
⎍⎍	0	1	0	1
⎍⎍	1	0	1	0
⎍⎍	1	1	Toggle ($\bar{Q}_n$)	Toggle (Q_n)

Fig. 8.28 : Truth Table

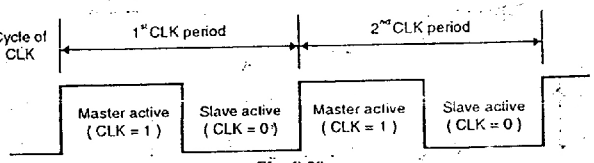


Fig. 8.29

You should remember mainly above two points and one more point that *whatever master does, slave copies it, but in next half cycle of CLK.*

Working

Case I : J = K = 0 Previous state $Q_n = 0$, $\bar{Q}_n = 1$

(i) CLK = 1, ∴ Master active, Slave inactive

∴ output of Master FF, $S = Q_n$, $R = \bar{Q}_n$

(ii) CLK = 0, Master inactive, Slave active

∴ output of Slave FF, $Q = Q_n$, $\bar{Q} = \bar{Q}_n$

Conclusion : No change condition.

Case II : $J = 0, K = 1, Q_n = 1, \bar{Q}_n = 0$

(i) CLK = 1, Master active, Slave inactive

∴ output of Master, S = 0 R = 1.

(ii) CLK = 0, Master inactive, Slave active

∴ output of Slave, Q = 0, $\bar{Q} = 1 \rightarrow Q_{n+1}$ changed

(iii) CLK = 1, Master active, Slave inactive

∴ output of Master, S = 0 R = 1.

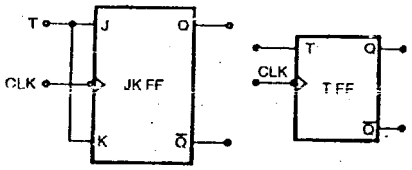
∴ Now the state is stable as (i) and (iii) are same.

Available J – K and M/S JK FF

(1)	Edge triggered JK FF	– 7470
(2)	JK Master Slave FF	– 7472
(3)	Dual JK M/S FF	– 7473
(4)	Master Slave JK FF	– 7476
(5)	JK M/S FF	– 74104/105
(6)	Dual JK positive edge triggered FF	– 74109

8.8 T FF (Toggle FF) :

The type T FF changes state (toggles) for each CLK pulse, if $T_{input} = 1$. In J – K FF, if J = K we get type T FF. It has only one input, referred to as T input.



(a) JK converted to T (b) Logic symbol

Fig. 8.30

CLK	T	Q_{n+1}	$\bar{Q}_{n+1}$
↓	0	Q_n	$\bar{Q}_n$
↓	1	$\bar{Q}_n$	Q_n
↑	X	Q_n	$\bar{Q}_n$
1	X	Q_n	$\bar{Q}_n$
0	X	Q_n	$\bar{Q}_n$

(c) Truth Table

State table

Previous state Q(n)	Next State Q(n + 1)	
	T = 0	T = 1
0	0	1
1	1	0

Table 8.12 : State table of TFF

State diagram

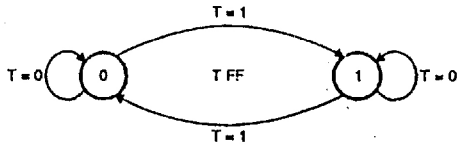


Fig. 8.31 : State diagram of TFF

Present state	to	Next state	If	T
0		0		0
0		1		1
1		0		1
1		1		0

Table 8.13 : Excitation table of TFF

State Equation

$$Q(n+1) = \bar{T} \cdot Q(n) + \overline{Q(n)} \cdot T$$

Ex. 5 : Draw output waveform for Fig. Ex. 8.5 :

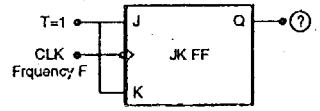


Fig. Ex. 8.5

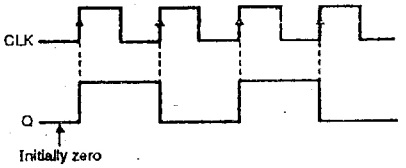


Fig. Ex. 8.5(a)

Soln. : Refer Fig. Ex. 8.5(a).

8.9 Conversion from one Type of FF to Another :

If we want to convert one type of FF to another type of FF, there is systematic approach implementing the same using K-Map. Fig. 8.32 shows generalized model for conversion from one type of FF to another.

Note : Refer Fig. 11.9.

(1) In cycle 0, everytning gets cleared.

(2) In cycle 1, $x_1 = x_2 = 1$. Therefore $y^+ = 1$ and $z = 0$. Status of y^+ will not immediately appear at y. It will appear at next positive CLK edge.

(3) In cycle 2, status of y^+ appears at output y of D FF. Now $x_1 = x_2 = 0$, and $y = 1$. Therefore $y^+ = 0$ and $z = 1$.

(4) In cycle 3, status of $y^+ = 0$, will appear at output y of D FF. Now $x_1 = 0$ and $x_2 = 1$, with $y = 0$ we get, $y^+ = 0$ and $z = 1$.

(5) Finally in cycle 4, $y^+ = 0$., $y = 0$. $x_1 = 1$ and $x_2 = 0$, $\therefore y^+ = 0$ and $z = 1$.

This describes full working of circuit.

Observation :

(1) Output of D FF changes only when CLK arrives. It won't change, eventhough input y^+ changes.

(2) Output of combinational circuit i.e. z changes, as soon as either x_1 or x_2 or y changes. In general, a primary output signal z_k is function of x_j, so the change in x_j immediately affects z_k.

To understand point 2 of observation, refer Fig. 11.9(b).

Refer Fig. 11.9(b). As shown in Fig. 11.9(b), a narrow 1 pulse in x_1, which is an externally generated glitch in the middle of cycle 3, produces a corresponding glitch in form of '0' pulse in z. Because at that instant,

$x_1 = 1, x_2 = 1, y = 0 \therefore x_1 + x_2 + y = z = 1 + 1 + 0 = 0$ with carry.

$\therefore$ We get $z = 0$. Glitch remains there for very short span. Therefore this 1 pulse (glitch) does not affect y, because it has disappeared by the time the flip flop is triggered at the start of cycle 4. Therefore to produce a single well defined output value or state change in any CLK period, each data input should be held constant for most of the clock period. Therefore for proper operation of the circuit, all input signals x_j should be well defined '0' or '1', pulses. Normally input should be maintained for full one clock period. More detail regarding the same will be given to you afterwards.

Ex. 11.2 : For given sequential circuit, draw state diagram and state table. Refer Fig. Ex. 11.2(a).

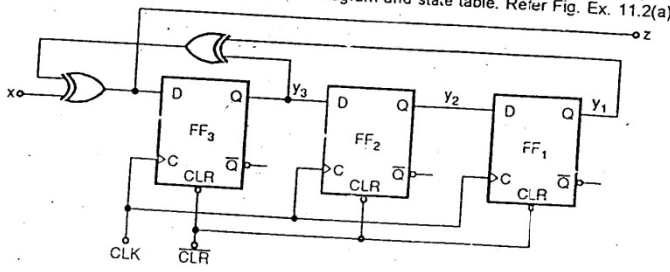


Fig. Ex. 11.2(a) : Sequential circuit

Sol. : (1) First we will write boolean expressions for the same.

$$z = x \oplus y_1 \oplus y_3$$

$$y_1^+ = y_2$$

$$y_2^+ = y_3$$

$$y_3^+ = x \oplus y_1 \oplus y_3 = z$$

(2) First we will write state transition table.

In our problem,

z = Output

$y_1 y_2 y_3 \longrightarrow$ Provides previous state

$y_1^+ y_2^+ y_3^+ \longrightarrow$ Provides next state

x = Input

$\therefore$ We can write, $y_1 y_2 y_3 \xrightarrow{x/z} y_1^+ y_2^+ y_3^+$

$\therefore$ We get table.

Present state			Next state ($y_1^+ y_2^+ y_3^+$)						Ouput (z)	
			x = 0			x = 1				
y_1	y_2	y_3							x = 0	x = 1
0	0	0	0	0	0	0	0	1	0	1
0	0	1	0	1	1	0	1	0	1	0
0	1	0	1	0	0	1	0	1	0	1
0	1	1	1	1	1	1	1	0	1	0
1	0	0	0	0	1	0	0	0	1	0
1	0	1	0	1	0	0	1	1	0	1
1	1	0	1	0	1	1	0	0	1	0
1	1	1	1	1	0	1	1	1	0	1

(3) Next step we follow is simply rearrange transition table to get state table. Only the important point here is we define labels to the state.

$\therefore$ State

$000 \rightarrow S_0$ $\quad$ $100 \rightarrow S_4$

$001 \rightarrow S_1$ $\quad$ $101 \rightarrow S_5$

$010 \rightarrow S_2$ $\quad$ $110 \rightarrow S_6$

$011 \rightarrow S_3$ $\quad$ $111 \rightarrow S_7$

State table

Present state	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
S_0	S_0	S_1	0	1
S_1	S_3	S_2	1	0
S_2	S_4	S_5	0	1
S_3	S_7	S_6	1	0
S_4	S_1	S_0	1	0
S_5	S_2	S_3	1	1
S_6	S_5	S_4	1	0
S_7	S_6	S_7	0	0

(4) The last step is to draw state diagram from state table.

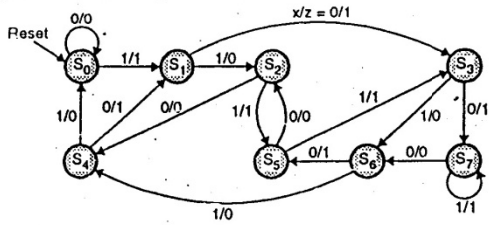


Fig. Ex. 11.2(b) : State diagram

ote : Till this point if you have observed carefully, we have used DFF. DFF is simple flip-flop, because, whatever is input, that will be the output. But for TFF, JK, FF, SRFF, this is not the case. We have to take into account input to the FF, and then we have to decide next state. Let's start with simple TFF.

x. 3 : Consider TFF circuit as shown in Fig. Ex. 11.3(a). Find out state table and diagram for the same.

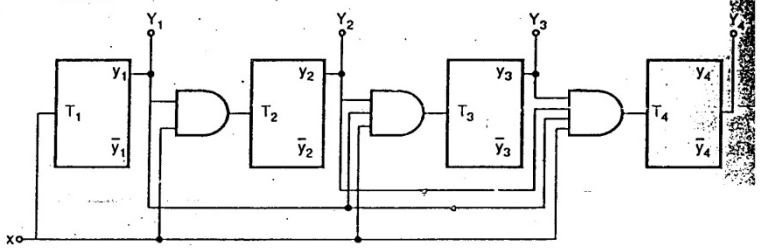


Fig. Ex. 11.3(a) : T FF sequential circuit

ln. : (1) First step is, we will write expressions for T_1, T_2, T_3 and T_4

$$T_1 = x$$

$$T_2 = xy_1$$

$$T_3 = xy_1y_2$$

$$T_4 = xy_1y_2y_3$$

(2) Output states are Y_4, Y_3, Y_2 and Y_1. Input line is x.

∴ We will first write excitation table.

Present state				Flip-flop inputs								Next state ($Y_i = T_i \oplus y_i$) i = 1, 2, 3, 4							
				x = 0				x = 1				x = 0				x = 1			
y_4	y_3	y_2	y_1	T_4	T_3	T_2	T_1	T_4	T_3	T_2	T_1	Y_4	Y_3	Y_2	Y_1	Y_4	Y_3	Y_2	Y_1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	0	0	1	1	1	0	0	1	1	0	1	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1
0	1	0	1	0	0	0	0	0	0	1	1	0	1	0	1	0	1	1	0
0	1	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	1
0	1	1	1	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0	0	0	1	1	1	0	1	0	1	0	1	0
1	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	1	1
1	0	1	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0
1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	0	1
1	1	0	1	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0
1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0

∴ It is module 16 counter. Secondly, if input x = 0, counter latches count and if x = 1, counter is free running.

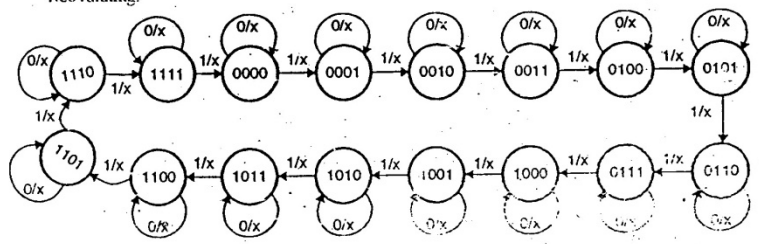


Fig. Ex. 11.3(b) : State diagram

Ex. 4 : For given circuit as shown in Fig. Ex. 11.4(a) find out state diagram and state table.

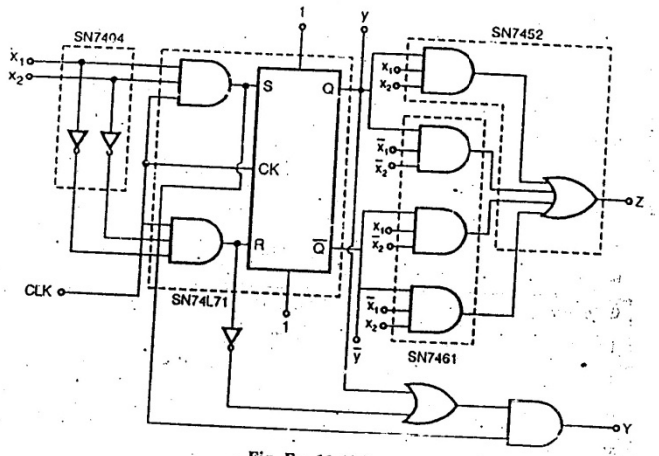


Fig. Ex. 11.4(a)

Soln. : Observing the circuit you will find that it is nothing but serial adder only. But the circuit uses SR instead of D FF as in Ex. 1. Now let's analyse the same, We should get same state table and state diagram as in Ex. 1.

Step 1 : Firstly we have to find excitation functions in terms of circuit inputs and flip-flop outputs from given circuit diagram.

$S = x_1 x_2 \qquad R = \bar{x}_1 \bar{x}_2$

Step 2 : Now let's obtain binary transition table, from excitation table. Binary transition table consists of Y and Z. Where $Y = S + \bar{R}y$ and $Z = x_1 x_2 y + \bar{x}_1 \bar{x}_2 y + x_1 \bar{x}_2 \bar{y} + \bar{x}_1 x_2 \bar{y}$

Now we will combine all the states, as well as output and draw final state table.

Present state (y)	Flip-flop inputs (S/R) $S = x_1 x_2$; $R = \bar{x}_1 \bar{x}_2$				Next state $Y = S + \bar{R} Y$				Output z $Z = x_1 x_2 y + x_1 \bar{x}_2 \bar{y} + \bar{x}_1 x_2 \bar{y}$			
	$x_1 x_2$ 0 0	$x_1 x_2$ 0 1	$x_1 x_2$ 1 1	$x_1 x_2$ 1 0	$x_1 x_2$ 0 0	$x_1 x_2$ 0 1	$x_1 x_2$ 1 1	$x_1 x_2$ 1 0	$x_1 x_2$ 0 0	$x_1 x_2$ 0 1	$x_1 x_2$ 1 1	$x_1 x_2$ 1 0
0	0/1	0/0	1/0	0/0	0	0	1	0	0	1	0	1
1	0/1	0/0	1/0	0/0	0	1	1	1	1	0	1	0

To check how we got output Y and Z, we will take one example.

$x_1 = 1$; $x_2 = 1$ and $y = 0$

$\therefore \quad Y = S + \bar{R} y$

$= x_1 x_2 + \overline{\bar{x}_1 \bar{x}_2}\, y = 1 + \bar{0} \cdot 0 = 1$

$Z = x_1 x_2 y + \bar{x}_1 \bar{x}_2 y + x_1 \bar{x}_2 \bar{y} + \bar{x}_1 x_2 \bar{y}$

$= 1 \cdot 1 \cdot 0 + \bar{1} \cdot \bar{1} \cdot 0 + 1 \cdot \bar{1} \cdot \bar{0} + \bar{1} \cdot 1 \cdot \bar{0}$

$= 0$

If we take state '0' as q_0 and state 1 as q_1 we get transition table as,

Present state (y)	Flip-flop inputs (S/R) $S = x_1 x_2$; $R = \bar{x}_1 \bar{x}_2$				Next state $Y = S + \bar{R} Y$				Output Z $Z = x_1 x_2 y + \bar{x}_1 \bar{x}_2 y + x_1 \bar{x}_2 \bar{y} + \bar{x}_1 x_2 \bar{y}$			
	$x_1 x_2$ 0 0	$x_1 x_2$ 0 1	$x_1 x_2$ 1 1	$x_1 x_2$ 1 0	$x_1 x_2$ 0 0	$x_1 x_2$ 0 1	$x_1 x_2$ 1 1	$x_1 x_2$ 1 0	$x_1 x_2$ 0 0	$x_1 x_2$ 0 1	$x_1 x_2$ 1 1	$x_1 x_2$ 1 0
0	0/1	0/0	1/0	0/0	q_0	q_0	q_1	q_0	0	1	0	1
1	0/1	0/0	1/0	0/0	q_0	q_1	q_1	q_1	1	0	1	0

State diagram :

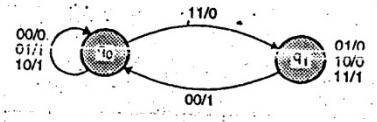


Fig. Ex. 11.4(b)

Note : In next example drawing pattern of state table is slightly changed, shown as 2D array.

Ex. 5 : Consider JK FF circuit shown in Fig. Ex. 11.5(a). Find out state table and state diagram for the same.

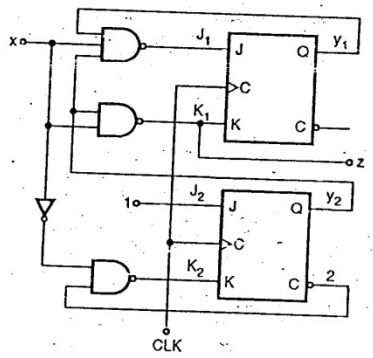


Fig. Ex. 11.5(a) : Sequential circuit containing JK FF

Soln. : (1) We will write expressions,

$$Z = \bar{x} + \bar{y}_2$$

$$J_1 = \bar{x} + \bar{y}_1 + \bar{y}_2$$

$$K_1 = \bar{x} + \bar{y}_2$$

$$J_2 = 1$$

$$K_2 = x + y_2$$

(2) Characteristic equation for JK is,

$$Q(n+1) = J(n)\,\overline{Q(n)} + \overline{K(n)}\,Q(n)$$

$$\therefore \quad y_1^+ = J_1 \bar{Q}_1 + \bar{K}_1 Q_1 = (\bar{x} + \bar{y}_1 + \bar{y}_2)\,\bar{y}_1 + (\overline{\bar{x} + \bar{y}_2})\,y_1$$

Simplify, $\boxed{y_1^+ = \bar{y}_1 + xy_2}$

$$y_2^+ = J_2 \bar{Q}_2 + \bar{K}_2 Q_2 = 1.\bar{y}_2 + (\overline{x + y_2})\,y_2$$

$$= \bar{y}_2 + (\overline{x + y_2})\,y_2$$

Simplifying $\boxed{y_2^+ = y_2}$

Transition table will be as shown :

$y_1 y_2$ \ Input x	0	1
0 0	1 1, 1	1 0, 0
0 0	1 0, 1	(1) (1), (1)
0 0	0 1, 1	0 i, 1
0 0	0 0, 1	1 0, 0

Present state / Next state (y_1^+, y_2^+, z)

(3) Let's say,

$0\,0 \rightarrow A$

$0\,1 \rightarrow B$

$1\,0 \rightarrow C$

$1\,1 \rightarrow D$

∴ State table will be,

Present state ($y_1\,y_2$)	Present input x = 0	Present input x = 1
A	D, 1	D, 1
B	C, 1	C, 0
C	B, 1	B, 1
D	A, 1	C, 0

(4) State diagram :

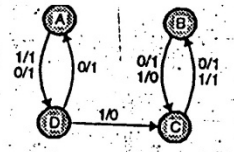


Fig. Ex. 11.5(b)

Ex. 6 : For given sequential circuit draw state table and state diagram. Refer Fig. Ex. 11.6(a).

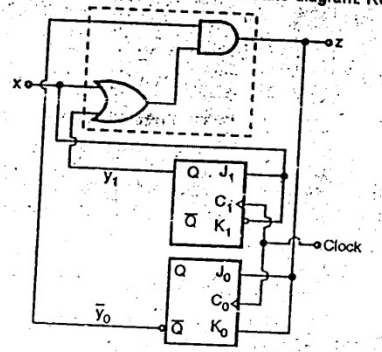


Fig. Ex. 11.6(a)

Soln. : (1) First step is to write equations :

$$J_1 = x \;;\; K_1 = \bar{x}$$

$$J_0 = z = (x + y_1)\,\bar{y}_0$$

$$K_0 = \overline{x + y_1\,\bar{y}_0}$$

From characteristic equation of JK we can write down,

$$y_1^+ = J_1 \bar{Q}_1 + \bar{K}_1 Q_1$$

$$= x\,\bar{y}_1 + \bar{x}\,y_1 = x\,\bar{y}_1 + x\,y_1$$

$$= x$$

$$y_0^+ = J_0 \bar{Q}_0 + \bar{K}_0 Q_0$$

$$= (x + y_1)\,\bar{y}_0 \cdot \bar{y}_0 + (x + y_1\,\bar{y}_0)\,y_0$$

$$= (x + y_1)\,\bar{y}_0$$