Digital Electronics

Book · January 2005

CITATIONS
0

READS
589,440

1 author:

D.K. Kaushik
Shobhit University, Gangoh (Saharanpur) India
45 PUBLICATIONS **52** CITATIONS

SEE PROFILE

DIGITAL ELECTRONICS

D.K. KAUSHIK

DHANPAT RAI PUBLISHING COMPANY

PREFACE

The book ***Digital Electronics*** contains twelve chapters with comprehensive material, discussed in a very systematic, elaborative and lucid manner. The stress is given on the design of digital circuits. It will prove to be good text book for B.E./B.Tech. students of all the engineering colleges in India. It will also cater to the needs of the students of B.Sc. (Electronics), B.Sc. (Computer Science), M.Sc. (IT) and MCA.

The book has been systematically organized and present form help the students to understand the fundamentals of digital electronics.

I am deeply indebted to Prof. P. J. George, Chairman, Depatment of Electronic Science, Kurukshetra University, Kururkshetra for giving me inspiration and enormous encouragement in completion of this book.

The author wishes to thank to Prof. Sandeep Arya, Chairman, Department of Electronics, G. J. University, Hisar, for the healthy discussions on the subject.

The author gratefully acknowledges the motivation from all colleagues and friends with special reference to Shri. Rajesh Kad, lecturer in electronics, Dayanand College, Hisar.

I am grateful to Prof. Subhash Sharma, Principal of the college, for his constant encouragement, guidance and blessings.

I also express my deep gratitude to my wife Pratibha Kaushik and son Amit Kaushik, for their patience, understanding and cooperation during the preparation of the manuscript.

Finally, the author wishes to thank Mr. K.K.Kapoor, Mr. Tarun Kapoor and Mr. Sumit Kapoor, Publishers, Dhanpat Rai Publishing Company, New Delhi for their keen interest in bringing out the first edition of this book.

Any constructive comments, suggestions and criticism from the faculty members and the students for further improvement of the subsequent edition will be highly appreciated and thankfully acknowledged.

HISAR D. K. KAUSHIK

SALIENT FEATURES:

- The material contained in the book is as per class room lectures. The material is neither too large nor too short.
- Written in the simple language but strong pedagogical approach.
- A large number of simple as well complicated solved problems have been introduced. Some unsolved problems with their answers have also been introduced at the end of each chapter.
- The contents are symmetrically arranged.
- It will prove to be good text book for all those who study digital Electronics. It will help the students preparing for NET/SET competitive examination.

Chapter1 Number System

Chapter 2 Binary Codes

Chapter 3 Boolean Algebra and Logic Gates

Chapter 4 Simplification of Boolean Functions

Chapter 5 Combinational Switching Circuits

Chapter 6 More Combinational Circuits

Chapter 7 Logic Families

Chapter 8 Flip-flops

Chapter 9 Shift Registers

Chapter 10 Counters

Chapter 11 Digital to Analog and Analog to Digital Converters

Chapter 12 Digital Memories

1

Number System

1.1 Number System: Every one is familiar with one number system known as decimal number system. The 'deci' means ten so this system has 10 distinct digits or symbols:

0 1 2 3 4 5 6 7 8 9

The decimal numbers falls in the category of positional number system, since the position of a digit indicates the significance to be attached to that digit. For example consider a number 7639. This number has ***7*** thousands, ***6*** hundreds, ***3*** tens and *9* units, which may be written as:

$$\mathbf{7639 = 7 \times 1000 + 6 \times 100 + 3 \times 10 + 9 \times 1}$$
$$\mathbf{= 7 \times 10^3 + 6 \times 10^2 + 3 \times 10^1 + 9 \times 10^0}$$

If a fractional decimal number is considered say 5367.42, then it may be written in the positional form as:

$$\mathbf{5367.42 = 5 \times 10^3 + 3 \times 10^2 + 6 \times 10^1 + 7 \times 10^0 + 4 \times 10^{-1} + 2 \times 10^{-2}}$$

In general any number in decimal number system can be written as:

$$N = a_n x 10^n .... + a_3 x 10^3 + a_2 x 10^2 + a_1 x 10^1 + a_0 x 10^0 + a_{-1} x 10^{-1} + a_{-2} x 10^{-2} + ... a_{-m} x 10^{-m}$$

where the coefficients a_n to a_{-m} are the elements or digits in the decimal numbers. Further these weighted coefficients are multiplied by the some power raised to 10. The power raised to 10 depends on the position of coefficients. In other words one may say in decimal system coefficients a_n to a_{-m} may be any number between 0 to 9 {i.e. between 0 to (10 – 1)} and the positional power is raised to 10, which is known as the radix or the base of this decimal number system.

On the basis of decimal number system discussed above one may define very easily some more number systems. The general form of any number system may be given as:

$$N = a_n x(r)^n .... + a_3 x(r)^3 + a_2 x(r)^2 + a_1 x(r)^1 + a_0 x(r)^0 + a_{-1} x(r)^{-1} + a_{-2} x(r)^{-2} + ... a_{-m} x(r)^{-m}$$

where *r* is called as radix or base of the number system. The weighted coefficients a_n to a_{-m}, may be any number (or digit) between 0 to (*r – 1*). The coefficient a_{-m} is called as the least significant digit (LSD) and a_n is known as the most significant digit (MSD).

1.2 Binary Number System: On the analogy of decimal number system one may define another number system whose radix or base is two (*r = 2*) and its elements or digits will be *0* & *1* only. This system is known as binary number system as its radix is two (binary means *2*). The digits *0* & *1* of this system are known as bits. This number system finds extensive use in digital electronics. The table 1.1 illustrates the counting in binary system with their decimal equivalents.

Table 1.1

Binary Numbers	Decimal equivalent	Binary Numbers	Decimal equivalent
0	0	1101	13
1	1	1110	14
10	2	1111	15
11	3	10000	16
100	4	10001	17
101	5	10010	18
110	6	10011	19
111	7	10100	20
1000	8	10101	21
1001	9	10110	22
1010	10	10111	23
1011	11	11000	24
1100	12	and so on	

The binary numbers are pronounced in the following manner:

0 is pronounced as zero
1 is pronounced as one
10 is pronounced as one zero not ten
11 is pronounced as one one not eleven
and so on.

The decimal equivalent of a binary number (say 10110) is 22, which can be verified as follows applying the same pattern as discussed in decimal number system.

$$(10110)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$
$$= 16 + 0 + 4 + 2 + 0$$
$$= (22)_{10}$$

It is very essential to show the suffix to the numbers which indicates the base of the number system.

Example 1.1: Find the decimal equivalent of the binary number 11011001.0101.
Solution:
$(11011001.0101)_2 = 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$
$= 128 + 64 + 0 + 16 + 8 + 0 + 0 + 1 + 0 + 0.25 + 0 + 0.125$
$= (217.375)_{10}$

1.3 Octal Number System

: The radix or base of the octal number system is 8 (octal means 8) and its digits will be 0 to 7 i.e. 0, 1, 2, 3, 4, 5, 6, 7. The table 1.2 illustrates the counting in octal system with their decimal equivalents.

Table 1.2

Octal Numbers	Decimal equivalent	Octal Numbers	Decimal equivalent
0	0	15	13
1	1	16	14
2	2	17	15
3	3	20	16
4	4	21	17
5	5	22	18
6	6	23	19
7	7	24	20
10	8	25	21
11	9	26	22
12	10	27	23
13	11	28	24
14	12	so on	

The decimal equivalent of octal number 24 is:

$$(24)_8 = 2 \times 8^1 + 4 \times 8^0$$
$$= 16 + 4$$
$$= (20)_{10}$$

Example 1.2: Find the decimal equivalent of the octal number 7126.45.
Solution:
$(7126.45)_8 = 7 \times 8^3 + 1 \times 8^2 + 2 \times 8^1 + 6 \times 8^0 + 4 \times 8^{-1} + 5 \times 8^{-2}$
$= 512 + 64 + 16 + 6 + 0.125 + 0.5 + 0.078125$
$= (598.703125)_{10}$

1.4 Hexadecimal Number System : In hexadecimal number system the radix or base is 16 and its digits will be 16 distinct elements which are given as: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.** The table 1.3 illustrates the counting in Hexadecimal number system with their decimal equivalents.

Table 1.3

Hexadecimal Numbers	Decimal equivalent	Hexadecimal Numbers	Decimal equivalent
0	0	D	13
1	1	E	14
2	2	F	15
3	3	10	16
4	4	11	17
5	5	12	18
6	6	13	19
7	7	14	20
8	8	15	21
9	9	16	22
A	10	17	23
B	11	18	24
C	12	and so on	

It can be verified that the decimal equivalent of the hexadecimal number $(17)_{16}$ is $(23)_{10}$.

$$\mathbf{(17)_{16} = 1\ x\ (16)^1 + 7\ x\ (16)^0}$$
$$\mathbf{=16 + 7}$$
$$\mathbf{= (23)_{10}}$$

Example 1.3: Find the decimal equivalent of the Hexadecimal number 3BC7.46
Solution:
$(3BC7.46)_{16}$
$= 3x\ (16)^3 + 11\ x\ (16)^2 + 12\ x\ (16)^1 + 7\ x\ (16)^0 + 4\ x\ (16)^{-1} + 6\ x\ (16)^{-2}$
$= 12288 + 2816 + 192 + 7 + 0.25 + 0.234375$
$= (15303.484375)_{10}$

1.5 Conversion of Integer Decimal Number to Binary Number: It is necessary to know the techniques with which the conversion of integer decimal number is possible directly to binary number. Consider an integer decimal number ***d*** which can be represented as:

$$d = a_n.2^n + a_{n-1}.2^{n-1} + ... + a_1.2^1 + a_0.2^0$$

If we divide *d* by a factor of *2* (radix of the binary number system), we obtain the quotient *q* as:

$$q = \frac{d}{2} = a_n.2^{n-1} + a_{n-1}.2^{n-2} + ... + a_1.2^0$$

and the coefficient a_0 becomes the remainder. Thus the least significant bit a_0 is determined. Again on dividing the quotient q by 2, the second least significant bit a_1 is obtained. If this procedure of division is continued till the quotient becomes zero, all the coefficients a_n to a_0 will be obtained.

In general one can convert the integer decimal numbers to their equivalent numbers in other number system by dividing the decimal number by the radix of the required number system. The remainders will give the required result.

Example 1.4: Convert the following decimal numbers into binary.

(i) 35 (ii) 127

Solution: (i)

2	35	
2	17	1
2	8	1
2	4	0
2	2	0
2	1	0
	0	1

So $(35)_{10} = (10011)_2$

(ii)

2	127	
2	63	1
2	31	1
2	15	1
2	7	1
2	3	1
2	1	1
	0	1

So $(127)_{10} = (1111111)_2$

Example 1.5: Convert the following decimal numbers into octal.
(i) 567 (ii) 1276

Solution: (i)

8	567	
8	70	7
8	8	6
8	1	0
	0	1

So $(567)_{10} = (1067)_8$

(ii)

8	1276	
8	159	4
8	19	7
8	2	3
	0	2

So $(1276)_{10} = (2374)_8$

Example 1.6: Convert the following decimal numbers into hexadecimal.
(i) 8537 (ii) 98765

Solution: (i)

16	8537	
16	533	9
16	33	5
16	2	1
	0	2

So $(8537)_{10} = (2159)_{16}$

(ii)

16	98765	
16	6172	D
16	385	C
16	24	1
16	1	8
	0	1

So $(98765)_{10} = (181CD)_{16}$

1.6 Conversion of Fractional Decimal Number to Binary Number:

Consider a fractional decimal number f represented in its equivalent binary form given by:

$$f = a_{-1}.2^{-1} + a_{-2}.2^{-2} + ... + a_{-n}.2^{-n}$$

In order to find the coefficients a_{-1}, a_{-2} .... a_{-n} the fraction number f is multiplied by a factor of 2 (radix of the binary number) as:

$$2xf = a_{-1} + a_{-2}.2^{-1} + ... + a_{-n}.2^{-n+1}$$

MSD (0 or 1) — fractional part say f_1

In this way the coefficient a_{-1} is obtained which is an integer 0 or 1. The fractional part f1 of the product is further multiplied by the factor 2 to have the coefficient a_{-2}. The procedure of multiplication is continued till the fractional part of the product becomes zero. Sometimes the fractional part does not become zero, in that case the multiplication process is stopped after getting the four five coefficients or till the recurring occurs.

A similar procedure may be used to convert the decimal fraction into its equivalent other number system by successive multiplication by the radix of the number system into which the number is required.

Example 1.7: Convert the following decimal numbers into binary.

(i) 0.625 (ii) 0.6

Solution: (i)

Decimal	Product	Integer part
.625 *x* 2	1.25	1
.250 *x* 2	0.5	0
.500 *x* 2	1.0	1
0		
Stop		

$$(0.625)_{10} = (0.101)_2$$

(ii)

Decimal	Product	Integer part
0.600 *x* 2	1.200	1
0.200 *x* 2	0.400	0
0.400 *x* 2	0.800	0
0.800 *x* 2	1.600	1

In this example non - terminating binary fraction is obtained as 0.6 recur beyond this point.

So $(0.6)_{10} = (0.1001(1001) \ldots)_2$

Example 1.8: Do the following conversions:

(i) $(965.125)_{10}$ to octal

(ii) $(8765.025)_{10}$ to hexadecimal

(iii) $(6754.05)_8$ to decimal.

Solution: (i) Integer part

8	965	
8	120	5
8	15	0
8	1	7
	0	1

Fractional part

Decimal	Product	Integer part
0.125 *x* 8	1. 00	1

So $(965.125)_{10} = (1705.1)_8$

(ii) Integer part

16	8765	
16	547	D
16	34	3
16	2	2
	0	2

Fractional part

Decimal	Product	Integer part
0.025 *x* 16	0. 4	0
0.4 x 16	6.4	6
0.4 x 16	6.4	6 ↓
		repeated value

So $(8765.025)_{10} = (223D.0666\ldots)_{16}$

(iii) $(6754.05)_8$ to decimal

$$6754.05 = 6x8^3 + 7x8^2 + 5x8^1 + 4x8^0 + 0x8^{-1} + 5x8^{-2}$$
$$= 3072 + 448 + 40 + 4 + 0 + .071285$$
$$= 3564.078125$$

So $(6754.05)_8 = (3564.078125)_{10}$

1.7 Conversion of Octal to Binary and Vice – Versa : The eight symbols of octal numbers 0, 1, 2, ….7 can be represented in to three bit binary numbers as $2^3 = 8$. So starting with the least significant bit of the binary number, the successive three bits are arranged together in the form of groups. These groups of three bits are replaced by their octal equivalents as shown in table 1.4.

Table 1.4

Octal Numbers	Binary Numbers
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	110

The binary numbers are converted to the octal numbers by making the groups of three bits from right to left in the integer part of the binary number and from left to right on the binary fractional part.. If the need arises for making the groups of three bits one or two zeros may be added to the left of most significant bit; and / or to the right of the least significant bit of the fractional part of the binary number. The octal equivalent of the groups may be written using the table 1.4. Similarly to convert the octal number to binary number, the binary equivalent of each octal number is written using the table 1.4.

Example 1.9: Do the following conversions:

(i) $(1100110111110.1011)_2$ to octal

(ii) $(2467.534)_8$ to binary

Solution: (i) Binary number : 001 100 110 111 110 . 101 100
Octal Equivalent: 1 4 6 7 6 3 4

So $(1100110111110.1011)_2 = (14676.34)_8$

(ii) Octal number : 2 4 6 7 . 5 3 4
Binary Equivalent: 010 100 110 111 . 101 011 100

So $(2467.534)_8 = (10100110111.1010111)_2$

1.8 Conversion of Hexadecimal to Binary and Vice – Versa: As is well known that the hexadecimal system has a base 16 ($2^4 = 16$) so every hexadecimal digit can be represented as a group of 4 bits as shown in table 1.5. For conversion of octal to binary and vice versa one can proceed in the similar fashion as in the case of octal to binary and vice versa.

Table 1.5

Hexadecimal umbers	Binary Numbers
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Example 1.10: Do the following conversions:
(i) $(110011010110111110.101)_2$ to hexadecimal
(ii) $(2AB6E7.5D4)_{16}$ to binary

Solution:
(i) Binary number : 0011 0011 0101 1011 1110 . 1010
Hexadecimal Equivalent: 3 3 5 B E A

So $(110011010110111110.101)_2 = (335BE.A)_{16}$

(ii) Hexadecimal: 2 A B 6 E 7 . 5 D 4
Binary 0010 1010 1011 0110 1110 0111 . 0101 1101 0100

So $(2AB6E7.5D4)_{16} = (10101010110110111001 11.0101110101)_2$

Example 1.11 Convert the hexadecimal number 4AC7.4B in to its equivalent octal number.

Solution: The given hexadecimal number is first converted to binary number and then converted to octal number.

Hexadecimal: 4 A C 7 . 4 B
Binary: 0100 1010 1100 0111 . 0100 1011

Now $(100101011000111.01001011)_2$ to octal

100101011000111.01001011 = 100 101 011 000 111 . 010 010 110
Octal : 4 5 3 0 7 2 2 6

Thus $(4AC7.4B)_{16} = (45307.226)_8$

1.9 Binary Addition: The counting of numbers in any system is a form of addition since successive numbers, while counting, are obtained by adding 1. In decimal number system, the successive addition is obtained as follows:

0 + 1 = 1
1 + 1 = 2
2 + 1 = 3
3 + 1 = 4
…
….
8 + 1 = 9
9 + 1 = 1 0 i.e. the sum is zero but have a carry to the next position.

From the above discussion it is clear that when 1 is added to the last digit of a number system, sum becomes zero and has one carry to the next position.

In the similar fashion if this rule is applied to the binary system the binary addition may be illustrated as follows:

0 + 1 = 1
1 + 1 = 1 0 i.e. sum is zero and carry is 1.

Table 1.6 shows the addition of two bits *a* and *b*, having the sum and carry to the next position. There are four possible combinations

Table 1.6

a	*b*	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

This table is known as Half adder table, as it gives the simple addition of two bits *a* and *b*. Table 1.7 known as full adder table shows the addition of maximum of three bits. These bits are the carry bits, if any, from the previous stage of addition, and the augend and addend bits.

Table 1.7

Augend bit	Addend bit	Carry from previous stage	Sum	Carry to next stage
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Example 1.12 : Perform the following binary additions.
(i) 110111 + 11010 (ii) 101101.101 + 101011.011
(iii) 1101101.101 + 110110.01

Solution:

(i) Carry → 1 1 1 1 1 0
1 1 0 1 1 1
0 1 1 0 1 0
1 0 1 0 0 0 1

(ii) Carry → 1 0 1 1 1 1 1 1 1
1 0 1 1 0 1 . 1 0 1
1 0 1 0 1 1 . 0 1 1
1 0 1 1 0 0 1 . 0 0 0

(iii) Carry → 1 1 1 1 1 0 0 0 0 0
1 1 0 1 1 0 1 . 1 0 1
0 1 1 0 1 1 0 . 0 1 0
1 0 1 0 0 0 1 1 . 1 1 1

1.10 Binary Subtraction: Half – subtractor table is used for subtraction similar to one used for addition. It is clear from the table 1.8 that when *1* is subtracted from *0*, a *1* is to be borrowed from the next adjacent higher position.

Table 1.8

a	*b*	difference	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

A full – subtractor table having the minuend, subtrahend and the borrow bit of the previous stage as the inputs and which gives the difference as well as the borrow bit to be taken from the next stage, is shown in table 1.9.

Table 1.9

Minuend	Subtrahend	Borrow bit from previous stage	Difference	Borrow from the next stage
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

Example 1.12 : Perform the following binary operations.
(i) 1101101 – 1100111 (ii) 11011.01 – 10101.11
(iii) 1101.101 – 1001.011

Solution:

(i) Borrow → 0 0 0 0 1 1 0
1 1 0 1 1 0 1
1 1 0 0 1 1 1
0 0 0 0 1 1 0

(ii) Borrow → 0 1 0 1 1 0
1 1 0 1 1 . 0 1
1 0 1 0 1 . 1 1
0 0 1 0 1 . 1 0

(iii) Borrow →

```
  0 0 0 0  1 0
  1 1 0 1 .1 0 1
  1 0 0 1 .0 1 1
  0 1 0 0 .0 1 0
```

1.11 Signed Numbers: The positive numbers were discussed so far in the preceding sections of this chapter. But most of the digital systems handle not only the positive numbers but also the negative numbers. Some means are, therefore, required to represent the sign of the binary numbers. In general, an extra bit is provided at the extreme left of the number. This extra bit is known as the sign bit. The extra bit is isolated from the magnitude of the binary number by a comma. The sign bit is 0 or 1. By convention, a 0 bit is used for the positive numbers and a 1 bit is used for negative numbers.

For example: + 9 is represented by 0, 1001

and – 9 is represented by 1, 1001

Though this method of representing the signed numbers is straight forward, yet it is not normally used in the digital system since the realization of this method by digital circuit is very complex. The most commonly used method for representing the signed binary numbers is 2's complement method. Before discussing signed binary arithmetic operations using the 2's complement method, it is necessary to show the 1's complement and 2's complement representation of binary numbers.

1.11.1 1's Complement Representation: The 1's complement of a binary number is obtained by converting each 0 bit of the binary number to a 1, and each 1 bit by a 0. The 1's complement value represents the negative number of the binary number.

For example: The 1's complement of the binary number 1011101 is obtained as:

```
1  0  1  1  1  0  1
↓  ↓  ↓  ↓  ↓  ↓  ↓
0  1  0  0  0  1  0
```

Thus 1's complement of binary number 1011101 is 0100010.

1.11.2 2's Complement Representation: The 2's complement of a binary number is obtained by taking the 1's complement of the number and adding 1 to the least significant bit position. The process for obtaining the 2's complement of $(25)_{10} = (11001)_2$ is given below:

```
  1 1 0 0 1      binary equivalent of 25
  0 0 1 1 0      1's complement of 25
+         1      add 1 to get 2's complement
  0 0 1 1 1
```

The 2's complement of 11001 is 00111.

The another method of obtaining the 2's complement of a binary number is to scan the number from right to left and complement all bits appearing after the first scan of a '1'.

For example 2's complement of $(42)_{10} = (101010)_2$ is 010110. Since first '1' appears in the second place from the right hand side, so all the bits after occurring first '1' at the second place are complemented. This can be verified by using the first method discussed above.

1.12 Signed Numbers using 2's Complement: Computer systems always process the words (digital) in a uniform fashion having a maximum limit of *N* bits. An *N* – bit machine can handle the unsigned decimal numbers from 0 to $2^N - 1$. Thus a 4 bit machine can handle 0 to 15 decimal numbers (unsigned) represented by binary numbers ranging from 0000 to 1111. Similarly an 8 – bit machine can handle 0 to 255 decimal numbers having binary numbers ranging from 00000000 to 11111111. However, for signed binary numbers, 4 bit machine will have the range from – 8 to +7 and 8 bit machine will have the range from – 128 to +127. Table 1.10 illustrates how the 4 bit machine represents the signed binary numbers.

Table 1.10

Decimal value	Signed binary numbers
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
+1	0001
+2	0010
+3	0011
+4	0100
+5	0101
+6	0110
+7	0111

Similarly, 8 bit machine will have the signed numbers ranging from – 128 (10000000_2) to +127 (01111111_2).

The following inferences are obtained from this table:

1. The largest negative number in an *N* – bit machine is $-2^{(N-1)}$ and the largest positive number is $+(2^{(N-1)} - 1)$.
2. All the positive numbers have most significant bit as 0 and the negative numbers as 1**.**
3. All the negative numbers are the 2's complement of positive numbers. For example 2's complement of 0110 (+6) is 1010 (– 6) and the 2's complement of 01111111_2 (+127) is 10000001 which equal to –127 in 8 bit machine.

If the 2's complement method is used to represent the negative numbers as discussed above then the subtraction of signed numbers can easily be performed only by the addition method. This will lead a simplification in the hardware circuits.

Example 1.13: What is the range of unsigned and signed decimal numbers as well as binary numbers that can be represented in a 10 bit system?

Solution: The range of unsigned decimal numbers will be:

0 to 1023 ($2^{10} - 1$) in the 10 bit system

i.e. 0000000000_2 to 1111111111_2.

The range of signed decimal number will be -512 (-2^9) to $+511$ ($2^9 - 1$)

i.e. 1000000000_2 to 0111111111_2.

Example 1.14: Represent the following decimal numbers as 8 bit signed numbers in the 2's complement form.

(i) +25 (ii) – 68 (iii) –128

Solution: (i) $+25 = 00011001_2$

(ii) -68 = 2's complement of $+68$ (01000100_2)

$= 10111100_2$

(iii) $-128 = 10000000_2$

1.13 Addition/Subtraction of Signed Numbers in 2's Complement Representation:

In the signed numbers the addition and subtraction of binary numbers are the same. The subtraction of two positive numbers means the addition of a negative number to the positive number. The negative number infect is the 2's complement of the positive number. During the addition of two signed numbers, if there is an end around carry, it should be ignored. The result is interpreted using the convention discussed above i.e. if MSB of the result is 0, then the answer is positive and on the contrary if MSB is 1, then the answer is negative (in 2's complement form). This can be illustrated by taking the following examples:

(i) Addition of positive number with smaller negative number: Consider the addition of +15 and – 9. The numbers +15 and –9 are represented in 5 bits signed binary form. These numbers can not be represented in 4 bit signed number as 4 bit machine will have the range from – 8 to +7.

```
+15      01111
– 9      10111      (10111 is the 2's complement of +9)
+ 6   1  00110
```

There is an end around carry which is ignored. So the answer is correct as 00110 represents + 6.

(ii) **Addition of positive number with larger negative number**: Consider the addition of +9 and –15.

+ 9	01001
–15	10001 (10001 is the 2's complement of +15)
– 6	11010

There is no end around carry so the answer is negative which is verified by the MSB of the answer. The answer is correct as 11010 represents – 6.

(iii) Addition of two positive numbers: consider the addition of positive numbers +15 and +9 :

+15	01111
+ 9	01001
+24	11000

The result 11000 is correct in unsigned binary numbers but incorrect in signed binary numbers as 11000 represents – 8 in 5 bit signed binary numbers. The correct answer could be obtained if 6 bit signed binary system was considered.

+15	001111
+ 9	001001
+24	011000

Now the answer is correct.

(iv) Addition of two negative numbers: Consider the addition of –15 and – 9.

–15	10001 (10001 is the 2's complement of +15)
– 9	10111 (10111 is the 2's complement of +9)
–24	1 01000

After ignoring the end around carry the answer 01000 is incorrect, as the maximum limit of 5 bit signed binary numbers is –16 to +15. To get the correct answer each number should have been represented in 6 bits signed binary form as follows:

–15	110001 (110001 is the 2's complement of +15)
– 9	110111 (110111 is the 2's complement of +9)
–24	1 101000

Now the answer is correct as after ignoring the end around carry 101000 represents –24 in signed binary form.

The overflow is said to have occurred in the above two examples as initially insufficient number of bits were used for representing the signed binary numbers. While working with 2's complement addition, one should ensure that the positive and negative number are expressed in 2's complement representation and the sum also lie within the specified range, otherwise wrong result will occur. However, in computers a special circuit is provided to detect any overflow condition and indicate the erroneous result.

Example 1.15: Perform the following operations in 8-bit system using 2's complement method. (i) – 49 – 26 (ii) 67 – 39 (iii) – 87 + 112 .

Solution:

(i)

–49		11001111	(2's complement of +49)
– 26		11100110	(2's complement of +26)
–75	1	10110101	

The end around carry is ignored. So the answer is 10110101 (–75).

(ii)

+67		01000011	
–39		11011001	(2's complement of +39)
+28	1	00011100	

The end around carry is ignored. So the answer is 00011100 (+28).

(iii)

–87		10101001	(2's complement of +87)
+112		01110000	
+25	1	00011001	

The end around carry is ignored. So the answer is 00011001 (+25).

1.14 Nine's and Ten's Complement of Decimal Numbers : In the preceding section, 1's and 2's complement of binary numbers were discussed, to represent the signed numbers. In the similar fashion 9's and 10's complement representation of decimal numbers may be used to represent the negative numbers. The 9's complement of a decimal number is obtained by subtracting each digit from 9. For example, 9's complement of 2457 is (9999 – 2457) = 7542 and 9's complement of 89031 is (99999 – 89031) = 10968. It is analogous to 1's complement of binary numbers. The 1's complement of binary number is obtained by subtracting each bit by 1, the largest or highest bit or digit of the number system (or by interchanging each bit by 0 to 1 and vice-versa).

Similarly, 10's complement of a decimal number is obtained by adding 1 to the 9's complement of that decimal number. For example 10's complement of 3697 is 6303 (9's complement of 3697 + 1). The other method of getting 10's complement of a decimal number of *n* digits is to subtract that number from 10^n.

i.e. 10's complement of 19874 is $(10^5 - 19874 = 100000 - 19874) = 80126$.

The 10's complement can be used for the addition of signed decimal numbers as given in the following example.

Example 1.16 : Add the following signed decimal numbers using 10's complement.
(i) Add (+6230) and (– 2394) (ii) Add (– 5260) and (+2987)

Solution:

(i)
$$\begin{array}{r} 6230 \\ 7606 \\ \hline 1\ 3836 \end{array}$$
(10's complement of 2394)

The end around carry is ignored. So the answer is +3836.

(ii)
$$\begin{array}{r} 4740 \\ 2987 \\ \hline 7727 \end{array}$$
(10's complement of 5260)

There is no end around carry so answer is negative and is in 10's complement form i.e. – 2273 (10's complement of 7727).

1.14.1 *r*'s and (*r* – 1)'s complement: In general one can define two types of complements in a number system of base *r*.

(i) (*r* – 1)'s complement: The (*r* – 1)'s complement in any number system of radix r is obtained by subtracting each digit of the number from (*r*– 1). For example 7's complement of 347 in octal number system is (777 –347) = $(430)_8$ and 5's complement of 23450 in a number system whose radix value is 6, is (55555 – 23450) = $(32105)_6$.

(ii) *r*'s complement or true complement: The *r*'s complement of nonzero number in a number system of radix *r* is obtained by getting the (*r* –1)'s complement of that number and adding 1 to it. If a number is zero its *r*'s complement or true complement is also zero. For example 8's complement of 37401 in octal number system is (77777–37401) + 1 = $(40377)_8$ and 5's complement of 23410 in a number system whose radix value is 5, is (44444 – 23410) + 1 = 21035 = $(21040)_5$.

1.15 Binary Multiplication: The process of multiplication of binary numbers is similar to that of decimal multiplication. The product of two binary numbers whose magnitude is n bits each, can be 2n bits long. Followings are the steps used in the multiplication of two binary numbers:

Step 1: Multiplier is scanned from the right hand side. If LSB is 1 the multiplicand is copied as the first partial product. If LSB is zero, then zeros are entered as the first partial product.

Step 2: Next bit (left to the previous bit) of the multiplier is examined, if it is 1 the multiplicand is copied as the next partial product after shifting left this partial product by one bit. If it is zero then enter zeros as the next partial product after shifting it left by one bit.

Step 3: Repeat step 2 till all bits in the multiplier have been considered.

Step 4: The final product is obtained by adding all the partial products.

This method of multiplication is known as long hand multiplication which is generally done by using paper and pencil. However, in digital machines, the multiplication of two binary numbers is considered in slightly different manner. Instead of providing digital circuits to store all the shifted partial products and finally adding these products, the

partial products corresponding to each bit of the multiplier are simultaneously added into the previous product which is shifted right by one bit rather than shifting to the left.

Example1.17: Multiply 10101 by 10011.

Solution:

Multiplicand	1 0 1 0 1
Multiplier	1 0 0 1 1
Partial products	1 0 1 0 1
	1 0 1 0 1
	0 0 0 0 0
	0 0 0 0 0
	1 0 1 0 1
Product	1 1 0 0 0 1 1 1 1

1.16 Binary Division: The process of division of binary numbers is similar to that of decimal division. The long hand division method is used for this purpose. In decimal division it is seen how many times the divisor goes into the dividend, but in binary division there are only two possibilities o and 1 i.e. if the divisor goes into the dividend, the quotient becomes 1, if it does not the quotient becomes zero. The divisor is then subtracted from dividend. The next bit of the dividend is copied in the remainder of the subtraction and again seen if the divisor goes into the dividend. This process is continued till all the bits of the dividend are considered. However, in digital machines in the division the subtraction is performed using the 2's complement method.

Example 1.17: Divide 1101101 by 101.

Solution:

Quotient

Divisor) *Divident*

$$
\begin{array}{r}
10101.11.. \\
101\overline{)1101101} \\
\underline{101} \\
00111 \\
\underline{00101} \\
0001001 \\
\underline{0000101} \\
00001000 \\
\underline{00000101} \\
000000110 \\
\underline{000000101} \\
000000001
\end{array}
$$

So quotient is $(10101.11)_2$ and remainder $(.01)_2$

1.17 Floating Point Representation of Binary Numbers: It is well known that the very small and very large decimal numbers can be expressed in scientific notation e.g. 2.48×10^{-24} and 6.75×10^{18}. Binary numbers can also be represented in the similar fashion. This form of notation will have a binary number of few bits known as the mantissa and an exponent of 2 (radix of binary numbers). The format of such representation will be different for different computing machines. The 16 bit machine will have 10 bit mantissa and 6 bit exponent and 24 bit machine will have 15 bit mantissa and 9 bit exponent. The format of 16 bit machine is given below.

10 bit Mantissa	6 bit Exponent
0 1 1 0 0 1 1 0 1 0	1 0 1 0 1 0

Fig. 1.

The mantissa is in 2's complement form; the leftmost bit is, therefore, used as sign bit. The binary point will be to the right of the sign bit. The 6 bit exponent can represent 0 to 63 or in the signed number from – 32 to + 31. However, a common system is used to represent exponent part. The exponent part is represented in excess 32 notation i.e. the number $(32)_{10}$ or $(100000)_2$ is added to the desired exponent. The table 1.11 illustrates representation of exponent part in this system.

Table 1.11

Desired Exponent	2's complement representation	Excess 32 notation (in 6 bits)	Binary Representation
– 32	100000	100000+100000 = 000000	000000
– 31	100001	100001+100000 = 000001	000001
– 30	100010	100010+100000 = 000010	000010
– 15	110001	110001+100000 = 010001	010001
0	000000	000000+100000 = 100000	100000
+ 1	000001	000001+100000 = 100001	100001
+ 15	001111	001111+100000 = 101111	101111
+ 30	011110	011110+100000 = 111110	111110
+ 31	011111	011111+100000 = 111111	111111

As discussed above the floating point number given in the above format is:

The mantissa part .110011010
The exponent part 101010
Subtracting 100000 001010
The number is $N = + (.110011010)_2 \times 2^{10}$
$= + (1100110100.00)_2$
$= + (820)_{10}$

Example 1.18: What floating point number do the following numbers represent? (i) 0100101001101011 (ii) 1010010110101111
(iii) 0110111010011101

Solution: (i) 0100101001101011

The mantissa part .100101001
The exponent part 101011
Subtracting 100000 001011
The number is $N = + (.100101001)_2 \times 2^{11}$
$= + (10010100100.0)_2$
$= + (1188)_{10}$

(ii) 1010010110101111

The mantissa part 1.010010110
– .101101010
The exponent part 101111
Subtracting 100000 001111
The number is $N = – (.101101010)_2 \times 2^{15}$
$= – (101101010000000.0)_2$
$= – (23168)_{10}$

(iii) 0110111010011101

The mantissa part .110111010
The exponent part 011101
Subtracting 100000 101011
The number is $N = + (.110111010)_2 \, x \, 2^{-21}$
$= + (.00000000000000000000011011101)_2$

Example 1.19: Express the following decimal numbers into 16 bit floating point number.
(i) $(45365.125)_{10}$ (ii) $-(335.625)_{10}$

Solution:

(i) Binary equivalent of $(45365.125)_{10}$ 1011000100110101.001
Binary format .1011000100110101 $x \, 2^{16}$
Mantissa + .101100010
Exponent 010000
Equivalent exponent 010000 + 100000 = 110000

So the floating point format will be 0101100010110000

(ii) Binary equivalent of $-(335.625)_{10}$ –101001111.101
1010110000.011
Binary format $- .010110000 \times 2^9$
Mantissa – .010110000
Exponent 001001
Equivalent exponent 001001 + 100000 = 101001

So the floating point format will be 1010110000101001

Example 1.20: In a number system of radix *R*, *A* and *B* are the successive digits such that $(AB)_R = (28)_{10}$ and $(BA)_R = (35)_{10}$. Find the radix *R* of the number system and the values of *A* and *B.*

Solution: According to the problem:

$AxR^1 + BxR^0 = 28$ and $BxR^1 + AxR^0 = 35$
or $AxR + B = 28$ $BxR + A = 35$
also $B = A + 1$
so $AxR + A = 27$ and $AxR + R + A = 35$
After solving these equations we get:
$R = 8$, $A = 3$ and $B = 4$

Example 1.21: Determine the radix value in the following cases.

(i) $\sqrt{(51)_R} = (6)_R$

(ii) $(11)_R + (25)_R = (102)_R$

Solution: (i) Decimal equivalent of the problem is given by;

$$\sqrt{5xR+1} = 6$$

Squaring on both side:

$$5R+1 = 36$$

or $$R = 7$$

(ii) Decimal equivalent of the problem is given by;

$$1xR+1+2xR+5 = 1xR^2 + 0xR + 2xR^0$$

or $$R^2 - 3R - 4 = 0$$

Solving for R we have:

$$R = 4 \text{ and } R = -1$$

The radix can not be negative, so the required result is 4.

Problems:

1. Discuss decimal number system. Define radix.
2. Define octal number system. How the counting in octal number system is made?
3. Define hexadecimal number system. Write the counting from 0 to 40 decimal numbers into its equivalent hexadecimal number system.
4. How the decimal integer numbers are converted to binary numbers? Explain.
5. How the decimal fractional numbers are converted to binary numbers? Explain.
6. Define a number system whose radix value is 3. Write the counting of first 30 decimal numbers into the system whose radix value is 3.
7. Define a number system whose radix value is 7. Write the counting of first 30 decimal numbers into the system whose radix value is 7.
8. Discuss how the octal numbers are converted into its equivalent binary numbers and vice – versa.
9. Discuss how the hexadecimal numbers are converted into its equivalent binary numbers and vice – versa.
10. Write numbers from 1 to 30 in the following number systems:
    (i) Binary (ii) Octal (iii) Hexadecimal
    (iv) to a system whose radix value is 6.
11. Discuss how the addition of binary numbers is performed. Draw the half adder and full adder tables.
12. Discuss how the subtraction of binary numbers is performed. Draw the half subtractor and full subtractor tables.
13. What are signed numbers? Give the different ways of representing the signed binary numbers in a digital system.
14. Explain the 1's and 2's complement representation of binary numbers.
15. What is the range of unsigned and signed decimal numbers as well as binary numbers that can be represented in a 12 bit system?
16. Explain the Addition/Subtraction method of Signed Numbers in 2's complement representation taking suitable examples.

17. Discuss 9's and 10's complement of decimal numbers. How 10's complement is used for the addition of signed decimal numbers.
18. Discuss $(r-1)$'s and r's complement of a number system whose radix is r.
19. Discuss how the multiplication of the binary numbers is performed.
20. Explain the floating representation of binary numbers in 16 bit machine.
21. Explain the floating representation of binary numbers in 24 bit machine.
22. Convert the following decimal numbers into their equivalent binary numbers: (i) 336 (ii) 679 (iii) 5797 (iv) 4391
    Ans.: (i) 101010000 (ii) 1010100111 (iii) 1011010100101
    (iv) 1000100100111

23. Convert the following binary numbers into their equivalent decimal numbers: (i) 1010111 (ii) 1110101 (iii) 100010011 (iv) 110010001
    Ans.: (i) 87 (ii) 117 (iii) 275 (iv) 401
24. Covert the following binary numbers into their octal, hexadecimal and decimal equivalent: (i) 1011101 (ii) 10101011101 (iii) 1001010111 (iv) 10111101
    Ans.: (i) $(135)_8$, $(5D)_{16}$, $(93)_{10}$ (ii) $(2535)_8$, $(55D)_{16}$, $(1373)_{10}$
    (iii) $(1127)_8$, $(257)_{16}$, $(599)_{10}$ (iv) $(275)_8$, $(BD)_{16}$, $(189)_{10}$

25. Convert the following hexadecimal number to binary and then to octal
    (i) 2BAFC (ii) 67DEF (iii) 2567C (iv) 2AB76
    Ans.: (i) $(101011101011111100)_2$, $(535374)_8$
    (ii) $(1100111110111101111)_2$, $(1476757)_8$
    (iii) $(101010101101110110)_2$, $(525566)_8$

26. Convert the following octal numbers into their decimal equivalent:
    (i) 26775 (ii) 67344 (iii) 53276 (iv) 15405
    Ans.: (i) $(11773)_{10}$ (ii) $(28388)_{10}$ (iii) $(22206)_{10}$ (iv) $(6917)_{10}$

27. Convert the following octal numbers into their binary equivalent:
    (i) 126705 (ii) 207344 (iii) 350276 (iv) 415005
    Ans.: (i) $(1010110111000101)_2$ (ii) $(10000111011100100)_2$
    (iii) $(11101000010111110)_2$ (iv) $(100001101000000101)_2$

28. Express the following decimal numbers into their equivalent octal and hexadecimal numbers.
    (i) 798562 (ii) 179856 (iii) 369852 (iv) 9120305
    Ans.: (i) $(3027542)_8$, $(C2F62)_{16}$ (ii) $(537220)_8$, $(2BE90)_{16}$
    (iii) $(1322274)_8$, $(5A4BC)_{16}$ (iv) $(42625061)_8$,$(8B2A31)_{16}$

29. Convert the following decimal numbers into binary numbers.
    (i) 697.625 (ii) 1457.23 (iii) 22097.96 (iv) 39870.0625
    Ans.: (i) $(1010111001.101)_2$
    (ii) $(10110110001.0011101011100001 \ldots.)_2$
    (iii) $(101011001010001.\ 111101011100001 \ldots)_2$
    (iv) $(1001101110111110.0001)_2$

30. Convert the following decimal numbers into octal numbers.
(i) 4537.362 (ii) 7192.025 (iii) 4389.125 (iv) 1767.3
Ans.: (i) $(10671.27126010\ldots)_8$ (ii) $(16030.1)_8$
(iii) $(3347.231463146\ldots)_8$

31. Convert the following binary numbers to their equivalent octal and hexadecimal numbers. (i) 11011011.011 (ii) 101110111.1011
(iii) 1011111001.111011 (iv) 1000101011.011011
Ans.: (i) $(333.3)_8$, $(DB.6)_{16}$ (ii) $(567.54)_8$, $(177.B)_{16}$
(iii) $(1371.73)_8$, $(2F9.EC)_{16}$ (iv) $(1053.33)_8$, $(228.6C)_{16}$

32. Express the following hexadecimal numbers to their equivalent binary and octal numbers. (i) 3AC45B.20B (ii) 6754A.2FE
(iii) 4596BC.31DF (iv) 2369.2AB7
Ans.:
(i) $(11101011000100010110 11.0011000111011111)_2$, $(16542133.143574)_8$ (ii) $(1100111010101001010.00101111111)_8$, $(1472512.1376)_8$
(iii) $(10001011001011010111100.0011000111011111)_2$, $(21313274.143574)_8$ (iv) $(10001101101001.0010101010110111)_2$ $(21551.125334)_8$

33. Convert the following decimal numbers into their equivalent numbers in base 3 and base 5.
(i) 8923 (ii) 45967 (iii) 543294 (iv) 30107
Ans.: (i) $(110020111)_3$, $(241143)_5$ (ii) $(2100001111)_3$, $(2432332)_5$ (iii) $(1000121021000)_3$, $(114341134)_5$ (iv) $(1112022002)_3$, $(1430412)_5$

34. Add the following numbers in binary:
(i) $(45)_{10} + (67)_{10}$ (ii) $(246)_{10} + (397)_{10}$
(iii) $(6754)_{10} + (2450)_{10}$ (iv) $(4096)_{10} + (256)_{10}$
Ans.: (i) $(1110000)_2$ (ii) $(1010000011)_2$
(iii) $(10001111110100)_2$ (iv) $(1000100000000)_2$

35. Subtract the following numbers in binary:
(i) $25763_{10} - 2454_{10}$ (ii) $9832_{10} - 2432_{10}$
(iii) $4506_{10} - 2004_{10}$ (iv) $9006_{10} - 4598_{10}$
Ans.: (i) 101101100001101_2 (ii) 1110011101000_2
(iii) 100111000110_2 (iv) 1000100111000_2

36. Perform the following binary additions.
(i) 11010111 +1011010 (ii) 10111101.101 + 10101001.011
(iii) 100101101.101 + 10010110.01
(iv) 111010110.1101+10111011.0101
Ans.: (i) 100110001 (ii) 101100111.000 (iii) 111000011.111
(iv) 1010010010.0010

37. Perform the following binary subtraction.
(i) 11010011 – 1010010 (ii) 10100101.101 – 10111001.001

(iii) 100101011.001 – 10100110.01
(iv) 110010110.1001–10100011.0111
Ans.: (i) 10000001 (ii) – 10011.100 (iii) 10000100.111
(iv) 11110011.0010

38. Solve the following:
(i) $(11011)_2 \times (101)_2 = (?)_2$
(ii) $(110010)_2 \times (1011)_2 = (?)_2$
(iii) $(1101.011)_2 \times (101.01)_2 = (?)_2$
(iv) $(1.10011)_2 \times (10.101)_2 = (?)_2$
Ans.: (i) 10000111 (ii) 1000100110 (iii) 1000110.00111
(iv) 100.00101111

39. Solve the following:
(i) $(11001)_2 \div (1011)_2 = (?)_2$ (ii) $(101010)_2 \div (1001)_2 = (?)_2$
(iii) $(10101.011)_2 \div (100.11)_2 = (?)_2$ (iv) $(1.00101)_2 \div (10.10)_2 = (?)_2$
Ans.: (i) Quotient $(10)_2$ and remainder $(11)_2$
(ii) Quotient $(100)_2$ and remainder $(110)_2$
(iii) Quotient $(100.1)_2$ and remainder $(00)_2$
(iv) Quotient $(.011)_2$ and remainder $(.111)_2$

40. Perform the following operations in 12-bit system using 2's complement method.
(i) – 149 – 126 (ii) 607 – 319 (iii) – 871 + 112 (iv) 312 – 540.
Ans.: (i) 111011101101 (ii) 000100100000 (iii) 110100001001
(iv) 111100011100

41. Subtract the following using 10's complement method:
(i) 94562074 – 495421 (ii) 3216547 – 9876540
Ans.: (i) 94066653 (ii) – 6659993

42. What floating point number do the following numbers represent?
(i) 0111101001101110 (ii) 1011110110101001
(iii) 0110100010010111 (iv) 1110100011010100
Ans.: (i) + $(11110100100000.00)_2$
(ii) – $(100001010.00)_2$ (iii) +$(111101001000.00)_2$
(iv) –$(.00000000000001011101)_2$

———

2

Binary Codes

In the preceding chapter the usage of binary numbers and their arithmetic operations have been discussed. While working with digital machines which use binary numbers, the data is generally given to the input as well the information is taken from the output in form of decimal numbers, because we are familiar only with the decimal numbers. The conversion of decimal numbers into binary and vice – versa is a slow process, which leads a communication problem between the man and the machine. In order to simplify this problem of communication between the man and the machine a number of codes for the decimal numbers have been devised. In the present chapter these codes known as binary codes will be discussed in detail.

2.1 Binary Coded Decimal Numbers: In a digital system which is capable of accepting or string only 0's and 1's, the usual way of conversion of decimal numbers to binary number and vice –versa is a slow process and requires large electronic circuitry. Therefore, instead of converting the decimal numbers to binary, it will be simpler to convert each decimal digit to binary, i.e. a coding system is used for the conversion of each decimal digit to binary. Such a coding system is known as Binary Coded Decimal or BCD in short. For example, a decimal number 13 in binary is written as 1101 whereas in BCD form individual decimal digit 1 and 3 may be written in four bit binary numbers as 0001 0011.

A large number of coding schemes is possible to encode 10 distinct symbols of decimal system namely, 0 1 2 ….. 9. To represent each symbol of decimal system in the form of 0's and 1's of binary system, at least 4 bits are required as $2^3 = 8 < 10$ and $2^4 = 16 > 10$. There are 16 possible combinations in which four bits are arranged and it is required to have only 10 combinations of 4 bits. In this way, to pick up an ordered sequence of 10 out of 16 items, as many as 30 billion (3×10^{10}) ways or codes are there. Out of these codes only a few are of importance which may be classified in the following categories.

1. Weighted codes
2. Self complementing codes
3. Cyclic codes
4. Error detecting codes
5. Error correcting codes

2.2 Weighted Codes: In the weighted codes, weights or values are assigned to the binary bits as per their bit position. The decimal value of a code is the algebraic sum of weighted bits. In other words, the decimal number *N* in the weighted codes is given by:

$$N = \sum_{i=1}^{4} W_i b_i$$

where W_i denotes the weight that is assigned to i[th] binary bit,

b_i is the binary bit (0 or 1) in the i[th] bit position.

The most popular weighted codes are 8421, $84\bar{2}\bar{1}$, 5421, 2421, 5211, 7421 etc. These weighted codes are given in Table 2.1.

Table 2.1

Decimal Numbers D	Weighted codes					
	8 4 2 1	8 4 $\bar{2}$ $\bar{1}$	5 4 2 1	2 4 2 1	5 2 1 1	7 4 2 1
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 1 1 1	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 1 1 0	0 0 1 0	0 0 1 0	0 0 1 1	0 0 1 0
3	0 0 1 1	0 1 0 1	0 0 1 1	0 0 1 1	0 1 0 1	0 0 1 1
4	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 1 1	0 1 0 0
5	0 1 0 1	1 0 1 1	1 0 0 0	1 0 1 1	1 0 0 0	0 1 0 1
6	0 1 1 0	1 0 1 0	1 0 0 1	1 1 0 0	1 0 1 0	0 1 1 0
7	0 1 1 1	1 0 0 1	1 0 1 0	1 1 0 1	1 1 0 0	1 0 0 0
8	1 0 0 0	1 0 0 0	1 0 1 1	1 1 1 0	1 1 1 0	1 0 0 1
9	1 0 0 1	1 1 1 1	1 1 0 0	1 1 1 1	1 1 1 1	1 0 1 0

In the 8421 code, the weight assigned to bit position 1(i = 1) is 1, second position (i = 2) is 2, third position (i = 3) is 4 and weight assigned to forth position (i = 3) is 8. So the binary number 0110 represents the decimal digit 6 as 0*x*8+1*x*4+1*x*2+0*x*1 = 6. In the similar fashion the representation of decimal numbers (0 through 9) in different weighted codes is done. In the weighted codes, the negative weights may also be assigned e.g. 8 4 $\bar{2}$ $\bar{1}$ have the negative weight (–1) to the least significant bit and (–2) to the second least significant bit.

It is important to note that 8421 code is nothing but uses the natural weights for the representation of binary numbers hence 8421 codes also called as natural binary coded decimal (NBCD).

2.3 Self Complementing Codes: A code is said to be self complementing if the binary representation of a decimal number D in that code is 1's complement of the

decimal number (9 – D). For example let D = 5 in some code then that code will be self complementing if the binary representation of D and (9 – D) are 1's complement of each other. The weighted codes 2421, 5211 and 8 4 $\bar{2}$ $\bar{1}$ are self complementing whereas 8421 code is not self complementing, which may be verified from the table (2.1). A necessary condition for a weighted code to be self complementing is that the sum of the weights of the code should be 9.

It is important to note that the binary representation of decimal digits in 2421 and 5211 may be done in different ways, but these are represented in the sequence as are given in the table 2.1, otherwise the codes will not show the self complementing property.

Further it is not a necessary condition that only the weighted codes are self complementing. Another important code is the excess – 3 (XS -3)code, which is shown in table 2.2. This code in not a weighted code but shows the self complementing property. Excess – code is derived from 8421 code by adding 3 (0011) to all code groups. The arithmetic becomes simple by the use of this code which will be discussed in the later section.

Table 2.2

Decimal numbers D	Excess – 3 code
0	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
4	0 1 1 1
5	1 0 0 0
6	1 0 0 1
7	1 0 1 0
8	1 0 1 1
9	1 1 0 0

Example 2.1: Encode the following decimal numbers into 8421, 2421 and excess – 3 codes.

1548 , 7896, 5602

Solution:

Decimal No.	8 4 2 1	2 4 2 1	Excess – 3
1548	0001010101001000	0001101101001110	0100100001111011
7896	0111100010010110	1101111011111100	1010101111001001
5602	0101011000000010	1011110000000010	1000100100110101
13	00010011	00010011	01000110

Example 2.2: Decode the following BCD numbers:

(i)	01000001	0111000000010010	1001000110000110
(ii)	01100001	0101000101010000	0101000000000001

Solution:

(i)	41	7012	9186
(ii)	61	5150	5001

2.4 Cyclic Codes: Another class of binary codes is the cyclic codes. Before discussing the cyclic codes it is necessary to explain the Hamming distance first. Hamming distance is defined as the number of places the binary bits differ in two consecutive numbers in a particular code. For example in 8421 code hamming distance between 0 (0000) and 1(0001) is one, as there is a change only in the one bit position (0 to 1 of LSB). Similarly, the hamming distance between 1 and 2 is 2; between 3 and 4 is 3. Hence one can say that the hamming distance between two successive code groups of 8421 code in not constant. There are many other codes in which the hamming distance is not unity. Cyclic codes have the unit hamming distance property. In many practical applications such as analog to digital converter, codes of unit hamming distance are used. Gray code is a particularly useful cyclic code and a four bit gray code is shown in table 2.3.

Table 2.3

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

This code may also be shown as the elements of K – map (Karnaugh map) shown in figure 2.1.

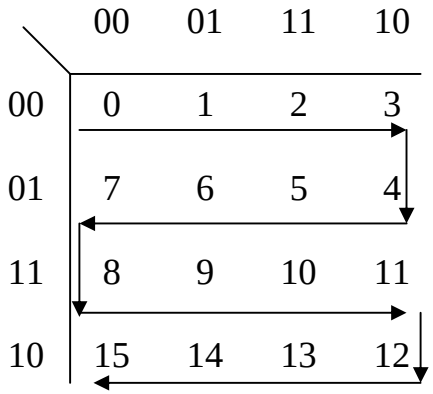


Fig. 2.1

Gray code is also called as the reflected binary code. The reflected binary code is given below. The method of writing the reflected binary code is that 0 and then 1 is written to the LSB and a mirror is supposed to be placed below 1. The mirror image of 0 & 1 will be 1 & 0. So sequence of the LSB of the code will be 0, 1, 1 & 0. Now 0 is written at the second place (as the second LSB) above the mirror and 1 to the numbers below the mirror. This code for two bits will be as 00, 01, 11 & 10. For the extension of this code to the three bits, a mirror is again supposed to be placed below 10. The mirror image for the two bits will be 10, 11, 01 & 00. To the third bit 0 is added to the binary number above the mirror and 1 is added to the mirror imaged numbers of two bits. The codes for three bits will, therefore, be 000, 001, 011, 010, 110, 111, 101 & 100. Similarly it can be extended for four bits, five bits etc.

Dec.No.	Gray Code				
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	1
3	0	0	0	1	0
4	0	0	1	1	0
5	0	0	1	1	1
6	0	0	1	0	1
7	0	0	1	0	0
8	0	1	1	0	0
9	0	1	1	0	1
10	0	1	1	1	1
11	0	1	1	1	0
12	0	1	0	1	0
13	0	1	0	1	1
14	0	1	0	0	1
15	0	1	0	0	0
16	1	1	0	0	0

17	1	1	0	0	1
18	1	1	0	1	1
19	1	1	0	1	0
20	1	1	1	1	0
21	1	1	1	1	1
22	1	1	1	0	1
23	1	1	1	0	0
24	1	0	1	0	0
25	1	0	1	0	1
26	1	0	1	1	1
27	1	0	1	1	0
28	1	0	0	1	0
29	1	0	0	1	1
30	1	0	0	0	1
31	1	0	0	0	0

Further before discussing the method of conversion of binary to gray code and vice versa, it is important to discuss the other important cyclic codes. In the gray code discussed above is not suitable for its use as cyclic BCD code, since when we move from decimal number 9 to 0 (successive digits), the hamming distance is three. The cyclic BCD code should have unit hamming distance for all successive digits. The most commonly used cyclic code is shown in table 2.4 and it K map in figure 2.2

Table 2.4

Decimal Number	Gray Code
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 0 1 0
4	0 1 1 0
5	1 1 1 0
6	1 0 1 0
7	1 0 1 1
8	1 0 0 1
9	1 0 0 0

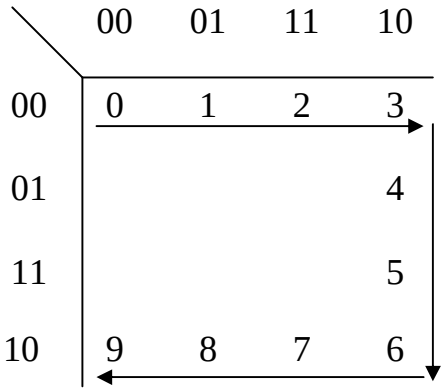
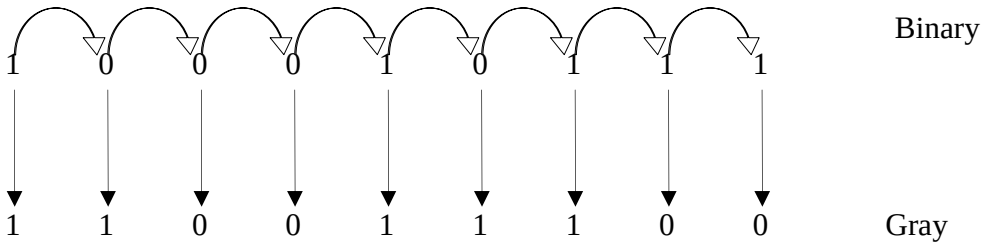


Fig. 2.2

It may also be observed that the cyclic code shown in the table 2.4 is a reflected BCD code. The reflection in any BCD code can be identified by comparing the upper five code words with the lower five. If the upper and lower codes are mirror imaged except for one bit, then the code is reflected. Reflection is useful property that makes 9's complementation easy to implement.

2.4.1 Conversion of Binary to Gray Code: The gray code being the reflected binary number is difficult to obtain for a large decimal number. So the conversion of binary numbers to gray code is required to be obtained directly. The method of converting the binary number to gray code is follows:

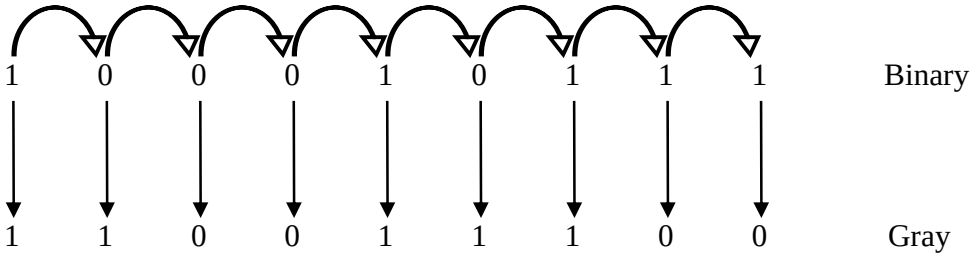
The most significant bit is recorded as the first most significant bit of the gray code, which is then added with the bit of next position. The sum is recorded as the next bit of the gray code, of course neglecting the carry, if any. This process is continued till the LSB is reached. For example for the conversion of binary number 100010111 we proceed as given below:



The Gray equivalent of binary number $(100010111)_2$ is $(110011100)_g$.

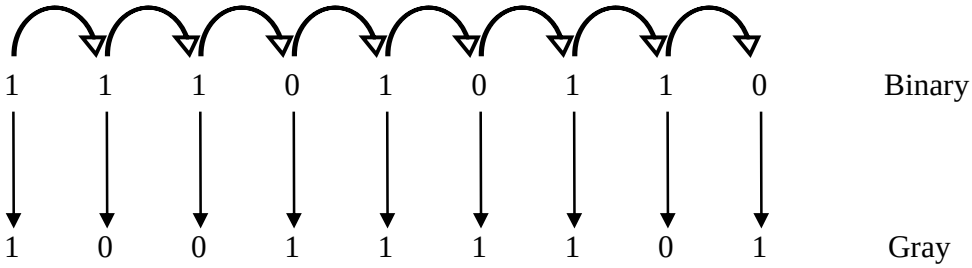
Example 2.3: Find the gray equivalent of the following binary numbers:
(i) 100010111 (ii) 111010110 (iii) 10000101011

Solution: (i)



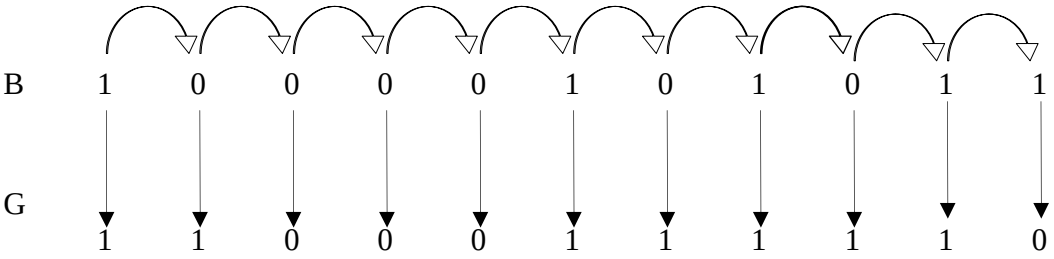
So $(100010111)_2 = (110011100)_g$

(ii)



So $(111010110)_2 = (100111101)_g$

(iii)



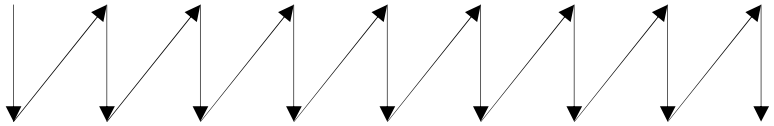
So $(10000101011)_2 = (11000111110)_g$

2.4.2 Conversion of Gray Code to Binary: The method of converting the gray code to binary number is follows:

The most significant number of gray code is recorded as the most significant of the binary number, which is then added with the next bit of the gray code. The sum is recorded as the next bit of the binary number, neglecting the carry if any. The process is continued till the least significant bit is obtained.

For example for the conversion of $(110011100)_g$ to binary we proceed as given below:

1 1 0 0 1 1 1 0 0 Gray



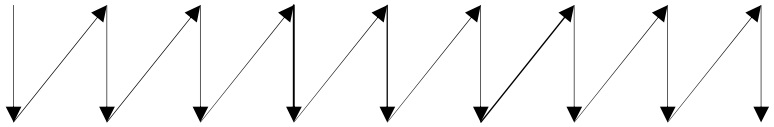
1 0 0 0 1 0 1 1 1 Binary

So the binary equivalent of gray code $(110011100)_g = (100010111)_2$.

Example 2.4: Find the binary equivalent of the following gray code numbers:
(i) 101010101 (ii) 110010101 (iii) 10010101111

Solution: (i)

1 0 1 0 1 0 1 0 1 Gray

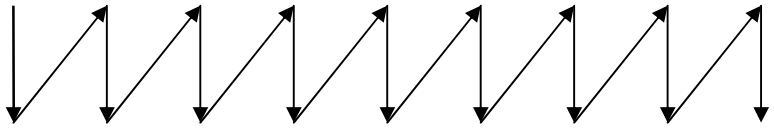


1 1 0 0 1 1 0 0 1 Binary

So $(101010101)_2 = (110011001)_g$.

(ii)

1 1 0 0 1 0 1 0 1 Gray

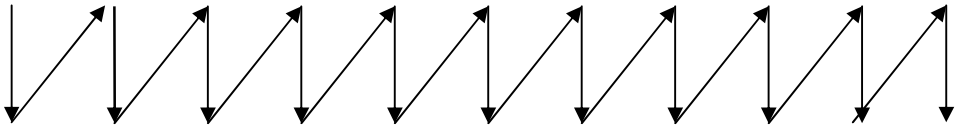


1 0 0 0 1 1 0 0 1 Binary

So $(110010101)_g = (100011001)_2$.

(iii)

1 0 0 1 0 1 0 1 1 1 1 Gray



1 1 1 0 0 1 1 0 1 0 1 Binary

So $(10010101111)_g = (11100110101)_2$.

2.5 Error Detecting Codes: A group of bits is known as word and it moves as an entity in the digital systems, i.e. a word is moved from one block to other block of the digital system or transmitted from one place to the other. During this transmission it is very likely a bit might change resulting a change in the word and an error is said to have occurred. The error (change in bit from 0 to 1 or vice versa) is introduced due to the external noise in the physical communication medium. An error detecting code can be used for the detection of error in the transmission. This code will simply detect the error but will not correct the error. In forming the error detecting codes (also called error checking codes), an additional bit is introduced with the word. The additional bit included with the word is known as parity bit and is used to make the total number of 1's in the word either even or odd.

Two types of parity may be considered for error detection namely even parity and odd parity. For even parity, the parity bit is set to 1 so that the sum of bits in the number is even i.e. number of 1's in the number is even. However, for the odd parity, the parity bit is set 1 so that the sum of bits in the number is odd. For example, in number 1001101 there are four 1's so a parity bit P introduced with the given number is 1 for odd parity (number of 1's becomes odd); and P is 0 for odd parity (number of 1's remains even). The number along with the parity bit will, therefore, be 10011011 for odd parity and 10011010 for even parity. A message of four bits and parity P is shown in table 2.5.

Table 2.5

Word of 4 bits	Parity bit P (Even)	Word of 4 bits	Parity bit P (Odd)
0000	0	0000	1
0001	1	0001	0
0010	1	0010	0
0011	0	0011	1
0100	1	0100	0
0101	0	0101	1
0110	0	0110	1
0111	1	0111	0
1000	1	1000	0
1001	0	1001	1
1010	0	1010	1
1011	1	1011	0
1100	0	1100	1
1101	1	1101	0
1110	1	1110	0
1111	0	1111	1

For the error detection the parity bit P generated by some electronic circuitry is transmitted along with the word at the transmitter end. The word along with the parity is received at the receiving end where the data and parity bit is checked. At the receiving end there will be parity check network, which will detect if the proper parity is received. The parity check network will give an alarm or indication if parity check fails. This error detecting code is suitable if there is a change only in one bit, three bits or to odd number

of bits. If on the other hand error occurs at two or even number of places, the double parity check method is used.

For the double parity check consider a block of 36 bits recorded on a magnetic tape in 6 tracks with 6 bits along each track. Odd parity bit is added as the seventh bit to each track. A seventh row (of 6 bits) is also introduced as the odd parity row for each column. The odd parity check network will be used for row as well as for the column. In this way, if there is erroneously transmission of bit the parity check fails, on the row and column and the place of error is detected. This is illustrated in figure 2.3.

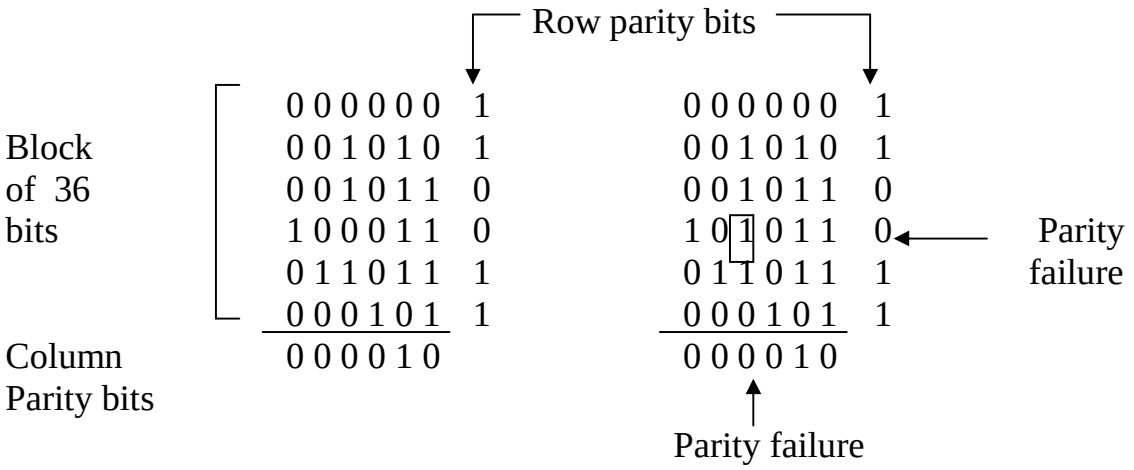


Fig. 2.3

2.6 Error Correcting Code or Hamming Code: In the forgoing section the error correcting codes were discussed which can only be used to detect the error occurred due to the transmission of binary information. It can neither indicate the place or bit position of error nor correct the incorrect bit. Hamming code also called self correcting code is most commonly used code which can not only detect the error but also finds the error position and correct it.

This code is being discussed for correcting a single error on information of any length. Suppose 8421 code bits are to be transmitted and the error for one bit position is to be corrected. For this at least 3 parity bits are to be used. So to find out the error position 7 bit hamming code will be constructed. The word format is given below:

7	6	5	4	3	2	1	←	Bit Number
D_7	D_6	D_5	P_4	D_3	P_2	P_1	←	Name of bit position

In the above format of the Hamming code, D represents the data bit and P represents the parity bit. So the bit positions 1, 2 and 4 (P_1, P_2, and P_4) are used for parity check bits and bit positions 3, 5, 6 and 7 (D_3, D_5, and D_7) as 4 bit word (8421 code data).
P_1 is the even parity bit for bits 3, 5, 7 (D_3, D_5 and D_7)
P_2 is the even parity bit for bits 3, 6, 7 (D_3, D_6 and D_7)
P_4 is the even parity bit for bits 5, 6, 7 (D_5, D_6 and D_7)

The 7 bit Hamming code for 8421 data is shown in table 2.6.

Table 2.6

Decimal numbers	7 D7	6 D6	5 D5	4 P4	3 D3	2 P2	1 P1
0	0	0	0	0	0	0	0
1	1	0	0	1	0	1	1
2	0	1	0	1	0	1	0
3	1	1	0	0	0	0	1
4	0	0	1	1	0	0	1
5	1	0	1	0	0	1	0
6	0	1	1	0	0	1	1
7	1	1	1	1	0	0	0
8	0	0	0	0	1	1	1
9	1	0	0	1	1	0	0

The following procedure is used to detect and correct the error after the code is received:

1. If P_1 satisfies as the even parity bit for bits 3, 5, 7 then assume $C_1 = 0$ else $C_1 = 1$.
2. If P_2 satisfies as the even parity bit for bits 3, 6, 7 then assume $C_2 = 0$ else $C_2 = 1$
3. If P_3 satisfies as the even parity bit for bits 5, 6, 7 then assume $C_3 = 0$ else $C_3 = 1$.
4. The decimal equivalent of $C_3C_2C_1$ gives the position of incorrect bit, which may be corrected. If $C_3C_2C_1 = 000$ then there is no error in the code.

For example a seven bit Hamming code is received as 1000010 and one has to find if there is any error in the received data.

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

Now C_1 is 1 as P_1 does not satisfy the even parity bit for bits 3, 5, 7.

C_2 is 0 as P_2 satisfies the even parity bit for bits 3, 6, 7.

C_3 is 1 as P_4 does not satisfy the even parity bit for 5, 6, 7.

So $C_3C_2C_1 = 101$, it indicates that there is an error in the fifth place. At the fifth place there is 0 which should be corrected to 1. So the correct hamming code is 1010010.

From the above discussion it is clear that this code can be used for detecting and correcting the error by using extra digital circuitry. This code can also be extended to transmit the data of any length by introducing more parity bits.

Example 2.5: Write the 7 bit Hamming code for a four bit word 1010.

Solution:

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

P_1 will be zero (even parity bit) for bits 3, 5, 7.
P_2 will be zero (even parity bit) for bits 3, 6, 7.
P_4 will be one (even parity bit) for bits 5, 6, 7.
So the 7 bit Hamming code will be 0101100.

Example 2.6: A seven bit Hamming code received at the receiver is 1110100. Is there any error in the received code? If yes, what is the correct code?
Solution:

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

Now C_1 is 1 as P_1 does not satisfy the even parity bit for bits 3, 5, 7.
C_2 is 1 as P_2 does not satisfy the even parity bit for bits 3, 6, 7.
C_3 is 1 as P_4 does not satisfy the even parity bit for 5, 6, 7.
So $C_3C_2C_1$ = 111, it indicates that there is an error in the seventh place. At the seventh place there is 1 which should be corrected to 0. So the correct hamming code is 0110100.

2.7 BCD Addition: In the present section the addition of BCD numbers will be discussed since in digital computers BCD numbers are processed. The BCD code (8421 Code) represents the decimal numbers in the similar fashion as binary numbers. The binary numbers 1010 through 1111 are the illegal codes in 8421 code. Due to these illegal codes the addition of decimal numbers in BCD will be different. To understand the BCD addition, two cases of decimal addition are considered.
Case I : decimal numbers 6 & 3 are to be added.

Decimal form	BCD form
6	0110
+ 3	+ 0011
9	1001

BCD number 1001 shows the correct answer as 1001 is equal to 9.

Case II : decimal numbers 6 & 7are to be added.

Decimal form	BCD form
6	0110
+ 7	+ 0111
13	1101

BCD number 1101 is not correct as it is an illegal code since it does not occur in BCD. The correct answer would be 0001 0011.

From the above discussion it is observed that if the sum is less than or equal to 9, the correct answer will be obtained. If on the contrary, the sum is more than 9, the incorrect answer is obtained because 6 illegal codes 1010 through 1111. So to get the correct answer, 0000 is to be added if the sum is less than or equal to 9; and 0110

(decimal 6) is to be added if the answer is more than 9. The answer is observed to be more than 9 if illegal codes 1010 through 1111 are obtained or a carry to the next BCD number is occurred.

For example 476 and 394 are to be added using BCD numbers.

BCD number for 476 is: 0100 0111 0110
BCD number for 394 is: 0011 1001 0100

```
                    1111 111   1
Additon:            0100 0111  0110
                +   0011 1001  0100
                    ---------------
                    1000 0000  1010
Correction to be applied: + 0000 0110  0110
                    ---------------
Correct answer:     1000 0111  0000
Decimal number       8    7     0
```

In this example 0110 is added to LSD and second LSD because 1010 is the illegal code in 8421 and the second LSD gives a carry to the MSD.

Example 2.7: Add 8765 and 7043 in BCD code.
Solution:

BCD number for 8765 is: 1000 0111 0110 0101
BCD number for 3943 is: 0011 1001 0100 0011

```
                        111 111  1      111
Additon:          0000  1000 0111 0110 0101
                + 0000  0011 1001 0100 0011
                  ---------------------------
                  0001  1100 0000 1010 1000
Correction to be applied: + 0000 0110 0110 0110 0000
                  ---------------------------

Correct answer:   0001 0010 0111 0000 1000
Decimal number     1    2    7    0    8
```

So the correct answer is 0001 0010 0111 0000 1000.

2.8 Excess–3 Addition: Addition of decimal numbers can also be performed using excess–3 codes. One may recall that in excess – 3 codes first three and last three numbers of 4 bit binary numbers (0000 through 0010 and 1101 through 1111) are illegal codes. So while adding the numbers in excess–3 codes, these illegal codes will have to be eliminated. To understand the excess–3 addition, two examples given below are considered:

Case I Two numbers 3 and 6 are to be added.

3 Excess- 3 of decimal number 3 is: 0110

+ 6		Excess- 3 of decimal number 6 is:	+ 1001
9		Sum of these numbers is:	1111

The sum is wrong due to illegal code 1111. The illegal code 1111 shows the excess - 6 because 0011 (3) is added is each number. So to get the correct answer 0011 (3) is to be subtracted from the above sum.

i.e. 1111 – 0011 = 1100

1100 gives the correct answer, as it shows 9 in excess – 3 code.

Case II Two numbers 7 and 8 are to be added.

7		Excess- 3 of decimal number 7 is:	1010
+ 8		Excess- 3 of decimal number 8 is:	+ 1011
15		Sum of these numbers is:	1 0101

In this case too the sum is wrong because of the illegal code 0101. To avoid the illegal code and to get the correct answer 0011 (3) is added.

i.e. 0101 + 0011 = 1000

1000 shows the correct answer for the LSD as it shows 5 in excess–3 code.

From the above discussion, one can get an inference that if the sum is less than or equal to 9, the correct answer is obtained by subtracting 0011 (3) from the incorrect answer. However, if the sum is more than 9, then 0011 (3) is to be added to the incorrect sum. The answer is observed to be more than 9 if a carry to the next digit is occurred.

For example addition of 45 and 38 using excess–3 code may be given as:

Excess – 3 of 45 :	0111	1000	
Excess – 3 of 38 : +	0110	1011	
	1110	0011	
	– 0011	+ 0011	
	1011	0110	= 83

Example 2.8: Add 876 and 704 in excess–3 code.

Solution:

Excess – 3 of 0876:	0011	1011	1010	1001	
Excess – 3 of 0704 : +	0011	1010	0011	0111	
	0100	0101	1011	0000	
	– 0011	+ 0011	– 0011	+ 0011	
	0100	1000	1011	0011	= 1580

2.9 Alphanumeric Codes: In the preceding sections of this chapter, different codes for numeric data have been discussed. But in computers or in digital systems the numeric data as well letters of alphabet, punctuation marks and other special characters are also processed. So for representing this type of information data, different codes (groups of 0's and 1's) are to be discussed. These codes are called as alphanumeric codes. To represent the decimal numbers 0 through 9 in binary form four bits are used as 2^4 = 16. However, in representing the 10 decimal numbers, 26 upper case letters (A, B, C…, Z), 26 lower case letters (a, b, c….., z), 7 punctuation marks (, : ; " ' . ?), and about 20 to 40 special characters (+, -, <, >, =, $, % …) codes of minimum of 6 bits are required. The 6 bit alphanumeric code referred to as internal code is shown in table 2.7.

Table 2.7

Character	6 bit internal code	Character	6 bit internal code
A	010 001	Y	111 000
B	010 010	Z	111 001
C	010 011	0	000 000
D	010 100	1	000 001
E	010 101	2	000 010
F	010 110	3	000 011
G	010 111	4	000 100
H	011 000	5	000 101
I	011 001	6	000 110
J	100 001	7	000 111
K	100 010	8	001 000
L	100 011	9	001 001
M	100 100	BLANK	110 000
N	100 101	.	011 011
O	100 110	(	111 100
P	100 111	+	010 000
Q	101 000	$	101 011
R	101 001	*	101 100
S	110 010	)	011 100
T	110 011	-	100 000
U	110 100	/	110 001
V	110 101	,	111 011
W	110 110	=	001 011
X	110 111		

A more commonly used alphanumeric code is the ASCII (American Standard Code for Information Interchange) code pronounced as “as-kee”. Basically, it is a 7 bit code which is shown in table 2.8. It is used for printers and teletypewriters when interfaced with computers. The 8 bit ASCII code is also used for practical purposes, in which 8th bit is added for parity.

Table 2.8

Character	7 bit ASCII code	Hex	Character	7 bit ASCII code	Hex
0	011 0000	30	X	101 1000	58
1	011 0001	31	Y	101 1001	59
2	011 0010	32	Z	101 1010	5A
3	011 0011	33	[	101 1011	5B
4	011 0100	34	\	101 1100	5C
5	011 0101	35	]	101 1101	5D
6	011 0110	36	^	101 1110	5E
7	011 0111	37	-	101 1111	5F
8	011 1000	38	.	110 0000	60
9	011 1001	39	a	110 0001	61
:	011 1010	3A	b	110 0010	62
;	011 1011	3B	c	110 0011	63
<	011 1100	3C	d	110 0100	64
=	011 1101	3D	e	110 0101	65
>	011 1110	3E	f	110 0110	66
?	011 111	3F	g	110 0111	67
@	100 0000	40	h	110 1000	68
A	100 0001	41	i	110 1001	69
B	100 0010	42	j	110 1010	6A
C	100 0011	43	k	110 1011	6B
D	100 0100	44	l	110 1100	6C
E	100 0101	45	m	110 1101	6D
F	100 0110	46	n	110 1110	6E
G	100 0111	47	o	110 1111	6F
H	100 1000	48	p	111 0000	70
I	100 1001	49	q	111 0001	71
J	100 1010	4A	r	111 0010	72
K	100 1011	4B	s	111 0011	73
L	100 1100	4C	t	111 0100	74
M	100 1101	4D	u	111 0101	75
N	100 1110	4E	v	111 0110	76
O	100 1111	4F	w	111 0111	77
P	101 0000	50	x	111 1000	78
Q	101 0001	51	y	111 1001	79
R	101 0010	52	z	111 1010	7A
S	101 0011	53	{	111 1011	7B
T	101 0100	54	:	111 1100	7C
U	101 0101	55	}	111 1101	7D
V	101 0110	56	~	111 1110	7E
W	101 0111	57	DELETE	111 1111	7F

Another quite often used 8 bit alphanumeric code is EBCDIC (Extended BCD Interchange Code) is shown in table 2.9.

Table 2.9

Character	8 bit EBCDIC code	Character	8 bit EBCDIC code
A	1100 0001	S	1110 0010
B	1100 0010	T	1110 0011
C	1100 0011	U	1110 0100
D	1100 0100	V	1110 0101
E	1100 0101	W	1110 0110
F	1100 0110	X	1110 0111
G	1100 0111	Y	1110 1000
H	1100 1000	Z	1110 1001
I	1100 1001	0	1111 0000
J	1101 0001	1	1111 0001
K	1101 0010	2	1111 0010
L	1101 0011	3	1111 0011
M	1101 0100	4	1111 0100
N	1101 0101	5	1111 0101
O	1101 0110	6	1111 0110
P	1101 0111	7	1111 0111
Q	1101 1000	8	1111 1000
R	1101 1001	9	1111 1001

Problems:

1. Distinguish between a binary and a BCD code. Why BCD codes are use for decimal numbers in digital systems?
2. Differentiate between weighted and non – weighted binary codes. List some weighted codes and define one of them.
3. What is self complementing code? Show that 2421 is a self complementing code whereas 8421 is not a self complementing code.
4. Discuss the excess–3 code and show that it is a self complementing code.
5. Describe the gray code. What are characteristics of gray code? It is also known as reflected binary code – comment.
6. Explain how the gray code is converted to binary numbers and vice– versa.
7. What is Hamming distance? Describe unit hamming distance cyclic code.
8. Discuss 7 bit even parity error correcting hamming code.
9. Explain how the BCD addition is performed.
10. Explain how the addition in excess–3 codes is performed.
11. Name some alphanumeric codes. Write the ASCII code for decimal numbers 0 through 9.
12. Write the following decimal numbers in 8421 code:
    7958, 5689, 209

(Ans: 0111100101011000, 0101011010001001, 001000001001)

13. Convert the following BCD (8421) code numbers to decimal numbers:
    (i) 0100001100000110
    (ii) 0010100101110000
    (iii) 1001100000000001
    (iv) 0101010000100001
    (Ans.: 4316, 2970, 9801, 5421)

14. Convert the following decimal numbers to XS3 (excess –3) code:
    1026, 4375, 6980, 4415
    (Ans.: 0100001101011001, 0111011010101000, 1001110010110011, 01110111 0100 1000)

15. Convert the following excess –codes to decimal numbers:
    (i) 1100011101011001
    (ii) 0101011000110110
    (iii) 0110011101000101
    (iv) 1000010010111001
    (Ans.: 9426, 2303, 3412, 5186)

16. Convert the following decimal numbers to 2421 code numbers:
    1014, 2397, 6419, 8474
    (Ans.: 0001000000010100, 0010001111111101, 1100010000011111, 11110010011010100)

17. Convert the following decimal numbers to gray code:
    8975, 4568, 23501, 10254
    (Hint.: first convert the decimal numbers to binary then to gray code)
    (Ans.: 11001010001000, 1100100110100, 111011000101011, 11110000001001)

18. Convert the following gray code numbers to binary numbers.
    (i) $(1010111010000101110)_g$
    (ii) $(1111001011011011011)_g$
    (iii) $(10110111011111101)_g$
    (iv) $(10000100100100100)_g$
    (Ans.: $(1100101100000110100)_2$, $(1010001101101101010)_2$ $(11011010010101001)_2$, $(11111000111000111)_2$

19. Construct 7 –bit even parity Hamming code for transmitting the following digital data:
    (i) 0101 (ii) 1000 (iii) 0110
    (Ans.: 1010010, 0000111, 0110011)

20. A seven bit Hamming code received at the receiver is 1001001. Is there any error in the received code? If yes, what is the correct code?
(Ans.: yes, 1001011)

21. A seven bit Hamming code received at the receiver is 0010100. Is there any error in the received code? If yes, what is the correct code? What is correct 4 bit data actually transmitted?
(Ans.: yes, 0110100, 1110)

22. Using the BCD (8421) code, perform the addition of following decimal numbers verify your answer:
(i) 0781 + 123 (ii) 1056 + 4891
(iii) 254 + 511 (iv) 3001 + 25

23. Using XS 3 code, perform the addition of decimal numbers given in example 21 and verify your answer.

24. Write your name in ASCII code.

25. Encode 0 to 25_{10} in ASCII, 8421, 5421 and 2421 codes.

26. Encode your name in 6 bit internal and 8 bit EBCDIC alphanumeric codes.

3

Boolean Algebra and Logic Gates

In the last two chapters, number system, binary numbers, binary codes and other alphanumeric codes have been discussed because in digital systems or digital computers binary numbers (groups of 0's and 1's) are processed. These binary bits 0 or 1 are designated by predefined voltage levels, making thereby the design of digital systems very simple. George Boole, an English mathematician later on became famous as logician developed an algebra called as Boolean algebra or logic algebra or switching algebra based on logics. Logic is basically human reasoning that tells us if certain proposition or declarative statement is true. For example, "a switch is ON", is a logic statement which is either true or false. The logic functions or digital functions for Boolean algebra will be formed by logics. In this chapter logic operations, logic gates and the Boolean algebra, will be discussed which will be used as the tools for the analysis and design of digital circuits.

3.1 Logic Operations: Three basic logic operations (AND, OR and NOT) are used in Boolean algebra which will now be discussed in detail.

AND operation: Consider a proposition or logical statement "Student having books **AND** his identity card can enter the college".

Outcome or the result of this statement is the entry of the student – True or False (Allowed or not allowed).

Student should have two essential things:

(i) Books – True or False (Student has the books or not).

(ii) I. Card – True or False (Student has the I. Card or not).

Table 3.1 illustrates that only the student who has books **AND** has his identity card, can enter the college. So the given proposition consists of two simple propositions (student having books and having his Identity card) connected with AND connective. This is known as AND operation.

Table 3.1

Student having		Student
Books	I. Card	Entry
False	False	False
False	True	False
True	False	False
True	True	True

This composite proposition can be shown by an electronic circuit as shown in figure 3.1, consisting of two switches A & B and a bulb L. The switch A represents the logic statement that the student having the books only. The ON & OFF positions of the switch A show the True and False of the above statement. Similarly the two positions of the switch B show the true and false of the second logical statement that the student has his identity card. The On and off positions of the bulb show the outcome or the result of the composite statement. On and off positions of the bulb respectively represent the true and false of the composite statement of the student entry in the college.

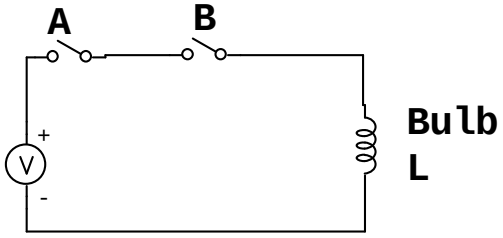


Fig. 3.1

Table 3.2 shows conditions for the bulb to glow. The bulb will glow only when both the switches are on, which is analogous to the statement that the student can enter the college when he has the books and his identity card.

Table 3.2

Switch		Bulb
A	B	L
off	off	off
off	on	off
on	off	off
on	on	on

The logical values may be assigned to the positions of the switches and the bulb. For example logic 0 is assigned to off position of the switches and the bulb; and logic 1 for the on positions. The truth table for the AND operation will therefore be given as shown in table 3.3.

Table 3.3

Switch		Bulb
A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

The AND operation may be represented in the mathematical form or the logic form as:

$$L = A \cdot B$$

It is pronounced as A dot B (A AND B)

Mathematically $0 \cdot 0 = 0$

$0 \cdot 1 = 1 \cdot 0 = 0$

$1 \cdot 1 = 1$

It is clear from all the above discussion A AND B means both. Both is the logic behind the word AND. The logic circuit designed for the demonstration of AND operation is known as AND gate. The symbolic representation of two input AND gate is shown in figure 3.2.

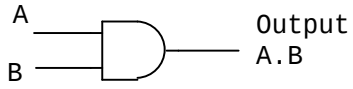


Fig. 3.2

The AND gates for more than two variables are also defined in the similar fashion.

OR operation: Consider another proposition "Student having books **OR** his identity card can enter the college".

Outcome or the result of this statement too is the same; entry of the student – True or False (Allowed or not allowed).

Student should have either of the two essential things:

(i) Books – True or False (Student has the books or not).

(ii) I. Card – True or False (Student has the I. Card or not).

Table 3.4 illustrates that only the student who has books **OR** has his identity card can enter the college. So the given proposition consists of two simple propositions (student having books **or** having his Identity card) connected with OR connective. This is known as OR operation.

Table 3.4

Student having		Student
Books	I. Card	Entry
False	False	False
False	True	True
True	False	True
True	True	True

The switching or electronic circuit for this operation may be given as shown in figure 3.3.

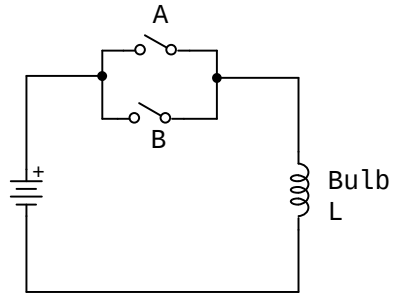


Fig. 3.3

The truth table for the OR operation is given in table 3.5, after assigning the logical values to the positions of the switches and the bulb. For example logic 0 is assigned to off position of the switches and the bulb; and logic 1 for the on positions.

Table 3.5

Switch		Bulb
A	B	L
0	0	0
0	1	1
1	0	1
1	1	1

The OR operation may be represented in the mathematical form or the logic form as:

L = A + B

It is pronounced as A OR B

Mathematically **0 + 0 = 0**

0 + 1 = 1 + 0 = 1

1+ 1 = 1

The logic circuit designed for the demonstration of OR operation is known as OR gate. The symbolic representation of two input OR gate is shown in figure 3.3.

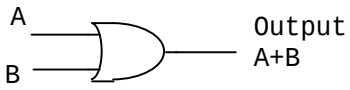


Fig. 3.3

OR gates for more than two variables may also be defined.

NOT operation: Consider the logic statement "The student who does not have the cell phone, is allowed to enter the college".

The student's entry is the outcome or the result of this statement – True or False (Allowed or not allowed).

Student should not have the cell phone, the only criterion at the check point. The student has the cell Phone or not (True or False).

Table 3.6 illustrates that the student who does **not** have the cell phone is allowed to enter the college. Hence it is known as NOT operator.

Table 3.6

Student has the cell phone	Student's entry
False	True
True	False

An electromagnetic relay may be used to demonstrate the NOT operation as shown in figure 3.4. When a positive and constant voltage is applied to the coil of the relay it gets energized. The bulb does not glow as it is connected to the normally close position of the relay. The bulb glows when no voltage is applied to the coil, as relay coil is de-energized. This shows the NOT or Inverter operation.

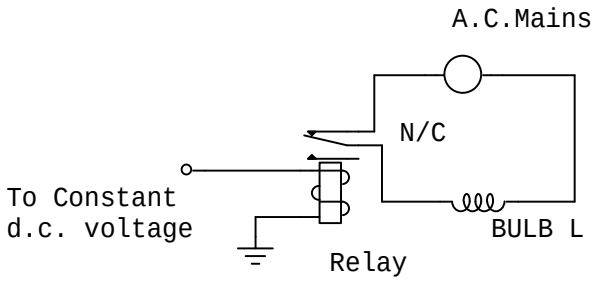


Fig. 3.4

Logically if input A is 0 the output is 1 represented by $\overline{A}$ (A bar).

So

Table 3.7

Input A	Output $\overline{A}$
0	1
1	0

The symbolic representation of NOT gate is shown in figure 3.5.

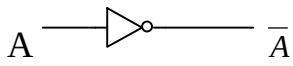


Fig. 3.5

AND & OR gates are called the binary gates because it is to be operated on at least two variables. The NOT or inverter gate is operated only on one variable hence it is

called as Unary operator. The detailed design of these gates will be discussed in a later chapter.

3.2 Postulates of Boolean Algebra: The two binary operators AND & OR $(\cdot, +)$ and one unary operator (NOT) $^{-}$ (a bar on a variable) discussed in the forgoing section are used in defining Boolean Algebra. The operands for these operations are the elements belong to a set. Let A and B are the elements which belong to a set S (A, B $\in$ S). Huntington in 1904 defined the following postulates of Boolean algebra. Postulates are the basic or universal rules, which are always assumed to be true and thus are not to be proved. The theorems of Boolean algebra may be derived from these postulates.

Postulate 1: Closure Property: For every A, B $\in$ S

(i) $D = A + B$ and $D \in S$: This is the closure property for OR operation.

(ii) $G = A.B$ and $G \in S$: This is the closure property for AND operation.

Postulate 2: Commutative Law: If A and B $\in$ S then

(i) $A + B = B + A$

(ii) $A \cdot B = B \cdot A$

Postulate 3: Identity Element:

(i) The identity element for OR operator is 0, if A $\in$ S and $0 \in S$ then $A + 0 = 0 + A = A$: 0 is known as addition identity.

(ii) The identity element for AND operator is 1, if A $\in$ S and $1 \in S$ then $A \cdot 1 = 1 \cdot A = A$: 1 is known as multiplication identity.

Postulate 4: Distributive Law: If A, B, C $\in$ S then

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

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

Postulate 5: Complementing Law: If A $\in$ S, there exist an element $\overline{A}$ (known as complement of a) which belong to S ($\overline{A} \in S$) such that

(i) $A + \overline{A} = 1$

(ii) $A \cdot \overline{A} = 0$

Postulate 6 : There are at least two elements A , B $\in$ S such that $A \neq B$.

Boolean algebra differs with ordinary algebra on the following points:

1. The distributive law A + B . C = (A + B) . (A + C) does not hold in ordinary algebra.

2. Complementing law does not hold in ordinary algebra, i.e., there is no equivalent of the unary operator (NOT operation) in ordinary algebra.
3. Boolean algebra does not have the additive inverse and multiplicative inverse due to which no subtraction or division operations exist.
4. Boolean algebra has only finite set of elements where as the ordinary algebra deals with real numbers which constitute a set with infinite number of elements. Switching algebra, a special class of Boolean algebra however, deals with two valued elements. The elements should have the values 0 and 1 only.

3.3 Two – Valued Boolean Algebra: The postulates of special class of Boolean algebra known as two valued Boolean algebra or switching algebra, may be discussed on the similar lines if a set of two elements 0 & 1 are assumed. The postulates are summarized below:

Table 3.8

OR operation	AND operation
0 + 0 = 0	1.1 = 1
0 + 1 = 1 + 0 = 1	1 . 0 = 0 . 1 = 0
1 + 1 = 1	0 . 0 = 0
$\overline{0} = 1$	$\overline{1} = 0$

These postulates are also given in the general form as:

$0 + A = A$ $\qquad$ $1 . A = A$

$A + \overline{A} = 1$ $\qquad$ $A.\overline{A} = 0$

3.4 Duality Principle: According to this theorem the postulates or theorems of Boolean algebra are given for one type of operation may be converted to other type of operation (i.e. OR to AND or vice versa) just by interchanging 0 with 1 and '+' with '.' This principle ensures that if a theorem is proved using the postulates of Boolean algebra then dual of this theorem automatically holds and need not to be proved separately.

3.5 Theorems of Boolean Algebra: The following are the general theorems or rules of Boolean algebra:

Theorem 1(a) $A + A = A$ 1(b) $A . A = A$

Proof: 1(a)

When A = 0 : 0 + 0 = 0 = A

When A = 1: 1 + 1 = 1 = A Thus A + A = A is proved.

1(b) is the dual of 1(a), which automatically holds.

Theorem 2(a) A + 1 = 1 2(b) A . 0 = 0

Proof: 2(a):

When A = 0 : 0 + 1 = 1
When A = 1: 1 + 1 = 1 Thus A + 1 = A is proved.

2(b) is the dual of 2(a).

Theorem 3(a) A + A . B = A 3(b) A . (A + B) = A

Proof 3(a):

L.H.S. = A + A . B
= A .1 + A . B (since A .1 = A)
= A . (1 + B)
= $A \cdot 1$ (since 1 + B = 1)
= A Proved.

3(b) is the dual of 3(a).

This theorem is also called as absorption theorem. Corollary of the absorption theorem is given as follows:

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

Theorem 4 $\overline{\overline{A}} = A$

Proof:

When A = 0 $\overline{A} = \overline{0} = 1, \overline{\overline{A}} = \overline{1} = 0 = A$

When A = 1 $\overline{A} = \overline{1} = 0, \overline{\overline{A}} = \overline{0} = 1 = A$ Thus $\overline{\overline{A}} = A$

Theorem 5(a) $A + \overline{A} \cdot B = A + B$ 5(b) $A \cdot (\overline{A} + B) = A \cdot B$

Proof 5(a):

L.H.S. = $A + \overline{A} \cdot B$
= $(A + \overline{A}) \cdot (A + B)$ Distributive law
= $1 \cdot (A + B)$ (since $A + \overline{A} = 1$)
= $A + B$ Proved

5(b) is the dual of 5(a).

This theorem is also called second absorption theorem. Corollary of this theorem is given as $\overline{A} + A \cdot B = A + B$ $\overline{A} \cdot (A + B) = \overline{A} \cdot B$

De Morgan's Theorem: De Morgan, a logician gave two very important theorems which are used in Boolean algebra, which is stated as:

The complement of a product of two variables is equal to the sum of the complemented variables. In equation form it is given as:

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

The dual of this theorem is given in equation form as:

Theorem 6(b) $\overline{A + B} = \overline{A} \cdot \overline{B}$

Which is stated as: The complement of a sum of two variables is equal to the product of the complemented variables.

Proof: Theorem 6(a) is illustrated in the truth tables (3.9) as the columns 4 and 7 of this table are identical. Theorem 6(b) is the dual of 6(a) so need not to be proved.

Table 3.9

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

The De Morgan's theorems hold good for n variables given below:

$$\overline{A_1 \cdot A_2 \cdot A_3 \cdots A_n} = \overline{A_1} + \overline{A_2} + \overline{A_3} + \cdots\cdots + \overline{A_n}$$

$$\overline{A_1 + A_2 + A_3 + \cdots + A_n} = \overline{A_1} \cdot \overline{A_2} \cdot \overline{A_3} \cdots\cdots \overline{A_n}$$

Example 3.1: Using the theorems of Boolean algebra, prove the following identities:

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

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

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

Solution: (i) L.H.S.

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

$= (A+B)\cdot(A+\overline{A}\cdot\overline{B})\cdot C + \overline{\overline{A}} + (\overline{B+\overline{C}}) + \overline{A}\cdot B + A\cdot B\cdot C$ (Demorgan's law)

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

$= (A\cdot(A+\overline{B}) + B\cdot(A+\overline{B}))\cdot C + A + \overline{B}\cdot C + \overline{A}\cdot B + A\cdot B\cdot C$ (Distributive law and Demorgan's law)

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

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

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

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

(ii) L.H.S.

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

= $A \,.\, A\,(C + \overline{C}\,(\overline{A} + B))(B + C)$ (Absorption law)

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

= $A\,(C + B)(B + C)$ (Absorption law)

= $A\,.(B + C)$ (Since A.A = A)

= A.B + A.C

= R.H.S.

(iii) L.H.S.

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

= $A \cdot \overline{B} \cdot 1 + 1 \cdot B \cdot \overline{C} + \overline{A} \cdot 1 \cdot C$ (Since A.1 = A)

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

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

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

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

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

= R.H.S.

3.6 Venn Diagram: The postulates and theorems of Boolean algebra may be illustrated by the pictorial model known as Venn diagram. The Venn diagram consists of a rectangle inside which a number of circles intersecting each other are drawn. One circle corresponds to one variable. Figure 3.6 shows the Venn diagram for one variable. The area inside the circle represents the variable itself (i.e. the shaded area in figure 3.6a represents the variable A); and the area outside the circle represents the complement of that variable (shaded area in figure 3.6 b shows $\overline{A}$).

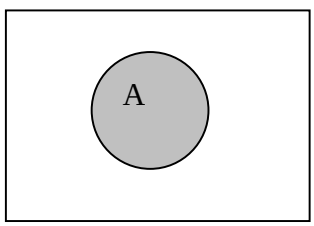


Figure 3.6 a

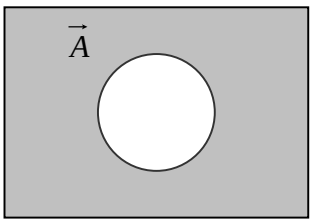


Figure 3.6 b

Figure 3.7 shows the Venn diagram for two variables consisting of a rectangular inside which two circles intersecting each other are drawn. The area common between the two circles shows the intersection of two variables represented by '.' sign. The shaded area in figure 3.7a represents the intersection of two areas $A \cdot B$. The union or the sum of two areas represents the OR operation of two variables. A+B is represented by the shaded area in figure 3.7b.

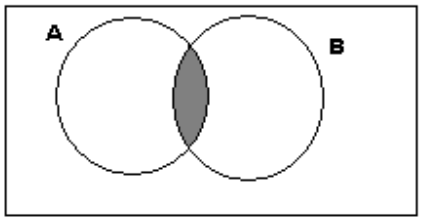


Fig. 3.7a

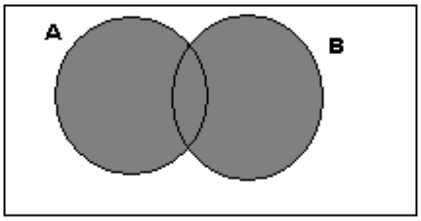


Fig. 3.7 b

The shaded area represented in figure 3.8 shows $\overline{A} \cdot B$ as it the intersection of $\overline{A}$ (outside area of A shown in figure 3.9 a) and B (area of the circle B shown in figure 3.9b).

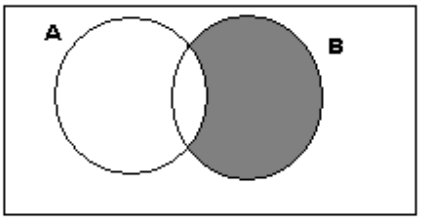


Fig. 3.8

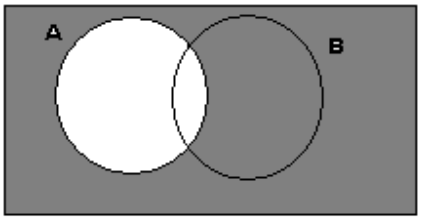


Fig. 3.9 a

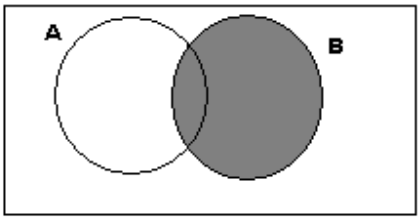


Fig. 3.9 b

The Venn diagram for $\overline{A} \cdot \overline{B}$ is shown in figure 3.10. It is the intersection of outside regions of the circles A and B (ref. fig.3.11 a & b).

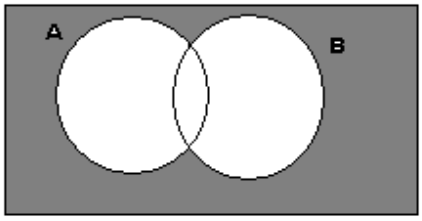


Fig. 3.10

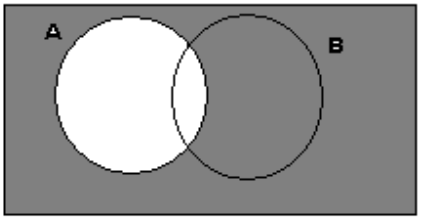


Fig. 3.11 a

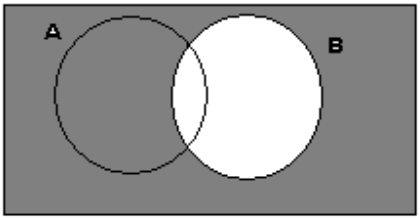


Fig. 3.11 b

Now the Boolean identity $A \cdot (B + C) = A \cdot B + A \cdot C$ will be illustrated using the Venn diagram. Left hand side of this identity is shown in figure 3.12 which is the intersection of the A and (B+C).

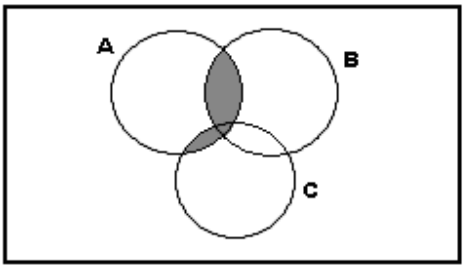


Fig. 3.12

The right hand side of the identity is shown in figure 3.13.

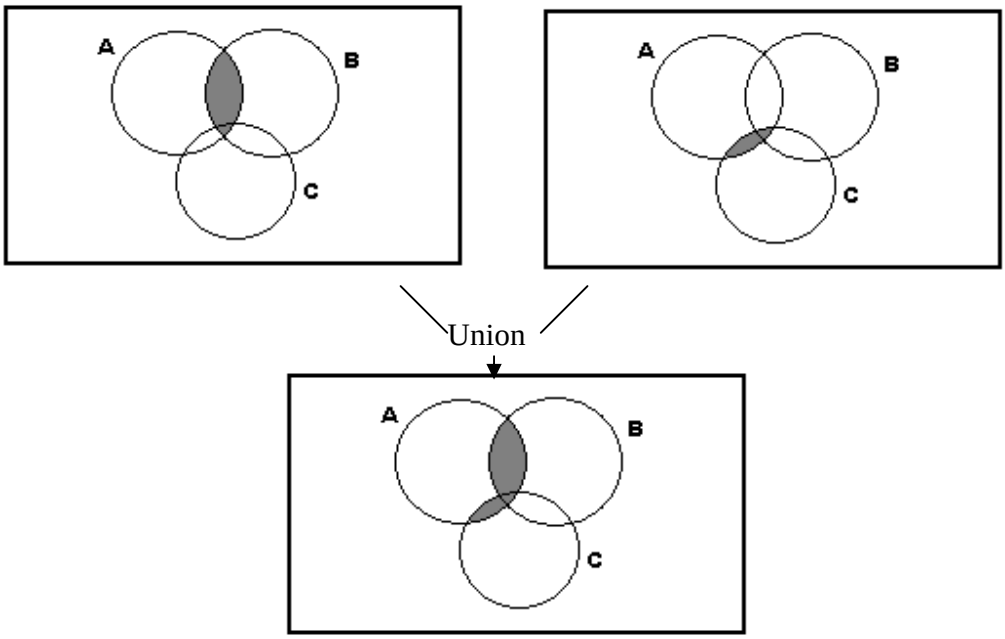


Fig. 3.13

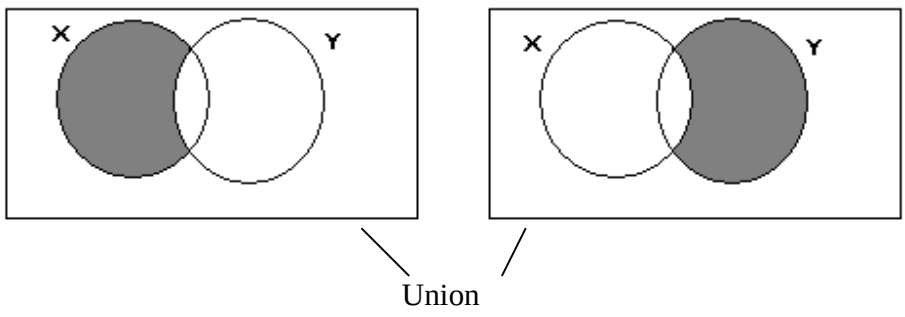
The shaded areas of figures 3.12 and 3.13 are identical which shows the given identity is proved.

Example 3.2: Using the Venn diagram prove the following identities:

(i) $X \cdot \overline{Y} + \overline{X} \cdot Y = (X + Y) \cdot (\overline{X} + \overline{Y})$

(ii) $\overline{X + Y} = \overline{X} \cdot \overline{Y}$

Solution: (i) Figure 3.14 shows the Venn diagram of left hand side of the given identity and the Venn diagram for right hand side is shown in figure 3.15.



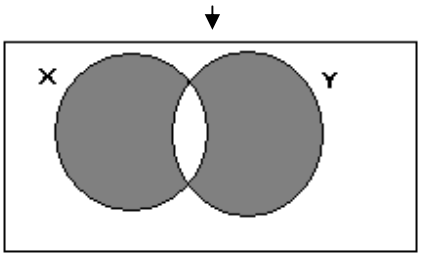


Fig. 3.14

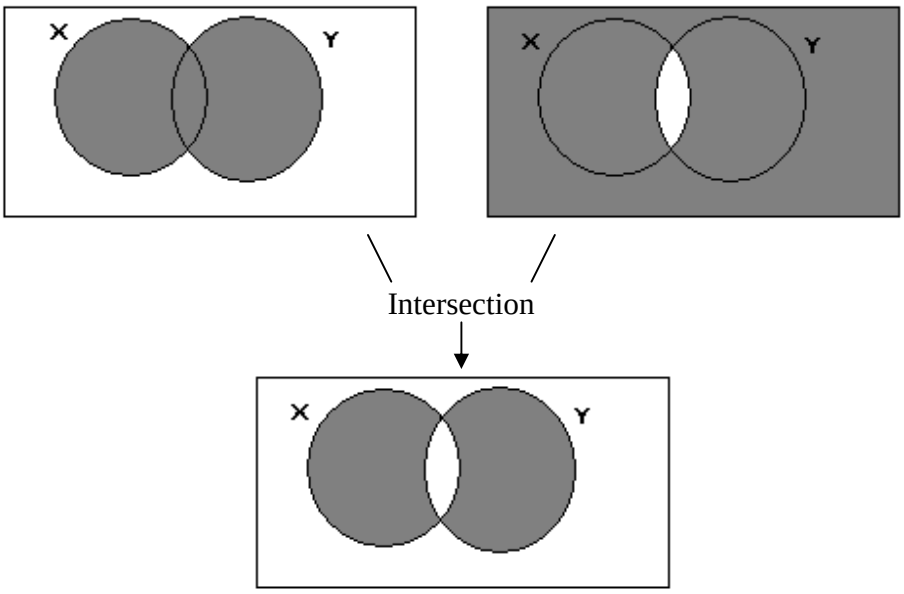
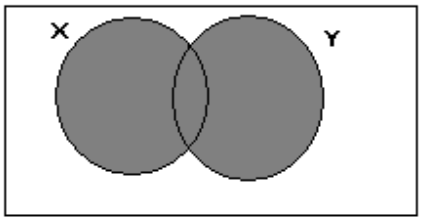


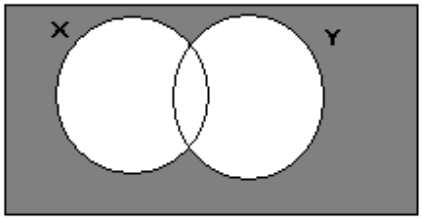
Fig. 3.15

From these two figures it is clear that the shaded areas are identical, so the Boolean identity is proved.

(ii) Figures 3.16 and 3.17 respectively show the Venn diagrams of left hand side and right hand side of the given identity $\overline{X+Y} = \overline{X} \cdot \overline{Y}$.



Shaded area is X + Y



Shaded area is $\overline{X+Y}$

Fig. 3.16

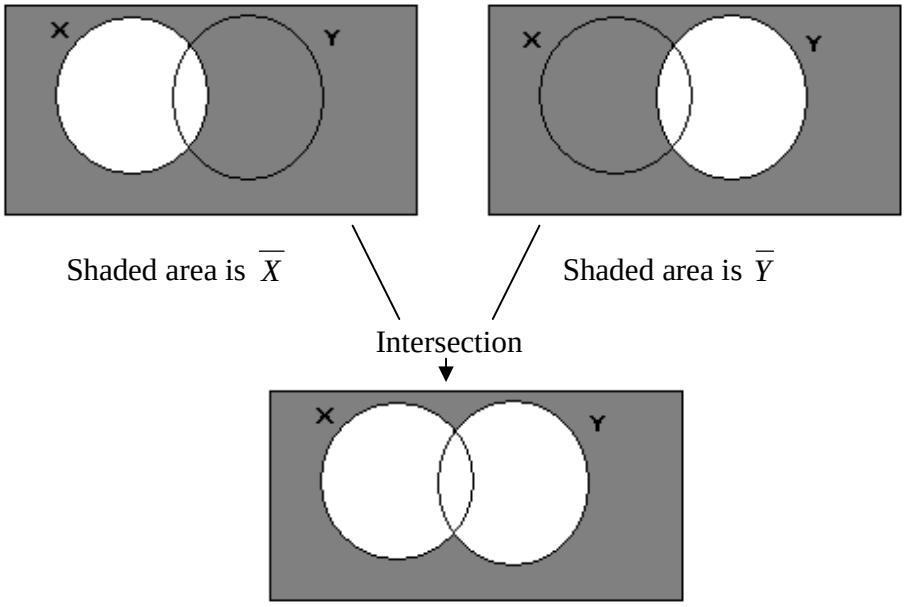


Fig. 3.17

The shaded areas of the Venn diagram shown above are identical, which indicate the given identity is proved.

3.7 Truth Table: Truth table gives the values of the output variables for all the possible combinations of the input variables. Consider a Boolean function F = A . B of the logic AND operation. In this function A & B are the two input independent variables and will have 2^N ($2^2 = 4$) possible input combinations, where N is the number of input variables. Each input combination gives rise an output. All possible values of input and output variables listed in the form of a table is known as truth table. The truth table of this AND operation is shown in table 3.10.

Table 3.10

Input Variables		output
A	B	F= $A \cdot B$
0	0	0
0	1	1
1	0	1
1	1	1

To draw the truth table of a given function following procedure is followed:

A table is drawn having one column each for independent variables and one for dependent variable. The entries of all the possible values of the independent variables are made in the different horizontal rows in binary progression. The number of horizontal rows will depend on the number of independent variables given by 2^N , where N is equal to the number of independent variables. The values in the dependent function are filled in the table after calculating it from the given function.

Example 3.3: Draw the truth table of a Boolean function given below:

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

Solution: The given expression has the three independent variables, so it will have 8 different horizontal rows (as 2^3 = 8). Putting all possible values of the independent variables in the binary progression and evaluated values of the dependent variable F from the given expression $F = \overline{A} \cdot B + C$, the required truth table is obtained which is shown below (table 3.11).

Table 3.11

Input variables			Dependent variable
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

3.8 Canonical Forms for Boolean Function: There are two basic forms of Boolean function corresponding to a given truth table. These forms are Canonical SP form (Sum of Products) and Canonical PS form (Product of Sums).

3.8.1 Canonical SP (or SOP) Form: The canonical SP form for Boolean function of the truth table are obtained by summing (ORing) the product (ANDed) terms corresponding to the 1's entry in the output column of the truth table. The product terms also known as minterms are formed by ANDing the complemented and un-complemented variables in such a way that the complement of variable is taken for the 0's entry to the input variable and the variable itself is taken for 1's entry in the input variable. The minterms (possible ANDed terms or products) for the two variables A and B are shown in table 3.12.

Table 3.12

Input variables		Minterms
A	B	
0	0	$\overline{A} \cdot \overline{B}$
0	1	$\overline{A} \cdot B$
1	0	$A \cdot \overline{B}$
1	1	$A \cdot B$

Similarly, table 3.13 shows the minterms for three variables truth table.

Table 3.13

Input variables A	B	C	Minterms	Minterm notation
0	0	0	$\overline{A} \cdot \overline{B} \cdot \overline{C}$	m_0
0	0	1	$\overline{A} \cdot \overline{B} \cdot C$	m_1
0	1	0	$\overline{A} \cdot B \cdot \overline{C}$	m_2
0	1	1	$\overline{A} \cdot B \cdot C$	m_3
1	0	0	$A \cdot \overline{B} \cdot \overline{C}$	m_4
1	0	1	$A \cdot \overline{B} \cdot C$	m_5
1	1	0	$A \cdot B \cdot \overline{C}$	m_6
1	1	1	$A \cdot B \cdot C$	m_7

In general, there will be 2^N different minterms for N variables. Further it is also convenient to refer the minterms in the form of minterm notations as shown in table 3.13. The subscript to *m* corresponds to the decimal equivalent of the binary number formed by the independent variables. For example the minterm for the minterm notation m_6 is $A \cdot B \cdot \overline{C}$, as 110 is the binary equivalent of the decimal number 6 (subscript of *m*). There will be 16 minterms for four variable truth table from m_0 to m_{15}.

The required canonical SP form of the Boolean expression corresponding to a given truth table is finally obtained by ORing the minterms that produce 1 output in the truth table.

Consider a truth table (table 3.14) whose Boolean expression in SP form is to be obtained.

Table 3.14

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The minterms corresponding to the input conditions that result 1 at the output in the above table are $\overline{A}\cdot B\cdot\overline{C}$, $A\cdot\overline{B}\cdot C$, $A\cdot B\cdot\overline{C}$, $A\cdot B\cdot C$. The required Boolean expression is obtained by ORing these minterms as:

$$F=\overline{A}\cdot B\cdot\overline{C}+A\cdot\overline{B}\cdot C+A\cdot B\cdot\overline{C}+A\cdot B\cdot C \quad \text{------ (3.1)}$$

In minterm notation this expression is written as:

$$F=m_2+m_5+m_6+m_7$$
$$=\sum(m_2,m_5,m_6,m_7) \quad \text{------ (3.2)}$$

Or simply

$$F=\sum(2,5,6,7) \quad \text{------ (3.3)}$$

The decimal numbers in the above expression indicate the subscript of the minterm notation.

3.8.2 Canonical PS (or POS) Form: The Boolean expression in canonical PS form of a truth table can be obtained by taking the product (ANDing) of the sum (ORed) terms corresponding to the 0's entry in the output column of the truth table. The ORed terms are called as maxterms. The maxterms are formed by ORing the complemented and Un-complemented variables present in a row of the truth table in such a way that the complement of variable is taken for the 1's entry to the input variable and the variable itself is taken for 0's entry in the input variable.

The maxterms with their notations for three variables are shown in table 3.15.

Table 3.15

Input variables			Maxterms	Maxterm notation
A	B	C		
0	0	0	$A+B+C$	M_0
0	0	1	$A+B+\overline{C}$	M_1
0	1	0	$A+\overline{B}+C$	M_2
0	1	1	$A+\overline{B}+\overline{C}$	M_3
1	0	0	$\overline{A}+B+C$	M_4
1	0	1	$\overline{A}+B+\overline{C}$	M_5
1	1	0	$\overline{A}+\overline{B}+C$	M_6
1	1	1	$\overline{A}+\overline{B}+\overline{C}$	M_7

The subscript to *M* corresponds to the decimal equivalent of the binary number formed by the independent variables. For example the minterm for the minterm notation M_5 is $\overline{A}+B+\overline{C}$, as 101 is the binary equivalent of the decimal number 5 (subscript of *M*). There will be 16 minterms for four variable truth table from M_0 to M_{15} .

The Boolean expression in canonical PS form of the truth table given in table 3.14 is obtained by ANDing the maxterms that produces o output in the truth table.

The maxterms corresponding to the input conditions that result 0 at the output in the table 3.14 are $A+B+C$, $A+B+\overline{C}$, $A+\overline{B}+\overline{C}$, $\overline{A}+B+C$. The required Boolean expression is therefore obtained as:

$$F(\text{in PS form}) = (A+B+C)\cdot(A+B+\overline{C})\cdot(A+\overline{B}+\overline{C})\cdot(\overline{A}+B+C) \quad \text{------ (3.4)}$$

In maxterm notation this expression is written as:

$$F(\text{in PS form}) = M_0.M_1.M_3.M_4 \quad \text{------ (3.5)}$$

$$= \prod(M_0, M_1, M_3, M_4)$$

Or Simply

$$\text{F(in PS form)} = \prod(0,1,3,4) \quad \text{------ (3.6)}$$

It is important to note that equations (3.1) and (3.4) represent the Boolean expressions in canonical SP and canonical PS forms respectively of the same truth table (table 3.14). These two expressions must be equivalent. The equivalence may be shown by considering directly the expression for $\overline{F}$ in SP form from table 3.14 as:

$$\overline{F} \text{ (in SP form)} = \sum m_0, m_1, m_3, m_4$$

or
$$\overline{F} = \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot C + A\cdot\overline{B}\cdot\overline{C} \quad \text{----- (3.7)}$$

Taking the complement on both side of this equation we get:

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

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

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

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

------ (3.8)

The equation (3.8) is the same as equation (3.4). This method gives us a method of getting the **PS form** from **SP form** of the same Boolean expression. Similarly, by taking $\overline{F}$ (in PS form) and then complementing on both sides the conversion of PS form to SP form is obtained.

Thus $F = \prod(0,1,3,4)$ in PS form is converted to

$F = \sum(2,5,6,7)$ in SP form.

------ (3.9)

From the above discussion it is concluded that the two standard forms of Boolean function may be obtained from a given truth table. These functions will not directly be realized using the basic gates. These functions are to be minimized using the theorem of Boolean algebra or other methods, which will be discussed in later chapter. It is clear from the equation (3.9) that the conversion of one standard form to other is obtained by interchanging $\sum$ and $\prod$ and having the numbers missing in the original form.

Example 3.4: Express the following function into canonical form:

(i) $F = (X + Y + Z) \cdot (X + \overline{Y}) \cdot (Y + \overline{Z})$

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

Solution: (i) $F = (X + Y + Z) \cdot (X + \overline{Y}) \cdot (Y + \overline{Z})$

Expending the terms we get:

$$\begin{aligned} F &= (X + Y + Z) \cdot (X + \overline{Y} + Z \cdot \overline{Z}) \cdot (X \cdot \overline{X} + Y + \overline{Z}) \\ &= (X + Y + Z) \cdot (X + \overline{Y} + Z) \cdot (X + \overline{Y} + \overline{Z}) \cdot (X + Y + \overline{Z}) \cdot (\overline{X} + Y + \overline{Z}) \\ &= M_0 M_2 M_3 M_1 M_5 \\ &= \prod (M_0, M_1, M_2, M_3, M_5) \end{aligned}$$

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

Expending the terms we get:

$$\begin{aligned} G &= A \cdot B \cdot C + \overline{A} \cdot B \cdot (C + \overline{C}) + (A + \overline{A}) \cdot \overline{B} \cdot C \\ &= A \cdot B \cdot C + \overline{A} \cdot B \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + \overline{A} \cdot \overline{B} \cdot C \\ &= m_7 + m_3 + m_2 + m_5 + m_1 \\ &= m_1 + m_2 + m_3 + m_5 + m_7 \\ &= \sum (m_1, m_2, m_3, m_5, m_7) \end{aligned}$$

Example 3.5: Express the following Boolean function in PS form.

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

Solution: We have $F = A \cdot \overline{B} + \overline{B} \cdot C$

Applying the distributive law, we get:

$$\begin{aligned} F &= (A \cdot \overline{B} + \overline{B}) \cdot (A \cdot \overline{B} + C) \\ &= \overline{B} \cdot (A + C) \cdot (\overline{B} + C) \\ &= (A \cdot \overline{A} + \overline{B}) \cdot (A + B \cdot \overline{B} + C) \cdot (A \cdot \overline{A} + \overline{B} + C) \\ &= (A + \overline{B}) \cdot (\overline{A} + \overline{B}) \cdot (A + B + C) \cdot (A + \overline{B} + C) \cdot (A + \overline{B} + C) \cdot (\overline{A} + \overline{B} + C) \\ &= (A + \overline{B} + C \cdot \overline{C}) \cdot (\overline{A} + \overline{B} + C \cdot \overline{C}) \cdot (A + B + C) \cdot (A + \overline{B} + C) \cdot (A + \overline{B} + C) \cdot (\overline{A} + \overline{B} + C) \end{aligned}$$

$$= (A+\overline{B}+C)\cdot(A+\overline{B}+\overline{C})\cdot(\overline{A}+\overline{B}+C)\cdot(\overline{A}+\overline{B}+\overline{C})\cdot(A+B+C)\cdot(A+\overline{B}+C)\cdot(\overline{A}+\overline{B}+C)$$
$$= (A+\overline{B}+C)\cdot(A+\overline{B}+\overline{C})\cdot(\overline{A}+\overline{B}+C)\cdot(\overline{A}+\overline{B}+\overline{C})\cdot(A+B+C)$$
$$= M_0 M_2 M_3 M_6 M_7$$
$$= \prod(M_0, M_2, M_3, M_6, M_7)$$

Example 3.6: Express the following Boolean function in SP form.

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

Solution: The given Boolean function is:

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

Expending the terms we get:

$$= A\cdot\overline{B}\cdot(C+\overline{C})+(A+\overline{A})\cdot C$$
$$= A\cdot\overline{B}\cdot C + A\cdot\overline{B}\cdot\overline{C} + A\cdot C + \overline{A}\cdot C$$
$$= A\cdot\overline{B}\cdot C + A\cdot\overline{B}\cdot\overline{C} + A\cdot(B+\overline{B})\cdot C + \overline{A}\cdot(B+\overline{B})\cdot C$$
$$= A\cdot\overline{B}\cdot C + A\cdot\overline{B}\cdot\overline{C} + A\cdot B\cdot C + A\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot C + \overline{A}\cdot\overline{B}\cdot C$$
$$= A\cdot\overline{B}\cdot C + A\cdot\overline{B}\cdot\overline{C} + A\cdot B\cdot C + A\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot C + \overline{A}\cdot\overline{B}\cdot C$$
$$= A\cdot\overline{B}\cdot C + A\cdot\overline{B}\cdot\overline{C} + A\cdot B\cdot C + \overline{A}\cdot B\cdot C + \overline{A}\cdot\overline{B}\cdot C$$
$$= m_5 + m_4 + m_7 + m_3 + m_1$$
$$= m_1 + m_3 + m_4 + m_5 + m_7$$
$$= \sum(m_1, m_3, m_4, m_5, m_7)$$

Example 3.7: Using the theorems of Boolean algebra, reduce the following functions:

(i) $F_1(A, B, C, D) = \sum(0,1,2,6,7,14,15)$

(ii) $F_2(X, Y, Z, W) = \sum(2,3,6,7,13,14,15)$

Solution:

(i) $F_1(A, B, C, D) = \sum(0,1,2,6,7,14,15)$
$$= \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\overline{D} + \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D + \overline{A}\cdot\overline{B}\cdot C\cdot\overline{D} + \overline{A}\cdot\overline{B}\cdot C\cdot D + \overline{A}\cdot B\cdot C\cdot\overline{D} + \overline{A}\cdot B\cdot C\cdot D + A\cdot B\cdot C\cdot\overline{D} + A\cdot B\cdot C\cdot D$$
$$= \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot(\overline{D}+D) + \overline{A}\cdot\overline{B}\cdot C\cdot(\overline{D}+D) + \overline{A}\cdot B\cdot C\cdot(\overline{D}+D) + A\cdot B\cdot C\cdot(\overline{D}+D)$$
$$= \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot C + A\cdot B\cdot C$$
$$= \overline{A}\cdot\overline{B}\cdot(\overline{C}+C) + B\cdot C\cdot(\overline{A}+A)$$
$$= \overline{A}\cdot\overline{B} + B\cdot C$$

(ii) $F_2(X, Y, Z, W) = \sum(2,3,6,7,13,14,15)$
$$= \overline{X}\cdot\overline{Y}\cdot Z\cdot\overline{W} + \overline{X}\cdot\overline{Y}\cdot Z\cdot W + \overline{X}\cdot Y\cdot Z\cdot\overline{W} + \overline{X}\cdot Y\cdot Z\cdot W + X\cdot Y\cdot\overline{Z}\cdot W + X\cdot Y\cdot Z\cdot\overline{W} + X\cdot Y\cdot Z\cdot W$$
$$= \overline{X}\cdot\overline{Y}\cdot Z\cdot(\overline{W}+W) + \overline{X}\cdot Y\cdot Z\cdot(\overline{W}+W) + X\cdot Y\cdot\overline{Z}\cdot W + X\cdot Y\cdot Z\cdot(\overline{W}+W)$$
$$= \overline{X}\cdot\overline{Y}\cdot Z + \overline{X}\cdot Y\cdot Z + X\cdot Y\cdot\overline{Z}\cdot W + X\cdot Y\cdot Z$$
$$= \overline{X}\cdot Z\cdot(\overline{Y}+Y) + X\cdot Y\cdot(\overline{Z}\cdot W + Z)$$

$= \overline{X} \cdot Z + X \cdot Y \cdot (W + Z)$ (Since $Z + \overline{Z} \cdot W = Z + W$)

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

$= Z(\overline{X} + X \cdot Y) + X \cdot Y \cdot W$

$= Z(\overline{X} + Y) + X \cdot Y \cdot W$

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

Example 3.8: Using the theorems of Boolean algebra, reduce the following functions:

(i) $F_1(a, b, c) = \prod (0,1,4,5,7)$

(ii) $F_2(a, b, c, d) = \sum (3,5,7,11,13,14,15)$

Solution:

(i) $F_1(a, b, c) = \prod (0,1,4,5,7)$

$= (a + b + c) \cdot (a + b + \overline{c}) \cdot (\overline{a} + b + c) \cdot (\overline{a} + b + \overline{c}) \cdot (\overline{a} + \overline{b} + \overline{c})$

$= (a + b + c \cdot \overline{c}) \cdot (\overline{a} + b + c \cdot \overline{c})(\overline{a} + \overline{b} + \overline{c})$ (Distributive law)

$= (a + b) \cdot (\overline{a} + b) \cdot (\overline{a} + \overline{b} + \overline{c})$ (Since $c \cdot \overline{c} = 0$)

$= (a + b) \cdot (\overline{a} + b \cdot (\overline{b} + \overline{c}))$ (Distributive law)

$= (a + b) \cdot (\overline{a} + b \cdot \overline{c})$

$= (a + b) \cdot (\overline{a} + b) \cdot (\overline{a} + \overline{c})$

$= (b + a \cdot \overline{a}) \cdot (\overline{a} + \overline{c})$

$= b \cdot (\overline{a} + \overline{c})$

(ii) $F_2(a, b, c, d) = \sum (3,5,7,11,13,14,15)$

$= (a+b+\overline{c}+\overline{d}) \cdot (a+\overline{b}+c+\overline{d}) \cdot (a+\overline{b}+\overline{c}+\overline{d}) \cdot (\overline{a}+b+\overline{c}+\overline{d}) \cdot (\overline{a}+\overline{b}+c+\overline{d}) \cdot (\overline{a}+\overline{b}+\overline{c}+d) \cdot (\overline{a}+\overline{b}+\overline{c}+\overline{d})$

$= (a+b+\overline{c}+\overline{d}) \cdot (\overline{a}+b+\overline{c}+\overline{d}) \cdot (a+\overline{b}+c+\overline{d}) \cdot (a+\overline{b}+\overline{c}+\overline{d}) \cdot (\overline{a}+\overline{b}+c+\overline{d}) \cdot (\overline{a}+\overline{b}+\overline{c}+d) \cdot (\overline{a}+\overline{b}+\overline{c}+\overline{d})$

$= (b+\overline{c}+\overline{d}+a \cdot \overline{a}) \cdot (a+\overline{b}+\overline{d}+c \cdot \overline{c}) \cdot (\overline{a}+\overline{b}+c+\overline{d}) \cdot (\overline{a}+\overline{b}+\overline{c}+d \cdot \overline{d})$

$= (b+\overline{c}+\overline{d}) \cdot (a+\overline{b}+\overline{d}) \cdot (\overline{a}+\overline{b}+c+\overline{d}) \cdot (\overline{a}+\overline{b}+\overline{c})$

$= (b + \overline{c} + \overline{d}) \cdot (a + \overline{b} + \overline{d}) \cdot (\overline{a} + \overline{b} + \overline{c} \cdot (c + \overline{d}))$ (Distributive law)

$= (b + \overline{c} + \overline{d}) \cdot (a + \overline{b} + \overline{d}) \cdot (\overline{a} + \overline{b} + \overline{c} \cdot \overline{d})$ (Since $\overline{c} \cdot (c + \overline{d}) = \overline{c} \cdot \overline{d}$)

$= (b + \overline{c} + \overline{d}) \cdot (a + \overline{b} + \overline{d}) \cdot (\overline{a} + \overline{b} + \overline{c}) \cdot (\overline{a} + \overline{b} + \overline{d})$ (Distributive law)

$= (b + \overline{c} + \overline{d}) \cdot (\overline{a} + \overline{b} + \overline{c}) \cdot (\overline{b} + \overline{d} + a \cdot \overline{a})$

$= (b + \overline{c} + \overline{d}) \cdot (\overline{a} + \overline{b} + \overline{c}) \cdot (\overline{b} + \overline{d})$

$= (\overline{d} + \overline{b} \cdot (b + \overline{c})) \cdot (\overline{a} + \overline{b} + \overline{c}) \cdot (\overline{b} + \overline{d})$

$= (\overline{d} + \overline{b} \cdot \overline{c}) \cdot (\overline{a} + \overline{b} + \overline{c}) \cdot (\overline{b} + \overline{d})$

$= (\overline{d} + \overline{b}) \cdot (\overline{d} + \overline{c}) \cdot (\overline{a} + \overline{b} + \overline{c}) \cdot (\overline{b} + \overline{d})$

$= (\overline{d} + \overline{b}) \cdot (\overline{d} + \overline{c}) \cdot (\overline{a} + \overline{b} + \overline{c})$ (Since $a \cdot a = a$)

3.9 Realization of Boolean Function Using Gates: The Boolean functions discussed above nay be realized using AND, OR and NOT gates. Consider a Boolean function:

$$f = \bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + \bar{a} \cdot b \cdot c + a \cdot b \cdot c \qquad \text{------ (3.10)}$$

Realization of this Boolean function is shown in figure (3.16).

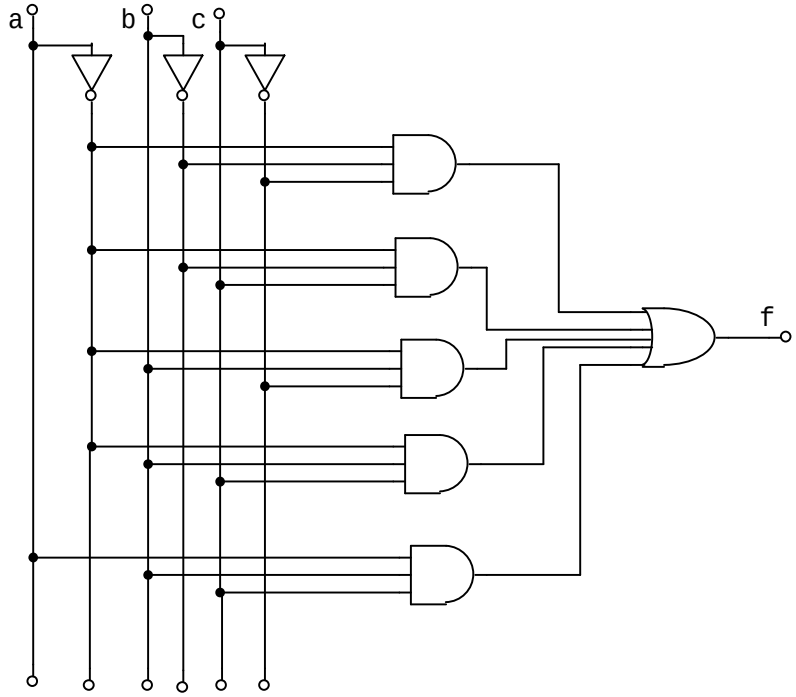


Figure 3.16

AND, OR and Not gates are called as universal gates because any Boolean function can be realized using these gate. It is further noted that this circuit needs 9 gates (three NOT gates, five 3 input AND gates and one 5 input OR gates) for its realization. Use of Boolean algebra help in getting the minimal Boolean function as given below:

$$\begin{aligned} f &= \bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + \bar{a} \cdot b \cdot c + a \cdot b \cdot c \\ &= \bar{a} \cdot \bar{b} \cdot (\bar{c} + c) + \bar{a} \cdot b \cdot (\bar{c} + c) + a \cdot b \cdot c \\ &= \bar{a} \cdot \bar{b} + \bar{a} \cdot b + a \cdot b \cdot c \\ &= \bar{a} \cdot (\bar{b} + b) + a \cdot b \cdot c \\ &= \bar{a} + a \cdot b \cdot c \\ &= \bar{a} + b \cdot c \qquad \text{(Since } \bar{a} + a \cdot b = \bar{a} + b \text{)} \end{aligned}$$

This reduced Boolean function need to have only three gates (one NOT gate, one 2 input AND gate and one 2 input OR gate) for its realization as shown in figure (3.17).

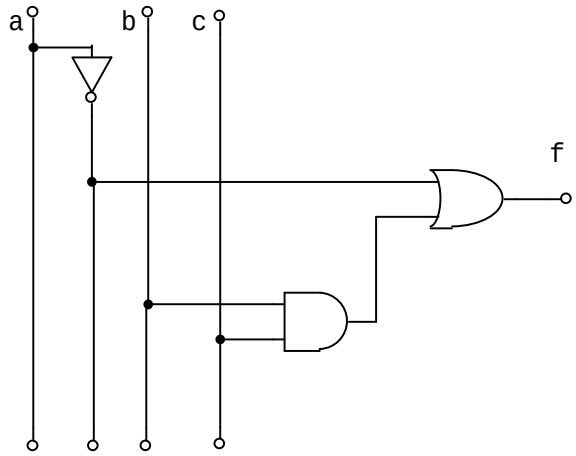


Figure 3.17

It is clear form the above discussion that the minimized Boolean function needs less number of gates for the realization. This signifies the importance of Boolean algebra. The minimized functions not only help in reducing the number of gates but also increase the reliability of the circuit and speed of operation.

3.10 Other Logic Operations and Logic Gates: The use of three Boolean operators namely AND, OR and NOT has been discussed in the forgoing sections of this chapter. The gates for these operators are known as universal gates as all the Boolean functions can be realized using these gates. Besides these operators there are other operators which will now be discussed. For two input variables two truth tables are known; one for AND operation and other for OR operation. However, for two variables there may be 16 ($= 2^{2^2}$) different truth tables. In general 2^{2^N} truth tables may be constructed, where N is the number of input variables. So for two variables 16 different Boolean operators or functions may be defined. The truth table of 16 functions for two variables *A* and *B* are shown in table 3.16.

Table 3.16

A	B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
										$\overline{f_7}$	$\overline{f_6}$	$\overline{f_5}$	$\overline{f_4}$	$\overline{f_3}$	$\overline{f_2}$	$\overline{f_1}$	$\overline{f_0}$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Out of the 16 function listed in table 3.16, eight functions are basically the complementation of other eight functions. These 16 functions are expressed algebraically in table 3.17.

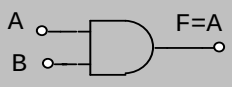
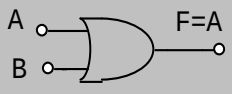
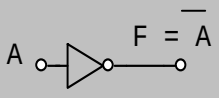
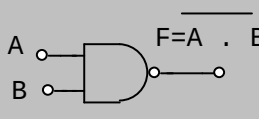
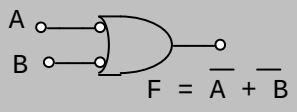
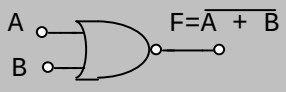
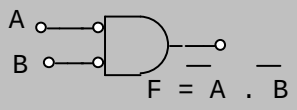
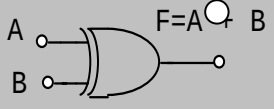
Table 3.17

Function	Operator	Symbol	Comments
$f_0 = 0$	NULL		Always 0
$f_1 = A \cdot B$	AND	$A \cdot B$	A and B
$f_2 = A \cdot \overline{B}$	INHIBITION	A / B	A but not B
$f_3 = A$			Always A
$f_4 = \overline{A} \cdot B$	INHIBITION	B / A	B but not A
$f_5 = B$			Always B
$f_6 = A\overline{B} + \overline{A}B$	EXCLUSIVE –OR	$A \oplus B$	A or B but not both
$f_7 = A + B$	OR	$A + B$	A or B
$f_8 = (\overline{A + B})$	NOR	$A \downarrow B$	Not (A or B)
$f_9 = \overline{A} \cdot \overline{B} + A \cdot B$	EQUIVALENCE	$A \odot B$	A equals B
$f_{10} = \overline{B}$	COMPLEMENT	$\overline{B}$	Not B
$f_{11} = A + \overline{B}$	IMPLECATION	$B \supset A$	If B then A
$f_{12} = \overline{A}$	COMPLEMENT	$\overline{A}$	Not A
$f_{13} = \overline{A} + B$	IMPLECATION	$A \supset B$	If A then B
$f_{14} = \overline{A \cdot B}$	NAND	$A \uparrow B$	Not (A and B)
$f_{15} = 1$	IDENTITY		Always 1

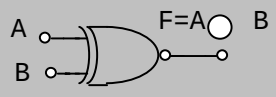
The Null, Identity, A and B functions are trivial, since Null and Identity functions always produce 0 and 1 respectively, and A and B functions always produce the input itself. We are already familiar with AND, OR and Complement (NOT or $\overline{A}$, $\overline{B}$) functions. The other functions are NAND, NOR, Exclusive OR, Equivalence, Inhibition and Implication. The NAND function is the complement of AND and it is also known as 'NOT of AND'. Similarly, NOR is the complement of OR which is also known as 'NOT of OR'. The Exclusive OR (also known as XOR or EOR) is similar to OR but produce an output 1 when either of two inputs is 1 but not both. In other words XOR produces output 1 if the inputs are dissimilar. The equivalence is the complement of XOR, hence it is also known as Exclusive-NOR (or XNOR). This function produces output as 1 when both the inputs are equal. The logicians may use the functions Implication and Inhibition but are seldom used in computer logic, since they are not commutative.

The electronic circuits which can perform the operation or functions discussed above are known as gates. The symbolic representation and truth tables for these gates are shown in table 3.18.

Table 3.18

Logic gates	Symbols	De-Morgan's representation	Truth Table
AND			A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR			A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
INVERTER			A | F 0 | 1 1 | 0
NAND			A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR			A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Ex-OR XOR			A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Ex-NOR
or
equivalence



A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

The logic gates for Inhibition and Implication operators or functions are not designed since these functions are not commutative. In fact, the binary logic gates are designed only for the operators which satisfy the following factors:

(i) The operators should be commutative and associative.

(ii) The gates for the operators itself or in association with other gates should be able to realize all the Boolean functions.

(iii) It should be feasible to design the gates and the cost for making the gates should not be very large.

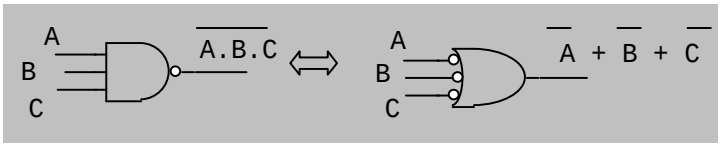
The exclusive-OR and Equivalence operators satisfy the first two properties but relatively more expensive to construct these gates for more than two inputs. However, NAND and NOR operators satisfy all the above properties so gates for these operators are constructed and commonly used. These gates also called the universal gates are used in preference to AND, OR, NOT gates.

It is necessary to discuss that the NAND/NOR are defined for only two inputs as these operators are not associative, i.e.

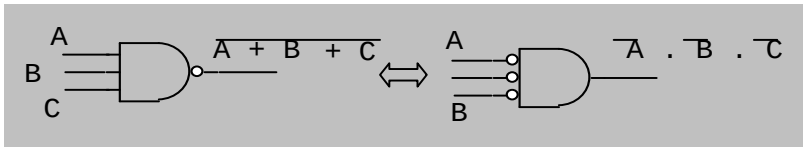
$$(A \uparrow B) \uparrow C \neq A \uparrow (B \uparrow C)$$

and $$(A \downarrow B) \downarrow C \neq A \downarrow (B \uparrow C)$$

So multiple input NAND/ NOR gates are defined as the complement of multiple input AND/OR gates as shown in figure 3.18.



NAND



NOR

Figure 3.18

Further AND, OR, NOT gates can be implemented with NAND's or NOR's alone as follows:

(i) NAND as NOT : $\overline{A} = \overline{A \cdot A}$

also $\overline{A} = \overline{A \cdot 1}$ illustrated as shown in figure 3.19(a):

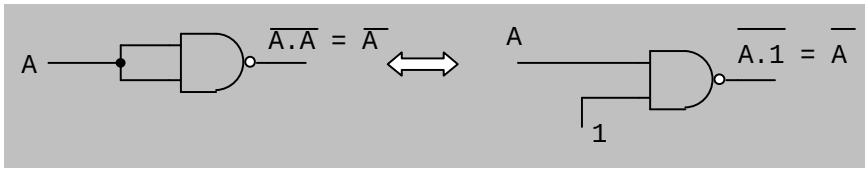


Fig. 3.19(a)

(ii) NAND as AND: $A \cdot B = \overline{\overline{A \cdot B}}$ = Complement of $\overline{A \cdot B}$

which may be illustrated as shown in figure 3.19(b).

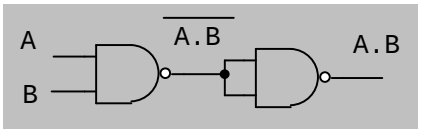


Fig. 3.19(b)

(iii) NAND as OR:

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

which may be illustrated as shown in figure 3.19 (c).

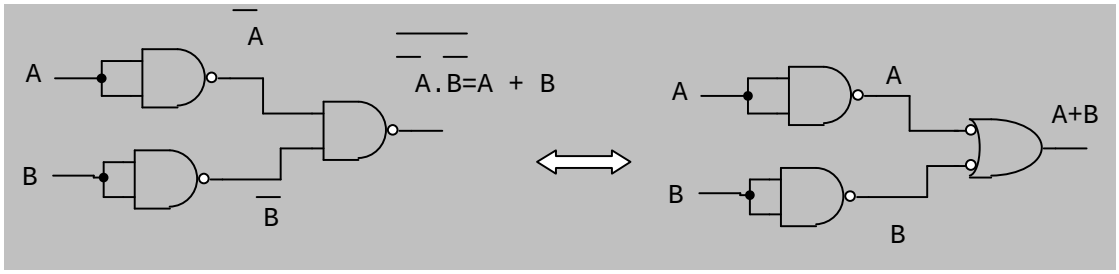


Fig. 3.19(c)

(i) NOR as NOT : $\overline{A} = \overline{A + A}$

also $\overline{A} = \overline{A + 0}$ illustrated as shown in figure 3.20(a):

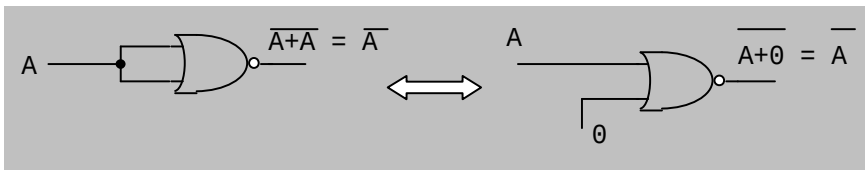


Fig.3.20(a)

(ii) NOR as OR:

$$A + B = \overline{\overline{A + B}} = \text{ Complement of } \overline{A + B}$$

which may be illustrated as shown in figure 3.20(b).

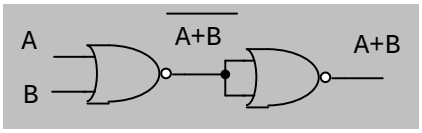


Fig. 3.20(b)

(iii) NOR as AND:

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

which may be illustrated as shown in figure 3.20 (c).

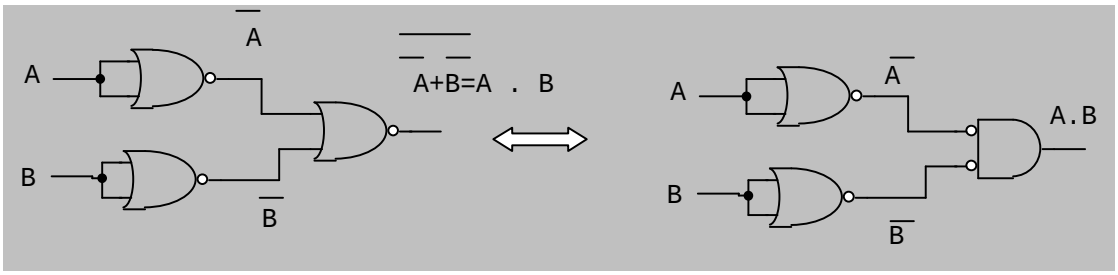


Fig. 3.20(c)

The minimization of Boolean expressions using the theorems of Boolean algebra have already been discussed which help in reducing the literals or variables. The reduced Boolean expressions can be realized with less number of AND, OR, NOT gates and also each (AND/OR) gate having minimum number of inputs. There are also other methods for minimizing the Boolean expressions which will be discussed in a later chapter. However, for realization of Boolean expressions using the NAND/NOR gates alone, more number of these gates are required. This is not true when IC's are used, since several gates are available in an IC. So for making the circuit using the NAND/NOR gates alone will not cost more.

3.11 Realization of Boolean expressions using NAND/NOR alone: The given Boolean expression is generally simplified using the theorems Boolean algebra or other methods to be discussed in a later chapter, to obtain the minimal Boolean expressions having less number of variables and their complements. Now the logic circuit diagram corresponding to the simplified Boolean expression is drawn. If the Boolean expression is in the sum of products (SP) form, then NAND gates should be used for realization. However, NOR gates should be used for realization, if the simplified Boolean expression is in product of sums (PS) form. This method enables the realization simple and requires least number of cascading gates.

The general rules for NAND gates realization of Boolean expression given in SP form, are given below:

1. For each product terms, use the literals or variables as inputs to NAND gates.
2. Feed the output of all such NAND gates to a second level NAND gate.
3. Any literal appearing alone as a term is complemented and connected to the NAND gate at the second level.

For example let $F = AD + BC + E$ is given for realization. NAND gates realization of this expression is shown in figure 3.21.

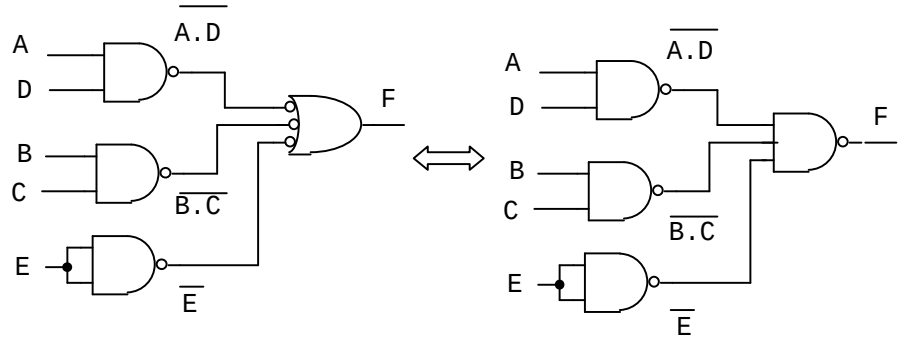


Fig. 3.21

Similarly, if we have a Boolean expression in PS form as $F = (A + D) \cdot (B + C) \cdot E$, the NAND realization of this expression is shown in figure 3.22. This circuit requires three levels of gating. Each level adds to the propagation delay. The aim of the circuit designers should be that there should be minimum number levels of gating. The realization of this circuit with NOR gates will have only two levels of gating. The NOR realization of this circuit is shown in figure 3.23. Hence NOR realization is preferred for the Boolean expressions given in PS form.

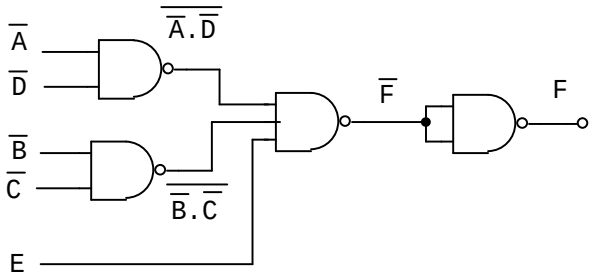


Fig. 3.22

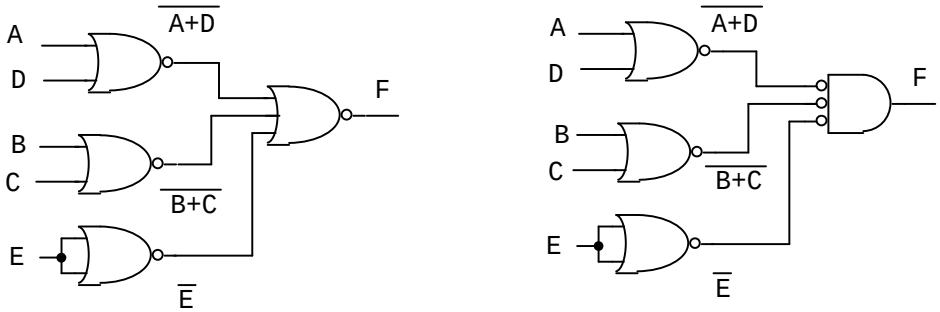


Fig. 3.23

The general rules for NOR gates realization of Boolean expression given in PS form, are given below:

4. For each sum terms use the literals as inputs to NOR gates.
5. Feed the output of all such NOR gates to a second level NOR gate.
6. Any literal appearing alone as a term is complemented and connected to the NOR gate at the second level.

Example 3.9: (i) Prove the associative law for exclusive – OR operation.

(iii) Prove that the NAND operation for three variables is not associative.

Solution: (i) The associative law for exclusive – OR operation is given as:

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

L.H.S. = $A \oplus (B \oplus C)$

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

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

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

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

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

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

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

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

R.H.S. $= (A \oplus B) \oplus C$

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

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

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

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

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

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

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

L.H.S. = R.H.S Hence proved.

(ii) We are to prove that $A \uparrow (B \uparrow C) \neq (A \uparrow B) \uparrow C$

L.H.S. $= A \uparrow (B \uparrow C)$

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

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

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

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

R.H.S. $= (A \uparrow B) \uparrow C$

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

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

L.H.S. $\neq$ R.H.S. Hence proved

PROBLEMS:

1. State and prove Demorgan's law for two variables. How can it be proved for n variables?
2. Discuss the theorems of Boolean algebra.
3. What is the difference between the ordinary algebra and Boolean algebra?
4. What do you understand by logics? Discuss the AND and OR operations using the suitable diagram. Draw the truth table for three input AND and OR operations.
5. What do you understand by logics? Discuss NOT operations using a suitable diagram.
6. What is Venn diagram? Prove the following identities using Venn diagrams:

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

   (ii) $a + b \cdot c = (a + b) \cdot (a + c)$

   (iii) $a + \overline{a} = a$

   (iv) $\overline{a + b} = \overline{a} \cdot \overline{b}$
7. Prove the following postulates of Boolean algebra and then verify the following identities using Venn diagram.

   (i) $a + \overline{a} \cdot b = a + b$

   (ii) $a + a \cdot b = a$

   (iii) $\overline{a} + \overline{a} \cdot b = \overline{a}$

   (iv) $\overline{a \cdot b} = \overline{a} + \overline{b}$
8. Describe the method of constructing truth table of a Boolean expression. Taking a suitable function of three variables draw the truth table.
9. Discuss the method of getting canonical PS form of Boolean function of a given truth table.
10. Discuss the method of getting canonical SP form of Boolean function of a given truth table.
11. Describe the method of converting the PS form of Boolean function to SP form.

12. Mention and explain the different Boolean operators from a two bit truth table. How many of them are used to deign the gates? Why NAND and NOR gates are known as universal gates.

13. Explain how AND, OR, NOT gates can be realized using NAND gates alone.

14. Explain how AND, OR, NOT gates can be realized using NOR gates alone.

15. Prove the associative law for exclusive - OR and equivalence (XNOR) operators.

16. Prove that the associative law does not hold for NAND operators for three variables.

17. Prove that the associative law does not hold for NOR operators for three variables.

18. Using the theorems of Boolean algebra , prove the following identities:

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

(ii) $\overline{X}\cdot(X+Z)+\overline{Y}+Y\cdot Z=\overline{Y}+Z$

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

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

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

19. Prove the following:

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

20. Using the theorems of Boolean algebra , prove the following identities:

(i) $\overline{X}\,\overline{Y}\,\overline{Z}\,\overline{W}+\overline{X}\,\overline{Y}Z\,\overline{W}+X\,\overline{Y}\,\overline{Z}\,\overline{W}+\overline{X}ZW+X\,\overline{Y}Z\,\overline{W}=\overline{Y}\,\overline{Z}+\overline{X}ZW$

(ii) $\overline{X}YZW+X\,\overline{Y}\,\overline{Z}\,\overline{W}+X\,\overline{Y}\,\overline{Z}W+X\,\overline{Y}Z\,\overline{W}+X\,\overline{Y}ZW$
$+XY\,\overline{Z}\,\overline{W}+XY\,\overline{Z}W+XYZ\,\overline{W}+XYZW=X+YZW$

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

21. Construct the truth table of the Boolean expressions and from the truth table find the canonical form of the Boolean expression.

(i) $Z=\overline{A}\cdot B+C\cdot\overline{D}+BD$

(ii) $F=\overline{X}\cdot Y+Y\cdot\overline{Z}$

22. Simplify the following functions using the theorems Boolean algebra.

(i) $F(a,b,c,d)=\sum(0,1,2,8,9,10)$

(ii) $F(a,b,c,d)=\sum(0,1,4,6,9,13,14,15)$

(iii) $F(X,Y,Z,W)=\sum(0,4,5,8,10,12,15)$

(iv) $F(A,B,C,D) = \sum (1,2,3,5,6,7,9,11,14,15)$

23. Realized the minimal Boolean expressions obtained in above problem (22) using

(i) AND, OR NOT gates.
(ii) NAND gates only.

24. Simplify the following functions using the theorems Boolean algebra.

(i) $P(X,Y,Z,W) = \prod (4,6,12,13,14,15)$

(ii) $Z(A,B,C,D) = \prod (3,4,6,7,11,12,13,14,15)$

(iii) $F(X,Y,Z,W) = \prod (0,1,2,3,8,9,10,11,13,15)$

25. Realized the minimal Boolean expressions obtained in above problem (24) using:

(i) AND, OR NOT gates.
(ii) NOR gates only.

26. Express the following function into canonical form:

(i) $F(a,b,c) = \bar{a} \cdot b + \bar{a} \cdot c$

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

(iii) $\mathrm{F(X, Y, Z, W)} = \overline{X} \cdot Y \cdot \bar{Z} + \bar{Y} \cdot \bar{Z} + X \cdot Y \cdot Z \cdot W$

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

27. Express the following function in to canonical SP form

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

28. Express the following function in to canonical PS form.

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

_____

4

Simplification of Boolean Functions

In the preceding chapter it has been discussed that the Boolean functions can be simplified using the theorems of Boolean algebra. The reduced Boolean expression helps in getting the simple, less expensive and smaller circuit. This method of simplification is not used in practice as it is difficult to apply. Further, it is impossible to guarantee that the reduced expression is minimal and it cannot be reduced beyond the obtained expression. The two other methods for simplifications of Boolean expression will be discussed in the following sections of this chapter. One is known as Karnaugh map (K – map) method and other is known as Quine – McClusky (Q – M) tabular method.

4.1 Karnaugh map (K – map) method: The Karnaugh map method is very commonly used for the simplification of Boolean expressions, since no algebraic rules are applied in this method. It is simply a graphic method and provides systematic approach for getting the simplified Boolean expression. If this method is properly used then the available Boolean expression will be minimal and will not further be simplified by any method. The Karnaugh map also called K – map method is suitable for simplification Boolean expression which contains four or less number of variables (or literals) with their complements.

4.1.2 Two Variable K – map: For two variable K – map two lines are drawn; one horizontal and the other vertical. On one line the complement of one variable followed by the variable itself is written and on the other line the complement of the other variable followed by that variable itself is written. Let the two variables are A and B. So $\overline{A}$ and variable A, are written on vertical line; $\overline{B}$ and variable B are written on horizontal line or otherwise as shown in figure (4.1). The other method of writing K – map for two variables is that in place of $\overline{A}$, 0 is written and in place of A, 1 is written; as illustrated in figure (4.2), where A and B are shown separately over and below of a leaning line.

Figure 4.1

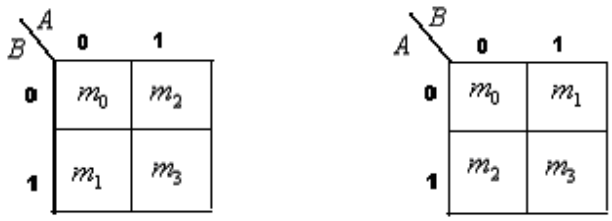


Figure 4.2

These two figures are similar and show the same meaning. One can use either of the two ways mentioned above. Each K – map show four squares represented by four minterms m_0, m_1, m_2, and m_3. In the truth table of two variables if there are 1's entry corresponding to the some minterms, then 1s are entered corresponding to those minterms. Let us assume that we have 1s entry for the minterms m_2 and m_3. The K – map for the same is represented as shown in figure 4.3.

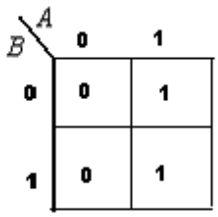


Figure 4.3

4.1.3 Three Variable K – map: For three variables two adjacent variables are taken on either side (vertical line or horizontal line) of the K – map and the remaining one variable on the other side. Let A, B and C are the three variables, the two variables will have four combinations labeled on one side as $\overline{A}.\overline{B}$, $\overline{A}.B$, $A.B$ and $A.\overline{B}$; $\overline{C}$ and C on the other side as shown in figure 4.4.

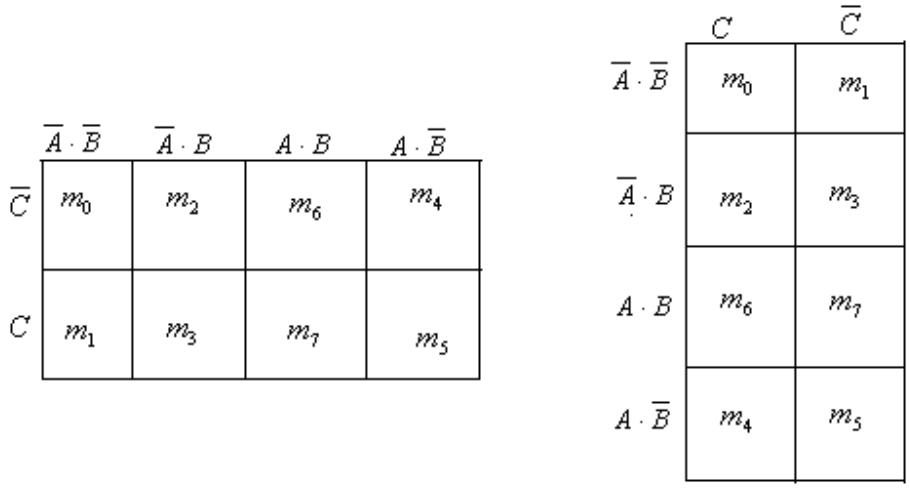


Fig. 4.4

The other method of writing K – map for three variables is that in place of the possible combinations of two variables A & B it is written as 00, 01, 11 and 10; and in place of $\overline{C}$ & C, 0 & 1 are written; as illustrated in figure (4.5), where AB and C are shown separately over and below of a leaning line.

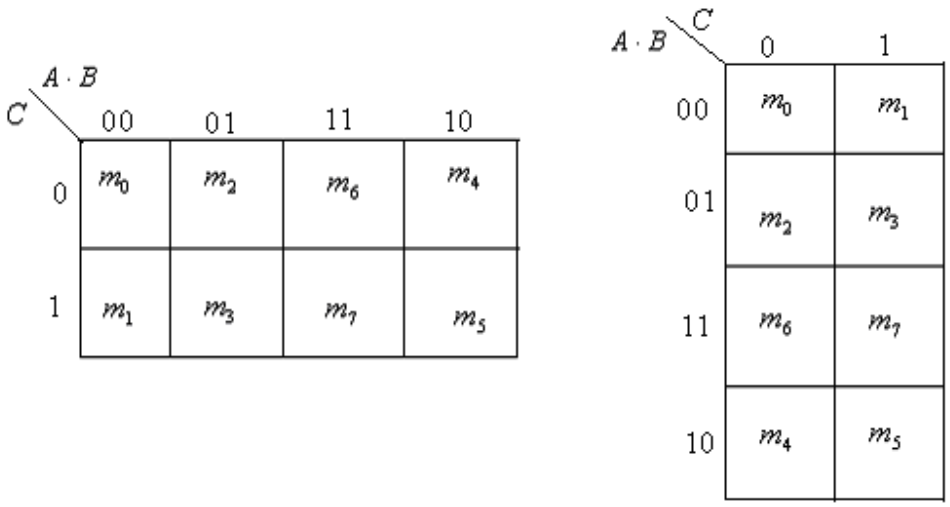


Fig. 4.5

4.1.4 Four Variable K – map: For four variables two adjacent variables are taken on either side (vertical line or horizontal line) of the K – map and the two variables on the other side. Let A, B, C and D are the four variables, the two variables will have four combinations labeled on one side as $\overline{A}.\overline{B}$, $\overline{A}.B$, $A.B$ and $A.\overline{B}$; and other two will also

have the four combinations as $\overline{C}.\overline{D}$, $\overline{C}.D$, $C.D$ and $C.\overline{D}$ on the other side as shown in figure 4.6.

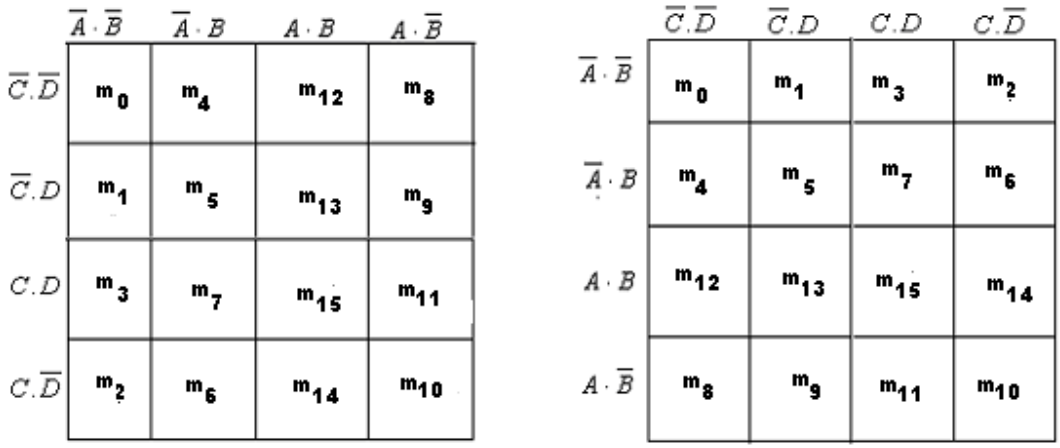


Fig. 4.6

The possible combinations of AB and CD (discussed above) may also be shown separately over and below of a leaning line as illustrated in figure 4.7.

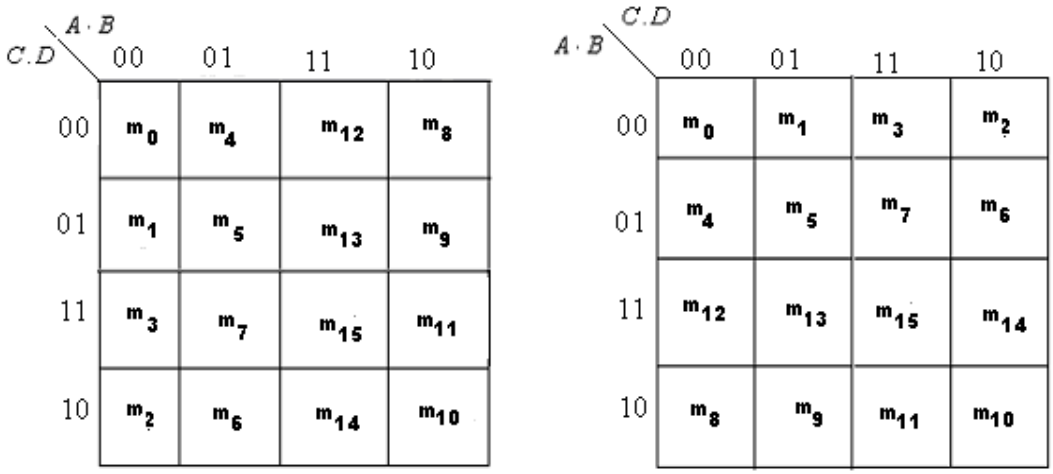


Fig. 4.7

If a Boolean function of three variables or four variables is given, the 1s entry in the K – map is done for those combinations which are present in the given expression and for the other combinations 0s entry are made.

Example 4.1: Draw the K – maps for the following Boolean function of three variables.

$$F_1(A,B,C) = \sum(m_1, m_3, m_5, m_6, m_7)$$

Solution: In the K – map of three variables 1s entry are made for the combinations m_1, m_3, m_5, m_6, m_7 and in the remaining combinations, 0s are entered. The K – map for the same is shown in figure 4.8.

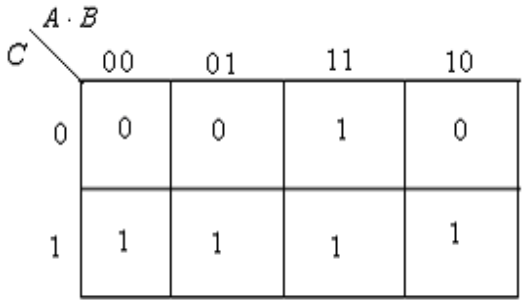


Fig. 4.8

Example 4.2: Draw the K – maps for the following Boolean function of four variables.

$$F_1(A,B,C,D) = \sum(m_2, m_3, m_4, m_6, m_7, m_{11}, m_{14}, m_{15})$$

Solution: In the K – map of four variables 1s entry are made for the combinations $m_2, m_3, m_4, m_6, m_7, m_{11}, m_{14}, m_{15}$ and in the remaining combinations, 0s are entered. The K – map for the same is shown in figure 4.9.

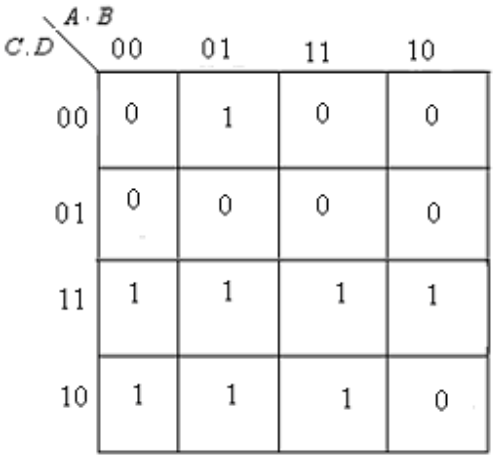


Fig. 4.9

4.2 Encircling of Groups: After constructing K – map, the pairs quads and octets of adjacent 1s in the K – map are made for getting the minimal Boolean expression. A pair eliminates one variable with its complement; a quad and an octet eliminate two variables and three variables respectively with their complements. Now it will be discussed how pairs, quads and octets are formed in the K – map and help in minimizing the Boolean expression.

4.2.1 Pairs: In the three-variable or four variable K – map having 1s and 0's entry, two adjacent 1s (vertically or horizontally) are encircled. The diagonally adjacent 1s are never encircled. The encircled 1s forms the pairs, as shown in figure 4.10.

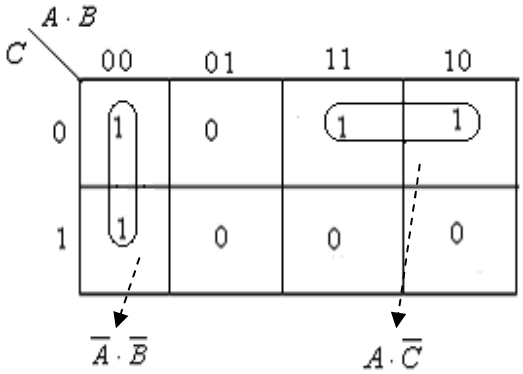


Fig. 4.10(a)

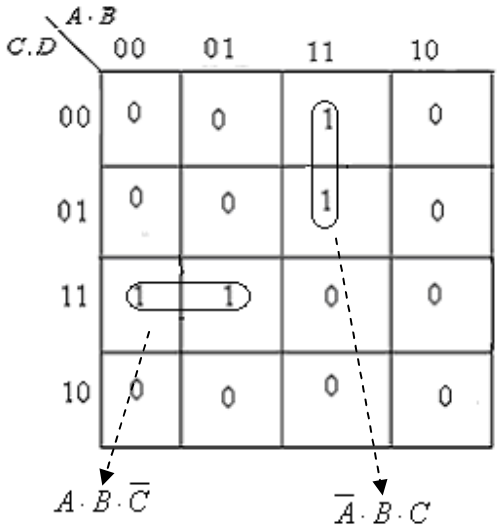


Fig. 4.10(b)

Now it is clear from the K – map of three variables (ref. figure 4.10 a) that there are two encircled pairs of adjacent 1s and these pairs correspond to the terms $\overline{A} \cdot \overline{B}$ and $A \cdot \overline{C}$. The method of writing these terms is that the variable, which gets changed from complemented form to un-complemented form or vice versa, is dropped with its complement. This can be illustrated as follows:

Consider the pair (fig. 4.10a) whose term is $\overline{A} \cdot \overline{B}$, the two 1s contained in the pair has the binary numbers 000 and 001 for the variables ABC. In the binary numbers, the variable C changes from 0 to 1 (complemented to un-complemented), so this variable is dropped with its complement. The binary number left is 00 having the term (in SOP form) as $\overline{A} \cdot \overline{B}$. Similarly, consider the second pair whose term is $A \cdot \overline{C}$. The two 1s contained in this pair have the binary numbers 110 & 100 for variables ABC. The variable B changes from 1 to 0 so this variable is dropped with its complement. The remaining (common) binary number 10 for variables AC will have the term (in SOP) as $A \cdot \overline{C}$.

The Boolean algebra is involved in getting the expression for a pair. In the first encircled pair (discussed above) of three variable K – map, each 1 of the pair show the terms $\overline{A} \cdot \overline{B} \cdot \overline{C}$ and $\overline{A} \cdot \overline{B} \cdot C$ (corresponding to binary numbers 000 & 001). When these terms are ANDed, $\overline{A} \cdot \overline{B}$ is obtained:

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

Similarly, one can verify the terms corresponding to encircled pairs in four variable K – map as given in figure (4.10 b).

From the above discussion it is clear that a pair eliminates one variable with its complements, i.e., the pair will contain the term of two variables in three variable K – map and it will contain the term of three variables in four variable K – map.

4.2.2 Quads: In the K –map if four 1s are adjacent in a row or column or in the form of a square, then these 1s are encircled called as quads. A term is written for each quad using the same techniques discussed above. The variables which changes from complemented to uncomplemented or vice versa are dropped and the variables which are common in all the four 1s of a quad are considered to write term in SOP form. Consider a K – map shown in figure 4.11.

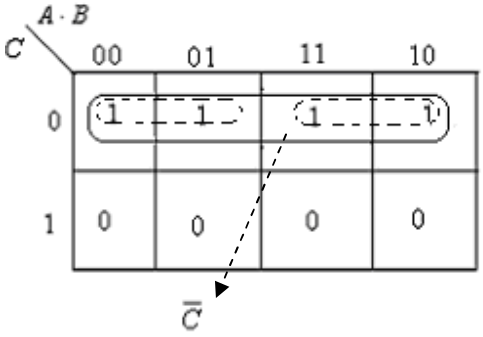


Fig. 4.11a

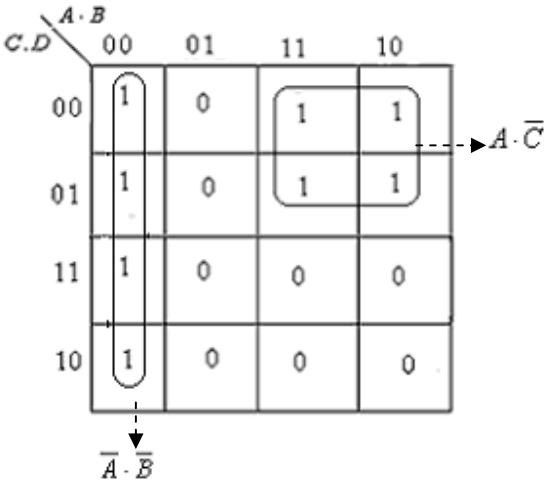


Fig.4.11b

In the K – map of three variables (figure 4.11a), the encircled group of 1s shows the quad whose four elements represent the binary numbers 000, 010, 110 & 100 for variables ABC. In the binary numbers, 0 for C is common for all the four binary numbers; so $\overline{C}$ (in SOP form) is the term for this quad. The variables AB are dropped as each of the variable changes 0 to 1 or 1 to 0. The Boolean algebra is involved in getting the expression for a quad. The quad is the combination of two pairs (shown by dotted encircles in figure 4.11a). In the encircled quad, the two pairs will show the terms $\overline{A} \cdot \overline{C}$ and $A \cdot \overline{C}$. When these terms are ANDed, $\overline{C}$ is obtained as:

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

Similarly in the K – map of four variables (figure 4.11b), the two quads are encircled. One can verify the terms for each quad.

It is clearly illustrated that the quad will contain the term of one variable in a K – map of three variables and it will contain the terms of two variables in the K – map four variables. It may, therefore, be stated that a quad eliminates two variables with their complements.

4.2.3 Octets: The eight adjacent 1s are encircled in a K – map known as octet. Figure 4.12 shows the encircled octet (solid line) in a K – map of 4 variables. The term for the octet is B, which is written with the same technique as used for pairs and quads.

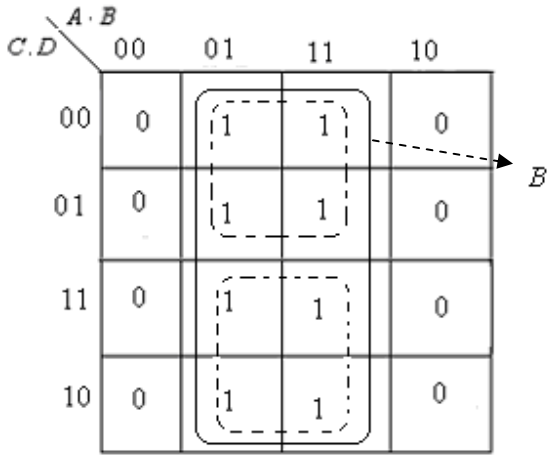


Fig. 4.12

An octet eliminates three variables with their complements and gives a term of one variable in a K –map of four variables. In fact an octet is a combination of two quads as shown by dotted lines (figure 4.112). The terms for two quads are $B \cdot \overline{C}$ and $B \cdot C$. When these two terms are combined it gives:

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

4.2.4 Overlapping groups: While making encircled groups in the K – map, it is always tried to have the groups of largest number of 1s first than others, i.e. octets are encircled first than quads and than pairs. It is important to use same 1 more than once. In other words same 1 may be used in more than one encircled groups. Such groups are called as the overlapped groups. Some overlapped groups are shown in figure 4.13.

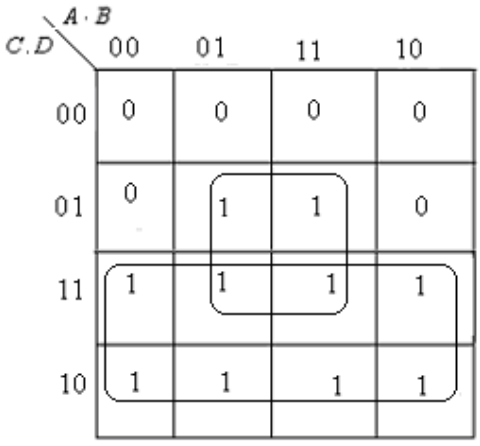


Fig. 4.13(a)

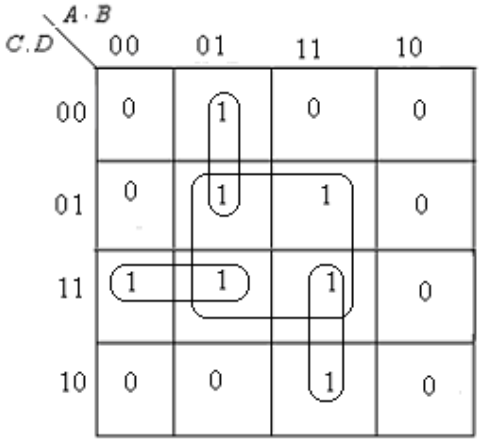


Fig. 4.13(b)

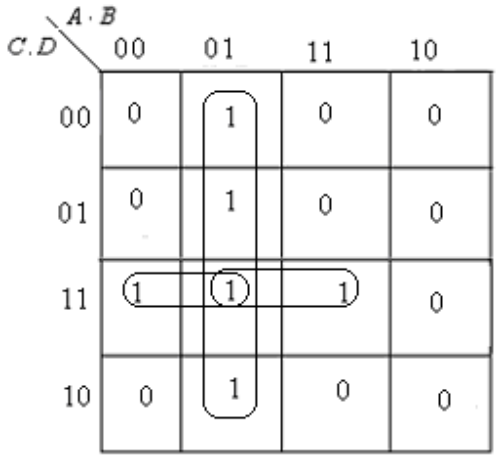


Fig. 4.13 (c)

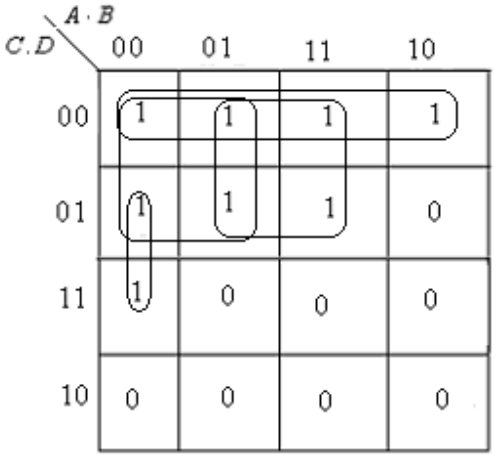


Fig. 4.13(d)

The terms for each encircled groups are written in the same manner as is done for normal pairs, quads and octets.

4.2.5 Rolling groups: It is also allowed to roll the K – map so that grouping of largest number of 1s may be formed. To understand this consider a K – map as shown in figure (4.14a). In this K – map while encircling, one can obtain two quads but using the rolling of K – map, an octet may be formed as shown in figure (4.14b). Here the rolling is done in such a way that the left hand side encircled quad touches the right hand side encircled quad. This in fact looks like an octet. The rolling is shown by half encircling the two groups as shown in figure (4.14b). Thus the term corresponding to the rolled octet is written in the same fashion as in normal encircling.

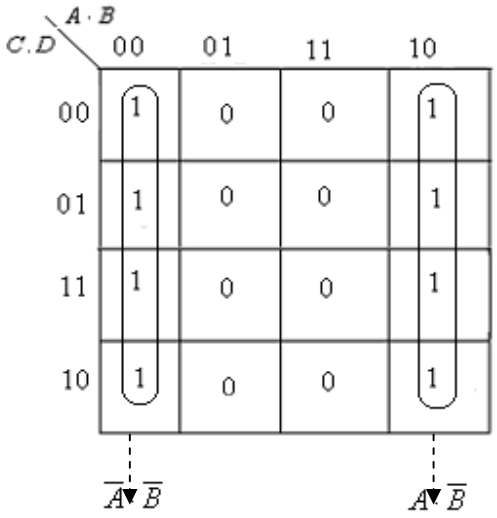


Fig. 4.14a

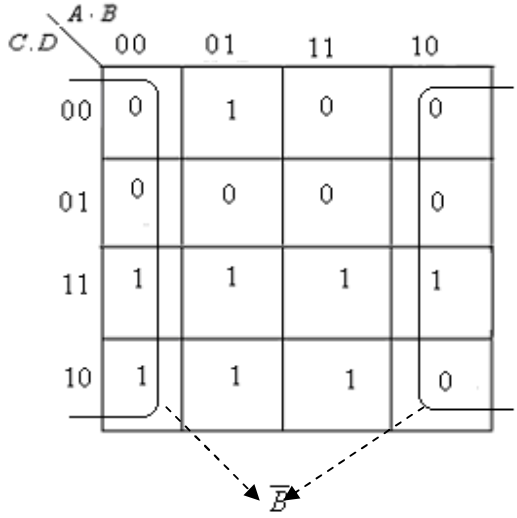


Fig.4.14b

The rolling is possible for quads and pairs also as illustrated in figure 4.15.

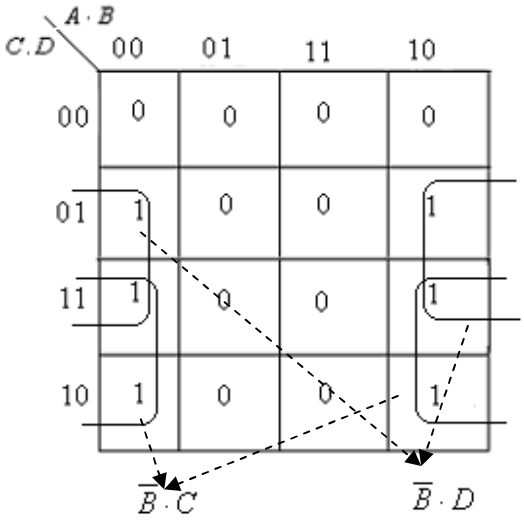


Fig. 4.15a

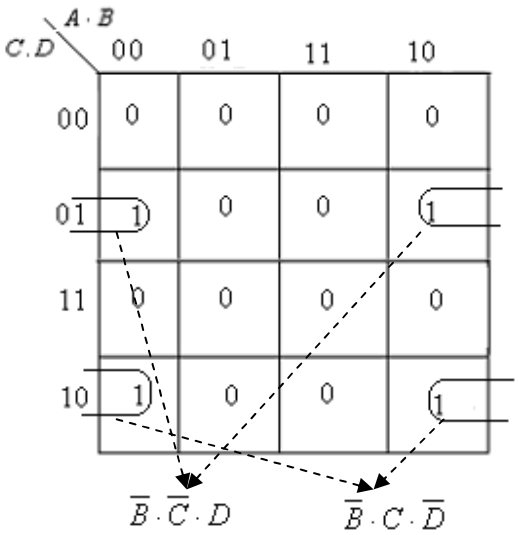


Fig. 4.15b

The rolling is not only possible with the 1s of extreme left columns and the 1s of extreme right columns of the K – map, but it is possible with the 1s of upper most row and the 1s of lower most column as shown in figure 4.16.

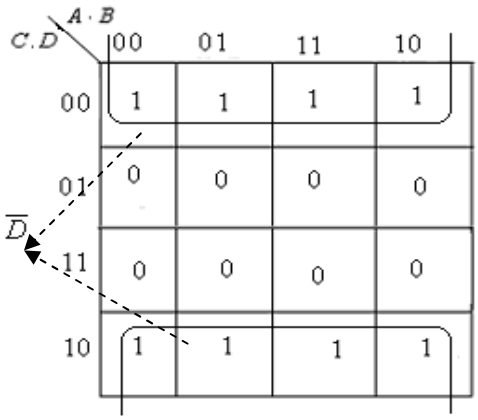


Fig.4.16a

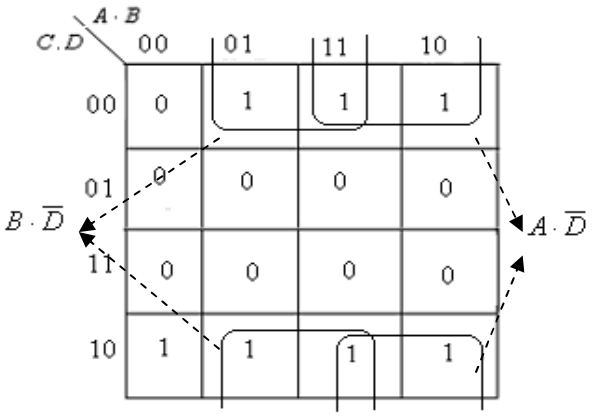


Fig. 4.16b

4.2.6 Redundant Groups: While encircling the groups in the K – maps, there is a possibility that all the elements (1s) of some group/groups are overlapped by other groups. Such a group whose all 1s are overlapped by other groups is called a redundant group. The redundant groups may be illustrated by considering a K – map shown in figure 4.17.

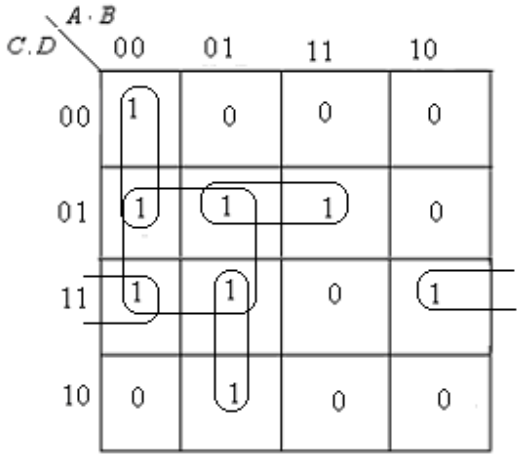


Fig.4.17

In this K – map the encircled groups are: one quad and four pairs. But quad is redundant since all its four 1s are used in forming other pairs. The quad is, therefore, eliminated. So the valid encircled groups will be as shown in figure 4.18.

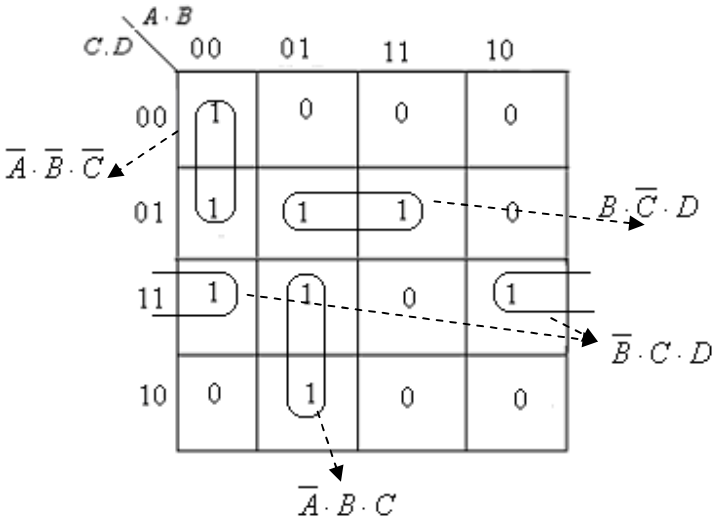


Fig. 4.18

The encircled groups of this figure are simplified one and the minimal expression of this K – map is given as :

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

4.3.7 Procedure for Simplification: The different rules for encircling the groups in the K – map have been discussed in the above sections. Now using these rules the method of getting the minimal Boolean expression of the given truth table (or function) will be summarized below:

- After forming the K – map, enter 1s for the min-terms that correspond to 1 in the truth table (or enter 1s for the min-terms of the given function to be simplified). Enter 0s for the remaining min-terms.
- Encircle octets, quads and pairs of course remembering rolling and overlapping. Try to form the groups of maximum number of 1s.
- If any such 1s occur which are not used in any of the encircled groups, then these isolated 1s are encircled separately.
- Review all the encircled groups and remove the redundant groups, if any.
- Write the terms for each encircled group.
- The final minimal Boolean expression corresponding to the K – map will be obtained by ORing all the terms obtained above.

Example 4.3: Using K –map simplify following Boolean function of three variables.

$$F(A,B,C) = \sum (m_0, m_1, m_5, m_6, m_7)$$

Solution: The K –map for the given function is drawn, after encircling the groups of 1s, as shown in figure 4.19. The required Boolean expression is given by:

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

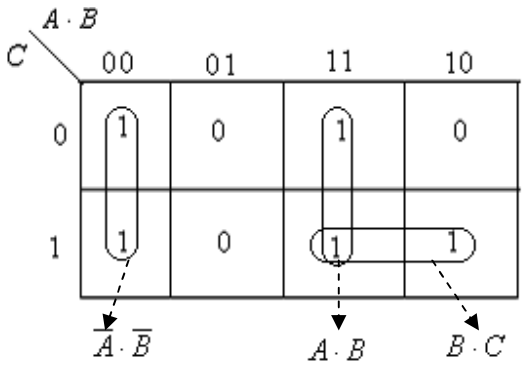


Figure 4.19

Example 4.4: Using K –map simplify following Boolean function of four variables.

$$F(A,B,C,D) = \sum (0,1,2,4,7,9,11,12)$$

Solution: The K –map for the given function is drawn, after encircling the groups of 1s, as shown in figure 4.20. The required Boolean expression is given by:

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

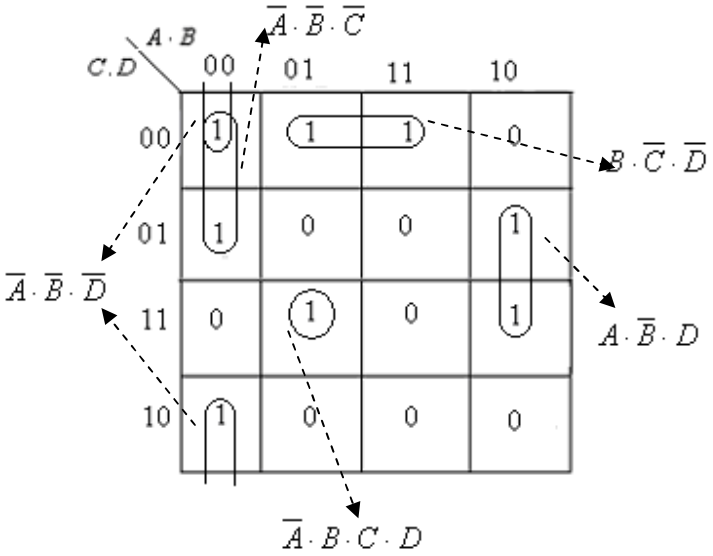


Fig. 4.20

Example 4.5: Minimize the following function using K – map and realize it with AND, OR & NOT logic gates.

$$X(A, B, C, D) = \sum (0,1,2,5,8,10,11,14,15)$$

Solution: The K –map drawn after encircling the groups of 1s, for the given function is shown in figure 4.21. The required Boolean expression is given by:

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

It is interested to note from this figure that the four 1s at the corners of the K – map forms a quad due to the rolling on both sides, whose term will be $\overline{B}\cdot\overline{D}$.

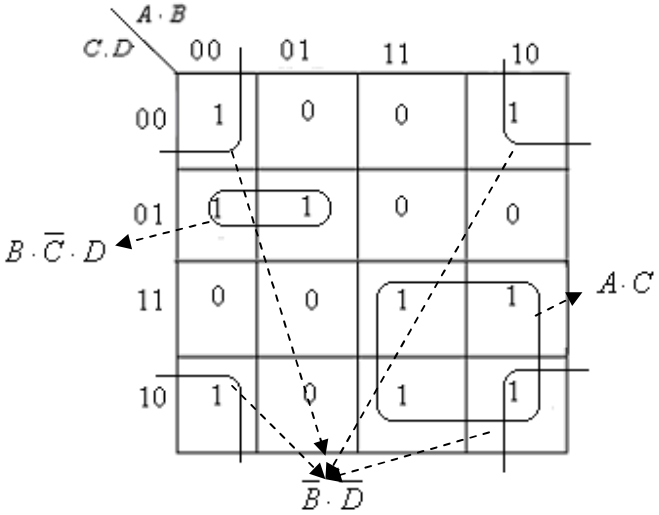


Fig.4.21

After getting the minimal Boolean expression, realization of the function will be as shown in figure 4.22.

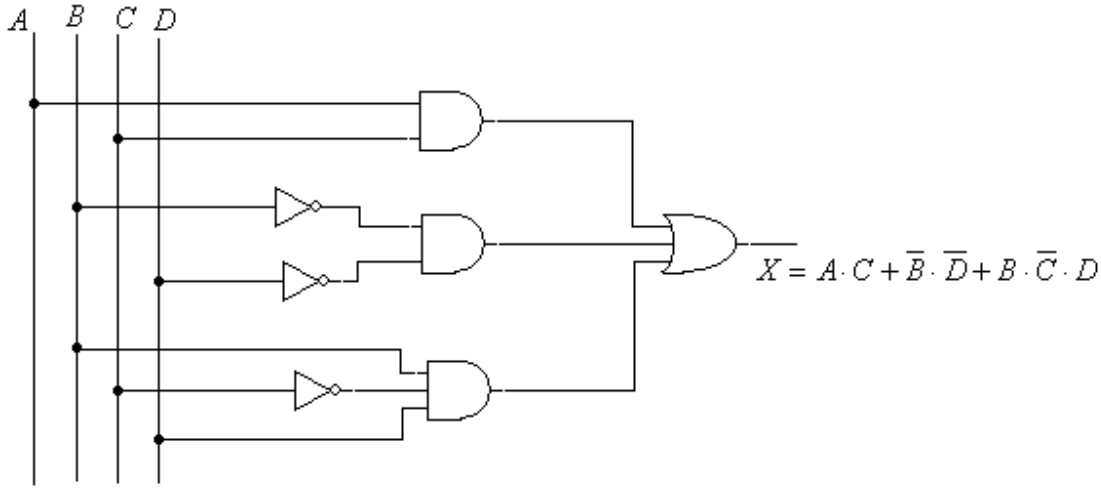


Fig. 4.22

4.3 Incompletely Specified Functions: In digital circuits we come across of two types of functions; namely completely specified functions and incompletely specified functions. The functions whose values are specified for all min-terms are known as completely specified functions and in the K –map either 0s or 1s are entered for all the min-terms; such functions have been considered so far. But sometimes we encounter situations in which some min-terms do not occur. For example: Input data to a digital system are sent in 8421 code in which the combinations 0000 through 1001 occur, and 1010 through 1111 are known as illegal combinations as these combinations do not occur. So the combinations 0000 to 1001 will have the output as 0 or 1 and accordingly these values may be entered in the K – map. The combinations 1010 to 1111 will have the output neither 0 nor 1, as these combinations do not occur in the given system. So these combinations are called as the incompletely specified functions. In the K –map φ is entered for every incompletely specified function. The φ is called as 'don't care' condition. While encircling the groups in the K –map φ may assume to be either 0 or 1, whichever helps in giving the simplest expression. The don't cares are treated as 1s inside encircled groups in the K –map and are treated as 0s outside the encircled groups.

This can be illustrated by taking an example. Suppose we wish to minimize the following function of four variable having the 'don't cares' also. The 'don't cares' are shown by φ below the summation symbol.

$$X(A, B, C, D) = \sum (1,2,3,5,13) + \sum_{\phi} (6,7,8,9,10,11)$$

This function shows that in the K –map 1s are entered corresponding to the min-terms 1,2,3,5,13 and φ are entered for 6,7,8,9,10,11 and 0s are entered for the remaining terms. The K –map with these entries are shown in figure 4.23(a) and its encircling & simplification is shown in figure 4.23(b). There are two encircled quads and φ which are inside the encircled groups are treated as 1s. The last column of this K –map is not

encircled to form the quad as its all elements are φ. It must be remembered that no such group is formed whose all the elements are φ. The minimal Boolean function of this K – map is given by:

$$X = \overline{A} \cdot C + \overline{C} \cdot D$$

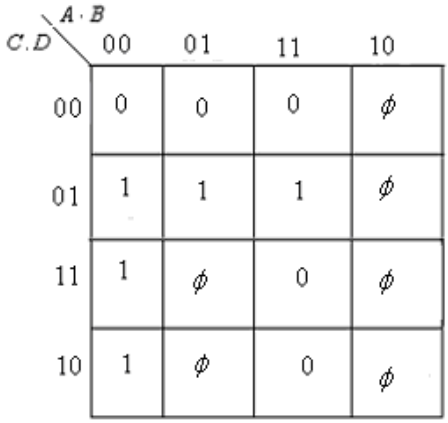


Fig. 4.23 a

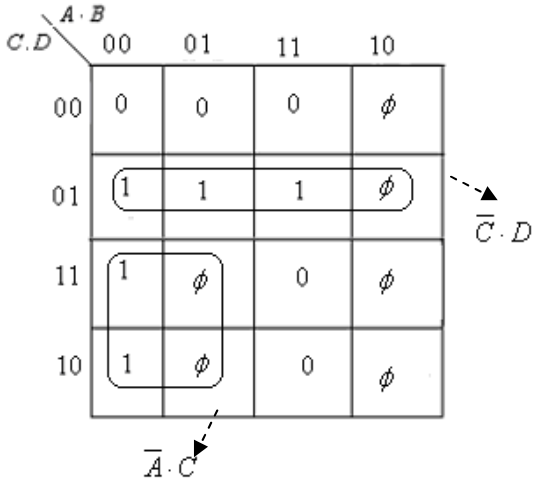


Fig.4.23 b

The general procedure of getting the minimal Boolean expression of a K-map including the 'don't care' conditions is summarized below:

- After forming the K – map, enter 1s for the min-terms that correspond to 1 in the truth table (or enter 1s for the min-terms of the given function to be simplified). Enter φ to the 'don't care' conditions and 0s for the remaining min-terms.
- Remembering rolling and overlapping, encircle octets, quads and pairs. The φ may be treated as 1 if these help in forming largest groups. No such group will be formed whose all the elements are φ.
- If any such 1s occur which are not used in any of the encircled groups, then these isolated 1s are encircled separately.
- Review all the encircled groups and remove the redundant groups, if any.
- Write the terms for each encircled group.
- The final minimal Boolean expression corresponding to the K – map will be obtained by ORing all the terms obtained above.

Example 4.6: Minimize the following function using K – map and realize it with NAND gates only.

$$F(W, X, Y, Z) = \sum (0,2,3,5,6,8,9) + \sum_{\phi} (10,11,12,13,14,15)$$

Solution: The K –map is drawn for the given function and encircling of the groups is done as shown in figure 4.24. The required Boolean expression is given by:

$$F = W + Y \cdot \overline{Z} + \overline{X} \cdot \overline{Z} + \overline{X} \cdot Y + X \cdot \overline{Y} \cdot Z$$

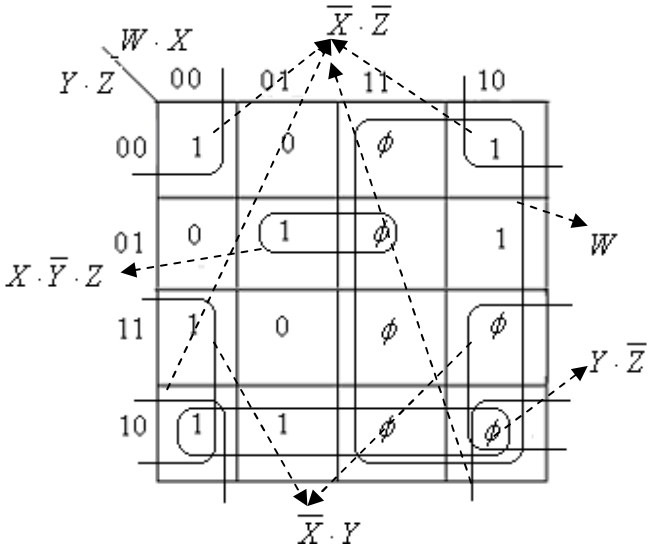


Fig. 4.24

The realization of this function using NAND gates only is shown in figure 4.25.

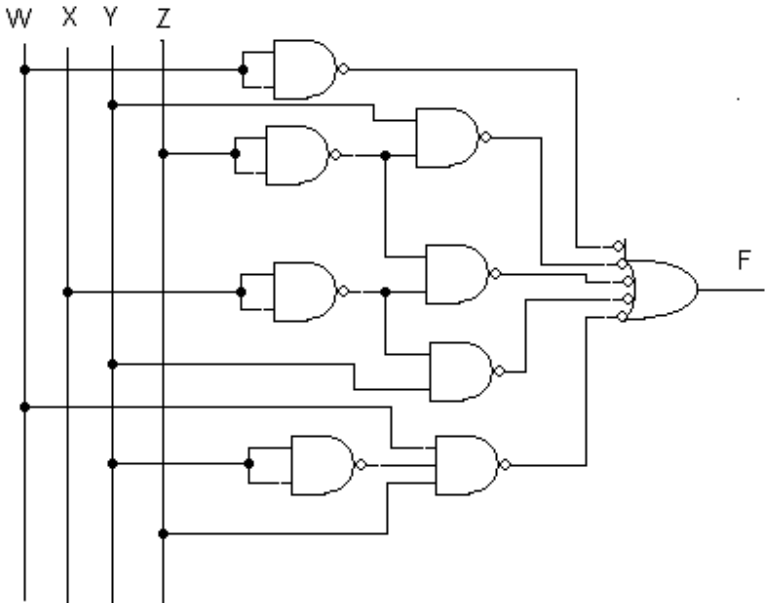


Fig. 4.25

4.4 NOR Implementation of Boolean Functions: The implementation of Boolean function with NOR gates requires the simplified Boolean function in product of sums form. But the K –map discussed above gives the simplified expression in sum of

products form. So for getting the Boolean expression in POS form the encircling may be done with 0s and don't care conditions. The same rules will be followed for encircling with 0s and φ. However, the terms for encircled groups are obtained in max-term form. The variables which get changed in moving from one element to the other adjacent element in the encircled groups of 0s will be eliminated with their complements; and the variables common to all the elements in the encircled group will be used in writing the max-term. For example if 01 is common for AB variables in an encircle group of 0s, then the max-term corresponding to 01 will be $(A+\overline{B})$. The final minimal Boolean expression for the K –map will be obtained by ANDing all the terms obtained above.

It can further be illustrated by taking an example. Find the minimal Boolean expression using K –map for the following function.

$$F(A,B,C,D) = \prod(4,6,7,8,10,11) \cdot \prod_{\phi}(0,1,2,13,14,15)$$

In this given function 0s are entered for the terms 4,6,7,8,10,11and φ are entered for 0,1,2,13,14,15 in the K –map and the minimal expression is obtained in POS form. The K –map is shown in figure 4.26.

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

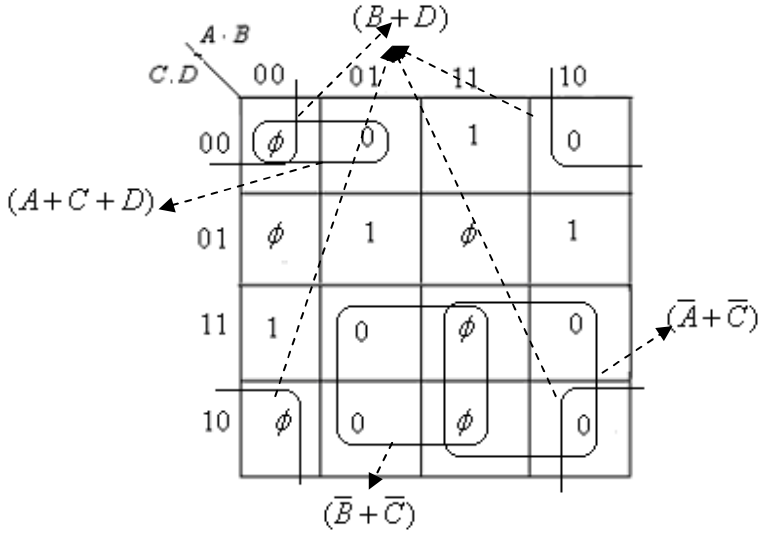


Fig. 4.26

Example 4.7: Minimize the following function using K – map and realize it with NOR gates only.

$$F(A,B,C,D) = \prod(0,1,4,5,8,12,14,15) \cdot \prod_{\phi}(9,11,13)$$

Solution: The K –map is drawn for the given function and encircling of the groups is done as shown in figure 4.27. The required Boolean expression is given by:

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

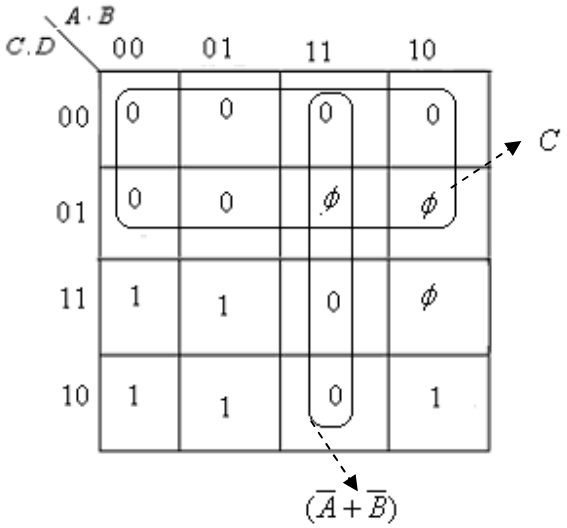


Fig. 4.27

The realization of this function using NOR gates is shown in figure 4.28.

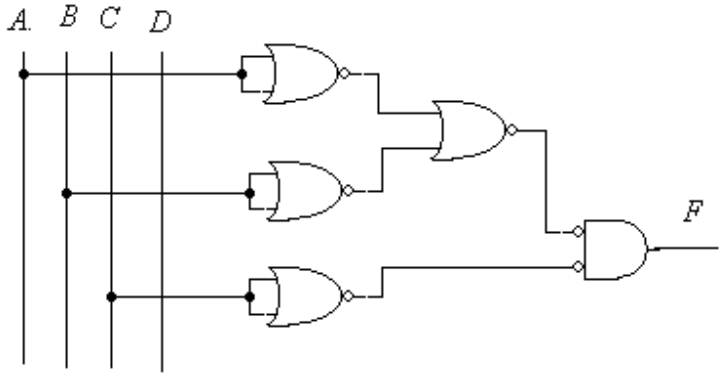


Fig. 4.28

Example 4.8: Minimize the following function using K – map and realize it with NOR gates only.

$$F(A,B,C,D) = \sum (0,1,5,8,10,14) + \sum_{\phi} (2,7,11,15)$$

Solution: The entries of 0s, 1s and don't care conditions in the K –map are made as per the given problem. The encircling of the groups are done with 0s and φ (shown in figure 4.29), since the minimized function is to be realized with NOR gates only. The minimized function in POS form is given by:

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

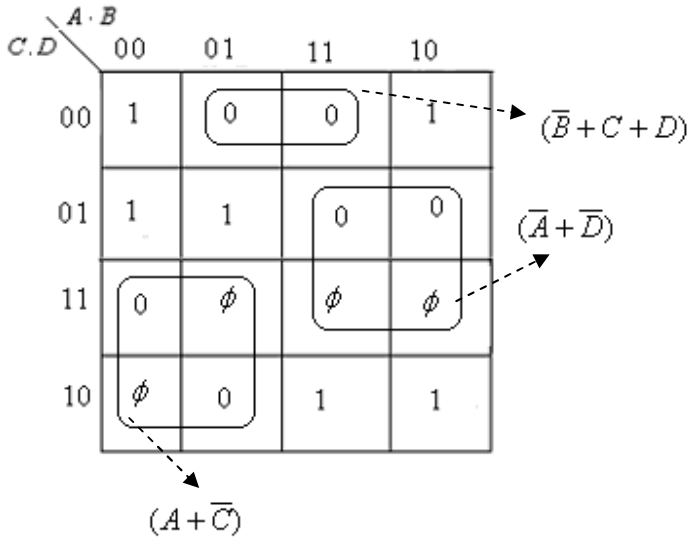


Fig. 4.29

The realization of this function using NOR gates is shown in figure 4.30.

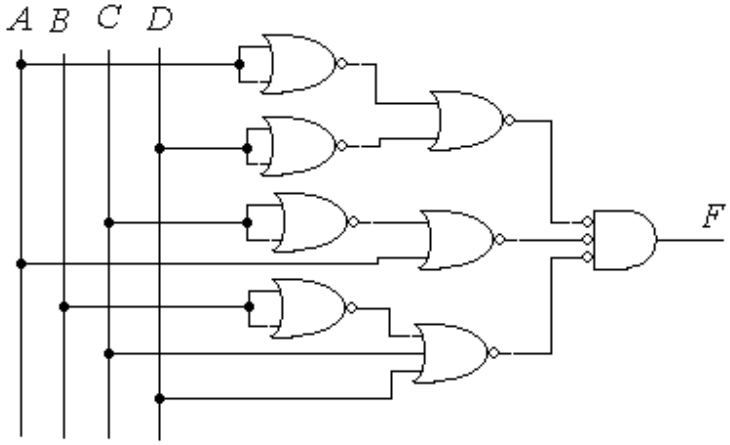


Fig. 4.30

4.5 Five and Six Variable K – map: The simplification of the Boolean function up to four variables have been discussed in the forgoing sections of this chapter. Maps for more than four variables become complicated and its use are not very simple. The five variable map should have 32 squares as it will have min-terms and 6 variable map will have 64 squares as it will have 64 min-terms. So the 5 variable K –map will have two blocks (two K –maps four variables each) of 16 squares as shown in figure 4.31. If the 5 variables are ABCDE then variable A will represent that the two K –maps (of four variables BCDE) for $\overline{A}$ and A.

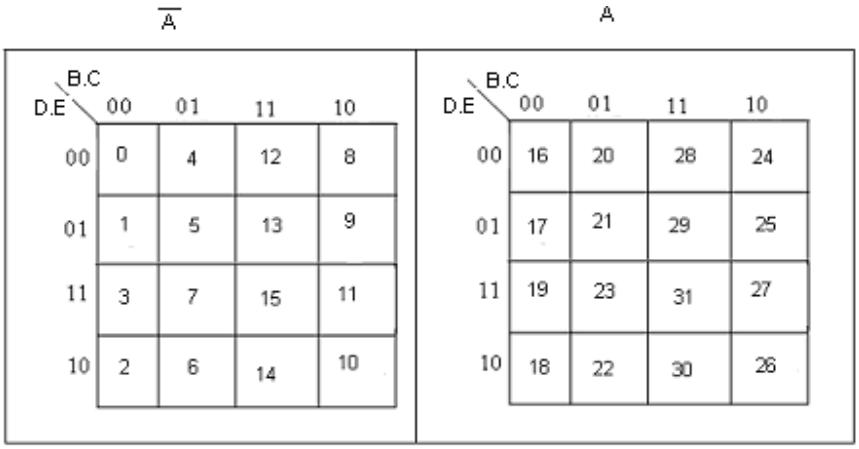


Fig 4.31

Similarly, map of six variables will have four blocks (four K –maps of four variables) of 16 squares each. If six variables are ABCDEF then the four K-maps of four variables (CDEF) will belong to $\overline{A}\cdot\overline{B}$, $\overline{A}\cdot B$, $A\cdot\overline{B}$ & $A\cdot B$ variables as shown in figure 4.32.

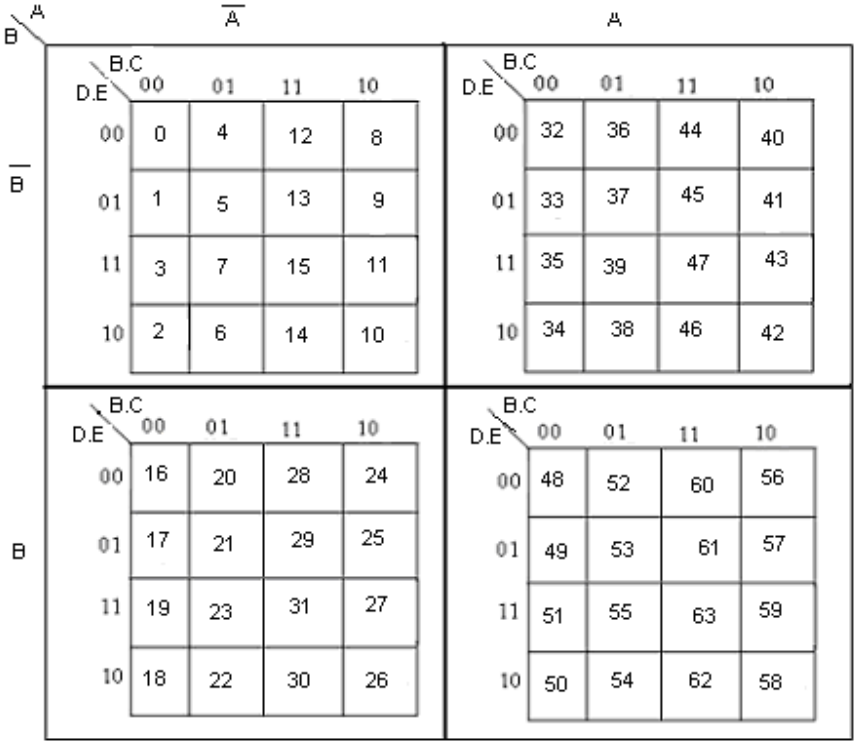


Fig. 4.32

4.5.1 Simplification of Five and Six Variable Maps: It has been discussed that the K –map of five variables has two blocks of 16 squares and the K –map of six variables has four blocks of 16 squares. A pair (two adjacent min-terms) in one block and

other pair in the other adjacent block will said to be adjacent if the positions of two pairs are same in their respective blocks. For example squares 13 & 9 (forming pair of one block) are adjacent to the squares of 29 & 25 (pair of the adjacent block) in five-variable K –map and thus reduces two variables using the same procedure as used in four-variable map. Similarly, a quad (four adjacent min-terms) of one block and other quad of the other adjacent bock will be adjacent if the positions of the two quads are the same in their respective blocks. That is the four squares 19, 23, 31 & 27 (forming quad of one block) are adjacent to the squares 51, 55, 63 & 59 in adjacent block of six-variable map; and these two quads will eliminate three variables. The elements of the diagonal blocks will not be adjacent even if their positions are same. The simplification can be illustrated by using the following two examples.

Example 4.9: Simplify the Boolean function of five variables:

$$F(A,B,C,D,E) = \sum(0,2,3,4,6,7,8,11,12,13,16,18,19,20,22,23,24,27,28,29)$$

Solution: The K-map for five variables is drawn and 1s are entered for the min terms given in the problem and remaining entries are filled with 0s as shown in figure 4.33.

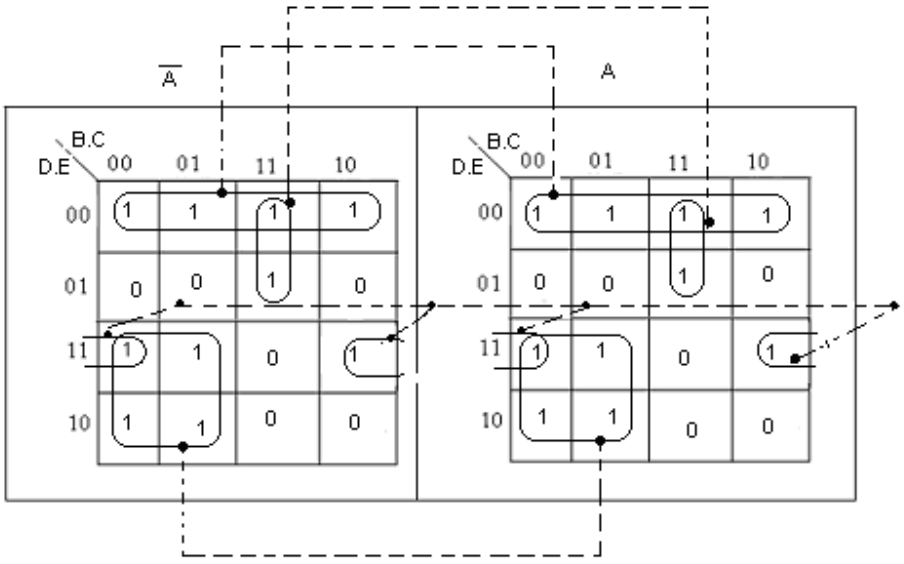


Fig. 4.33

The encircling is done as given in figure 4.33.

The min-terms 0, 4, 12 & 8 of block $\overline{A}$ (forming quad) and the min-terms 16, 20, 28 & 24 of other block A are adjacent, thus giving the term of two variables $\overline{D} \cdot \overline{E}$.

The min-terms 3, 2, 7 & 6 of block $\overline{A}$ (forming quad) and the min-terms 19, 18, 23 & 22 of other block A are adjacent, thus giving the term of two variables $\overline{B} \cdot D$.

The min-terms 12, 13 of block $\overline{A}$ (forming a pair) and the min-terms 28, 29 of other block A are adjacent, thus giving the term of three variables $B \cdot C \cdot \overline{D}$.

Similarly, isolated 1 (min-term 11) of block $\overline{A}$ and isolated 1 (min-term 27) of block A are adjacent, thus gives the term $B \cdot \overline{C} \cdot D \cdot E$.

Now ORing all the terms obtained above, the minimized Boolean expression is given by:

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

Example 4.10: Simplify the Boolean function of six variables:

$F(A, B, C, D, E) = \sum (2,3,6,7,10,11,14,15,18,20,22,24,26,28,30,41,50,52,54,56,58,60,62)$

Solution: Figure 4.34 shows the K –map of the given problem.

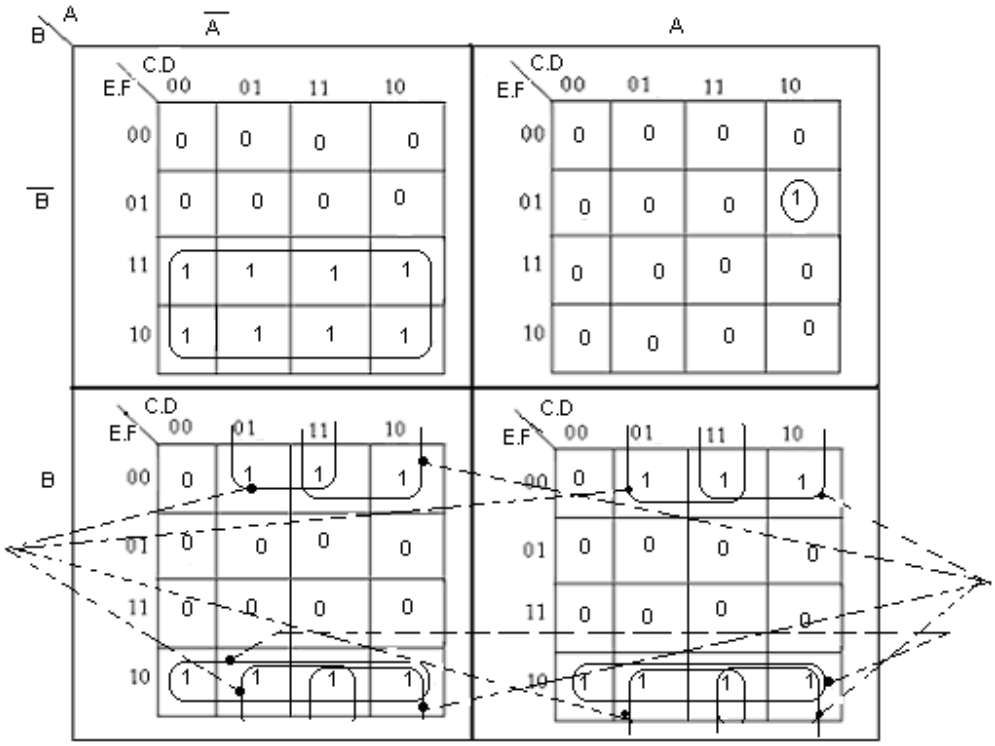


Fig. 4.34

The encircling is done as given in figure 4.34.

The adjacent groups of 1s are shown by dotted lines and the minimized Boolean expression is shown as:

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

4.6 Quine – McCluskey Method : The K –map for the simplification of Boolean function has been discussed in detail. This method was proved to useful tool for the simplification of Boolean function up to four variables. However, this method can be used for more than four variables (five or six variables) but it not very simple to use, since it is difficult to find if the best selection of encircled groups have been made.

A method developed by Quine and improved by McCluskey was found to be good for the simplification of Boolean functions of any number of variables. This method is known as Quine – McCluskey method or Q – M tabular method or simply tabular method.

In mapping it has been observed that any adjacent minterm can be reduced because they differ by only one literal. This is also the fundamental principle of the Q – M tabular method of minimization of Boolean functions.

The following steps are to be used for the minimization of Boolean functions in this method:

Step I All minterms are arranged in groups of same number of 1s in their binary equivalents.

Step II Each term of groups of low number of 1s is compared with every term of groups of higher number of 1s. These terms are then combined. Two terms of adjacent groups are said to be combined, if their binary representation differ by single bit in the same position. The combined terms consist of the original fixed representation with a dash (–) sign at the differing place. A tick mark (√) is placed on the right hand side of every term, which has been combined with at least one term.

Step III Compare and combine the new terms obtained in step II with the terms of groups of higher numbers in the similar fashion. The two terms are combined which differ by only single 1 and whose dashes are in the same positions. This procedure is continued till no further combinations are possible.

Step IV All those terms which remained without tick mark (√) are known as the prime implicants, as they can not be reduced further. Finally the necessary prime implicants are obtained by rejecting those implicants which have been covered in one or more prime implicants. The necessary prime implicants will then give the required Boolean expression.

This method can well be illustrated by considering the following example.

Example 4.11: Simplify the Boolean function given in example 4.9, using Q – M method.

Solution: The given Boolean function is :

$$F(A,B,C,D,E) = \sum (0,2,3,4,6,7,8,11,12,13,16,18,19,20,22,23,24,27,28,29)$$

We follow the steps given in Q – M method and find the prime implicants:

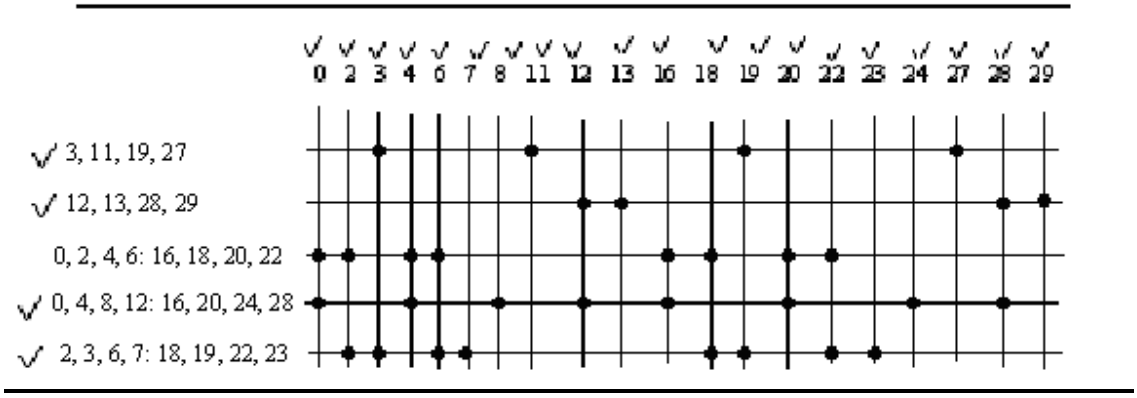
No. of zeros	Min-terms	Binary equivalents	Combination with Binary Nos.	Combination II with binary Nos.
0	0 √	0 0 0 0 0	0,2 √ 0 0 0 – 0	0,2,4,6 √ 0 0 – – 0
			0,4 √ 0 0 – 0 0	0,2,16,18 √ – 0 0 – 0
	2 √	0 0 0 1 0	0,8 √ 0 – 0 0 0	0,4,8,12 √ 0 – – 0 0
1	4 √	0 0 1 0 0	0,16 √ – 0 0 0 0	0,4,16,20 √ – 0 – 0 0
	8 √	0 1 0 0 0		0,8,16,24 √ – – 0 0 0
	16 √	1 0 0 0 0	2,3 √ 0 0 0 1 –	
			2,6 √ 0 0 – 1 0	2,3,6,7 √ 0 0 – 1 –
	3 √	0 0 0 1 1	2,18 √ – 0 0 1 0	2,3,18,20 √ – 0 0 1 –
	6 √	0 0 1 1 0	4,6 √ 0 0 1 – 0	2,6,18,22 √ – 0 – 1 0
2	12 √	0 1 1 0 0	4,12 √ 0 – 1 0 0	4,6,20,22 √ – 0 1 – 0
	18 √	1 0 0 1 0	4,20 √ – 0 1 0 0	4,12,20,28 √ – – 1 0 0
	20 √	1 0 1 0 0	8,12 √ 0 1 – 0 0	8,12,24,28 √ – 1 – 0 0
	24 √	1 1 0 0 0	8,24 √ – 1 0 0 0	16,18,20,22 √ 1 0 – – 0
			16,18 √ 1 0 0 – 0	16,20,24,28 √ 1 – – 0 0
	7 √	0 0 1 1 1	16,20 √ 1 0 – 0 0	
	11 √	0 1 0 1 1	16,24 √ 1 – 0 0 0	3,7,19,23 √ – 0 – 1 1
3	13 √	0 1 1 0 1		3,11,19,27 – – 0 1 1
	19 √	1 0 0 1 1	3,7 √ 0 0 – 1 1	6,7,22,23 √ – 0 1 1 –
	22 √	1 0 1 1 0	3,11 √ 0 – 0 1 1	12,13,28,29 – 1 1 0 –
	28 √	1 1 1 0 0	3,19 √ – 0 0 1 1	18,19,22,23 √ 1 0 – 1 –
			6,7 √ 0 0 1 1 –	
	23 √	1 0 1 1 1	6,22 √ – 0 1 1 0	
4	27 √	1 1 0 1 1	12,13 √ 0 1 1 0 –	
	29 √	1 1 1 0 1	12,28 √ – 1 1 0 0	
			18,19 √ 1 0 0 1 –	
			18,22 √ 1 0 – 1 0	
			20,22 √ 1 0 1 – 0	
			20,28 √ 1 – 1 0 0	
			24,28 √ 1 1 – 0 0	
			7,23 √ – 0 1 1 1	
			11,27 √ – 1 0 1 1	
			13,29 √ – 1 1 0 1	
			19,23 √ 1 0 – 1 1	
			19,27 √ 1 – 0 1 1	
			22,23 √ 1 0 1 1 –	
			28,29 √ 1 1 1 0 –	

Contd.

Contd.

Combination III with binary Nos.	
0,2,4,6:16,18,20,22	– 0 – – 0
0,4,8,12:16,20,24,28	– – – 0 0
2,3,6,7:18,19,22,23	– 0 – 1 –

The prime implicants from this table are those terms which remained without tick mark (√), as they can not be reduced further. Now the essential prime implicants will be obtained as given below:



The essential prime implicants (ticked marked) are represented in the following form:

3,11,19,27	$--011$	$=\overline{C}\cdot D\cdot E$
12,13,28,29	$-110-$	$=B\cdot C\cdot\overline{D}$
0,4,8,12:16,20,24,28	$---00$	$=\overline{D}\cdot\overline{E}$
2,3,6,7:18,19,22,23	$-0-1-$	$=\overline{B}\cdot D$

Thus the required minimized Boolean expression is given by:

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

The result is the sane as obtained in the example 4.9.

PROBLEMS:

1. Discuss K –map for the reduction of Boolean function of 4 variables.
2. Taking a suitable example, verify that a quad eliminates two variables and an octet eliminates three variables in a K - map of four variables.
3. What are pairs quads and octets? What is their importance in K –maps?
4. What are the rules for getting the minimal Boolean function using K – maps? Illustrate with examples.
5. Discuss the redundant groups in K – map.
6. What do you understand by incompletely specified functions how these are used in eliminating the Boolean functions?
7. Discuss K –map method of reduction the Boolean function of five variables.
8. Discuss K –map method of reduction the Boolean function of six five variables.

9. Discuss the Quine – Mccluskey method of reduction of Boolean functions.

10. Simplify the following Boolean functions using K –map and verify your answer using the theorems of Boolean algebra also.

(i) $F_1(a,b,c) = \sum (0,1,4,5,7)$

(ii) $F_2(a,b,c) = \sum (0,1,2,3,4,5,7)$

(iii) $F_3(W,X,Y) = \sum (1,3,6,7)$

Ans.: (i) $F_1 = \overline{b} + a \cdot c$

(ii) $F_2 = \overline{a} + \overline{b} + c$

(iii) $F_3 = W \cdot X + \overline{W} \cdot Y$

11. Simplify the following Boolean functions using K –map and realized the minimized functions with NAND gates only.

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

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

(iii) $f = \overline{W} \cdot \overline{X} \cdot \overline{Y} + \overline{W} \cdot \overline{X} \cdot Y + W \cdot X \cdot \overline{Y} + \overline{W} \cdot X \cdot Y + W \cdot \overline{X} \cdot \overline{Y}$

Ans.: (i) $Z = \overline{A} \cdot \overline{C} + \overline{A} \cdot \overline{B} + A \cdot B \cdot C$

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

(iii) $f = W \cdot \overline{Y} + \overline{X} \cdot \overline{Y} + \overline{W} \cdot Y$

12. Simplify the following Boolean functions using K –map and verify your answer using the theorems of Boolean algebra also.

(i) $f_1 = (A + \overline{B} + \overline{C}) \cdot (\overline{A} + \overline{B} + C) + (\overline{A} + B + C) + (A + \overline{B} + \overline{C})$

(ii) $f_2 = (A + \overline{B} + C) \cdot (A + \overline{B} + \overline{C}) \cdot (\overline{A} + \overline{B} + C)$

(iii) $f_3 = (A + B + C) \cdot (A + \overline{B} + C) \cdot (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C})$

Ans.: (i) $f_1 = (\overline{A} + C) \cdot (\overline{A} + B) \cdot (A + \overline{B} + \overline{C})$

(ii) $f_2 = (A + \overline{B}) \cdot (\overline{B} + C)$

(iii) $f_3 = (A + C) \cdot (\overline{A} + B)$

13. Minimize the following Boolean functions using K –map and then realize them with NOR gates only.

(i) $Z(a,b,c) = \prod (1,2,3,4,6,7)$

(ii) $Z(a,b,c) = \prod (0,2,3,4,5,6)$

(iii) $Z(a,b,c) = \prod (0,1,2,3,5,6)$

Ans.: (i) $Z = \overline{b} \cdot (\overline{a} + c) \cdot (a + \overline{c})$

(ii) $Z = c \cdot (a + \overline{b}) \cdot (\overline{a} + b)$

(iii) $Z = a \cdot (b + \overline{c}) \cdot (\overline{b} + c)$

14. Get the minimal Boolean functions of the following using K – map:

(i) $f_1(A,B,C) = \sum (0,2,5) + \sum_{\phi} (1,3,6)$

(ii) $f_2(X, Y, Z) = \sum (1,4) + \sum_{\phi} (0,5,6,7)$

(iii) $f_3(a, b, c) = \sum (5,6) + \sum_{\phi} (0,1,2,3)$

Ans.: (i) $f_1 = \overline{A} + \overline{B} \cdot C$

(ii) $f_2 = X + \overline{Y}$

(iii) $f_3 = \overline{b} \cdot c + b \cdot \overline{c}$

15. Obtain the minimal SOP expression of the following functions and implement them using NAND gates only.

(i) $F_1(A, B, C, D) = \sum (0,1,4,6,9,13,14,15)$

(ii) $F_2(A, B, C, D) = \sum (0,1,2,3,4,7,8,9,13,14,15)$

(iii) $F_3(A, B, C, D) = \sum (0,2,3,5,6,8,9,11,12,14,15)$

(iv) $F_4(A, B, C, D) = \sum (0,4,5,8,10,12,15)$

(v) $F_5(A, B, C, D) = \sum (1,2,3,5,7,15)$

Ans.: (i) $F_1 = \overline{A} \cdot \overline{C} \cdot \overline{D} + \overline{B} \cdot \overline{C} \cdot D + A \cdot B \cdot D + B \cdot C \cdot \overline{D}$

(ii) $F_2 = \overline{A} \cdot \overline{B} + \overline{C} \cdot \overline{D} + A \cdot B + A \cdot \overline{C} + B \cdot C \cdot D$

(iii) $F_3 = \overline{A} \cdot \overline{B} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} + B \cdot C \cdot \overline{D} + A \cdot B \cdot C + \overline{A} \cdot B \cdot \overline{C} \cdot D$

(iv) $F_4 = \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} + A \cdot B \cdot C \cdot D$

(v) $F_5 = \overline{A} \cdot D + B \cdot C \cdot D + \overline{A} \cdot \overline{B} \cdot C$

16. Obtain the minimal Boolean functions of the following, using K – map:

(i) $F_1(A, B, C, D) = \sum (0,2,3,5,6,7,8,9) + \sum_{\phi} (10,11,12,13,14,15)$

(ii) $F_2(A, B, C, D) = \sum (0,1,2,3,4,7,8,9) + \sum_{\phi} (10,11,12,13,14,15)$

(iii) $F_3(A, B, C, D) = \sum (1,2,3,5,13) + \sum_{\phi} (6,7,8,9,11,15)$

(iv) $F_4(A, B, C, D) = \sum (2,9,10,12,13) + \sum_{\phi} (1,5,14,15)$

(v) $F_5(A, B, C, D) = \sum (0,1,5,8,10,14) + \sum_{\phi} (2,11,15)$

Ans.: (i) $F_1 = A + C + B \cdot D + \overline{B} \cdot \overline{D}$

(ii) $F_2 = \overline{B} + C \cdot D + \overline{C} \cdot \overline{D}$

(iii) $F_3 = \overline{A} \cdot C + D$

(iv) $F_4 = \overline{C} \cdot D + A \cdot B + \overline{B} \cdot C \cdot \overline{D}$

(v) $F_5 = \overline{B} \cdot \overline{D} + A \cdot C + \overline{A} \cdot \overline{C} \cdot D$

17. Using K – map, obtain the minimal POS expressions of the following and implement them with NOR gates only.

(i) $F_1(A,B,C,D) = \prod (0,1,2,4,5,9,10,11,12,13,14,15)$

(ii) $F_2(A,B,C,D) = \prod (2,3,5,7,9,11,12,13,14,15)$

(iii) $F_3(A,B,C,D) = \prod (0,1,2,3,4,5,8,10,13,14)$

(iv) $F_4(A,B,C,D) = \prod (0,1,2,4,6,7,8,9,11,13)$

(v) $F_5(A,B,C,D) = \prod (1,5,6,7,11,12,13,15)$

Ans.: (i) $F_1 = (C+\overline{D})\cdot(\overline{A}+\overline{C})\cdot(\overline{A}+\overline{B})\cdot(A+C)\cdot(A+B+D)$

(ii) $F_2 = (\overline{A}+\overline{B})\cdot(\overline{B}+\overline{D})\cdot(\overline{A}+\overline{D})\cdot(A+B+\overline{C})$

(iii) $F_3 = (A+B)\cdot(B+D)\cdot(A+C)\cdot(\overline{A}+\overline{C}+D)\cdot(\overline{B}+C+D)$

(iv) $F_4 = (A+D)\cdot(B+C)\cdot(\overline{A}+C+\overline{D})\cdot(\overline{A}+B+\overline{D})$

(v) $F_5 = (\overline{A}+\overline{B}+C)\cdot(A+\overline{B}+\overline{C})\cdot(A+C+\overline{D})\cdot(\overline{A}+\overline{C}+\overline{D})$

18. Using K – map, obtain the minimal POS expressions of the following and implement them with NOR gates only.

(i) $F_1(A,B,C,D) = \prod (1,5,6,12,13,14)\cdot\prod_{\phi}(2,4)$

(ii) $F_2(A,B,C,D) = \prod (2,3,7,10,11,14,15)\cdot\prod_{\phi}(1,5,12)$

(iii) $F_3(A,B,C,D) = \prod (2,3,5,6,7,8)\cdot\prod_{\phi}(10,11,12,13,14,15)$

(iv) $F_4(A,B,C,D) = \prod (2,4,5,7,9,12)\cdot\prod_{\phi}(0,1,6)$

(v) $F_5(A,B,C,D) = \prod (0,1,3,4,5,6,7,14,15)\cdot\prod_{\phi}(2,9,13)$

Ans.: (i) $F_1 = (\overline{B}+D)\cdot(\overline{B}+C)\cdot(A+C+\overline{D})$

(ii) $F_2 = (\overline{A}+\overline{C})\cdot(B+\overline{C})\cdot(\overline{C}+\overline{D})$

(iii) $F_3 = \overline{C}\cdot(\overline{B}+\overline{D})$

(iv) $F_4 = (A+\overline{B})\cdot(A+D)\cdot(\overline{B}+C+D)$

(v) $F_5 = A\cdot(\overline{B}+\overline{C})$

19. Minimize the following functions using K – map method.

(i) $F_1(A,B,C,D,E) = \sum (0,1,2,3,4,7,8,9,14,15,16,17,18,19,25,31)$

(ii) $F_2(A,B,C,D,E) = \sum (4,5,6,7,10,12,13,14,15,20,21,22,23,29) + \sum_{\phi}(0,1,2,24,25,26)$

(iii) $F_3(A,B,C,D,E) = \sum (1,2,4,6,7,9,10,11,16,17,19,20,22,23) + \sum_{\phi}(12,13,24,28,29,30,31)$

(iv) $F_4(A,B,C,D,E,F) = \prod (0,1,2,4,5,7,8,9,10,14,15,17,19,20,28,29,34,36,40,41,42,43)$

(v) $F_5(A,B,C,D,E,F) = \sum (0,2,4,,8,10,13,15,16,18,20,24,26,32,34,40,42,45,47,48,50,56,57,58,60,61)$

(vi) $F_6(A,B,C,D,E) = \sum (4,5,9,10,11,12,13,14,15,16,18,19,20,21,24,25,26,27,29,30,31)$

Ans.:(i) $F_1 = \overline{B}\cdot\overline{C} + \overline{C}\cdot\overline{D} + \overline{A}\cdot\overline{B}\cdot\overline{D}\cdot\overline{E} + \overline{A}\cdot B\cdot C\cdot D + \overline{A}\cdot B\cdot D\cdot E + B\cdot C\cdot D\cdot E + B\cdot\overline{C}\cdot\overline{D}\cdot E$

(ii) $F_2 = \overline{A}\cdot C + \overline{B}\cdot C + \overline{A}\cdot B\cdot D\cdot\overline{E}$

(iii) $F_3 = C\cdot\overline{D}\cdot\overline{E} + \overline{B}\cdot\overline{C}\cdot\overline{D}\cdot E + \overline{B}\cdot C\cdot D + \overline{A}\cdot\overline{B}\cdot D\cdot\overline{E} + \overline{A}\cdot B\cdot\overline{C}\cdot E + A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D} + A\cdot\overline{B}\cdot C\cdot D + A\cdot B\cdot\overline{C}\cdot D$

(iv) $F_4 = (A+B+C+E)\cdot(B+\overline{C}+D+E)\cdot(\overline{A}+B+\overline{C}+D)\cdot(B+D+\overline{E}+F)\cdot$
$(A+B+\overline{C}+\overline{D}+\overline{E})\cdot(A+B+\overline{D}+\overline{E}+\overline{F})\cdot(A+\overline{B}+C+D+\overline{F})\cdot$
$(A+\overline{B}+\overline{C}+\overline{D}+E)\cdot(B+C+\overline{D}+E+F)$

(v) $F_5 = \overline{D}\cdot\overline{F} + \overline{B}\cdot C\cdot D\cdot F + \overline{A}\cdot\overline{C}\cdot\overline{E}\cdot\overline{F} + A\cdot B\cdot C\cdot\overline{E}$

(vi) $F_6 = B\cdot D + B\cdot E + \overline{B}\cdot C\cdot\overline{D} + A\cdot\overline{C}\cdot\overline{E} + A\cdot\overline{C}\cdot D + \overline{A}\cdot B\cdot C$

20. Minimize the functions given in problem 19 using Q –M tabular method.
21. Minimize the functions given in problem 16 using Q –M tabular method.
22. Minimize the functions given in problem 15 using Q –M tabular method.

_____

5

Combinational Switching Circuits

In the forgoing chapters of this book, detailed study of the Boolean algebra and various methods of simplification of Boolean functions have been made. The different logic gates may be used to implement the simplified Boolean functions. In the present chapter, however, the design of the special class of logic circuits for digital systems known as combinational switching circuits will be discussed. Basically there are two types of switching circuits namely the combinational and sequential switching circuits. The combinational circuits depend on the verbal statement of the problem. That is the input and output variables are obtained from the given statement; which then lead to provide the minterms for simplification and implementation of the logic circuits. The sequential switching circuits will be discussed in a later chapter.

5.1 Combinational Circuits: The combinational circuits are the network of logic gates having a set of input independent variables, and outputs as the Boolean functions of inputs. In these circuits the independent input variables are obtained from the word statement of the requirement of the digital system to be designed. The output variables in these circuits depend only on the present value of the inputs and do not depend upon their previous values. That is the combinational logic circuits need not to have the memory elements. The other class of the switching circuits called sequential circuits do have the memory elements in addition to the input and output variables. Figure 5.1 illustrates the input – output relationship of the combinational circuits.

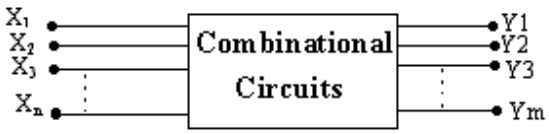


Fig. 5.1

The output variables Y_1,Y_2,Y_3 ….Y_m are some functions of the input variables X_1,X_2,X_3 …X_n such that :

$$Y_1 = F_1(X_1, X_2, X_3.....X_n)$$

$$Y_2 = F_2(X_1, X_2, X_3.....X_n)$$

……..

……..

$$Y_m = F_1(X_1, X_2, X_3.....X_n)$$

The procedure for the design of the combinational logic circuit is given below:

- From the word statement of the problem input independent variables and output dependent variables are isolated.
- The logical symbols as well as the logic values (0 or 1) are assigned to these variables.
- The truth table is formed between the required output variables and the given input variables.
- Using K – map or Q – M tabular method, the simplified Boolean function for each output variables is obtained.
- The logic circuit is then drawn using the gates for each output variables.

A few examples for the design of the combinational circuits will now be discussed.

Example 5.1: A railway station has four platforms marked as P_1, P_2, P_3 and P_4 as shown in the figure 5.2. The trains can come only from left hand side and enter these platforms. The trains are to be routed to these platforms in the order of preference P_1, P_2, P_3 and in the last to P_4. Each platform has a switch will be turned ON if the platform is not empty. There is an outer signal S which will be either green or red. This signal will be green if it allows the train to enter the station otherwise red. There are three track changer switches T_1, T_2, T_3 which allows changing the tracks. Design a railway track switching circuit using AND, OR and NOT gates, which can perform the operations mentioned above.

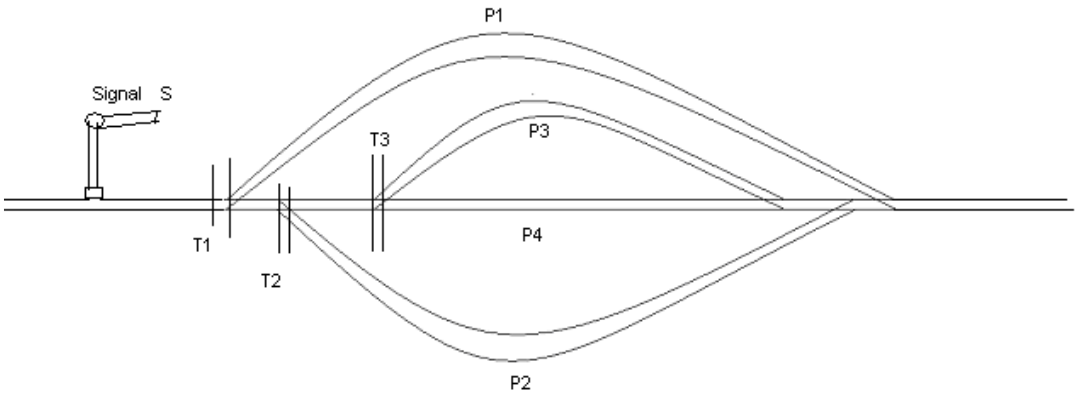


Fig. 5.2

Solution: From the word statement of the problem it is clear that P1, P2, P3 and P4 are the four input variables, outer signal S and track changers T1, T2 and T3 are the four output variables. The input as well as output variables are two valued functions, since the platforms are either empty or occupies, track changers are either to be changed or not to be changed, similarly the outer signal S has two options that it is either green or red. The switching system having input and output variables is shown in figure 5.3.

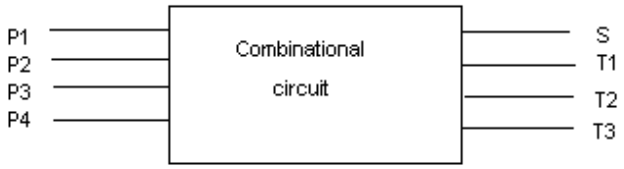


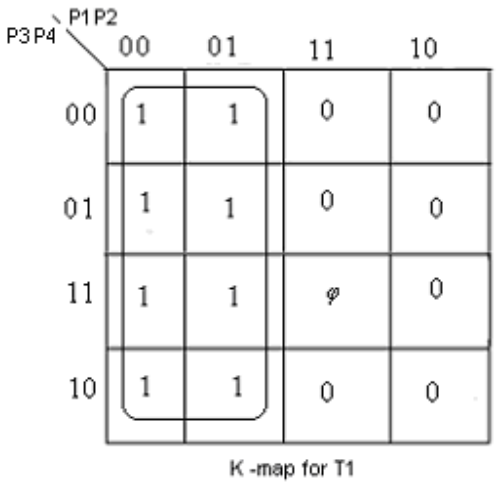
Fig. 5.3

Now the logic values are assigned to the input and output variables. Logic 0's are assigned to the platforms P1, P2, P3 & P4 if these are empty otherwise logic 1. The track changer T1 is not to be changed if the train is allowed to enter P1 otherwise it is to be changed. So logic 1 is assigned to T1 if track is not to be changed and logic o if the track is to be changed. Similarly logic values are assigned to the other track changers. The Signal S is assigned logic 1 to the green signal and logic 0 to the red signal.

The truth table will be drawn for all the input and output variables as given in table 5.1. Also the K-maps for the output variables are drawn as shown in figure 5.4.

Table 5.1

Input Variables				Output Variables			
P1	P2	P3	P4	S	T1	T2	T3
0	0	0	0	1	1	φ	φ
0	0	0	1	1	1	φ	φ
0	0	1	0	1	1	φ	φ
0	0	1	1	1	1	φ	φ
0	1	0	0	1	1	φ	1
0	1	0	1	1	1	φ	1
0	1	1	0	1	1	φ	0
0	1	1	1	1	1	φ	φ
1	0	0	0	1	0	1	φ
1	0	0	1	1	0	1	φ
1	0	1	0	1	0	1	φ
1	0	1	1	1	0	1	φ
1	1	0	0	1	0	0	1
1	1	0	1	1	0	0	1
1	1	1	0	1	0	0	0
1	1	1	1	0	φ	φ	φ



K -map for T1

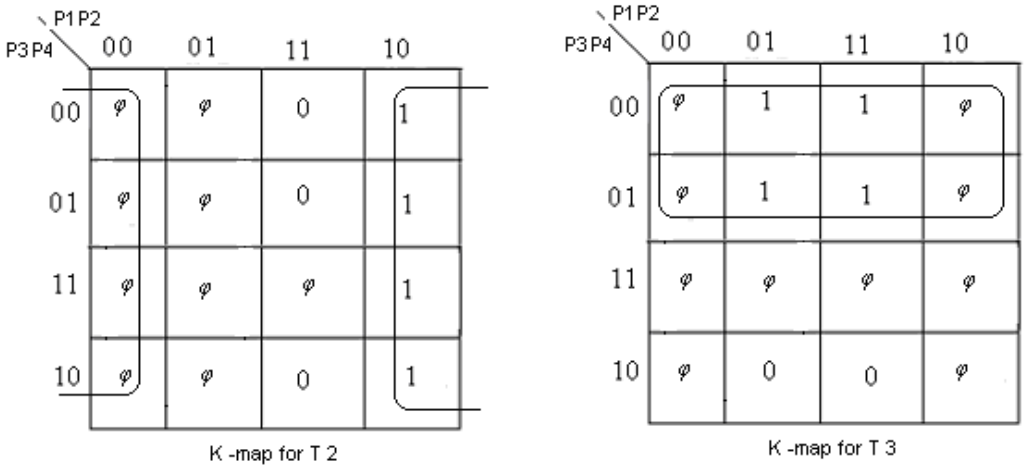


Fig. 5.4

The Boolean expressions for S can directly be obtained as:

$$S = \overline{P1} + \overline{P2} + \overline{P3} + \overline{P4}$$

$$= \overline{P1 \cdot P2 \cdot P3 \cdot P4}$$

The expressions for T1, T2 and T3 are obtained from their respective K –map as:

$$T1 = \overline{P1}$$
$$T2 = \overline{P2}$$
$$T3 = \overline{P3}$$

The switching circuit for the railway track circuit is given in figure 5.5.

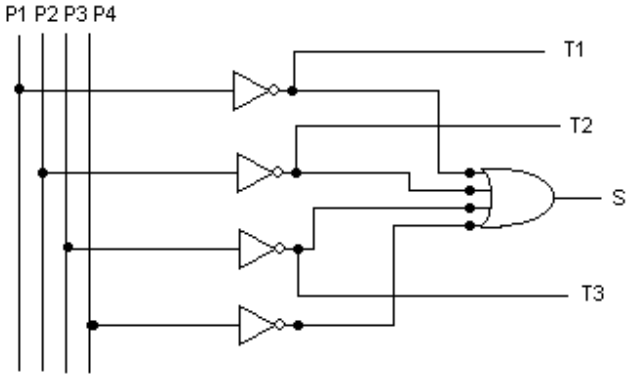


Fig. 5.5

Example 5.2: Design the combinational logic circuit using NAND gates only for the following word statement.

The insurance policy will be issued to the applicant, if he is:

(i) a married female of 22 years or more, or

(ii) a female under 22 years, or

(iii) a married male under 22 years and who has not been involved in a car accident, or

(iv) a married male who has been involved in a car accident, or

(v) a married male of 22 years old or above and who has not been involved in a car accident.

Design the circuit which can issue the insurance policy to the applicant.

Solution: From the word statement of the problem that it has four input variables and one output variables.

The input variables are

(i) The applicant is married or not –we assign the symbol X for it. Logic 1 is assigned to X if the applicant is married otherwise assign logic 0.

(ii) The applicant is male or not – assign the symbol Y for it. Logic 1 is assigned to Y if the applicant is male and logic 0 to female.

(iii) The applicant is 22 years old or more – assign the symbol Z for it. Logic 1 is assigned if the applicant is below 22 years and logic 0 is assigned if the applicant is 22 years old or more.

(iv) The applicant is involved in a car accident- assign the symbol W for it. Logic 1 is assigned to W if the applicant has involved in a car accident otherwise W is assigned logic 0.

Output is the policy issued to the applicant. Let P is the symbol for the policy. Logic 1 is assigned to P if the poly is issued to the applicant otherwise P is assigned logic 0.

The switching system having input and output variables is shown in figure 5.6. Table 5.2 shows the truth table for all the conditions discussed above. The K-map for the output variable P is shown in figure 5.7.

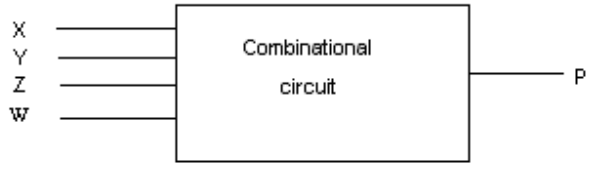


Fig. 5.6

Table 5.2

Inputs				Output
X	Y	Z	W	P
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

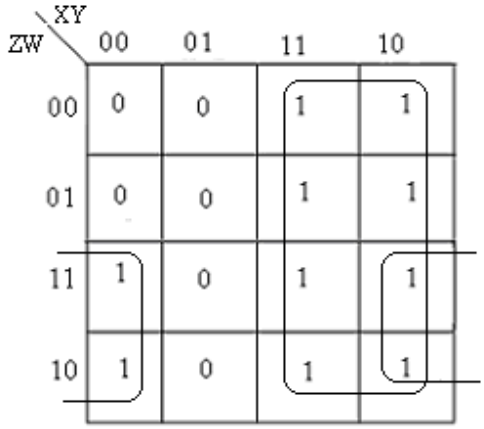


Fig. 5.7

The Boolean expression for the output variable P is given as:

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

From this expression it is clear that the policy will be issued to the applicant who is married or a female under 22 years. The circuit will be realized using NAND gates as shown in figure 5.8.

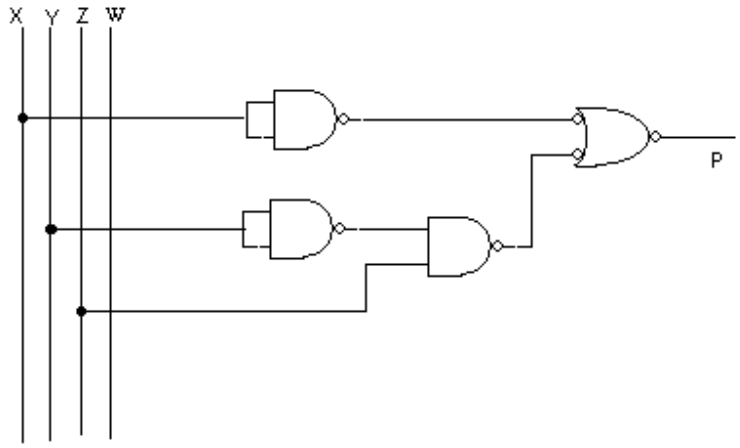


Fig. 5.8

Example 5.3: The entrance to a group of four flats has a tube light. The tube light is to be switched ON and OFF independently by the tenants of the four flats using switches located in their flats. Design a switching circuit to implement this using:

(i) Exclusive – OR gates.

(ii) NAND gates only.

Solution: The input variables to this problem are the four switches each located in the flats of four tenants. Let these switches are S1, S2, S3, S4, which are two valued functions. Logic 0 is assigned to OFF position of the switch and logic 1 ON position of the switch. The output variable is the tube light L, which will either glow or not glow. Logic 1 is assigned if the tube light glows and logic 0 is assigned if it does not glow. So the switching circuit to be designed has four input variables (four switches) and one output variable L (tube light) as shown in figure 5.9.

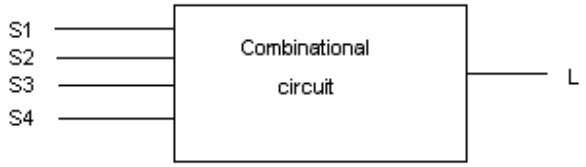
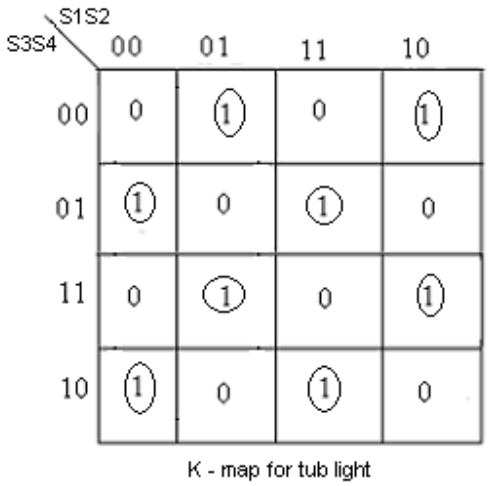


Fig. 5. 9

Table 5.4 shows the values of the output variable for each possible combination of input variables. The K – map is drawn for this table as given in figure 5.10.

Table 5.4

Input Variables				Output
S1	S2	S3	S4	L
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0



K - map for tub light

Fig. 5.10

The Boolean expression for the tube light L is given by:

$$L = \overline{S1}\cdot\overline{S2}\cdot\overline{S3}\cdot S4 + \overline{S1}\cdot\overline{S2}\cdot S3\cdot\overline{S4} + \overline{S1}\cdot S2\cdot\overline{S3}\cdot\overline{S4} + \overline{S1}\cdot S2\cdot S3\cdot S4 +$$
$$S1\cdot\overline{S2}\cdot\overline{S3}\cdot\overline{S4} + S1\cdot\overline{S2}\cdot S3\cdot S4 + S1\cdot S2\cdot\overline{S3}\cdot S4 + S1\cdot S2\cdot S3\cdot\overline{S4}$$
$$= \overline{S1}\cdot\overline{S2}\cdot(\overline{S3}\cdot S4 + S3\cdot\overline{S4}) + \overline{S1}\cdot S2\cdot(\overline{S3}\cdot\overline{S4} + S3\cdot S4) +$$
$$S1\cdot\overline{S2}\cdot(\overline{S3}\cdot\overline{S4} + S3\cdot S4) + S1\cdot S2\cdot(\overline{S3}\cdot S4 + S3\cdot\overline{S4})$$

$$= (\overline{S3} \cdot S4 + S3 \cdot \overline{S4}) \cdot (\overline{S1} \cdot \overline{S2} + S1 \cdot S2) + (\overline{S3} \cdot \overline{S4} + S3 \cdot S4) \cdot (\overline{S1} \cdot S2 + \overline{S1} \cdot S2)$$

$$= (S3 \oplus S4) \cdot (\overline{S1 \oplus S2}) + (\overline{S3 \oplus S4}) \cdot (S1 \oplus S2)$$

$$= (S1 \oplus S2) \oplus (S3 \oplus S4)$$

$$= S1 \oplus S2 \oplus S3 \oplus S4$$

From this expression it is clear that the tube light will be ON when any one of the four switches is ON or any three switches are ON. Similarly, the tube light will be OFF when all the switches are OFF or any two are OFF or all the switches are OFF.

(i) The circuit can be realized using exclusive -OR gates as shown in figure 5.11.

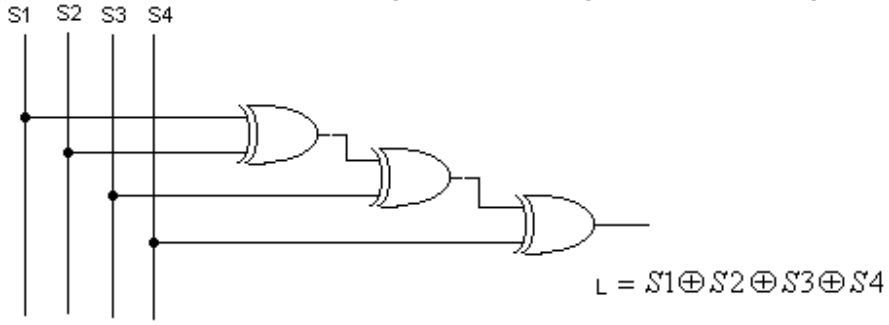


Fig. 5.11

(ii) The circuit realized with NAND gates is shown in figure 5.12.

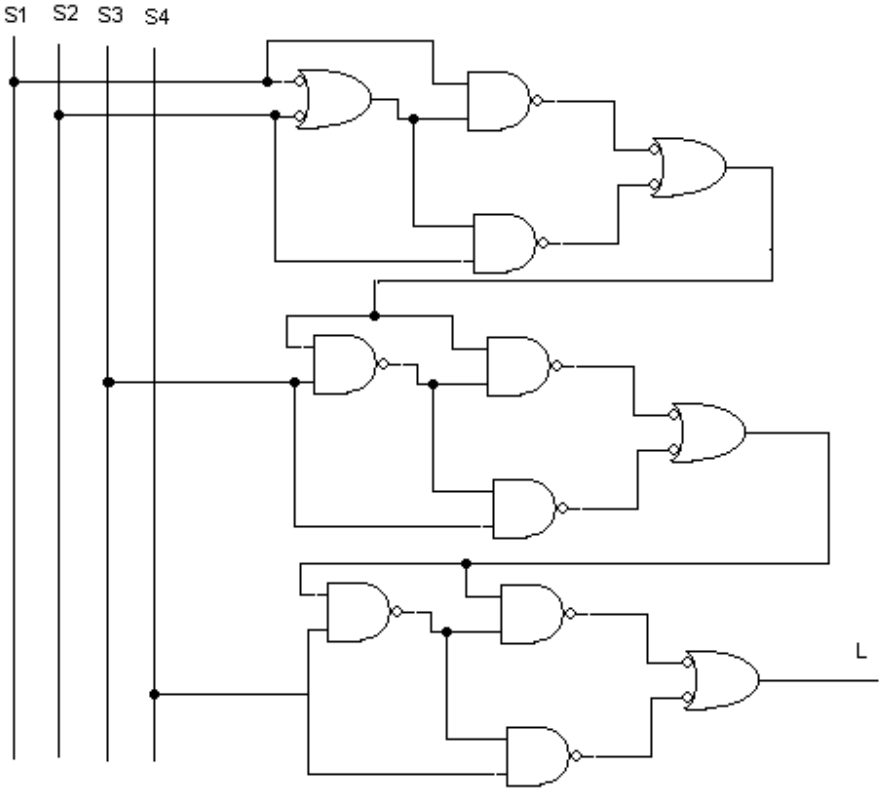


Fig. 5.12

Example 5.4: There are five board of directors (A, B, C, D, E) of a company. The board of director A owns 10% shares, B owns 30% shares, C owns 20% shares, D owns 25% shares and E 15% shares of the total shares. For the adoption of the particular policy to be passed in the board's meeting more than 66% should vote in favour of the policy. The weightage to the votes depend upon the percentage shares owned by the directors. In the board's room each director has a switch which he turns ON if votes in favour of policy. Design a switching circuit to ring a bell if policy is accepted in the board's meeting. Only the NAND gates should be used to realize the circuit.

Solution: From this problem it is clear that there are five input variables and one output variable. A, B, C, D and E, five switches of the board of directors are the input variables to which logic 1 is assigned if the switch is turned ON otherwise logic 0. Similarly, the output variable R is for bell to which logic 1 is assigned if it rings otherwise logic 0.

The switching system having input and output variables is shown in figure 5.13. Table 5.5 shows the truth table for all the conditions discussed above.

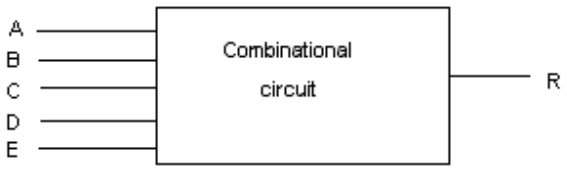


Fig. 5.13

Table 5.5

A	B	C	D	E	R
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	0
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1

Since there are five input variables so the five variable K-map for the output variable R is shown in figure 5.14.

The expression for the alarm R is given by:

$$R = B \cdot D \cdot E + B \cdot C \cdot D + A \cdot B \cdot C \cdot E + A \cdot C \cdot D \cdot E$$

This expression indicates that in order to pass a policy (ring the alarm), the board of directors BDE or BCD or ABCE or ACDE should vote in favour of policy.

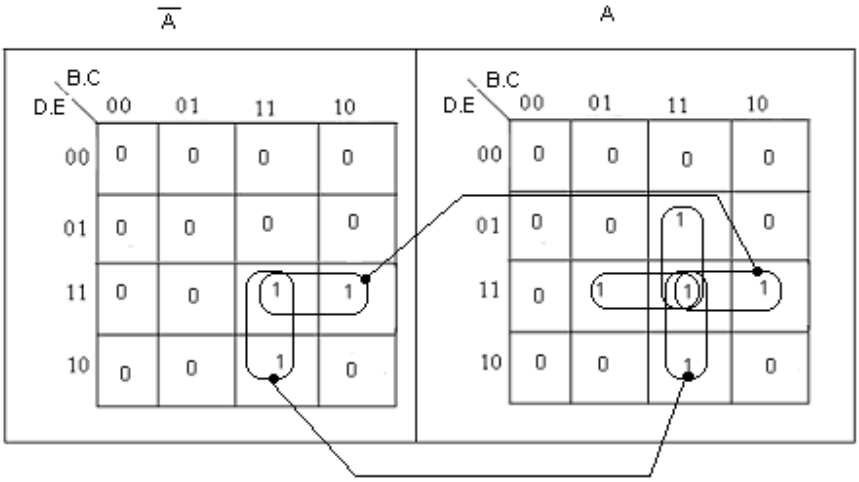


Fig. 5.14

The realization of this circuit with NAND gates is shown in figure 5.15.

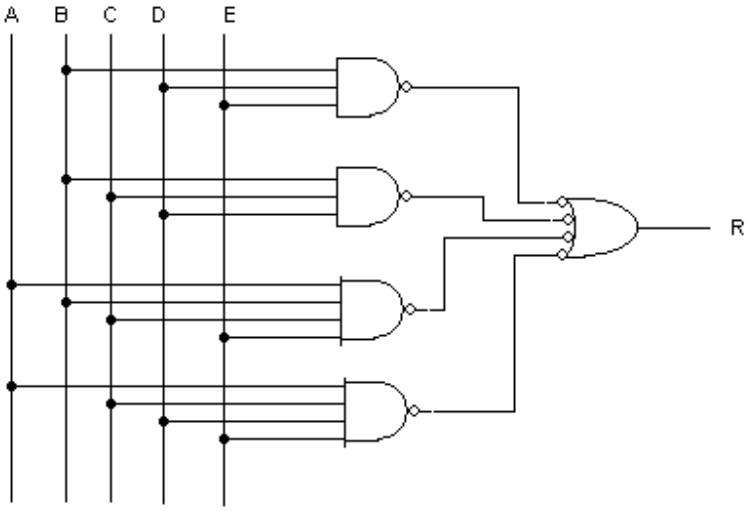


Fig. 5.15

Example 5.5: Design a combinational circuit which multiplies two 3-bit binary numbers $a_2a_1a_0$ and $b_2b_1b_0$, the bits a_2 and b_2 are the sign bits for the two numbers. The five bit output $x_4x_3x_2x_1x_0$ should have the right sign indicating by x_4 and right magnitude $x_3x_2x_1x_0$.

Solution: The logic circuit to be designed has the six input variables and five output variables as shown in figure 5.16. The sign bit a_2, b_2 (input bits) and x_3 (output bit) are assigned logic 0 if these are positive and logic 1 if negative. The other input output variables will have the usual logic values.

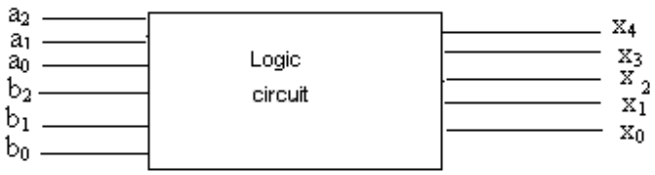


Fig. 5. 16

It is well known that multiplication of two positive numbers or two negative numbers is positive and multiplication one positive number and other negative number is negative. Truth table 5.6 shows the outcome of two different sign bits. The Boolean expression for sign bit X_4 in terms of input sign bits is given by:

Table 5.6

a_2	b_2	x_4
0	0	0
0	1	1
1	0	1
1	1	0

$$x_4 = \overline{a_2} \cdot b_2 + a_2 \cdot \overline{b_2} = a_2 \oplus b_2$$

The table 5.7 shows all possible combinations of the input and output variables.

The Boolean expressions for x_2, x_1, x_0 are obtained from the K-maps drawn for each variable as shown in figure 5.17. However, the expression for x_3 may directly be obtained from the truth table as given below:

Table 5.7

a_1	a_0	b_1	b_0	x_3	x_2	x_1	x_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

The Boolean expressions for x_2, x_1, x_0 are obtained from the K-maps drawn for each variable as shown in figure 5.17. However, the expression for x_3 may directly be obtained from the truth table as given below:

$$x_3 = a_1 \cdot a_0 \cdot b_1 \cdot b_0$$

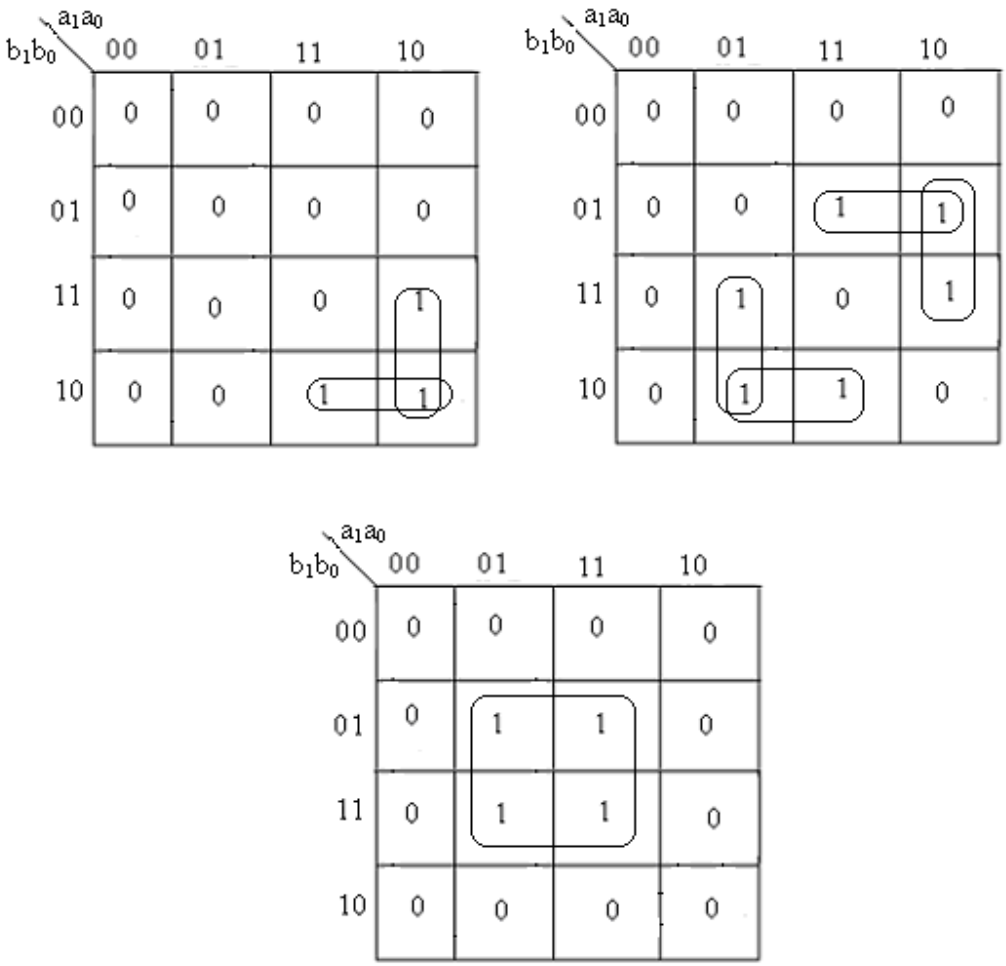


Fig. 5.17

$$x_2 = a_1 \cdot \overline{a_0} \cdot b_1 + a_1 \cdot b_1 \cdot \overline{b_0}$$
$$x_1 = a_1 \cdot \overline{a_0} \cdot b_0 + a_1 \cdot \overline{b_1} \cdot b_0 + \overline{a_1} \cdot a_0 \cdot b_1 + \overline{b_0} \cdot a_0 \cdot b_1$$
$$x_0 = a_0 \cdot b_0$$

The realization of these expressions with And, OR and Not gates is shown in figure 5.18.

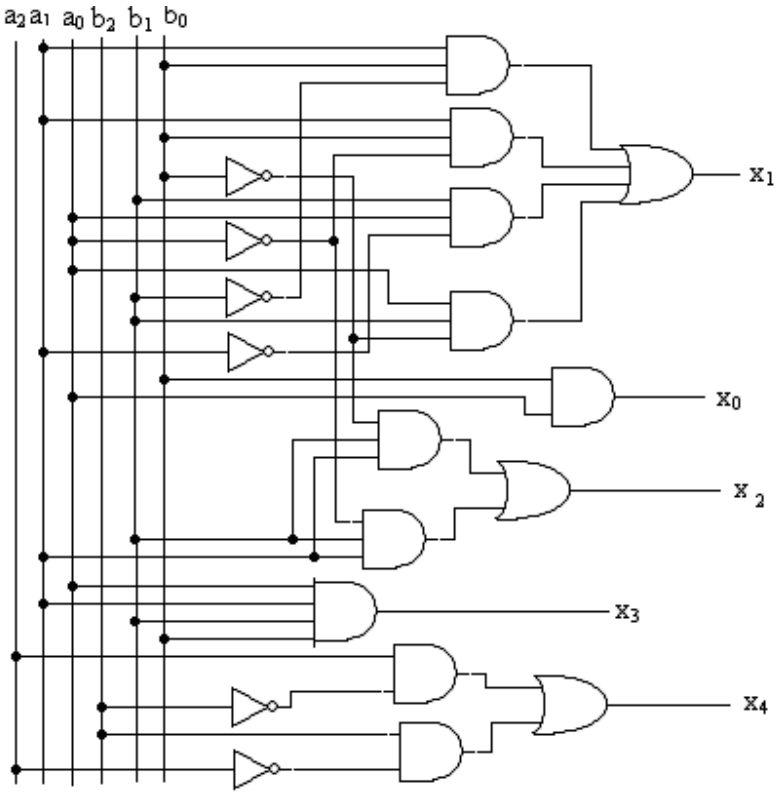


Fig. 5.18

5.2 Half Adder: A half adder is one which adds two binary digits simultaneously. It also falls in the category of combinational circuits. Let A and B are the two binary digits which are to be added together and are the two valued input variables. It will give two outputs as Sum and Carry. It is recalled that when a binary digit 0 is added with 0 the sum is 0 and it will have no carry. If 0 is added with 1 sum is 1 and no carry. Similarly 1 is added with 1 sum is 0 and it will have a carry as 1. The table 5.8 shows the truth table for half adder.

Table 5.8

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

The Boolean expressions for Sum S and Carry C are given by:

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

The expression for sum S is nothing but the exclusive OR function of the two input digits A and B. Figure 5.19 shows the circuit diagram for half adder using exclusive – OR and AND gates.

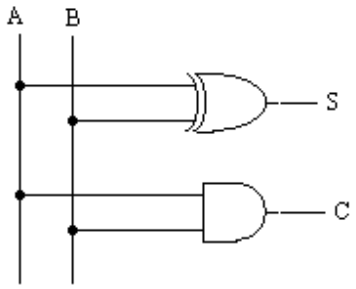


Figure 5.19

The above circuit may also be realized by using NAND gates only as given in figure 5.20.

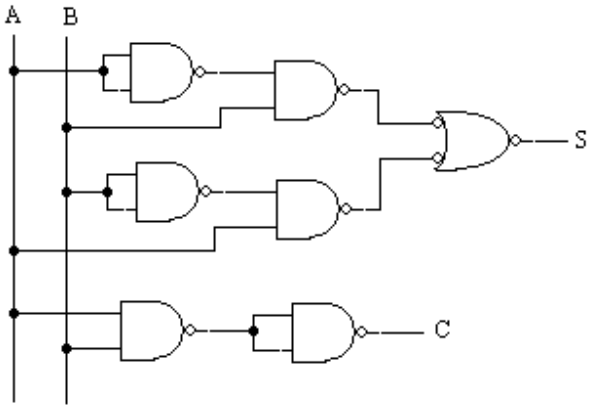


Figure 5.20

This circuit utilizes 7 NAND gates for its realization. The circuit may further be modified to realize it using 5 NAND gates only as given in figure 5.21. The modification of the circuit is not straight forward but can only be modified by inspection. The symbolic representation of the half adder is given in figure 5.22.

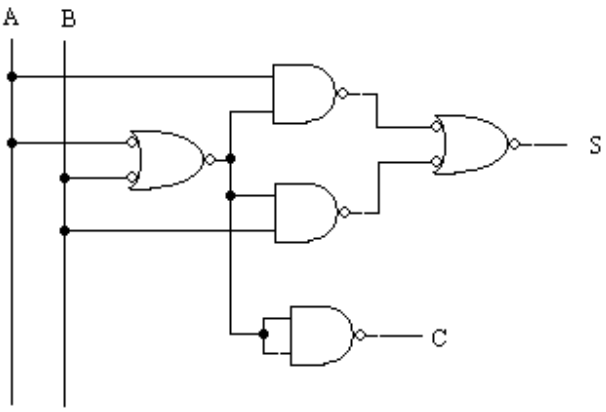


Figure 5.21

Figure 5.22

Example 5.1: Design the half adder using NOR gates only.

Solution: Since the half adder is to be designed with NOR gates only, so the expressions for sum S is obtained in POS form as given below (using the half adder table 5.8).

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

and the carry C is: $C = A \cdot B$

The realization of these expressions for S and C is shown in figure 5.23.

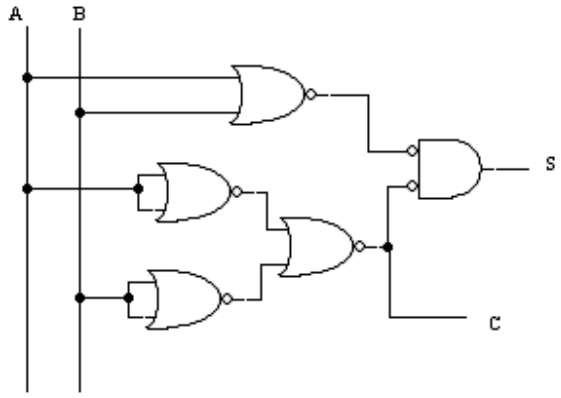


Figure 5.23

5.3 Full Adder: When two binary numbers of two bits are added (A_1A_0 and B_1B_0), then first A_0 and B_0 are added, and the sum S_0 and carry C_0 to the next bit are obtained. A half adder is used for this. For the addition of A_1 B_1 bits, there may be a third bit known as carry bit from the previous column. The result will be the sum S_1 and the carry to the next bit C_1. The addition of three bits is known as full adder.

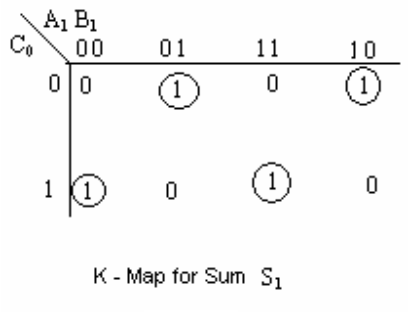
$$\begin{array}{cccc} C_1 & C_0 & & \\ & A_1 & A_0 & \\ & B_1 & B_0 & \\ \hline C_1 & S_1 & S_0 & \end{array}$$

The truth table for Full adder is shown in table 5.9

Table 5.9

A_1	B_1	C_0	S_1	C_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The minimal Boolean expression for S_1 and C_1 is obtained using K – map. The K –map for S_1 and is given in figure 5.24.



K - Map for Sum S_1

Fig. 5.24

The expression for sum S_1 is given by:

$$S_1 = \overline{A}_1 \cdot \overline{B}_1 \cdot C_0 + \overline{A}_1 \cdot B_1 \cdot \overline{C}_0 + A_1 \cdot B_1 \cdot C_0 + A_1 \cdot \overline{B}_1 \cdot \overline{C}_0$$
$$= (\overline{A}_1 \cdot \overline{B}_1 + A_1 \cdot B_1) \cdot C_0 + (A_1 \cdot B_1 + A_1 \cdot B_1) \cdot \overline{C}_0$$
$$= (\overline{A_1 \oplus B_1}) \cdot C_0 + (A_1 \oplus B_1) \cdot \overline{C}_0$$
$$= A_1 \oplus B_1 \oplus C_0$$

It is recalled that $A_1 \oplus B_1$ is the sum of half adder S so S_1 is further given by:

$$S_1 = S \cdot \overline{C}_0 + \overline{S} \cdot C_0 = S \oplus C_0$$

The K –map for carry C_1 is given in figure 5.25

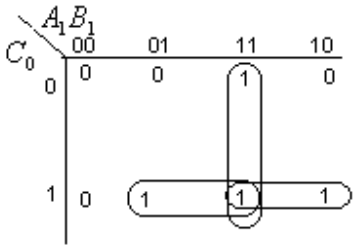


Fig. 5.25

The expression for carry C_1 is given by:

$$C_0 = A_1 \cdot B_1 + B_1 \cdot C_0 + A_1 \cdot C_0$$

This expression may further be expanded in the following form:

$$C_1 = A_1 \cdot B_1 + (A_1 + \overline{A}_1) \cdot B_1 \cdot C_0 + A_1 \cdot (B_1 + \overline{B_1}) \cdot C_0$$
$$= A_1 \cdot B_1 + A_1 \cdot B_1 \cdot C_0 + \overline{A}_1 \cdot B_1 \cdot C_0 + A_1 \cdot \overline{B}_1 \cdot C_0$$
$$= A_1 \cdot B_1 \cdot (1 + C_0) + (\overline{A}_1 \cdot B_1 + A_1 \cdot \overline{B_1}) \cdot C_0$$
$$= A_1 \cdot B_1 + (A_1 \oplus B_1) \cdot C_0$$

The term $A_1 \cdot B_1$ is the carry bit (say *C*) of the half adder (adder of two bits A_1 & B_1) and $A_1 \oplus B_1$ is the sum *S* of half adder. So the expression for C_1 may be rewritten as:

$$C_1 = C + S \cdot C_0$$

The full adder may therefore, be realized as shown in figure 5.26.

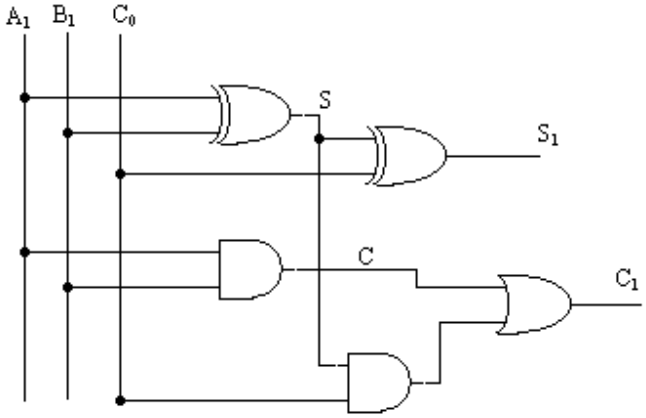


Fig. 5.26

It is clear that a full adder consists of two half adders and an OR gate as given in figure 5.27.

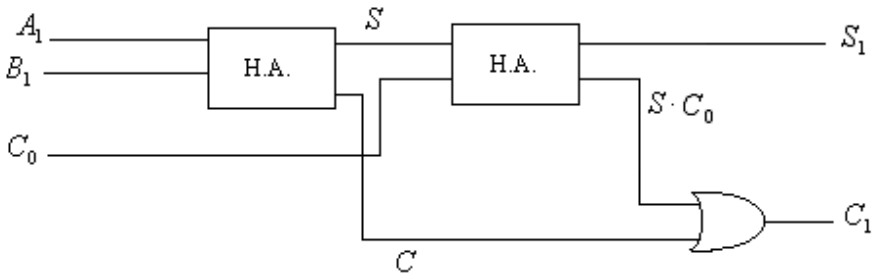


Fig. 5.27

The realization of Full Adder with 9 NAND gates is shown in figure 5.28. The symbolic representation of the half adder is given in figure 5.29.

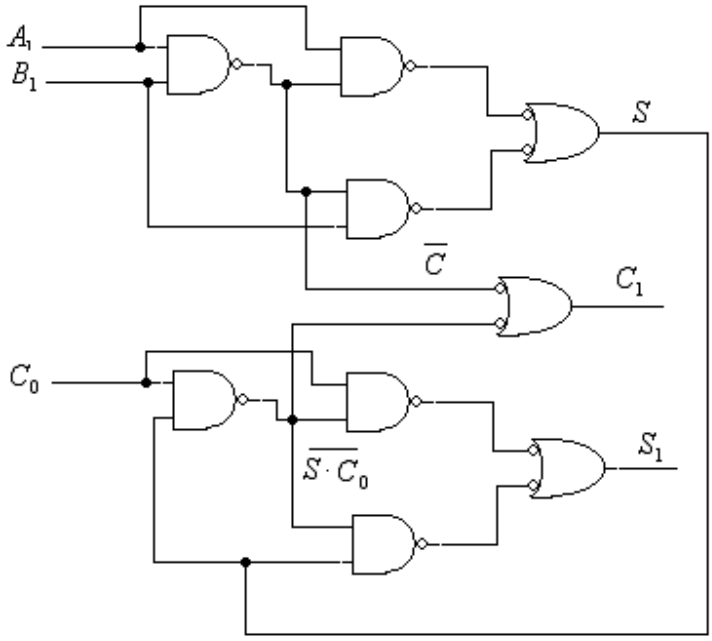


Fig. 5.28

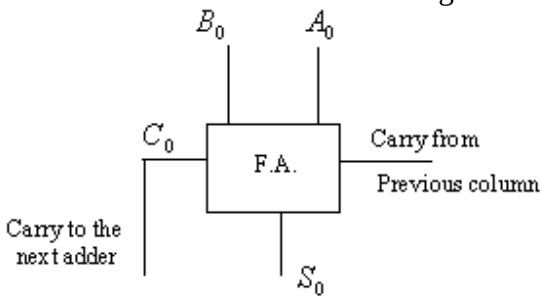


Fig. 5.29

Example 5.2: Design the Full adder using NOR gates only.

Solution: Since the full adder is the combination of two half adders, so realization of full adder with NOR gates only, is shown in figure 5.30.

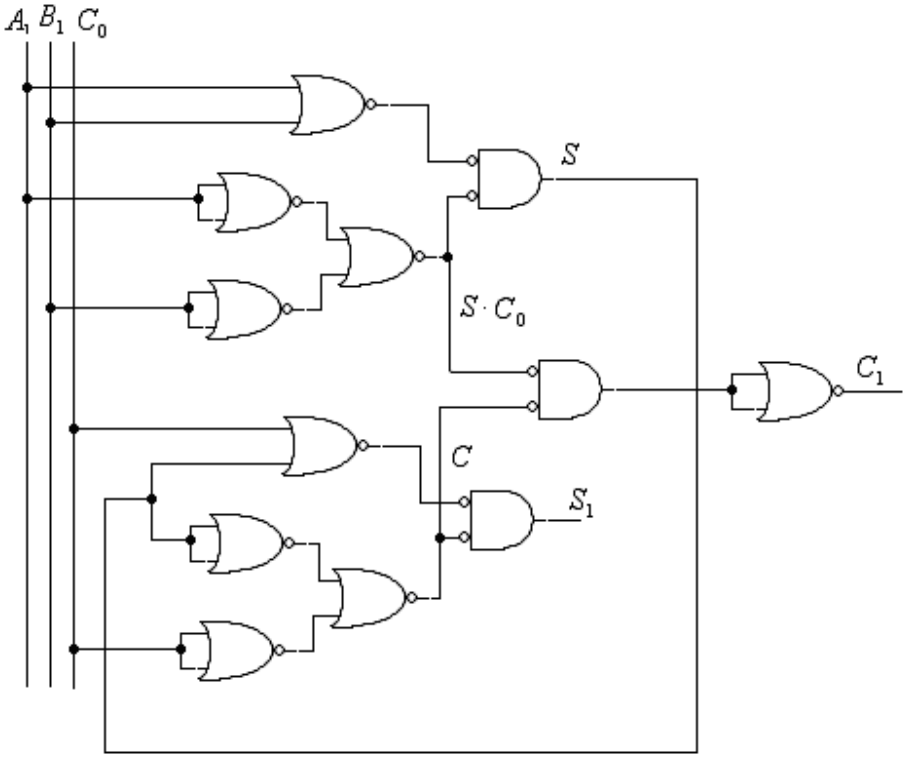


Fig. 5.30

5.4 Parallel Binary Adder: Four full adders may be connected as shown in figure 5.31, to add two binary numbers each of 4 bit long. The addition of two four bit numbers is given by:

C_3	C_2	C_1	C_0	
	A_3	A_2	A_1	A_0
	B_3	B_2	B_1	B_0
S_4	S_3	S_2	S_1	S_0

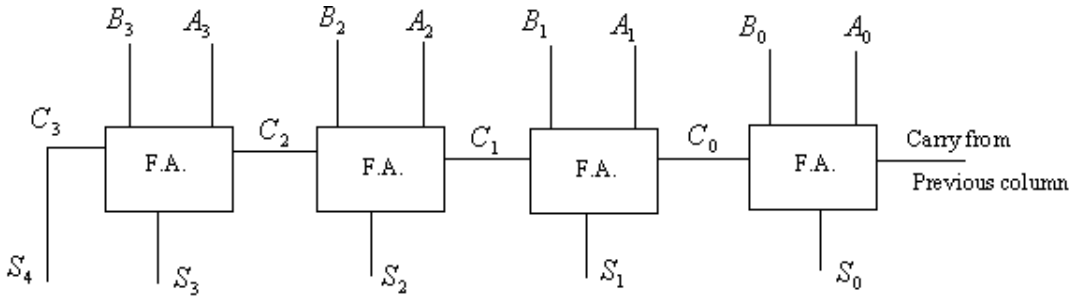


Fig. 5.31

The addition of more number of bits may be added in the similar fashion. This is known as parallel binary adder.

Parallel binary adders are available in the form of ICs. The two-bit binary full adder IC is 74LS82; its functional block diagram is given in figure 5.32.

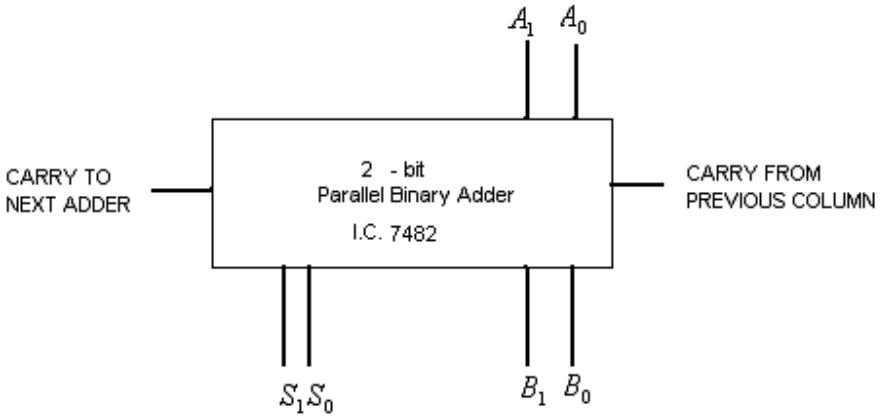


Figure 5.32

The two 7482 IC's may be connected to use a four bit adder. The 'carry to the next adder' pin of one IC may be connected to the 'carry from the previous column' pin of the second IC. However, 4-bit parallel adder IC 74C83 is also available, whose functional block diagram is shown in figure 5.33. The two such IC's may be connected to use as the 8 – bit adder. This can be extended to any number of bits.

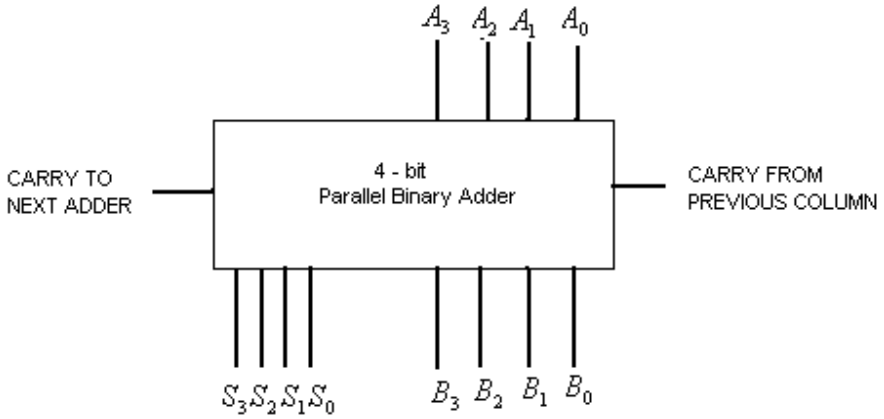


Fig. 5.33

Example 5.4: Design a half subtractor using NAND gates only.

Solution: A half subtractor may be designed by the same method as the half adder. The truth table for the half subtractor is given in table 5.10.

Table 5.10

X_0	Y_0	D_0	B_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

In this table X_0 and Y_0 are the minuend and subtrahend respectively; D_0 the difference of the two bits (X_0 & Y_0) and B_0 is the borrow bit from the next bit. The Boolean expressions for Difference D_0 and borrow bit B_0 are given as:

$$D_0 = \overline{X_0} \cdot Y_0 + X_0 \cdot \overline{Y_0} = X_0 \oplus Y_0$$
$$B_0 = \overline{X_0} \cdot Y_0$$

The realization of these expressions using NAND gates only may, therefore, be given in figure 5.34.

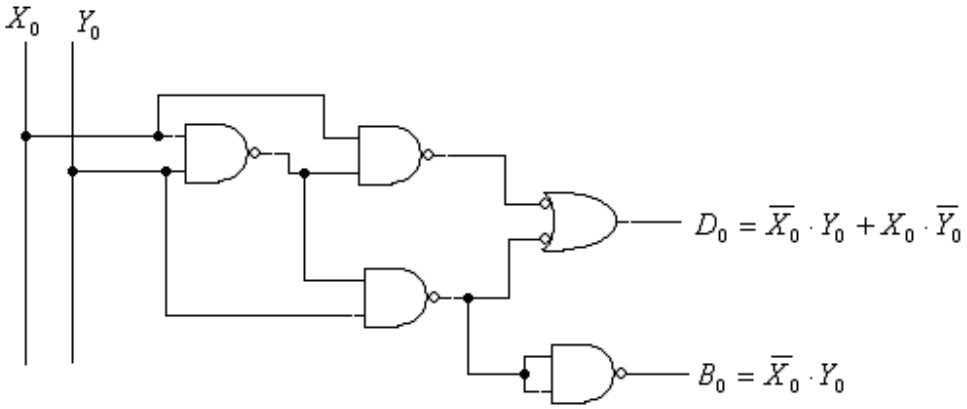


Fig. 5.34

The symbolic representation of half subtractor is given in figure 5.35.

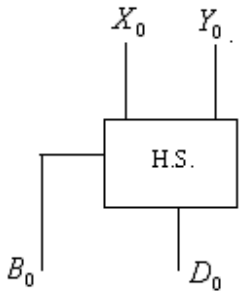


Fig. 5.35

Example 5.4: Design a Full subtractor using NAND gates only.

Solution: The truth table for full subtractor is shown in table 5.11, in which X_1, Y_1 are the minuend and subtrahend respectively and B_0 is the borrow bit from the previous bit. The output terms D_1 and B_1 are the difference bit and borrow to the next bit.

Table 5.11

X_1	Y_1	B_0	D_1	B_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

The Boolean expression for difference D_1 is given by:

$$D_1 = \overline{X_1} \cdot \overline{Y_1} \cdot B_0 + \overline{X_1} \cdot Y_1 \cdot \overline{B_0} + X_1 \cdot \overline{Y_1} \cdot \overline{B_0} + X_1 \cdot Y_1 \cdot B_0$$
$$= (\overline{X_1} \cdot \overline{Y_1} + X_1 \cdot Y_1) \cdot B_0 + (\overline{X_1} \cdot Y_1 + X_1 \cdot \overline{Y_1}) \cdot \overline{B_0}$$
$$= \overline{(X_1 \oplus Y_1)} \cdot B_0 + (X_1 \oplus Y_1) \cdot \overline{B_0}$$
$$= X_1 \oplus Y_1 \oplus B_0$$

The Boolean expression for the borrow B_1 to the next bit is obtained from the K –map shown in figure 5.36.

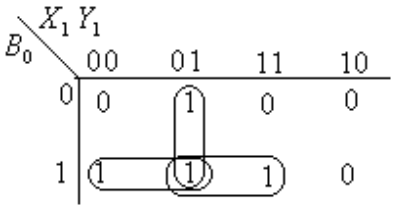


Fig. 5.36

$$B_1 = \overline{X_1} \cdot Y_1 + Y_1 \cdot B_0 + \overline{X_1} \cdot B_0$$
$$= \overline{X_1} \cdot Y_1 + (X_1 + \overline{X_1}) \cdot Y_1 \cdot B_0 + \overline{X_1} \cdot (Y_1 + \overline{Y_1}) \cdot B_0$$
$$= \overline{X_1} \cdot Y_1 + X_1 \cdot Y_1 \cdot B_0 + \overline{X_1} \cdot Y_1 \cdot B_0 + \overline{X_1} \cdot Y_1 \cdot B_0 + \overline{X_1} \cdot \overline{Y_1} \cdot B_0$$
$$= \overline{X_1} \cdot Y_1 \cdot (1 + B_0) + (X_1 \cdot Y_1 + \overline{X_1} \cdot \overline{Y_1}) \cdot B_0$$
$$= \overline{X_1} \cdot Y_1 + (X_1 \cdot Y_1 + \overline{X_1} \cdot \overline{Y_1}) \cdot B_0$$
$$= \overline{X_1} \cdot Y_1 + \overline{D_0} \cdot B_0$$

where D_0 is the difference of the half subtractor.

The full subtractor circuit may now be shown to be implemented using NAND gates as illustrated in figure 5.37

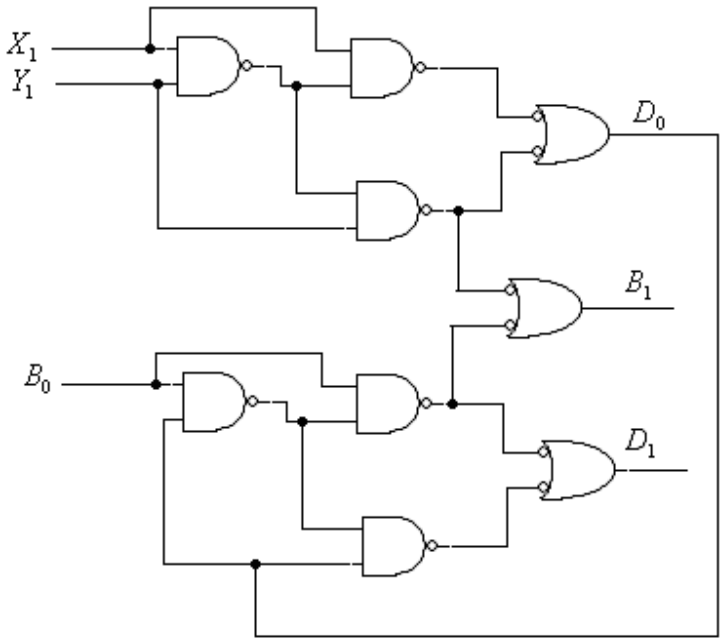


Fig. 5.37

The full subtractor is the combination of two half subtractors and gates as shown in figure 5.38.

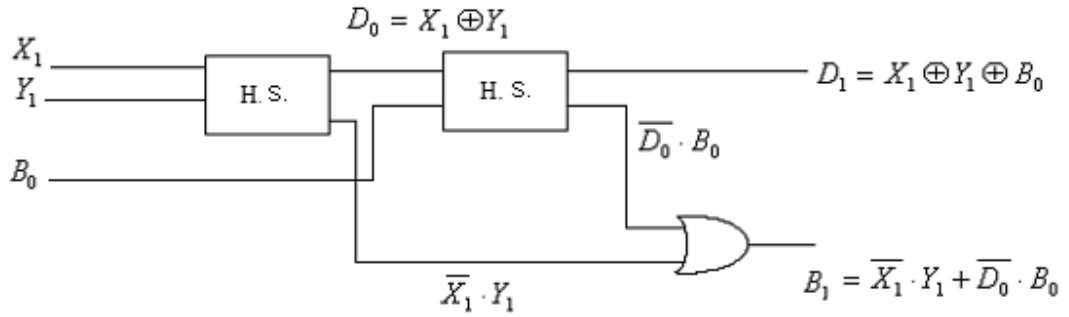


Fig. 5.38

The symbolic representation of half subtractor is given in figure 5.39

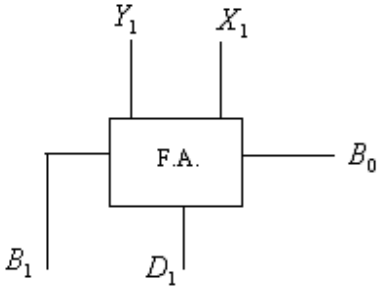


Fig. 5.39

5.5 BCD or 8421 Adder: The BCD numbers are generally processed in digital systems, so it is necessary to design BCD adder. Before discussing the design details of the 8421 adder, it is essential to know how the two decimal numbers are added in this code.

Let the two decimal numbers 3 & 4 added in 8421 code.

4	0 1 0 0
+3	0 0 1 1
7	0 1 1 1

The result 0111 (+ 7) is correct.

Further the addition of two other decimal numbers 7 and 6 in 8421 code gives the incorrect answer. This is illustrated as given below:

7	0 1 1 1
+6	0 1 1 0
13	1 1 0 1

The result 1101 is correct in natural binary number but it is incorrect in 8421 code. Its answer should have been 00010011 (13). However, to get the correct answer 6 (0110) is added to the incorrect sum, as it avoids the illegal number 1010 through 1111 of

the 8421 code. So when 0110 is added to the incorrect answer 1101, the correct answer is obtained as follows:

$$\begin{array}{r} 1\,1\,0\,1 \\ 0\,1\,1\,0 \\ \hline 1 \quad 0\,1\,1\,1 \end{array} \quad = 13$$

It is, therefore, concluded that if the sum of the numbers is more than 9, then 6 (0110) is added to the incorrect sum otherwise nothing is added.

Now general approach of the addition of two numbers is considered, say $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are being added in 8421 code as:

$$\begin{array}{r} A_3\ A_2\ A_1\ A_0 \\ B_3 B_2\ B_1 B_0 \\ \hline S_4\ S_3\ S_2\ S_1\ S_0 \end{array}$$

Now if the sum $S_4S_3S_2S_1S_0$ is more than 9, then 0110 is added to it otherwise 0000 is added. Manually the addition of decimal number 6 (0110) can easily be carried out, but for the design of BCD adder, 6 should automatically be got added whenever the sum is more than 9.

The following approach will help in getting the automatic addition of decimal number 6 in the incorrect sum. The sum $S_4S_3S_2S_1S_0$ is more than 9 if S_4 is 1 and/ or $S_3S_2S_1S_0$ is any of the 6 illegal BCD numbers 1010 through 1111. So a term $X = S_4 + S_3S_2 + S_3S_1$ will indicate if the sum is more than 9. That is if X is 1 than 0110 is added to the incorrect sum otherwise 0000 is added to it. In general 0XX0 is always added to the incorrect sum as follows:

$$\begin{array}{r} A_3\ A_2\ A_1\ A_0 \\ +\ B_3 B_2\ B_1 B_0 \\ \hline S_4\ S_3\ S_2\ S_1\ S_0 \\ +\ 0\ X\ X\ 0 \\ \hline X_4 X_3 X_2 X_1 X_0 \end{array}$$

X_4 is the carry bit which is not to be used, as the term X will take care of the carry bit for the next digit. The circuit for BCD addition for full one decimal digit (four bits) can easily be drawn as shown in figure 5.40.

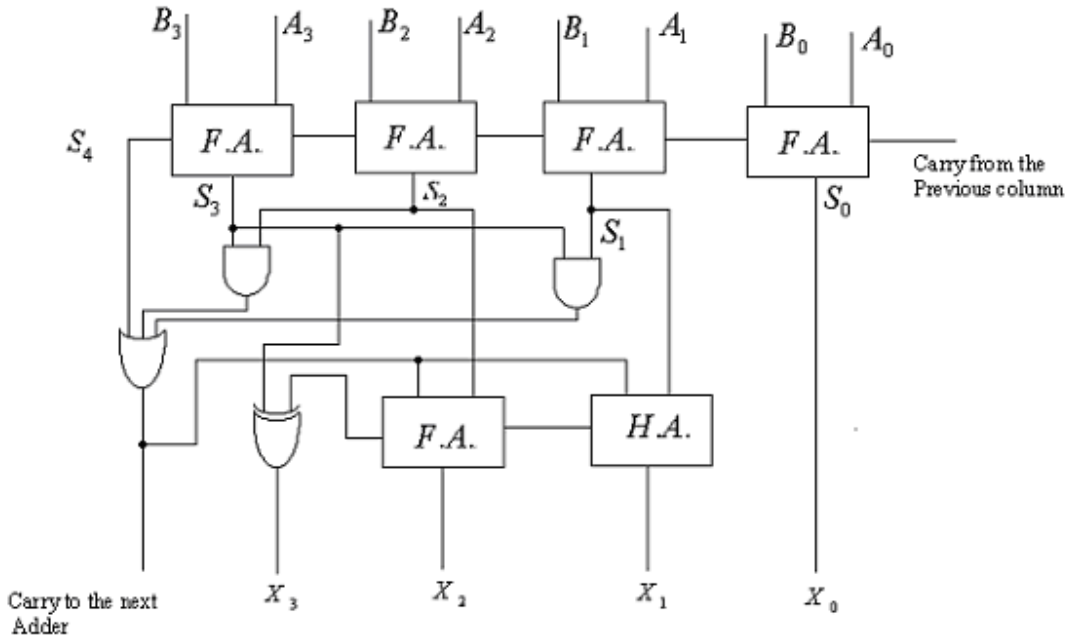


Fig. 5.40

The BCD adder can also be designed using two parallel binary adders (2 IC's 74LS83) and a few gates as shown in figure 5.41.

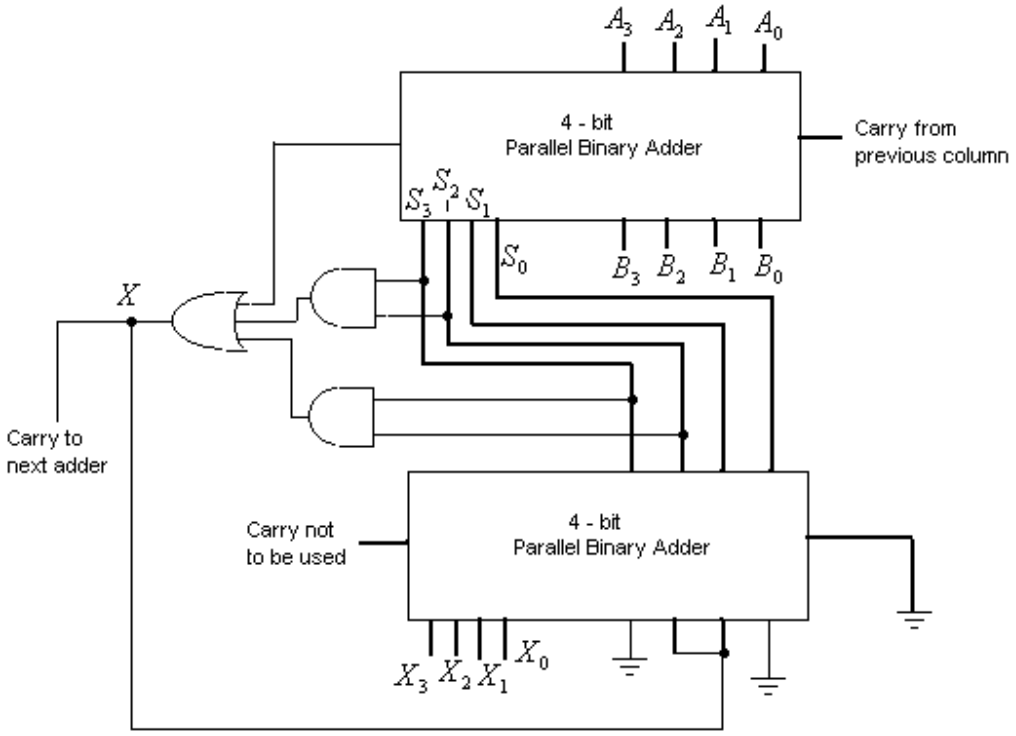


Fig. 5.41

5.6 Excess – 3 Adder: For the design of the excess –3 adder one should remember the addition of decimal numbers in XS –3 code. In XS –3 code first three number 0000 through 0010 and the last three numbers 1101 through 1111 are illegal. So to avoid these illegal numbers, 0011 is added to the incorrect answer if it is more than 9 else 0011 is to be subtracted.

Consider the two numbers say $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ to be added in XS –3 code as:

$$\begin{array}{r} A_3\ A_2\ A_1\ A_0 \\ B_3\ B_2\ B_1\ B_0 \\ \hline S_4\ S_3\ S_2\ S_1\ S_0 \\ \hline \end{array}$$

The sum $S_4S_3S_2S_1S_0$ is more than 9, if S_4 is 1 otherwise sum is less than or equal to 9. So if S_4 is more than 9, then 0011 is to be added to the incorrect sum else 0011 is to be subtracted.

For the subtraction of 0011 from the uncorrected sum 1's complement method is used i.e. 1100 (1's complement of 0011) is added to the incorrect sum and finally end around carry (EAC) is added to it.

It is concluded that if sum is more than 9 ($S_4 = 1$), then 0011 is to be added to the incorrect sum; and if sum is less than or equal to 9 ($S_4 = 0$), then 1100 (1's complement of 0011) is added. Therefore, one can say that in both the cases $\overline{S_4}.\overline{S_4}.S_4.S_4$ is to be added as given below:

$$\begin{array}{r} A_3\ A_2\ A_1\ A_0 \\ +\ B_3\ B_2\ B_1\ B_0 \\ \hline S_4\ S_3\ S_2\ S_1\ S_0 \\ +\ \overline{S_4}\ \overline{S_4}\ S_4\ S_4 \\ \hline X_3\ X_2\ X_1\ X_0 \\ +\ X_4 \quad \text{— EAC} \\ \hline \end{array}$$

The circuit for XS –3 addition for full one decimal digit (four bits) can easily be implemented as shown in figure 5.42.

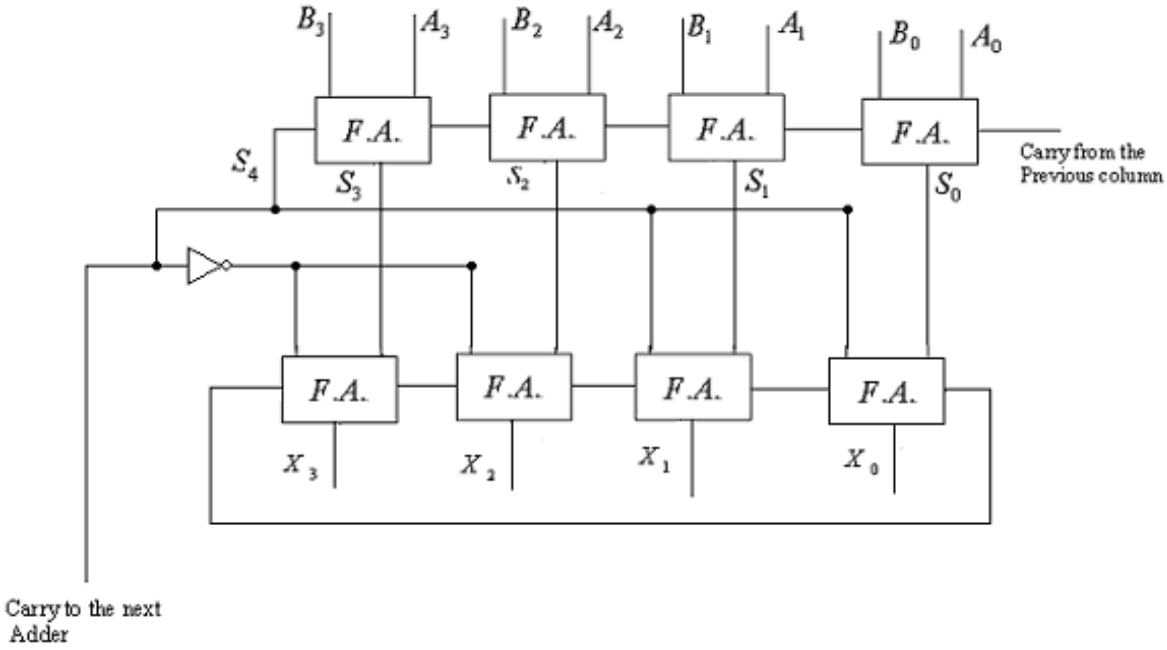


Fig. 5.42

5.7 Two's Complement Adder/Subtractor: The two's complement adder/subtractor is most commonly used in arithmetic circuits because it greatly simplifies the method of operation. It can be used to add and subtract the binary numbers.

Consider a binary number $B_3B_2B_1B_0$ to be subtracted from or added to another binary number $A_3A_2A_1A_0$. For addition, $B_3B_2B_1B_0$ is directly added to $A_3A_2A_1A_0$; and for subtraction 2's complement of $B_3B_2B_1B_0$ is added to it. The 2's complement is taken by inverting $B_3B_2B_1B_0$ and adding 1 to it.

Figure 5.43 shows the circuit diagram of 2's complement adder/ subtractor. A SUB signal provided in the circuit, is used to directly load B's to the full adders for addition of binary numbers and for subtraction 2's complement of the B's are loaded to the full adders.

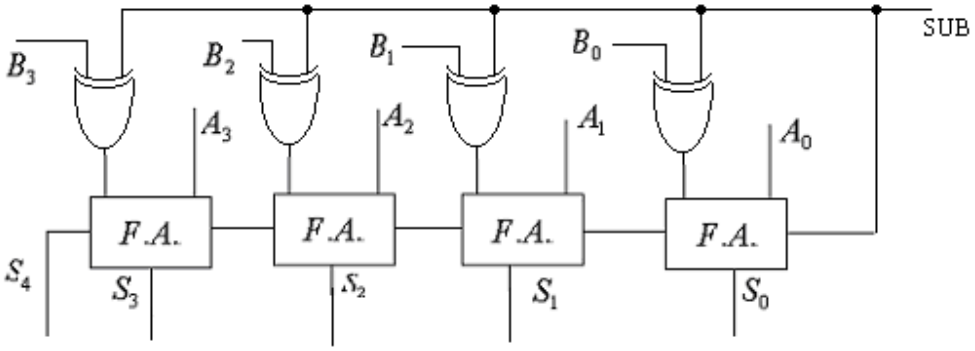


Fig. 5.43

In this circuit B's are given to the full adders through exclusive- OR gates. The SUB signal is given as low (logic 0) when the circuit is to be used as binary adder and it is given as high (logic 1) for subtractor.

When SUB signal is low, the bits B's are passed through the exclusive– OR gates to the full adders without inversion, since $B \oplus SUB = B$ (for SUB = 0). The circuit, therefore, act as simple parallel binary adder. If on the other hand SUB signal is high, the bits B's gets inverted through exclusive – OR gates, since $B \oplus SUB = \overline{B}$ (for SUB = 1). So 1's complement of the B's goes to the full adder and also SUB signal connected to the carry terminal of the first full adder, results the 2's complement of the bits B's. The circuit, therefore, works as the subtractor. The final carry S_4 is as the carry bit for the binary adder and it is used as the sign bit for the subtractor.

PROBLEMS

1. What are combinational circuits? Give the design procedure of combinational logic circuits.

2. Discuss the design of a railway switching circuit. The word statement of the problem is the same as that given in the solved example 5.1 (given in the text) with the difference that the railway station has three platforms instead of four also there are only two track changers.

3. Repeat the problem 2, if the trains are allowed to enter from either direction. (Hint: one more variable may be assumed to illustrate the direction. If coming from L.H.S., that variable is 0 otherwise 1. This problem will have four input variables).

4. The entrance to a group of three flats has a tube light. The tube light is to be switched ON and OFF independently by the tenants of the three flats using switches located in their flats. Design a switching circuit to implement this using:

   (i) Exclusive – OR gates. (ii) NOR gates only.

5. Design a switching circuit to generate even parity bit for the decimal numbers transmitted in excess – 3 code. Use

   (i) NOR gates to realize the circuit

   (ii) NAND gates to realize the circuit.

6. Four inputs A, B, C and D control three LEDs. The red LED glows when:

A is 1, B is 0	B is 1, C is 0
A is 0, B is 1	B is 1, C is 1
C is 1, D is 1	B & C are 1
C is 0, D is 1	A & C are 1

   The green LED will glow, when:

   B and C are 1
   C and D are 1
   A and D are 1

A and D are 1

The yellow LED will glow when:

A and B are 1
C and D are 1
All are 1
B is o, D is 1

Draw the simplest logic circuit to implement this.

7. Repeat the solved example 5.5, having only four board of directors instead of five. The board of directors has 45%, 20%, 10% and 25% shares.

8. Design a combinational circuit, which can receive only valid 4-bit excess- 3 or 4-bit 2421 BCD code. The circuit should have two outputs, one to indicate the valid excess –3 signal and other to indicate the valid 2421 signal. NAND gates should be used to implement the circuit.

9. A circuit receives four- bit 2421 code. Design the simplest logic circuit which gives an output 1 whenever the inputs are equivalent odd decimal numbers.

10. A circuit receives four- bit 8421 code. Design the simplest logic circuit which gives an output 1 whenever the inputs are equivalent even decimal numbers.

11. Design a simple logic circuit using OR gates only, that will output 1 whenever ay of the following binary numbers appears at the input.

0000, 0100, 1010, 1110, 1111

12. What is half adder? Discuss the design of half adder circuit using:

(i) 5 NAND gates, (ii) 5 NOR gates.

13. What is full adder? Discuss the design of full adder circuit using:

(i) two exclusive-OR gates, two AND gates and one OR gate, (ii) 9 NAND gates, (iii) NOR gates.

Also show that a full adder is a combination of two half adder.

14. What is half subtractor? Discuss the design of half subtractor circuit using:

(i) 5 NAND gates (ii) 5 NOR gates.

15. What is full subtractor? Discuss the design of full subtractor circuit using:
(i) NAND gates, (ii) NOR gates.

Also show that a full subtractor is a combination of two half subtractors.

16. Discuss the parallel binary adder. Explain how two 7482 ICs may be connected to form a four bit adder.

17. Explain four bit parallel binary adder IC 7483. How two 7483 ICs are connected to form an eight bit adder?

18. Discuss the details of the design of 8421 adder.

19. Explain how the 8421 adder is designed using two 4-bit parallel binary adder and a few gates.

20. Give the details of the design of the excess – 3 adder.

21. Discuss 2's complement adder/ subtractor circuit. Give its design details.

_____

6

More Combinational Circuits

The construction details, working and applications of some more combinational circuits will be discussed in this chapter. These will include the multiplexers, de-multiplexers, decoders, encoders, code converters, PLA's, magnitude comparators and parity generator cum checkers etc.

6.1 Multiplexers: A multiplexer (MUX) also known as data selector, is a logic circuit which allows the digital information from multi-inputs to a single output line. The selection of the input data to be routed to the output line is done by the select terminals. The number of select terminals depends on the number of input lines to be routed to output line, given by the general formula as:

$$2^K = N ,$$

where N is the number of input lines and K is the number of select terminals. In other words, if there are 4 input lines to be routed to output line, then two select terminals are needed as $2^2 = 4$.

The block diagram for 4:1 multiplexer is shown in figure 6.1.

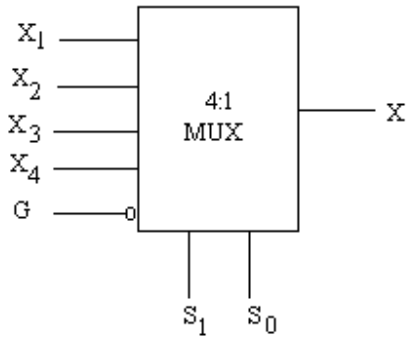


Fig. 6.1

In which X_0, X_1, X_2, X_3 are the 4 input lines and S_1, S_0 are the select terminals and X is the output terminal. Normally a strobe terminal or enable terminal (G) is provided in the MUXs which is normally active-low. The active-low means it performs the operation when it is low; it also helps to cascade the MUXs. The Boolean function to perform the multiplexing action is given as:

$$X = X_0 \cdot \overline{S}_1 \cdot \overline{S}_0 + X_1 \cdot \overline{S}_1 \cdot S_0 + X_2 \cdot S_1 \cdot \overline{S}_0 + X_3 \cdot S_1 \cdot S_0$$

The output X will follow the input data depending on the select terminals S_1, S_0, as given in the table 6.1.

Table 6.1

Select terminals		Output
S_1	S_0	X
0	0	$X = X_0$
0	1	$X = X_1$
1	0	$X = X_2$
1	1	$X = X_3$

Note that only one of the inputs X_0, X_1, X_2, X_3 is routed to the output X (one at a time). The realization of the Boolean function X with NAND gates only is shown in figure 6.2.

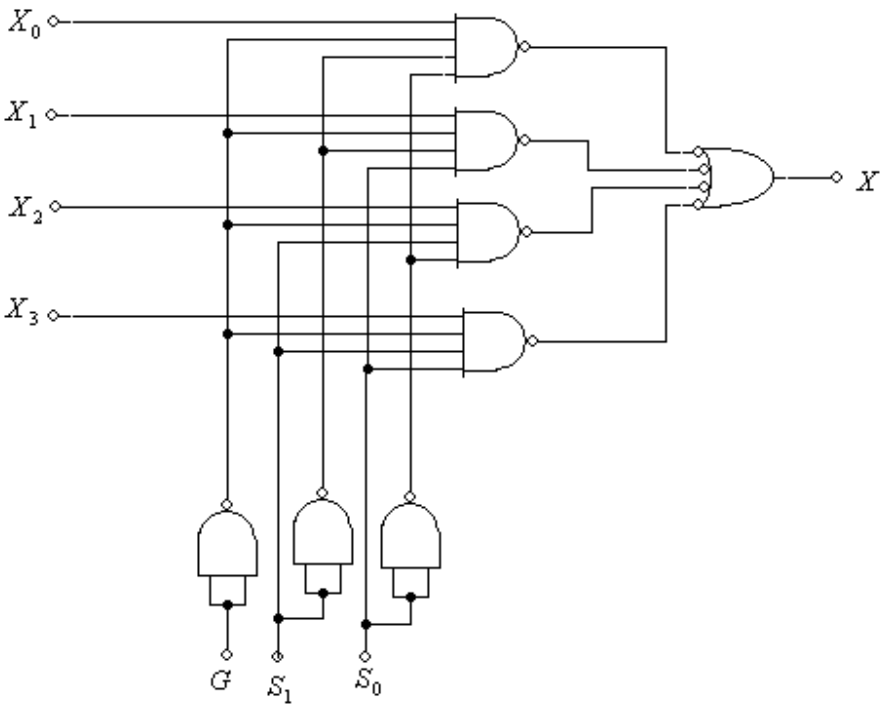


Fig. 6.2

The MUXs are available in the form of the following IC's:

74157 quadruple two – input multiplexer/data selector.

74151A eight - input multiplexer/data selector.

74150 sixteen - input multiplexer/data selector.

74157 Quadruple two – input multiplexer/data selector: The internal logic diagram of the IC 74157 is given in figure 6.3. It consists of four two input multiplexers on a single

chip. Each of the four multiplexers has a common data select line S and a common chip enable terminal G. A low signal to the chip enable terminal G allows the selected input data to rout to the output. Since there are only two inputs to be selected from each multiplexer, a single data select terminal is sufficient.

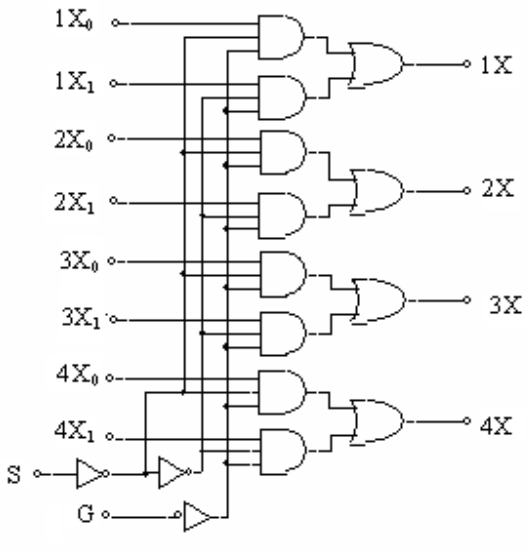


Fig. 6.3

74151A Eight - input multiplexer/data selector: Figure 6.4 shows the logic block diagram for a 8 – input multiplexer/ data selector IC 74151A. It has 8 data inputs X_0 through X_7, three data select terminals S_2 S_1, & S_0 and an enable terminal G. when enable terminal G is high, the multiplexer is disabled and output X is zero irrespective of the select input terminal. However, when the enable terminal G is low the input data is routed to the output as per data select terminals S_2 S_1, & S_0 as illustrated in table 6.2.

Table 6.2

Inputs				Output
G	S_2	S_1	S_0	X
H	ø	ø	ø	L
L	L	L	L	X_0
L	L	L	H	X_1
L	L	H	L	X_2
L	L	H	H	X_3
L	H	L	L	X_4
L	H	L	H	X_5
L	H	H	L	X_6
L	H	H	H	X_7

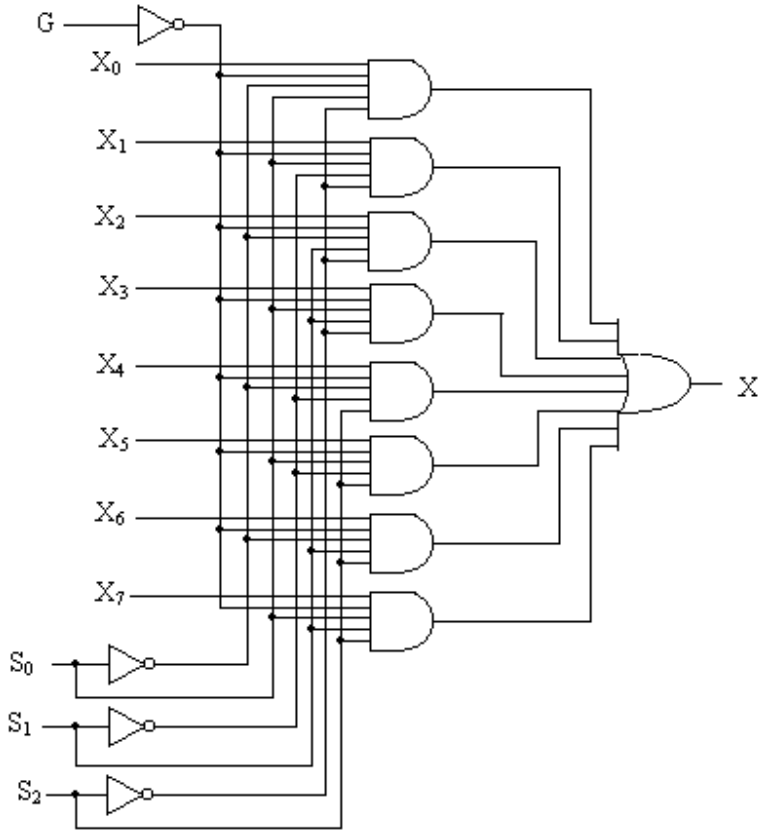


Fig. 6.4

74150 Sixteen - input multiplexer/data selector: The IC 74150 is 16:1 multiplexer having 16 input lines and one output line. It has four select terminals S_3, S_2, S_1& S_0 and one enable terminal G which is kept low for multiplexing action. The block diagram of the 16:1 MUX is given in figure 6.5.

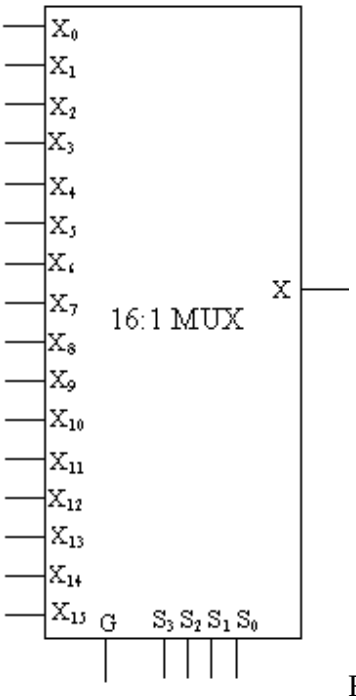


Fig. 6.5

6.1.1 Expansion of Multiplexers: For the expansion of number of input terminals of the multiplexers two MUXs may be cascaded. Two 4:1 MUXs may be cascaded to form 8:1 MUX. Similarly two 8:1 MUXs may be cascaded to have a 16:1 multiplexer and so on. Figure 6.6 illustrates how two 4:1 MUXs are cascaded to form 8:1 MUX. The enable terminal G of the MUXs in-conjunction with a NOT gate provides the third select terminal. When S_2 is zero the first MUX will be enabled and inputs X_0 through X_3 will be routed to its output; and when S_2 is 1, the second MUX will be enabled, X_4 through X_7 will be routed to the output of the second MUX. The outputs of the two MUXs are connected to the inputs of an OR gate which then gives the final output (ref. fig. 6.6).

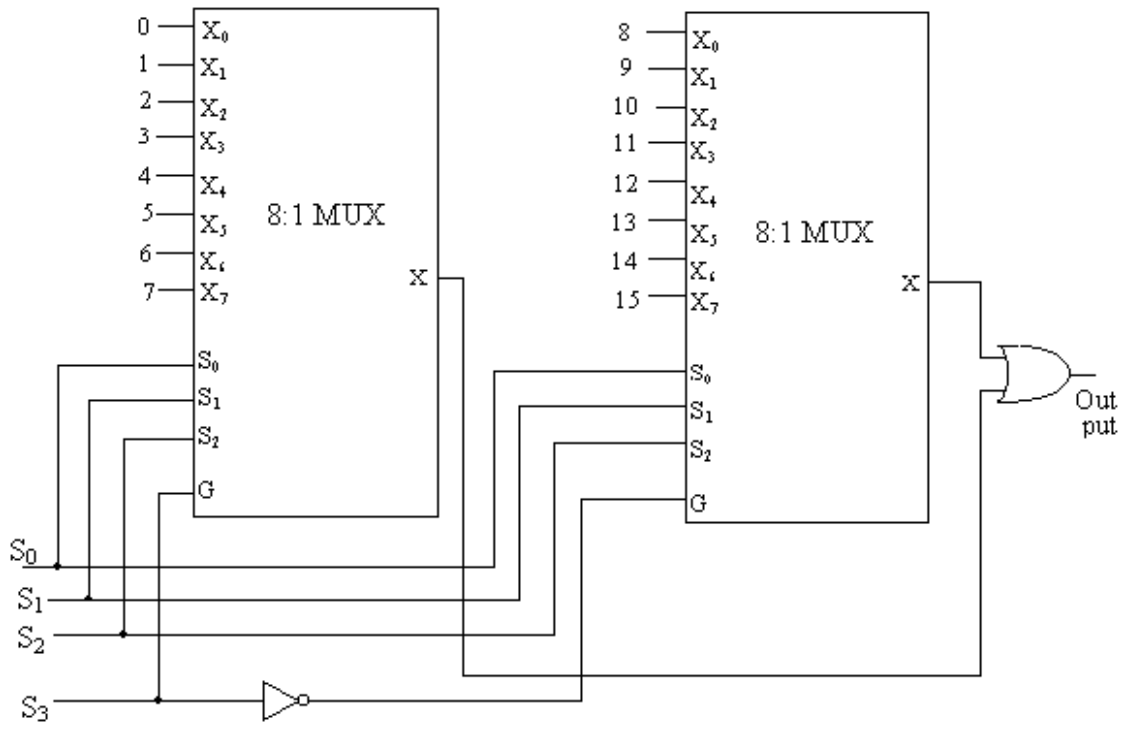


Fig. 6.6

6.1.2 Applications of Multiplexers: Primary aim of the MUXs is the multiplexing operation, that is, the selected input is routed to the output. In addition to this, Implementation of Boolean function can easily be done with MUXs, since MUXs are available in the form of integrated circuits. The method of implementing the Boolean function is that the truth table is first constructed for the given function to be implemented. Then logic 1 is connected to each data input of the multiplexer corresponding to each combination of the input variables which has 1 in the output column of the truth table. The logic 0 is, however, connected to the remaining inputs of the MUX. The variables are connected to the data select inputs of the multiplexer.

The Boolean functions of N – variables can also be implemented by the Multiplexers of (N – 1) select lines. A function of 4 variables can be implemented with 8:1 multiplexer having 3 select lines. Let A, B, C, D are the input variables of the function F, which is to be implemented with a multiplexer. The variable A is the most significant bit and D is the least significant bit. Variables B, C, D are assumed to be the select terminals for the multiplexer. The truth table is drawn for the given function. It is well known that in the truth table variables BCD progresses twice through the sequence 000, 001…. 111; once with A = 0 and other with A = 1. The connections to be made to the data inputs of the multiplexer, following rules are observed.

1. A logical 0 is connected to the data input of MUX, if the 0 occurs at the output in the truth table, both times when MSB is 0 and 1 (other variables having the same value).
2. A logical 1 is connected to the data input of MUX, if the 1 occurs at the

output in the truth table, both times when MSB is 0 and 1 (other variables having the same value).

3. MSB is connected to the data input of MUX, if the output in the truth table is different both times when MSB is 0 and 1(other variables having the same value); and also output is the same as the MSB.
4. The complement of the MSB is connected to the data input of MUX, if the output in the truth table is same both times when MSB is 0 and 1(other variables having the same value); and also output is the same as the complement of MSB.

Example 6.1: Realize the following function of three variables with 8:1 MUX.

$$F(A, B, C) = \sum (0,1,3,4,7)$$

Solution: The truth table of the given function is drawn as shown in table 6.3. To realize the given function using 8:1 MUX, the variable A, B, C are assumed to be the three select terminals as shown in figure 6.7. The logic 1 is connected to each data input of the multiplexer corresponding to each combination of the input variables which has 1 in the output column of the truth table. The logic 0 is connected to the remaining inputs of the MUX. The inputs X_0, X_1, X_3, X_4, X_7 are, therefore, connected to the logic 1 and X_2, X_5, X_6 are connected to logic 0.

Table 6.3

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

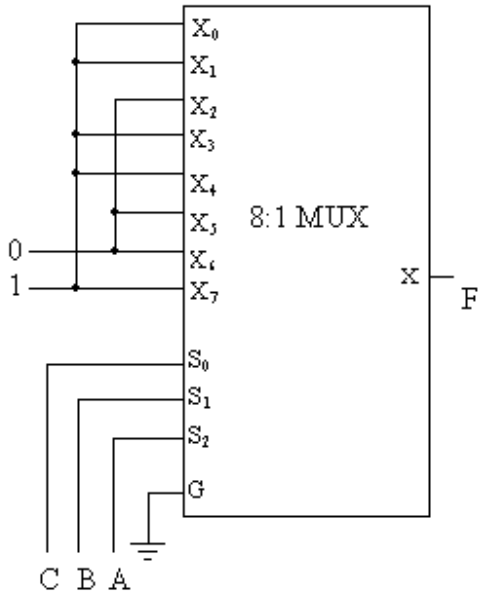


Fig. 6.7

Example 6.2: Use Multiplexers to implement of Full adder.

Solution: It is well known that a full adder adds three bits of information. Let A B C are three bits to be added. Let augend bit is A, addend bit is B and C is the carry from the previous column; SUM and CARRY to the next bit are given in the table 6.4. Implementation of SUM and CARRY is shown in figure 6.8.

Table 6.4

A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

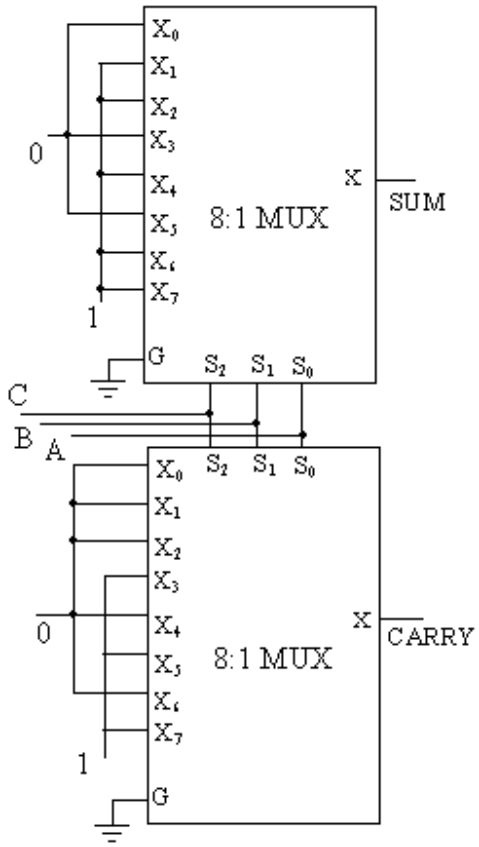


Fig. 6.8

Example 6.3: Realize the following function of four variables with 8:1 MUX.

$$F(A, B, C, D) = \sum (0,1,3,5,7,11,13,15)$$

Solution: The truth table for the given function is first of all drawn (table 6.5) and YZW are assumed to be the select terminals of the 8:1 MUX. The inputs to the multiplexer are obtained from the truth table as given below.

$X_0 = \overline{A}$ $X_1 = \overline{A}$
$X_2 = 0$ $X_3 = 1$
$X_4 = 0$ $X_5 = 1$
$X_6 = 0$ $X_7 = 1$

Figure 6.9 shows the implementation of the given function.

Table 6.5

A	S_2 B	S_1 C	S_0 D	F	Input to MUX
0	0	0	0	1	$\overline{A}$
0	0	0	1	1	$\overline{A}$
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	$\overline{A}$
1	0	0	1	0	$\overline{A}$
1	0	1	0	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	1	1	1

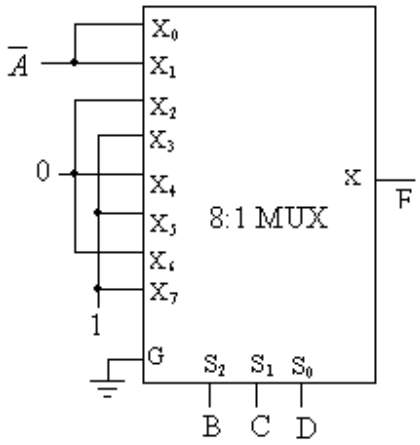


Fig. 6.9

6.2 Demultiplexers: A demultiplexer performs the reverse process of multiplexer; it receives the information on a single line and steers to several output lines. Demultiplexer can also be called the Data Distributor as it can transmit the same data to the different lines. It transmits the data to 2^N output lines, for which the select terminals of N bits are required. For example, to transmit the single data to four output lines (1:4 DMUX), select terminals of two bits are required; similarly for 1:8 DMUX select terminals of 3 bits are required and so on. The functional block diagrams of 1:4 DMUX and 1:8 DMUX are shown in figure 6.10(a) and 6.10(b) respectively.

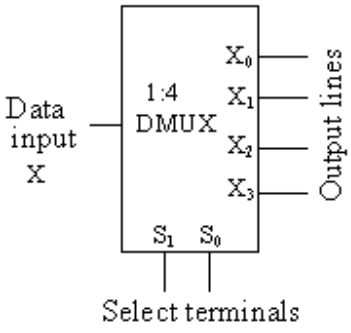


Fig. 6.10(a)

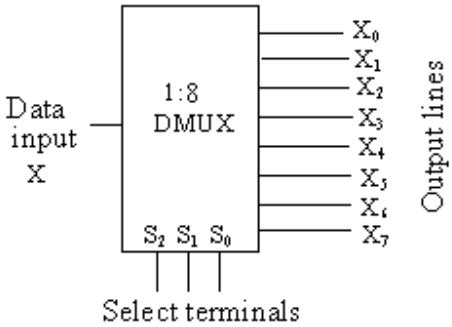


Fig. 6.10(b)

In a 1:4 DMUX, let X is the data input which is to be steered to 4 output lines X_0, X_1, X_2, X_3; the select terminals are S_1, S_0.

If $S_1S_0 = 00$, the input data X will be go to the output X_0.

If $S_1S_0 = 01$, the input data X will be go to the output X_1.

If $S_1S_0 = 10$, the input data X will be go to the output X_2.

If $S_1S_0 = 11$, the input data X will be go to the output X_3.

The Boolean expressions for X_0, X_1, X_2, X_3 are given by:

$$X_0 = X \cdot \overline{S}_1 \cdot \overline{S}_0$$

$$X_1 = X \cdot \overline{S}_1 \cdot S_0$$

$$X_2 = X \cdot S_1 \cdot \overline{S}_0$$

$$X_3 = X \cdot S_1 \cdot S_0$$

The implementation of these functions (or 1:4 DMUX) can be done as shown in figure 6.11.

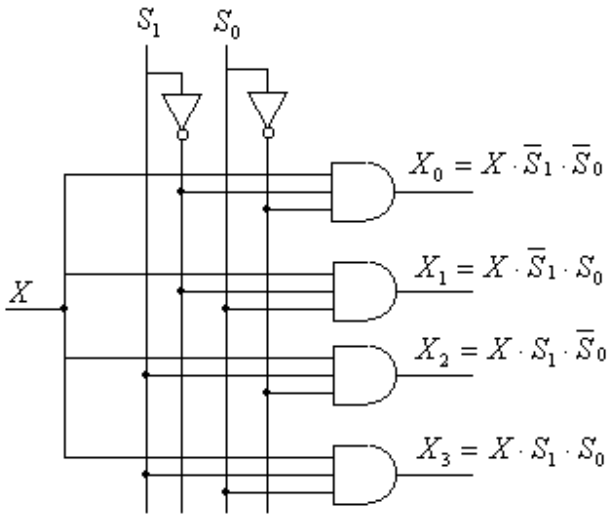


Fig. 6.11

6.3 Decoder: A decoder is a logic circuit which has a set of inputs representing a binary number and gives only one output corresponding to the input number. The decoder activates one output at a time depending upon the input binary number; all other outputs will be inactive. Figure 6.12 shows the functional block diagram of a decoder having *N* inputs and K outputs. The possible combinations of *N* inputs will be $2^N = K$, so there will be K outputs.

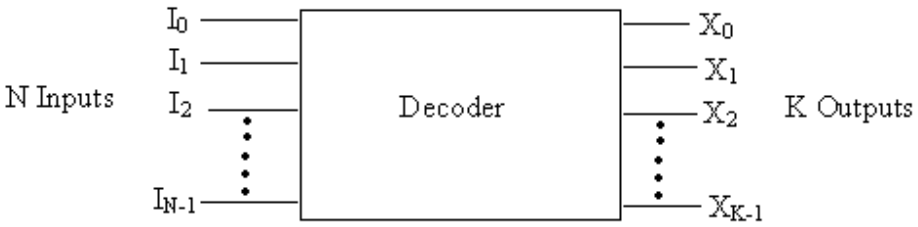


Fig. 6.12

Figure 6.13 shows the circuit diagram of 3 – to – 8 line decoder. It will have three input lines and $2^3 = 8$ output lines. When the three bit binary number is fed to the input of the decoder, as discussed above one output line corresponding to input binary is activated and all other output lines will be inactive. It is also called binary to octal decoder or converter because it takes a binary code as input and activates one of the eight (octal) output lines corresponding to the input binary code. The 3 – to – 8 can also be referred to as a 1 – of – 8 line decoder, because only one of the eight outputs is activated at a time. The truth table for 3 – to – 8 line decoder is shown in table 6.6.

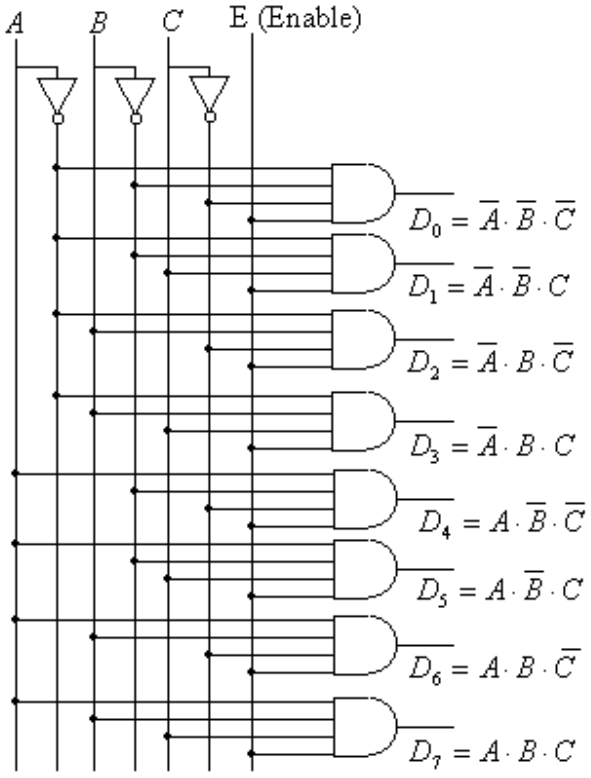


Fig. 6.13

Table 6.6

Inputs			Enable	Outputs							
A	B	C	E	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
∅	∅	∅	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

It is clear from the figure 6.13 that an Enable input line is connected to the fourth input of each gate. When the Enable input is connected to logic 0, all the gates will be disabled and force all output to be zero irrespective of the input data (ABC). However, the decoder will give the required data when the Enable terminal is held at logic 1. The functional block diagram of 3 to 8 line decoder is shown in figure 6.14.

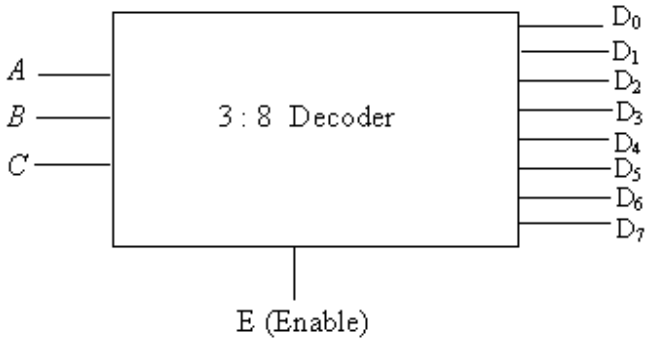


Fig. 6.14

The 4:16 line decoder can also be explained on the same pattern. It may be mentioned here that if AND are used in designing the decoder circuit, then Enable and all outputs will be active high. If on the other hand the decoder circuit is designed using NAND gates then the Enable as well as the outputs terminals will be active low.

Further it is interesting to note that the decoder can function as a demultiplexer. For example a 2:4 line decoder with Enable terminal can be used as a 1:4 DMUX, if the Enable terminal E is used as the data input line for the DMUX and the two input A & B of the decoder as the select terminals for the DMUX. It is illustrated in figure 6.15.

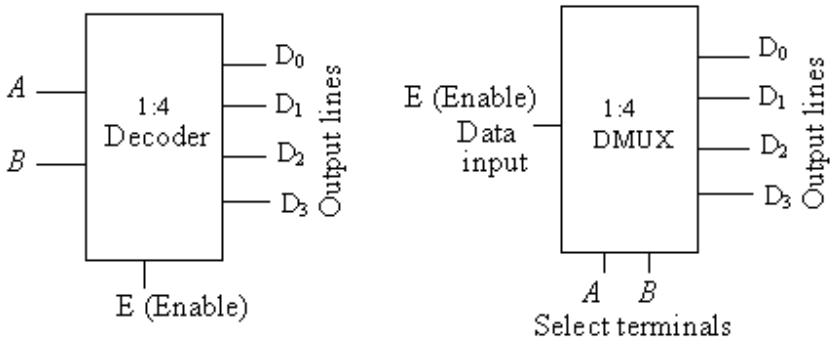


Fig. 6.15

Decoder/Demultiplexer circuits can be expanded to form the larger decoder circuit. For example two 3:8 line decoders with Enable terminal can be connected to form a 4:16 line decoder. Figure 6.16 shows the construction of a 4:16 line decoder with two 3:8 line decoders. From this figure it is clear that when the enable terminal E is 0, decoder (1) is enabled and decoder (2) is disabled. The decoder (1), therefore, gives the outputs as per the value of ABC. When the enable terminal E is 1, decoder (1) is disabled

and decoder (2) is enabled. The decoder (2) now gives the output as the input values. Here E terminal works as the most significant bit and C as the least significant bit. So EABC generates the binary input 0000 through 1111.

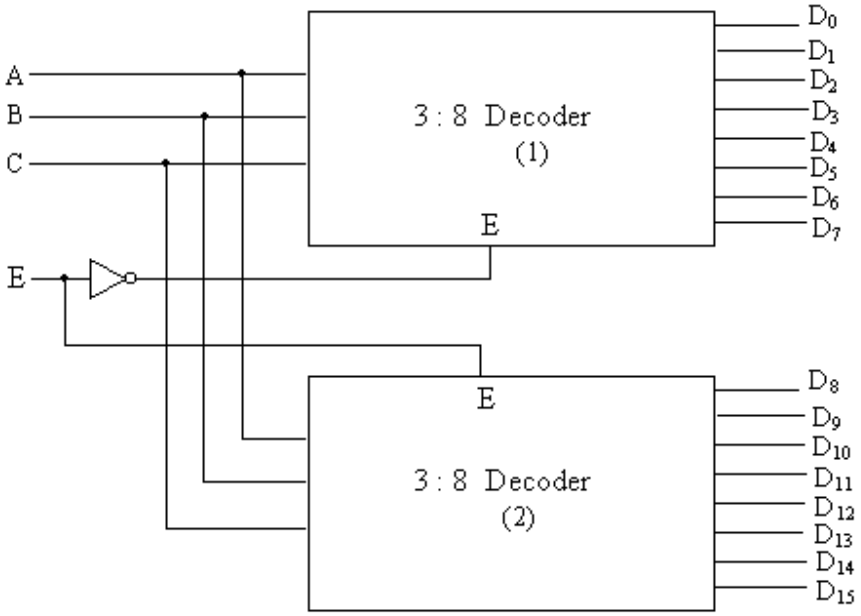


Fig. 6.16

The Boolean functions given in standard SOP form can be realized using the decoder circuits. For the realization of Boolean expressions, the decoder requires some gates also. The use of decoder for the implementation is more economical, as number of Boolean expressions can be implemented using one decoder and a few gates. However, in multiplexers one MUX is used for one Boolean function.

Example 6.4: Using a 4 –to – 16 line decoder, implement the following functions given in standard SOP form.

$$F_1(A, B, C, D) = \sum (0,1,2,4,6,7,12,14)$$

$$F_2(A, B, C, D) = \sum (3,5,8,10,13,15)$$

$$F_3(A, B, C, D) = \sum (5,6,7,11,12)$$

Solution: The realization of the given Boolean functions using one decoder and a few gates is shown in figure 6.17. The decoder used here is active high, so enable terminal E is connected to logic 1. Three OR gates are used for the implementation of three function; one OR gate for one function.

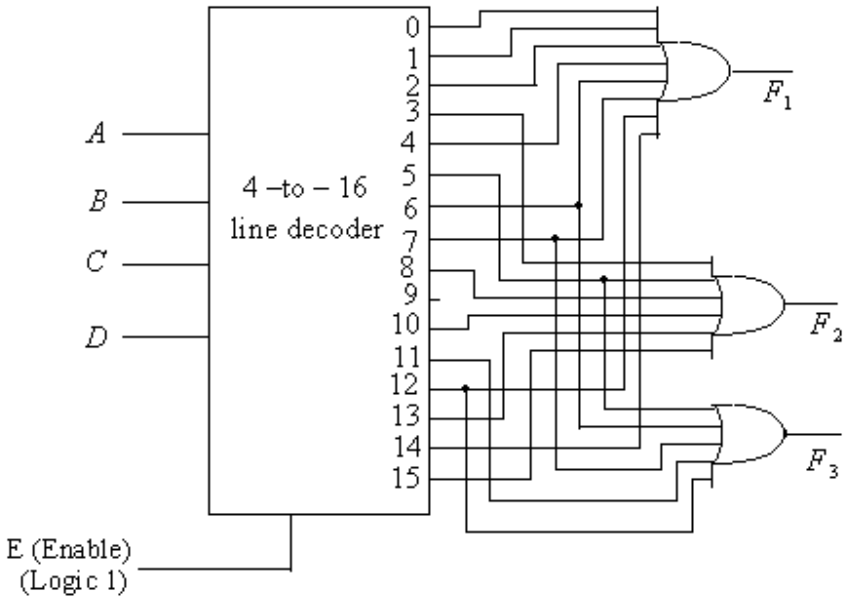


Fig. 6.17

Example 6.5: Implement a full subtractor circuit with a 3 to 8 line decoder and two OR gates.

Solution: The Boolean expressions for Difference D and Borrow B bits of full subtractor are given as follows (refer chapter 5):

$$D = \sum (1,2,4,7)$$

$$B = \sum (1,2,3,7)$$

The realization of these functions with 3 to 8 line decoder and two OR gates, is shown in figure 6.18.

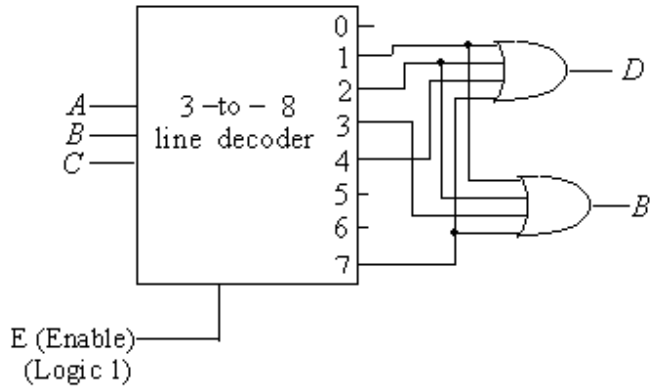


Fig. 6.18

6.3.1 BCD – to – Decimal Decoder: The BCD to Decimal decoder converts each BCD input character (8421 code) into one of ten possible decimal form. It is also referred to as 4 – to – 10 line decoder. The method of implementation is essentially the same as for 4 – to – 16 line decoder discussed above, with the difference that it has only ten decimal digits 0 through 9. The BCD to Decimal decoder is also available in the form of IC. The most commonly used BCD to decimal decoder TTL IC is 74LS42. It is designed using NAND gates, which therefore gives the active low outputs. Figures 6.19(a) and 6.19(b) show the logic and block diagram of BCD to decimal decoder IC 7442 respectively. Table 6.7 shows the truth table of IC 7442.

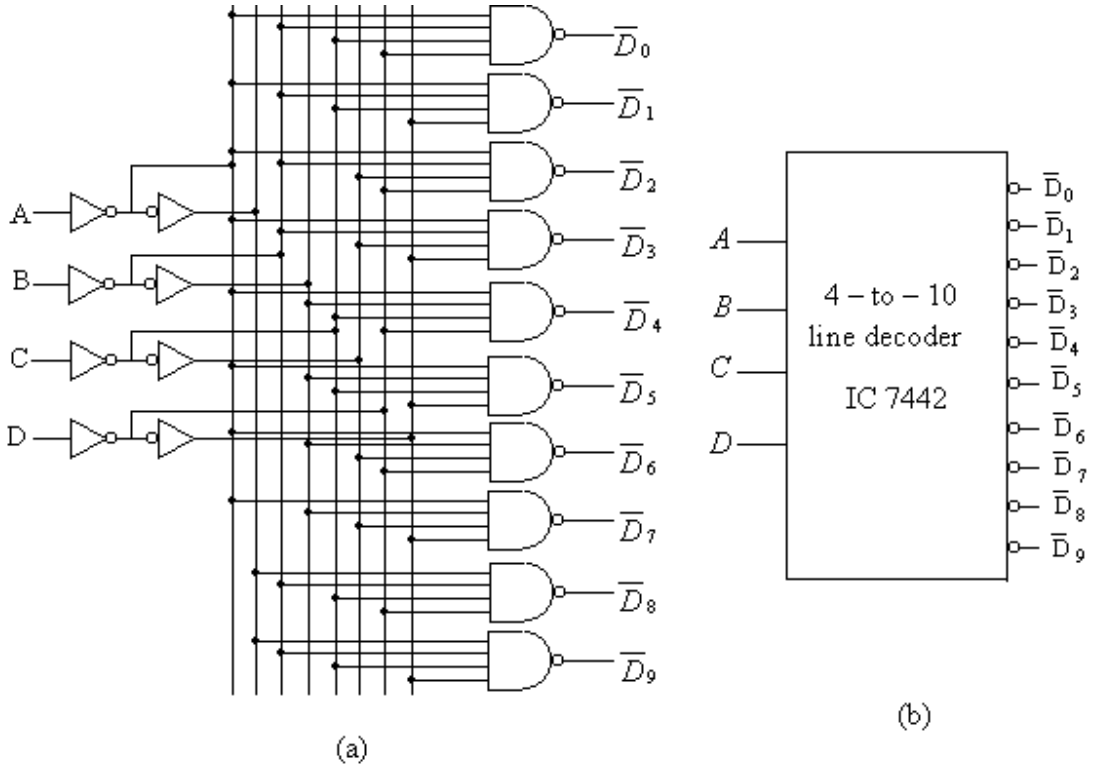


Fig.6.19

Given below the list of most commonly used demultiplexer ICs available in the market:

Description	IC No.
Dual 1:4 DMUX (2:4 line decoder)	74155
1:8 DMUX (3:8 line decoder)	74138
1:16 DMUX (4:16 line decoder)	74154

Table 6.7

BCD input				
A	B	C	D	Output
0	0	0	0	$\overline{D}_0 = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$
0	0	0	1	$\overline{D}_1 = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D$
0	0	1	0	$\overline{D}_2 = \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D}$
0	0	1	1	$\overline{D}_3 = \overline{A} \cdot \overline{B} \cdot C \cdot D$
0	1	0	0	$\overline{D}_4 = \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D}$
0	1	0	1	$\overline{D}_5 = \overline{A} \cdot B \cdot \overline{C} \cdot D$
0	1	1	0	$\overline{D}_6 = \overline{A} \cdot B \cdot C \cdot \overline{D}$
0	1	1	1	$\overline{D}_7 = \overline{A} \cdot B \cdot C \cdot D$
1	0	0	0	$\overline{D}_8 = A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$
1	0	0	1	$\overline{D}_9 = A \cdot \overline{B} \cdot \overline{C} \cdot D$
1	0	1	0	None
1	0	1	1	None
1	1	0	0	None
1	1	0	1	None
1	1	1	0	None
1	1	1	1	None

6.3.2 BCD – to – Seven – Segment Decoder: A decoder for BCD to 7 – segment will now be discussed. A seven segment display consists of seven display lights (segments) arranged in a pattern shown in figure 6.20. The light emitting gallium arsenide or phosphide diodes are generally used for the segments of these display devices. These devices, also known as seven – segment LED display devices, are operated at low voltage and low power and hence directly connected to ICs. The segments of the display devices are marked as a, b, c, d, e, f, g. The numeric digits 0 through 9 may be displayed if the corresponding segments glow as shown in figure 6.20 by the darken segments.

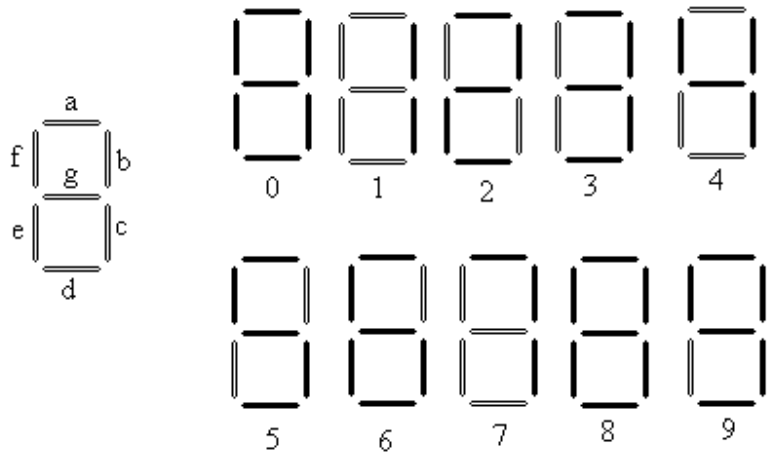


Fig. 6.20

The seven – segment LED display devices are of two types, one is known as common cathode and the other is known as common anode. In the common cathode LED display device, the cathodes of all its LEDs are connected to the common terminal of the device. When the common terminal is grounded and positive voltages are applied to the anodes of the corresponding LEDs of the display device, then the numerals will be displayed on the devices. However, in the common anode LED display devices, the anodes of all its LEDs are connected to the common terminal of the device which is to be connected to the positive supply; and when the low voltages are applied to the anodes of the devices, the numerals are displayed. BCD to seven - segment decoders are available in the form of ICs. The common cathode LED display devices are connected to such BCD to seven segment decoder ICs which provide active high outputs and common anode LED display devices to such decoder ICs which provide active low outputs. Other display devices are LCD (Liquid Crystal Devices).

The design of a combinational circuit will be discussed. It will decode 4 – bit BCD codes to decimal digits. The logic circuit will have 4 inputs and seven outputs (figure 6.21). Seven outputs will correspond to the segments of the display.

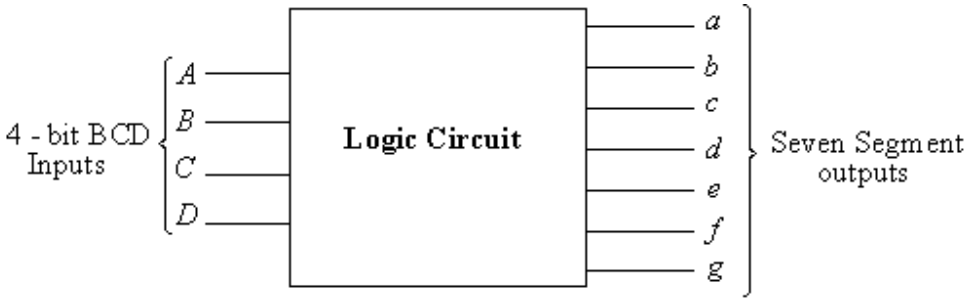


Fig. 6.21

A truth table indicating the 4 – bit BCD inputs and seven segment outputs is shown in table 6.8. Seven segments show the output 1 if it is to glow. The K – maps for the seven segments a, b, c, d, e, f, g are shown in figure 6.22 (a) to (g). From these K – maps the minimal Boolean expressions are obtained for each segment. The expressions are given as:

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

$$b = \overline{B} + C \cdot D + \overline{C} \cdot \overline{D}$$

$$c = B + \overline{C} + D$$

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

$$e = \overline{B} \cdot \overline{D} + C \cdot \overline{D}$$

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

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

The expressions for the seven segments a through d can be implemented using the AND OR and Not gates as shown in figure 6.23.

Table 6.8

BCD Inputs				Seven Segment outputs						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	ø	ø	ø	ø	ø	ø	ø
1	0	1	1	ø	ø	ø	ø	ø	ø	ø
1	1	0	0	ø	ø	ø	ø	ø	ø	ø
1	1	0	1	ø	ø	ø	ø	ø	ø	ø
1	1	1	0	ø	ø	ø	ø	ø	ø	ø
1	1	1	1	ø	ø	ø	ø	ø	ø	ø

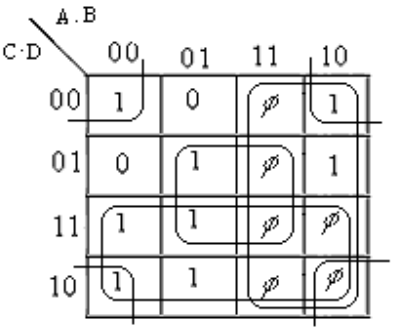


Fig. 6.22(a)

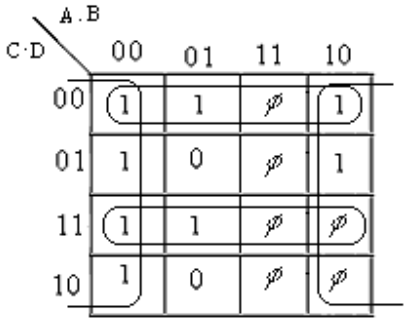


Fig. 6.22(b)

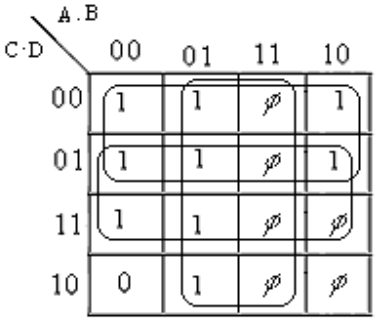


Fig. 6.22(c)

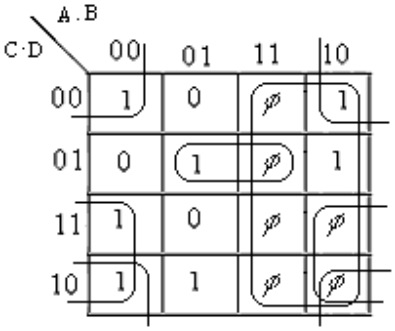


Fig. 6.22(d)

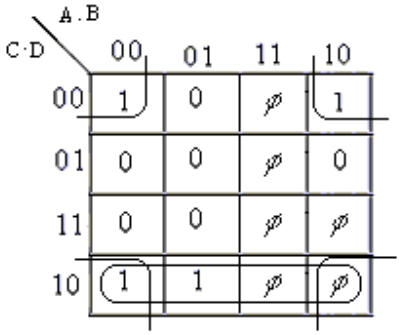


Fig. 6.22(e)

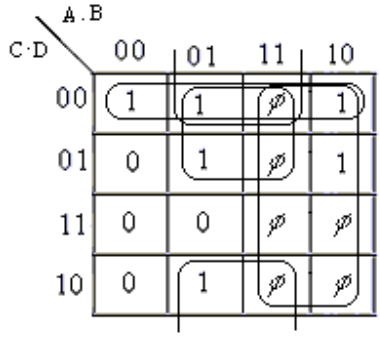


Fig.6.22(f)

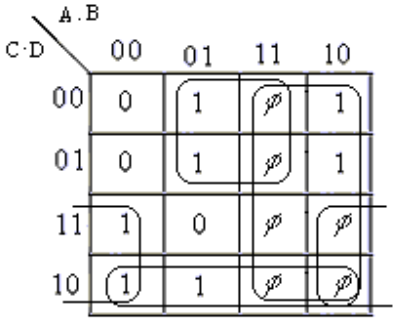


Fig. 6.22(g)

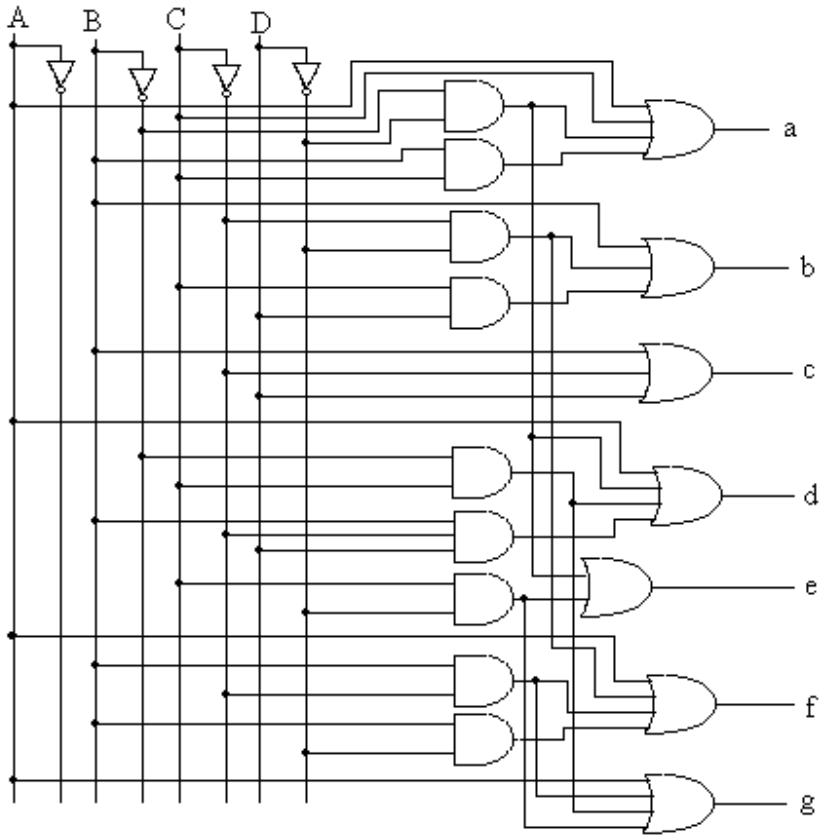


Fig. 6.23

A few ICs are available for BCD to seven segment decoder/driver. ICs 7447 & 7446 are generally used BCD to seven segment decoder/driver. These decoder ICs has four input lines and 7 output lines for each segment of the display device. The both ICs give active low outputs and their pin configuration is same. The maximum voltage rating of IC 7447 is 15 volts where as it is 30 volts for IC 7446. The function of lamp test (LT), Ripple blanking input (RBI), ripple blanking output (RBO) and Blanking inputs are also provided in these decoder ICs. The lamp test is used to check the segments of the display device. If LT is at logic 0 then all the segments of the display device will be ON. For normal operation of the decoder LT should be connected to logic 1. For normal operation of the decoder the ripple blanking input (RBI) should be connected to logic 1. For blanking out leading zeros in multi – digit display, RBI is to be connected to logic 0. The terminal blanking input and ripple blanking out (BI/RBO) is also used for blanking out 0s in multiplexed display. The set up for single seven – segment LED display using BCD – to – seven segment decoder/driver IC 7447 is shown in figure 6.24.

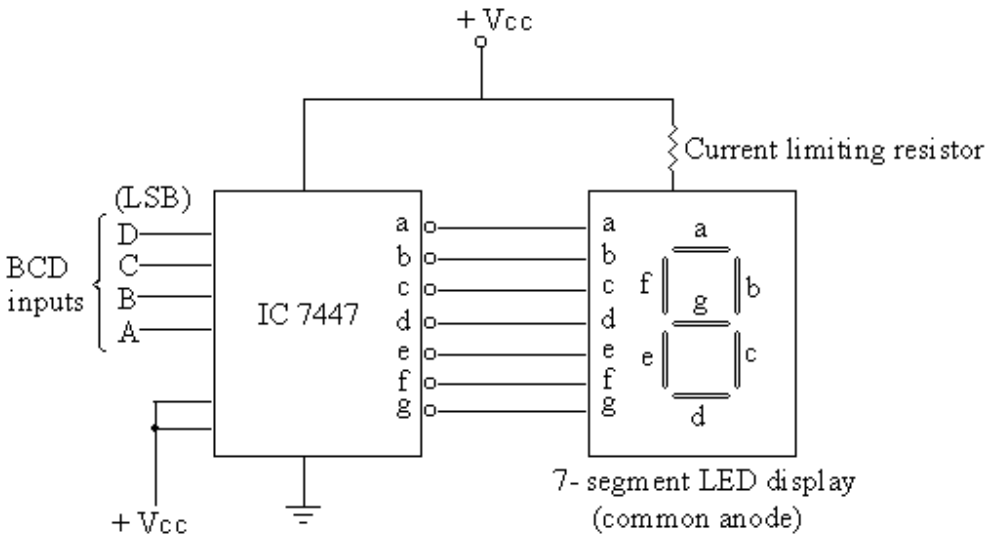


Fig. 6.24

6.4 Code converter: Code converter is most commonly used in digital systems. Sometimes binary numbers are provided in one type of binary codes and required the numbers in other types of binary codes. So the code converter converts the binary numbers provided in one type of codes to other type of codes. The process of code converter is illustrated by taking an example. Suppose it is desired to convert the digits given in 8421 to cyclic code. A truth table is drawn in which four input variables say a,b,c,d are taken for the given code and flour output variables for output variables say X,Y,Z,W are taken for the required code. The binary numbers in the given code are written for the input variables and their corresponding binary numbers in the required code are written for the output variables (table 6.9).

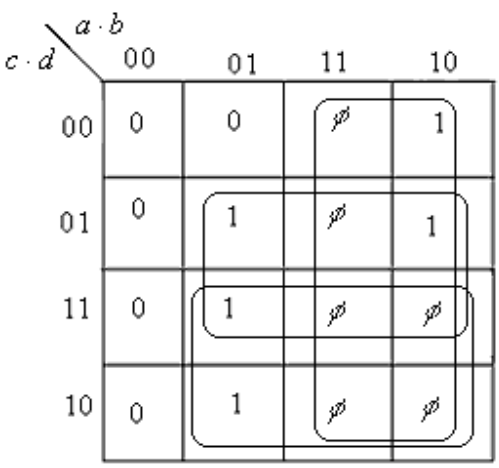
Using the K – map, simplified Boolean functions for each variable in the required code is obtained in terms of the variable of the given code. In the above example of conversion of 8421 code to cyclic code, the Boolean function of the variables of cyclic codes are obtained in terms of the variables of 8421 code. These expressions are given as:

$$X = a + b \cdot d + b \cdot c$$
$$Y = b \cdot \bar{c}$$
$$Z = b + c$$
$$W = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot d + \bar{b} \cdot c \cdot \bar{d} + b \cdot c \cdot d + a \cdot \bar{d}$$

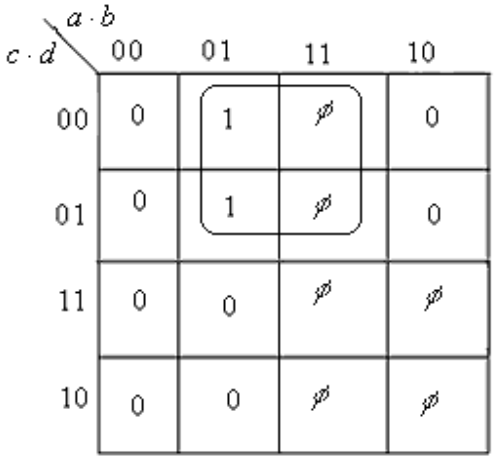
Figure 6.25 shows the K – map for each expression. The realization of these expressions using NAND gates is shown in figure 6.26.

Table 6.9

8 – 4 – 2 – 1 Code				Cyclic Code			
a	b	c	d	X	Y	Z	W
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0



$X = a + b \cdot d + b \cdot c$



$Y = b \cdot \bar{c}$

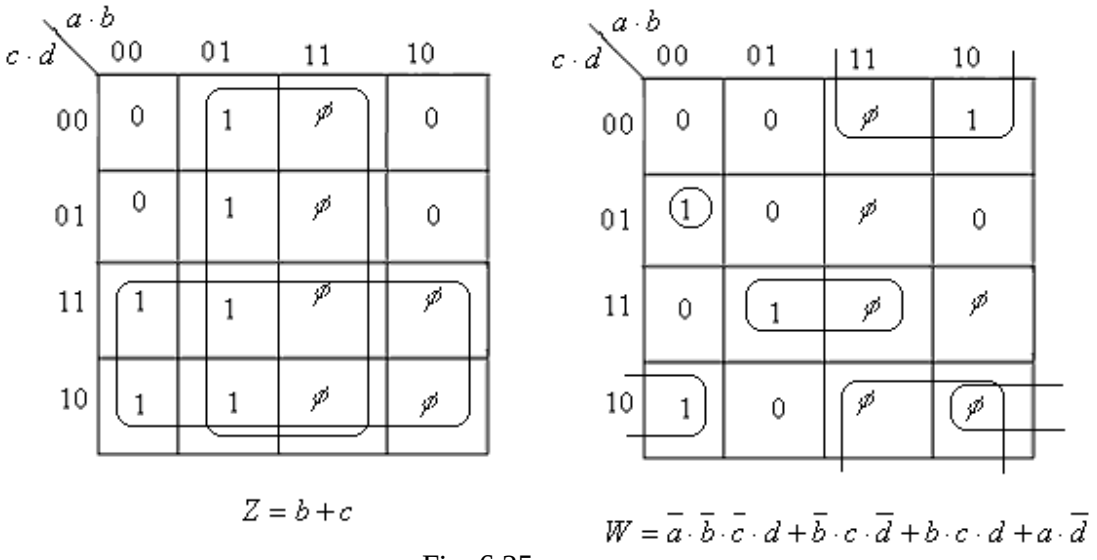


Fig. 6.25

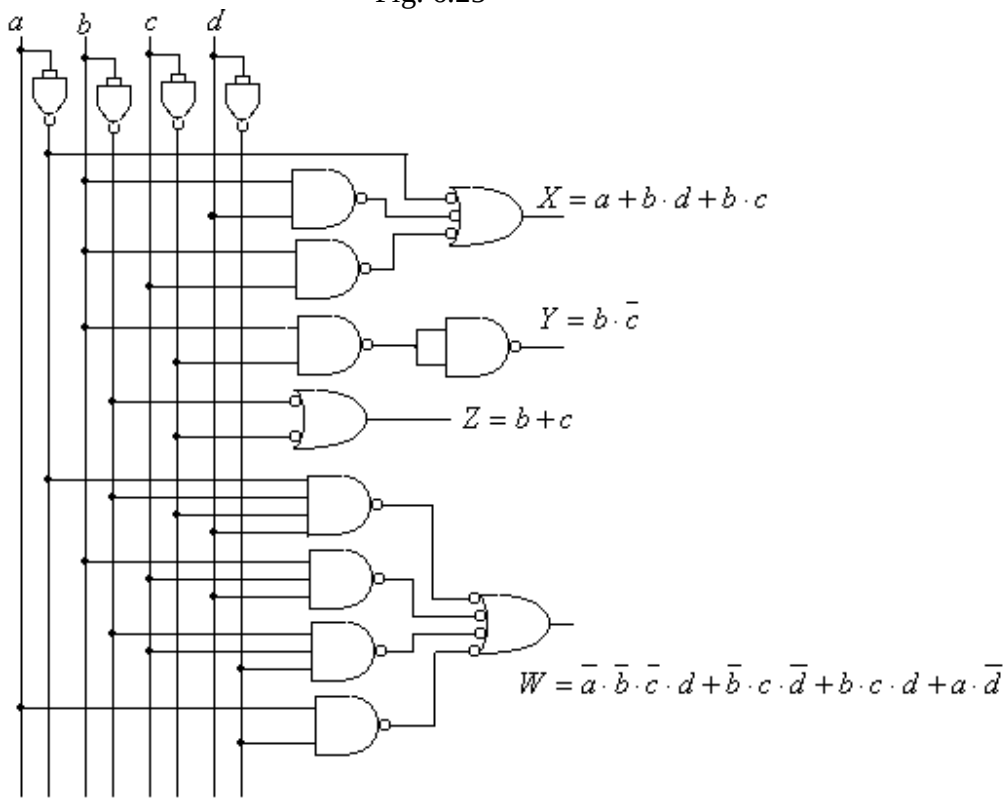


Fig. 6.26

6.5 Encoders: An encoder a combinational circuit which performs the reverse operation of decoder. The decoder accepts N bit input code and activates one of the several out lines corresponding to that code. However, an encoder has a number of input lines, only one of which is activated at a time. It provides the N bit code at the output corresponding to activated input line. The decoders studied in the foregoing section were

binary to octal, BCD to decimal decoder etc. The encoders will therefore, be like octal to binary and decimal to BCD encoders. Figure 6.27 shows the functional block diagram of an encoder having K inputs and N outputs. In the K input lines only one line will be high at a time.

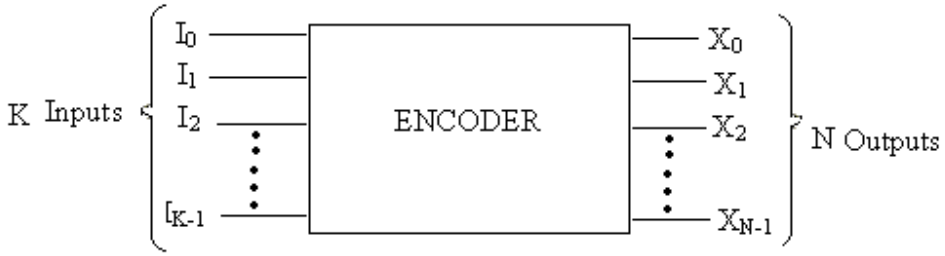


Fig. 6.27

6.5.1 Octal – to – Binary Encoder: An octal – to – binary encoder (also known as 8 – line to 3 – line encoder) has 8 input lines and provides three bit output lines for producing output code corresponding to the activated input line. The truth table for octal – to – binary encoder is given in table 6.10. From this table it may be noted that binary output X_0 gives the logic 1 if any of the input digits D_1 or D_3 or D_5 or D_7 is at logic 1. Therefore the Boolean expression for X_0 is given by:

$$X_0 = D_1 + D_3 + D_5 + D_7$$

Similarly, the expressions for X_1 and X_2 may be given as:

$$X_1 = D_2 + D_3 + D_6 + D_7$$

$$X_2 = D_4 + D_5 + D_6 + D_7$$

The logic circuit for the octal – to – binary encoder with active high inputs is shown in figure 6.28.

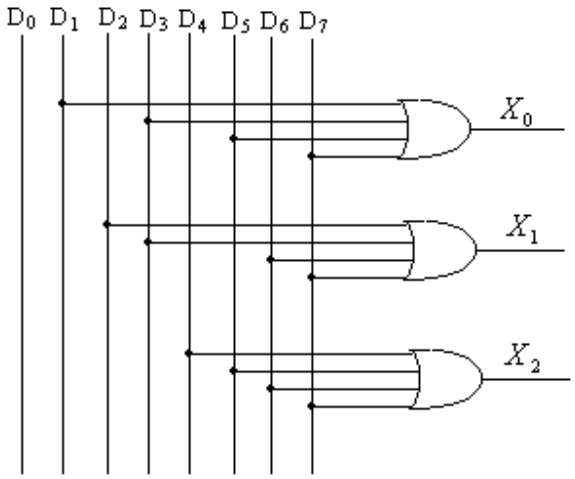


Fig. 6.28

Table 6.10

Octal digits		Outputs		
		X_2	X_1	X_0
0	D_0	0	0	0
1	D_1	0	0	1
2	D_2	0	1	0
3	D_3	0	1	1
4	D_4	1	0	0
5	D_5	1	0	1
6	D_6	1	1	0
7	D_7	1	1	1

6.5.2 Decimal – to – BCD Encoder: The decimal – to – BCD encoder has 10 inputs – one for each decimal digit, and 4 output lines for BCD codes. The logic symbol of this encoder is shown in figure 6.29. Table 6.11 shows the truth table for decimal to BCD encoder. The expressions for the output variables with respect to the truth table are given by:

$$X_0 = D_1 + D_3 + D_5 + D_7 + D_9$$
$$X_1 = D_2 + D_3 + D_6 + D_7$$
$$X_2 = D_4 + D_5 + D_6 + D_7$$
$$X_3 = D_8 + D_9$$

The logic circuit for the decimal – to – BCD encoder with active high inputs is shown in figure 6.30.

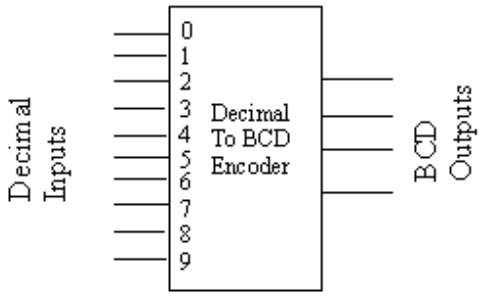


Fig. 6.29

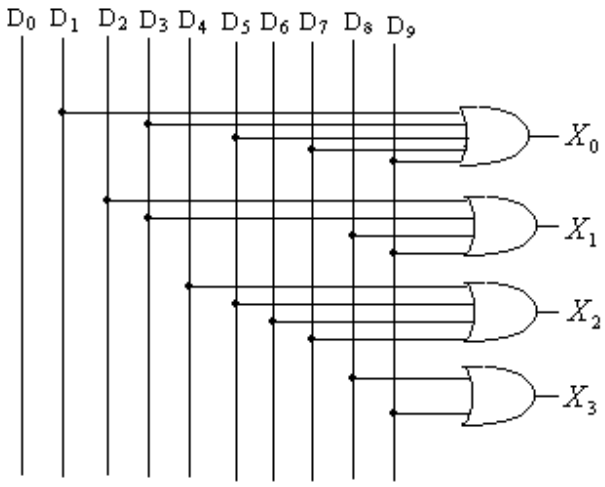


Table 6.11

Decimal Inputs		BCD Outputs			
		X_3	X_2	X_1	X_0
0	D_0	0	0	0	0
1	D_1	0	0	0	1
2	D_2	0	0	1	0
3	D_3	0	0	1	1
4	D_4	0	1	0	0
5	D_5	0	1	0	1
6	D_6	0	1	1	0
7	D_7	0	1	1	1
8	D_8	1	0	0	0
9	D_9	1	0	0	1

Fig. 6.30

6.6 Priority Encoder: In the logic circuit for encoders, it has been discussed that only one of the inputs is kept high at a time and output is obtained corresponding to the high input. But it is worth mentioning that if the two or more inputs are inadvertently activated at a time then undesirable results will be obtained. The priority encoder

performs the same logic function as that of encoder with the additional facility of priority function, when two or more input lines are activated simultaneously. The priority function means that the encoder will provide the output corresponding to the highest order activated input line. Decimal to BCD priority encoder will now be discussed in detail.

6.6.1 Decimal – to – BCD Priority Encoder: Decimal – to – BCD priority encoder should have ten input lines D_0 through D_9 and four output lines X_0 to X_3 like normal Decimal to BCD encoder. The additional facility provided in the priority encoder is that when two more lines say D_4 and D_8 are activated simultaneously, the BCD output will be available corresponding to the line which has higher number i.e. the output will be available corresponding to D_8 line. The additional logic circuitry will provide the priority function to the encoder. This is accomplished as follows:

Referring to the table 6.11, the truth table for the decimal to BCD encoder, X_0 is high when D_1 or D_3 or D_5 or D_7 or D_9 is high. But for priority function, D_0 must be allowed to activate the output X_0 only if no higher order digits other than those that also activate X_0 are high. This can be stated as:

X_0 is high if D_1 is high and D_2, D_4, D_6 and D_8 are low, OR

D_3 is high and D_4, D_6, and D_8 are low, OR

D_5 is high and D_6 and D_8 are low, OR

D_7 is high and D_8 is low, OR

D_9 is high.

The above statements can be expressed in the form of expression for X_0 as:

$$X_0 = D_1 \cdot \overline{D}_2 \cdot \overline{D}_4 \cdot \overline{D}_6 \cdot \overline{D}_8 + D_3 \cdot \overline{D}_4 \cdot \overline{D}_6 \cdot \overline{D}_8 + D_5 \cdot \overline{D}_6 \cdot \overline{D}_8 + D_7 \cdot \overline{D}_8 + D_9$$

The statements for getting the expression for output X_1 are:

X_1 is high when D_2 or D_3 or D_6 or D_7 is high. So for priority encoder

X_1 is high if D_2 is high and D_4, D_5, D_8 and D_9 are low, OR

D_3 is high and D_4, D_5, D_8 and D_9 are low, OR

D_6 is high and D_8 and D_9 are low, OR

D_7 is high and D_8 and D_9 are low.

The expression for X_1 is of the form:

$$X_1 = D_2 \cdot \overline{D}_4 \cdot \overline{D}_5 \cdot \overline{D}_8 \cdot \overline{D}_9 + D_3 \cdot \overline{D}_4 \cdot \overline{D}_5 \cdot \overline{D}_8 \cdot \overline{D}_9 + D_6 \cdot \overline{D}_8 \cdot \overline{D}_9 + D_7 \cdot \overline{D}_8 \cdot \overline{D}_9$$

The statements for getting the expression for output X_2 are:

X_2 is high when D_4 or D_5 or D_6 or D_7 is high. So for priority encoder

X_2 is high if D_4 is high and D_8 and D_9 are low, OR

D_5 is high and D_8 and D_9 are low, OR

D_6 is high and D_8 and D_9 are low, OR

D_7 is high and D_8 and D_9 are low.

The expression for X_2 is of the form:

$$X_1 = D_4 \cdot \overline{D}_8 \cdot \overline{D}_9 + D_5 \cdot \overline{D}_8 \cdot \overline{D}_9 + D_6 \cdot \overline{D}_8 \cdot \overline{D}_9 + D_7 \cdot \overline{D}_8 \cdot \overline{D}_9$$

Similarly, the statements for getting the expression for output X_3 are:

X_3 is high when D_8 or D_9 is high. So for priority encoder

X_3 is high if D_8 is high or D_9 is high.

The expression for X_3 is, therefore, given by:

$$X_3 = D_8 + D_9$$

The logic circuit diagram for the Decimal – to – BCD priority encoder is shown in figure 6.31 with active high outputs.

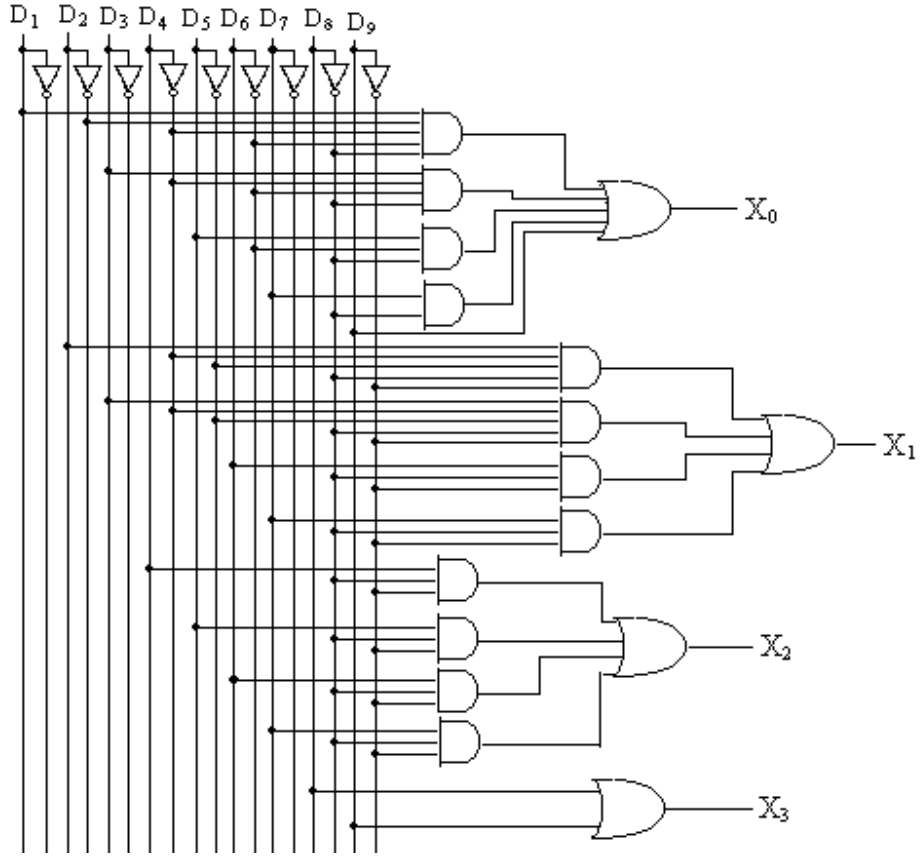


Fig. 6.31

Decimal to BCD priority encoder, is available in the form of IC 74147. The input and output variables in this IC are active low. The block diagram of this IC is shown in figure 6.32 and table 6.12 illustrates its truth table.

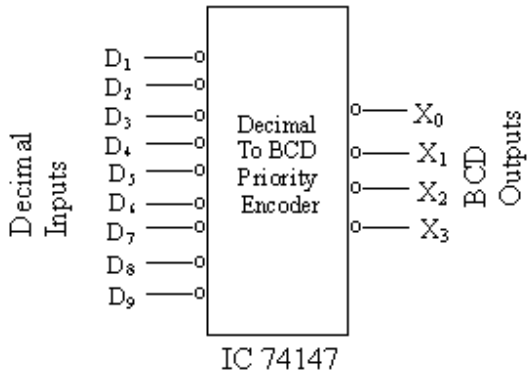


Fig. 6.32

Table 6.12

Inputs									Outputs			
D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	X_3	X_2	X_1	X_0
1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0
ø	0	1	1	1	1	1	1	1	1	1	0	1
ø	ø	0	1	1	1	1	1	1	1	1	0	0
ø	ø	ø	0	1	1	1	1	1	1	0	1	1
ø	ø	ø	ø	0	1	1	1	1	1	0	1	0
ø	ø	ø	ø	ø	0	1	1	1	1	0	0	1
ø	ø	ø	ø	ø	ø	0	1	1	1	0	0	0
ø	ø	ø	ø	ø	ø	ø	0	1	0	1	1	1
ø	ø	ø	ø	ø	ø	ø	ø	0	0	1	1	0

6.6.2 Octal to Binary Priority Encoder: The block diagram of octal to binary priority encoder IC 74148 is shown in figure 6.33, and the truth table for the same in given in table 6.13. The internal logic circuit for this IC has active low inputs and active low

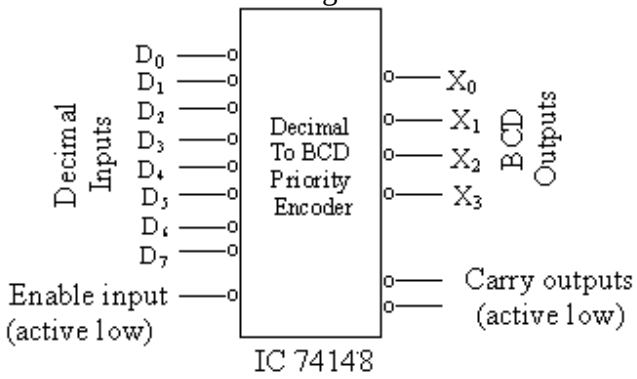


Fig. 6.33

Table 6.13

EI	Inputs								Outputs				
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X_2	X_1	X_0	GS	Eo
1	ø	ø	ø	ø	ø	ø	ø	ø	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	ø	0	1	1	1	1	1	1	1	1	0	0	1
0	ø	ø	0	1	1	1	1	1	1	0	1	0	1
0	ø	ø	ø	0	1	1	1	1	1	0	0	0	1
0	ø	ø	ø	ø	0	1	1	1	0	1	1	0	1
0	ø	ø	ø	ø	ø	0	1	1	0	1	0	0	1
0	ø	ø	ø	ø	ø	ø	0	1	0	0	1	0	1
0	ø	ø	ø	ø	ø	ø	ø	0	0	0	0	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0

outputs. One enable input is provided in this IC which is also active low. Two active low carry outputs are also provided in the IC. The enable input and carry output help to cascade circuits to handle more inputs. Very useful circuits such as hexadecimal to binary encoder are designed by cascading octal to binary priority encoders. A hexadecimal to binary priority encoder finds wide use in computers and microprocessors etc.

6.7 Magnitude Comparator: Magnitude comparator also called as the magnitude digital (or binary) comparator. It compares and indicates if the binary number P is equal to or greater than or less than the other binary number Q. Let P_0 and Q_0 are the two bits to be compared. The result for the equality of these two bits may be given by XNOR gate. The XNOR gate gives an output as logic 1 if two bits are equal otherwise logic 0. This condition is given by:

$$T = \overline{\overline{P_0} \cdot Q_0 + P_0 \cdot \overline{Q_0}} = \overline{P_0 \oplus Q_0} = \begin{cases} 1 & P_0 = Q_0 \\ 0 & P_0 \neq Q_0 \end{cases}$$

The condition $P_0 > Q_0$ is given by:

$$R = P_0 \cdot \overline{Q_0} = \begin{cases} 1 \;\; if \;\; P_0 > Q_0 \\ 0 \;\; if \;\; P_0 \leq Q_0 \end{cases}$$

In this expression if $P_0 > Q_0$ (P_0 = 1 and Q_0 = 0), R = 1 and if on the other hand $P_0 \leq Q_0$ (P_0 = 0 and Q_0 = 1 or $P_0 = Q_0$ = 0 or $P_0 = Q_0$ = 1), R = 0.

The condition $P_0 < Q_0$ is given by:

$$S = \overline{P_0} \cdot Q_0 = \begin{cases} 1 \;\; if \;\; P_0 < Q_0 \\ 0 \;\; if \;\; P_0 \geq Q_0 \end{cases}$$

It is clear from this expression that if $P_0 < Q_0$ (P_0 = 0 and Q_0 = 1), S = 1 and if on the other hand $P_0 \geq Q_0$ (P_0 = 1 and Q_0 = 0 or $P_0 = Q_0$ = 0 or $P_0 = Q_0$ = 1), S = 0.

The logic diagram for one bit comparator is shown in figure 6.34.

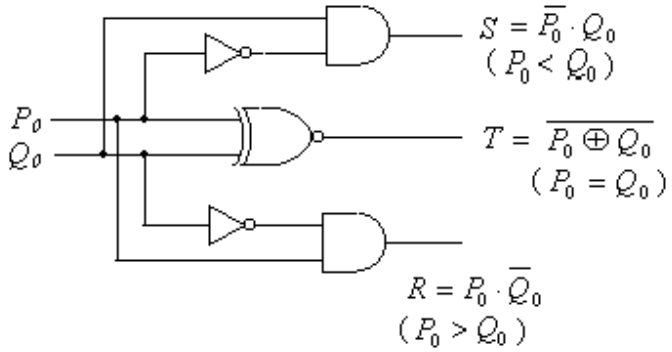


Fig. 6.34

Now the comparator which compares two unsigned four – bit binary numbers will be discussed. Let $P_3P_2P_1P_0$ and $Q_3Q_2Q_1Q_0$ are the two unsigned binary numbers of four bits. The comparator gives three outputs indicating if Ps > Qs or Ps = Qs or Ps < Qs. The two binary numbers will be equal if and only if $P_3 = Q_3$, $P_2 = Q_2$, $P_1 = Q_1$ and $P_0 = Q_0$. The logic expression for the equality of two binary numbers will be given by the AND operation of the equality of the individual bit (XNOR of individual bit), as:

$$(Ps = Qs) = (\overline{P_3 \oplus Q_3}) \cdot (\overline{P_2 \oplus Q_2}) \cdot (\overline{P_1 \oplus Q_1}) \cdot (\overline{P_0 \oplus Q_0})$$

The statements for getting the expression for *Ps* > *Qs* are:

Ps will be greater than *Qs*

if $P_3 = 1$ and $P_3 = 0$ OR

if $P_3 = Q_3$ and if $P_2 = 1$ and $Q_2 = 0$ OR

if $P_3 = Q_3$, and if $P_2 = Q_2$, and if $P_1 = 1$ and $Q_1 = 0$ OR

if $P_3 = Q_3$, and if $P_2 = Q_2$, and if $P_1 = Q_1$ and if $P_0 = 1$ and $Q_0 = 0$

These statements can be expressed in the form of expression for Ps > Qs as:

$$(Ps > Qs) = P_3 \cdot \overline{Q}_3 + (\overline{P_3 \oplus Q_3}) \cdot P_2 \cdot \overline{Q}_2 + (\overline{P_3 \oplus Q_3}) \cdot (\overline{P_2 \oplus Q_2}) \cdot P_1 \cdot \overline{Q}_1 + (\overline{P_3 \oplus Q_3}) \cdot (\overline{P_2 \oplus Q_2}) \cdot (\overline{P_1 \oplus Q_1}) \cdot P_0 \cdot \overline{Q}_0$$

Similarly, the statements for getting the expression for Ps < Qs are:

Ps will be less than *Qs*

if $P_3 = 0$ and $P_3 = 1$ OR

if $P_3 = Q_3$ and if $P_2 = 0$ and $Q_2 = 1$ OR

if $P_3 = Q_3$, and if $P_2 = Q_2$, and if $P_1 = 0$ and $Q_1 = 1$ OR

if $P_3 = Q_3$, and if $P_2 = Q_2$, and if $P_1 = Q_1$ and if $P_0 = 0$ and $Q_0 = 1$

These statements give the expressions for Ps < Qs as:

$$(Ps < Qs) = \overline{P}_3 \cdot Q_3 + (\overline{P_3 \oplus Q_3}) \cdot \overline{P}_2 \cdot Q_2 + (\overline{P_3 \oplus Q_3}) \cdot (\overline{P_2 \oplus Q_2}) \cdot \overline{P}_1 \cdot Q_1 + (\overline{P_3 \oplus Q_3}) \cdot (\overline{P_2 \oplus Q_2}) \cdot (\overline{P_1 \oplus Q_1}) \cdot \overline{P}_0 \cdot Q_0$$

Figure 6.35 shows the implementation of the three expressions for *Ps* > *Qs*, *Ps* = *Qs* and *Ps* < *Qs* of the four bit magnitude comparator. The outputs of this comparator are active high.

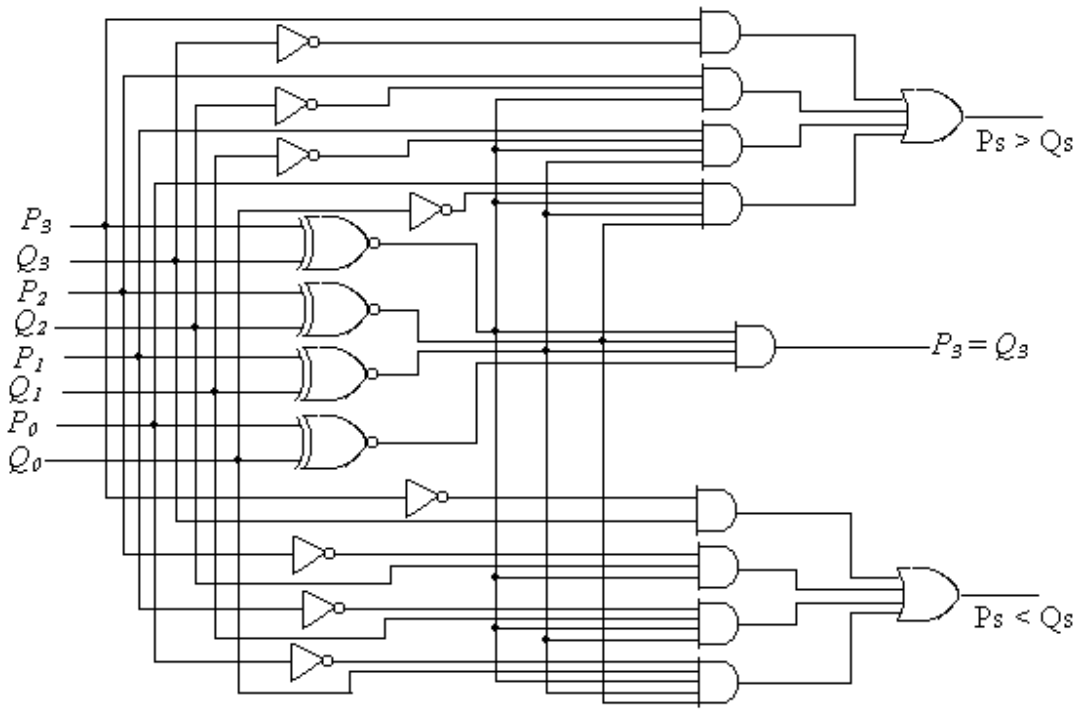


Fig. 6.35

Figure 6.36 shows the logic diagram of 4 – bit comparator IC 7485. This IC has 4 input and 4 output terminals with active high, and in addition it has three cascading inputs. These inputs allow several comparators to cascade for comparison of any number of bits. For the expansion of the comparators, Ps > Qs, Ps = Qs and Ps < Qs outputs of the one comparator (to which the least significant data is connected) to the corresponding cascading inputs of the second comparator (to which next significant data is connected). The cascading inputs of the first comparator (to which the least significant data is connected) must be connected as follows:

Cascading Input "= " to logic 1 and cascading inputs ">" and "<" to logic 0.

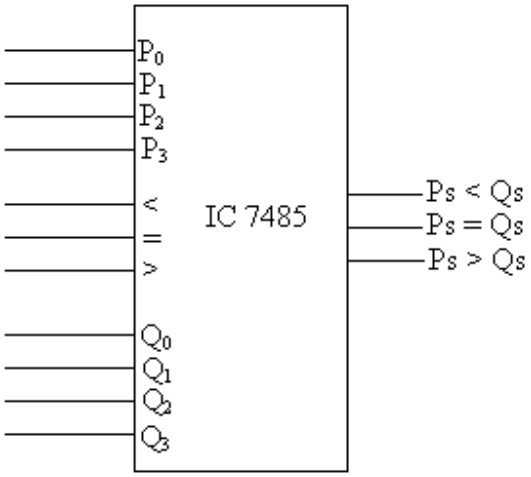


Fig. 6.36

Cascading of two comparators (ICs 7485) to compare the magnitudes of two 8 – bit binary numbers is shown in figure 6.37.

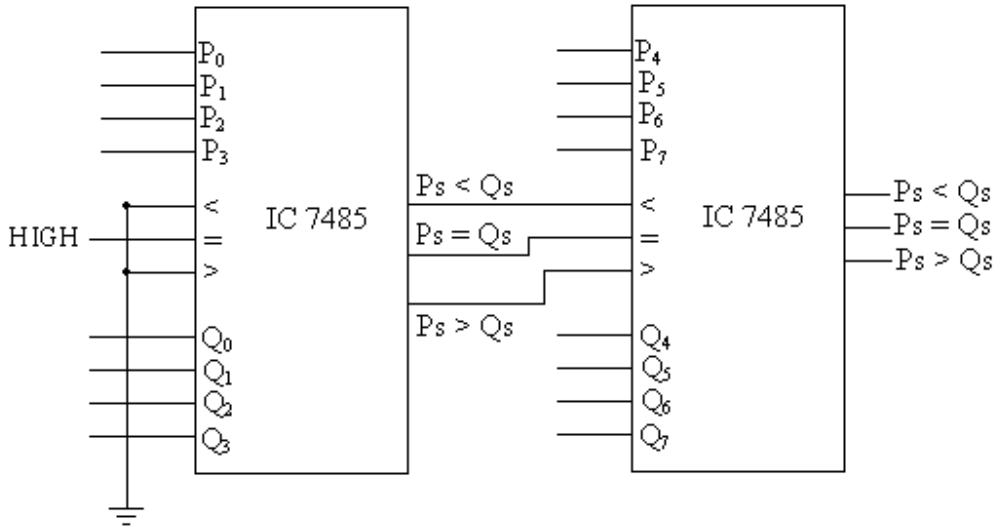


Fig. 6.37

6.8 Parity Generator/ Checker: In some digital systems the data or information in the form of the binary bits is sent from one block or the system to the other block or the system. In the transmission of the data, error may occur due to change of data bit (0 by 1 or vice versa). This change may be due to component malfunctions or the electrical noise. This problem is removed by adding one additional bit in the data to be transmitted. This extra bit is known as parity bit. The parity bit detects the single error in the transmission. Parity is the number of 1's in the given data or word. If the number of 1's in the given data is even then parity is called as even parity; if on the other hand the number of 1's is odd then the parity is called as odd parity. The parity bit of the data or the word is generated by the parity generator. The logic diagram of the parity bit generator of four bit is shown in figure 6.38. This parity generator gives output P (parity bit) as logic 1 if the

number of 1's in the four bit input data is even; and P is logic 0 if the number of 1's in the four bit input data is odd. That is for even parity of the input data, output is 1 and for odd parity of the input data, output is 0.

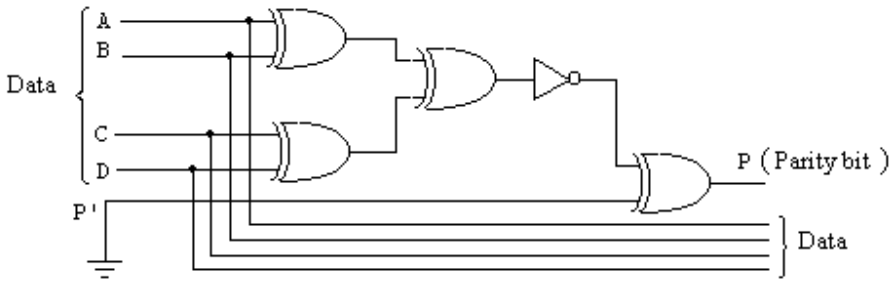


Fig. 6.38

The parity bit P generated by the parity generator is sent along with the data and at the receiver end data as well as the parity bit is checked by the parity checker. The logic circuit for the parity checker (fig.6.39) is the same as that of the parity generator with the only difference that in the parity checker the terminal P' in not grounded, but the parity bit received at the receiver end is connected to the point P'. So at the receiver the received data and the parity bit form the five bit data which is always having the odd parity. It is clear from the fact that if the data A, B, C and D is odd (even) then parity bit is 0 (1), and therefore the received data and the parity bit is always is odd.

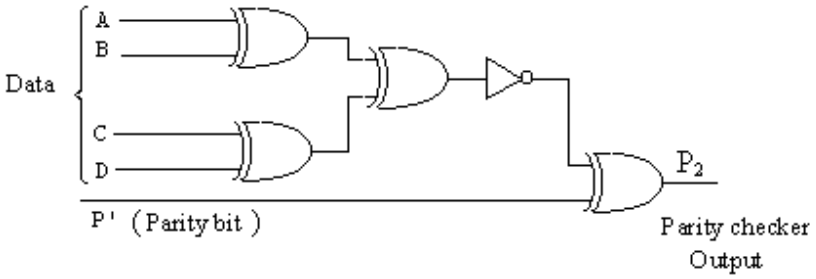


Fig. 6.39

As illustrated in figure 6.40, a parity bit P_1 is generated and transmitted along with the data. At the receiver, the received data and parity bit are tested. If the output P_2 of the checker is 0, then no error is there in the received data. If on the other hand output P_2 is 1, then there is an error in the received data.

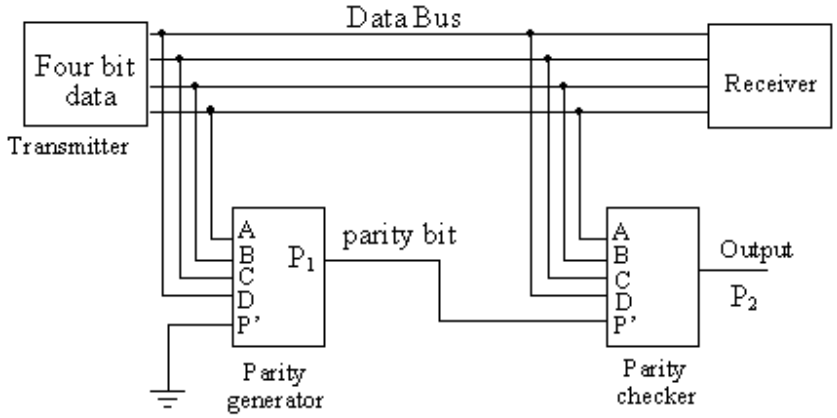


Fig. 6.40

Parity generator/checker is available in the form of IC. Figure 6.41 shows the block diagram of 8 bit parity generator/checker IC 74180 and its truth table is given in table 6.14. This IC can be used to check for even or odd parity on a 9 – bit code (8 – bit data and one parity bit). It can also be used to generate a 9 – bit even or odd parity code.

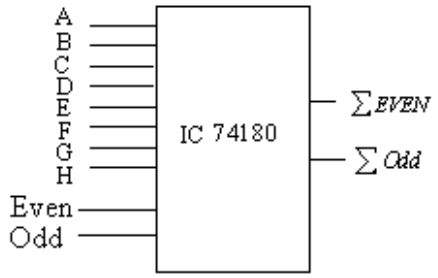


Fig. 6.41

Table 6.14

Parity of	Inputs		Outputs	
8 bit data	Even	Odd	$\sum_{even}$	$\sum_{Odd}$
Even	1	0	1	0
Odd	1	0	0	1
Even	0	1	0	1
Odd	0	1	1	0
∅	1	1	0	0
∅	0	0	1	1

6.9 Programmable Logic Devices: Different gates and other combinational logic circuits available in the form of ICs are used for the logic designs. In many system designs, the designers use large number of ICs, since such circuits have several input and output variables. The recent development of programmable logic devices has presented a cost effective method of realizing such circuits. The programmable logic devices (PLDs) are medium scale integrated circuits and these devices can replace a number of standard ICs. Thus PLDs help in designing larger circuit on small space with ease.

A PLD is a programmable IC which contains large number of interconnected gates, flip – flops and registers etc. Many of the interconnections are fusible. The connections which are not required by the designers are fused or broken. Programming of fuse blowing as per the required circuit pattern is done by the manufacturer or by the customer.

PLDs fall into three categories. They are known as:

(i) Field Programmable Logic Array (FPLA)

(ii) Programmable Array logic

(iii) Programmable Read Only Memory (PROM)

PLDs consist of an array of AND gates followed by an array of OR gates. Both true and complement form of the input variables are fed to AND array. Simplified procedure is adopted to represent the internal circuitry of these devices. Figure 6.42 demonstrate the connection to an AND gate. The circled cross marks ($\otimes$) to the input lines shows the fusible connections to the input lines. If there is no circled cross mark, it indicates that the connection has been broken or fused. Further, the dot marks ($\bullet$) on the input lines show the hard wire connections to the corresponding input lines. Figure 6.42 (a) indicates a four input AND gate with fusible connections to $A, \overline{A}, B, \overline{B}$ inputs. If the connections are fused to A and B inputs, then no mark will be shown to these points refer figure 6.42 (b). The output of this AND gate is $\overline{A} \cdot \overline{B}$. Similarly, the dot marks ($\bullet$) to the input lines (figure 6.42 c) indicate the hardwire connection to the input lines. The output of this gate is $\overline{A} \cdot B$. Similar connections are used for OR array also.

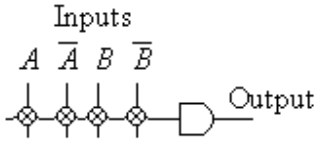


Fig. 6.42 (a)

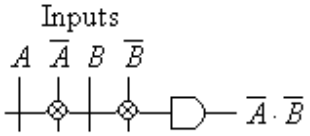


Fig. 6.42(b)

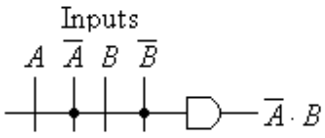


Fig. 6.42(c)

6.9.1 Field Programmable Logic Array (FPLA): Figure 6.43 demonstrates the basic structure of Field Programmable Logic array (FPLA). In this logic device, both AND array and OR array are programmable. The circled cross marks to the input lines of AND and OR gates indicate that these connections are fusible or programmable. It may be noted from the architecture of the FPLA that when it is not programmed all the true and complemented variables are connected to the inputs of each AND gate. The AND gates will give the outputs 0, since $X_0 \cdot \overline{X}_0 \cdot X_1 \cdot \overline{X}_1 \cdots = 0$. The outputs of the OR gates will also be zero since it is the summation of outputs of all AND gates. So when FPLA is not programmed all outputs of the device will be zero. Now if the circled cross mark of some inputs of AND gate are burnt (or programmed to remove fuse), then min-term of the remaining input variables (or their complements) will be obtained. Similarly by burning the unused circled cross marks of OR array will give the required outputs in SOP form. So the programming of the device allows the implementation of arbitrary logic functions in a two level sum – of – product (SOP) form. The AND array creates the required min-terms, while the OR array takes the sum of products to form the outputs. It is very versatile since both AND and OR arrays are programmable. However, it has some disadvantages; it is more difficult to manufacture, program or test than other PLDs.

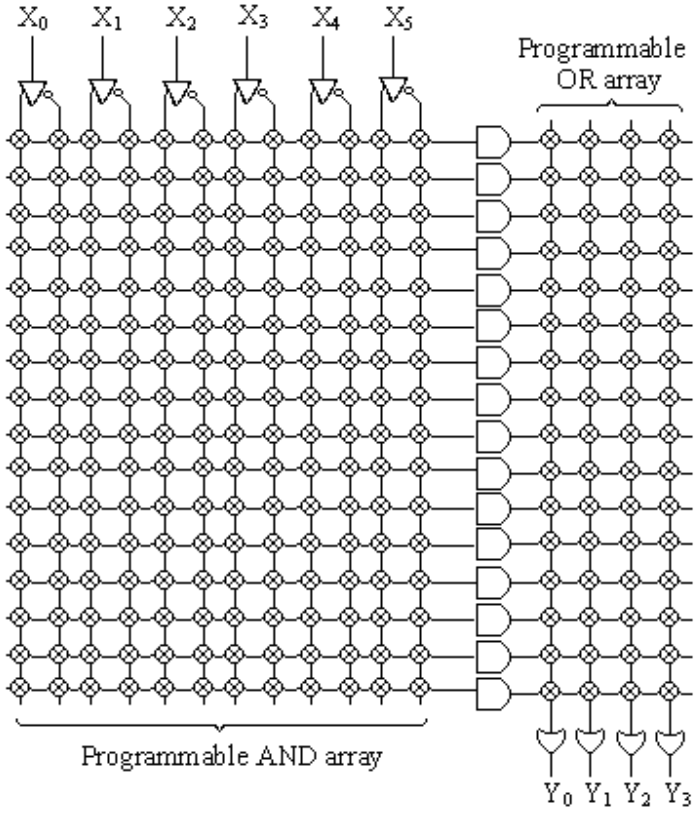


Fig. 6.43

FPLA has a number of input variables, AND gates and OR gates. The actual FPLA available are specified *by p x q x r*, where *p* is the number of input variables, *q* is the number of AND gates and *r* is the number of OR gates (outputs). One FPLA 840 has 14 input variables, 32 AND gates and 6 OR gates.

Example 6.6: Consider a FPLA of 4 input variables, 10 AND gates and 4 OR gates

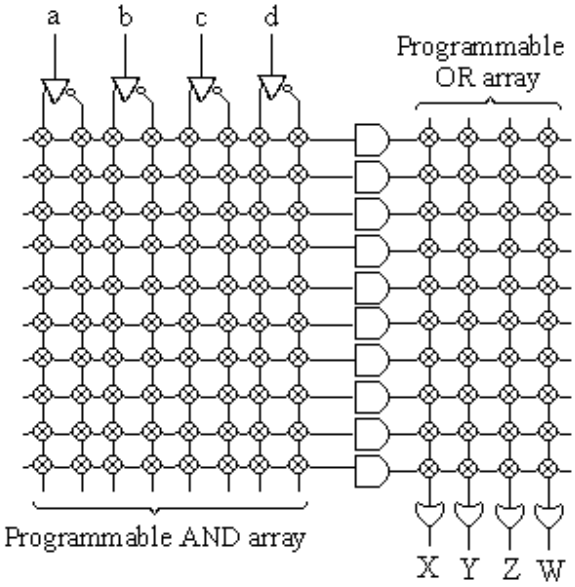


Fig. 6.44

shown in figure 6.44. How would it be programmed to implement the logic circuit for 8421 code to cyclic code converter?

Solution: In section 6.4, logic circuit for 8421 code to cyclic code has been implemented using AND, OR and NOT gates. The Boolean expressions for four variables X, Y, Z, and W of cyclic code given in terms of a, b, c and d variables of 8421 code are reproduced below (from section 6.4) as:

$$X = a + b \cdot d + b \cdot c$$

$$Y = b \cdot \bar{c}$$

$$Z = b + c$$

$$W = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot d + \bar{b} \cdot c \cdot \bar{d} + b \cdot c \cdot d + a \cdot \bar{d}$$

These expressions are realized using FPLA as shown in figure 6.45.

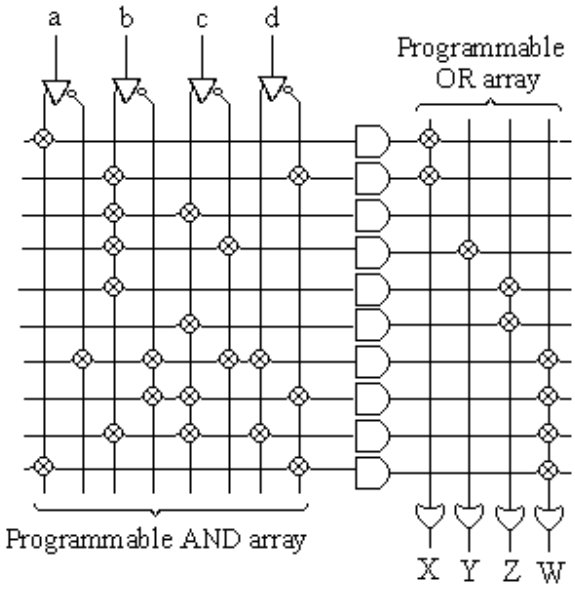


Fig. 6.45

Example 6.7: Implement a BCD to seven segment decoder circuit using a FPLA of proper specification.

Solution: The expressions for the seven segments of BCD to seven segment decoder are given by after K – map minimization (as discussed in section 6.3.2) as:

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

$$b = B + C \cdot D + \bar{C} \cdot \bar{D}$$

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

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

$$e = \bar{B} \cdot \bar{D} + C \cdot \bar{D}$$

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

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

There are 15 independent min-terms in these expressions. So for their realization the FPLA should have 4 input variables, 15 AND gates and 6 OR gate. The realization of 7 outputs of BCD to seven segment display is shown in figure 6.46.

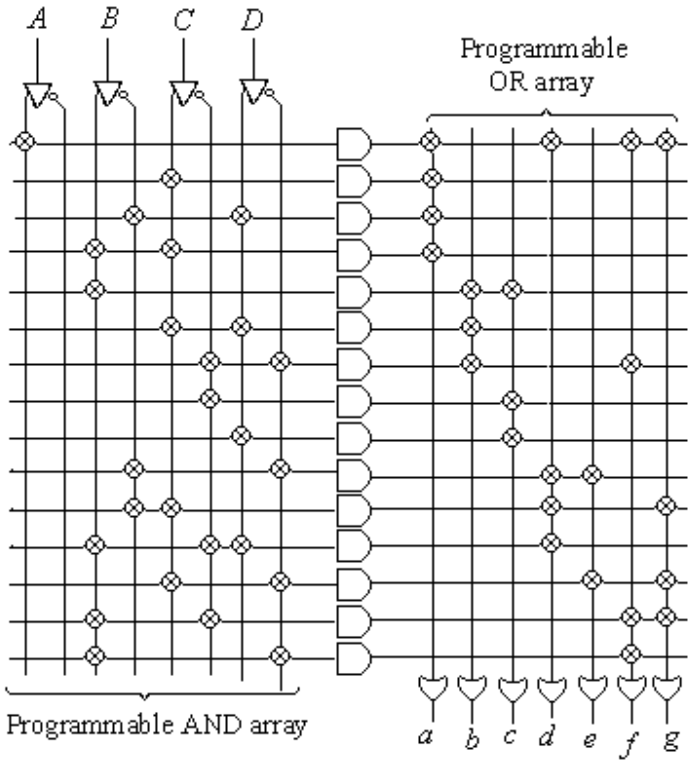


Fig. 4.46

6.9.2 Programmable Array Logic (PAL): Another class of Programmable logic devices is the programmable array logic (PAL) which is widely used and easily programmable. The PAL has an AND array followed by OR array similar to FPLA, with the difference that the inputs to AND array are programmable while the inputs to OR gates are hard wired (fixed OR array). Figure 6.47 shows the architecture of a PAL device having 16 AND gates and four OR gates. Every AND gate can be programmed to generate any desired product of 6 input variables and their complements. Each OR gate is hard wire to only four AND outputs. This limits each output function to four min-terms. If the function requires more than four product terms then one has to use such a PAL which has more OR inputs. If on the other hand one needs less than four product terms, the unneeded terms to the input of OR gate are made 0 by not burning (or programming) the corresponding input lines of AND gates. The PAL structure is the most generic one for the implementation of arbitrary logic functions.

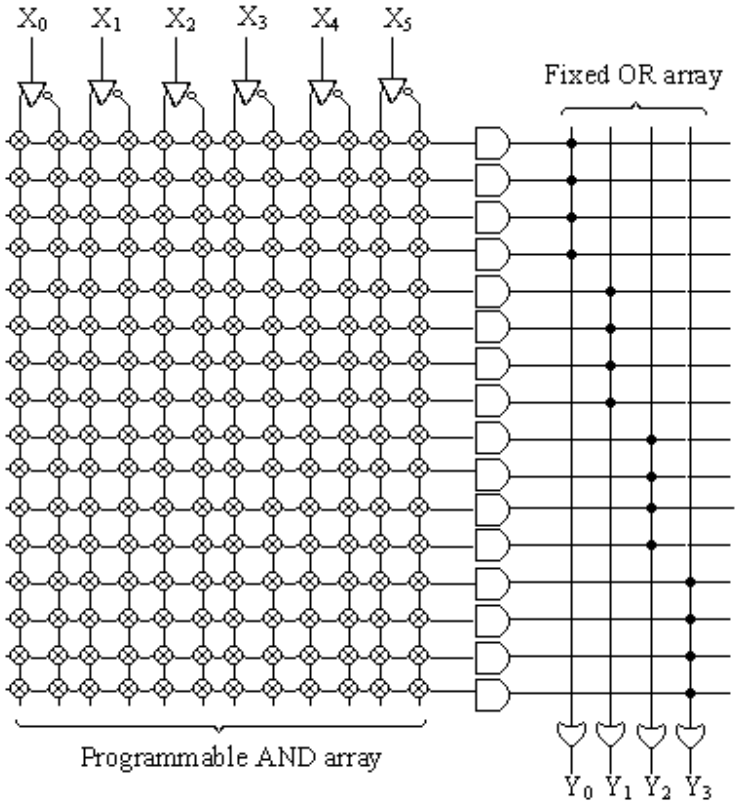


Fig. 6.47

Example 6.8: Using the PAL shown in figure 6.47, implement the following SOP functions of 4 variables.

$$Y_0 = \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D + A \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot C \cdot D$$

$$Y_1 = A \cdot \overline{B} \cdot C \cdot \overline{D}$$

$$Y_2 = \overline{A} \cdot \overline{B} + D + \overline{C}$$

$$Y_3 = \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} + A \cdot B \cdot \overline{C} \cdot \overline{D} + B \cdot C \cdot D$$

Solution: Figure 6.48 shows the implementation of the given functions using the PAL.

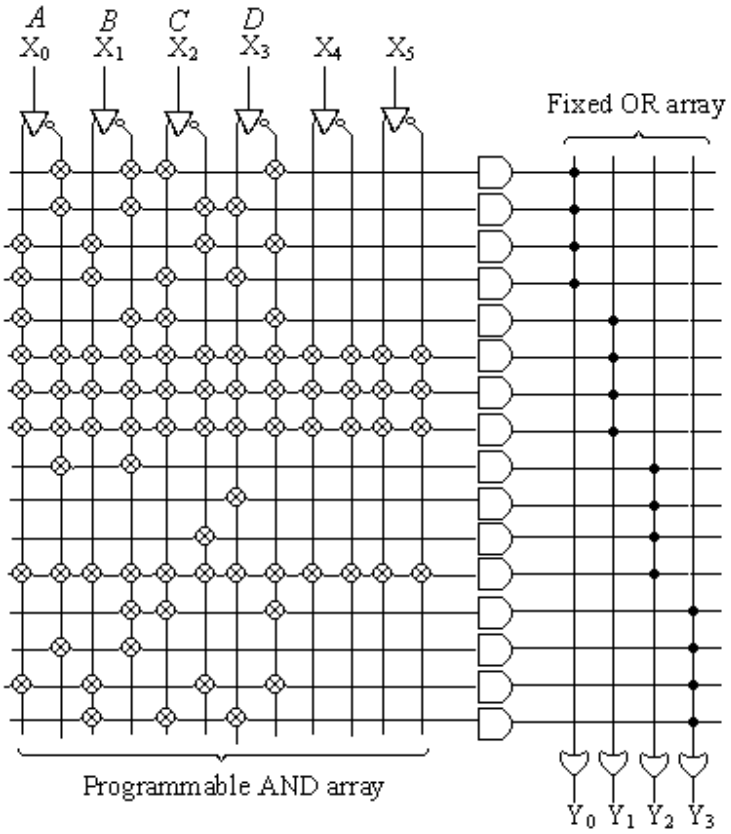


Fig. 6.48

6.9.3 Programmable Read Only Memory (PROM): Another class of PLDs is programmable read only memory (PROM), in which AND array is not programmable while OR array is programmable. So connections from input lines to the AND gates are hard wired, while the connections to the OR gates from the outputs of the AND array are programmable (each joint is marked with circled cross mark ⊗). If there are N input variables, then 2^N product terms are generated. One AND gate will be used for each product term, so there will be 2^N AND gates or rows. The OR array will be of any number. Figure 6.49 shows 16 X 4 PROM. Since $16 = 2^4$, so it will have 4 address or inputs lines and 4 data outputs. For the programming of OR array, circled cross marks are removed or fused for the unused product terms and these marks are retained for the used product terms. PROMs find many applications like the implementation of Boolean functions, code converters and data storage tables.

Example 6.9: Using 16 X 4 PROM, implement the following functions of 4 variables.

$$Y_0(A, B, C, D) = \sum (0,1,4,5,8,9,10,14,15)$$
$$Y_1(A, B, C, D) = \sum (2,3,4,9,10,11,13,15)$$
$$Y_2(A, B, C, D) = \sum (4,5,7,8,10,12,15)$$
$$Y_3(A, B, C, D) = \sum (5,6,7,10,13)$$

Solution: By making the suitable programming of OR array, the given Boolean functions are realized using 16 X 4 PROM as shown in figure 6.50.

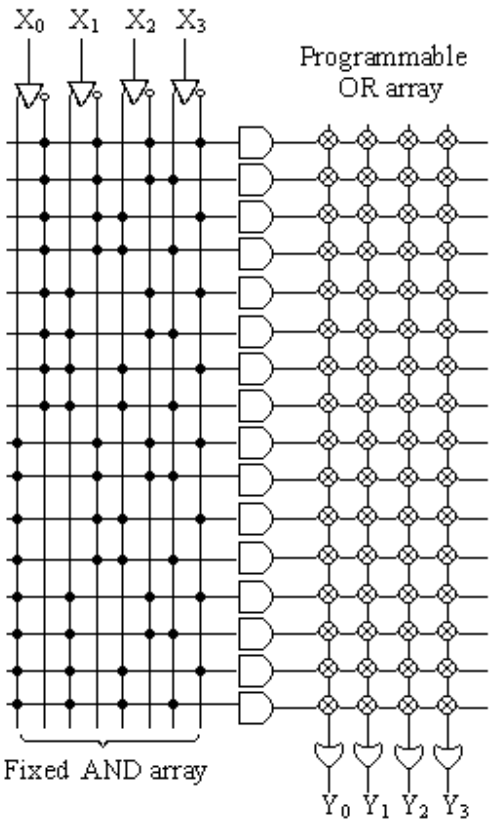


Fig. 6.49

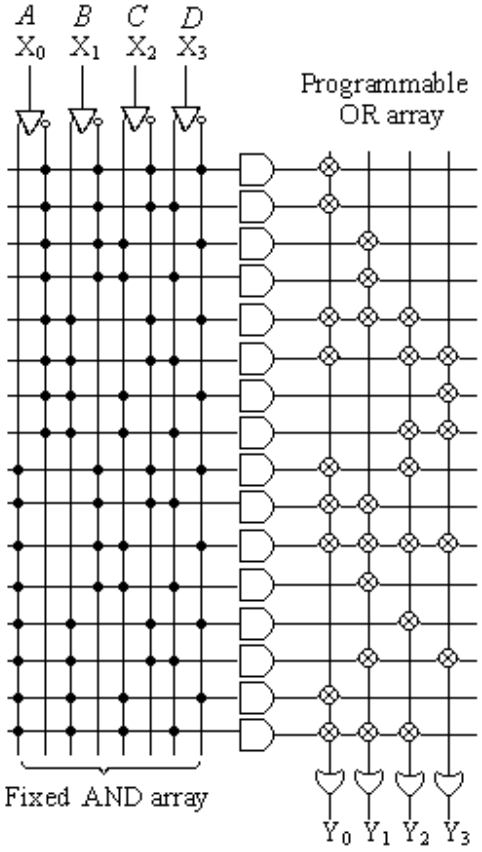


Fig. 6.50

Example 6.10: Using 16 X 4 PROM, implement the 4 –bit binary– to– gray conversion.

Solution: The conversion table of 4 –bit binary to gray is shown in table 6.15. The implementation of this converter is, therefore, is shown in figure 6.51.

The expressions for the outputs of gray code are given by:

$$Y_0(A, B, C, D) = \sum (8,9,10,11,12,13,14,15)$$

$$Y_1(A, B, C, D) = \sum (4,5,6,7,8,9,10,11)$$

$$Y_2(A, B, C, D) = \sum (2,3,4,5,10,11,12,13)$$

$$Y_3(A, B, C, D) = \sum (1,2,5,6,9,10,13,14)$$

Table 6.15

Binary Inputs				Gray outputs			
A	B	C	D	Y_0	Y_1	Y_2	Y_3
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

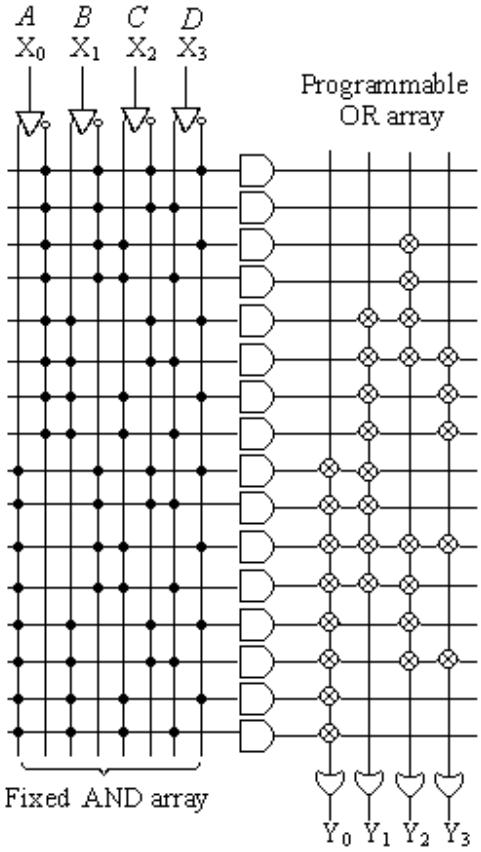


Fig. 6.51

PROBLEMS

1. Explain the working of a multiplexer. What are its uses?
2. Discuss the method of implementing the Boolean expressions using Multiplexers.
3. Implement a full subtractor circuit using MUXs.
4. How can two 8:1MUXs be cascaded to use it a 16:1 MUX?
5. What is a Demultiplexer? Draw the logic circuit of 1: 4 demultiplexer and discuss it working.
6. What is a decoder? Discuss 3 to 8 line decoder having an enable terminal (active high). Also show that a decoder and Demultiplexer are same.
7. Implement SUM and Carry of a full adder with 3 to 8 line decoder and two OR gates.
8. What is BCD to decimal decoder? Draw its logic diagram and explain its working.
9. How can two 3 to 8 line decoder be used as a 4 to 16 line decoder?

10. Design a decoder that displays 4 bit BCD input to seven segment form. Realize the circuit using
    (i) 4 : 1 MUXs
    (ii) NAND gates alone
    (iii) NOR gates alone
11. Repeat the problem 10 if the inputs are in 4 bit excess – 3 code.
12. What is a code converter? Design an 8421 to 2421 code converter. Draw its logic diagram using
    (i) NAND gates
    (ii) NOR gates
    (iii) MUXs
    (iv) 4 X 16 PROM
13. What is an encoder? Draw the logic diagram of octal to binary encoder.
14. Draw and explain the logic diagram of Decimal to BCD encoder.
15. What is priority encoder? Draw the logic diagram of decimal to BCD priority encoder.
16. Discuss octal to Binary priority encoder.
17. What is magnitude comparator? Draw the logic diagram of 4 – bit magnitude comparator and explain its working.
18. How two 4 – bit magnitude comparators be used as a 8 –bit comparator.
19. What are programmable logic devices? Name popularly known PLDs. Explain any one of them in detail.
20. Discuss 4 – bit parity bit generator cum checker.
21. Realize the following function of four variables using 8:1 MUXs.
    (i) $F_1(A, B, C, D) = \sum (0,2,4,6,7,13,15)$
    (ii) $F_2(A, B, C, D) = \sum (0,1,3,4,5,8,9,10,14,15)$
    (iii) $F_3(A, B, C, D) = \sum (0,4,6,7,8,9,10,12,13,14,15)$
    (iv) $F_4(A, B, C, D) = \sum (0,1,2,3,5,8,9,12,13,14,15)$
22. Repeat the problem 21 using 4 to 16 line decoder and 4 OR gates.
23. What is Field Programmable Logic Array (FPLA)? Explain how the programming of AND and OR arrays in FPLA is done.
24. What are Programmable array logic (PAL) devices? What is the difference between FPLA and PAL devices?
25. Explain Programmable Read only Memory (PROM). How does the architecture of a FPLA differ from those of PROM and PAL?
26. Implement a excess – 3 to seven segment decoder using FLA of proper specification.

27. Using the PAL shown in figure 6.47, implement the following SOP functions of 4 variables.

$$X_0 = \overline{A} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot D + A \cdot B \cdot \overline{C} \cdot D + A \cdot C \cdot \overline{D}$$

$$X_1 = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$$

$$X_2 = \overline{A} + C \cdot \overline{D} + A \cdot \overline{C}$$

$$X_3 = \overline{B} \cdot C + \overline{A} \cdot \overline{B} + A \cdot \overline{D} + B \cdot D$$

28. Using 16 X 4 PROM, implement the 4 –bit binary– to– Excess 3 conversion

_____

7

Logic Families

In the last two chapters, discussions on the design of combinational circuits have thoroughly been made. The combinational logic circuits were implemented with the use different logic gates knowing only the characteristics of these gates. However, the electronic hardware of these gates has not so far been discussed. The discussion in this chapter will, therefore, confine to the hardware of different logic families with their operational characteristics and their relative advantages and disadvantages.

7.1 AND Gate: Consider the circuit for two input positive logic AND gate as shown in figure 7.1. The positive logic means that logic 1 is assumed to higher voltage and logic 0 is assumed to lower voltage. Similarly, if logic 1 is assumed for lower voltage and logic 0 is assumed for higher voltage then it is referred to as negative logic. It consists of two diodes D_1 and D_2, the anodes of which are connected to positive supply through a resistance R. The output is taken across the load resistance R_L. The operation of this circuit may be explained as given below:

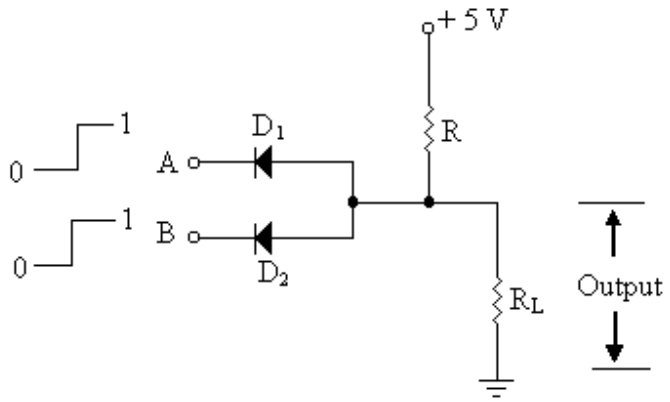


Fig. 7.1

(i) When both Inputs are at logic 0: When both the two inputs A and B are connected to logic 0 (grounded), then both the diodes will be in forward bias. Since the anodes of these diodes are connected to positive supply and cathodes are grounded. The voltage across the load resistance R_L will, therefore, be equal to the forward voltage drop of the

diode. If the diodes are silicon diode, then in no case this voltage will be more than 0.7 volt, which is assumed to be logic o.

(ii) When either of two inputs is at logic 1: In this case, the diode whose cathode is at logic 1 (+5 Volts) will be in the reverse bias and other diode (whose cathode is grounded) will be in forward bias. The voltage across the load resistance R_L will, therefore, be equal to the forward voltage drop of the diode. So the output is at logic 0.

(iii) When both the inputs are at logic 1: In this case both the diodes will be in reverse bias. So the output voltage will be equal to logic 1, as no current will flow through the diode and whole of the current will flow through the load resistance. So the voltage across the load resistance will be approximately +5 volts (logic 1).

It verifies the operation of AND gate. Similarly one can explain the operation of more than two variables AND gate.

7.2 OR Gate: The operation of positive logic two input OR gate may be explained by considering the circuit shown in figure 7.2.

(i) When both Inputs are at logic 0: When both the two inputs A and B of the OR gate are connected to ground (logic 0), the output will be zero (logic 0), since positive terminal of the supply is isolated from rest of the circuit.

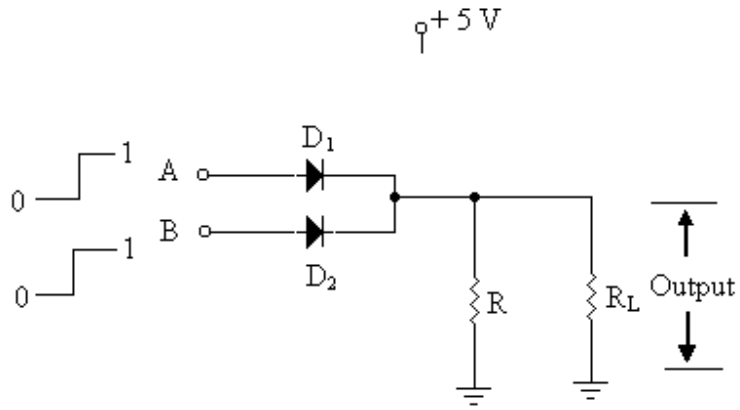


Fig. 7.2

(ii) When either of two inputs is at logic 1: In this case, the anode of the diode whose input is at logic 1 gets connected to the positive terminal of the supply. That diode will be in forward bias, giving the voltage drop of 0.7 volt across the diode. The total voltage across the load resistance R_L will, therefore, be approximately 4.3 volts (logic 1).

(iii) When both the inputs are at logic 1: In this case when both the inputs are at logic 1 i.e. when the anodes of both the diodes are at positive terminal of the supply, both the diodes will be in forward bias. The voltage drop across the load resistance RL will be equal to 4.3 volts (logic 1).

It verifies the operation of OR gate. The operation of more than two variables OR gate may be explained in the same fashion.

7.3 NOT (Inverter) gate: Figure 7.3 shows the circuit diagram of an inverter (NOT) gate. It consists of a transistor in common emitter configuration. It is a unary gate since input to this gate is only one. The principle of operation may be explained as:

When the input A is at logic 0, the emitter base junction of the transistor will be in reverse bias and therefore the transistor goes into cutoff. The collector (output) voltage will be nearly equal to $+V_{CC}$ (logic 1). If on the other hand the input is at logic 1 ($+ V_{CC}$), the transistor will go in to saturation. The value of resistance R is so chosen so that it ensures the emitter base voltage to be equal to $V_{BE,Sat} \approx 0.8$ volts. The collector (output) voltage will be equal to $V_{CE,Sat} \approx 0.2$ volts (logic 0).

So when input is logic 0, output is logic1; if input is logic 1, output is logic 0. This shows the inverter operation.

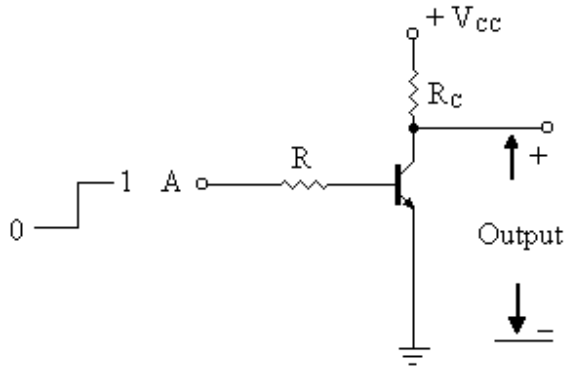


Fig. 7.3

7.4 Logic Families: The basic logic gates discussed above were designed using discrete components like diodes, transistors and resistances etc. In the recent past, it has been possible to fabricate many hundreds of thousands of active and passive components on a small silicon chip. Such fabricated devices are known as integrated circuits (ICs). The Integrated circuits are broadly classified in two categories namely Linear or analog ICs and digital ICs. The analog ICs mainly contain amplifiers, operational amplifiers, audio and power amplifiers etc. However, the digital ICs contain logic gates etc. The variety of logic gates are fabricated in digital ICs using various technologies. The digital ICs may further be classified into following categories depending upon their level of integration:

(i) Small Scale Integrated Circuits (SSI): Twelve gates per IC are fabricated in SSI and total number of components per chip is less than 100.

(ii) Medium Scale Integrated Circuits (MSI): These ICs contain 12 to 100 gates per IC and total number of components per IC is 100 to 1000.

(iii) Large Scale Integrated Circuits (LSI): The large scale integrated circuits contain 100 to 1000 gates per IC and number of components is 1000 to 10000 per IC.

(iv) Very Large Scale Integrated Circuits (LSI): These ICs contain more than 1000 and less than 10000 gates per IC and total number of components per chip is 10000 to 100000.

(v) Ultra Large Scale Integrated Circuits (LSI): More than 10000 gates per IC are fabricated and total components are more than 100000 per chip.

The logic families are classified into two categories depending upon the technologies used for fabrication.

1. Bipolar Logic Families
2. Uni-polar Logic Families

The bipolar logic families are mainly of two types.

a. Saturated Logic Circuits: In which the transistors are driven into saturation.

b. Non-Saturated Logic: In non-saturated transistor logic circuits, the transistors are avoided to go into saturation.

The Saturated logic circuits may further be classified into the following categories:

1. Resistor – Transistor Logic (RTL)
2. Direct Coupled Transistor Logic (DCTL)
3. Integrated Injection Logic (IIL or I^2L)
4. Diode – Transistor Logic (DTL)
5. High Threshold Logic (HTL)
6. Transistor – Transistor Logic (TTL)

The non-saturated logic families are:

1. Schottky Transistor – Transistor Logic (STTL)
2. Emitter Coupled Logic (ECL)

The Uni-polar logic families contains MOS FETs, these are:

1. NMOS or PMOS Logic
2. CMOS (Complementary MOS) logic

Before discussing the details of logic families mentioned above, it is necessary to explain the following characteristics related to them. These parameters will help in comparing the performances of the logic families.

(i) Fan – in: The maximum number of inputs that can be applied to a logic gate is known as Fan – in. Thus a three input AND has fan – in as three.

(ii) Fan – out: The fan –out of logic gate is the number of gates that can be driven by it. Thus, if a fan-out of a typical gate is 10, then it implies that this gate can drive 10 such gates.

(iii) Propagation Delay Time: The propagation delay time of a gate is defined as the time interval between the application of the inputs to a gate and appearance of the signal at the output of the gate. In other words it is defined as the time interval between a

change in input state and the resulting change in output state of the gate. This delay is a very small quantity; it is of the order of few nano second say 20 nsec ($20x10^{-9}$ sec) or 50 nsec ($50x10^{-9}$ sec). The propagation delay of the gate also specifies the speed of the logic gate. The delay time is measured between 50% voltage levels of input and output waveforms. Figure 7.4 shows the input and output waveforms of an inverter. If t_{PHL} is the delay time when the output goes from low state (logic 0) to high state (logic 1) and t_{PLH} is the delay time when the output goes from high state (logic 1) to low state (logic 0), the propagation delay time of the gate t_{pd} expressed as the average of the two delays as:

$$t_{pd} = \frac{t_{PHL} + t_{PLH}}{2}$$

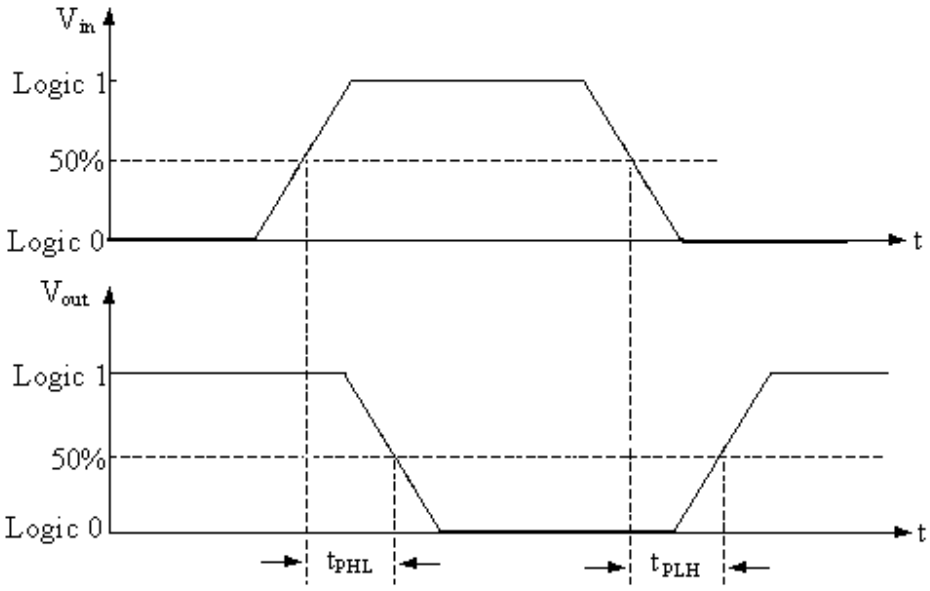


Fig. 7.4

(iv) Power Dissipation: It is defined as the amount of power that can be dissipated in an IC. It is calculated as the product of the d.c. voltage applied to an IC and the current drawn from the d.c. source. It is always desirable to have low power dissipation per gate. The normal working power per gate is required from few micro-watts to few milli-watts. The product of speed and power dissipation per gate is known as the figure of merit of the logic family. A low value of this product is desirable.

(v) Operating Temperature: The temperature range in which an IC functions properly is known as the operating temperature of the gate. It is specified by the manufacturer. The acceptable temperature range of the ICs is from 0 to +70 ^{0}C for commercial applications and this range is from – 55 ^{0}C to 125 ^{0}C for military purposes.

(vi) Noise Margin: Spurious signals called noise are sometimes generated in the connecting leads of the logic circuits due to the stray electric and magnetic fields in the surroundings. This results the unpredictable operation of the logic circuit. The noise margin is sometimes called Noise- immunity. It is defined as the difference between the maximum permitted low input and the maximum guaranteed low output, and that

between the minimum permitted high input and the minimum guaranteed high output. The idea of noise margin is illustrated in figure 7.5.

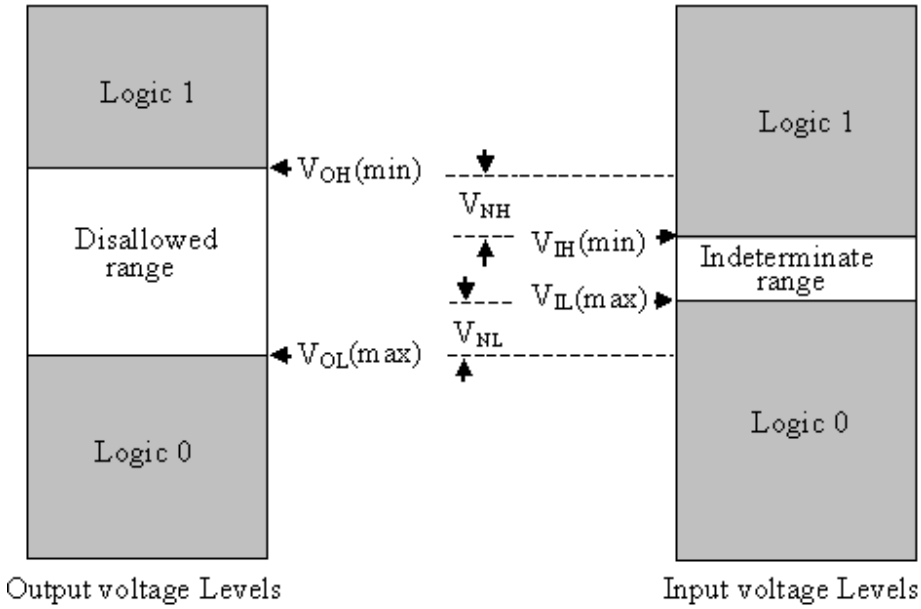


Fig. 7.5

Figure 7.5 shows that $V_{OH}(min)$ is the minimum high voltage for logic 1 and $V_{OL}(max)$ is the maximum low voltage for logic 0. The output should not occur in the disallowed range. Similarly, $V_{IH}(min)$ is the minimum high input voltage and $V_{IL}(max)$ is the maximum low input voltage and the voltage level between $V_{IH}(min)$ and $V_{IL}(max)$ is the indeterminate range and this voltage range should not be applied to the inputs of the logic gate.

As per definition of the noise margin, the noise margin for high state (V_{NH}) and the noise margin for low state (V_{NL}) are given by:

$$V_{NH} = V_{OH}(min) - V_{IH}(min)$$

$$V_{NL} = V_{OL}(max) - V_{IL}(max)$$

The large noise margin is always desirable.

7.5 Resistor – Transistor Logic (RTL): The resistor–transistor logic is the most common family of logic circuits. It consists of resistors and transistors hence known as resistor transistor logic. Figure 7.6 shows the basic circuit for two - input RTL NOR gate. The operation of this circuit may be explained as follows:

When both the inputs A and B are at logic 0, the two transistors T_1 and T_2 will be in cutoff and no current flows through collector emitter circuit of the transistors. The output will, therefore, be high (logic 1). When either of the two inputs is at logic 1, the corresponding transistor will go into saturation and output will be $V_{CE,Sat}$ of the transistor (≈ 0.2 V). The output is said to be at logic 0. The output will also be low, if both the inputs are at logic 0, as both the transistors will saturate. It is concluded that it performs the operation of the NOR gate.

Though this is a simplest logic circuit yet it has become obsolete. RTL has the advantage that its power dissipation per gate is low. The disadvantages of this family are that it has low noise margin and its propagation delay is relatively larger.

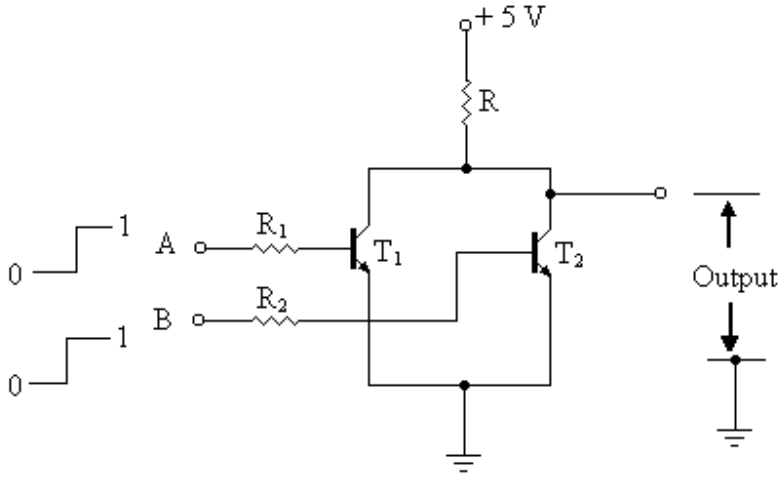


Fig. 7.6

7.6 Direct Coupled Transistor Logic (DCTL): The direct coupled transistor logic circuit is similar to RTL, which obtained by omitting the base resistances in RTL. The DCTL circuit for two-input NOR gate is shown in figure 7.7. When one or both the inputs are high (logic1), the corresponding transistor or transistors will be conducting and the current flows through the resistance R gives the output low (logic 0). It, however, corresponds to high output voltage when both the inputs are at low. This logic is very simple and requires a few components but it has the disadvantage of low noise margin.

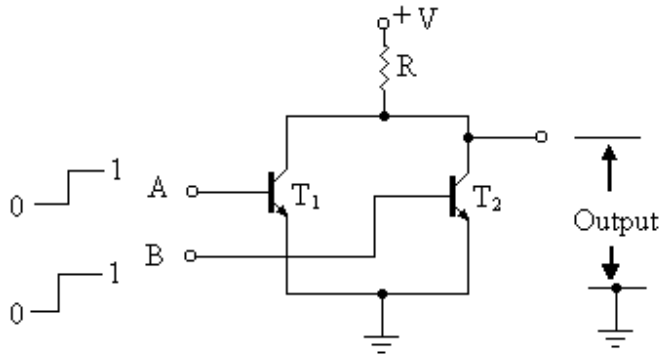


Fig. 7.7

7.7 Integrated Injection Logic (IIL or I^2L): This family of bipolar transistors is the simplest logic family and it has high packing density due to which a large number of digital functions can be formed on a single chip. The I^2L family is available in LSI package for complex digital functions such as microprocessor etc and thus individual gates in SSI package are not available.

Figure 7.8 shows the logic diagram of three - input I^2L NOR gate. The basic unit of this circuit is an inverter which is shown in the shaded box. The PNP transistor T_1 serves as a constant current source that injects the current into the base of the transistor

T_4. If the input is at logic 0 (grounded), the injected current becomes grounded thus diverting the current from the base of transistor T_4. This transistor, therefore, goes into cutoff and the output is high. If on the other hand when the input A of the inverter is high, the injected current from the current source flows into the base of the transistor T_4 thus turning it ON. The output is low. The circuit for three - input NOR gate is the combination of three inverters and its operation may be explained in the similar fashion. It has a low power requirement and reasonably good switching speeds.

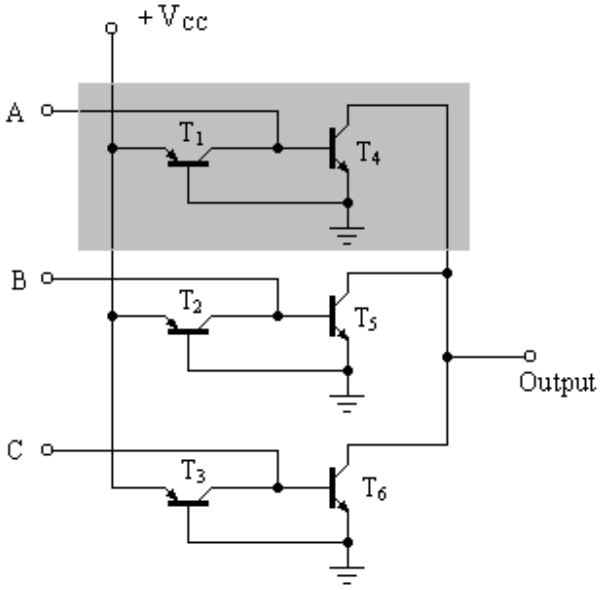


Fig. 7.8

7.8 Diode – Transistor Logic (DTL): The next family after RTL was diode transistor logic (DTL), which has high noise margin though slow speed. In DTL diodes and transistor are used hence the name diode transistor logic. Figure 7.9 shows the positive logic two input DTL NAND gate. Its operation may be explained as given below:

When both the inputs are at logic 0, the diodes D_1 and D_2 will be in forward bias and the voltage at the point P will be equal to the forward voltage drop of the diode ≈ 0.7 V). This voltage is being applied to the base of the transistor T_1 through the diode D_3, due to which the transistor T_1 goes in to cutoff. The output voltage will, therefore, be high (logic1). The diode D_3 ensures that the transistor T1 is in cutoff. In the absence of this diode the transistor could be in active region and output would not be high enough. When either of the two inputs is at logic 1, the corresponding diode will be in reverse bias and the other diode will be in forward bias due to which the voltage at point P will be equal to the forward voltage of the diode. This takes the transistor T_1 into cutoff, giving the output voltage to be high (logic 1). Now when both the inputs are connected to logic 1, both the diodes D_1 and D_2 will be in reverse bias and the voltage at the point P is high due to which the transistor T_1 goes into saturation. The output will be $V_{CE,Sat}$ of the transistor (≈ 0.2 V). The output is said to be at logic 0. The function of resistance R_2 connected between the base of the transistor T_1 and ground is to remove the stored base charge when the transistor has to be turned off from the saturated state. The lesser the value of

this resistance lesser will be the propagation delay time of the gate, but the value of this resistance can not be decreased beyond certain value, otherwise the transistor T_1 will never be in saturation.

The propagation delay of this logic is high approximately 50 nsec.

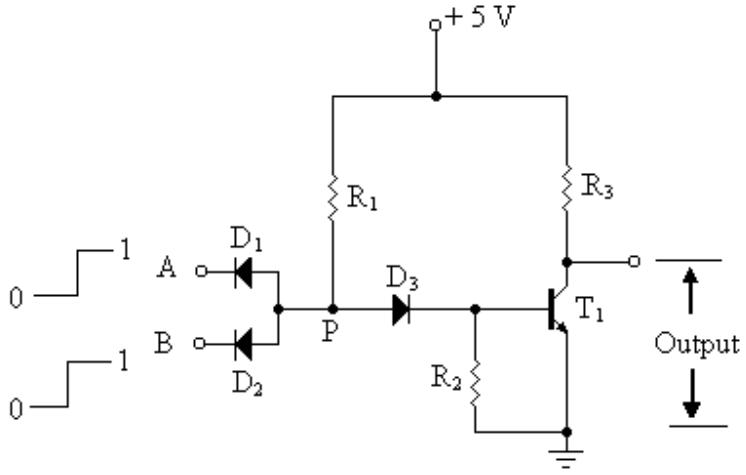


Fig. 7.9

7.9 High – Threshold Logic (HTL): A high threshold logic gate is a modification of a DTL gate. It is designed for industrial applications by providing large noise margin. Figure 7.10 shows the logic diagram of two – input HTL NAND gate. This logic circuit has been designed for higher supply voltage (15 V). It utilizes a zener diode of breakdown voltage of 6.9 V.

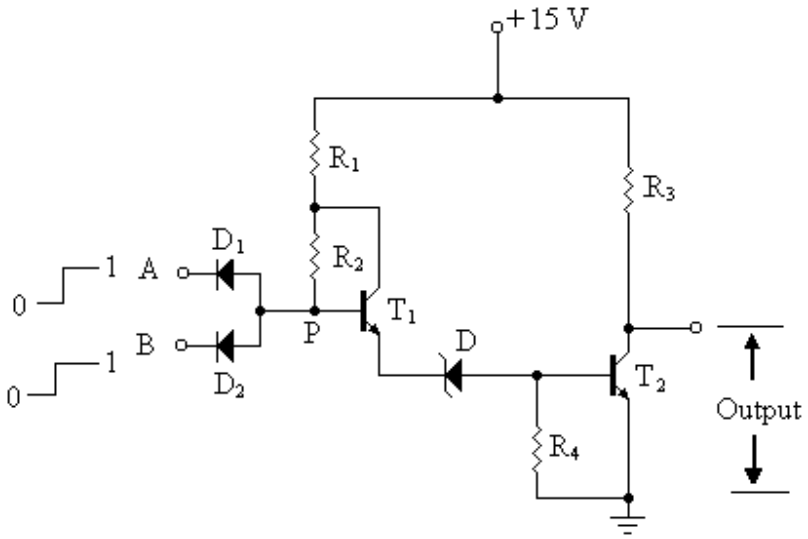


Fig. 7.10

The transistor T_2 will conduct when the emitter of transistor T_1 is at 7.5 V, as the sum of 6.9 V zener voltage and V_{BE} of T_2 (0.6 V). The low output level of the HTL gate will be 0.2 V and high level will be about 15 V. when one of the inputs or both the inputs are at low transistor T_2 is off. When both the inputs are high transistor T_2 saturates.

The advantage of this gate is that its noise margin is high however it is slow in speed.

7.10 Transistor – Transistor Logic (TTL): The TTL is the most popular amongst all logic families and is widely used IC technology.. It is the modified form of DTL. The propagation delay time is reduced in TTL by using multi-emitter transistor in place of diodes. Figure 7.11 (a) shows the schematic diagram of a basic TTL positive logic NAND gate. It consists of a multi-emitter transistor T_1. A two emitter transistor is equivalent to two transistors with common base and common collector as shown in figure 7.11 (b).

The operation of TTL NAND gate may be explained as follows:

When either of two inputs A or both the inputs are at logic 0, emitter base junction of the multi-emitter transistor will be in forward bias and base current is supplied by the resistor R_1. The transistor T_1 saturates and the voltage at the point will be equal to $V_{CE,Sat}$ of the transistor (≈ 0.2 V). The transistor T_2 will be in cutoff and output voltage will be high (logic 1).

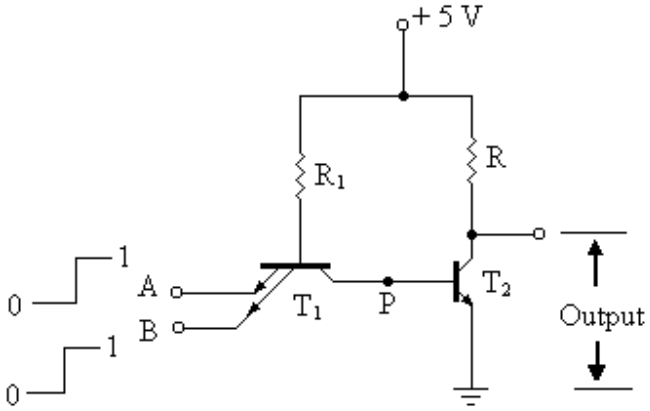


Fig. 7.11 (a)

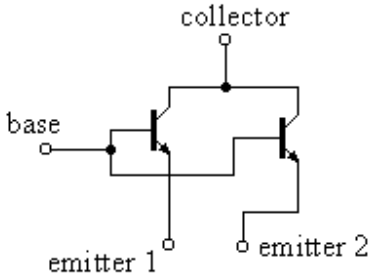


Fig. 7.11 (b)

When both the inputs are at logic 1 (+5 V), the emitter base junctions of transistor T_1 will be reverse biased and current will flow, through R_1 and through the forward biased base collector junction of transistor T_1 into the base of transistor T_2. In this mode the transistor is said to be operated in the inverted mode, as the collector of transistor T_1 operates as emitter and the emitter as collector. The voltage at the point P will be sufficient to drive the transistor T_2 into saturation, the output voltage will therefore, be equal to $V_{CE,Sat}$ (≈ 0.2 V) or logic 0.

The propagation delay time of this gate is smaller than that of DTL NAND gate, since when the transistor T_2 goes into cutoff region from saturation region, the transistor

T_1 saturates and provides a low impedance path to ground. Thus the stored base charge of the transistor T_2 is quickly removed thereby reducing the propagation delay time.

The output resistance of the basic TTL circuit (fig. 7.11 a) is low when the transistor T_2 saturates or output is low (logic 0). However, the output resistance of this circuit is almost equal to the resistance R, when the transistor is in cutoff or output is high (logic 1). This will restrict the fan out of the gate. The reduction in resistor R would increase the power dissipation in R and in the gate. Also the reduction in the value of R would difficult to saturate the transistor T_2. To overcome this difficulty, TTL gate with totem pole arrangement is used.

7.10.1 TTL NAND Gate with Totem-pole Output: Figure 7.12 shows the standard form of a TTL circuit with input NAND gate. The circuit works as follows:

When either the inputs or both the inputs are low (logic 0), the transistor T_2 goes into cutoff. The transistor T_4 will also be in cutoff, as the voltage drop across the resistor R_3 is nearly zero. Now the transistor T_3 conducts and works as emitter follower. The output voltage available at the emitter of this transistor will be equal to the collector voltage of the transistor T_2, which is high (logic1). The emitter follower, however, provides a low output resistance to the input of the driven gate.

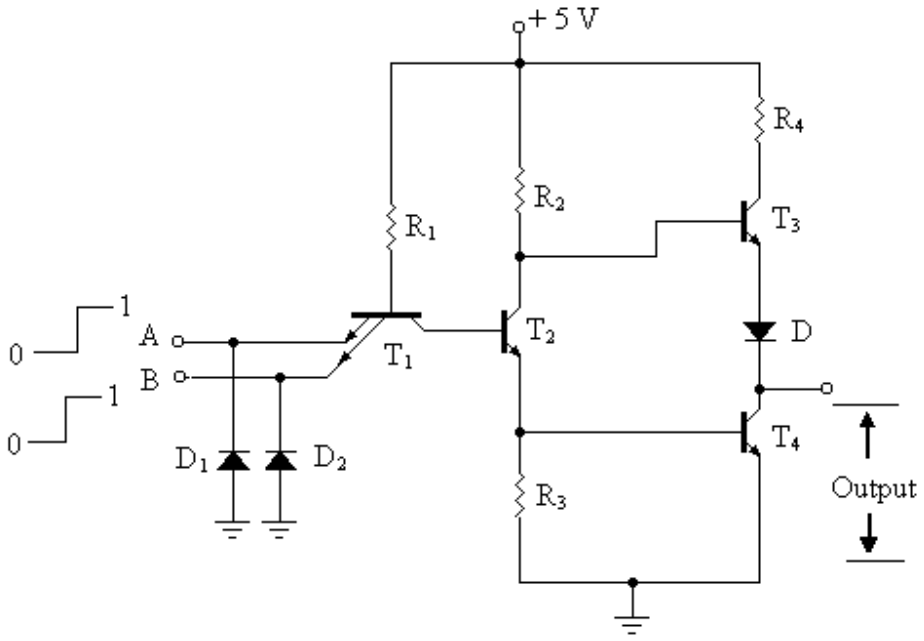


Fig. 7.12

When both the inputs are high (or at logic 1), transistor T_2 conducts and acts as an emitter follower. The potential across R_3 will be sufficient to drive the transistor T_4 into saturation. Because the transistor T_4 saturates, the output voltage will, therefore, be equal to $V_{CE,Sat}$ (≈ 0.2 V) or logic 0. Since this output is taken at the collector of the transistor T_4, which is in saturation, so it provides the low output impedance. The diode D prevents the transistor T_3 from being conducting when the transistor T4 saturates. The potential

across the emitter base junction of the transistor T_4 is approximately 0.8 V ($V_{BE,Sat}$) and collector emitter voltage of T_2 is 0.2 V ($V_{CE,Sat}$). This means a total of 1.0 V is applied to the base of transistor T_3. In the absence of the diode D, this voltage would be sufficient for the conduction of the transistor T_3. The diode D, however, reduces the base emitter voltage of transistor T_3 below 0.7 V, required voltage for the conduction of a transistor. Thus the diode D drives the transistor T_3 into cutoff when T_4 saturates.

Diodes D_1 and D_2 protect the transistor T_1 from being damaged when the negative spikes of the voltage appears at the inputs. When the negative spikes appear at the input terminals the diodes conducts and the spikes are grounded. The transistors T_3 and T_4 and the diode D form the totem pole output, which provides the low output impedance in every case. The TTL gates are faster having the propagation delay of about 15 nsec.

7.10.2 TTL Inverter: Figure 7.13 shows a TTL circuit for an inverter. The operation principle of this is same as discussed for TTL NAND gate, with the difference that it has only one input. So when input A is at logic 0, output will be high (logic 1) and if input is high (logic 1), output will be low (logic 0). This circuit also has the totem-pole output.

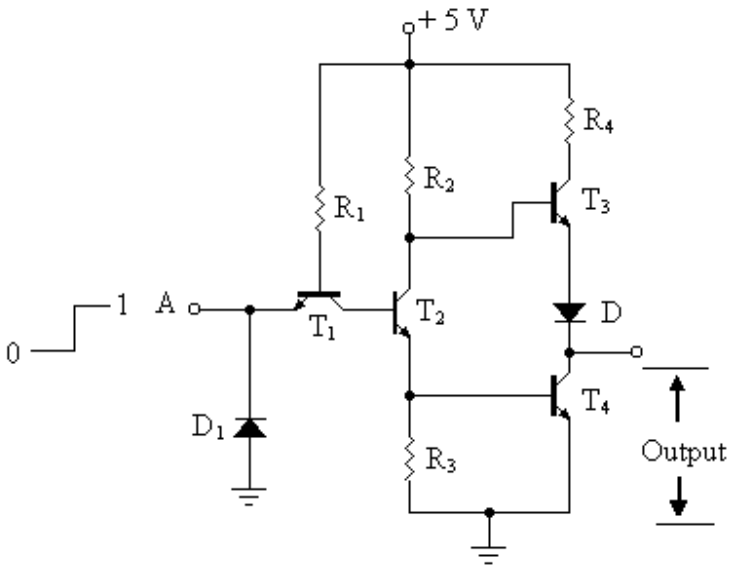


Fig. 7.13

7.10.3 TTL NOR Gate: Figure 7.14 shows a TTL circuit for two input NOR gate. It consists of two input transistors T_1 and T_2 and two other transistors T_3 and T_4 connected in parallel which acts as a phase splitter. In addition the output is obtained using the totem pole circuit comprising transistors T_3, T_4 and diode D. The operation of this circuit may be explained as follows:

When both the inputs are low, the emitter base junctions of the input transistors will be in forward bias, no current will flow through the base of transistors T_3 and T_4. So

these transistors will be in cutoff. The transistor T_5 will, therefore, conduct and T_6 will be in cutoff, producing a high (logic 1) output.

When input A is low and input B is high, the transistor T_3 is in cutoff and transistor T_4 saturates. The transistor T_6 will, therefore, conduct and T_5 will be in cutoff, producing a low (logic 0) output.

Similarly, when input A is high and input B is low, the transistor T_4 saturates and transistor T_3 goes in cutoff. The transistor T_6 will, therefore, conduct and T_5 will be in cutoff, producing a low (logic 0) output.

When both the inputs are high, the emitter base junctions of the input transistors will be in reverse bias, the current will flow through the base of transistors T_3 and T_4. So these transistors will saturate. The transistor T_6 will, therefore, conduct and T_5 will be in cutoff, producing a high (logic 0) output.

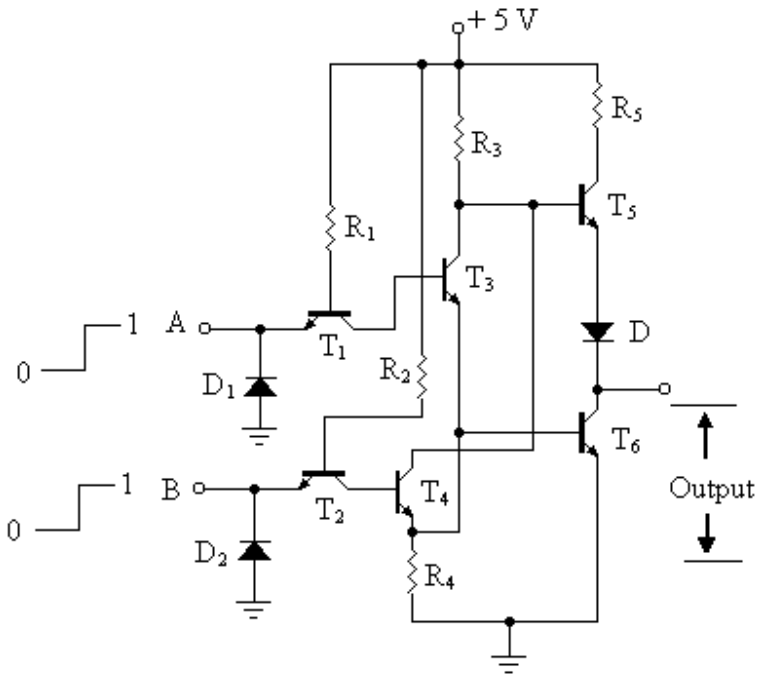


Fig. 7.14

7.10.4 TTL AND Gate : Figure 7.15 shows a TTL two input AND gate. The AND operation is obtained by inserting an extra inversion circuit before the totem output of the TTL NAND gate. This converts the NAND gate to an AND gate. The extra inversion circuit comprises the transistors T_2 and T_3.

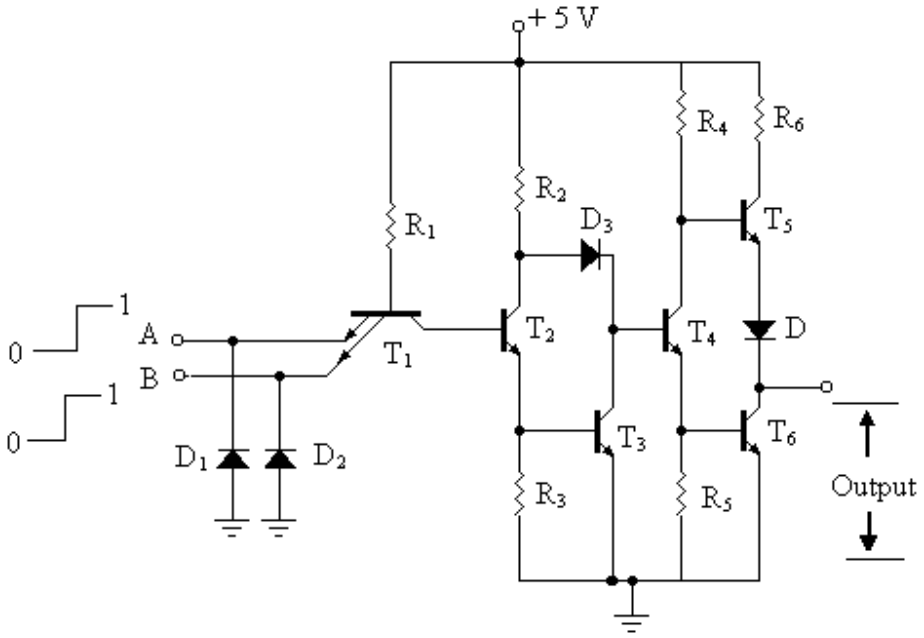


Fig. 7.15

7.10.5 TTL OR Gate : The TTL OR gate is obtained by inserting a common emitter circuit before the totem pole output of the TTL NOR gate as shown in figure 7.16. The common emitter circuit provides an inversion, which converts the NOR gate to an OR gate. The transistor T_5 with associated components forms the common emitter circuit.

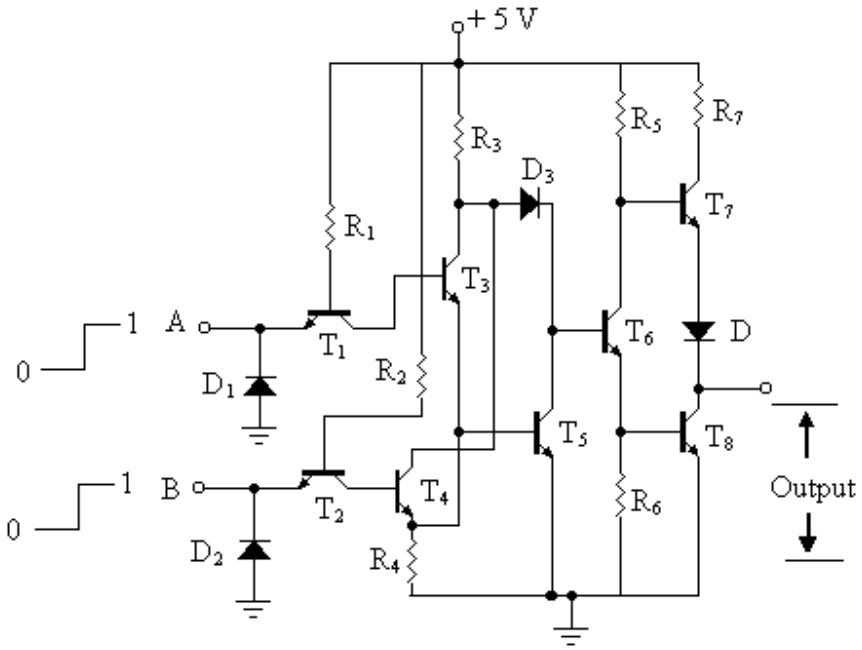


Fig. 7.16

7.10.6 Open Collector TTL Gates: It has been discussed in the previous sections that in all TTL gates totem pole output circuit is connected. The integrated circuits for TTL gates are also available with open collector output. Figure 7.17 (a) shows a two input TTL NAND gate with open collector. The other gates are also available with open collector outputs. In the open collector output gates the lower transistor of the totem pole circuit is used with its collector open or floating. In order to get the proper output, one has to connect an external pull-up resistor between the collector and the positive supply as shown in figure 7.17 (b).

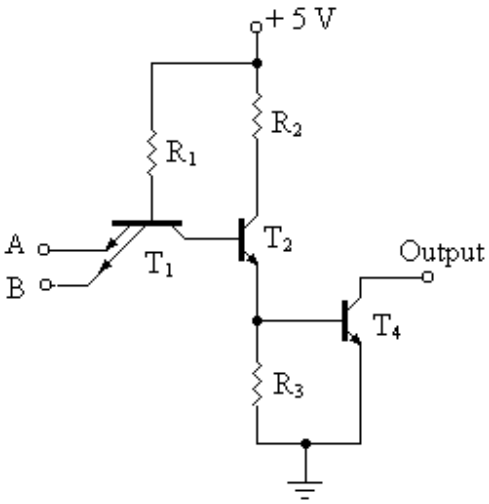


Fig. 7.17 (a)

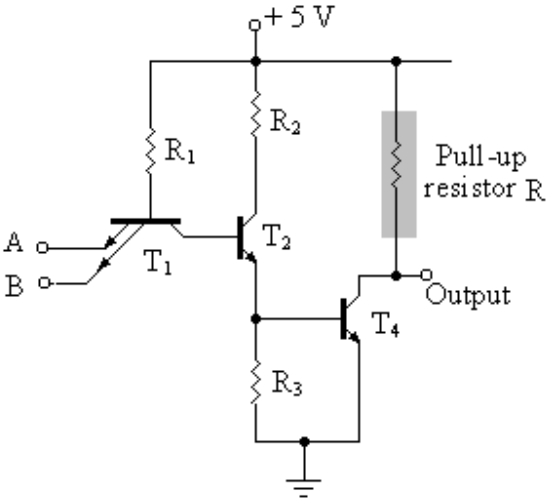


Fig. 7.17 (b)

The advantage of the open collector gates is that their outputs can be wired together and connected to a common pull-up resistor, thus eliminating the need of an AND gate. This can be illustrated by connecting the open collectors of three NAND gates together with a pull-up resistor R as shown in figure 7.18 (a). Its equivalent circuit is given in figure 7.18 (b), in which output of three NAND gates (open collector) are connected together to a pull-up resistor R.

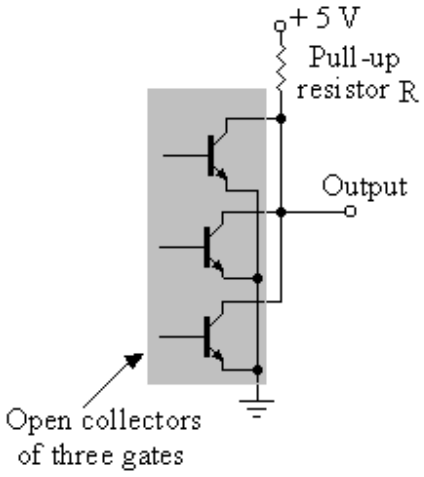


Fig. 7.18 (a)

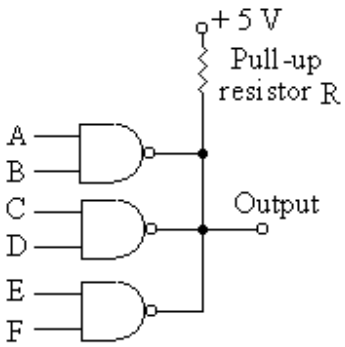


Fig. 7.18 (b)

When any or all transistors are in saturation, the output voltage is pulled down to a low value. On the other hand, if all the transistors are in cutoff, the pull up resistor R pulls the output voltage to a high value. It therefore produces the ANDing of the outputs of three gates. To get the ANDing operation by wiring the outputs of open collector devices to a common pull-up resistor is known as wire –AND. Any number of gates may be ANDed together with this method. The output of circuit shown in figure 7.18 (b) is given by:

$$Output = (\overline{A \cdot B}) \cdot (\overline{C \cdot D}) \cdot \overline{(E \cdot F)}$$

The wire – AND is not possible with the TTL devices having totem pole outputs. If the outputs of two or more such devices are connected together and one output is low and the other high, the final output gets short circuited, resulting thereby too much power dissipation. So for ANDing the outputs of TTL devices, a separate AND gate is needed.

The main disadvantage of open-collector gates is their slow speed.

7.10.7 Tri-state TTL Gates: It has been observed from the above discussion that the open collector gate has the facility for wire – AND, but they are slow in speed. However, the gates with totem pole outputs are faster in speed but the connections for wire –AND are not possible. This led to the development of new device called tri-state TTL gates. The tri-state devices allow three possible output states namely, High, Low and High impedance. The high impedance state offers high impedance between the output terminal and ground or positive supply. Output in this case is floating. A simple tri-state TTL circuit for inverter is shown in figure 7.19 (b) and its logical symbol is given in figure 7.19 (b). In this circuit input A is the normal logic input while the ENABLE E terminal is an enable input that can produce high impedance output.

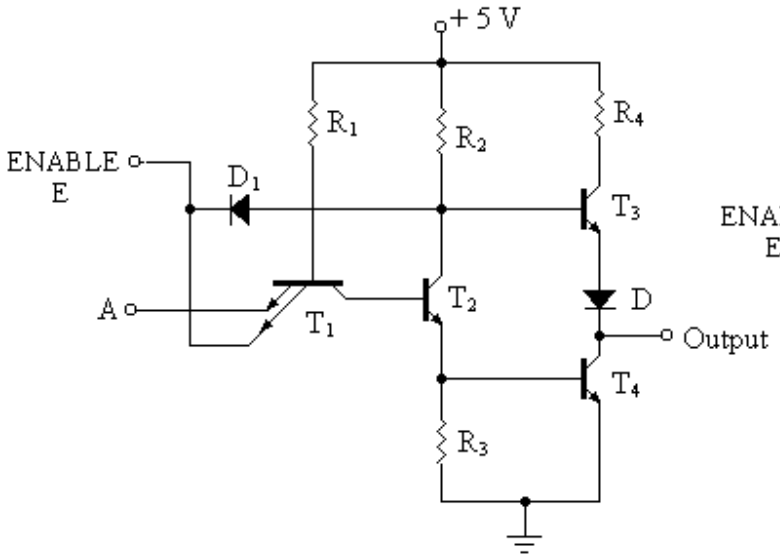


Fig. 7.19 (a)

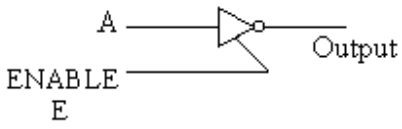


Fig. 7.19 (b)

When ENABLE E terminal is high (logic 1), the diode D_1 remains in reverse bias so it has no effect on the working of transistors T_3 and T_4 and therefore circuit operates as normal inverter. When ENABLE E terminal is low (logic 0), the diode D_1 will be in

forward bias and it takes away the base current of transistor T_3. So this transistor will be turned off. The forward bias diode D_1 also forward biases the emitter base junction of the transistor T_1, transistor T_2 will therefore be turned off, which in turn turns off the transistor T_4. So by applying logic 0 to the ENABLE E terminal both the transistors T_3 and T_4 of totem pole output go in cutoff state.

The tri-state configuration is possible with other gates also with the similar circuits. The advantage of this configuration is that wire –ANDing of the outputs of tri-state ICs is possible and its speed is also fast.

7.10.8 More TTL Circuits: There are three families of TTL circuits, namely:

High Speed TTL circuits

Medium Speed TTL Circuits

Slow Speed TTL Circuits

The circuit of TTL NAND gate has been reproduced in figure 7.20 with three values of each resistor R_1, R_2, R_3 and R_4 for the three families. The low values of these resistances are for high speed but the power dissipation will be larger because low values of resistances will draw large current from the supply. The 54H/74H series for TTL gates are available and designed for high speed. The alphabet H represent for high speed. The typical propagation delay for high speed gate is 6 nsec and power consumption is 22 mW. The medium values of these resistances are for medium speed. The 54/74 series is available for medium speed TTL gates. This is the standard series and the typical propagation delay for this series is 10 nsec and power consumption is 10 mW. For slow speed TTL gates the values of resistances used are high and the series available for slow speed is 54L/74L. The typical propagation delay for slow speed gate is 33 nsec and power consumption is 1 mW. The 54 series the counterpart of 74 series and both are equivalent. The 54 series is used generally for military purposes, as this series can be operated for wider temperature range and voltage ratings.

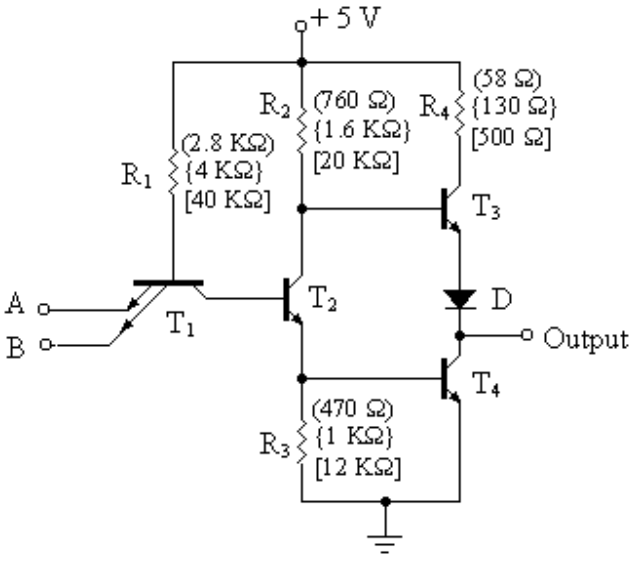


Fig. 7.20

7.11 Schottky Transistor – Transistor Logic (STTL): In Schottky TTL circuits, the operation speed is much more larger than the high speed TTL circuits. The transistors used in TTL circuits take certain time when the transistors switch form saturation to cutoff. This limits the propagation delay of the gates. This delay can however, be reduced by replacing the transistors in TTL circuits by the Schottky transistors. The Schottky transistor is formed by connecting Schottky barrier diode between base and collector of a transistor as shown in figure 7.21 (a). The Schottky barrier diode (SBD) has a forward drop of only 0.25 V, it therefore prevent the transistor from saturating fully. Figure 7.21 (b) shows the circuit diagram of two-input Schottky TTL NAND gate. Notice the transistor T_4 is the ordinary transistor.

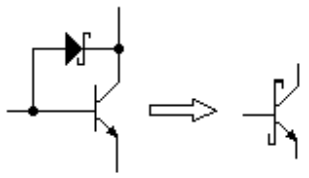


Fig. 7.21 (a)

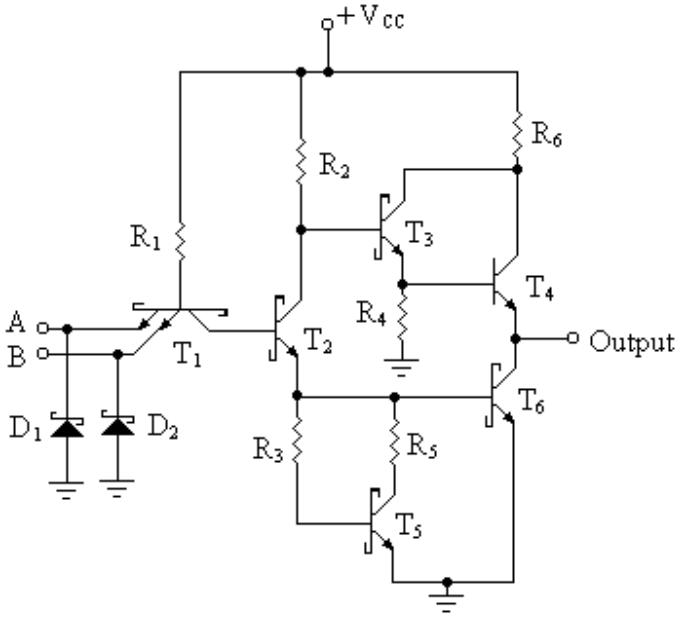


Fig. 7.21 (b)

The 54S/74S series is available for Schottky Transistor – Transistor Logic (STTL) gates. This series of logic family has less power consumption as compared to 54H/74H series and the speed is double to that of 54/74 series. Still low power 54LS/74LS series of Schottky TTL is available. This series is obtained by increasing the resistances used in 54S/74S series. This family of logic gates therefore has the same switching speed as that of standard TTL family (54/74), and the power dissipation is 1/5 of the 54/74 series.

7.12 Emitter Coupled Logic (ECL): Emitter Coupled Logic (ECL) circuits fall in the category of non-saturated digital logic family i.e. the transistors in this family do not saturate. This eliminates the storage time delay, so the speed of operation of this family is increased. This logic family has the fastest speed and propagation delay time per gate is approximately 1 nsec.

Figure 7.22 (a) shows the basic circuit of four-input ECL OR/NOR gate. The outputs provide both OR and NOR functions. The transistors T_1 through T_5 form the differential amplifier circuit, transistor T_6 forms the internal temperature and voltage

compensation bias network and the transistors T_7 and T_8 gives the emitter follower outputs for OR and NOR functions. Logic levels for this family are negative, – 0.9 V is assumed for logic 1 and – 1.75 V for logic 0. The operation of this circuit may be explained as follows:

When all the inputs are at low (– 1.75 V), the transistors T_1 through T_4 are off, as emitter base junctions are reverse biased. The transistor T_5 is conducting not saturated. Due to the proper biasing of the transistor T_6, the base of transistor T_5 remains at – 1.29 V. Therefore its emitter is at – 2.09 V which is 0.8 V below the base voltage. The transistor T5 therefore conducts. The differential voltage between base and emitter of the transistors T_1 through T_4 is about –0.34 V, so they are in cutoff. The emitter follower transistors T_7 and T_8 give the outputs – 1.75 V (logic 0) and – 0.9 V (logic1) respectively.

When any one or all the inputs are at – 0.9 V (logic1), in that condition the corresponding transistor or transistors will conduct. The voltage at the emitters of T_1 through T_5 therefore rises to – 2.09 V. Since the base of transistor T_5 is held constant at – 1.29 V due to the bias network, it goes into cutoff. The emitter follower transistors T_7 and T_8 give the outputs – 0.9 V (logic1) and – 1.75 V (logic 0) respectively. Symbolic representation of OR/NOR ECL gate is shown in figure 7.22 (b).

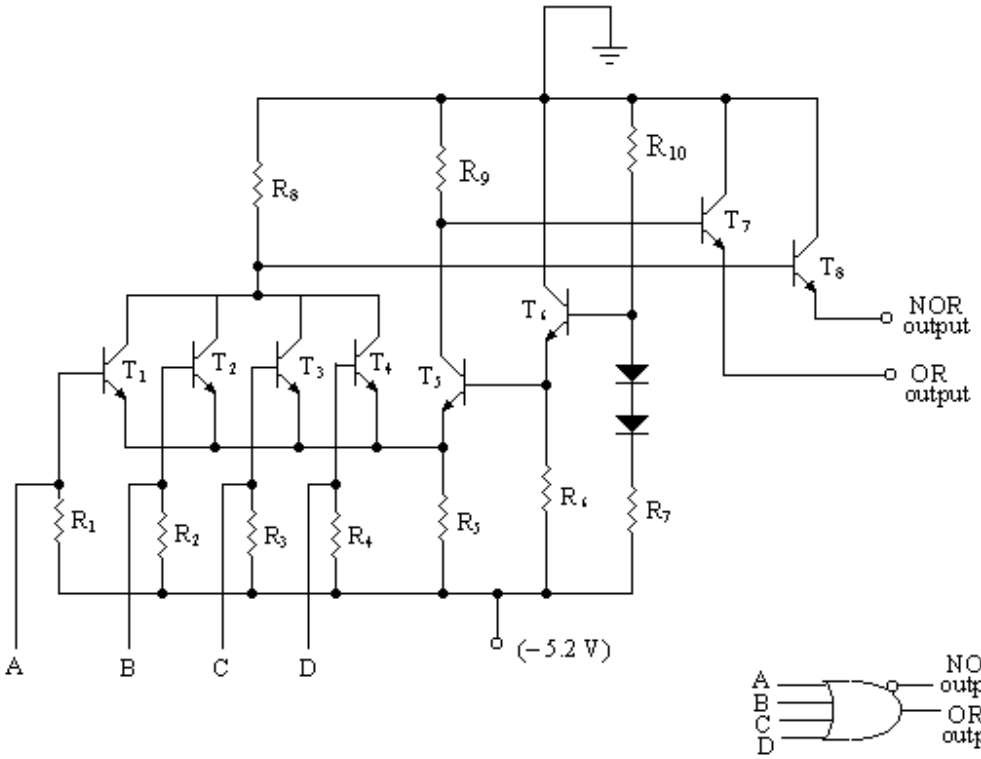


Fig. 7.22 (a)

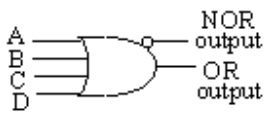


Fig. 7.22 (b)

The wired logic can be formed by connecting together the outputs of two or more ECL gates as shown in figure 7.23. The external -wired connection of two NOR outputs produces a wired –OR function. The internal –wired connection of two OR outputs in some ECL ICs is used to produce a wired –AND logic.

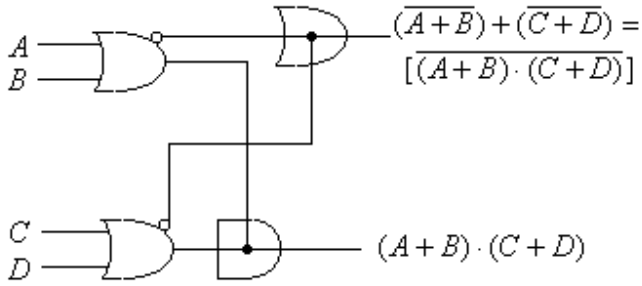


Fig. 7.23

7.13 MOS Logic: The logic families discussed so far were based on bipolar transistor. Their comparisons were made with respect to certain parameters of the logic family. One more logic family based on the unipolar devices such as Metal Oxide Semiconductor field effect transistor (MOS FET) will now be discussed. The MOS logic family is the simplest to fabricate and occupies less space. It requires N channel MOS or P channel MOS field effect transistors and no other components such as resistors, diodes etc. This logic family has the high packing density, low power dissipation and high fan-out.

The logic circuits may be designed using NMOS (enhancement type N channel MOS FET's) or PMOS (enhancement type P channel MOS FET's). From the operations of MOS FET's one can note following characteristics of MOS FET's. The NMOS conducts when gate is at a positive potential with respect to source and PMOS, however, conducts when gate is at a negative potential with respect to source. If the gate is at zero potential neither of the two MOS FET's will conduct.

7.13.1 MOS inverter: Figure 7.24 (a) shows the circuit diagram for NMOS inverter and figure 7.24(b) shows for PMOS inverter. The working operation of the circuits is same. The MOS FET T_1 in both the circuits work as resistor since T_1 is conducting as gate is connected to drain.

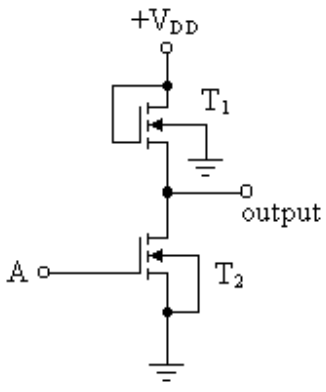


Fig. 7.24(a)

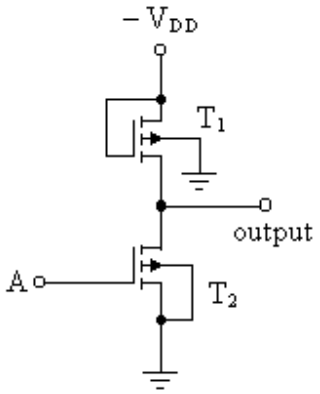


Fig. 7.24(b)

In figure 7.24(a) when input A is at logic 0 (ground potential), the MOS FET T_2 will be OFF giving the high voltage at the output. So the output is at logic 1. If on the other hand input A is at logic 1 (V_{DD} potential), the MOS FET T_2 will be ON and output will be at logic 0. This verifies the operation of inverter. The operation of PMOS will be discussed in the similar fashion with the only difference that it works for negative logic.

7.13.2 MOS NOR gate: Figure 7.25(a) shows the circuit diagram of NMOS positive logic three-input NOR gate and 7.25(b) for PMOS negative logic three-input NOR gate. In NMOS NOR gate (ref. fig. 7.25 a), when all the three inputs are at logic 0 (ground potential), MOS FET's T_2 through T_4 will be off giving the high output (logic 1). If all the three inputs or any (one or two) of the three inputs are at logic 1, the corresponding MOS FET or MOS FETs will conduct giving low output (logic 0). This verifies the operation of positive logic NOR gate. The working operation of PMOS NOR gate may be explained in the similar which works for negative logic. The MOS FET T_1 acts as a resistor in both the circuits.

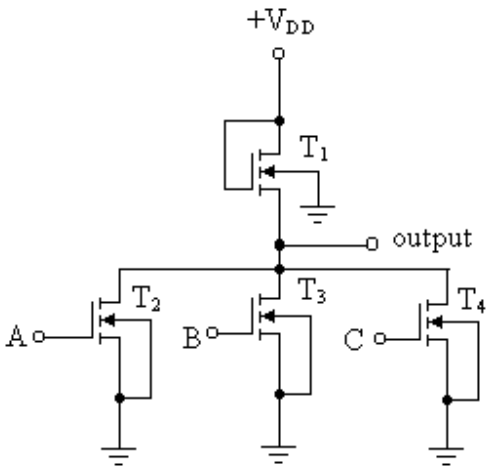


Fig. 7.25(a)

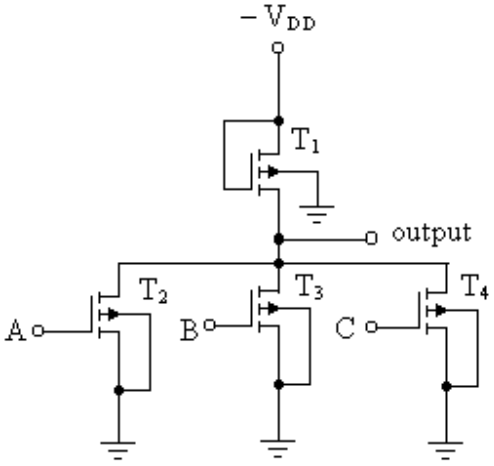


Fig. 7.25(b)

7.13.3 MOS NAND gate: Three input NAND gate with NMOS and PMOS transistors are shown in figure 7.26(a) and 7.26(b) respectively. The NMOS NAND gate works with positive logic and PMOS NAND gate work with negative logic. The working of NMOS NAND gate is explained as follows (ref. 7.26 A).

The NMOS FET's T_2 through T_4 will conduct when all the three inputs are at logic 1 (+ V_{DD}), giving the output low (logic 0). When either of the three inputs or any (one or two) of the inputs is at logic 0 (ground potential), the corresponding MOS FET or MOS FET's will be off giving the high output (logic1). This verifies the operation of positive logic NAND gate,

Similarly, one can explain the operation of PMOS NAND gate which work with negative logic.

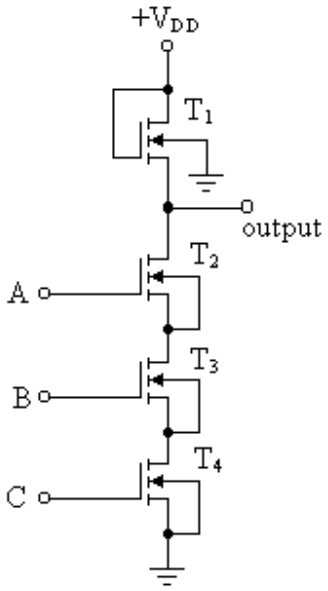


Fig. 7.26(a)

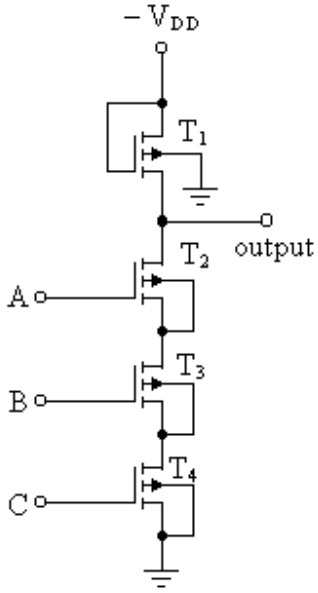


Fig. 7.26(b)

7.14 Complementary MOS (CMOS) Logic: The complementary metal oxide semiconductor (CMOS) logic family contains both enhancement type P-channel and N-channel MOS FET's arranged in a complementary connection. The power consumption of CMOS logic family is very less as neither of P-channel or N-channel MOS FET's conducts simultaneously when no signal is applied to the input terminals of the logic. Thus only the leakage current flows between the terminals of the supply. The CMOS gate can be operated on wide range of supply voltage between 3 V to 15 V. It has good noise margin better than TTL devices. Fan-out of this is much larger. The speed of the CMOS logic is comparable with that of TTL circuits but larger than Schottky TTL circuits.

7.14.1 CMOS Inverter: Figure 7.27 shows the circuit diagram of CMOS inverter which consist of a PMOS transistor T_1 and an NMOS transistor T_2 which are connected in complementary mode. The drains of both the transistors are connected together, through which the output is taken. The source terminal of PMOS transistor T_1 is connected to the positive supply, where as the source of the NMOS transistor T_2 is grounded.

When the input A is grounded (logic 0), the gate of PMOS transistor T_1 is at the negative potential with respect to its source, so it is ON. The gate of NMOS transistor T_2 is at ground potential, so it is off. The output is, therefore, high ($+V_{DD}$), logic 1.

If on the other hand input A is high (logic 1), the gate of PMOS transistor T_1 is at zero potential with respect to its source, so it is off. The gate of NMOS transistor T_2 is at the positive potential with respect to ground, so it is ON. The output is, therefore, low logic 0.

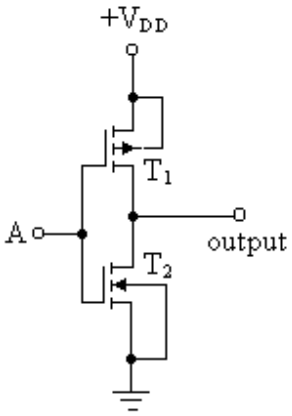


Fig. 7.27

7.14.2 CMOS NAND Gate: The circuit diagram of CMOS NAND gate is shown in figure 7.28. The two PMOS transistors T_1 and T_2 are connected in parallel with the sources connected together and two NMOS transistors T_3 and T_4 are connected in series.

When both the inputs are at logic 0 (grounded), the gates of T_1 and T_2 are at negative potentials with respect to their sources; the gates of T_3 and T_4 are at zero potential. So both PMOS transistors (T_1 and T_2) are ON and NMOS transistors T_3 and T_4 are off. The output will, therefore, be high (logic 1).

When input A is at logic 0 (grounded) and input B is at logic 1, the gate of T_1 is at negative potential with respect to its source and the gate of T_2 will be zero; the gates of T_4 and T_3 are at zero potential and V_{DD} potential respectively. So T_1 and T_3 are ON and T_2 and T_4 are off. The output will, therefore, be high (logic 1).

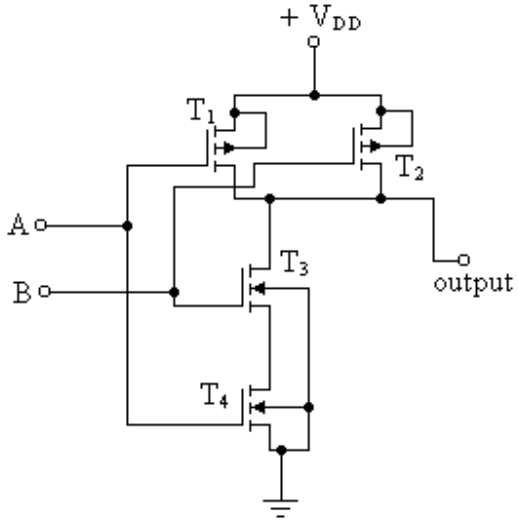


Fig. 7.28

When input A is at logic 1 and input B is at logic 0 (grounded), T_1 and T_3 will be off and T_2 and T_4 will be ON. The output will, therefore, be high (logic 1).

When both the inputs are at logic 1 ($+V_{DD}$), the gates of T_1 and T_2 are at zero potential; the gates of T_3 and T_4 are at negative potentials with respect to their sources. So both PMOS transistors (T_1 and T_2) are off and NMOS transistors T_3 and T_4 are ON. The output will, therefore, be grounded (logic0).

7.14.3 CMOS NOR Gate: : The circuit diagram of CMOS NOR gate is given in figure 7.29. The two PMOS transistors T_1 and T_2 are connected in series and two NMOS transistors T_3 and T_4 are connected in parallel.

When both the inputs are at logic 0 (grounded), the gate of T_1 and T_2 are at negative potentials; the gates of T_3 and T_4 are at zero potential. So both PMOS transistors (T_1 and T_2) are ON and NMOS transistors T_3 and T_4 are off. The output will, therefore, be high (logic 1).

When input A is at logic 0 (grounded) and input B is at logic 1, the gate of T_1 is at negative potential with respect to its source and the gate of T_2 will be zero; the gates of T_3 and T_4 are at zero potential and V_{DD} potential respectively. So T_1 and T_3 are ON and T_2 and T_4 are off. The output will, therefore, be low (logic 0).

When input A is at logic 1 and input B is at logic 0 (grounded), T_1 and T_3 will be off and T_2 and T_4 will be ON. The output will be low (logic 0).

When both the inputs are at logic 1 ($+V_{DD}$), the gates of T_1 and T_2 are at zero potential; the gates of T_3 and T_4 are at V_{DD} potential. So both PMOS transistors (T_1 and T_2) are off and NMOS transistors T_3 and T_4 are ON. The output will, therefore, be grounded (logic0).

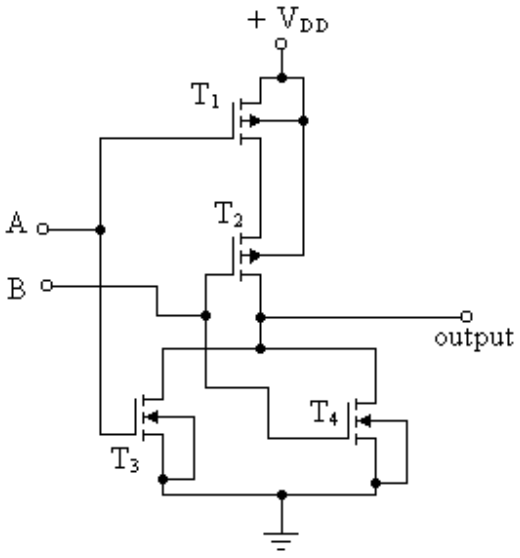


Fig. 7.29

7.15 Comparison of Logic Families: The comparison of important logic families are given in table 7.1 in respect of logic parameters.

Table 7.1

Logic Parameters	RTL	DTL	TTL	ECL	MOS	CMOS
Basic gates with +ve logic	NOR	NAND	NAND	OR/NOR	NAND	NAND/NOR
Maximum fan-in	5	10	8	5	8	8
Fan-out	5	8	10	25	20	>50
Power dissipation / gate (in mW)	12	10	10	50	1	0.01 static at 1 MHz
Propagation delay per gate (nsec)	20	30	12	4	400	70
Noise immunity	Nominal	Good	Very Good	Good	Nominal	Very good
Number of functions	High	Fairly high	Very high	High	Fair	Good
Clock rate, MHz	5	12	15	300	2	5

Following is the list of general TTL gates available in the form of 54/74 series SSI.

54/7400	Quad two-input NAND gate
54/7402	Quad two-input NOR gate
54/7403	Quad two-input NAND gate with open collector
54/7404	Hex inverter
54/7405	Hex inverter with open collector
54/7406	Hex inverter buffer
54/7407	Hex buffer
54/7410	Triple three-input NAND gate
54/7411	Triple three-input AND gate
54/7420	Dual four-input NAND gate
54/7430	Single eight-input NAND gate
54/7440	Dual eight-input NAND buffer

Following is the list of general CMOS gates available in the form of CD 40 series SSI

4000	Dual three-input NOR gates plus one inverter
4001	Quad two-input NOR gate
4002	Dual four-input NOR gate
4011	Quad four-input NAND gate
4012	Dual four-input NAND gate
4023	Triple three-input NAND gate

Problems:

1. What are positive and negative logics? Explain the logic diagram of two-input AND gate.
2. What do you understand by the term logic? Discuss three-input diode OR gate.
3. How an inverter circuit works, explain with neat diagram using a transistor.
4. What is logic family? Give the classification of logic family. Mention the classification of digital ICs.
5. Define the following parameters related to logic gates:
   Fan-in, Fan-out, Propagation delay time, Power dissipation and Noise margin.
6. Draw the logic circuit diagram of RTL NOR gate. Explain its operation. Mention its advantages and disadvantages of RTL family
7. Draw the DCTL circuit of three input-NOR gate and explain its operation.
8. Discuss the operation of three-input I^2L NOR gate. Mention its advantages and disadvantages of this logic family.
9. Draw the circuit diagram of DTL NAND gate for three-inputs. Explain its working. What is the function of resistance connected between the base and emitter of the transistor used in the circuit? What are the disadvantages of this logic family?
10. Draw the logic diagram of two-input HTL NAND gate. Explain its operation. Mention its advantages also.
11. Draw the basic circuit diagram of positive logic two-input TTL NAND gate. Explain its operation and mention its disadvantages. How the disadvantages of this basic circuit are removed?
12. Draw and explain the circuit diagram of positive logic two-input TTL NAND gate with totem-pole output. What are the advantages and disadvantages of this logic family?
13. Draw and explain the following TTL circuits with Totem-pole outputs:
    (i) TTL inverter
    (ii) Positive logic two-input TTL NOR gate
    (iii) Positive logic two-input TTL AND gate
    (iv) Positive logic two-input TTL OR gate
14. Draw and explain the circuit of open collector two input TTL NAND gate. What is the main advantage of this open collector gate?
15. Show open collector TTL NAND gate can be used as wire-AND.
16. Prove that two open collector TTL inverters when connected together produce the NOR operation.
17. Write short note on Tri-state inverter.

18. Draw and explain the working of two-input Schottky TTL NAND gate. What is the use of Schottky transistors in this logic?

19. What are the advantages of emitter coupled logic? Discuss the working of four-input NOR/OR ECL gate. Show that external wire connections for two ECL NOR outputs (or OR outputs) produce a wired-OR (wired-AND) function.

20. Draw and discuss the following NMOS gates:
    (i) inverter
    (ii) Positive logic two-input NOR gate
    (iii) Positive logic two-input NAND gate

21. Discuss CMOS NAND and NOR gates. What are advantages of CMOS logic.

22. Write short note on CMOS inverter.

23. State the various logic families available in the market. Give the comparison of the logic families with respect to following parameters:
    Fan-in, Fan-out, power dissipation, propagation delay, Noise immunity and clock rate.

8

Flip-flops

Basically, two types of switching circuits are used in digital systems, namely combinational and sequential switching circuits. The combinational circuits which are the combinations of logic circuits have been discussed in 5th and 6th chapters of this book. The other class of switching circuits is known as sequential circuits. In sequential circuits the outputs not only depend on the instant (present) values of the input variables but also on the past outputs. The past outputs are, in fact, the functions of the previous inputs. So the sequential circuits have the direct inputs which are externally controlled and known as primary inputs. An arrangement is also made to feedback the past outputs to the input terminals. These feedback terminals are known as the secondary inputs. The secondary inputs are the delayed outputs and act as the memory elements. A basic memory element which is capable of storing one bit of information is the flip-flop. The detailed discussion on the various flip-flops will be made in this chapter.

8.1 R S Flip-flop: Flip-flop is a basic memory element used in sequential circuits. The flip-flop has two stable states – logic 0 or logic 1. The flip-flop will either be in one of the two stable states after application of the input signals; it will remain to be in that state even if the inputs are removed. Flip-flops are also known as the latch or toggle. The RS flip-flop is the simplest flip-flop which can be constructed using NOR gates or NAND gates. Figure 8.1 shows the basic circuit of RS flip-flop constructed using NOR

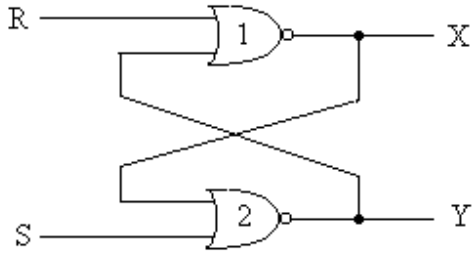


Fig. 8.1

gates. In this circuit R & S are the two inputs and X & Y are the outputs which are being applied back to the input terminals of the NOR gates. The behaviour of this circuit may

be analysed by replacing each NOR gate by an ideal NOR gate and a delay factor represented by a rectangular as shown in figure 8.2. The delay factor is the propagation

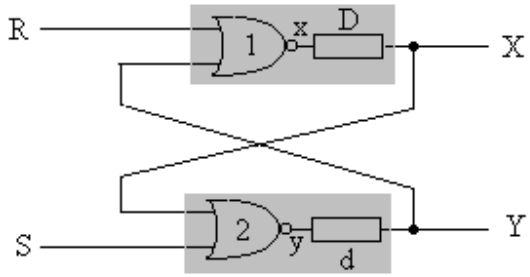


Fig. 8.2

delay time of each gate which is supposed to be different for each gate. Let delay of one gate is D and that of other gate is d. Further it is assumed that x and y are outputs of ideal NOR gates, which is transferred to the final output after the delay of each gate. So at any instant of time t the output *X* was the same as x was before (t – D) sec. Similarly one can explain for *Y* output.

So $X(t) = x(t - D)$ and $Y(t) = y(t - d)$

The outputs *x* and *y* are given by:

$$x = (\overline{R + Y}) = \overline{R} \cdot \overline{Y} \qquad y = (\overline{S + X}) = \overline{S} \cdot \overline{X}$$

The K-maps for x, y and xy (values of both x and y are placed together) are shown in figure 8.3(a), (b) and (c) respectively.

XY \ RS	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

Fig. 8.3(a)

XY \ RS	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	0	0
10	0	0	0	0

Fig. 8.3(b)

XY \ RS	00	01	11	10
00	11	10	(00) D	01
01	(01) A	00	00	(01) E
11	00	00	00	00
10	(10) B	(10) C	00	00

Fig. 8.3(c)

For particular values of input variables R & S, if the values of *X* and *Y* are not equal to *x* and *y*, then the circuit is unstable and further change will take place. This change will go on till the values of *X* and Y are not equal to x and y. So when the values of XY are equal to xy, then the circuit is said to have attained the stable state. In that condition no further change will take place. In the K-map (fig.8.3 c), encircled values show the stable states, since the encircled values (xy) are the same as XY in the same row of the map.

Now we analyse the behaviour of this RS latch. Let us consider a situation that RS = 00 and XY = 11 corresponding to which xy = 00 (3^{rd} row and 4^{th} column of K-map of figure 8.3 c). This is unstable state as the values of xy are not equal to the values of XY,

so further change will take place. Thus XY must acquire the values of xy. This means that both X and Y must change from 1 to 0, but this change will not simultaneously occur as the delays of the two NOR gates are not same. Two cases will occur.

(i) It is assumed that D < d, X will therefore change faster than Y. So X will have the value as 0 and Y is yet to be changed. In this process XY will have the intermediate value as 01. This intermediate value of XY as 01 will immediately produce the value of xy as 01. So for RS = 10 and xy = XY = 01, the circuit will be in the stable state E (2^{nd} row and 4^{th} column of K-map of fig. 8.3c).

(ii) If on the other hand d < D, Y would have changed faster than X. So XY will have the intermediate value as 10, due to which xy will immediately get the value as 00 (4^{th} row and 4^{th} column in the K – map of fig. 8.3c). This is not a stable state and further change will, therefore, take place. Now XY will attain the value as 00 after some delay; this new value of XY will produce the new value of xy as 01, which is still not a stable state. So XY will change to as 01. This will lead the value of xy as 01 (shifts to 2^{nd} row), i.e. the circuit will reach to stable state E.

From the above discussion, it is clear that the latch will reach to the stable state E either directly through one transition stage or through several stages (shown by arrows in the K-map). This is called the race condition. In this type of race the destination is the same stable state E, however, one would never know which path will be followed for the transitions as the delay in the gates is an inherent quantity. So this type of race is a valid race as it gives the predictable output.

Similarly one can find that if RS = 11, the circuit will reach to the stable state D and if RS = 01, circuit will be in the stable state C.

If RS = 00, the circuit may either be in the stable state A or B, as there are two stable state in the first column (fig. 8.3c). The actual state attained by the circuit will depend upon the previous values of inputs RS. The values of RS could be changed to 00 either from 01 or from 10.

(i) The values of RS are changed to 00 from 10. In this condition the circuit was in the stable state E when RS were 10 (xy = XY = 01). Now RS are changed to 00. So RS = 00 and XY = 01, xy will be 01. The circuit will reach to the stable state A.

(ii) The second case is now considered that the values of RS are changed to 00 from 01. When the circuit was having the values of RS as 01, the circuit was in the stable state C where xy = XY =10. Now the values RS are changed to 00. In this condition of RS = 00 and XY = 10, xy will have the values 10, the circuit attains the stable state B.

It is interesting to note from the above discussion that when RS are changed to 00 either from 01 or 10, the values of the outputs are their previous stable values (before the change).

There is one more possibility that the values of RS are changed to 00 from 11. The circuit was therefore in the stable state D (XY = xy = 00) before the change. Now RS

are changed to 00. So RS = 00 and XY = 00, it will have xy = 11 (1^{st} row and 1^{st} column), which is not a stable state so further change will take place. Thus XY must acquire the values of xy. This means that both X and Y must change from 0 to 1, but this change will not simultaneously occur as the delays of the two NOR gates are not same. Two cases will further occur.

(a) It is assumed that D < d, X will, therefore, change faster than Y. So X will have the value as 1 and Y is yet to be changed. In this process XY will have the intermediate value as 10. This intermediate value of XY as 10 will immediately produce the value of xy as 10. So for RS = 00 and xy = XY = 10, the circuit will be in the stable state B.

(b) If on the other hand d < D, Y would have changed faster than X. So XY will have the intermediate value as 01, due to which xy will immediately get the value as 01 the circuit will reach to stable state A.

From the above discussion, it is clear that the latch will reach either to the stable state B or to the stable state A. There is again a race condition. In this type of race the destination is not the same. One would never know the outcome as it will depend on the inherent delay of the gates. So this type of race, known as critical race, is not a valid race as output is not predictable. This type of race is avoided in such circuits.

Following are the inferences of the above analysis of R S flip-flop:

1. If the values of RS are changed to 01 both from 00 or 10 or 11, the flip-flop will reach to the stable state and output will be XY = xy = 10. It may be noted that X and Y are complement of each other.
2. If the values of RS are changed to 10 both from 00 or 01 or 11, the flip-flop will reach to the stable state and output will be XY = xy = 01. Further X and Y are complement of each other.
3. If the values of RS are changed to 00 either from 01 or 10, the flip-flop will have the previous stable state. The output will either be XY = xy = 10 or 01. X and Y are complement of each other.
4. If the values of RS are changed to 11 both from 01 or 10 or 00, the flip-flop will reach to the stable state and output will be XY = xy = 00. It may be noted that X and Y are not complement of each other.
5. If the values of RS are changed to 00 from 11, a critical race will occur in the latch and the output will be unpredictable. It may be XY = xy = 01 or 10.

Thus if the condition RS = 11 is disallowed, the behaviour of the latch will be predictable and XY will always be complement of each other. The outputs X and Y are therefore, renamed as Q and $\overline{Q}$ respectively. So the behaviour of the latch is summarized in table 8.1. The symbolic representation of this R S flip-flop is shown in figure 8.4. As it is this flip-flop is known as asynchronous, since its behaviour depends upon the sequence in which the input signals change. The outputs will be affected whenever the inputs changes.

Table 8.1

Inputs R S	Output X = Q	Mode	Remarks
1 0	0	Reset	Resets the output to 0.
0 1	1	Set	Sets the output to 1.
0 0	0 or 1	Store	Store the previous value.
1 1	Disallowed		Critical race

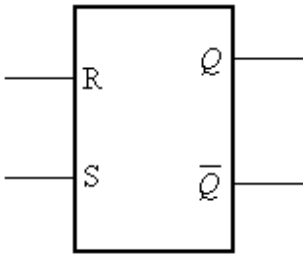


Fig. 8.4

8.1.1 R S Flip-flop with NAND Gates: The R S latch constructed using NOR gates, has been reproduced in figure 8.5(a). The equivalent circuit of this latch is shown in figure 8.5(b), in which inputs and outputs are inverted. The gates 1 and 2 of figure 8.5(b) are the Demorgan's form of NAND gates. So the latch is further redrawn with alternate symbols of NAND gates as shown in figure 8.5(c). This circuit is, therefore, the R S latch with NAND gates. Basically all the three circuits are same so their behaviour will also be same as summarized in table 8.1.

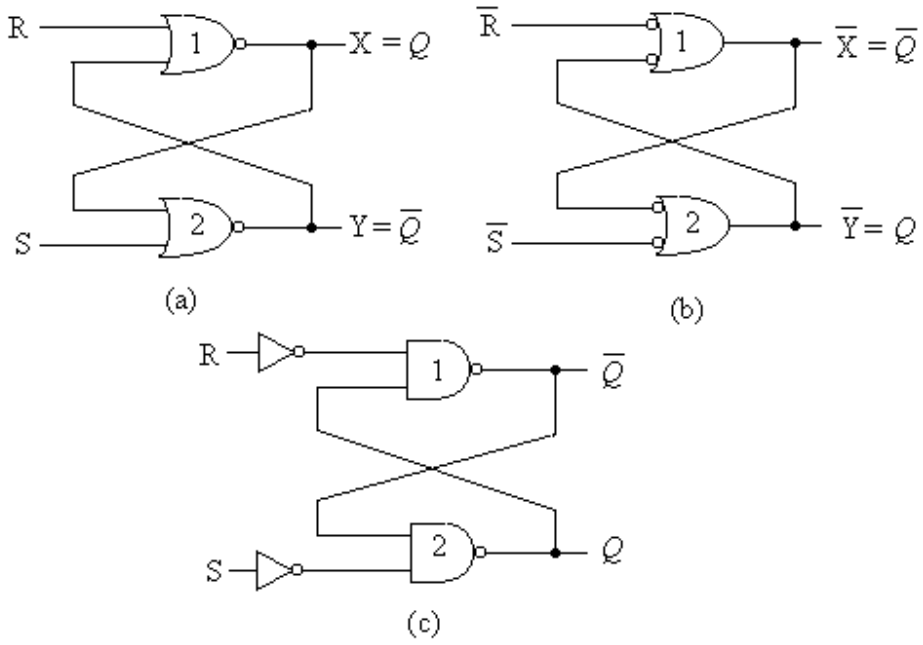


Fig. 8.5

8.1.2 Active Low R S Flip-flop with NAND Gates: Consider the circuit shown in figure 8.6, in which R and S inputs are directly applied to the cross coupled NAND gates and not through the two NOT gates. This circuit is known as the active low R S latch with NAND gates and its characteristic table is shown in table 8.2. It may be noted from this table that when both the inputs are 11, the latch stores the previous value. It sets and resets the latch when the inputs are 10 and 01 respectively. This NAND latch gives the ambiguous output (disallowed output) when inputs are 00. The direct approach is used to explain its behaviour though it may be explained in the similar fashion as for NOR latch.

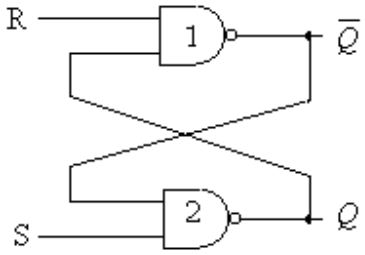


Fig. 8.6

Table 8.2

Inputs R S	Output X = Q	Mode	Remarks
0 1	0	Reset	Resets the output to 0.
1 0	1	Set	Sets the output to 1.
11	0 or 1	Store	Store the previous value
0 0	Disallowed		Critical race

Referring to figure 8.6, let initially, the output $Q = 0$ and $\overline{Q} = 1$. If 01 are applied to the inputs (R = 0 and S = 1), then output of gate 1 will remain as 1 and the output of gate 2 as 0. If on the contrary $Q = 1$ and $\overline{Q} = 0$, and inputs are 01 (R = 0 and S = 1), the output of gate 1 will be 1 ($\overline{Q} = 1$) and that of gate 2 will be 0 ($Q = 0$). So if R = 0 and S = 1, the output Q will always be 0 i.e. it resets the latch.

If the output $Q = 0$ and $\overline{Q} = 1$, and 10 are applied to the inputs (R = 1 and S = 0), then output of gate 1 will be 0 and the output of gate 2 as1. If on the contrary $Q = 1$ and $\overline{Q} = 0$, and inputs are 10 (R = 1 and S = 0), the output of gate 1 will be 0 ($\overline{Q} = 0$) and that of gate 2 will be 1 ($Q = 1$). So if R = 1 and S = 0, the output Q will always be 1 i.e. it sets the latch.

If the output $Q = 0$ and $\overline{Q} = 1$, and inputs are 11 (R = 1 and S = 1), then output of gate 1 will be 1 and the output of gate 2 as 0. If on the contrary $Q = 1$ and $\overline{Q} = 0$, and inputs are 11 (R = 1 and S = 1), the output of gate 1 will be 0 ($\overline{Q} = 0$) and that of gate 2 will be 1 ($Q = 1$). So if R = 1 and S = 1, the output will always be its previous value i.e. it is in store mode.

If the output $Q = 0$ and $\overline{Q} = 1$, and inputs are 0 0 (R = 0 and S = 0), then outputs of gate 1 and gate 2 will be 0. If on the contrary $Q = 1$ and $\overline{Q} = 0$, and inputs are 0 0 (R = 0 and S = 0), the output of gate 1 and that of gate 2 will be 1. So if R = 1 and S = 1, then outputs $Q = \overline{Q} = 1$ irrespective of previous state. This is a disallowed condition.

The table 8.2 is verified from the above discussion.

8.2 Clocked R S Flip-flop: The R S flip-flop or latch discussed in the previous sections was known as asynchronous flip-flop, since its behaviour depends upon the sequence in which the input signals change. The outputs were affected whenever the inputs changes. Now another type of flip-flop called the clocked R S flip-flop will be discussed which fall in the category of the synchronous flip-flop. In synchronous flip-flop the behaviour of the circuit can be defined from the knowledge of its signals at discrete instants of time. The synchronization is achieved by the timing device known as system clock. The system clock generates the periodic train of clock pulses and the outputs are affected to the application of clock pulse. Synchronous flip-flops are extensively used in

the sequential circuits because of their high reliability and ease of the design. The periodic train of clock pulses is shown in figure 8.7. The clock pulse remains high for short time is known as pulse width or pulse duration denoted by T_P. The front edge of the pulse is called as the leading edge of the pulse and the back edge of the pulse is known as trailing edge of the pulse. The time duration of the complete wave denoted by T is known as the time period.

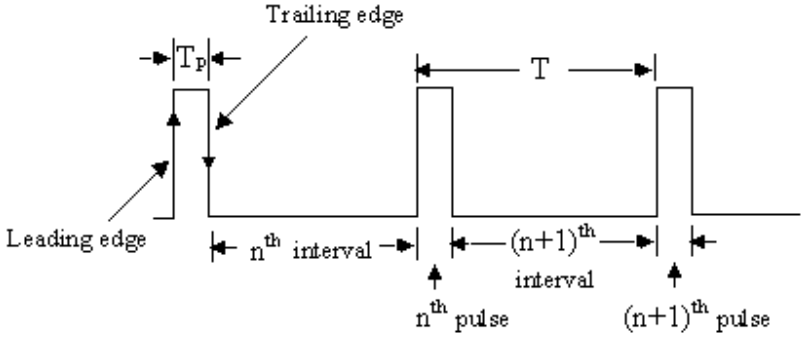


Fig. 8.7

The clocked R S flip-flop is illustrated in figure 8.8(a), in which the inputs R and S are applied along with the clock pulse to the NOR latch through two AND gates (called loading gates). During the pulse width T_P, both the AND gates will be enabled and R S inputs gets connected to the inputs of the latch. However, when the clock pulse is low both the AND gates will be disabled and inputs of the latch will be low. Therefore the latch will be in the store mode. The behaviour of this circuit may be explained below.

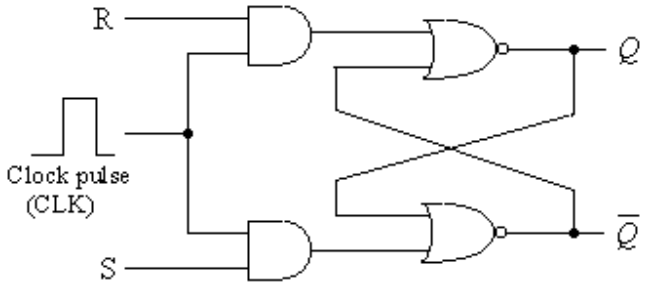


Fig. 8.8(a)

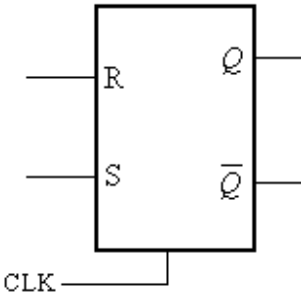


Fig. 8.8(b)

During the arrival of the n^{th} clock pulse, let the inputs to AND gates are R_n and S_n, which are applied to the inputs of the latch. Now as the internal of $(n+1)^{th}$ pulse start, the output Q_{n+1} will depend upon R_n, S_n and Q_n as:

$$Q_{n+1} = \overline{R_n + \overline{Q}_n} \quad \text{and} \quad \overline{Q}_n = \overline{S_n + Q_n}$$

So $$Q_{n+1} = \overline{R_n + \overline{S_n + Q_n}} = \overline{R}_n \cdot (S_n + Q_n)$$

It can further be verified that when $R_n = S_n = 0$ then output $Q_{(n+1)}$ will be same as Q_n. So if $Q_n = 0$ then Q_{n+1} will be 0; if $Q_n = 1$ then Q_{n+1} will also be 1. The flip-flop is said to in the store mode.

If $R_n = 0$ and $S_n = 1$, then the flop-flop will be in the set mode and gives the output as 1 irrespective of Q_n is 0 or 1.

The flip-flop will be in the set mode ($Q_{n+1} = 1$), if the inputs Rn = 1 and Sn = 0; the previous output could be either 0 or 1.

The flip-flop disallows the condition for $R_n = S_n = 1$.

The behaviour of this clocked R S flip-flop is characterized in table 8.3. The Boolean expression for Q_{n+1} can also be verified from K-map of table 8.3. The symbolic representation of the clocked R S flip-flop is shown in figure 8.8(b).

Table 8.3

R_n	S_n	Q_n	Q_{n+1}	$Q_{n+1} = \overline{R_n} \cdot (S_n + Q_n)$
0	0	0	0 Stores	$Q_{n+1} = Q_n$
0	0	1	1 Stores	
0	1	0	1 Sets	$Q_{n+1} = 1$
0	1	1	1 Sets	
1	0	0	0 Resets	$Q_{n+1} = 0$
1	0	1	0 Resets	
1	1	0	Disallowed	
1	1	1	Disallowed	

8.2.1 Clocked R S Flip-flop with NAND Latch: The asynchronous R S flip-flop (active high) is designed by the circuit shown by shaded portion in figure 8.9(a). Now two AND gates are used for loading the inputs R S and the clock pulse. Instead of using AND gates,

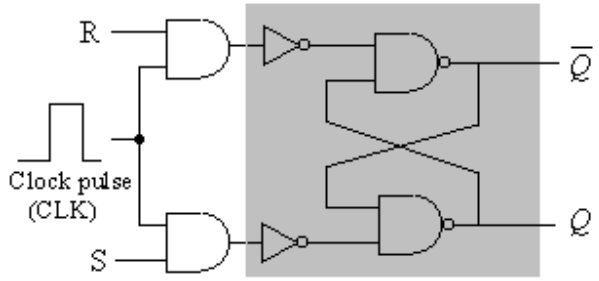


Fig.8.9(a)

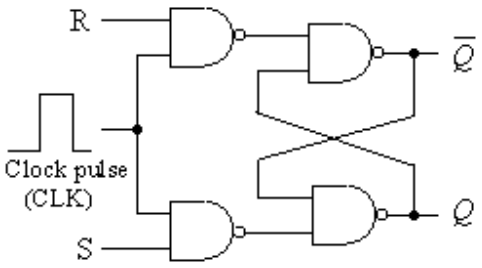


Fig. 8.9(b)

NAND gates may be used as shown in figure 8.9(b). It may be noted that AND and NOT gates form NAND gates. When the clock pulse is low, the outputs of loading NAND gates are high putting the NAND latch in store mode. So Q and $\overline{Q}$ will have its previous value. Now when the clock pulse is high, the circuit will behave as the clocked R S flip-flop discussed above. It also verifies the table 8.3.

Example 8.1 Draw the waveform of the output Q of clocked R S flip-flop, if R and S inputs applied to it, are as represented in figure 8.10(a). The latch is initially reset.

Solution: The waveform of the output Q of clocked R S flip-flop is shown in fig. 8.10(b), in which mode of operation of the flip-flop is indicated. The changes at the output take place at the leading edge of the clock pulse (CLK).

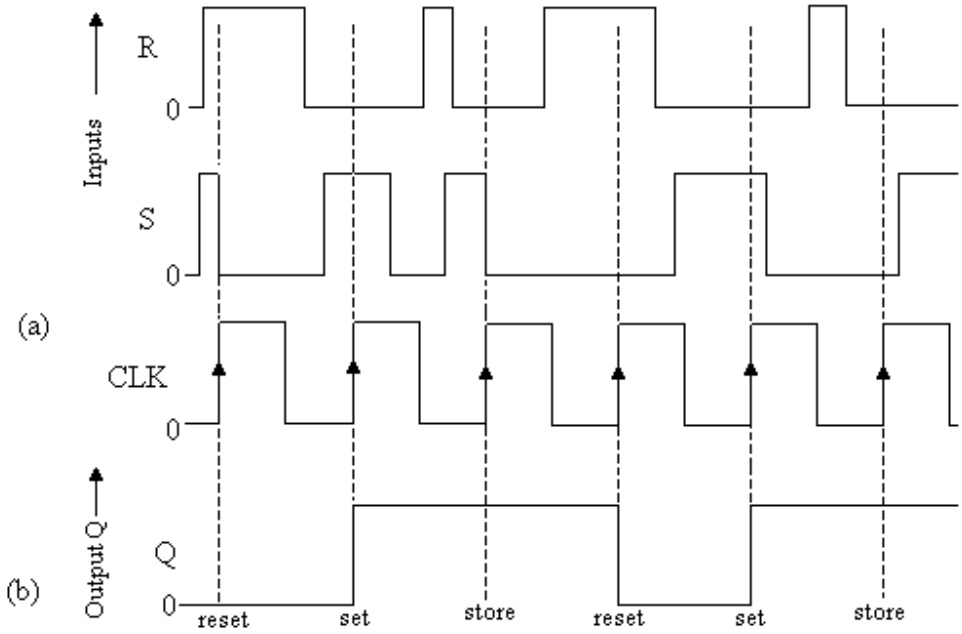


Fig. 8.10

Example 8.2: Modify an asynchronous R S flip-flop so that when both the inputs R and S are 1, the flip-flop is set.

Solution: The flip-flop is modified as shown in figure 8.11, in which two new inputs named as R' and S' are used. The table 8.4 verifies that when R' and S' are 11, the inputs R S are 01 and flip-flop is set.

Table 8.4

Inputs		Inputs		Mode of operation
R'	S'	R	S	
0	0	0	0	Store
0	1	0	1	Set
1	0	1	0	Reset
1	1	0	1	Set

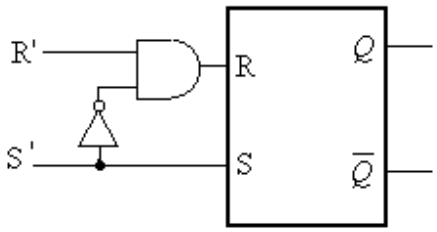


Fig. 8.11

8.3 Triggering of Flip-flops: It has been discussed in clocked flip-flops that the flip-flops work after the application of clock pulse, i.e. when the clock pulse goes low to high, the flip-flop triggers (or flip-flop enables). Such type of triggering, known as level triggering is generally used. But in some digital systems changes in the output occur either at leading edge (positive edge or rising edge) or at the trailing edge (negative edge or falling edge) of the clock pulse. Such type of triggering is known as edge triggering.

So the flip-flops should either be positive edge triggered flip-flops or negative edge triggered flip-flops. A small triangle shown at the clock terminal of the flip-flop indicates the positive edge triggered flip-flop. Figure 8.12(a) shows the symbol of positive edge triggered R S flip-flop. However, small triangle with a bubble at the clock terminal of the flip-flop indicates the negative edge triggered flip-flop. The symbol for negative edge triggered R S flip-flop is shown in figure 8.12(b).

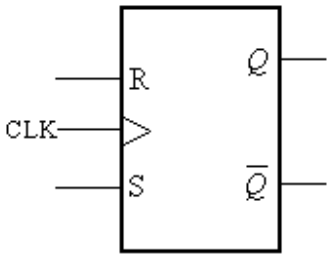


Fig. 8.12(a)

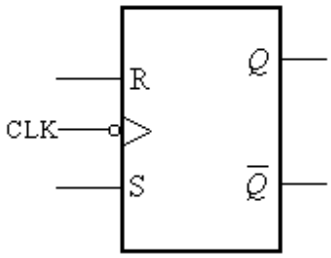


Fig. 8.12(b)

8.3.1 Edge Detector Circuit: The narrow spikes at the leading edge or at the trailing edge of the clock pulse is obtained by the edge detector circuit. Figure 8.13(a) shows the edge detector circuit for the generation of narrow spikes at the leading edge of the clock

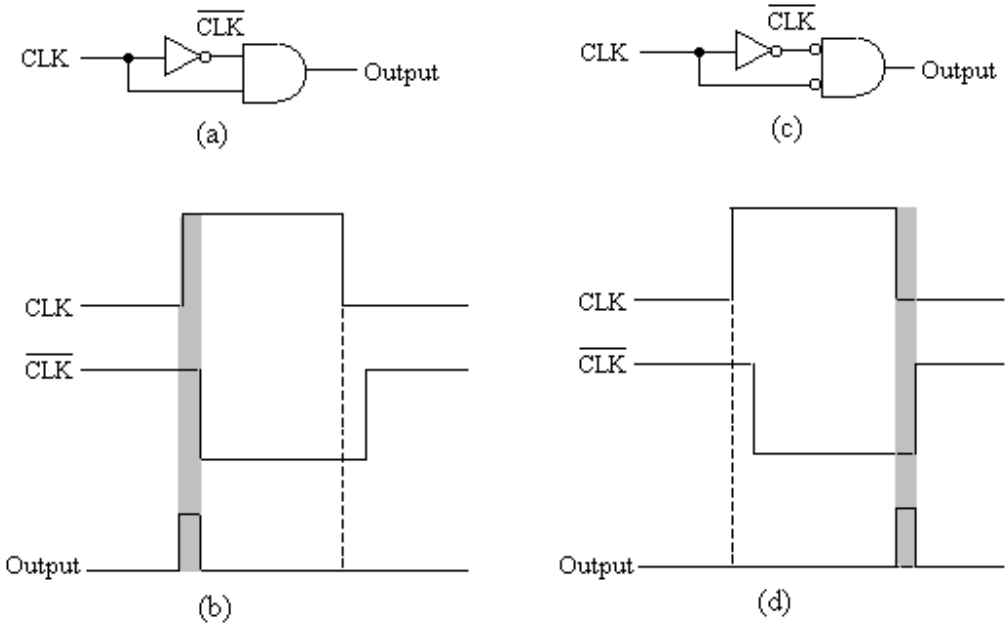


Fig. 8.13

pulse. In this circuit the clock pulse is applied to an inverter and an AND gate. The inverter produces an output ($\overline{CLK}$) after some time delay of a few nanoseconds because of the propagation delay of NOT gate. The AND gate will produce high output for few nanosecond (equal to the propagation delay of inverter), when both CLK and $\overline{CLK}$ are high. The AND gate, therefore, produces a narrow spike at the leading edge of the clock pulse (CLK) as shown in figure 8.13(b).

Similarly, the edge detector circuit may be drawn for the generation of narrow spikes at the trailing edge of the clock pulse. Figure 8.13(c) shows such circuit in which an inverter and an active low AND gate are used. The active low AND gate will produce high output for few nanosecond (equal to the propagation delay of inverter), when both CLK and $\overline{CLK}$ are low. This circuit, therefore, produces a narrow spike at the trailing edge of the clock pulse (CLK) as shown in figure 8.13(d).

8.4 The D Flip-flop: The modified form of clocked R S flip-flop is a D flip-flop which is illustrated in figure 8.14(a). The D flip-flop has only one input in addition to the clock pulse. The R and S inputs of R S flip-flop are not used in some applications when both the inputs are 00 or 11. This condition also eliminates the condition of RS = 11. So in D flip-flop R S inputs are always kept complement of each other and D input is applied to the S input and complement of S is applied to R input as shown in figure 8.14 (a) and (b).

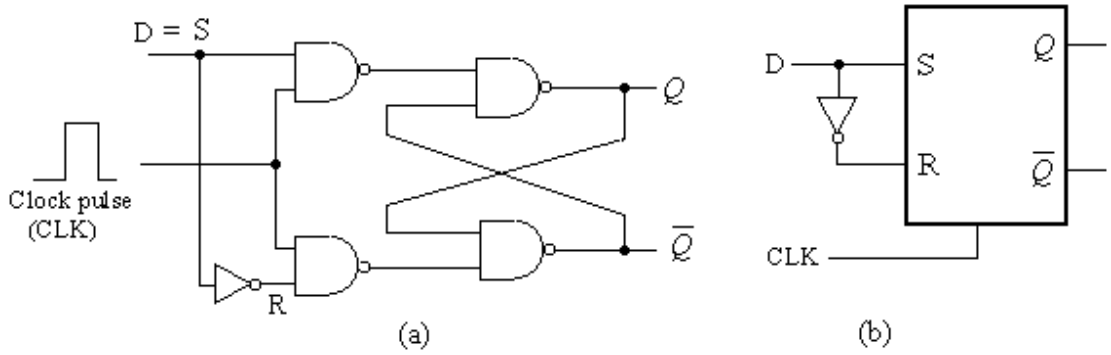


Fig. 8.14

When the input D is high and positive clock pulse is applied the latch will set, irrespective of previous value of the output Q. Similarly, if D input is low and clock pulse is applied, the latch will be reset; again the output will not depend upon its previous value. The behaviour of this flip-flop is shown in table 8.5. It may be noted that the value of D (data) will reach the output after the application of clock pulse i.e. $Q_{n+1} = D_n$. When no pulse is applied to the flip-flop or CLK is 0, the value of D will not reach to the output and the output will have its previous value. In other words the output Q will follow the input D when the clock is high. The D flip-flop is also called as the delay flip-flop. The symbolic representation of this flip-flop is shown in figure 8.15.

Table 8.5

D_n	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

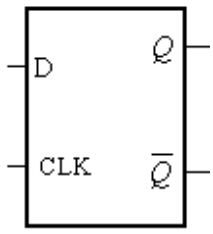


Fig. 8.15

Example 8.3: Draw the waveform of the output Q of D flip-flop, if D input and clock pulse are represented in figure 8.16 (a). The latch is initially reset.

Solution: The required wave Q output of d flip-flop is shown in figure 8.16 (b). The output follows the input D when the clock is high.

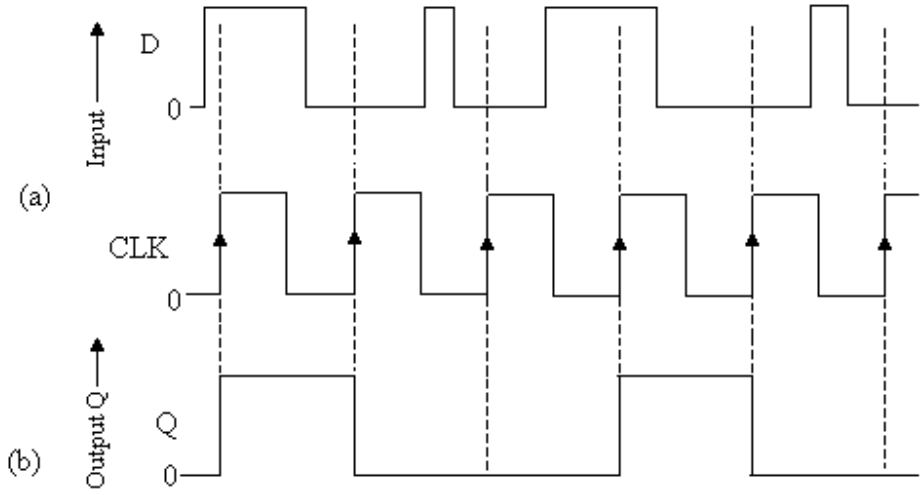


Fig. 8.16

8.5 The J K Flip-flop: In R S flip-flop when both the inputs are 11, the outputs (Q and $\overline{Q}$) were not complement to each other and this condition was disallowed. The R S flip-flop can be modified so that even when R and S inputs are 11, the outputs are complements of each other. The J K flip-flop shown in figure 8.17(a) is the modified

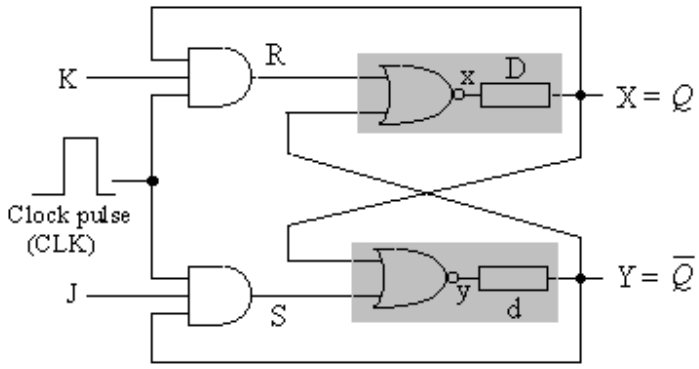


Fig. 8.17(a)

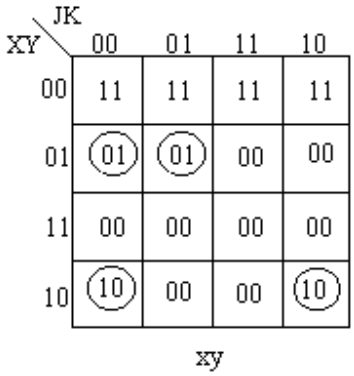


Fig. 8.17(b)

form of R S flip-flop. This flip-flop has two inputs J and K, the J input correspond to S (set) input and K input correspond to R (Reset) input. It may be noted that the outputs of the latch are connecting to its own loading gate, which forms the complementing outputs when both J and K inputs are 11. If the flip-flop is set before being JK = 11, the R and S inputs will become 01 when JK = 11 and the flip-flop will be set giving the output Q as 1. If on the other hand flip-flop is set before being JK = 11, the flip-flop will be reset after becoming J K to be 11.

To analyse the behaviour of this flip-flop, the K-map for xy is drawn as shown in figure 8.17 (b), by getting the following functions:

When CLK = 1 $\quad R = X \cdot K \quad$ and $\quad S = J \cdot Y$

$$x = \overline{R + Y} = \overline{K \cdot X + Y} \qquad \text{and} \qquad y = \overline{S + X} = \overline{J \cdot Y + X}$$

In the K-map all the stable states are encircled and one can observe from these stable states that x and y are complements to each other. When the inputs J K are 00, the outputs will be either 01 or 10 storing their previous values. When JK = 01 the flip flop will be in the reset mode and when JK = 10 it will be in the set mode.

When JK = 11, the outputs will be complement to each other, but this will not give a stable state. This condition will now be discussed in detail. The inputs J K will be changed to 11 either from 00 or from 01 or from 10. Let the inputs J K change from 10 to 11. When the inputs J K were 10, the flip-flop was in the set mode and Q was equal to1. After changing J K to 11, the inputs R S become 10 which will reset the latch. Thus the content of the latch is complemented i.e. the outputs Q $\overline{Q}$ change 01 to 10. But this is not a stable state. As the outputs Q $\overline{Q}$ are 10, these outputs will be applied back to the input terminals. The latch will, therefore, be reset if the clock pulse still high and output will again be the complementation of the previous outputs. This way the complementation will go endlessly till the clock pulse is high. So the complementation to occur only once, it becomes necessary that the clock pulse should be 0, before the output data (after delay) is applied back to the input terminals. This is possible if the width of the clock pulse is less than the delay in latch i.e. $T_P < d$ or D. This condition is known as race around condition.

8.5.1 Edge Triggered J K Flip-flop: It has been observed that the width of the clock pulse in J K flip-flop should be less than the delay in latch. However, the delay is an inherent quantity which can not be known by the user for the particular IC. This problem of pulse width can be eliminated if narrow spikes of few nanoseconds at the leading or trailing edge of the clock pulse is used. The narrow spikes can be generated by the edge detector circuit discussed in section 8.3.1. Figure 8.18 shows the symbolic representation of positive edge triggered J K flip-flop and its operation is given in table 8.6.

The upward arrows in the characteristic table of J K flip-flop shows that the transition at outputs of flip-flop will occur at the leading edge of the clock pulse.

When J and K inputs are 00, no change in the output values will take place at the positive edge of the clock pulse i.e. $Q_{n+1} = Q_n$, the circuit is in store mode.

When J K = 01, the flip-flop resets at the positive edge of the clock pulse i.e. $Q_{n+1} = 0$.

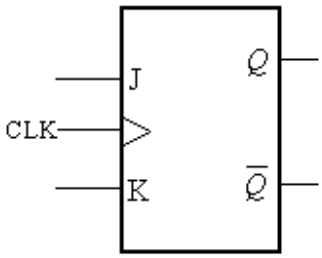


Fig. 8.18

Table 8.6

J_n	K_n	Q_n	Clock	Q_{n+1}	
0	0	0	↑	0	No Change
0	0	1	↑	1	
0	1	0	↑	0	Reset
0	1	1	↑	0	
1	0	0	↑	1	Set
1	0	1	↑	1	
1	1	0	↑	1	Complement
1	1	1	↑	0	

When J K = 11, the flip-flop toggles at the positive edge of the clock pulse i.e. it gives the complements of the previous outputs $Q_{n+1} = \overline{Q}_n$.

Figure 8.19 shows the symbolic representation of negative edge triggered J K flip-flop and its operation is given in table 8.7.

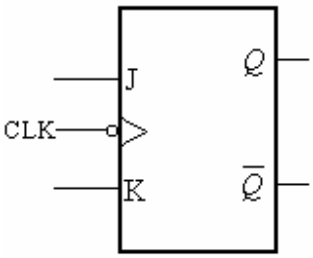


Fig. 8.19

Table 8.7

J_n	K_n	Q_n	Clock	Q_{n+1}	
0	0	0	↓	0	No Change
0	0	1	↓	1	
0	1	0	↓	0	Reset
0	1	1	↓	0	
1	0	0	↓	1	Set
1	0	1	↓	1	
1	1	0	↓	1	Complement
1	1	1	↓	0	

For simplicity the circuit diagram of edge triggered J K flip-flops using NOR and NAND latch are shown in figures 8.20 and 8.21 respectively.

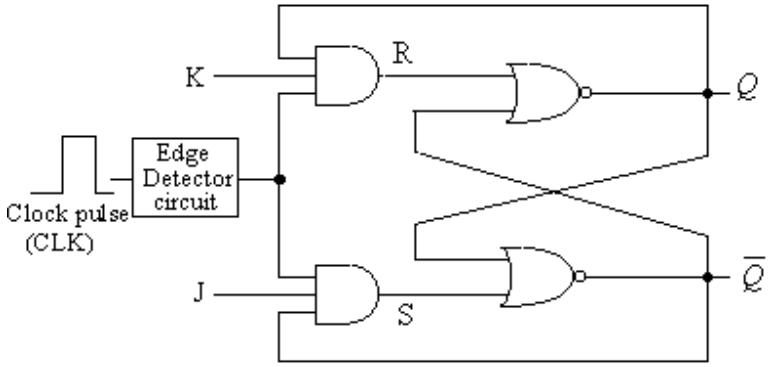


Fig. 8.20

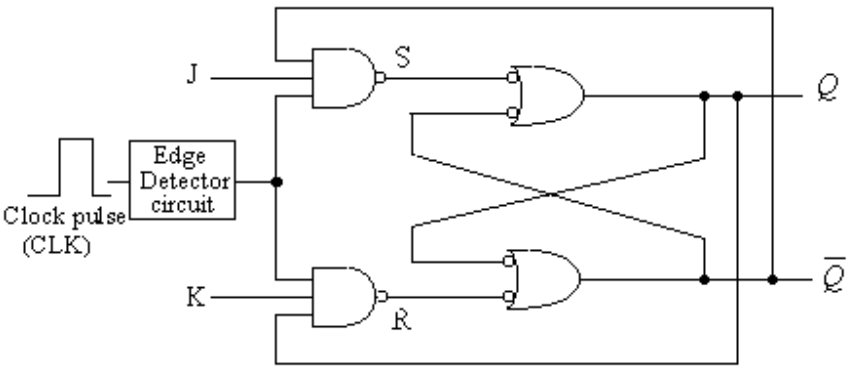


Fig. 8.21

Example 8.4: Draw the waveform of the output Q of a negative edge triggered J K flip-flop (figure 8.22 a), if J K inputs and clock pulse are represented in figure 8.22 (b). The flip-flop is initially reset.

Solution: The wave form of the output Q of the negative edge triggered J K flip-flop is shown in figure 8.22(C), the mode of operations at the trailing edge of the clock pulse (CLK) are also indicated in the figure.

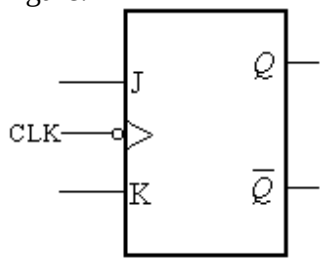


Fig. 8.22 (a)

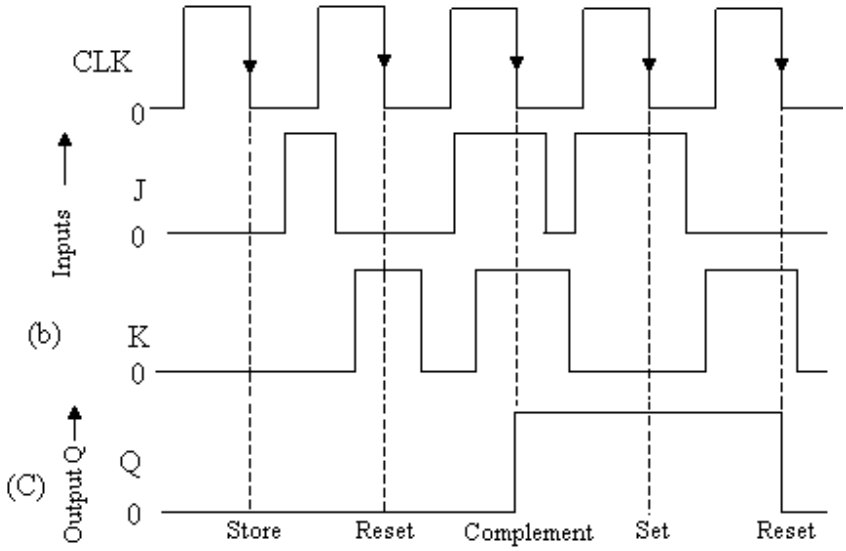


Fig. 8.22

8.5.2 Edge Triggered T (Toggle) Flip-flop: When J K inputs of an edge triggered Flip-flop are connected together to form a single input marked as T is known as Toggle

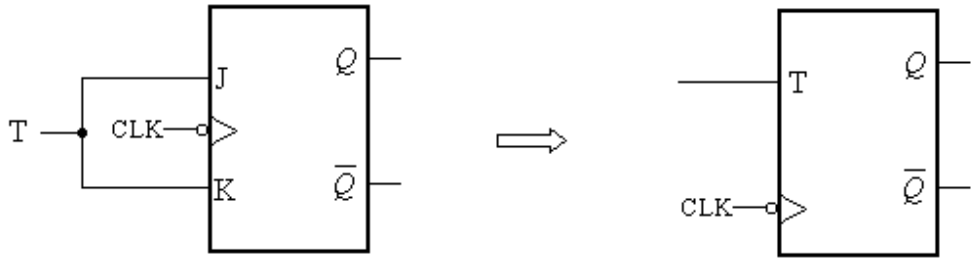


Fig. 8.23

(T) flip-flop. Figure 8.23 shows a negative edge triggered T flip-flop. This flip-flop will have only two options, i.e. when T = 0, the flip-flop will be in the store mode and gives

no change in the output. When input T = 1, the flip-flop will be in the complement mode i.e. it toggles or gives the complemented output of the previous value at the trailing edge of the clock pulse. The behaviour of T flip-flop is given in table 8.8.

Table 8.8

Input T	Previous output Q	Next Output Q	Mode of Operation
0	0	0	Store
0	1	1	
1	0	1	Complement
1	1	0	

Example 8.5: Draw the waveform of the output Q of a negative edge triggered T flip-flop (figure 8.24 a), if T input is connected to 1 (+5 V).The clock pulse are represented in figure 8.24 (b). The flip-flop is initially reset.

Solution: The T input of the flip-flop is connected to logic 1, so this flip-flop will toggle at the trailing edge of the clock pulse as shown in figure 8.24(c). It is clear from this figure that the frequency of the output wave is half of the input frequency. The T flip may, therefore, be used for the frequency division of the input clock.

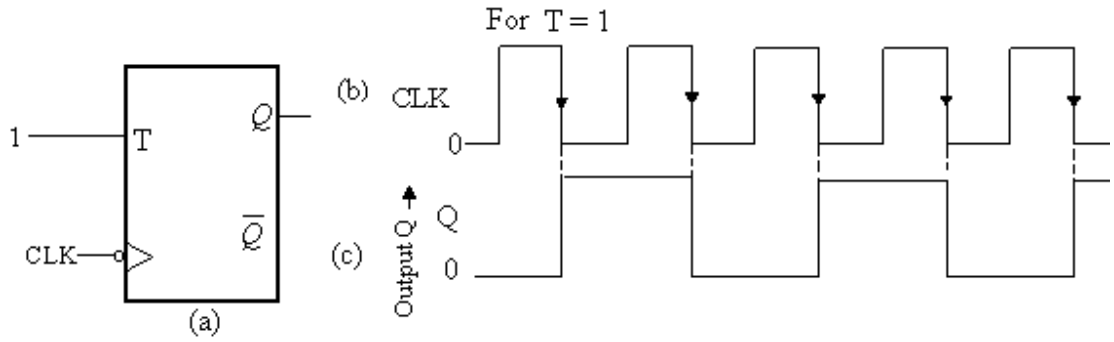


Fig. 8.24

8.5.3 Asynchronous Inputs: In synchronous flip-flops discussed above, the data is transferred synchronously to the outputs of flip-flops, only at the triggering edge of the clock pulse. These inputs connected to the flip-flops are known as synchronous inputs. In these flip-flops when power is switched ON, the state of flip-flop will be uncertain i.e. the output may either be in the set mode or in the reset mode. In many applications it is desired that the flip-flop is initially set or reset. This can be done by providing the extra inputs known as preset ($\overline{PRE}$) and Clear ($\overline{CLR}$) inputs. Figure 8.25(a) shows edge triggered J K flip-flop with preset ($\overline{PRE}$) and clear ($\overline{CLR}$) inputs. These inputs are called as asynchronous inputs as these inputs operate independently and do not depend on the clock pulse.

If both $\overline{PRE}$ and $\overline{CLR}$ asynchronous inputs are 1, the circuit behaves as a normal J K flip-flop and gives the outputs as per its characteristic table.

If $\overline{PRE}$ = 0 and $\overline{CLR}$ = 1, the output of gate 3 will be 1 (Q = 1). Consequently, all the three inputs of the gate 4 will be 1 giving the output $\overline{Q}$ = 0. Hence $\overline{PRE}$ = 0 sets the flip-flop.

If $\overline{PRE}$ = 1 and $\overline{CLR}$ =0, the output of gate 4 will be 1 ($\overline{Q}$ = 1). Consequently, all the three inputs of the gate 3 will be 1 giving the output Q = 0. Hence $\overline{CLR}$ = 0 resets the flip-flop.

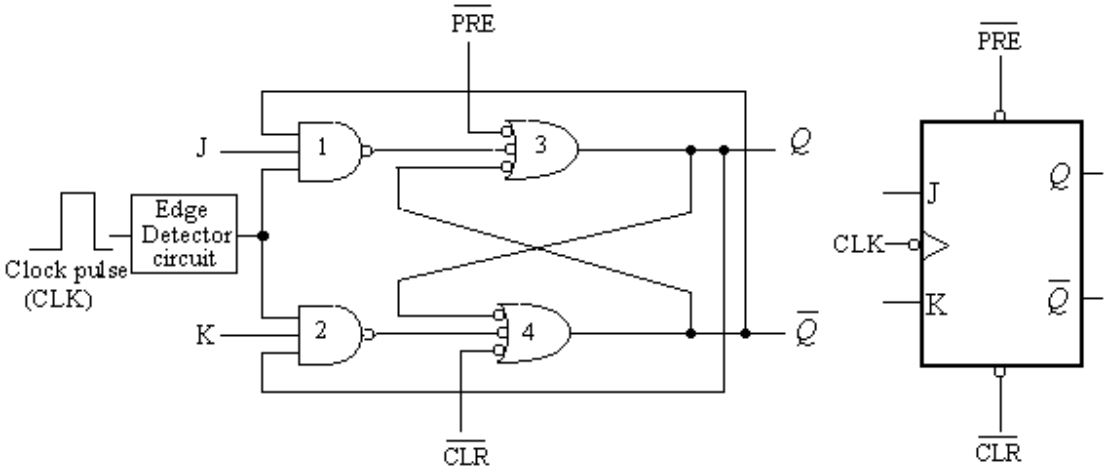


Fig. 8.25

If both $\overline{PRE}$ and $\overline{CLR}$ asynchronous inputs are 0, the circuit gives uncertain state and this condition must not be used.

One this thing should be remembered that once the state of the flip-flop is established asynchronously (set or reset), these asynchronous inputs $\overline{PRE}$ and $\overline{CLR}$ must be connected to 1, before the application of next clock pulse. Figure 8.24(b) shows the logic symbol of this flip-flop, in which asynchronous inputs are indicated separately. These inputs are active-low. The table 8.9 summarizes the operation of this flip-flop.

Table 8.9

Inputs			Output	Mode of opeariton
CLK	$\overline{PRE}$	$\overline{CLR}$	Q	
↑	1	1	Q_{n+1}	Normal FF
0	0	1	0	Preset
0	1	0	1	Clear

8.6 Master Slave J K flip-flop: Before the development of edge triggered flip-flops, the race around condition in J K flip-flop was removed using master slave flip-flop. A master slave J K fillip-flop is constructed using two J K flip-flops as shown in figure 8.26; one flip-flop is known as master flip-flop and other as slave flip-flop. The master flip-flop works on leading edge of the clock pulse and that of slave flip-flop works on the trailing edge of the clock pulse. So the master flip-flop responds to the inputs before the slave flip-flop. The master-slave flip-flop is in fact the pulse triggered flip-flop.

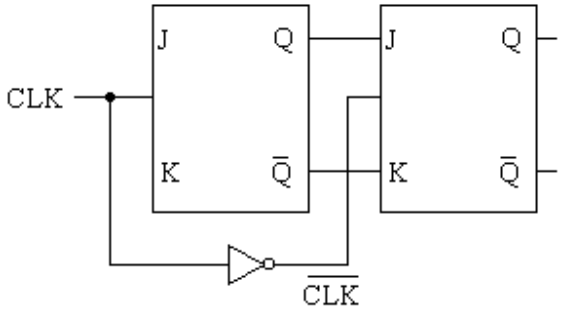


Fig. 8.26

The complete circuit diagram of a master-slave J K flip-flop using NOR latch is given in figure 8.27. Note that the outputs of the slave flip-flop are feedback to the master flip-flop. This will make the output of the flip-flop complement when JK = 11. At the

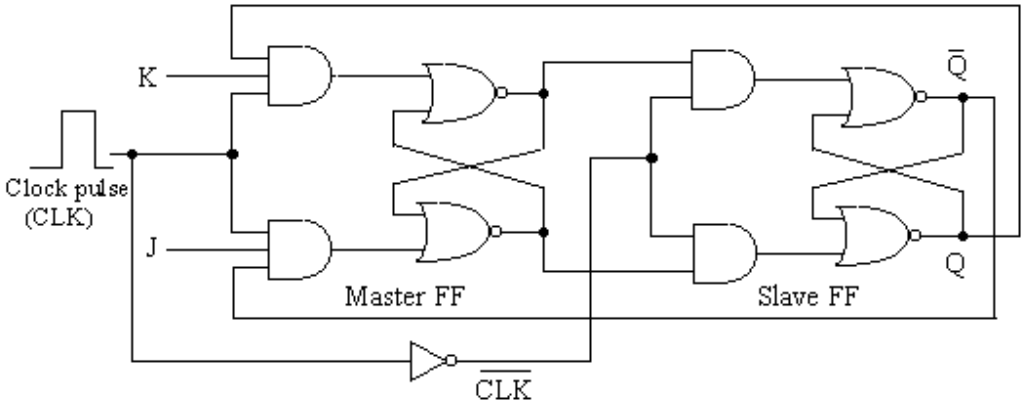


Fig. 8.28

start of the leading of the clock pulse, the master flip-flop works with the previous values of the outputs and the present values of the J K inputs, and whatever change to occur will

occur in the output of the master flip-flop till the clock pulse is high. So the final output is obtained at the trailing edge of the clock pulse. Table 8.7 may be verified with this circuit. The master – slave J K flip-flop with NAND latch is shown in figure 8.29.

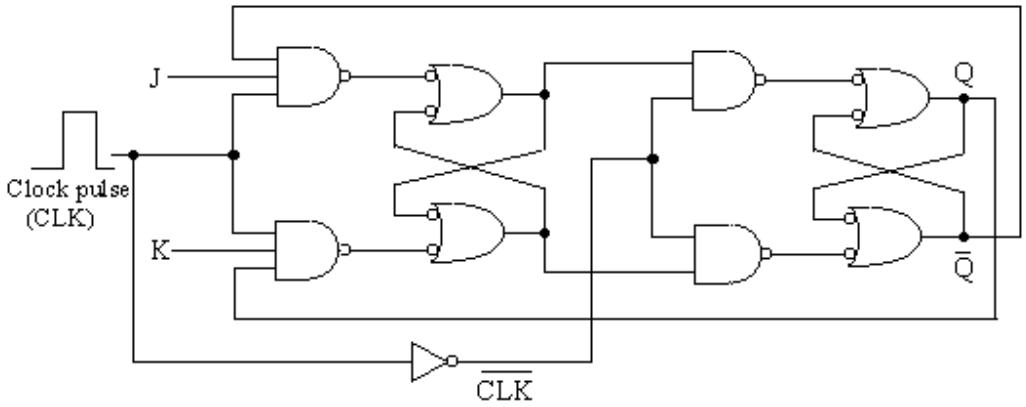


Fig. 8.29

8.7 Excitation Table of Flip-flops: The operational characteristics of various flip-flops have been discussed in earlier sections of this chapter. The truth table also referred to as characteristic table gives the operation of flip-flop. The characteristic table specifies the next state of the flip-flop when the inputs and present state is known. The characteristic tables for R S, J K, D and T type flip-flops are reproduced in tables 8.10(a) to (d) respectively. The suffix n indicated in the inputs and the outputs denote the present

Table 8.10(a)

R_n	S_n	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	Disallowed
1	1	1	Disallowed

Table 8.10(b)

J_n	K_n	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Table 8.10(c)

D_n	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

Table 8.10(d)

T_n	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

state of inputs and outputs; and the suffix (n + 1) in the outputs denotes the next state. It is generally required the input conditions of flip-flops for different transition from present state to next state. The table which illustrates these transitions is known as the excitation

table. The excitation table may be obtained from the characteristic table of the flip-flop. The excitation tables for R S, J K, D and T type flip-flops are given in tables 8.11(a) to (d) respectively. It may be noted from table 8.11(a), that when there is transition from 0 to 1 (present state to next state, Q_n to Q_{n+1}), the inputs R_n S_n should be either 00 or 10. That is input S should definitely be 0, where as input R can be 0 or 1 (may be denoted by φ, don't care condition). So for 0 to 0 transitions, inputs R S should be φ 0. Similarly, one can draw and verify the excitation table for other flip-flops.

Table 8.11(a)

Transitons $Q_n \rightarrow Q_{n+1}$	Inputs R_n	Inputs S_n	Inputs R_n	Inputs S_n
0 → 0	0	0	φ	0
	1	0		
0 → 1	0	1	0	1
1 → 0	1	0	1	0
1 → 1	0	0	0	φ
	0	1		

Table 8.11(c)

Transitons $Q_n \rightarrow Q_{n+1}$	Input D_n
0 → 0	0
0 → 1	1
1 → 0	0
1 → 1	1

Table 8.11(b)

Transitons $Q_n \rightarrow Q_{n+1}$	Inputs J_n	Inputs K_n	Inputs J_n	Inputs K_n
0 → 0	0	0	0	φ
	0	1		
0 → 1	1	0	1	φ
	1	1		
1 → 0	0	1	φ	1
	1	1		
1 → 1	0	0	φ	0
	1	0		

Table 8.11(d)

Transitons $Q_n \rightarrow Q_{n+1}$	Input T_n
0 → 0	0
0 → 1	1
1 → 0	1
1 → 1	0

8.8 Conversion of Flip-flops: A combinational circuit is designed for the conversion of one type of given flip-flop to other type. The general model for such conversion is illustrated in figure 8.30. The inputs of the required flip-flop are fed as

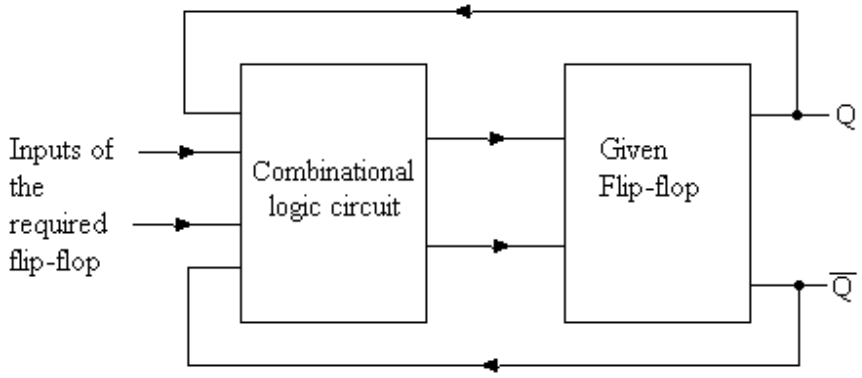


Fig. 8.30

inputs to the combinational circuit which are connected to the inputs of the given flip-flop. The outputs of the given flip-flop will be the outputs of the required flip-flop. The combinational logic circuit is obtained from the excitation tables of both the flip-flops (given and required). The Boolean expressions for the inputs of the required flip-flop are obtained from the K-map drawn for the inputs and outputs of the given flip-flop. The combinational logic circuit may be drawn as usual. The conversion may be illustrated in the following examples.

Example 8.6: Convert a J K flip-flop to R S flip-flop.

Solution: From the excitation tables (8.11b and 8.11a) of J K and R S flip-flops respectively, truth table is drawn as given in table 8.12.

Table 8.12

Inputs of the required FF			Output	Inputs of the given FF	
R_n	S_n	Q_n	Q_{n+1}	J_n	K_n
0	0	0	0	0	φ
0	0	1	1	φ	0
0	1	0	1	1	φ
0	1	1	1	φ	0
1	0	0	0	0	φ
1	0	1	0	φ	1

Now from this truth table, the Boolean expressions for J and K are obtained from the K –maps drawn in figure 8.31(a & b) as:

$$J = S \qquad \text{and} \qquad K = R$$

The logic diagram showing the conversion from J K flip-flop to R S flip-flop is given in figure 8.31(c).

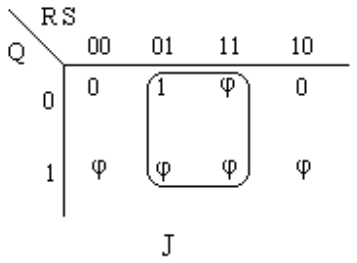


Fig. 8.31(a)

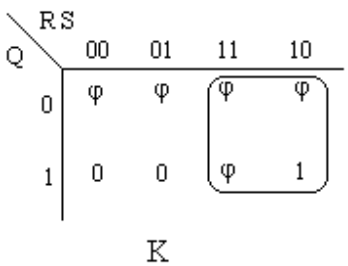


Fig. 8.31(b)

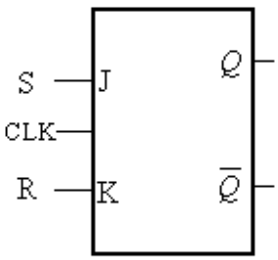


Fig. 8.31(c)

Example 8.7: Convert a D flip-flop to J K flip-flop.

Solution: From the excitation tables of D and J K flip-flops, truth table is drawn as given in table 8.13.

Now from this truth table, the Boolean expressions for D input is obtained from the K –maps drawn in figure 8.32 (a) as:

$$D = J \cdot \overline{Q} + \overline{K} \cdot Q$$

Table 8.13

Inputs of the required FF			Output	Input of the given FF
J_n	K_n	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

The logic diagram showing the conversion from D flip-flop to J K flip-flop is given in figure 8.32 (b).

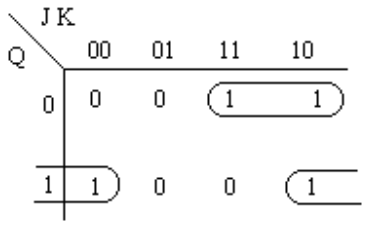


Fig. 8.32(a)

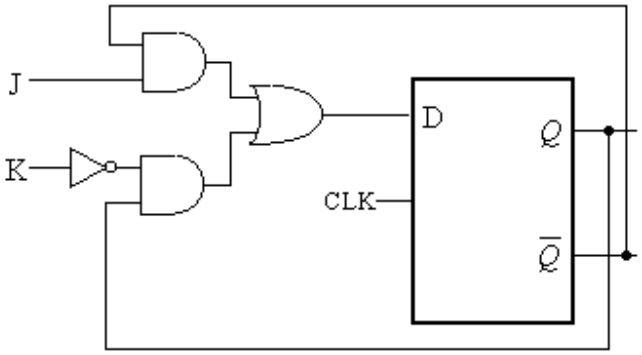


Fig. 8.32(b)

8.9 Flip-flop Parameters: Several parameters for the application of flip-flop will be discussed which are important to specify the performance, operating requirements and limitations of the circuit. These parameters are available in the data sheets for flip-flop ICs, and are applicable to all types of flip-flops.

Propagation Delay Time: As the flip-flops are the combination of logic gates, the flip-flop will also not respond immediately after the application of the clock pulse or asynchronous inputs. The flip-flops will also have the propagation delay time. The propagation time delay for the flip-flops may be defined as the time interval between the application of triggering edge or asynchronous inputs and the resulting output of the flip-

flop. The propagation delays that occur for the positive transition of the clock pulse applied to a flip-flop are illustrated in figure 8.33. They are defined as:

1. **Propagation Delay** t_{PLH} measured from the triggering edge of the clock pulse to the low to high transition of the output (refer figure 8.33a).
2. **Propagation Delay** t_{PHL} measured from the triggering edge of the clock pulse to the high to low transition of the output (refer figure 8.33b).

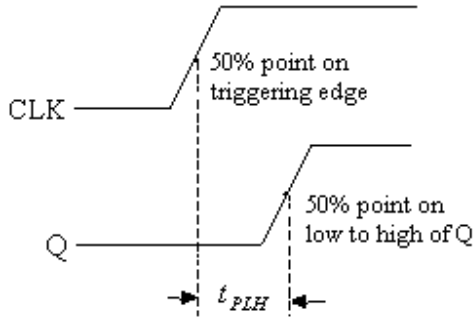


Fig. 8.33(a)

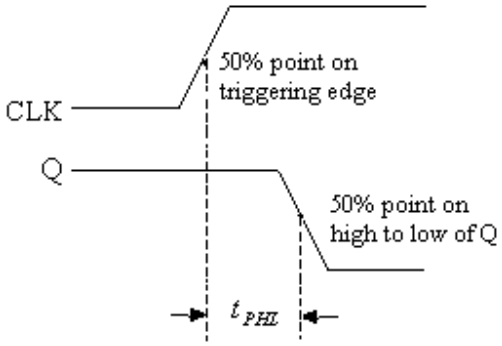


Fig. 8.33(b)

The propagation delays that occur for the asynchronous inputs applied to a flip-flop are defined as:

1. **Propagation Delay** t_{PLH} measured from the $\overline{PRE}$ input to the low to high transition of the output (refer figure 8.34a).
2. **Propagation Delay** t_{PHL} measured from the $\overline{CLR}$ input to the high to low transition of the output (refer figure 8.34b).

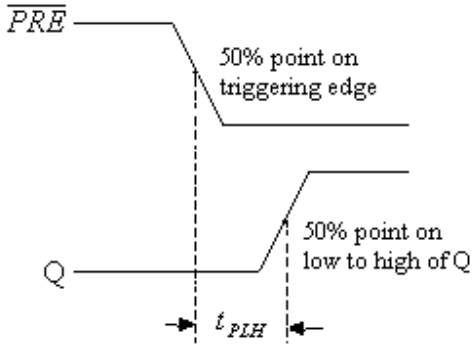


Fig. 8.34(a)

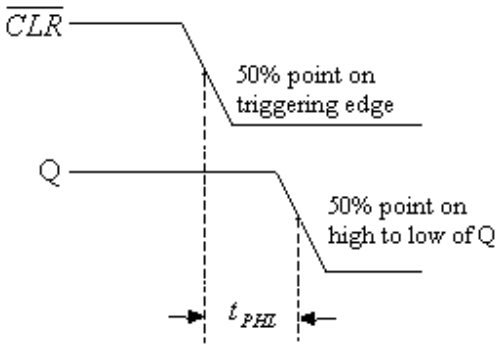


Fig. 8.34(b)

Set-up Time: It is minimum time required for the inputs to settle before the triggering edge of the clock. It is generally denoted by t_S and figure 8.35(a) illustrates the set-up time for a D flip-flop.

Hold Time: It is the time for which the data must be stable after the triggering edge of the clock. It is denoted by t_H and hold time for a D flip-flop is illustrated in figure 8.35(b).

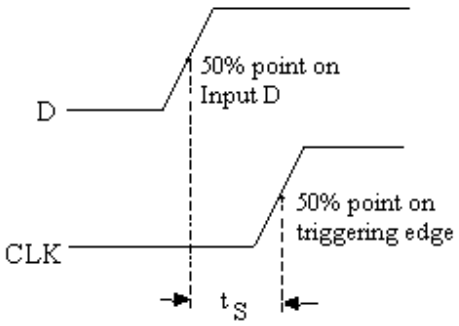


Fig. 8.35(a)

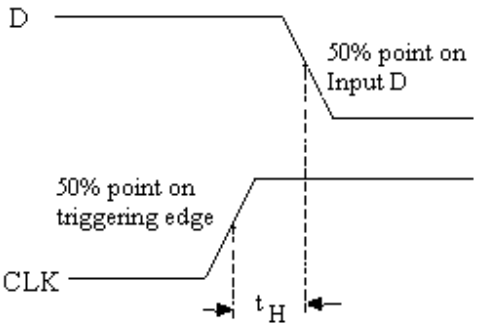


Fig. 8.35(b)

Maximum Clock Frequency: It is the maximum frequency at which a flip-flop can be reliably triggered. If the clock frequency is more than this frequency, the flip-flop will not function properly.

Pulse Widths: The minimum pulse widths for the clock and the asynchronous inputs are specified by the manufacturer of the flip-flop ICs. The minimum high time (t_{CH}) and minimum low time (t_{CL}) of the clock pulse are shown in figure 8.36(a). Similarly, the minimum low time for asynchronous inputs is given in figure 8.36(b).

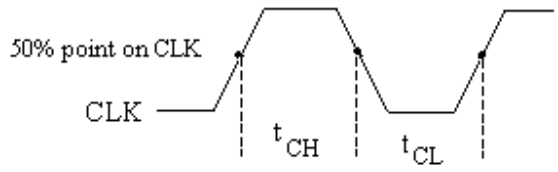


Fig. 8.36(a)

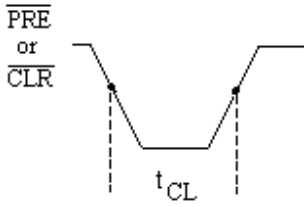


Fig. 8.36(b)

Problems:

1. Draw the circuit of R S flip-flop with NOR gates and discuss the behaviour of this circuit.
2. Discuss various types of races in asynchronous flip-flops.
3. What is the difference between asynchronous and synchronous flip-flops? Draw and explain clocked R S flip-flop with NOR latch.
4. Explain the behaviour of R S flip-flop with NAND latch.
5. Modify an asynchronous R S flip-flop so that when both the inputs R and S are 1, the flip-flop is reset.
6. Discuss the edge detector circuits for triggering the flip-flops.
7. Draw and explain D flip-flop with NAND latch.

8. How does a J K flip-flop differ from S R flip-flop in its basic operation? What are its advantages over S R flip-flop?

9. Discuss standard J K flip-flop with NOR latch and show that for inputs J K = 11, the complementation in the output will be obtained when width of the clock pulse is less than the delay in latch.

10. What do you mean by level trigger flip-flop? How does it differ from an edge trigger flip-flop?

11. Discuss the edge trigger J K flip-flop.

12. What is purpose of asynchronous inputs in flip-flop? How these inputs work?

13. What is master slave flip-flop? Discuss its working.

14. Describe the working of edge trigger T flip-flop. How a T flip-flop be used as divide-by-two device?

15. What is excitation table? How the excitation tables for R S, J K, D and T type flip-flops are formed?

16. Define the following terms related to flip-flops.

    (i) Propagation delay time (ii) Set up time
    (iii) Hold time (iv) Maximum clock frequency
    (v) Pulse width

17. Discuss the method of converting one type of flip-flop to another type. Convert J-K flip flop to D flip-flop.

18. Carry out the following conversions:

    (i) D to J K FF (ii) D to R S FF
    (iii) T to R S FF (iv) R S to D FF

19. Carry out the following conversions:

    (i) T to D FF (ii) J K to D FF
    (iii) J K to T FF (iv) R S to T FF

20. A J K flip-flop can be used as R S flip-flop, but R S flip-flop can not be used as J K flip-flop – Comment on this statement.

21. If $\overline{Q}$ output of D flip-flop is connected to its D input, verify that this circuit behaves as a T flip-flop.

22. Verify that the circuit shown in figure 8.37 behaves as J K flip-flop.

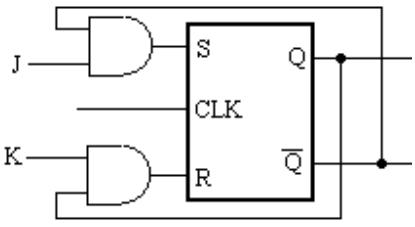


Fig. 8.37

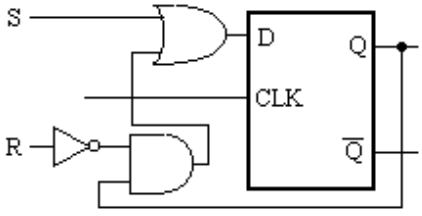


Fig. 8.38

23. Prepare the truth table for the circuit shown in figure 8.38 and show that it works as R S flip-flop.

24. Verify that the circuit shown in figure 8.39, works as T – type flip-flop.

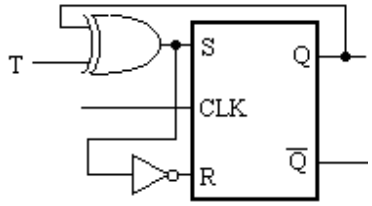


Fig. 8.39

25. A clock is connected to an S R flip-flop as shown in figure 8.40; draw the output waveform in relation to clock. Also mention the function it performs.

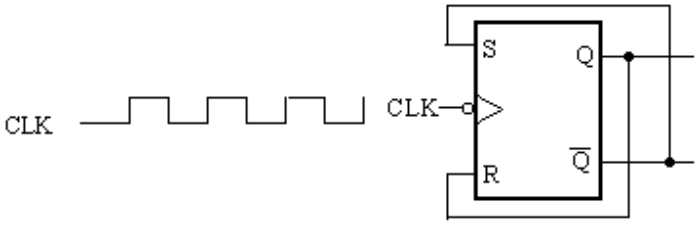


Fig. 8.40

9

Shift Registers

A register is another form of a sequential circuit that can be set to a specific state and retains until externally changed. A register is used to manipulate data for computational purposes by shifting the data in a register in either the left or right directions, and therefore, is called a shift register. This chapter will explain how to design a shift register and implement the operations Shift Left, Shift Right and bi-directional rotation of the data. A number of applications of shift register will also be discussed.

9.1 Registers: In computers or digital system a string of bits are normally stored and processed. A register is, in fact, a unit which can store a string of bits. Since a flip-flop can store a bit so for constructing a register for storing n number of bits, n flip-flops can be used. A single bit register is designed using a single D flip-flop. Consider a negative edge triggered D flip-flop as shown in figure 9.1. It is recalled from the characteristic table of D flip-flop that the flip-flop transfer the data applied to the D input to its output at the trailing edge of the clock pulse. So when logic 1 is applied to the D input of the flip-flop, then after the application of the clock pulse input 1 is transferred to its output at the trailing edge of the clock pulse. Now if the input 1 is removed, the flip-flop will continue be in the set state, retaining thereby the logic 1. Similarly logic 0 may be retained or stored in the D flip-flop.

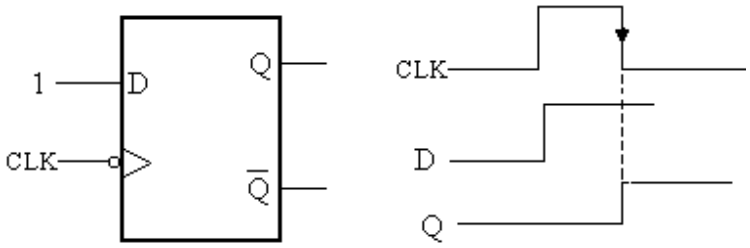


Fig.9.1

An n-bit Register is a set of n flip-flops with a common clock. This n-bit register can store n-bit word. All the flip-flops of a given register should respond to the clock pulse simultaneously. Figure 9.2 shows four bit register having a common clock. All the flip-flops can be cleared by applying 0 to $\overline{CLR}$ terminal. Data inputs are given at D inputs of the flip-flops. The four bit data is transferred to the outputs at the trailing edge of the clock pulse and the data is retained until other pulse.

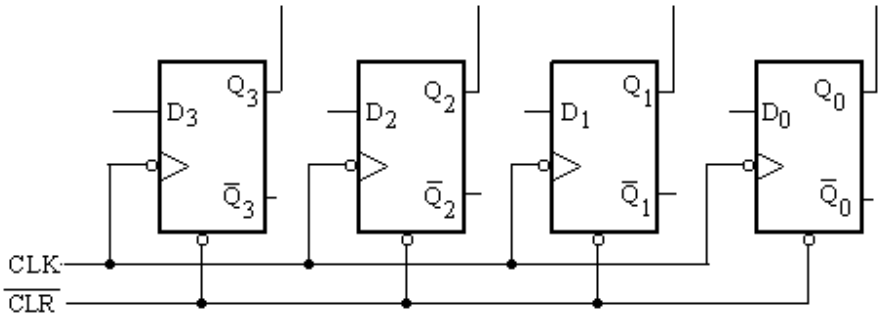


Figure 9.2

9.2 Classifications of Registers: The process of information in a register is called loading the register. Shifting the data in a register in either the left or right directions is called a shift register. Transferring information into all the flip-flops of a register can be done in two ways. One way is that all the data bits are loaded simultaneously, other data bits are loaded bit by bit (i.e. one bit at a time). Shift registers may, therefore, be classified into:

Serial Load Shift Register

Parallel Load Shift Register

The stored information in these shift registers can be transferred out of the register in parallel or in series. Based on these configurations, four combinations of loading and reading the data are possible. They are given as:

1. **Serial In Parallel Out (SIPO) Shift Register**: In this type of register, the data is loaded serially, one bit at a time; and when the output is required, the data stored in the register can be read in parallel form.
2. **Serial In Serial Out (SISO) Shift Register**: In this type of register, data can be moved serially in and out of the register one bit at a time.
3. **Parallel In Parallel Out (PIPO) Shift Register**: In this type of register the data is loaded simultaneously to all the flip-flops, and when the output is required, the data stored is read serially from the register one bit at a time under clock control.
4. **Parallel In Serial Out (PISO) Shift Register**: In this type of register the data is loaded simultaneously to all the flip-flops, and when the output is required, the data stored in the register can be read in parallel form.

9.3 Serial In Parallel Out (SIPO) Shift Register: Consider the schematic diagram of SISO shift register shown in Figure 9.3. For simplification purposes, the flip-flops chosen are D type, but they can also be JK types. The flip-flops are negative edge triggered. Firstly $\overline{CLR}$ signal is applied as 0, which clears all the flip-flops giving the Q's outputs 0. The clock pulse (CLK) is applied and at the trailing edge of the clock pulse, the input on the INPUT DATA line is transferred to the output of the first flip-flop.

Whatever the output of the first flip-flop at that time is transferred to the output of the second flip-flop and, similarly, the operation extends to the remaining flip-flops to the right until the last flip-flop. Since the data is loaded to the flip-flop serially with each clock pulse, so it is called serial loading of registers (Serial In). If the output is sensed at each one of the flip-flop outputs (each Q), the circuit is termed a parallel-out register. So such register in which data is fed serially to the input and output is taken in parallel fashion, are called serial in parallel out (SIPO) shift register.

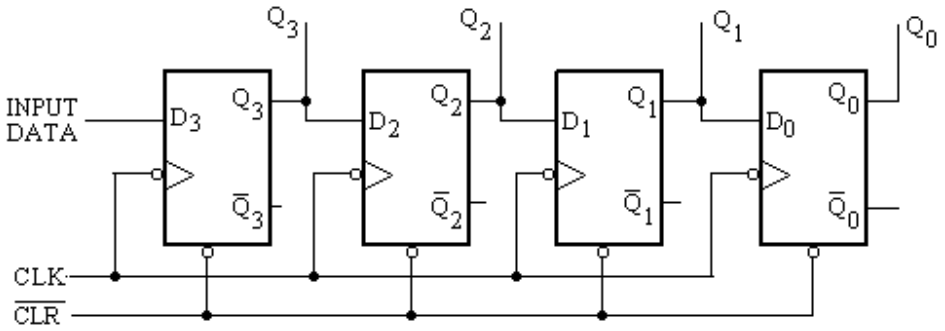


Fig. 9.3

To understand the operation of this shift register, consider the data 1111 is to be loaded serially and one wants to obtain the data at the output in parallel fashion. The $\overline{CLR}$ signal first resets all the flip-flops giving $Q_3\ Q_2\ Q_1\ Q_0$ as 0 0 0 0. Now the data is applied as 1 to input data terminal and at the trailing edge of the first clock pulse data as 1 will be shifted to the right side giving the outputs $Q_3\ Q_2\ Q_1\ Q_0$ as 1 0 0 0. Similarly at the trailing edge of the second clock pulse the outputs $Q_3\ Q_2\ Q_1\ Q_0$ will be 1 1 0 0. During the trailing edge of the third and fourth pulse the outputs will be 1 1 1 0 and 1 1 1 1 respectively. This way the data 1111 is loaded to the register serially and outputs are obtained simultaneously at $Q_3\ Q_2\ Q_1\ Q_0$.

Table 9.1

Input Data	Clock pulse No.	Outputs Q_3	Q_2	Q_1	Q_0
-	Zero	0	0	0	0
1	First	1	0	0	0
1	Second	1	1	0	0
1	Third	1	1	1	0
1	Fourth	1	1	1	1

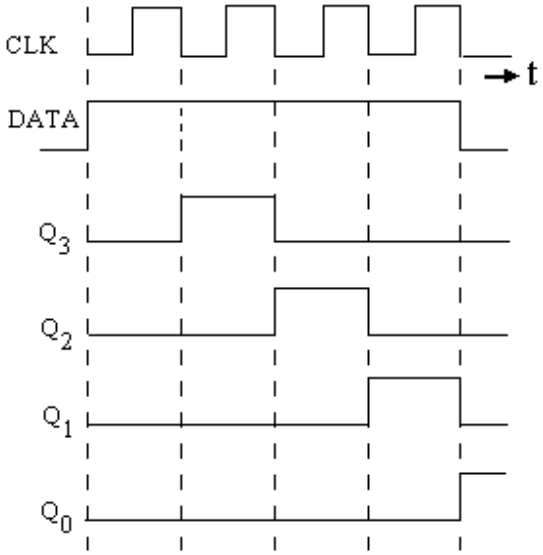


Fig. 9.4

The systematic shifting of data is given in table 9.1 and its timing diagram is shown in figure 9.4. The logic block diagram of SIPO shift register is shown in figure 9.5.

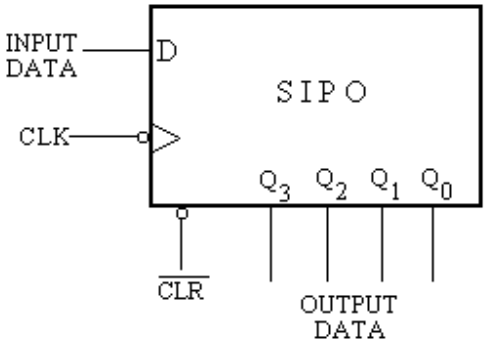


Fig. 9.5

9.4 Serial In Serial Out (SISO) Shift Register: A basic four-bit serial in serial out shift register can be constructed using four D flip-flops, as shown in figure 9.6. The operation of the circuit is as follows. The register is first cleared by applying $\overline{CLR}$ signal as 0, forcing all four outputs to zero. The input data is then applied sequentially to the input data terminal of the first flip-flop. During each clock pulse, one bit is transmitted from left to right. The data is entered into the register serially during first four clock pulses in the similar manner as has been discussed in SIPO shift register. To get the data out of the register serially, entered data must now be shifted serially and taken off at the Q_0 output. For this purpose four more clock pulses are applied and four bit required data will be available serially at the Q_0 output. Assume a data word 1101 is to be entered serially and taken at the output Q_0 bit by bit.

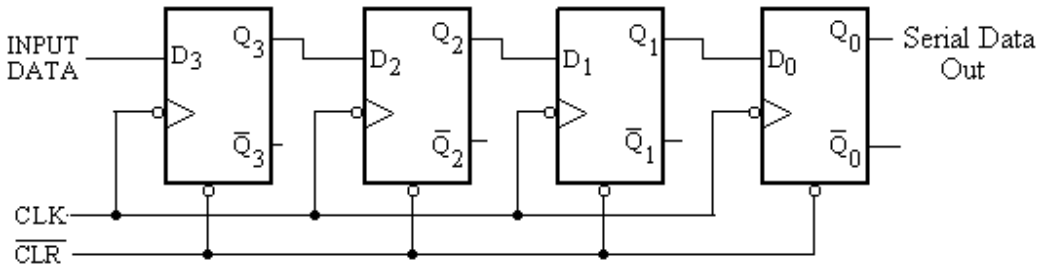


Fig. 9.6

Consider a data word 1101 is to be loaded serially and one wants to obtain this data at the output Q_0 bit by bit. Firstly all the flip-flops are cleared by applying $\overline{CLR}$ signal. Now the LSB 1 of the data is applied to the input data terminal and then during the negative edge of the clock pulse data 1 is shifted to Q3 and giving the outputs Q_3 Q_2 Q_1 Q_0 as 1 0 0 0. The second LSB 0 of the data is applied to the input terminal and at the trailing edge of the clock pulse outputs Q_3 Q_2 Q_1 Q_0 will be available as 0 1 0 0. Similarly at the trailing edge of the third and fourth clock with input bits as 1 1, the outputs Q_3 Q_2 Q_1 Q_0 will be 1 0 1 0 and 1 1 0 1 respectively. So at the 4^{th} clock pulse LSB 1 is available at Q_0 output. During 5^{th}, 6^{th} and 7^{th} pulse with input data as 0 0 0, the serial output

terminal Q_0 will deliver second LSB, third LSB and MSB of the data. All the flip-flops may also be cleared by applying one more clock pulse with 0 as the input data. The systematic shifting of data is illustrated in table 9.12. The logic block diagram of SISO shift register is shown in figure 9.7.

Table 9.2

Input data	Clock pulse No.	Outputs Q_3	Q_2	Q_1	Q_0
-	Zero	0	0	0	0
1	One	1	0	0	0
0	Two	0	1	0	0
1	Three	1	0	1	0
1	Four	1	1	0	1
0	Five	0	1	1	0
0	Six	0	0	1	1
0	Seven	0	0	0	1
0	Eight	0	0	0	0

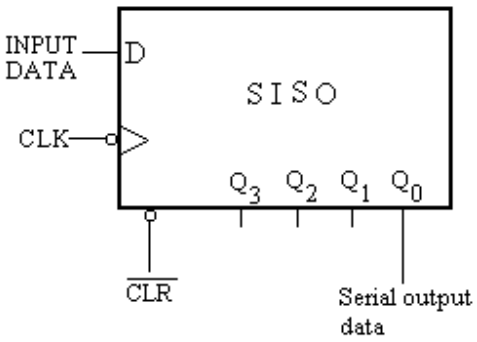


Fig. 9.7

9.5 Parallel In Parallel Out (PIPO) Shift Register: For parallel in - parallel out shift registers, all data bits appear on the parallel outputs immediately following the simultaneous entry of the data bits. The four-bit parallel in - parallel out shift register constructed by D flip-flops is shown in figure 9.8.

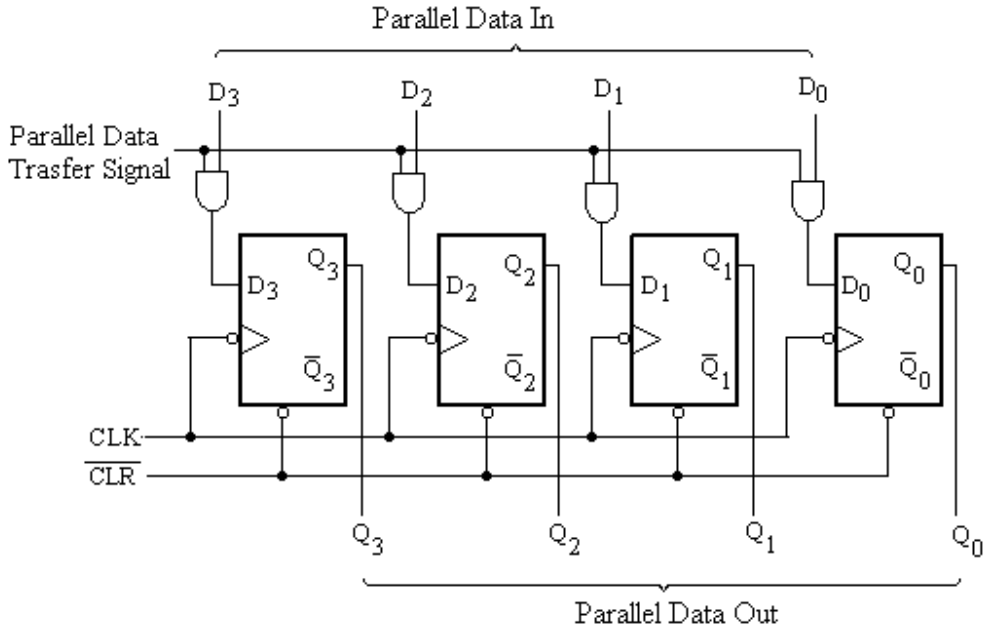


Fig. 9.8

The D's are the parallel inputs and the Q's are the parallel outputs. Once the parallel transfer signal is high all the AND gates will be enabled and the data bits gets connected to their respective flip-flop. Now after the application of clock pulse all the data at the D inputs appear at the corresponding Q outputs simultaneously. The logic block diagram of PIPO shift register is shown in figure 9.9.

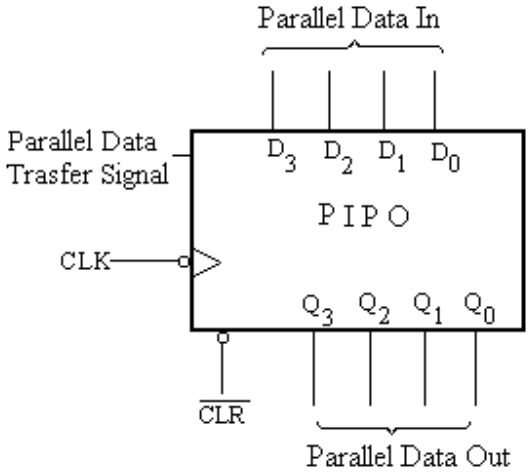


Fig. 9.9

9.6 Parallel In Serial Out (PISO) Shift Register: A four-bit parallel in - serial out shift register is shown in figure 9.10. The circuit uses D flip-flops and NAND gates for entering data (i.e. writing) to the register. D3, D2, D1 and D0 are the parallel inputs, where D3 is the most significant bit and D0 is the least significant bit. To write data in, the mode control line ($\overline{WRITE}$ / *SHIFT*) is applied low and the data is clocked in. The data can be shifted when the mode control line is high as SHIFT is active high. The register performs right shift operation on the application of a clock pulses.

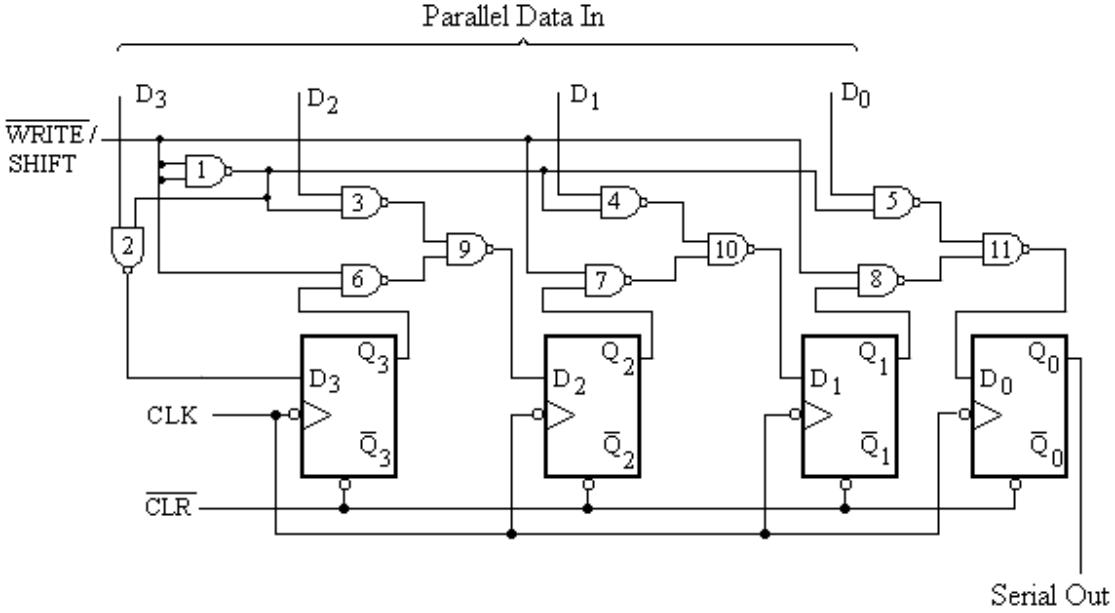


Fig. 9.10

It can be understood in more detail, the data to be entered to the register is applied to the parallel data input terminal and mode control line ($\overline{WRITE}/SHIFT$) is made low, it allows all the four bits of the data word to be entered in parallel into the register. In this condition NAND gates 1 through 5 will be enabled which allow each data bit to be entered into D inputs of their respective flip-flop. After the application of clock pulse the data applied to the inputs of flip-flops will appear at the output of the respective flip-flop. The data is said to be stored in the register.

After the data is stored into the register, mode control line ($\overline{WRITE}/SHIFT$) is made high, NAND gates 6 through 8 will be enabled which will force the data to be shifted from their present state to next state after the application of clock pulse. Further shifting is possible with next consecutive pulses. Table 9.3 illustrates the entering of the data (say 1001) into the register and it's shifting with clock pulses. The logic block diagram of PISO shift register is shown in figure 9.11.

Table 9. 3

$\overline{WRITE}$ / SHIFT	Clock pulse No.	Outputs Q_3	Q_2	Q_1	Q_0	Serial out Q_0
Low	One	1	0	0	1	1
High	Two	0	1	0	0	0
High	Three	0	0	1	0	0
High	Four	0	0	0	1	1

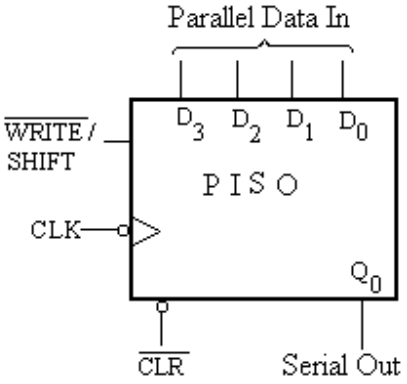


Fig. 9.11

9.7 Bidirectional Shift Register: The registers discussed so far involved only right shift operations. Each right shift operation has the effect of successively dividing the binary number by two. If the operation is reversed (left shift), this has the effect of multiplying the number by two. With suitable gating arrangement a serial shift register can perform both operations. A bidirectional, or reversible, shift register is one in which the data can be shift either left or right. A four-bit bidirectional shift register using D flip-flops is shown in figure 9.12.

Here a set of NAND gates are configured as OR gates to select data inputs from the right or left adjacent bi-stables, as selected by the $SHR/\overline{SHL}$ control line. When this control line is high data is moved or shifted to the right side, and if it is low data will be moved to the left side. Hence this shift register is also called as Left Right or Bidirectional shift register.

The operation of this circuit may be explained as follows. In the beginning $\overline{CLR}$ signal is made low which clear all the flip-flops. When $SHR/\overline{SHL}$ control line is high, one can understand from the logic of the gates connected in this circuit that the output Q of all the flip-flops gets connected to the D input of the following flip-flop. The data bit gets connected to D input of first flip-flop. Now after the application clock pulse to the CLK terminal, data bits are shifted one place to the right. Further occurrence of the nest

pulses will shift data in right. However, when the $SHR/\overline{SHL}$ control line is low, the configuration of logic gates makes the data bit to connect to D input of 4th flip-flop also the output of each flip-flop is passed through to the D input of the preceding flip-flop. After the application clock pulse to the CLK terminal, data bits are shifted one place to the left. In this mode this circuit works as the left shift register.

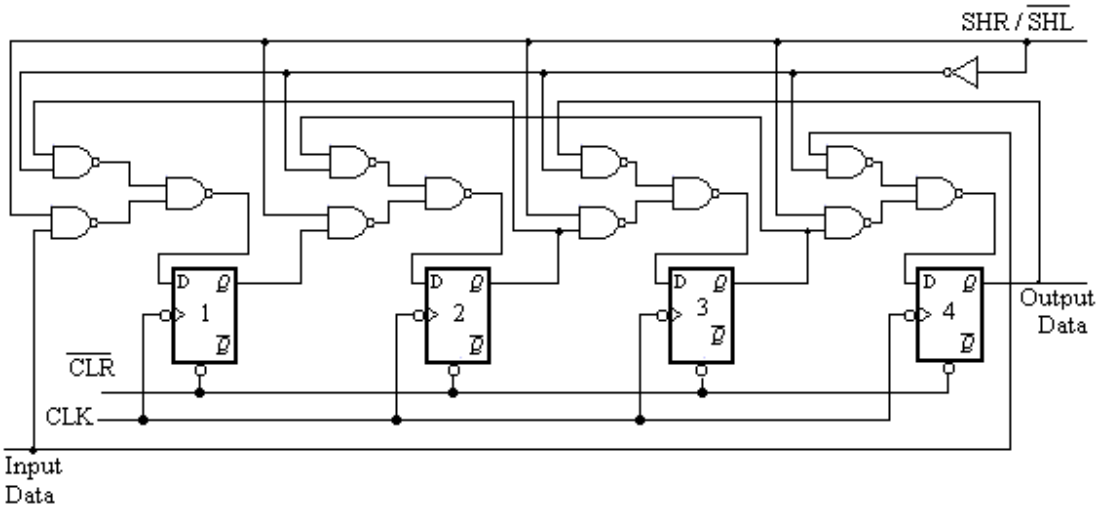


Fig. 9.12

9.8 Universal Shift Register: Consider the schematic diagram shown in Figure 9.13. It has facilities for serial loading (serial in) and serial output and parallel loading (parallel in) and parallel output, and, additionally, it has shift-left and shift-right facilities. Therefore, this kind of shift register, which can operate in all the four different modes discussed in preceding sections and also has the facility of bi-directional shifting of data, is called a universal shift register. Figure 9.13 shows the logic diagram of 4-bit universal shift register.

It has four D flip-flops and the associated NAND gates, which makes it possible to shift the data to the right or left direction. The mode control inputs S_0 and S_1 enable the required operating mode of the register. The different modes are shown in table 9.4. From this table it is clear that when both these mode control inputs are 00 or 11, no shifting occurs. When both are 00, the content of the register will have its previous value (i.e. no change) and when both are 11, the input data D_3 D_2 D_1 D_0 are loaded in parallel fashion in the register. When control inputs S_1 S_0 are 01 or 10, the data is shifted right or left respectively after the application of clock pulse. The asynchronous input $\overline{CLR}$ is used to clear or reset the register. This is an active low input.

Table 9.4

S_1	S_0	Operating Mode
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

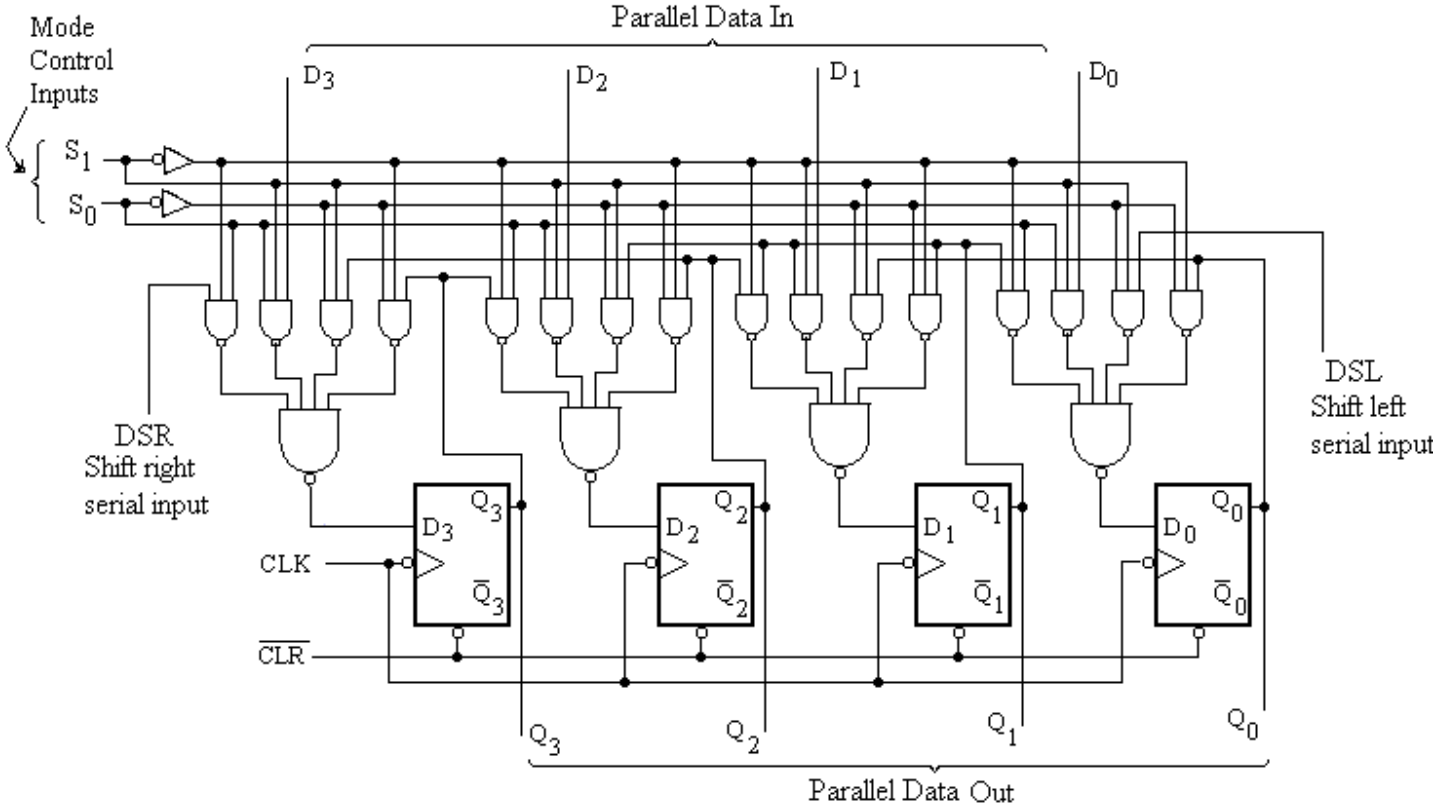

Parallel Data In
Mode
Control
Inputs
D_3
D_2
D_1
D_0
S_1
S_0
DSR
Shift right
serial input
DSL
Shift left
serial input
Q_3
Q_2
Q_1
Q_0
D_3
D_2
D_1
D_0
CLK
$\overline{Q}_3$
$\overline{Q}_2$
$\overline{Q}_1$
$\overline{Q}_0$
$\overline{CLR}$
Q_3
Q_2
Q_1
Q_0
Parallel Data Out

Fig. 9.13

9.9 Cyclic Shift Registers: In a regular shift register, a given number can be shifted to the left or right when a shift pulse is applied. Bits shifted out one end of the register may be lost. However, in a cyclic shift register, bits shifted out one end are shifted back in the other end. The cyclic shift registers also known as shift register counters are constructed by modifying serial in serial out (SISO) shift registers. There are two following types of cyclic shift registers:

1. **Ring Counter**
2. **Johnson Counter or Twisted Ring Counter**

These cyclic shift registers will be discussed in the following sections.

9.9.1 Ring Counter: The ring counter can be obtained from a serial in serial out (SISO) shift register by connecting the Q0 output of the last flip-flop to the D input of the first flip-flop. The ring counter, constructed using the D flip-flops is shown in figure 9.14.

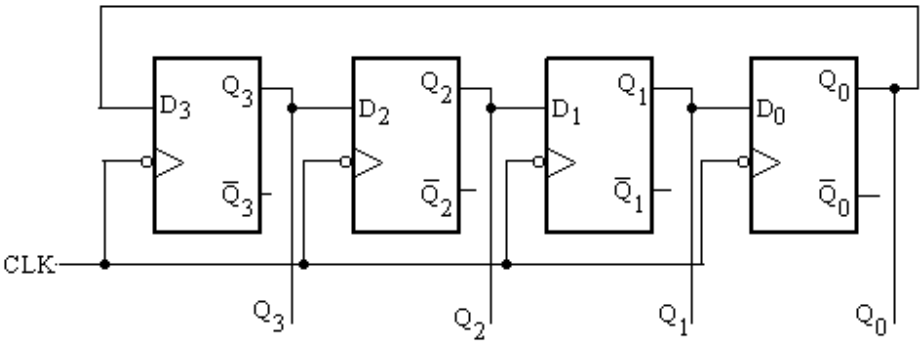


Figure 9.14

In this counter a single 1 is stored in the register and it is made to circulate in the register after the application of clock pulsed. Initially, the output Q_3 is made 1 by presetting this flip-flop and other flip-flops are reset so that the outputs Q_3 Q_2 Q_1 Q_0 are 1000. Since the output Q of each flip-flop is connected to the D input of the next stage, so the contents of each register are shifted to the right by one bit after the application of clock pulses. After the first pulse, the content of the shift register is 0100. After a second pulse, the state of the register is 0010, then 0001, and the register returns to the initial state of 1000 at the fourth pulse. In this shift register at a time one flip-flop gives the output 1. The 1 can be used to switch on a sequence of machines, one after another, whose operating time is controlled by the length of the clock pulses. A ring counter will have as many different codes as there are flip-flops since the only difference between outputs of the ring code is the place where the 1 is. The cyclic Shift registers are used in calculators. On the display of some pocket calculators, it can be seen that the numbers shift over as each new number is keyed in. This is due to the shift register.

The systematic shifting of data is given in table 9.5 and its timing diagram is shown in figure 9.15.

Table 9. 5

Clock pulse No.	Outputs			
	Q_3	Q_2	Q_1	Q_0
Zero	1	0	0	0
One	0	1	0	0
Two	0	0	1	0
Three	0	0	0	1
Four	1	0	0	0

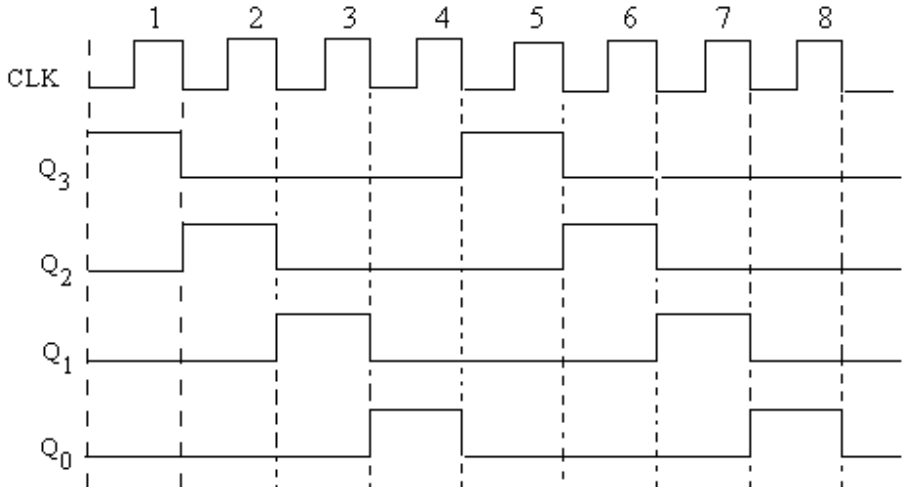


Fig. 9.15

9.9.2 Johnson Counter or Twisted Ring Counter: The basic difference between the Johnson counter and ring counter is that in the Johnson counter the complement of the output of the last flip flop is connected back to the D input of the first flip flop rather than the output itself. This feedback arrangement produces a unique sequence of states. The four bit Johnson counter has a total of 8 states. In general, an n stage Johnson counter will produce a modulus of 2n, (n will also be the number of stages of the counter).

The schematic diagram of 4 bit Johnson counter also known as twisted ring counter is shown in figure 9.16. The working of this circuit may be explained as follows:

Initially, it is considered that the counter is set to 0 (Q_3 Q_2 Q_1 Q_0 are 0000) as the first pulse occurs. At the occurrence of the second clock pulse first flip flop changes its output state from 0 to 1 (Q_3 becomes 1). Now as Q_1 = 1 so the second flip flop will also changes the state from 0 to 1 at the next pulse. The flip flops 1, 3 and 4 remain unchanged. The similar changes occur at the next pulses as given in the table 9.5. The wave form of this counter is shown in figure 9.17.

The twisted ring counters are very useful especially when a sequence of events takes place one after the other without reverting to initial value.

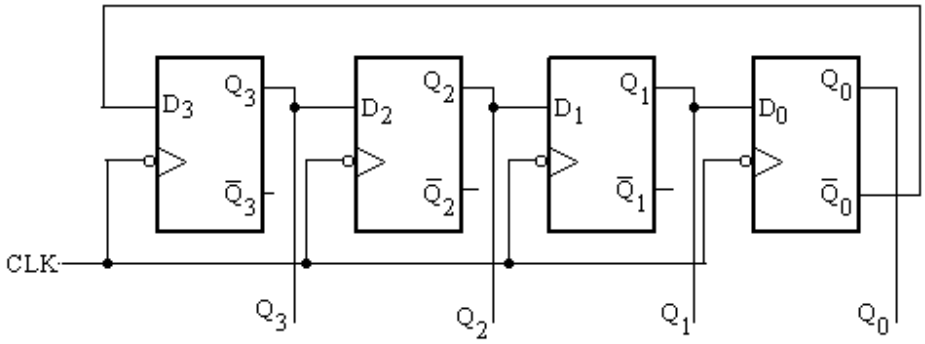


Fig. 9.15

Table 9. 6

Clock pulse No.	Outputs			
	Q_3	Q_2	Q_1	Q_0
One	0	0	0	0
Two	1	0	0	0
Three	1	1	0	0
Four	1	1	1	0
Five	1	1	1	1
Six	0	0	0	0
Seven	1	0	0	0

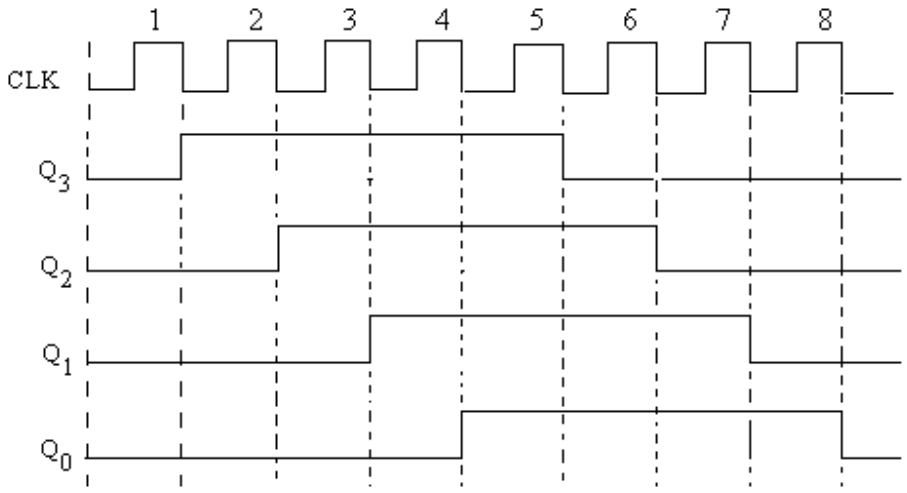


Fig. 9.17

9.10 Shift Register IC Details: Shift register IC's currently available are given in the table 9.7:

Table 9.7

IC No.	Description	No. of bits
7491, 7491A	Serial-in Serial out	8 - bit
7494	Parallel-in Serial-out	4 - bit
7495	Serial/Parallel-in Parallel out (left shift, right shift)	4 - bit
7496	Parallel-in/Parallel-out Serial-in /Serial-out	5 - bit
7499	Bi – directional (Universal)	4 - bit
74164	Serial-in Parallel-out	8 - bit
74165	Serial/Parallel-in Serial-out	8 - bit
74166	Serial/Parallel-in Serail-out	8 - bit
74178, 74179	Bi – directional (Universal)	4 - bit
74194	Bi – directional (Universal)	4 - bit
74195	Serial/Parallel-in Parallel-out	4 - bit
74198	Bi – directional (Universal)	8 - bit
74199	Serial/Parallel-in Parallel-out	8 - bit
74295A	Tri-state Serial/Parallel-in Parallel-out bi-directional	4 - bit
74395	Tri-state cascaded Serial/Parallel-in Serial/Parallel-out	4 - bit

Brief details of a few IC's mentioned above are given below:

IC 7491: It is an 8-bit serial in serial out shift register. The logic diagram of this IC has 8 S – R flip flops used as D flip flops as usual. There are two gated data input lines, A and B, for serial data entry. When the bit is entered through the input A, input B must be high and vice versa. The serial data output is QH and its complement is $\overline{QH}$. The pin diagram of IC7491 is shown in figure 9.18.

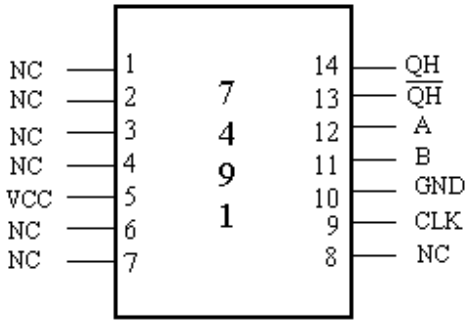


Fig. 9.18

IC 74164 : The 74164 is an eight bit serial- in parallel-out shift register. The pin diagram of this IC is shown in figure9.19. This device has two gated serial inputs, A and B, and a clear $\overline{CLR}$ input with active low. The parallel outputs (8 bits) are QA through QH. When the bit is entered through the input A, input B must be high and vice versa.

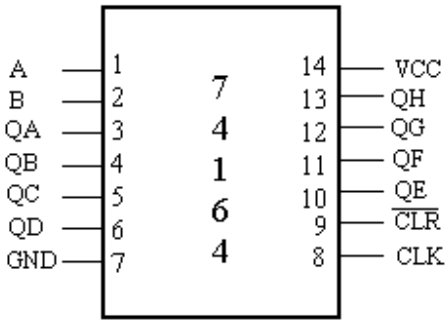


Fig. 9.19

IC 74194: The 74194 is a bidirectional shift register in IC form. The pin diagram of this IC is given in figure9.20. Parallel loading, which is synchronous with a positive transition of the clock, is accomplished by applying the four bits of data to the parallel inputs and a high to S_0 and S_1 inputs.

When S_0 is high and S_1 is low, shift right operation is performed with the positive edge of the clock pulse. Serial data in this mode are entered at the shift right serial input (SR) terminal.

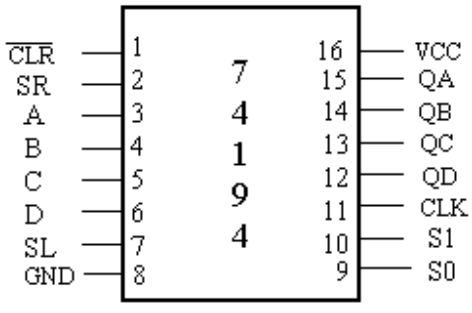


Fig. 9.20

When S_0 is low and S_1 is high, data bits shift left and the data is entered at the shift left (SL) terminal.

IC 74195: The IC 74195 is a 4 bit Serial/Parallel-in Parallel-out shift register. Pin diagram of this IC is shown in figure 3.17. When the SH/LD input is low, the data on the parallel inputs are entered synchronously on the positive edge of the clock pulse. When this input is high, stored data will shift right QA through QD synchronously with the clock pulse. Since J and $\overline{K}$ are connected together, it can be used as the serial data inputs to the first stage of the register QA; QD can be used for serial output data.

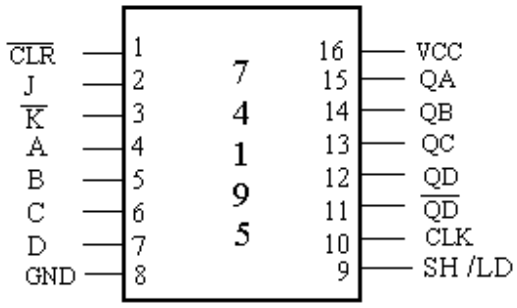


Fig.9.21

9.11 Applications of Shift Registers: Shift registers are primarily used for temporary storage of data and bit manipulations, but these find numerous applications in digital systems. A few important applications of shift registers will be discussed here.

9.11.1 Serial Adder: The most important application of shift registers is the serial adder. Figure 9.22 shows the block diagram of the serial adder, in which the augend bits and addend bits are loaded in parallel fashion to the register A and register B respectively. These bits (augend and addend) are shifted in the right direction so that they get added (bit by bit) in full adder circuit. Initially carry bit is set to zero using a flip-flop. The output CARRY of the full adder is transferred to a D flip-flop, which gets added to the next bit. The SUM bit of the full adder is transferred to the register A (augend register).

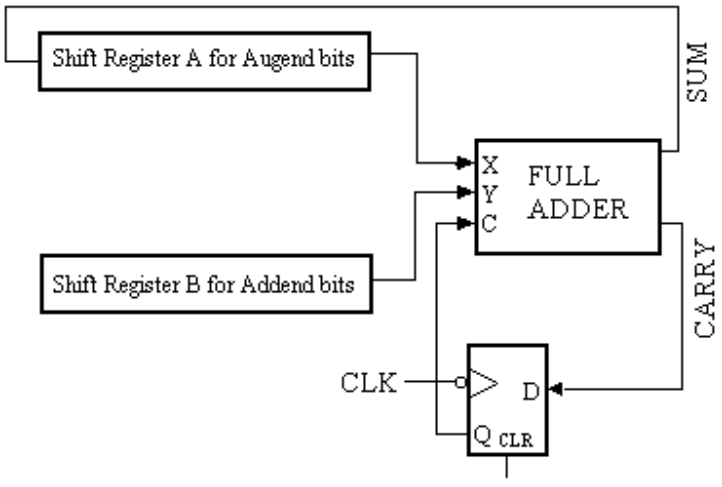


Fig. 9.22

The complete schematic diagram as well as the control signals necessary to implement the serial addition is shown in figure 9.23. The shift register along with the full adder and modulo 5 counter/ decoder makes it possible to add two data words each of

4 bits. The full adder adds the augend and addend bits serially and the result is placed in the position occupied by the augend.

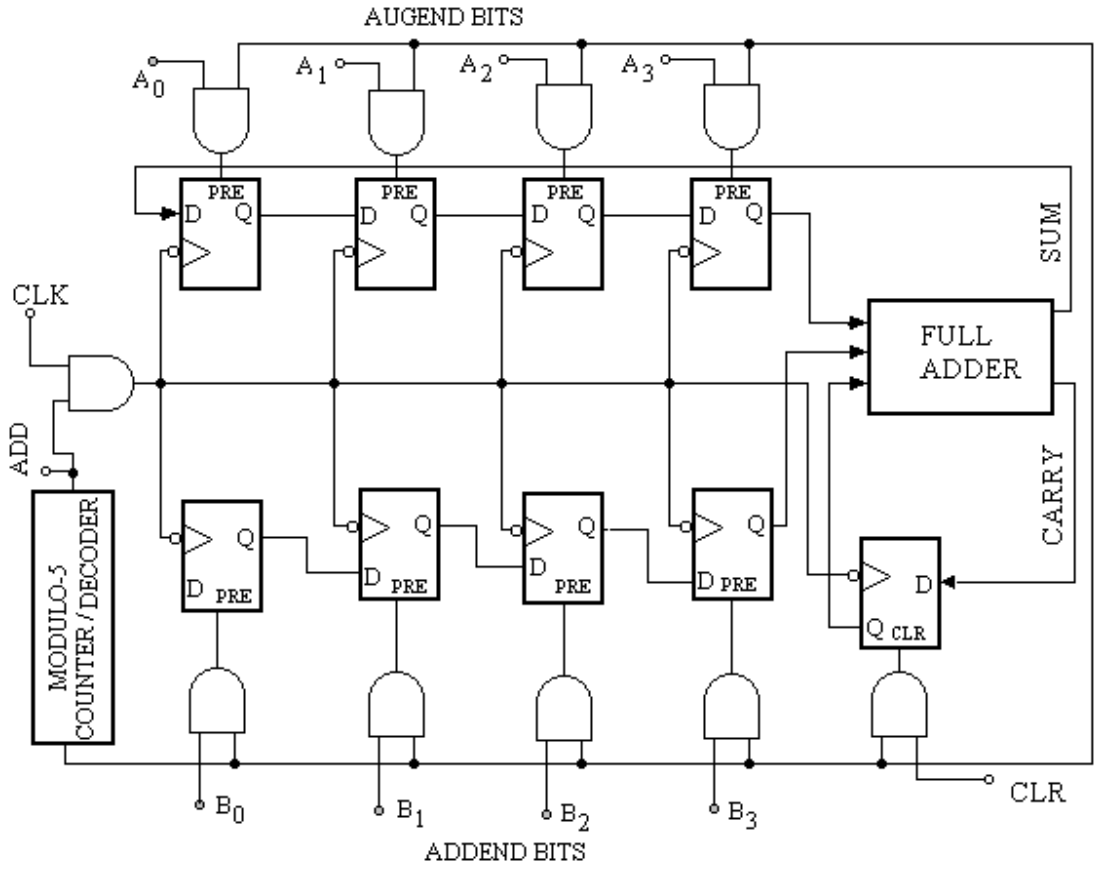


Fig. 9.23

The circuit diagram contains four D - flip flops to store the augend bits in parallel fashion and other 4 D - flip flops for addend bits. One more D – flip flop is connected to the carry bit of the full adder to store and shift it to the augend positions (augend bit register). Modulo 5 counter / decoder delivers the control signal to the circuit as shown in figure 9.24.

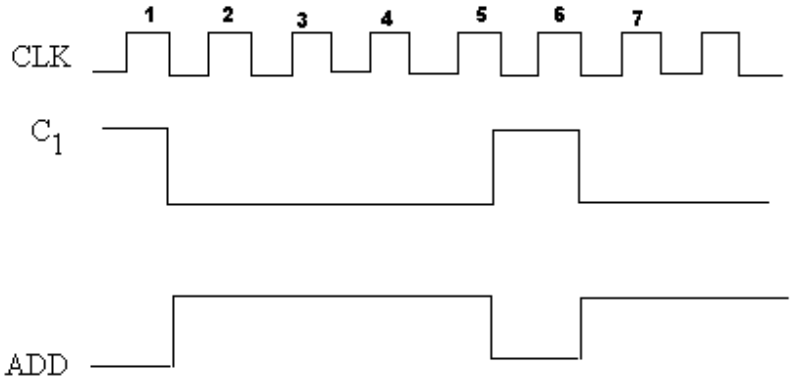


Fig.9.24

The working of this serial adder may be explained as follows:

During the first clock pulse, carry flip flop is cleared and the augend and addend bits ($A_3A_2A_1A_0$ and $B_3B_2B_1B_0$) are entered in their respective flip flops. At the second clock pulse, add signal is set and it remains set till the completion of the addition operation (i.e. for four clock pulses).

When the add signal is 1, the contents of the augend and addend registers are shifted right and are added bit by bit in the full adder circuit. The carry, if any, sets the carry flip flop at the trailing edge of the clock pulse. The content of the carry flip flop gets added during the next pulse by the full adder. Each bit of the sum re-enters the augend register. Overflow, if any, will be available in the carry flip flop.

After addition is complete the sum $S_3S_2S_1S_0$ are stored in augend flip flop and carry bit generated is stored in carry flip flop.

9.11.2 Parity Generator cum Checker: Another application to generate and check the parity bit of 4 bit number will be discussed. The schematic diagram for the same is shown in figure 9.25.

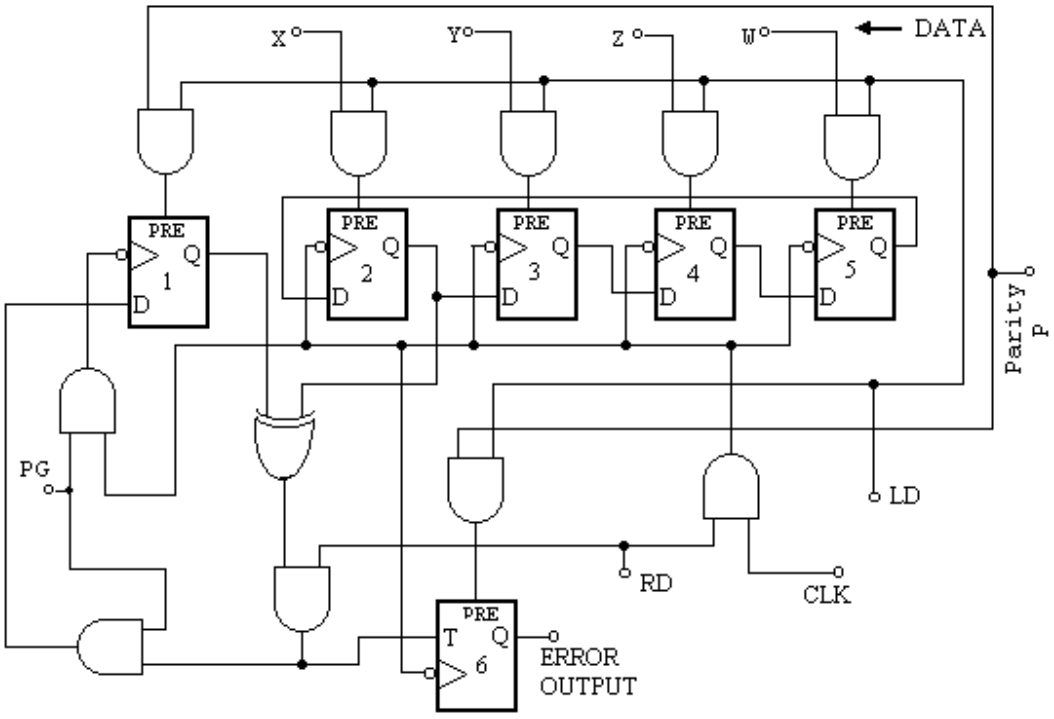


Fig. 9.25

This circuit contains five flip flops, four to store the 4 bit number and to store the parity bit. The control signal for the parity generator cum checker is shown in figure9.26.

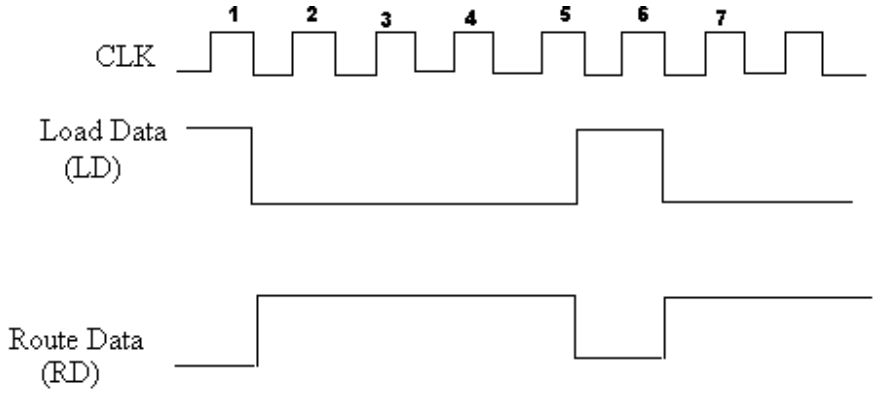


Fig. 9.26

Parity Checker: To use the circuit as parity checker, the parity generator signal (PG) is set to 0. The working of this circuit is explained as follows:

Load the data signal (LD) is set to 1 during the first clock pulse. Parity bit P and the data bits (XYZW) are loaded in the parity and shift register flip flops through the preset (PRE) terminal of the flip flops. The error flip flop (6) is also set to parity bit P. The route data signal (RD) is kept high for four clock pulses. During this time contents (XYZW) are shifted 4 times. As parity generator signal is zero, the parity flip flop is unaffected.

If parity bit P is 0, then error flip flop (6) is toggled by the output of the excusive-OR gate for each data bit having a value 1 i.e. error flip flop(6) toggles as many times as equal to the number of 1's in data XYZW. So after the 4 clock pulses, if the output of error flip flop 1 then parity check fails, further if 0 is there at the output of error flip flop (6) then parity check is correct.

If parity bit P is 1, then error flip flop (6) is toggled by the output of the excusive-OR gate for each data bit having a value 0 i.e. error flip flop toggles as many times as equal to the number of 0's in data XYZW. So after the 4 clock pulses, if the output of error flip flop 1 then parity check fails, further if 0 is there at the output of error flip flop(6) then parity check is correct.

Parity Generator: For parity generator, parity generator signal (PG) is set to 1 and P is reset to 0. Now when route data signal (RD) is 1, data bits (XYZW) are entered into the shift register flip flops. The output of the exclusive OR gate is feedback to the P flip flop. If the odd number of 1's is there in the data (XYZW) then output Q of flip-flop 1 is set to 1 at the end of fifth clock pulse. If on the other hand even number of 1's is there in the data then output Q of flip-flop 1 is set to 0. Thus correct parity bit is generated and stored in the flip flop 1.

9.11.3 Time Delay: Serial In Serial Out (SISO) shift register can be used to introduce time delay $\Delta\tau$ in digital signals from input to output given by:

$$\Delta\tau = N.T_{CLK} = N.\frac{1}{f_{CLK}}$$

Where N is the number of flip-flops or number of stages,

T_{CLK} is the time period of the clock,

f_{CLK} is the frequency of the clock.

From this equation it is clear that the delay can be controlled by changing the value of N. Consider a serial in serial out (SISO) shift register of N stages as shown in figure 9.27(a), to which a clock of 500 KHz frequency (2 μ S time period) is applied. The data input connected to the serial input of the shift register, will be obtained at the serial out terminal after as delay of 16 μ S. The timing diagram for the delay is shown in figure 9.27(b).

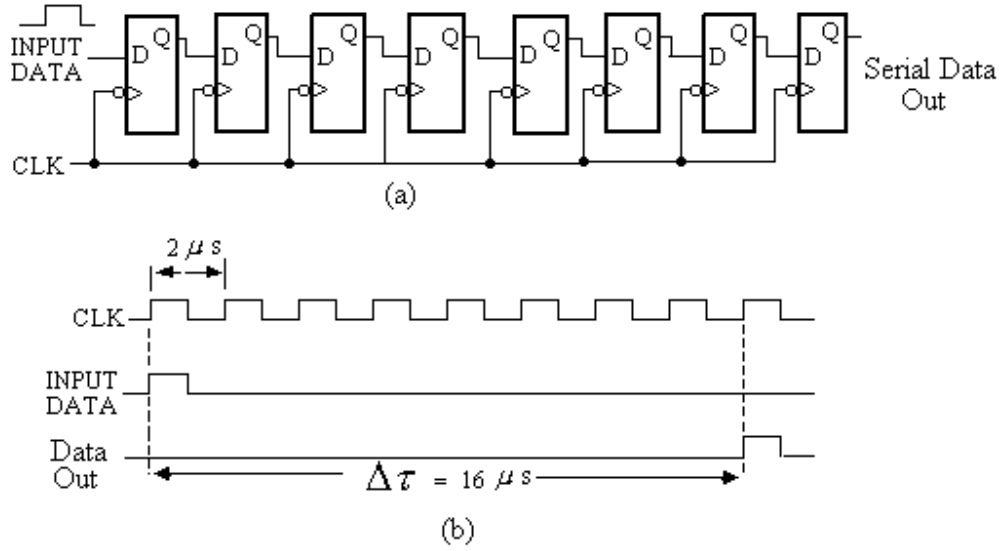


Fig. 9.27

9.11.4 Data Conversion: The data conversion means data available in serial form is to convert in parallel form or vice-versa. The conversion from serial to parallel form is possible with Serial in parallel out (SIPO) shift register and the conversion of data from parallel form to serial form is possible with parallel in serial out (PISO) shift register.

9.11.5 Sequence Generator: An important use of shift register is a sequence generator,

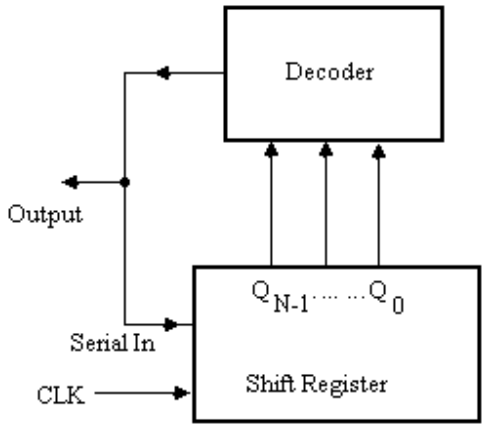


Fig. 9.28

which generates a prescribed sequence of binary bits in synchronization with the clock. This system is also referred to as a word generator or code generator. The basic structure of sequence generator is shown in figure 9.28. The parallel outputs of the shift register are connected to the inputs of some combinational circuit, whose output may be connected to the serial input of the shift register.

As an example, consider the sequence generator that can generate the binary sequence 101011….. In the required sequence there are 6 bits for which three state shift register is needed since $p \le 2^N$, where N is the number flip-flops in the shift register and p is the number of bits in the required sequence. Table 9.8 illustrates the six combinations of three bits which are required to generate the given sequence. The output Q_2 of the shift register should be the required sequence of the generator and the outputs Q_1 and Q_0 are the same sequence delayed by one and two clock pulses respectively.

Table 9.8

Clock pulses	Q_2	Q_1	Q_0
1	1	1	1
2	0	1	1
3	1	0	1
4	0	1	0
5	1	0	1
6	1	1	0

The states in the table are not distinct as row 3rd and 5th are same. Now consider N = 4 and a table is prepared in the similar manner as shown in table 9.9. The last column in the table shows the output (W) which gives the required sequence (Q_3) shifted after clock pulses. The expression for W is obtained from the K – map (figure 9.29) of table 9.8 as:

$$W = Q_2 \cdot Q_0 + \overline{Q}_2 \cdot \overline{Q}_0$$

Table 9.9

Clock pulses	Q_3	Q_2	Q_1	Q_0	Output W
1	1	1	1	0	0
2	0	1	1	1	1
3	1	0	1	1	0
4	0	1	0	1	1
5	1	0	1	0	1
6	1	1	0	1	1

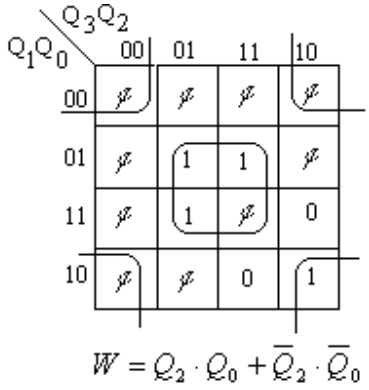


Fig. 9.29

The circuit for the sequence generator may be drawn as shown in figure 9.30.

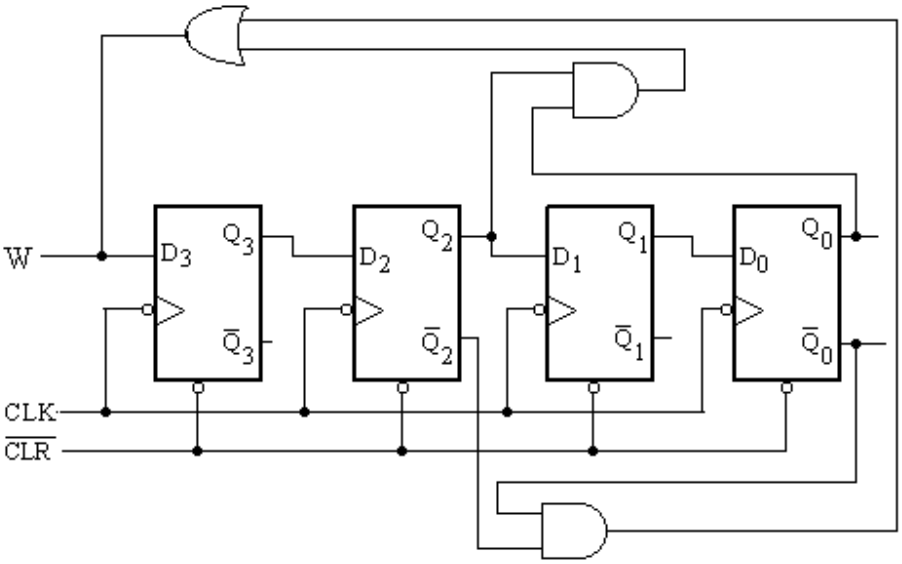


Fig. 9.30

Problems:

1. What is register? How can a flip-flop be used to store a bit.
2. Mention the classifications of registers. Describe the working of serial in serial out shift register.
3. Describe the working of parallel in parallel out shift register. Explain how number can be shifted in or out from this register.
4. What is the difference between a parallel in parallel out shift register and parallel in serial out shift register? Discuss the working of parallel in serial out shift register.
5. Explain the operation of serial in parallel out shift register.
6. Draw the circuit of four bit bi-directional shift register, explain its working.
7. A 4-bit serial in parallel out right shift register initially contains 0101. The data 1011 is to be entered. After three clock pulses, what is the data at outputs of the register?
8. What do you understand by the cyclic shift register? Explain the operation of ring counter using timing diagram .
9. What is the difference between a ring counter and twisted ring counter? Discuss the operation of four bit Johnson counter using timing diagram and the sequence.
10. If the initial state of a 6-bit ring counter is 101101, what is the state of the counter after the third clock pulse? Explain with trimming diagram.

11. What is the main advantage of a universal shift register? Draw and explain the circuit of four bit universal shift register.
12. Name any two applications of shift register. Explain one application in detail.
13. Explain how can a shift registers be used in serial adder circuit. Draw and explain the circuit of four bit serial adder circuit.
14. Discuss the use of shift registers for generating and checking the parity bit.
15. Discuss the use of shift register in sequence generator.
16. Design a sequence generator to generate a sequence ….10110 …
17. Design a sequence generator to generate a sequence ….110011 …

_____

10

Counters

Counters are the important building block of digital systems. These are used to count the pulses. The clock pulses occur at regular and known intervals, so a counter can be used to measure time and consequently frequency and time period. So counters are sequential logic circuits that proceed through a well-defined sequence of states after application of clock pulses. The counters are constructed using flip-flops and logic gates. Counters are classified into two following broad categories.

1. Asynchronous or ripple counter
2. Synchronous counter

In asynchronous counters, external clock is applied to the first flip-flop and other successive flip-flops are triggered by the outputs of the preceding flip-flops. However, in synchronous counters all the flip-flops are triggered simultaneously by the external clock pulses. In this chapter the design of these counters with up and down counting sequences will be discussed.

10.1 ASYNCHRONOUS COUNTERS

It is well known that if a clock of frequency *f* is applied to the clock input of a negative triggered T flip-flop, whose T input is connected to high (logic 1), it toggles at the trailing edge of each pulse (figure 10.1). The frequency

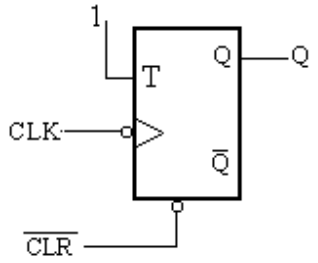


Fig. 10.1(a)

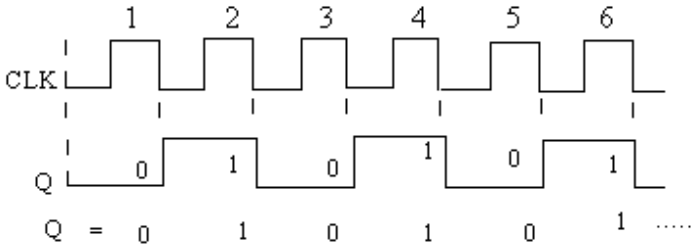


Fig. 10.1(b)

of the output will be *f* / 2. If two flip-flops are connected in series as shown in figure 10.2, the output frequency obtained will be the division of input frequency by a factor of 4. This will have the four unique states 00, 01, 10, 11 (shown in timing diagram of figure 10.2a). The frequency division is basically a counter. This circuit is called as two bit asynchronous or ripple counter.

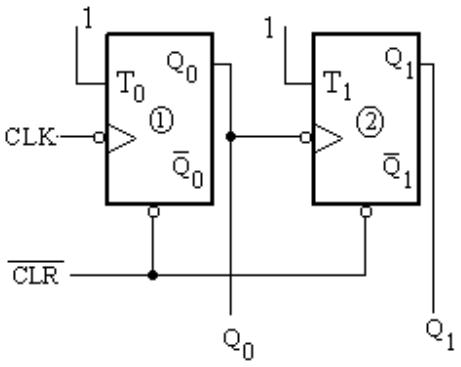


Fig. 10.2(a)

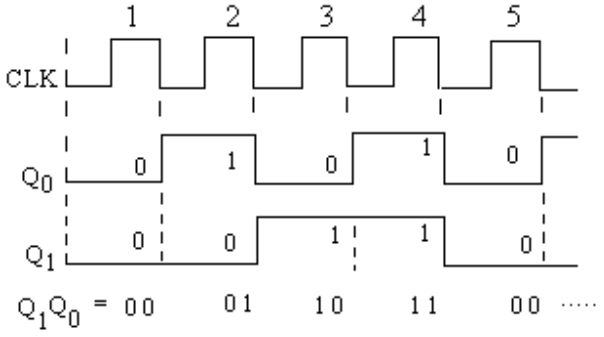


Fig. 10.2(b)

In this wary any number of flip-flops may be connected in series. A counter with *n* number of flip-flops will have 2^n unique states which will count in natural sequence and the counter is called a modulus (short mod) 2^n counter. The modulus of a counter represents the total number of states through which the counter can move. The binary counter with one flip-flop will have two states and the counter is called as Mod-2 or divide-by-2 counter (ref. fig. 10.1). The counter with 2 flip-flops will have 4 (as $2^n = 4$) states (including zero) and the counter is called as Mod-4 or divide-by-4 counter (ref. fig.10.2). It will have 00, 01, 10 and 11…..states. Similarly, Mod-8, Mod-16…counters may be discussed.

In these asynchronous or ripple counters all the flip-flops are not synchronously controlled by the same clock pulse. As discussed above, in asynchronous counters external clock is applied to the first flip-flop and other successive flip-flops are triggered by the outputs of the preceding flip-flops. Further, it is well known that when an input pulse is applied to a flip-flop, it gives an output after some time delay (propagation delay). Consider a counter with two flip-flops connected in series and if propagation delay of each flip-flop is 20 nsec., then the output of the second flip-flop will be obtained after a time delay of 40 nsec. Since each flip-flop is toggled by the changing state of the preceding flip-flop, the delay accumulates with the number of flip-flops. That is this delay ripples through the flip-flops and becomes quite appreciable when the number of flip-flops is increased. This delay may become comparable to the period of the clock (or more than the clock period). In this condition there is a possibility that the first flip-flop responds to the new clock pulse before the previous pulse has effected transition of the last flip-flop, this may lead the skipping of certain count which is undesirable. So the asynchronous counter becomes too slow for carrying out the counting, if the clock frequency is large enough or the number of flip-flops are increased.

10.2 ASYNCHRONOUS BINARY (MOD-16) COUNTER

Asynchronous binary counter or Mod-16 counter will have 16 unique states and needs 4 flip-flops to design this circuit as $2^4 = 16$. A series combination of four T flip-flops is shown in figure 10.3, in which the output of first flip-flop is connected to the clock input of the second; the output of the second is connected to the clock input of the third, and so on. The clock is applied to the clock input of first T flip-flop and T inputs of

all the flip-flops are connected to high (logic 1). The clear terminal of all the flip-flops are connected together to $\overline{CLR}$, which resets the counter when logic 0 is applied to $\overline{CLR}$.

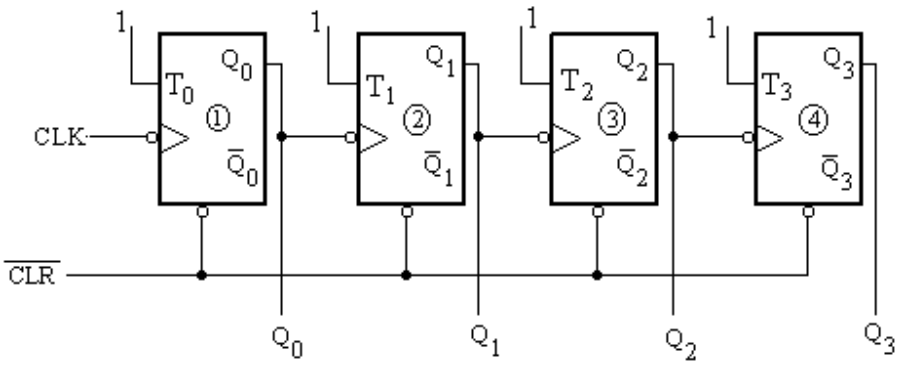


Fig. 10.3

It is well known that a T flip-flop toggles at the trailing edge of the clock pulse, so first flip-flop changes it state each time the clock input goes from high to low. The subsequent flip-flops change state when their inputs change from 1 to 0. The waveforms at the input and outputs of all the flip-flops are shown in figure 10.4, while the states of flip-flops corresponding to input clock pulses are given in table 10.1.

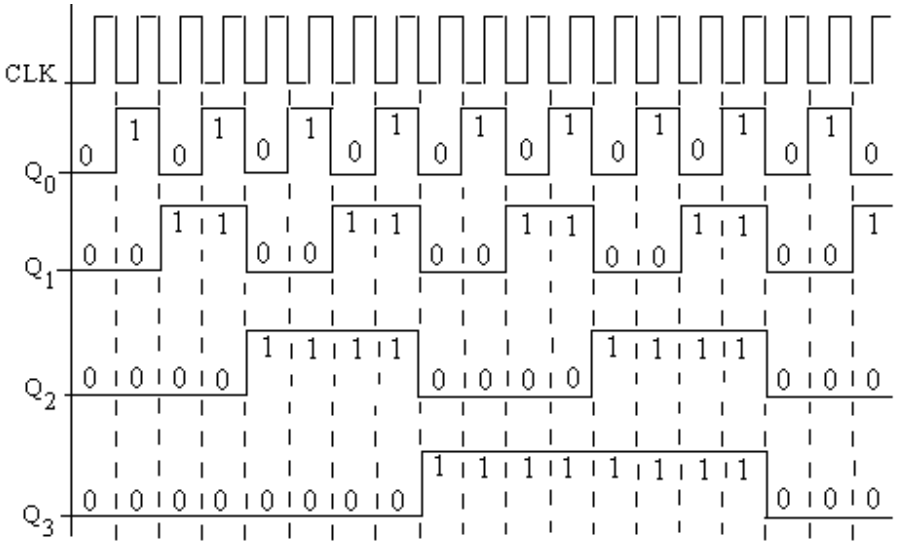


Fig. 10.4

In the asynchronous counters one can use J K flip-flops in place of T flip-flops. The J and K inputs of J K flip-flops may be tide together which works as T input. The ripple mod -16 counter designed using J K flip-flops is shown in figure 10.5. The figure 10.6 shows this counter designed using D flip-flops, in which $\overline{Q}$ outputs of all the flip-flops are connected to their D inputs. The D flip-flops connected like this work as T flip-flops.

Table 10.1

After clock pulses No.	Outputs or Counts			
	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0

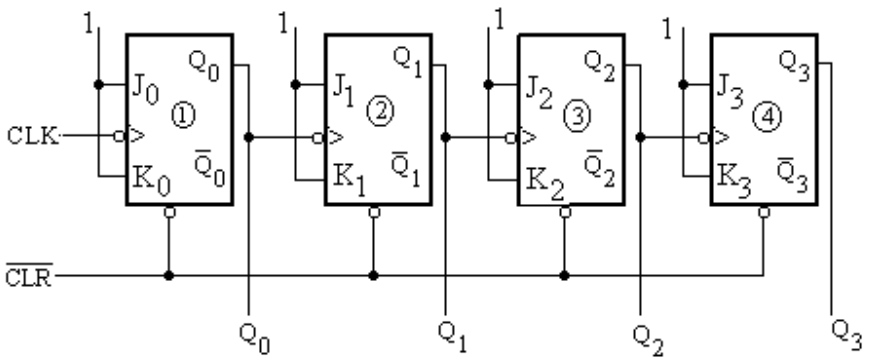


Fig. 10.5

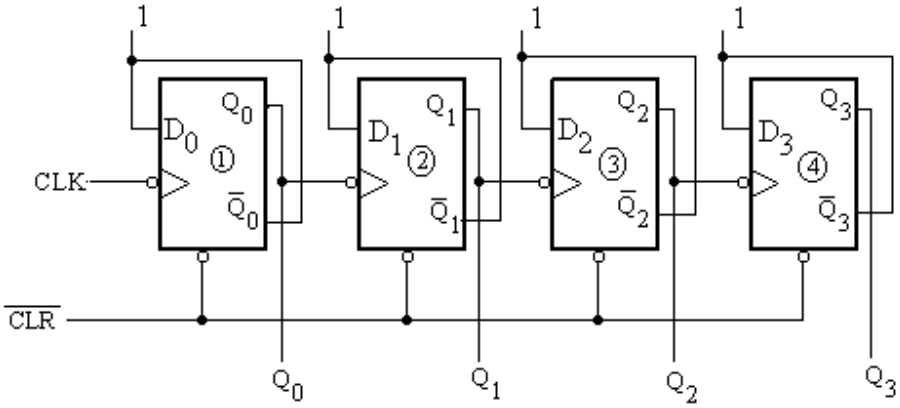


Fig. 10.6

Example 10.1: Design a mod-8 asynchronous counter using T flip-flops.

Solution: A mod-8 counter also called divide-by-8 counter can be designed using 3 T flip-flops as $2^3 = 8$. This counter will have 8 possible states starting from 000 to 111. At the eighth clock pulse the counter is reset and counting is started from the beginning with the next pulse. The three T flip-flops are connected to be connected in series as shown in figure 10.7. The waveforms at the input and outputs of all the flip-flops are shown in figure 10.8, while the states of flip-flops corresponding to input clock pulses are given in table 10.2.

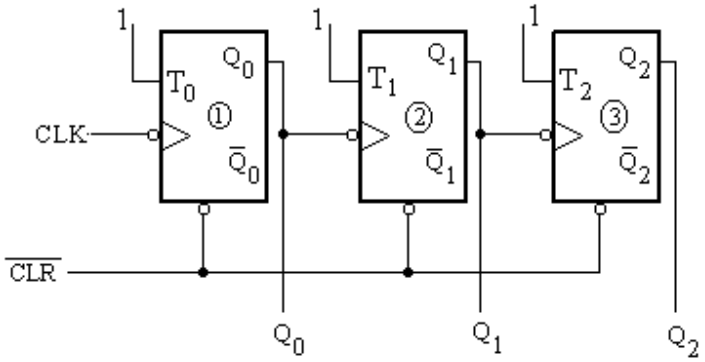


Fig.10.7

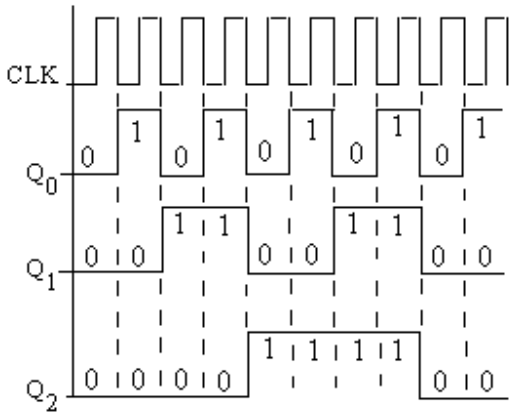


Table 10.2

After clock pulses No.	Outputs or Counts		
	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Fig.10.8

10.3 ASYNCHRONOUS DOWN COUNTERS

So far the counters which count in the forward direction are considered. Some times it is also desirable to have the digital counters which count in the backward or reverse direction like ….. 3 2 1 0 3… (or 11 10 01 00 11….). In the up counter or forward counters, the external clock is applied to the first flip-flop and other successive flip-flops are triggered by the outputs of the preceding flip-flops. In the down counter the external clock is applied to the clock terminal of the first flip-flop as in the case of the up counter, the other successive flip-flops are, however, triggered by the $\bar{Q}$ outputs of the preceding flip-flop. The outputs are taken at Q's outputs. The figure 10.9(a) shows the logic diagram of asynchronous Mod-4 down counter. The output waveforms of this counter are shown in figure 10.9(b), which are in the down sequence.

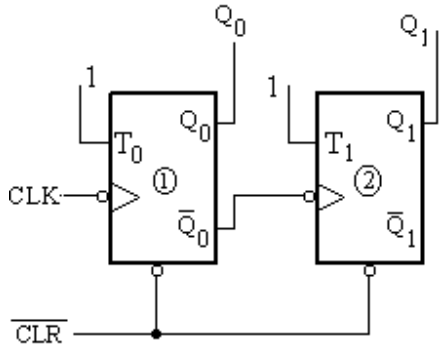


Fig. 10.9(a)

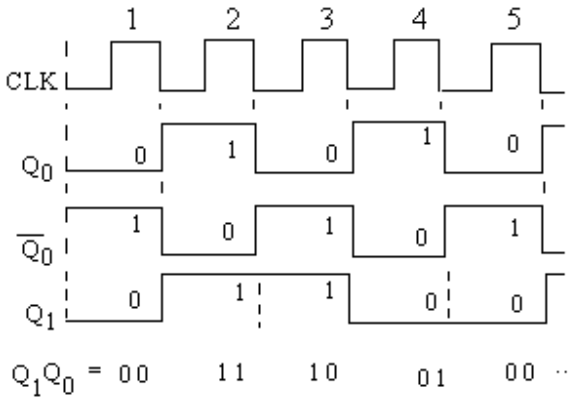


Fig.10.9 (b)

10.4 ASYNCHRONOUS MOD-16 DOWN COUNTER

The asynchronous mod-16 down counter will count in the down sequence having 16 distinct states. It needs 4 T flip-flops as $2^4 = 16$. The down sequence will be 15, 14, 13, 12 ……1, 0, 15 ……It may be designed in the similar fashion as mod-4 down counter. Figure 10.10 shows the down counter, in which external clock pulse is applied to T_0 input of first flip-flop. The $\overline{Q}_0$ output of 1st flip-flop is connected to T_1 input of 2nd flip-flop, similarly $\overline{Q}_1$ output to T_2 and $\overline{Q}_2$ output to T_3.

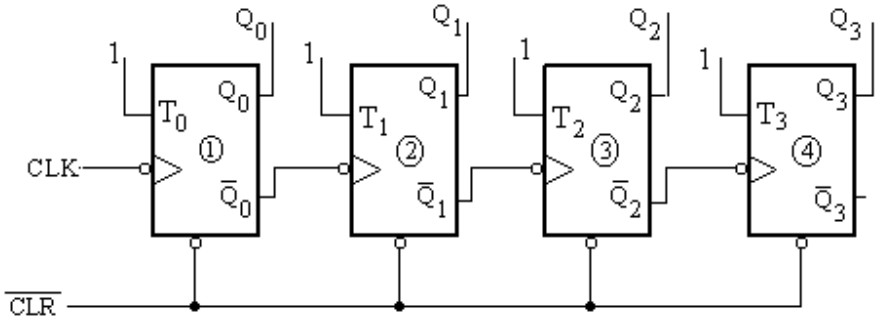


Fig. 10.10

This circuit may also be designed with J K flip-flops, as shown in figure 10.11.

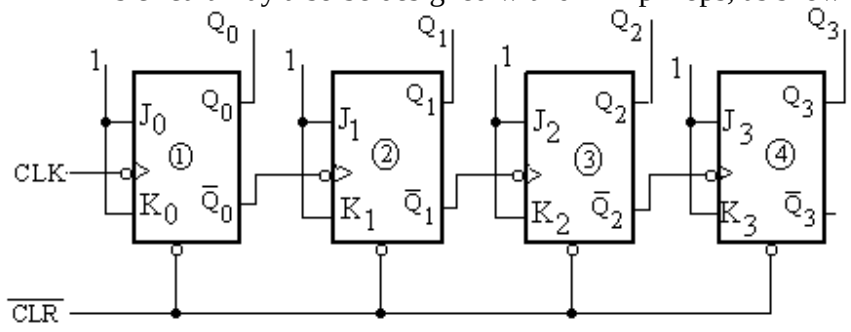


Fig. 10.11

The output waveforms are shown in figure 10.12, which correspond to down or reverse counting. The Q_0 output will toggle at the trailing edge of the clock pulse, and Q_1 through Q_3 will change their outputs at the trailing edge of $\overline{Q}$'s outputs of the preceding flip-flops.

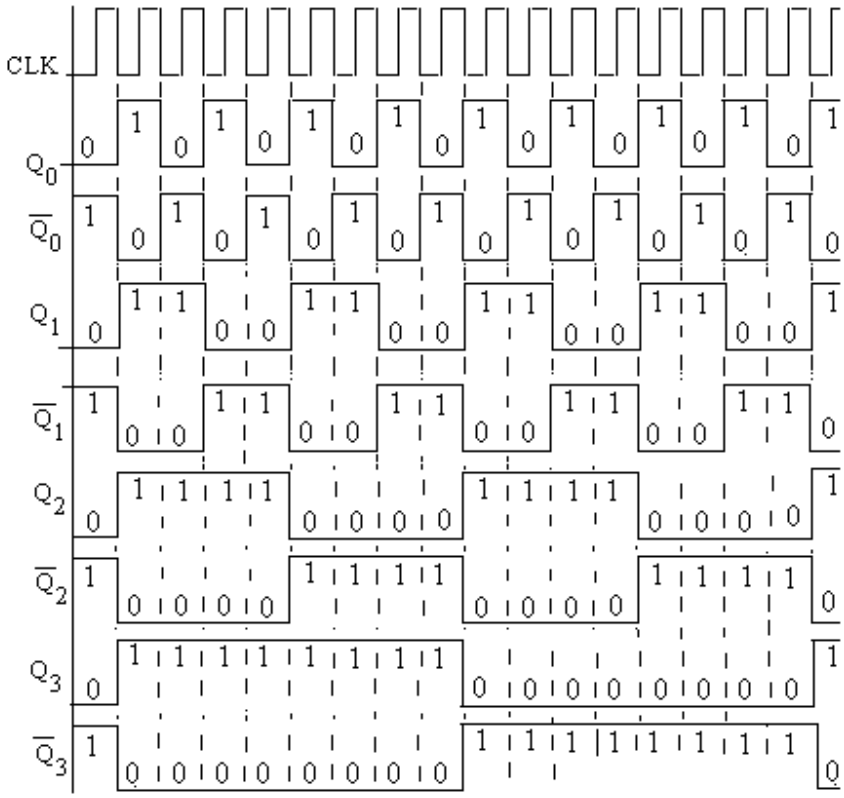


Fig. 10.12

The states of the flip-flop outputs in the reverse sequence are given in table 10.3.

Table 10. 3

After clock pulses No.	Outputs or Counts			
	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1
10	0	1	1	0
11	0	1	0	1
12	0	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
16	0	0	0	0

10.5 ASYNCHRONOUS MOD-16 UP / DOWN COUNTER

A counter can work both as Up counter and down counter (up / down counter) if AND- OR control gates are used for connecting the Q's and $\overline{Q}$'s outputs of the preceding stage to the input of the next stage. Figure 10.13 shows the circuit for a mod-16 up / down asynchronous counter. Here the T flip-flops are used along with AND-OR control gates.

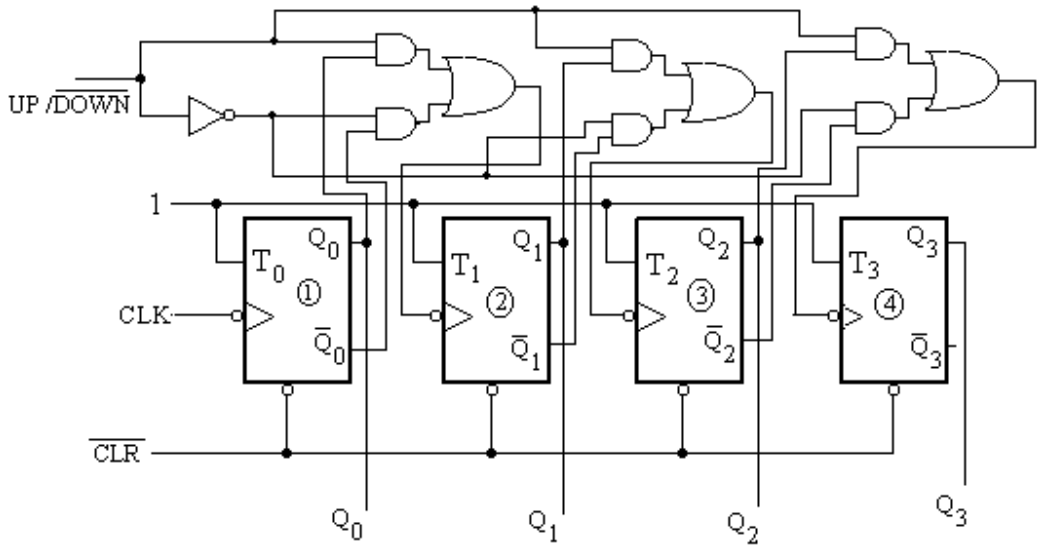


Fig. 10.13

Fig. 10.13

In this circuit when control input $UP / \overline{DOWN}$ is high, the counter works as up or forward counter as the outputs Q_0, Q_1, Q_2 gets connected to the T inputs of the next stages. Again when $UP / \overline{DOWN}$ input is low, the counter works as down counter as the complemented outputs gets connected to the T inputs of the next stages.

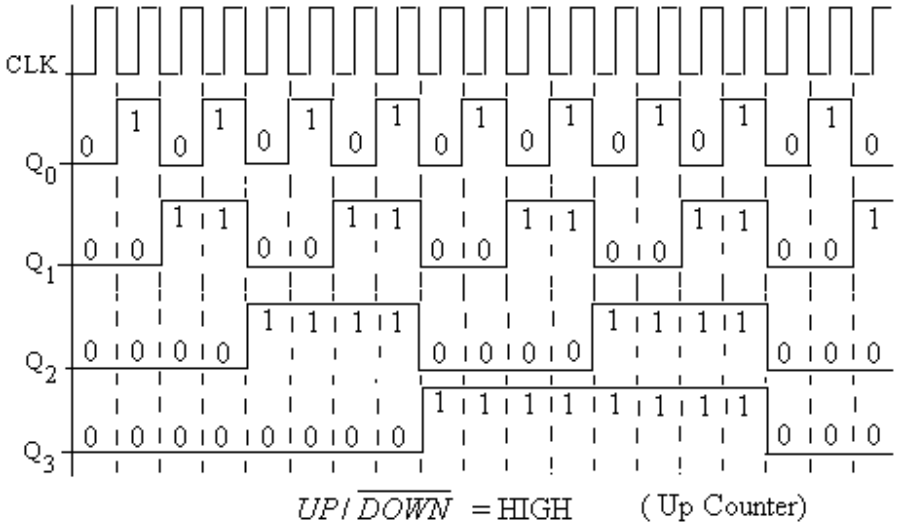


Fig. 10.14(a)

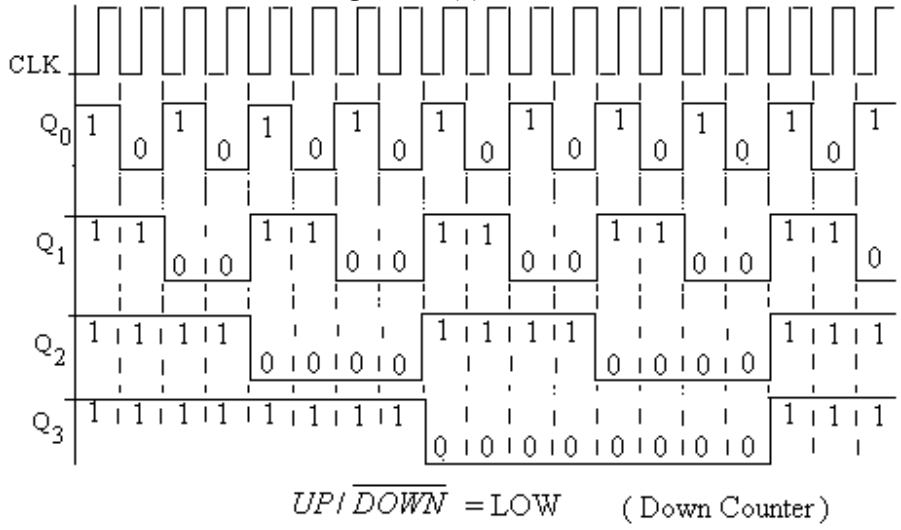


Fig. 10.14 (b)

Figures 10.14 (a) and (b) show the waveforms taken at Q's outputs for Up and down counter respectively. The counting sequences are shown in Table 10.4 for working the circuit as UP or down counter.

Table 10.4

UP COUNTER When UP / $\overline{DOWN}$ = High					DOWN COUNTER When UP / $\overline{DOWN}$ = Low			
Outputs or Counts				After clock pulses No.	Outputs or Counts			
Q_3	Q_2	Q_1	Q_0		Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1	0
0	0	1	0	2	1	1	0	1
0	0	1	1	3	1	1	0	0
0	1	0	0	4	1	0	1	1
0	1	0	1	5	1	0	1	0
0	1	1	0	6	1	0	0	1
0	1	1	1	7	1	0	0	0
1	0	0	0	8	0	1	1	1
1	0	0	1	9	0	1	1	0
1	0	1	0	10	0	1	0	1
1	0	1	1	11	0	1	0	0
1	1	0	0	12	0	0	1	1
1	1	0	1	13	0	0	1	0
1	1	1	0	14	0	0	0	1
1	1	1	1	15	0	0	0	0
0	0	0	0	16	1	1	1	1

10.6 OTHER ASYNCHRONOUS COUNTERS

It is quite often desired to have counters which can count through modulo other than 2, 4, 8, 16 etc., that is not a power of 2, for example Mod-3, Mod-5, Mod-6, Mod-10 etc. These are obtained from the binary counters of higher modulo by providing a feedback to $\overline{CLR}$ which resets all the flip-flops after the desired count. Combinational logic circuits are used for the reset pulse.

10.6.1 Asynchronous Decade Counter

The decade counter also called as divide-by-10 counter can have 10 distinct states. It can count from 0 to 9 and then reset to count again in the same sequence. Four T flip-flops are needed to design this counter. Table 10.5 shows the counting sequence for this counter. It is clear from this table that the counter should reset when $Q_3\ Q_2\ Q_1\ Q_0$ becomes 1 0 1 0 i.e. a low pulse should be generated when $Q_3\ Q_1 = 11$. So Q_3 and Q_1 outputs should be applied to a NAND gate, whose output will be low when $Q_3\ Q_1 = 11$. This low pulse should be applied to $\overline{CLR}$ terminals of all the flip-flops.

Table 10. 5

After clock pulses No.	Outputs or Counts			
	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
	0	0	0	0

Figure 10.15 shows the logic diagram of the ripple decade counter. At the trailing edge of the 10^{th} pulse, the counter temporarily goes to 1010 state, but immediately resets to 0000, because of the feedback provided by the output of the NAND gate.

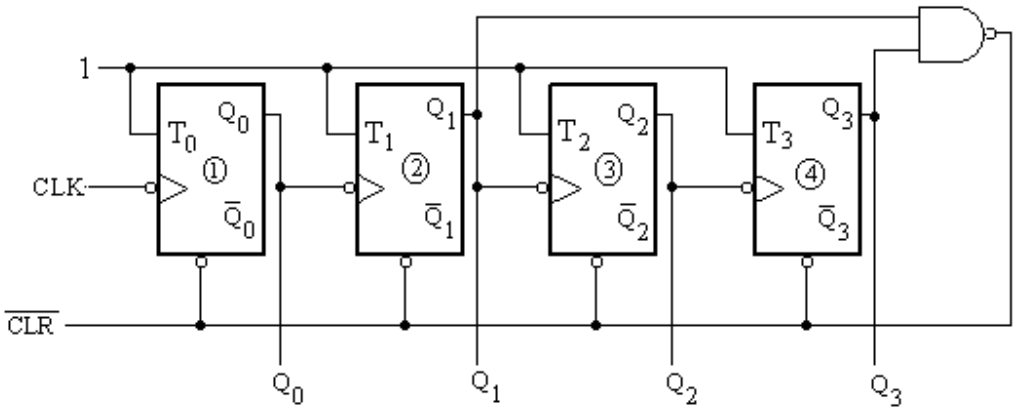


Fig. 10.15

The waveforms taken at Q's outputs are shown in figure 10.16.

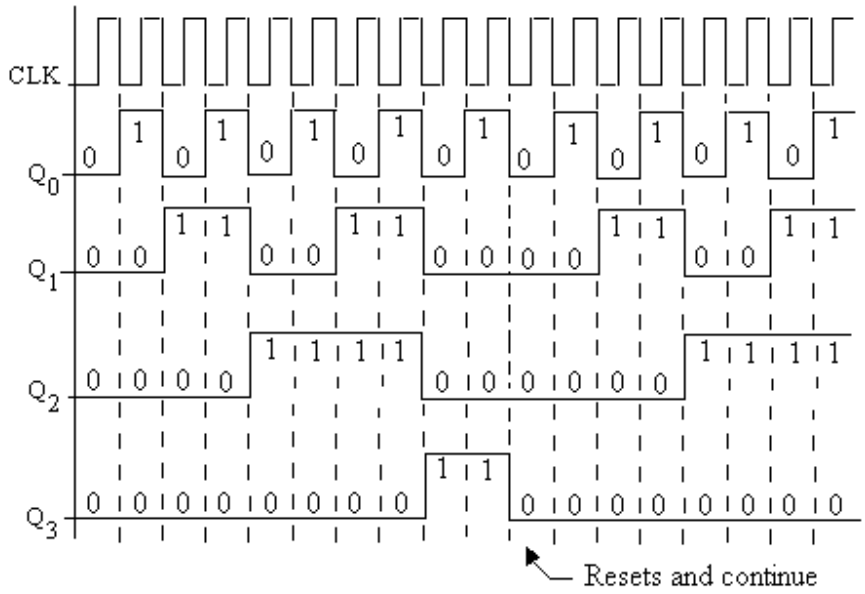


Fig 10.16

10.7 SYNCHRONOUS COUNTERS

In the forgoing sections of this chapter, the asynchronous or ripple counters were discussed in which the flip-flops were connected in series. These counters are simple to design but have delay which is the sum of delays of individual stages. The total accumulated delay in the counters cause a limitation on the speed of the asynchronous or ripples counters. In order to overcome this advantage synchronous counters are used. In these counters all the flip-flops are triggered simultaneously by the same clock pulse. The transition of all the flip-flops from present state to the next state will, therefore, occur at the same time, which reduces the delay of the counter. Synchronous counters can be designed by using JK, D or T type flip-flops.

10.7.1 Synchronous Binary Counter

The manner in which the counts progress in a binary counter is shown in table 10.6. Four T flip-flops are needed to design this binary counter. It may be noted from the table 10.6 that the output Q_0 changes its state for every clock pulse. So to get the toggled output Q_0 for every pulse, the T input of first flip-flop should be connected to high (logic 1). The output Q_1 changes its state, whenever Q_0 is 1 and stores when $Q_0 = 0$. The T input of second flip-flop should therefore be connected to Q_0 output of first flip-flop. Similarly, output Q_2 toggles when Q_0 and Q_1 are both 1. The output Q_3 toggles when Q_1, Q_2 and Q_3 are 1. From the above discussion the Boolean expressions for inputs of all the flip-flops are given by:

For the design of this counter it needs four flip-flops. A table is drawn in which inputs are outputs of the four flip-flops say $Q_3\ Q_2\ Q_1\ Q_0$ used as the inputs in the table. The outputs Q_3 through Q_0 should be in binary sequence and resets after $Q_3\ Q_2\ Q_1\ Q_0$ = 1001. The J K inputs for each flip-flop for the transition from present state to next state

are obtained from the excitation table (8.11) of the corresponding flip-flop. This is shown in table 10.8.

Table 10. 6

Outputs or Counts			
Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1
0	0	0	0

$T_0 = 1, \qquad T_1 = Q_0, \qquad T_2 = Q_0 \cdot Q_1 \quad$ and $\quad T_3 = Q_0 \cdot Q_1 \cdot Q_2$

The logic circuit diagram of synchronous binary counter is shown in figure 10.17.

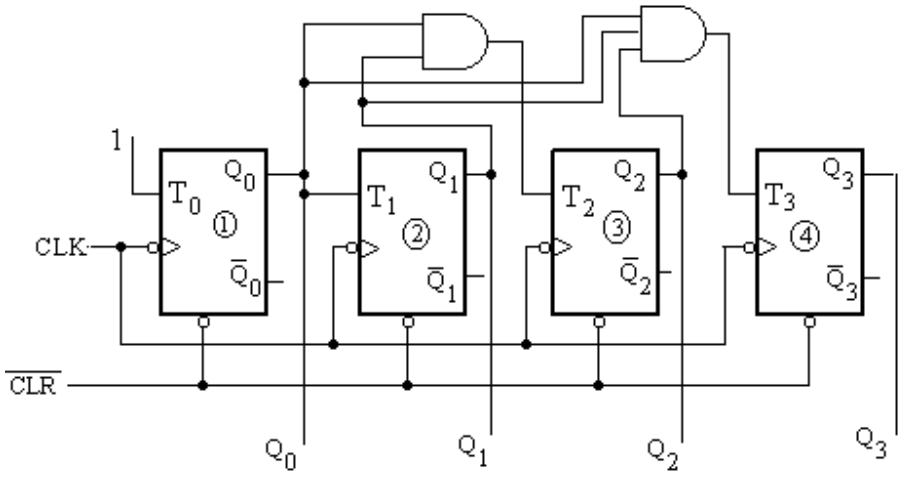


Fig. 10.17

It is clear from this figure that the clock inputs of all the flip-flops are connected together. This counter can also be designed by using 4 J K flip-flops (J and K inputs tied together) as shown in figure 10.18.

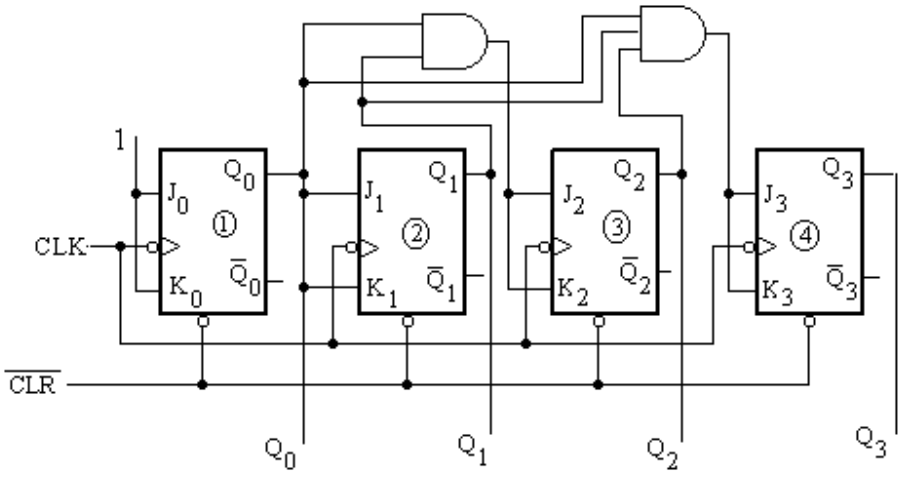


Fig. 10.18

Figures 10.19 show the waveforms taken at Q's outputs of all the flip-flops of synchronous binary counter. Note from the waveforms that output toggles at the trailing edge of the pulse and output Q_1 toggle when Q_0 is 1, Q_2 toggles when both Q_0 and Q_1 are 1 and Q_3 toggles when $Q_0 Q_1 Q_2$ are all 1.

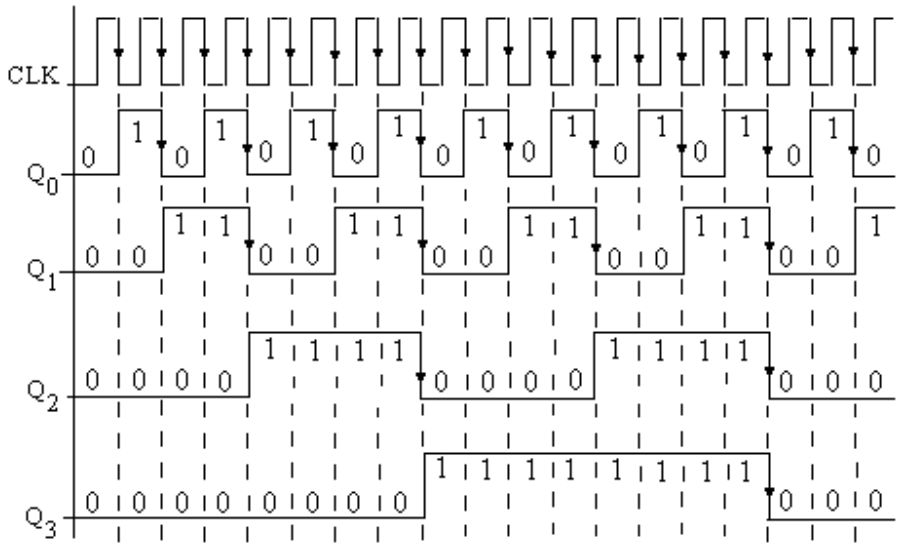


Fig. 10.19

10.7.2 Design of Synchronous Mod – N Counter

The design process of synchronous Mod – N counter will now be discussed. The value of N need not necessarily be a power of 2, for example Mod-5, Mod-6, Mod-8, Mod-10, Mod-11 etc. It is also desired to have counters in which the counting sequence is not always being the natural binary sequence. The counting sequence may be in cyclic code, 5421 code, 2421 code etc. The following procedure should be adopted for the design of synchronous counters for any counting sequence and modulus:

- First find the number of flip-flops *n* required to design a counter of Mod – N. It is obtained from the equation $N \leq 2^n$.

- A table is formed whose inputs are the required counting sequence of the counter.
- Flip-flop inputs are obtained from the excitation table of the flip-flops (discussed in the preceding chapter) for each counting sequence of the table (obtained in step second). The flip-flop inputs, that are capable of producing next state of the counter from the present state, are entered in the table.
- Karnaugh map is formed for each flip-flop input in terms of flip-flop outputs as the input variables.
- Simplify the K-map and get the minimal Boolean expression for each flip-flop input.
- Finally the required counter circuit is obtained by connecting the flip-flops and other gates as per the expressions obtained above.

The excitation table for R S, D, J K and T type flip-flops are reproduced in table 10.7 for the ready reference to the readers.

Table 10.7

Transitons $Q_n \rightarrow Q_{n+1}$	Inputs R_n S_n	Inputs J_n K_n	Input D_n	Input T_n
0 → 0	φ 0	0 φ	0	0
0 → 1	0 1	1 φ	1	1
1 → 0	1 0	φ 1	0	1
1 → 1	0 φ	φ 0	1	0

Using the procedure discussed above for the design of synchronous Mod-N counter, a few counters will be discussed in the following section.

10.7.3 Synchronous Decade counter

A decade counter also known as Mod-10 or divide-by-ten counter, can count from 0 to 9 and then it resets and count again. Let the counter counts in the natural binary sequence and it is designed using J K flip-flops.

For the design of this counter it needs four flip-flops. A table is drawn in which inputs are outputs of the four flip-flops say Q_3 Q_2 Q_1 Q_0 used as the inputs in the table. The outputs Q_3 through Q_0 should be in binary sequence and resets after Q_3 Q_2 Q_1 Q_0 = 1001. The J K inputs for each flip-flop for the transition from present state to next state are obtained from the excitation table (10.7) of the corresponding flip-flop. This is shown in table 10.8.

Table 10.8

Outputs or Counts											
Q_3	Q_2	Q_1	Q_0	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	φ	0	φ	0	φ	1	φ
0	0	0	1	0	φ	0	φ	1	φ	φ	1
0	0	1	0	0	φ	0	φ	φ	0	1	φ
0	0	1	1	0	φ	1	φ	φ	1	φ	1
0	1	0	0	0	φ	φ	0	0	φ	1	φ
0	1	0	1	0	φ	φ	0	1	φ	φ	1
0	1	1	0	0	φ	φ	0	φ	0	1	φ
0	1	1	1	1	φ	φ	1	φ	1	φ	1
1	0	0	0	φ	0	0	φ	0	φ	1	φ
1	0	0	1	φ	1	0	φ	0	φ	φ	1
0	0	0	0								

The K-maps for J_3, K_3, J_2, K_2, J_1 and K_1 are drawn as shown in figures 10.20(a) through (f) and the expressions for these input variables of J K flip-flops are given as:

$$J_3 = Q_2 \cdot Q_1 \cdot Q_0 \qquad K_3 = Q_0$$
$$J_2 = Q_1 \cdot Q_0 \qquad K_2 = Q_1 \cdot Q_0$$
$$J_1 = \overline{Q}_3 \cdot Q_0 \qquad K_1 = Q_0$$

The expression for K_1 may be taken as:

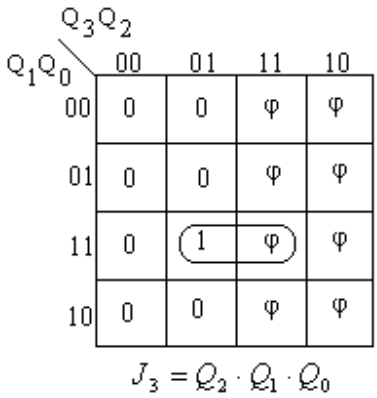
$$K_1 = \overline{Q}_3 \cdot Q_0$$

Since it become equal to J_1

i.e. $$J_1 = K_1 = \overline{Q}_3 \cdot Q_0$$

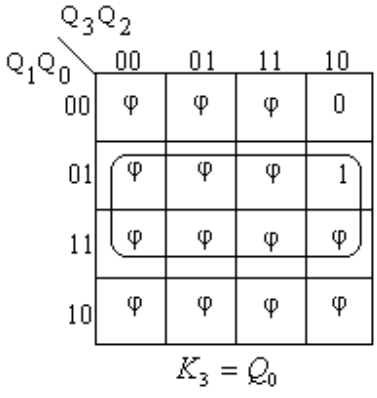
The expressions for J_0 and K_0 may directly be written from the table 10.8, as

$$J_0 = K_0 = 1$$



$J_3 = Q_2 \cdot Q_1 \cdot Q_0$

Fig.10.20(a)



$K_3 = Q_0$

Fig. 10.20(b)

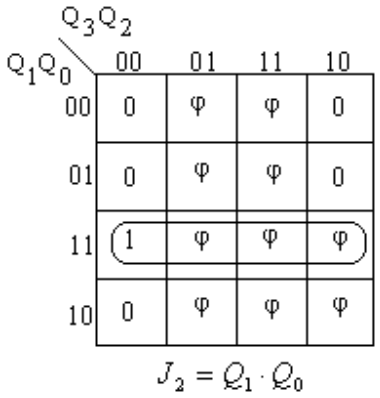


Fig. 10.20(c)

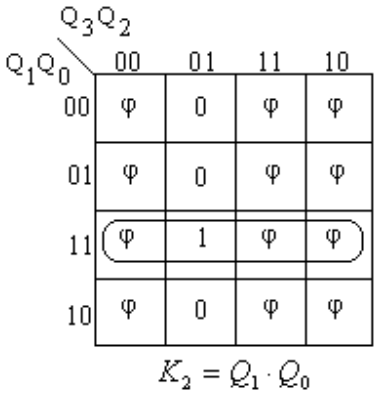


Fig. 10.20(d)

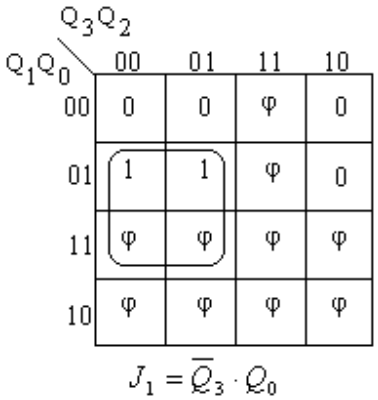


Fig.10.20(e)

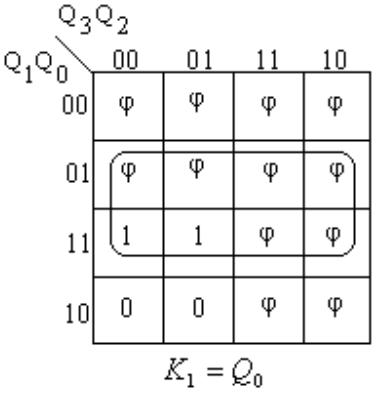


Fig.10.20(f)

The logic circuit diagram of synchronous decade counter whose outputs will be in the straight binary number is given in figure 10.21.

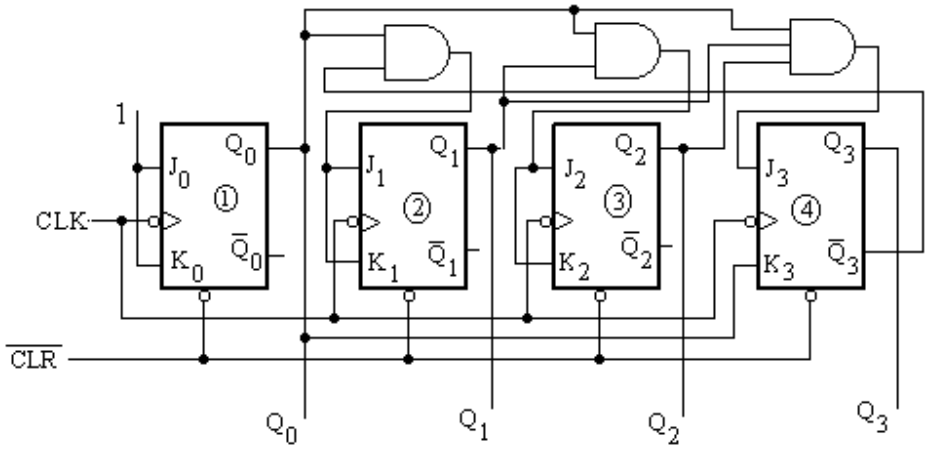


Fig. 10.21

Figure 10.21 shows the waveforms at the outputs of all the flip-flops. The sequence of the counter can be verified from the waveforms. At the trailing edge of the

clock pulse Q_0 output toggles. The various modes of operation of other outputs are shown in figure 10.21. These are obtained from the expressions of inputs of flip-flops by putting the previous values of outputs just before the trailing edge of the clock pulse.

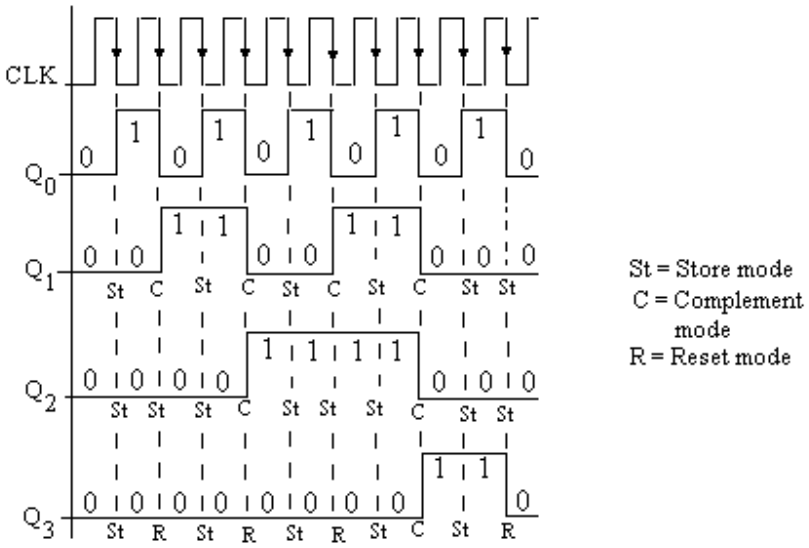


Fig. 10.21

Example 10.2: Design a synchronous Mod-12 up counter. The counting was made in natural binary sequence. Use T flip-flops to design the counter.

Solution: For the design of this counter it needs four flip-flops. This counter will count from 0000 to 1011 and resets to 0000 after this and again count. Table 10.9 shows the counting sequence of this counter. The outputs of the four flip-flops say $Q_3\,Q_2\,Q_1\,Q_0$ used as the inputs in the table and the T inputs for each flip-flop for the transition from present state to next state are obtained from the excitation table (11.7) of the corresponding flip-flop.

Table 10. 9

Outputs or Counts							
Q_3	Q_2	Q_1	Q_0	T_3	T_2	T_1	T_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	1
0	0	1	1	0	1	1	1
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	1
1	0	1	1	1	0	1	1
0	0	0	0				

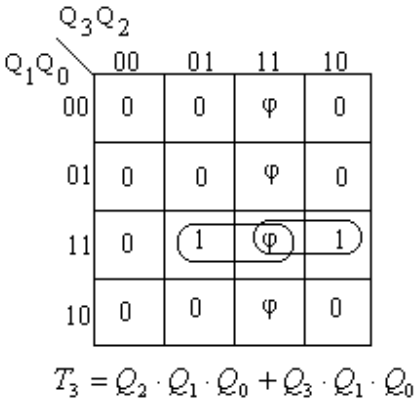


Fig. 10.22 (a)

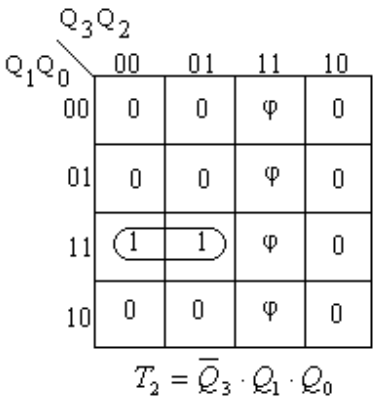


Fig. 10.22(b)

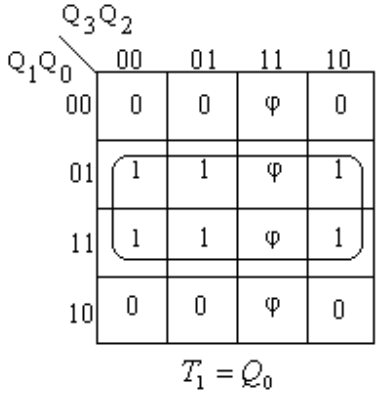


Fig. 10.22(c)

The K-maps for T_3, T_2 and T_1 are drawn as shown in figures 10.22 (a) through (c) and the expressions for these input variables of T flip-flops are given as:

$$T_3 = Q_2 \cdot Q_1 \cdot Q_0 + Q_3 \cdot Q_1 \cdot Q_0$$

$$T_2 = \overline{Q}_3 \cdot Q_1 \cdot Q_0$$

$$T_1 = Q_0$$

The expressions for T_0 may be obtained directly from the table 10.9, as all entries in T0 column of this table are 1.

$$T_0 = 1$$

The logic circuit diagram of synchronous Mod.-12 counter, whose outputs will be in the straight binary number from 0000 to 1011, is given in figure 10.23.

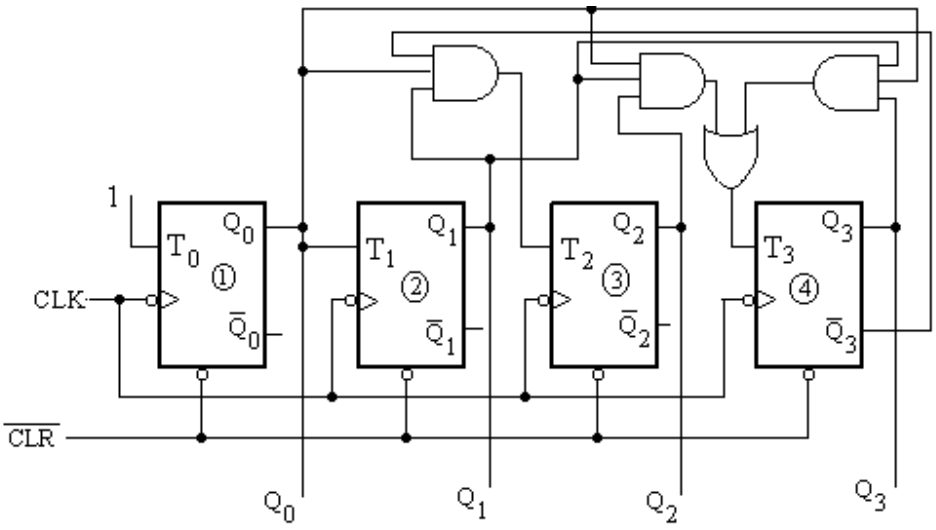


Fig. 10.23

Figure 10.24 shows the waveforms at the outputs of all the flip-flops. The sequence of the counter can be verified from the waveforms.

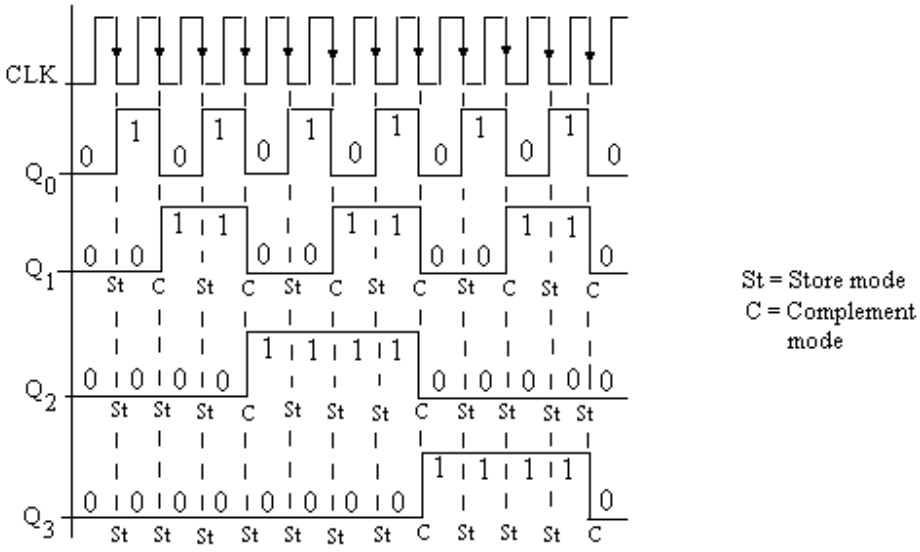


Fig. 10.24

10.8 SYNCHRONOUS COUNTERS WITH ARBITRARY COUNTING SEQUENCE

There are applications where it is required to design N-bit counters counting in some arbitrary counting sequence. For the design of such counters, the state diagram showing all the required states is drawn, then a table is formed in which present and the next states of the counters are written. The values of the inputs of the required number of flip-flops are entered as per the excitation table of the flip-flops. Finally, by getting the simplified Boolean expressions for the flip-flops inputs, the logic circuit for the counter is designed. The design of such counters may be understood by considering a following example.

Example 10.3: Design a synchronous Mod-10 counter to count in the sequence 0, 2, 4, 5, 6, 8, 9, 3, 1, 7, 0. Use J K flip-flops to design the counter.

Solution: For the design of this decade counter, four J K flip-flops are required. The state diagram showing the required states in the counter in sequence wise is given in figure 10.25.

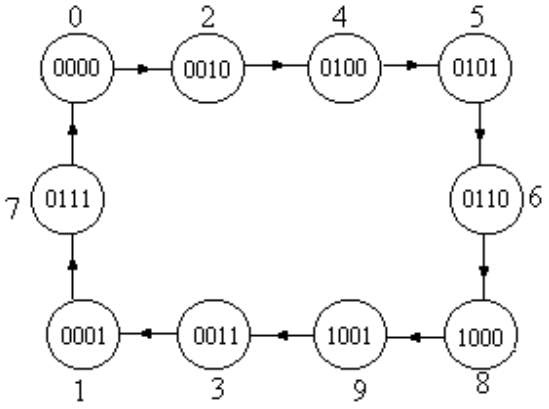


Fig. 10.25

Table 10.10 shows present states of the counting sequence and next states after the clock pulse and input values of the flip-flops.

Table 10.10

Presnt States				Next States											
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	1	0	0	φ	0	φ	1	φ	0	φ
0	0	0	1	0	1	1	1	0	φ	1	φ	1	φ	φ	0
0	0	1	0	0	1	0	0	0	φ	1	φ	φ	1	0	φ
0	0	1	1	0	0	0	1	0	φ	0	φ	φ	1	φ	0
0	1	0	0	0	1	0	1	0	φ	φ	0	0	φ	1	φ
0	1	0	1	0	1	1	0	0	φ	φ	0	1	φ	φ	1
0	1	1	0	1	0	0	0	1	φ	φ	1	φ	1	0	φ
0	1	1	1	0	0	0	0	0	φ	φ	1	φ	1	φ	1
1	0	0	0	1	0	0	1	φ	0	0	φ	0	φ	1	φ
1	0	0	1	0	0	1	1	φ	1	0	φ	1	φ	φ	0

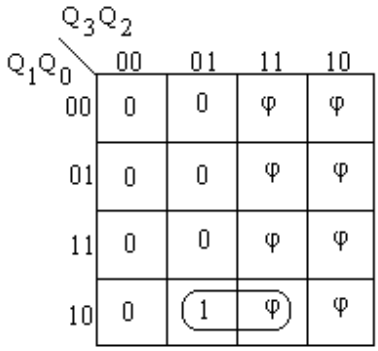
The K-maps for all inputs of the flip-flops are drawn as shown in figures 10.26 (a) through (g) and the expressions for these inputs are given as:

$$J_3 = Q_2 \cdot Q_1 \cdot \overline{Q}_0, \qquad K_3 = Q_0$$

$$J_2 = Q_1 \cdot \overline{Q}_0 + \overline{Q}_3 \cdot \overline{Q}_1 \cdot Q_0, \qquad K_2 = Q_1$$

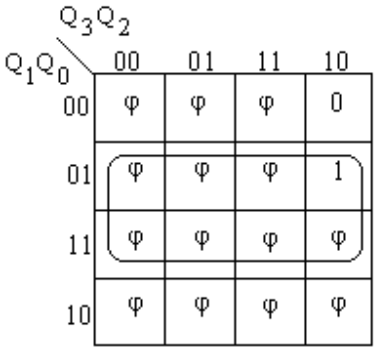
$$J_1 = \overline{Q}_3 \cdot \overline{Q}_2 + Q_0, \qquad K_1 = 1 \text{ (directly from the table)}$$

$$J_0 = Q_2 \cdot \overline{Q}_1 + Q_3, \qquad K_0 = Q_2 \cdot Q_0$$



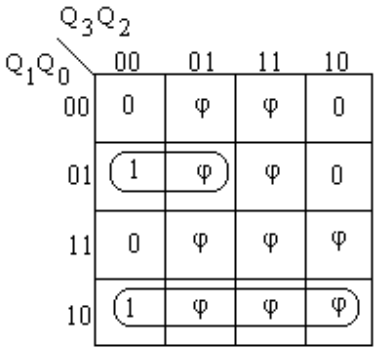
$J_3 = Q_2 \cdot Q_1 \cdot \overline{Q}_0$

Fig. 10.26(a)



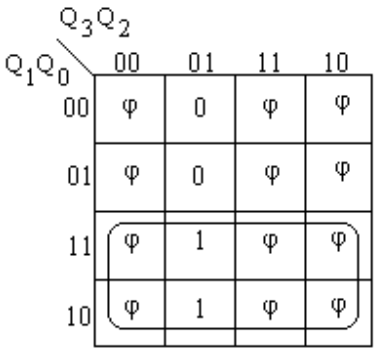
$K_3 = Q_0$

Fig. 10.26(b)



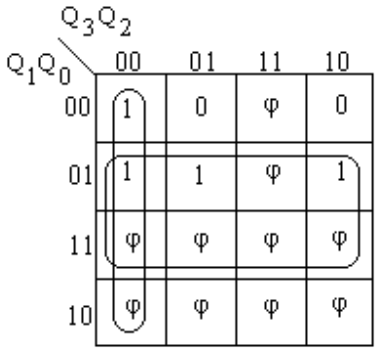
$J_2 = Q_1 \cdot \overline{Q}_0 + \overline{Q}_3 \cdot \overline{Q}_1 \cdot Q_0$

Fig. 10.26(c)



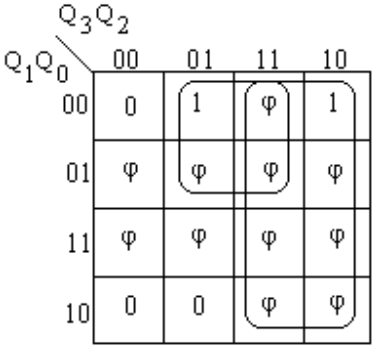
$K_2 = Q_1$

Fig. 10.26(d)



$J_1 = \overline{Q}_3 \cdot \overline{Q}_2 + Q_0$

Fig. 10.26(e)



$J_0 = Q_2 \cdot \overline{Q}_1 + Q_3$

Fig. 10.26(f)

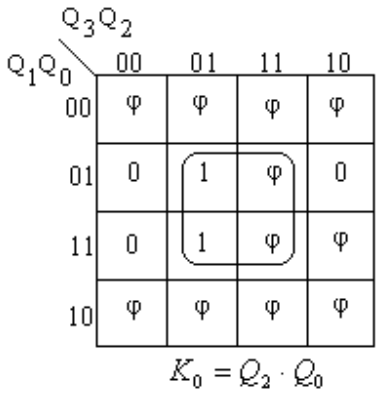


Fig. 10.26(g)

The logic circuit diagram of this synchronous counter, whose outputs will be in the given sequence, is shown in figure 10.27.

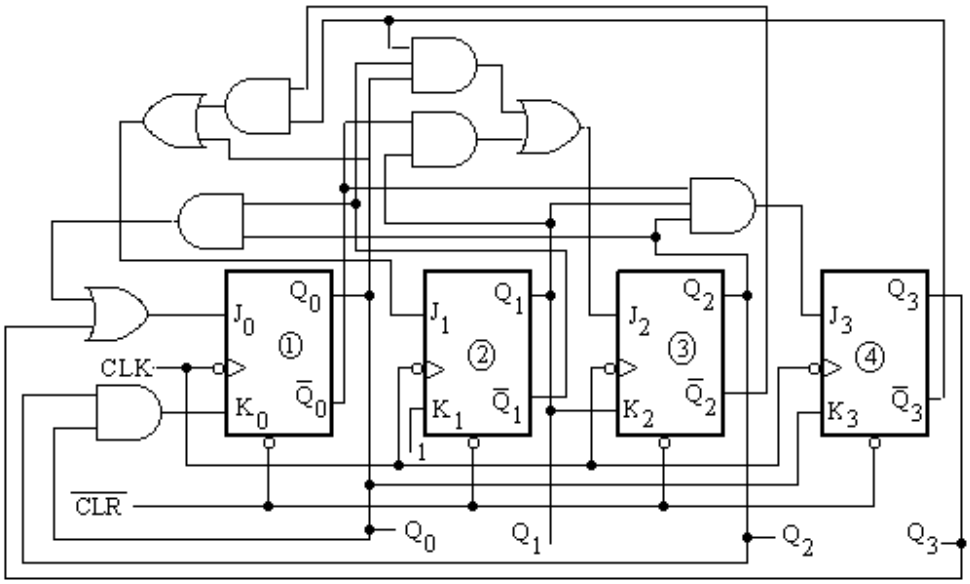


Fig. 10.27

Figure 10.28 shows the waveforms at the outputs of all the flip-flops. The sequence of the counter is verified from the waveforms.

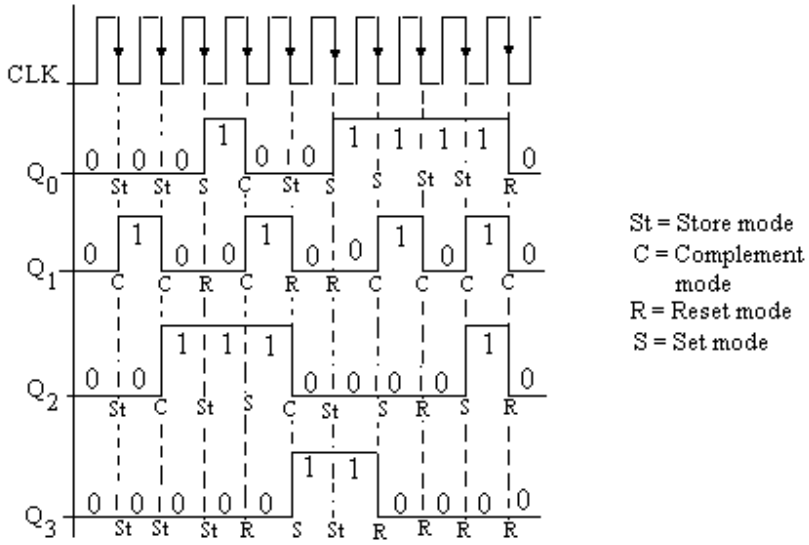


Fig. 10.28

10.9 SYNCHRONOUS CONTROLLED COUNTERS

Another class of synchronous counters called controlled counters will now be discussed, in which a control input is applied. The control input will decide which sequence is to be followed by the counter. The up-down counters fall in the category of controlled counters. The control input will decide whether the counter is used for up counter or down counter. The procedure for the design of such counter is the same as the other synchronous counters discussed above. Any type of flip-flops may be used for the design of such counters.

Consider the design of a counter that can count in mod-8 or mod-4 counter with an additional control input S. If the control input S is 0, the counter works as mod-4 counter and if S is 1, it works as mod-8 counter.

For the design of this counter, three J K flip-flops are required as $2^3 = 8$. The state diagram showing the required states in the counter in sequence wise is given in figure 10.29. The transition from 000 to 001 will take place when the control input S is 0 or 1

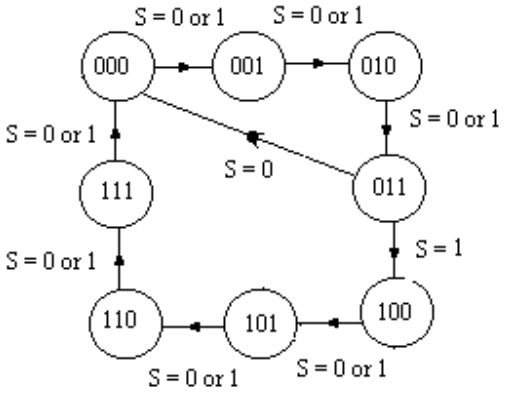


Fig. 10.29

and the transition from 011 to 000 will take place when S = 0, and transition from 011 to 100 will take place when S = 1. Table 10.11 shows present states of the counting

sequence and next states after the clock pulse and input values of the flip-flops. In the present state the control signal is also taken as one of the inputs. The table is according to the required sequence.

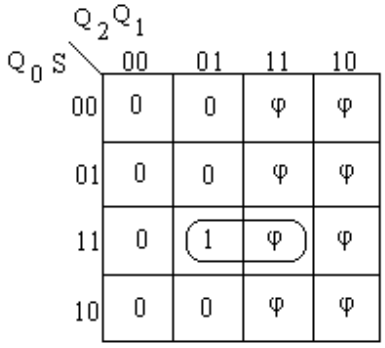
Table 10.11

Presnt States				Next States								
Q_2	Q_1	Q_0	S	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	1	0	φ	0	φ	1	φ
0	0	0	1	0	0	1	0	φ	0	φ	1	φ
0	0	1	0	0	1	0	0	φ	1	φ	φ	1
0	0	1	1	0	1	0	0	φ	1	φ	φ	1
0	1	0	0	0	1	1	0	φ	φ	0	1	φ
0	1	0	1	0	1	1	0	φ	φ	0	1	φ
0	1	1	0	0	0	0	0	φ	φ	1	φ	1
0	1	1	1	1	0	0	1	φ	φ	1	φ	1
1	0	0	0	1	0	1	φ	0	0	φ	1	φ
1	0	0	1	1	0	1	φ	0	0	φ	1	φ
1	0	1	0	1	1	0	φ	0	1	φ	φ	1
1	0	1	1	1	1	0	φ	0	1	φ	φ	1
1	1	0	0	1	1	1	φ	0	φ	0	1	φ
1	1	0	1	1	1	1	φ	0	φ	0	1	φ
1	1	1	0	0	0	0	φ	1	φ	1	φ	1
1	1	1	1	0	0	0	φ	1	φ	1	φ	1

The expressions for J_0 and K_0 are obtained directly from the table. The K-maps for other inputs of the flip-flops are drawn as shown in figures 10.30 (a) through (d) and the expressions for these inputs are given as:

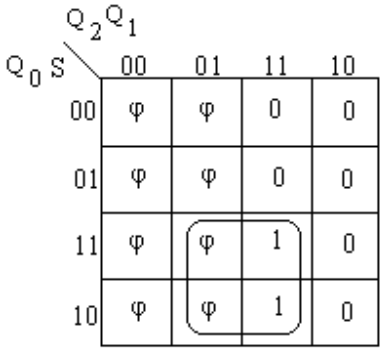
$$J_0 = K_0 = 1 \qquad J_1 = K_1 = Q_0$$

$$J_2 = Q_1 \cdot Q_0 \cdot S \qquad K_2 = Q_1 \cdot Q_0$$



$J_2 = Q_1 \cdot Q_0 \cdot S$

Fig. 10.30 (a)



$K_2 = Q_1 \cdot Q_0$

Fig. 10.30 (b)

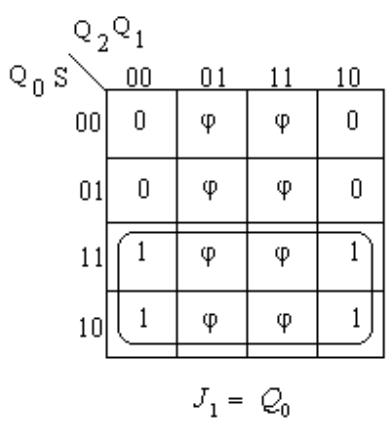


Fig. 10.30 (c)

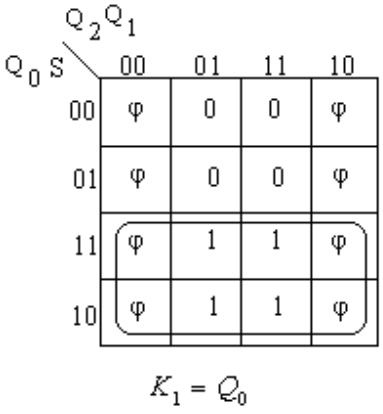


Fig. 10.30 (d)

The logic circuit diagram of this synchronous counter is shown in figure 10.31.

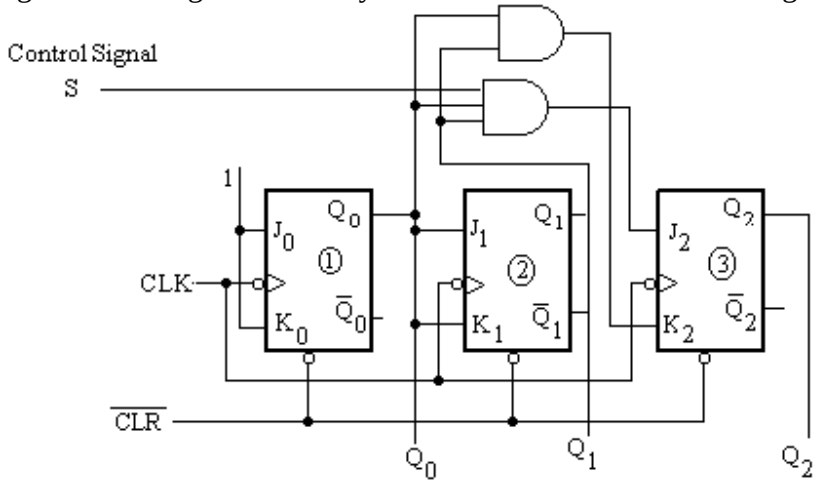


Fig. 10.31

Figure 10.32 shows the waveforms at the outputs of all the flip-flops. The sequence of the counter is verified from the waveforms. If the counter is reset and the control input *S* is zero then it will follow the sequence of mod – 4 i.e. it will count 000, 001, 010, 011 and repeats. However, if the counter is reset and the control input *S* is 1, then the counter will count the sequence 000, 001, 010, 011, 100, 101, 110, 111 and repeats.

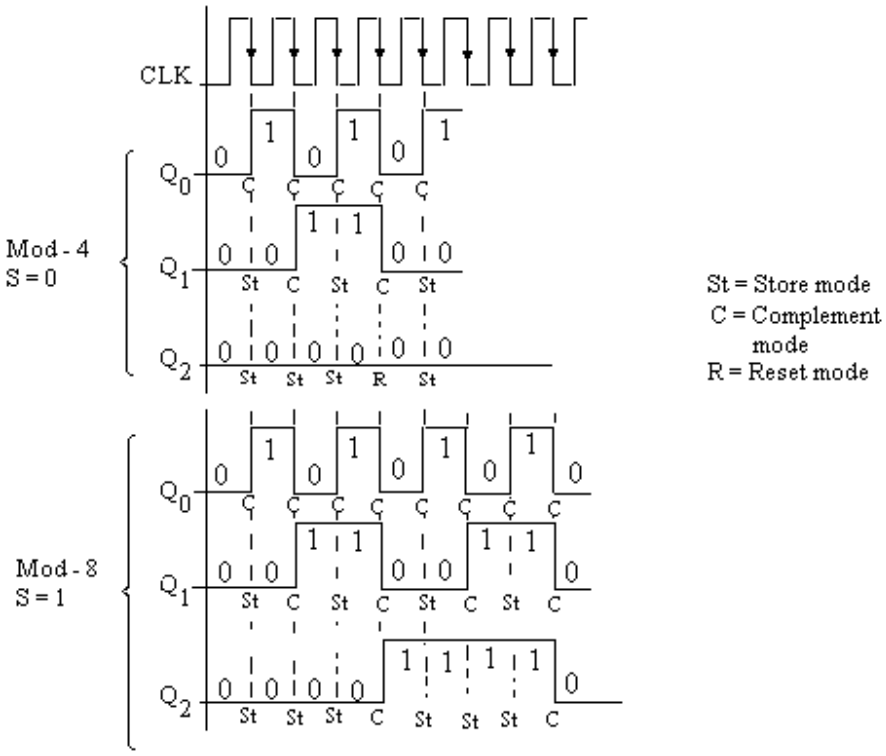


Fig. 10.32

Example 10.4: Design a synchronous Mod-8 up down counter. A control input may be used that allows the counter to count in the up sequence or down sequence. Use T flip-flops to design this counter.

Solution: For the design of this counter, three T flip-flops are required as $2^3 = 8$. The state diagram showing the required states in the counter in sequence wise is given in figure 10.33. A control signal S is used in the counter. If the control signal is 1, the counter works as up counter and if S = 0, the counter works as down counter.

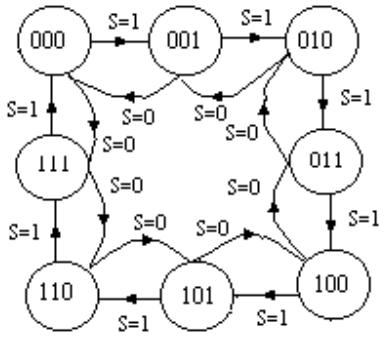


Fig. 10.33

Table 10.11 shows present states of the counting sequence and next states after the clock pulse and input values of the flip-flops. In the present state the control signal is also taken as one of the inputs. Use of transition table of T flip-flops is made for getting T inputs of the flip-flops. The table is according to the required sequence.

Table 10.12

Presnt States				Next States					
Q_2	Q_1	Q_0	S	Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	0	0	0	0	1
0	0	1	1	0	1	0	0	1	1
0	1	0	0	0	0	1	0	1	1
0	1	0	1	0	1	1	0	0	1
0	1	1	0	0	1	0	0	0	1
0	1	1	1	1	0	0	1	1	1
1	0	0	0	0	1	1	1	1	1
1	0	0	1	1	0	1	0	0	1
1	0	1	0	1	0	0	0	0	1
1	0	1	1	1	1	0	0	1	1
1	1	0	0	1	0	1	0	1	1
1	1	0	1	1	1	1	0	0	1
1	1	1	0	1	1	0	0	0	1
1	1	1	1	0	0	0	1	1	1

The expressions for T_0 is obtained directly from the table. The K-maps for other inputs of the flip-flops are drawn as shown in figures 10.34 (a) and (b) and the expressions for these inputs are given as:

$$T_2 = Q_1 \cdot Q_O \cdot S + \overline{Q}_1 \cdot \overline{Q}_O \cdot \overline{S} \qquad T_1 = Q_O \cdot S + \overline{Q}_O \cdot \overline{S} \qquad T_0 = 1$$

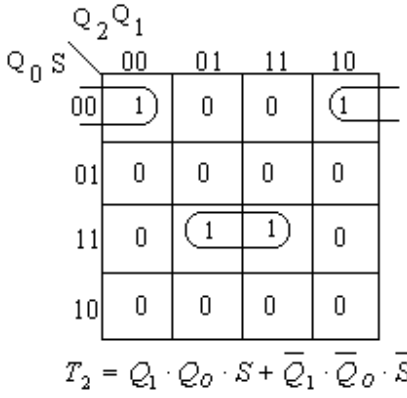


Fig. 10.34(a)

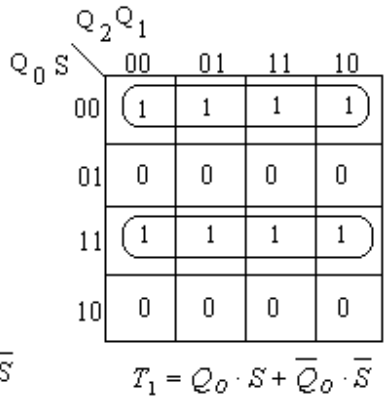


Fig. 10.34(b)

The logic circuit diagram of this synchronous counter is shown in figure 10.35. Figure 10.36 shows the waveforms at the outputs of all the flip-flops. The sequence of the counter is verified from the waveforms. If the counter is reset and the control input *S* is 1 then it works as mod -8 up counter i.e. it will count 000, 001, 010, 011, 100, 101, 110, 111 and repeats. However, if the counter is reset and the control input *S* is 0, then it

works as mod -8 down counter i.e. it will count 000, 111, 110, 101, 100, 011, 010, 001 and repeats.

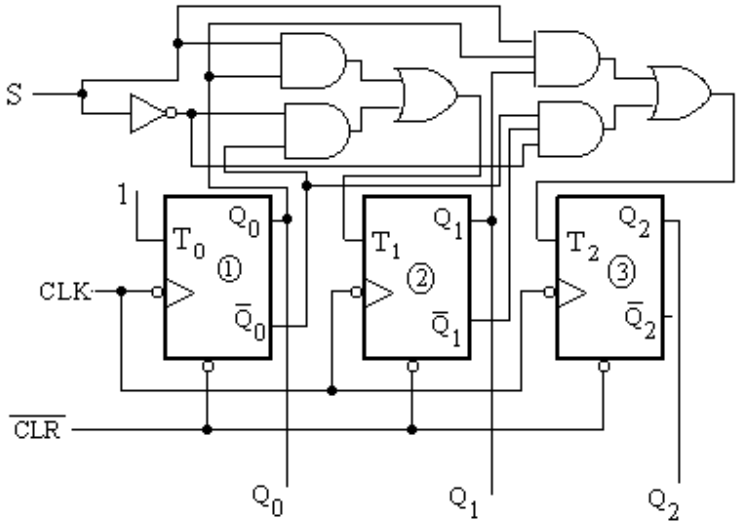


Fig. 10.35

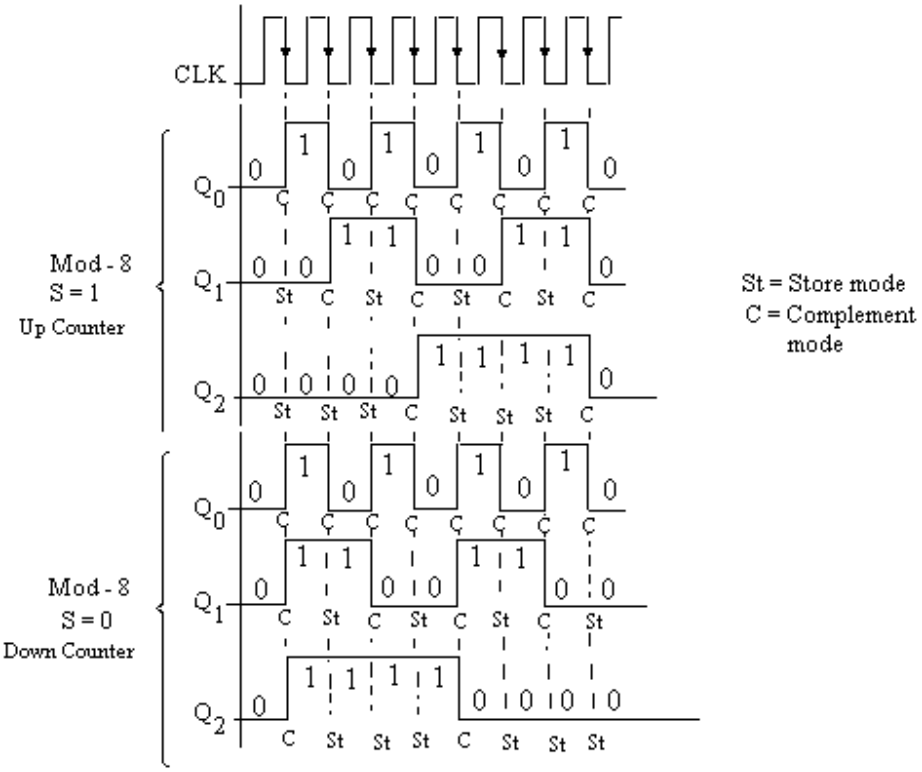


Fig. 10.36

10.10 GENERATION OF CONTROL SIGNALS

In many digital applications control signals are required to start, execute as well as step various operations in a specified time sequence. For the design of such control

signals, a counter circuit is designed whose outputs are connected to a decoder. The decoder gives the required control signal. The counter circuit may be synchronous or asynchronous. The procedure for designing the counter is the same as discussed above in this chapter. The block diagram for generating the control signal is shown in figure 10.37. The design of control signal may well be understood by considering the following example.

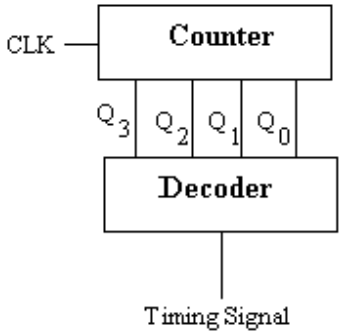


Fig. 10.37

Example 10.5 Generate a control signal which can deliver the following pulse train. The pulse train repeats after 7 pulses. The counter may be designed using T flip-flops.

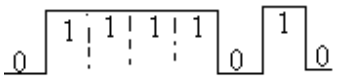


Fig. 10.38

Solution: From the given problem it is clear that a control signal (say S) is to be generated which gives the periodic pulse train of 0111101 and then repeats. The pulse train repeats after 7 seven pulses. So the outputs of Mod-7 counter are to be connected to a decoder circuit. For this a mod –7 counter is to be designed. Three T flips are required for the design of mod -7 counter. Table 10.13 shows the counting sequence and required inputs of T flip-flops. The expressions for inputs of T flip-flops obtained from the K – maps shown in figure 10.39 are given as:

$$T_2 = Q_2 \cdot Q_1 + Q_1 \cdot Q_0 \qquad T_1 = Q_2 \cdot Q_1 + Q_0$$
$$T_0 = \overline{Q}_2 + \overline{Q}_1$$

Table 10.13

Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	1	1	0
0	0	0			

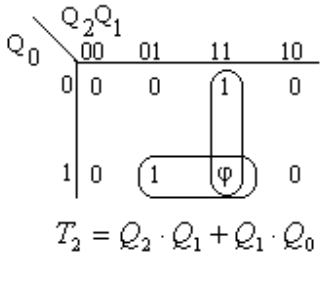
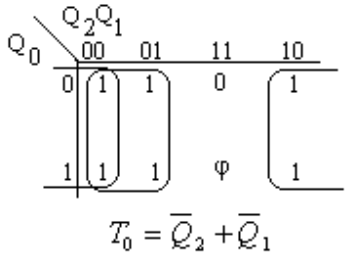


Fig. 1039(a)



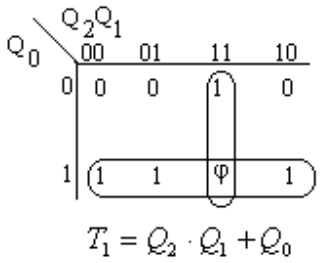


Fig. 10.39(b) **Fig. 10.39**(c)

The truth table for the decoder is obtained by direct observation of the given timing sequence (ref. fig. 10.38). It is shown in table 10.14. The Boolean expression for the output S of decoder is obtained using the K-map of figure 10.40, the unused counts are treated as don't care conditions. The expression is given by:

$$S = Q_1 + \overline{Q}_2.Q_0 + Q_2.\overline{Q}_0$$

Table 10.14

Q_2	Q_1	Q_0	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1

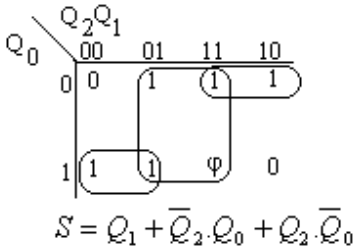


Fig. 10.40

The complete logic diagram of the counter with decoder is shown in figure 10.41 and timing diagram for such control signal may also be drawn as shown in figure 10.42.

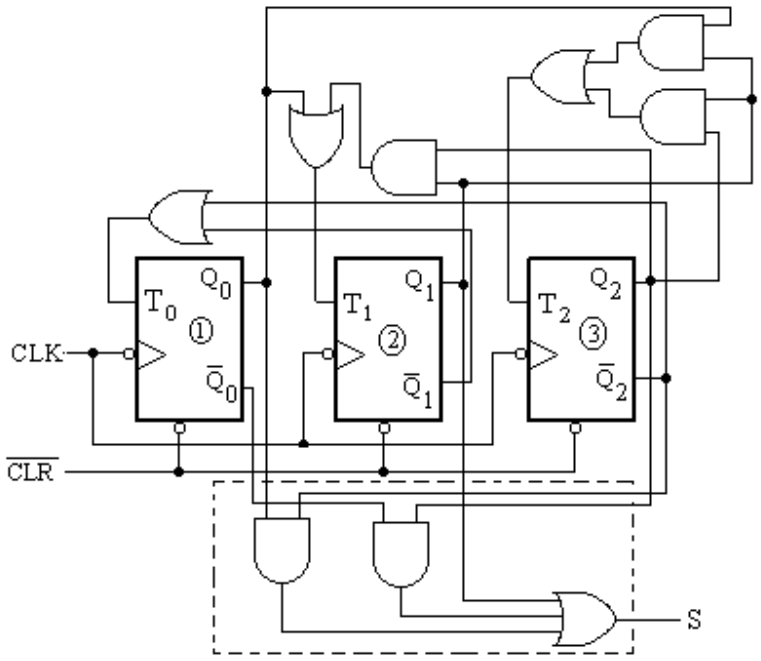


Fig. 10.41

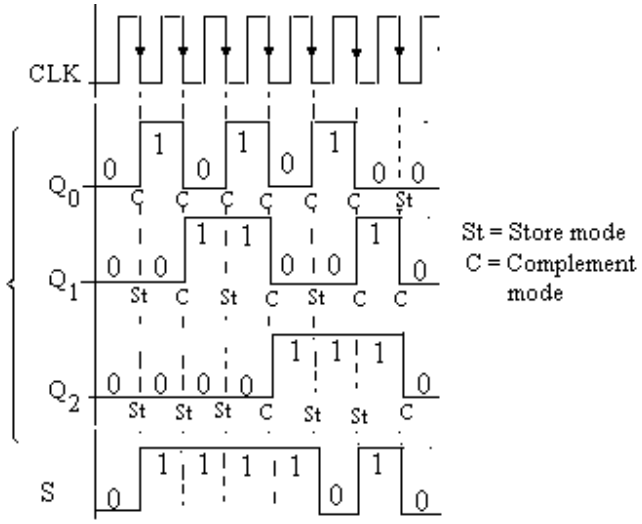


Fig. 10.42

10.11 COUNTER ICs

Counters are available in the form of ICs, a few most commonly used asynchronous counter ICs are given in table 10.15.

Table 10.15

IC No.	Description
7490, 74290	BCD counter
7492	Divide-by-12 counter
7493, 74293	4-bit binary counter
74176, 74196	BCD counter with preset
74393	Dual decade counter
743931	Dual 4-bit binary counter

Details of few of the above mentioned ICs are given below:

IC 7490 Decade Counter (divide-by-two and divide-by-five): This IC consists of four master slave flip-flops internally connected to provide divide by two and divide by five counter. The logic diagram of this IC is given in figure 10.43(a) with its pin diagram and logic symbol in 10.43(b) and 10.43(c) respectively. The output from flip-flop (0) is not internally connected to the succeeding stages; therefore the count may be separated into two independent count modes.

(i) It is used as a binary coded decimal decade counter; the $\overline{CLK}1$ clock input must be externally connected to the Q_0 output. The clock input $\overline{CLK}$ receives the incoming count, and a count sequence is obtained in accordance with the BCD count for 9's complement decimal application.

(ii) If a symmetrical divide-by-ten count is desired for frequency synthesizers or other applications requiring division of a binary count by a power of ten, Q_3 output must be externally connected to $\overline{CLK}$ input. The input count is then applied at the $\overline{CLK}1$ input and a divide by ten square wave is obtained at output Q_0.

(iii) For operation as divide-by-two counter and a divide by five counter, no external interconnections are required. Flip-flop (0) is used as a binary element for the divide-by-two function. The $\overline{CLK}1$ input is used to obtain binary divide-by-five operation at the Q_1, Q_2 and Q_3 outputs. In this mode two counters operate independently; however, all four flip-flops are reset simultaneously.

Tables 10.16 and 10.17 indicate BCD counting sequence and reset conditions respectively.

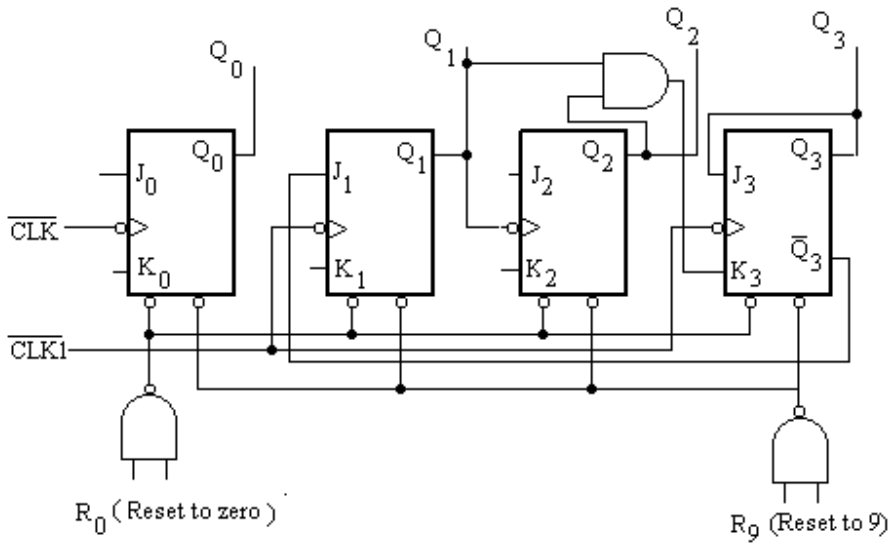


Fig 10.43(a)

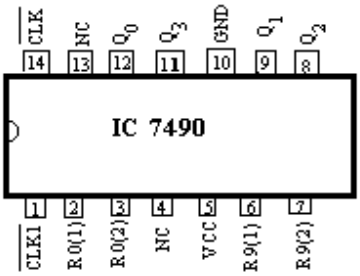


Fig. 10.43(b)

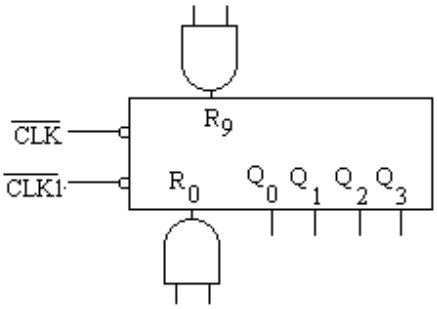


Fig. 10.43(b)

Table 10.16

COUNTS	OUTPUTS			
	Q_3	Q_2	Q_1	Q_0
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

Table 10.17

RESET INPUTS				OUTPUTS			
R0(1)	R0(2)	R9(1)	R9(2)	Q_3	Q_2	Q_1	Q_0
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
X	L	X	L	COUNT			
L	X	L	X	COUNT			
L	X	X	L	COUNT			
X	L	L	X	COUNT			

X Indicates that either a HIGH or a LOW level is present.

IC 7492 Divide-by-twelve Counter (divide-by-two and divide-by-six):

This IC is a 4-bit binary counter consisting of four master slave flip-flops which are internally connected to provide a divide-by-two counter and divide-by-six counter. A gated direct reset line is provided which inhibits the count inputs and simultaneously returns the four flip-flop outputs to a low level. The logic diagram of this IC is given in figure 10.44(a) with its pin diagram and logic symbol in 10.44(b) and 10.44(c) respectively. The output from flip-flop (0) is not internally connected to the succeeding stages; therefore the counter may be operated into two independent count modes.

(i) To use it as a divide-by-two counter, output Q_0 must be externally connected to clock input $\overline{CLK}1$. The input count pulses are applied to input $\overline{CLK}$. This IC when used as divide-by-twelve counter, it counts from 0 to 11, but counts from 0 to 13 with skipping counts 6 and 7 as shown in truth table 10.18.

(ii) When it is used as divide-by-six counter, the input count pulses are applied to input $\overline{CLK}1$. Simultaneous frequency division of 3 and 6 are available at the Q2 and Q3 outputs. The truth table 10.19 shows the reset conditions of this IC.

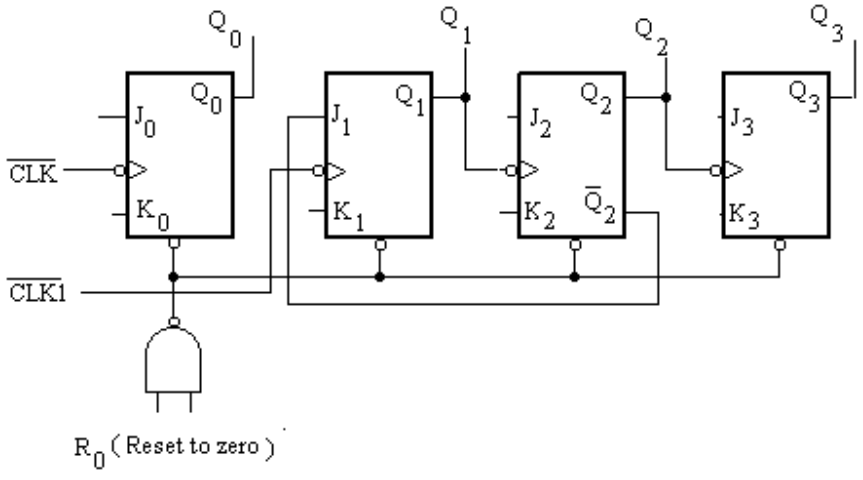


Fig. 10.44(a)

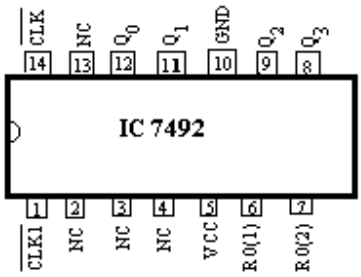


Fig. 10.44(b)

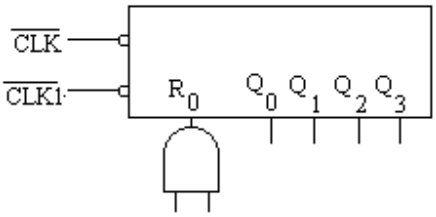


Fig. 10.44(c)

Table 10.18

COUNTS	OUTPUTS			
	Q_3	Q_2	Q_1	Q_0
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	H	L	L	L
7	H	L	L	H
8	H	L	H	L
9	H	L	H	H
10	H	H	L	L
11	H	H	L	H

Table 10.19

RESET INPUTS		OUTPUTS			
R0(1)	R0(2)	Q_3	Q_2	Q_1	Q_0
H	H	L	L	L	L
L	X	COUNT			
X	L	COUNT			

X Indicates that either a HIGH or a LOW level is present.

IC 7493 Divide-by-sixteen Counter (divide-by-two and divide-by-eight):

This IC is a 4-bit binary counter consisting of four master slave flip-flops which are internally connected to provide a divide-by-two counter and a divide-by-six counter. A gated direct reset line is provided which inhibits the count inputs and simultaneously returns the four flip-flop outputs to a low level. The logic diagram of this IC is given in figure 10.45(a) with its pin diagram and logic symbol in 10.45(b) and 10.45(c) respectively. The output from flip-flop (0) is not internally connected to the succeeding stages; therefore the counter may be operated into two independent count modes.

(i) When this is used as a 4-bit ripple counter, the ouput Q_0 must be externally connected to input $\overline{CLR}1$. The input count pulses are applied to input $\overline{CLR}$. Division of 2, 4, 8 and 16 are simultaneously performed as shown in truth table 10.20.

(ii) It can be used as a 3-bit ripple counter, the input count pulses are applied to input $\overline{CLR}1$. Simultaneous frequency divisions of 2, 4 and are available at Q_1, Q_2 and Q_3 outputs. Table 10.21 shows the reset conditions of this IC.

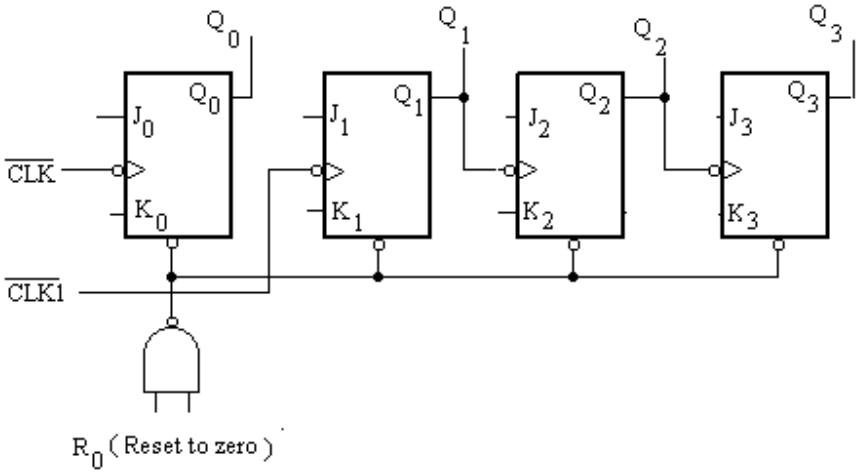


Fig. 10.45(a)

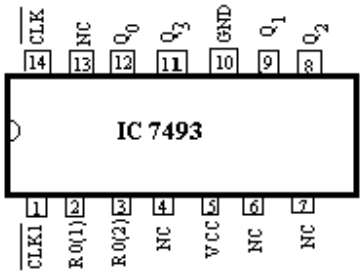


Fig. 10.45(b)

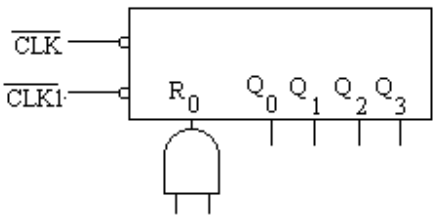


Fig. 10.45(c)

Table 10.20

COUNTS	OUTPUTS			
	Q_3	Q_2	Q_1	Q_0
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

Table 10.21

RESET INPUTS		OUTPUTS			
R0(1)	R0(2)	Q_3	Q_2	Q_1	Q_0
H	H	L	L	L	L
L	X	COUNT			
X	L	COUNT			

X Indicates that either a HIGH or a LOW level is present.

The list of synchronous counter ICs is given in table 10.22. The detailed description of few of these ICs will be discussed below:

Table 10.22

IC No.	Description
74160	Synchronous Decade Counter with clear
74161	Synchronous 4-bit Binary Counter with clear
74162	Synchronous Decade Counter
74163	Synchronous 4-bit Binary Counter
74190	Synchronous Up/Down Decade Counter
74191	Synchronous Up/Down Binary Counter
74192	Synchronous Up/Down Decade Counter Dual clock with clear
74193	Synchronous 4-bit Binary Up/Down Counter Dual clock with clear

IC 74160 Synchronous Decade Counter with Clear: The logic pin diagram of this IC is shown in figure 10.46. It works on the positive edge of the clock pulse, which is applied to T input (pin -2) terminal. The truth table of this IC is shown in table 10.23. It may be preset to any BCD count data applied to the A B C D inputs, and set terminal S as low and reset terminal R is high. A low on R will reset the counter. The carry out terminal CO will be high on the terminal count 1001, which helps in cascading several such counter ICs. The counter enable terminals FE1 and FE2 must be high to count the input pulses

Table 10.23

Input					Output				
R	S	FE1	FE2	T	QA	QB	QC	QD	CO
L	X	X	X	X	L	L	L	L	L
H	L	X	X	↑	Data is loaded				
H	H	H	H	↑	Counts progresses				
					H	L	L	H	H

X Indicates that either a HIGH or a LOW level is present.

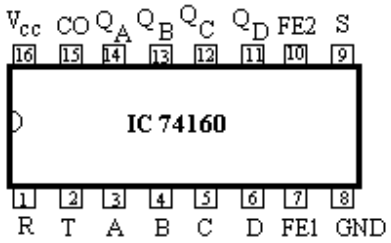


Fig. 10.46

IC 74163 Synchronous Four-bit Binary Counter The logic pin diagram of this IC is shown in figure 10.47. It works on the positive edge of the clock pulse, which is applied to T input (pin -2) terminal. The truth table of this IC is shown in table 10.24. It may be preset to any BCD count data applied to the A B C D inputs, and set terminal S as low and reset terminal R is high. A low on R will reset the counter. The carry out terminal U will be high on the terminal count 1111, which helps in cascading several such counter ICs. The counter enable terminals FE1 and FE2 must be high to count the input pulses

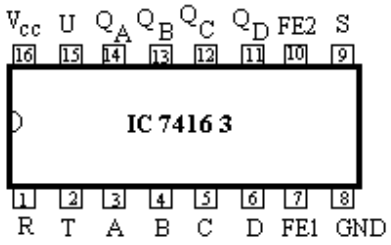


Fig. 10.47

Table 10.24

Input					Output				
R	S	FE1	FE2	T	QA	QB	QC	QD	U
L	X	X	X	↑	L	L	L	L	L
H	L	X	X	↑	Data is loaded				
H	H	H	H	↑	Counts progresses				
					H	H	H	H	H

X Indicates that either a HIGH or a LOW level is present.

IC 74190 Synchronous UP/Down Decade Counter: Figure 10.48 shows the logic pin diagram of the IC 74190 which can work in either up direction or down ward direction. The clock pulse is applied to T input (pin -14) Terminal. The truth table of this IC is shown in table 10.25. When BA terminal is high the counter counts down and then it is low, the counter counts up. It may be preset to any BCD count data applied to the D C B A inputs, and set terminal S as low and reset terminal R is high. A low on R will reset the counter. The pin U produces a high pulse when terminal count 9 (1001) is reached in the up counting or when the terminal count 0 (0000) is reached in the down counting.

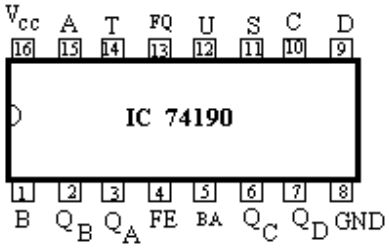


Fig. 10.48

Table 10.25

Input								Output					
FE	S	D	C	B	A	BA	T	QD	QC	QB	QA	U	FQ
H	H	X	X	X	X	X	X	No Change				L	H
X	L					X	X	D	C	B	A	L	H
L	H	X	X	X	X	L		COUNT UP				L	H
L	H	X	X	X	X	H		COUNT DOWN				L	H

10.12 Counter Applications

There are numerous applications of counters in digital circuits. A few important applications of counters, such as event counter, digital clock and digital frequency meter, are being discussed below:

10.12.1 Event Counter

Event counter is one which can count and display the physical counts. For example, the number of persons entering to a room or hall is to be counted and displayed in the digital form. This system will have a beam of light to fall on the Light Dependent Resistor (LDR), the output of which is connected to the clock input of the counter and display circuit. Whenever someone crosses or interrupts the light to fall on the LDR, a pulse will be produced. These pulses will be counted and simultaneously displayed on the display device. The complete circuit diagram with 4-digit display device is shown in figure 10.49, which is capable of storing the numbers from 0000 to 9999. It consists of four decade counter ICs 7490, which gives the outputs in BCD form. The BCD outputs are converted to seven segment outputs using BCD to seven segment decoder/driver ICs 7447. The seven segment outputs, when connected to FNDs, give the decimal display of the counted pulses. The counter may be reset if the reset switch is momentarily switched off. The IC9 (7413) produces a positive going pulse whenever a beam of light is interrupted by the entrants in the hall.

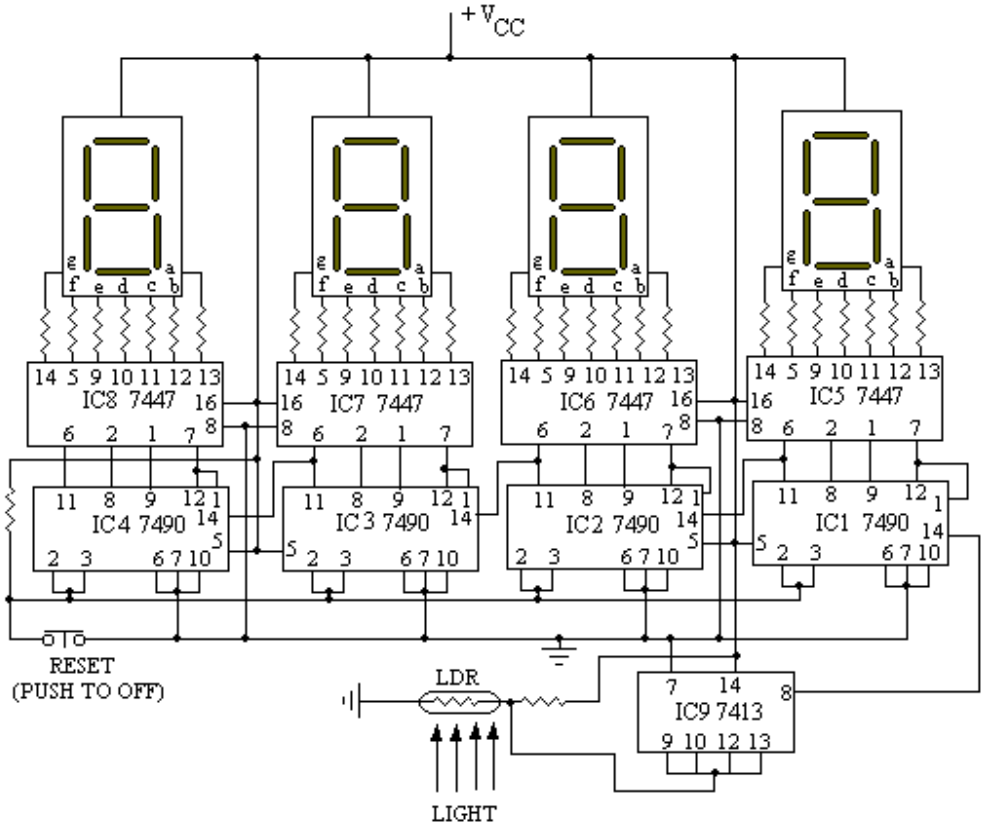


Fig. 10.49

10.12.2 Digital Frequency Meter

The digital frequency meter is an electronic instrument used to measure the frequency of a periodic waveform. The basic principle for the precise determination of frequency of an unknown signal is illustrated in figure 10.50. The unknown signal is applied to amplifier/attenuator, where the signal is amplified if it is a weak signal and attenuated if the signal of high amplitude. The amplified signal is then connected to a Schmitt trigger circuit where the signal is converted to a square wave. The square is differentiated to get the narrow pulse train. The number of pulses in the pulse train is equal to the frequency of the input unknown signal. This narrow pulse train is then applied to one input of a two input AND gate. The second input of this AND gate is connected another standard sample pulse of constant width. The sample pulse controls for how long the pulse train is allowed to pass through the AND gate to the digital counter. If the width of this sample pulse is kept as 1 second, then the AND gate will allow the pulse train to go to the input of the counter for 1 second. The counter will display the counts on the display devices (in digital form) counted by it for 1 second. The number displayed on the display devices will show the frequency of the input signal directly in Hz, since the number of pulses in the pulse train is equal to the frequency of the input unknown signal. The digital counter basically contains BCD counter, decoder and display unit (seven segment display).

Further for the continuous counting of the pulses the counter should be reset at the beginning of the sample pulse, which is done with the help of resetting and latching pulse generator. A positive going pulse is applied to reset the counter. Counter latching operation is performed if the reset terminal of the counter is grounded.

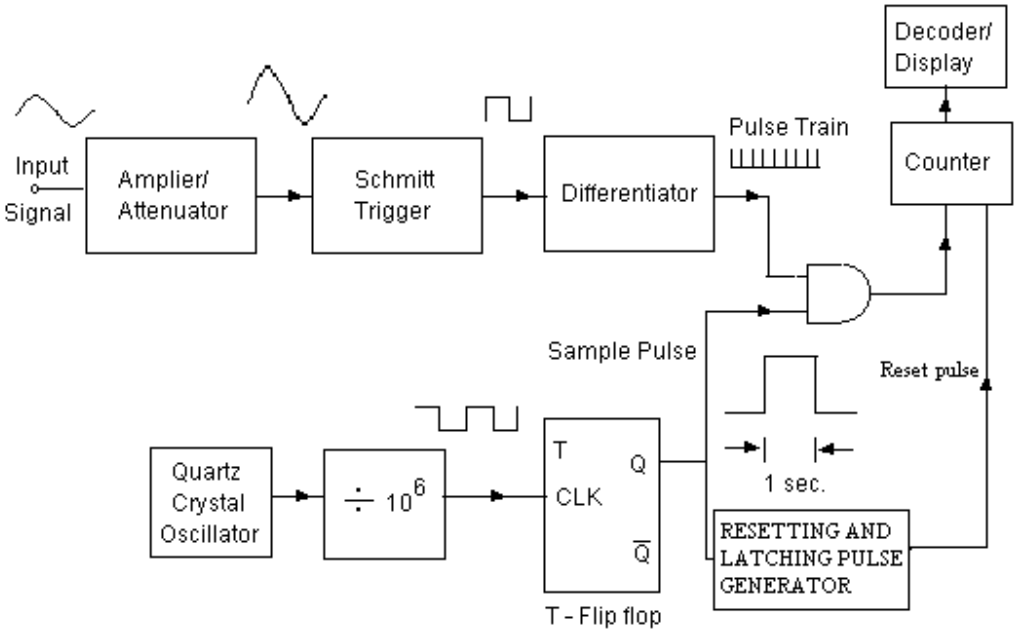


Fig. 10.50

The accuracy of the counter will depend on the accuracy of the width of the sample pulse. The sample pulse of standard time period is, therefore, obtained from a

high frequency quartz crystal oscillator, say, 1 MHz. The frequency of this crystal oscillator is divided by a factor of 10^6 using frequency divider circuit, which gives a square wave of 1 Hz frequency. Finally the 1 Hz frequency is divided a factor of 2, to obtain a square whose pulse width is 1 second. If the width of the sample pulse is taken as 1 msec, then the counter will display the frequency directly in KHz, if it is taken as 1 µsec, the counter will display the frequency in MHz.

A very simple digital frequency meter is shown in figure 10.51, which is capable of measuring the frequency of the signal directly in Hz. The frequency to be measured is applied to one input of Schmitt trigger NAND gate 3. The gate period is controlled by a square wave (½ Hz sample pulse), which is connected to the second input of this gate. The ½ Hz pulse is generated from some external source. This pulse keeps the counter latching for 1 sec and counting is stopped for next 1sec. During the low period of this pulse, the display unit will show the updated counts directly in Hz. For getting the resetting pulse, the sample pulse is differentiated by R and C network of proper value, the negative peaks of the differentiated is clipped off using the switching diode D_2. The positive spike at the leading edge of the sample pulse is therefore obtained which is used to reset the counter. The counter and display circuit is basically the same as discussed in the object counter.

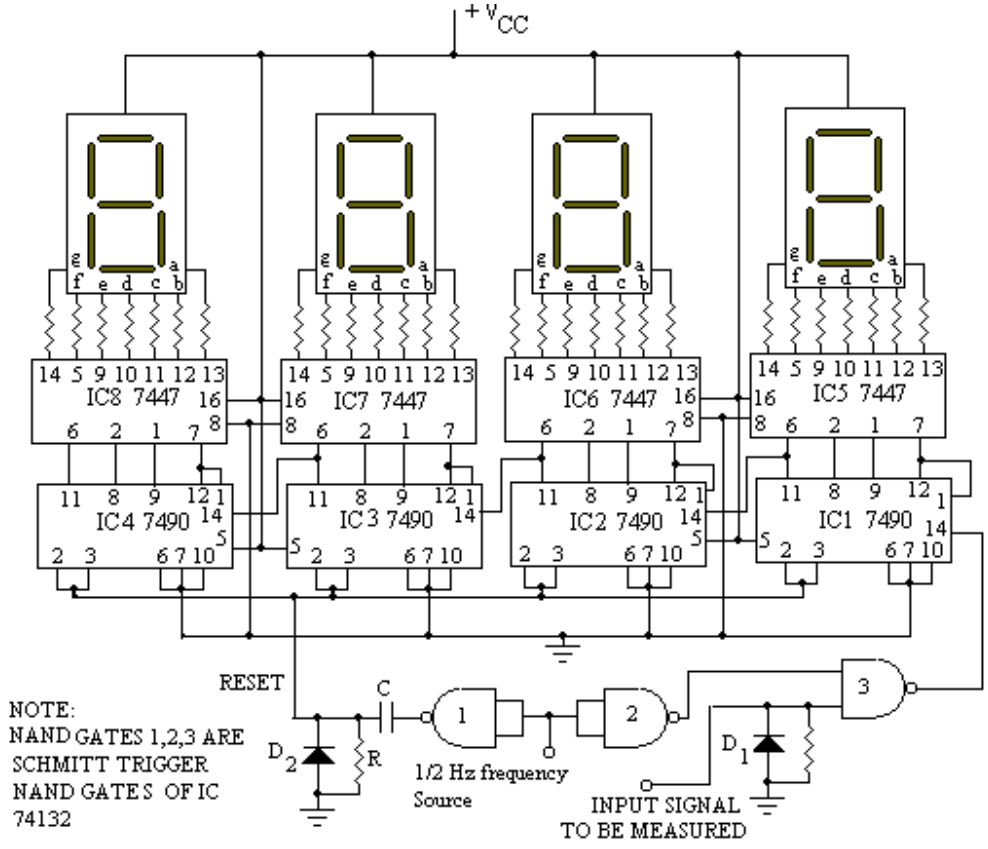


Fig. 10.51

10.12.3 Digital Clock

Digital clock makes use of the counter and decoding circuits. In digital clock the real time is displayed in the digital form. It displays Hours, minutes and seconds. The block diagram shown in figure 10.52 illustrates the design principle of digital clock. For the design of clock a 1 Hz continuous is obtained from some standard oscillator. The accuracy of the clock will depend on this standard oscillator. This 1 Hz pulse is counted by divide-by-sixty counter (a decade counter and a divide-by-six counter) which will count from 00 to 59 and reset to 00 at 60. The output of this counter will give a pulse for every 60 seconds (or 1 minute). The 1 cycle/minute pulses will again be counted by another divide-by-sixty counter which will be reset to 00 after the count 59 and thus a pulse for every one hour. The 1 cycle/hour pulses will now be counted by divide by twelve or divide-by-twenty four counter. The decoder and display devices (FNDs) are connected to the every counter circuit which gives the digital read out of the real time.

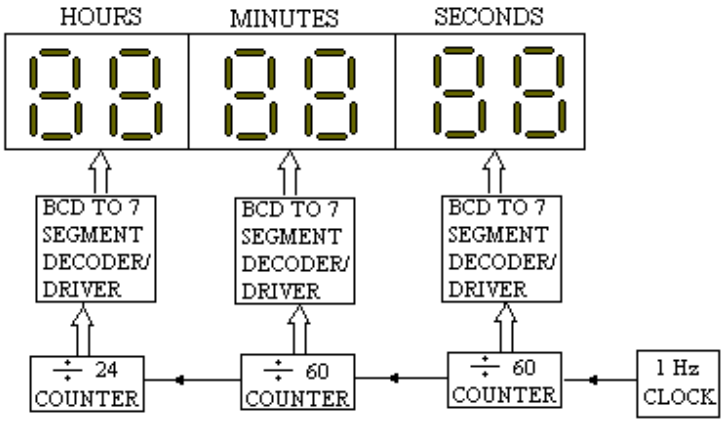


Fig. 10.52

The complete circuit diagram of digital clock (twenty hours) is shown in figure 10.53. It makes use of the counter circuits, designed using decade counter ICs 7490, BCD to seven segment decoder ICs 7447 and FNDs (common anode). The decade counter IC1 is wired in divide-by-ten mode, so that it resets to zero as soon as the tenth pulse is reached to counter. Similarly, the IC2 (decade counter IC 7490) is wired in divide-by-six mode so that when the sixth pulse occurs at the input of this IC, it resets to zero. At the 6^{th} pulse BCD output of the IC2 becomes 0110, so C B inputs (pins 8 and 9 of IC2) are connected to the pins 2 and 3 respectively to rest this counter at this pulse. The most significant bit is not be used as the counting in this case is limited to 5, so pin 11 of this IC is not used. The carry out signal for the next IC3 is therefore taken from pin 8 of IC2. The ICs 3^{rd} and 4^{th} are wired exactly in the similar way as the ICs 1^{st} and 2^{nd} respectively. The IC 5 and IC 6 are wired in divide-by-twenty four counter. The IC5 and IC6 are wired in divide by ten mode but these ICs are reset when the counting is 24 (0010 0100) i.e. when pin 8 of IC5 and pin 9 of IC6 are simultaneously are 1. These pins (pin 8 of IC5 and pin 9 of IC6) are connected to the reset pins of IC5 and IC6 as shown in figure 5.53. Now to each counter IC (7490) is connected to BCD to decimal decoder IC (7447) whose outputs are connected to different FNDs.

Two switches S1 and S2 are used for current time setting. These switches are push to ON switches. When switch S1 is pressed 1 Hz pulse directly goes to the minutes counter, the switch may be released when the minus are set. Similarly, switch S2 is used to set the Hours.

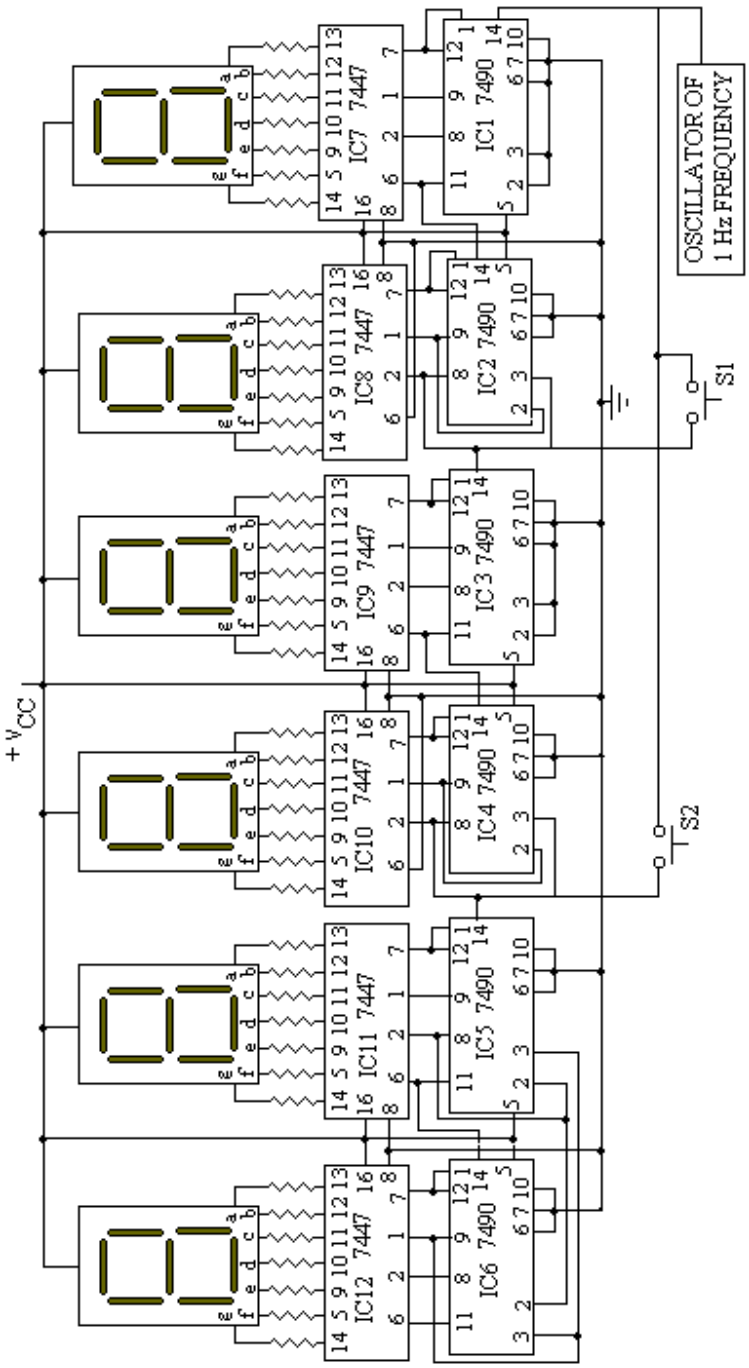


Fig. 10.52

10.12.4 Parallel to Serial Data Conversion

The counter circuits can also be used to convert the parallel data into serial form. The parallel data is applied to a multiplexer inputs and the serial data is taken at the output of the multiplexer. The data select terminals of the multiplexer is connected to a counter circuits, whose output drives the MUX and data at the output terminal of the multiplexer will be in serial form.

Figure 10.54 illustrates how eight bit parallel data is converted to serial data. For this 8:1 multiplexer is taken. The three data select terminals (S_0 to S_2) are connected to the output of mod-8 counter, which gives the data in binary from 000 to 111 on every clock pulse. So when counter's output is 000, the D0 data will be available at the multiplexer output. Similarly at the arrival of the 001 at the counter output second data D_1 will be transmitted to the MUX output and so on.

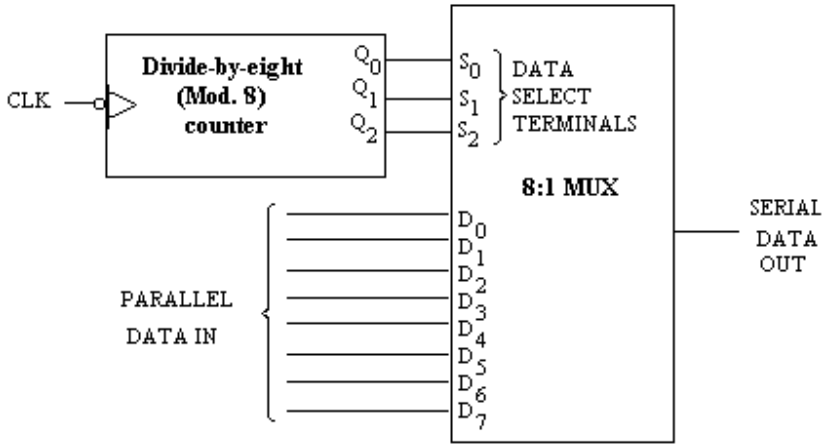


Fig. 10.54

PROBLEMS

1. What do you understand by counters? What is the difference between the asynchronous and synchronous counters?
2. Explain the meaning of counter. Draw the circuit of a 4-stage ripple counter and show the waveform at the various output stages.
3. Draw and explain the circuit of asynchronous binary counter (Mod-16). Also draw the wave shapes at different output stages.
4. Design a Mod-12 ripple counter and show the output states and wave forms of each flip-flop.
5. Design a Mod-14 ripple counter and show the output states and wave forms of each flip-flop.
6. Design an asynchronous decade counter and show the output states and wave forms of each flip-flop.
7. Design a Mod-16 ripple down counter and show the output states and wave forms of each flip-flop.
8. Discuss the design of a Mod-16 ripple up/down counter and show the output states and wave forms of each flip-flop.

9. Design a Mod-11 ripple counter and show the output states and wave forms of each flip-flop.
10. Discuss the design of a synchronous decade counter using T flip-flops and show the output states and wave forms of each flip-flop.
11. Repeat the problem 10 with J K flip-flops.
12. Design a Mod.-8 synchronous counter using J K flip-flops and show the output states and wave forms of each flip-flop.
13. Repeat the problem 12 with T flip-flops.
14. Design a synchronous binary counter (Mod-16) using J K flip-flops, and show the output states and wave forms of each flip-flop.
15. Repeat the problem 14 with T flip-flops.
16. Discuss the design of a synchronous decade counter using R S flip-flops and show the output states and wave forms of each flip-flop.
17. Discuss the design of synchronous decade counter using J K flip-flops; the counting is made in 2421 code. Show the output states and wave forms of each flip-flop.
18. Repeat the problem 17 using T flip-flops.
19. Design a synchronous decimal counter to count in excess-3 code. Use T flip-flops to design the counter. Show the output states and wave forms of each flip-flop.
20. Repeat the problem 19 using R S flip-flops.
21. Repeat the problem 19 using J K flip-flops.
22. Design a Mod-13 synchronous counter to count in natural binary sequence. Use T flip-flops to realize the circuit. Show the output states and wave forms of each flip-flop.
23. Repeat the problem 22 using R S flip-flops.
24. Repeat the problem 22 using J K flip-flops.
25. Design a controlled counter that can count Mod-5 if the control input is 0 and count Mod-8 if the control input is 1. Use J K flip-flops to realize the circuit. Also show the output states and wave forms of each flip-flop.
26. Repeat the problem 25 using T flip-flops.
27. Design a synchronous counter that can count in the following sequence 1, 3, 4, 5, 8, 9 ,0, 2, 6, 7 and repeats. Use J K flip-flops to realize the circuit. Also show the output states and wave forms of each flip-flop.
28. Repeat the problem 27 using T flip-flops.
29. Design a synchronous Mod-8 up/down counter use J K flip-flops to realize the circuit. Also show the output states and wave forms of each flip-flop.
30. Repeat the problem 29 using T flip-flops.
31. Design a synchronous Mod-7 up/down counter use J K flip-flops to realize the circuit. Also show the output states and wave forms of each flip-flop.
32. Design a synchronous Mod-6 up/down counter use J K flip-flops to realize the circuit. Also show the output states and wave forms of each flip-flop.
33. Design a circuit using a counter to generate the following pulse train 110100 and repeats.
34. Design a circuit using a counter to generate the following pulse train 011001 and repeats.
35. Discuss how a counter is used to convert the parallel data to serial data.

36. Discuss the design principle of digital frequency meter.
37. Discuss the design principle of digital clock.
38. How a four digit event counter is designed using the counters.

_____

11

DIGITAL TO ANALOG AND ANALOG TO DIGITAL CONVERTERS

Sometimes the information available for processing is in digital form while in most of the cases it is available in analog form. For example, the outputs of digital voltmeter, digital frequency meter, digital clock and calculators etc. are available in digital form but most physical quantities such as temperature, pressure, light, voltage and current etc. gives information in analog form. It is often necessary to convert information in one form to another form. For example, to convert the temperature (reading of mercury thermometer which is in analog form) in digital readout or in digital form, a transducer such as thermocouple or thermister is first used to convert the physical quantity to electrical quantity; an analog to digital converter then converts this quantity to digital form. Similarly, for plotting the output of a digital system on a curve plotter or X-Y recorder, the digital output is first converted to analog output with the help of digital to analog converter, the output of which drives a servomotor. So analog to digital (A/D) converters or digital to analog (D/A) converters are the interfacing devices which couple the digital system to analog or vice-versa. In this chapter various types of A/D and D/A converters will be discussed.

11.1 DIGITAL TO ANALOG CONVERTER

Digital to Analog (D/A) converter converts the digital information into analog form. The input may be of n-bit long having different voltage levels. So in the D/A converters some method is to be used which can convert this voltage level of n-bits to its equivalent analog form. This can be accomplished by using different resistive networks. Following two types of resistive networks are basically used for this purpose:

1. Resistive Divider Network or weighted resistor network
2. Binary Ladder Network or R-2R network

The converter which comprises the resistive divider network is known as Resistive divider D/A converter and the D/A converter which comprises the binary ladder network is known as binary ladder D/A converter. These converters will now be discussed separately.

11.1.1 Resistive Divider D/A converter

As discussed above the resistive divider D/A converter consists of a resistive divider network, so before discussing the complete circuit diagram of a resistive divider D/A converter it is better to understand the working of resistive divider network. The

resistive divider network changes each of the n-bit digital level into its equivalent analog output. The discussion is now made for the method of converting the n-bit digital input to its equivalent analog signal. A weight is assigned to each bit of n-bit digital input in such a way that the sum of weight must be equal to 1. In general, the binary weight assigned to LSB in an n-bit digital input is $\frac{1}{2^n - 1}$. The weights assigned to 2^{nd} LSB, 3^{rd} LSB, 4^{th} LSB and so on are obtained by multiplying the weight of LSB to 2^1(=2), 2^2 (=4), 2^3 (8)…. respectively. For instance, weights assigned to different bits of 4-bit binary input $b_3\ b_2\ b_1\ b_0$ are:

Weight assigned to LSB (b_0 bit) is $\frac{2^0}{2^4 - 1} = \frac{1}{15}$

Weight assigned to 2^{nd} LSB (b_1 bit) is $\frac{2^1}{2^4 - 1} = \frac{2}{15}$

Weight assigned to 3^{rd} LSB (b_2 bit) is $\frac{2^2}{2^4 - 1} = \frac{4}{15}$

Weight assigned to MSB (b_3 bit) is $\frac{2^3}{2^4 - 1} = \frac{8}{15}$

The sum of weights assigned to each bit of 4-bit digital input is 1 as $\frac{1}{15} + \frac{2}{15} + \frac{4}{15} + \frac{8}{15} = 1$.

In a four bit binary system there will be 16 different possible input combinations, corresponding to which the analog signal will be obtained if it is assumed that a certain reference voltage (V_{REF}) is applied whenever there is a 1 in binary bit. In a 4 bit digital system if V_{REF} =15 volts, the analog voltage available for each combination of binary input should be as given in table 11.1.

Table 11.1

b_3	b_2	b_1	b_0	Weight	Analog voltage
0	0	0	0	0/15	(0/15) V_{REF} = 0 Volt
0	0	0	1	1/15	(1/15) V_{REF} = 1 Volt
0	0	1	0	2/15	(2/15) V_{REF} = 2 Volts
0	0	1	1	3/15	(3/15) V_{REF} = 3 Volts
0	1	0	0	4/15	(4/15) V_{REF} = 4 Volts
0	1	0	1	5/15	(5/15) V_{REF} = 5 Volts
0	1	1	0	6/15	(6/15) V_{REF} = 6 Volts
0	1	1	1	7/15	(7/15) V_{REF} = 7 Volts
1	0	0	0	8/15	(8/15) V_{REF} = 8 Volts
1	0	0	1	9/15	(9/15) V_{REF} = 9 Volts
1	0	1	0	10/15	(10/15) V_{REF} = 10 Volts
1	0	1	1	11/15	(11/15) V_{REF} = 11 Volts
1	1	0	0	12/15	(12/15) V_{REF} = 12 Volts
1	1	0	1	13/15	(13/15) V_{REF} = 13 Volts
1	1	1	0	14/15	(14/15) V_{REF} = 14 Volts
1	1	1	1	15/15	(15/15) V_{REF} = 15 Volts

So the analog voltage for binary word = (weight of the binary word) x V_{REF}

It may be noted from this table 11.1 that the analog voltage corresponding to binary equivalent is discrete step value as given in figure 11.1. The discrete step is of 1 volt if V_{REF} is assumed to be 15 volts in a four bit digital input. The step voltage (analog)

will be dependent on the reference voltage. There will, however, be 2^n steps in n-bit digital system.

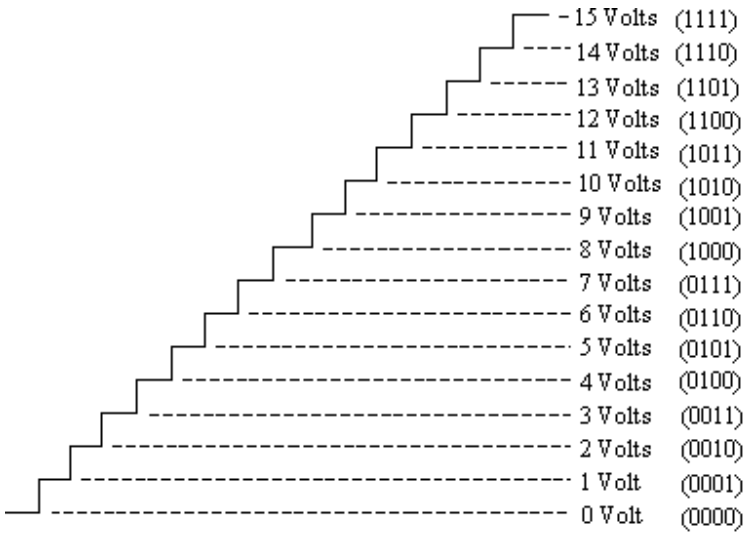


Fig. 11.1

Resistive divider network is used for converting digital inputs to analog outputs. The network for 6 bit binary system shown in figure 11.2 is known as the weighted network, as the resistors are weighted inversely with their current values. The input binary bits are b_5 b_4 b_3 b_2 b_1 b_0 where b0 is the LSB and b5 is MSB. These binary bits may be logic 0 or 1. Logic 0 may further be assumed as 0 volt and logic 1 as V_{REF}. So V_0, V_1, V_2, V_3, V_4 and V_5 are the input voltage levels which may be 0 volt or VREF depending on the binary bits. The resistors R_0, R_1, R_2, R_3, R_4 and R_5 are connected to bits b_0, b_1, b_2 , b_3, b_4, b_5 respectively. It may be noted from this network that the resistor connected to the binary bit is half the value of resistor connected to the previous (lower bit). Hence this network also called as the resistive divider network. Let R_L is the load resistance which is supposed to very high i.e. very much higher than the resistor R_0.

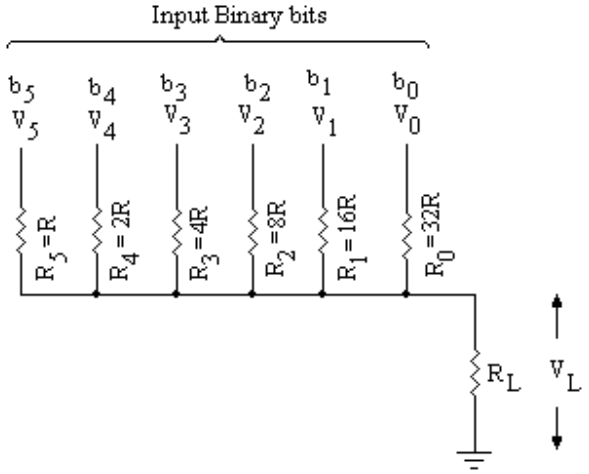


Fig. 11.2

Now the voltage V_L across the load resistance R_L can be obtained by using Millman's theorem. This theorem states that the voltage appearing at any node in a resistive network is equal to the sum of all the currents that would enter to the node divided by the sum of conductances connected to that node.

Thus
$$V_L = \frac{\frac{V_5}{R_5}+\frac{V_4}{R_4}+\frac{V_3}{R_3}+\frac{V_2}{R_2}+\frac{V_1}{R_1}+\frac{V_0}{R_0}}{\frac{1}{R_5}+\frac{1}{R_4}+\frac{1}{R_3}+\frac{1}{R_2}+\frac{1}{R_1}+\frac{1}{R_0}}$$

$$= \frac{\frac{V_5}{R}+\frac{V_4}{2R}+\frac{V_3}{4R}+\frac{V_2}{8R}+\frac{V_1}{16R}+\frac{V_0}{32R}}{\frac{1}{R}+\frac{1}{2R}+\frac{1}{4R}+\frac{1}{8R}+\frac{1}{16R}+\frac{1}{32R}}$$

$$= \frac{\frac{[32V_5+16V_4+8V_3+4V_2+2V_1+V_0]}{32}}{\frac{[32+16+8+4+2+1]}{32}}$$

$$= \frac{V_0+2V_1+4V_2+8V_3+16V_4+32V_5}{63}$$

$$= \frac{1}{(2^6-1)}(2^0V_0+2^1V_1+2^2V_2+2^3V_3+2^4V_4+2^5V_5) \quad \ldots(11.1)$$

In this equation (11.1) the load resistance R_L is not considered as it is assumed to be large enough offering low (almost zero) conductance. From this equation it is clear that if the input binary bits are all 1 (in a six bit system) and reference voltage V_{REF} = 6.4 volts (say), the V_L is given by:

$$V_L = \frac{1}{63} x 63 V_{REF} = 6.4 volts$$

In general, the equation (11.1) for output voltage of n-bit binary digits is given as:

$$V_L = \frac{1}{(2^n-1)}(2^0V_0+2^1V_1+2^2V_2+2^3V_3+2^4V_4.....+2^{n-1}V_{n-1})$$

$$\ldots(11.2)$$

The output of this network is as per our requirement, and is proportional to the input binary data.

Using the network discussed above, a D/A converter (called binary weighted D/A converter or Resistive divider D/A converter) can be designed as given below. The schematic diagram of 6-bit D/A converter is shown in figure 11.3. It consists of the following major parts.

(i) n switches, one for each bit applied to the input,

(ii) A binary weighted resistive network which changes each of the digital level into equivalent binary weighted voltage or current.

(iii) A reference voltage source V_{REF}.

(iv) A summing amplifier that adds the currents flowing in the resistors of the network to develop a signal that is proportional to the digital input.

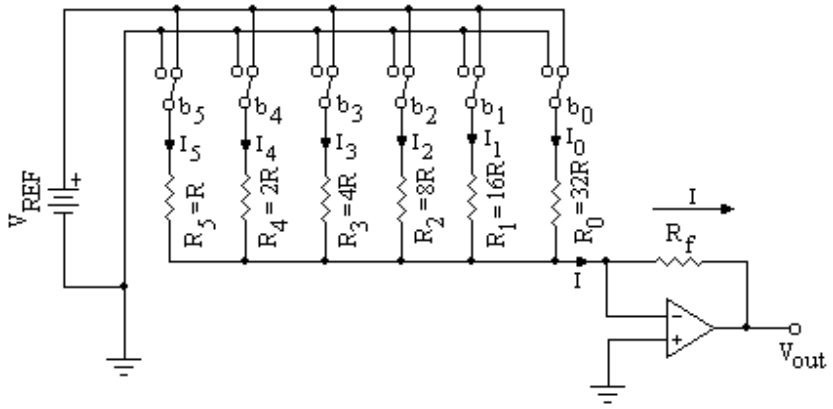


Fig. 11.3

In this circuit, one switch is connected to each binary bit. Infact these switches are such that when the binary bit is 0, the corresponding resistor of the network gets connected to the ground potential and when the binary bit is 1, the corresponding resistor of the network gets connected to the V_{REF} volt. The current flowing through any branch of the network will be the logical voltage (0volt or V_{REF} volts) divided by the corresponding resistor.

So the total current I will be given by (ref. fig. 11.3):

$$I = \frac{V_5}{R_5} + \frac{V_4}{R_4} + \frac{V_3}{R_3} + \frac{V_2}{R_2} + \frac{V_1}{R_1} + \frac{V_0}{R_0}$$

$$= \frac{V_5}{R} + \frac{V_4}{2R} + \frac{V_3}{4R} + \frac{V_2}{8R} + \frac{V_1}{16R} + \frac{V_0}{32R}$$

Since the voltages V_5 through V_0 are either 0 volt or V_{REF} volts depending upon the bit value, so it customary to take common voltage V_{REF} and bits are kept in place of voltages. So V_5 is replaced by $V_{REF}.b_5$, V_4 by $V_{REF}.b_4$ and so on; the bits b5, b4, b3 etc will be 0 or 1. The current I may, therefore, be represented as follows:

$$I = \frac{V_{REF}}{32R}\left[32b_5 + 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0\right]$$

$$= \frac{V_{REF}}{2^5.R}\left[2^0 b_0 + 2^1 b_1 + 2^2 b_2 + 2^3 b_3 + 2^4 b_4 + 2^5 b_5\right]$$

This is the equation of current *I* for 6 input bits. The general equation of current *I* for n input bits is given by:

$$I = \frac{V_{REF}}{2^{n-1}.R}\left[2^0 b_0 + 2^1 b_1 + 2^2 b_2 + 2^3 b_3 + 2^4 b_4 ..... + 2^{n-1} b_{n-1}\right]$$

…(11.3)

The voltage at the output of operational amplifier will be given by:

$$V_{out} = -R_f.I$$

The resistor R_f is the feed back resistance in the operational amplifier. The output voltage of the operational amplifier is proportional to input binary data.

The switches connected in figure 11.3 can be replaced by the electronic switches (transistorized) as shown in figure 11.4. When the bit is at logic 1, the corresponding transistor conducts and the current flows through the collector resistor as required; and when the bit is at logic 0 the transistor goes into cutoff and no collector current flows.

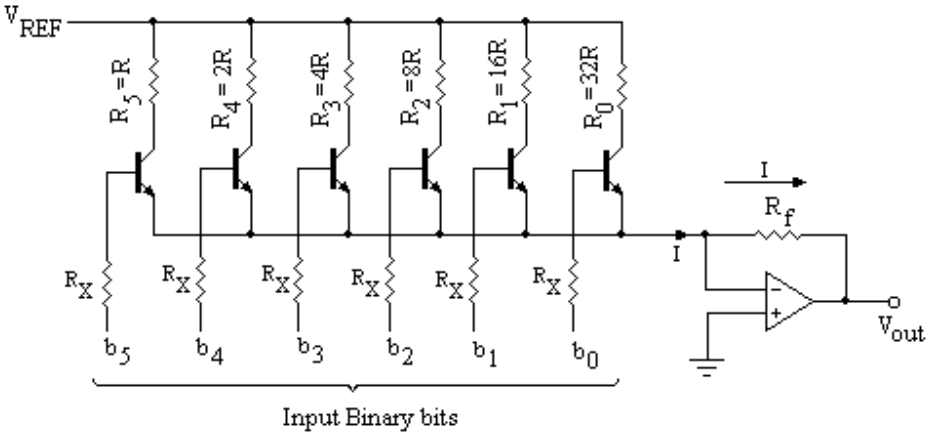


Fig. 11.4

This D/A converter is economical and simple method to design but suffers the following serious drawbacks:

1. The network in this D/A converter is constructed using the precession resistors and resistor has a different value. So it is difficult in practice to choose the resistors with accuracy and stability.
2. When the number of bits in the network is large, then the current from the source will be large enough. The current in the LSB branch (resistor) will be much larger than MSB branch. In a 10 bit D/A converter, the current in LSB branch will be 512 times larger than the MSB branch.

Example 11.1: A 6 bit resistive divider network has 10 volts full scale output, find output voltage for an input of 110110.

Solution: $V_{REF} = 10$ volts

The output voltage for 6 bit resistive divider network is given by:

$$V_L = \frac{1}{(2^6 - 1)}(2^0 V_0 + 2^1 V_1 + 2^2 V_2 + 2^3 V_3 + 2^4 V_4 + 2^5 V_5)$$

$$= \frac{V_{REF}}{63}(2^0 b_0 + 2^1 b_1 + 2^2 b_2 + 2^3 b_3 + 2^4 b_4 + 2^5 b_5)$$

$b_0 = 0$, $b_1 = 1$, $b_2 = 1$, $b_3 = 0$, $b_4 = 1$ and $b_5 = 1$

So $$V_L = \frac{10}{63}(2^0.0 + 2^1.1 + 2^2.1 + 2^3.0 + 2^4.1 + 2^5.1)$$

$$= \frac{10}{63}(0 + 2 + 4 + 0 + 16 + 32)$$

$$= \frac{540}{63} = 8.57 \ volts$$

Example 11.2: A 5-bit resistive divider D/A converter has a resistor of 10 KΩ in MSB branch. The reference voltage is 10 volts. The resistance in the feedback path of the operational amplifier is 5 KΩ. What will be the output for 11010 input?

Solution: V_{REF} = 10 volts R = 10 KΩ and R_f = 5 KΩ

$$I = \frac{V_{REF}}{2^{n-1}.R}\left[2^0 b_0 + 2^1 b_1 + 2^2 b_2 + 2^3 b_3 + 2^4 b_4 ..... + 2^{n-1} b_{n-1}\right]$$

$$= \frac{10}{16x10K\Omega}\left[2^0.0 + 2^1.1 + 2^2.0 + 2^3.1 + 2^4.1\right]$$

$$= \frac{10\ x\ 26}{16\ x10\ K\ \Omega} = 1.625\ \ mA$$

$$V_{out} = -R_f.I = -1.625\,mAx\,5K\Omega = -8.125\,volts$$

11.1.2 Binary Ladder D/A Converter

A more commonly used D/A converter is a binary ladder D/A converter, which removes the drawbacks discussed in resistive divider D/A converter. This type of D/A converter contains an R-2R ladder network. The R-2R resistive ladder network will now be discussed, which gives the output a weighted sum of digital inputs. Such a ladder network for 4-bit input is shown in figure 11.5. This network is constructed having only two resistor values i.e. R and 2R. In this network b_0, b_1, b_2 and b_3 are the input binary bits and b_0 is the LSB and MSB is b_3. Any of these bits will be at the ground potential when the corresponding bit is at logic 0 or at the reference potential (V_{REF}) when the input bit is at logic 1.

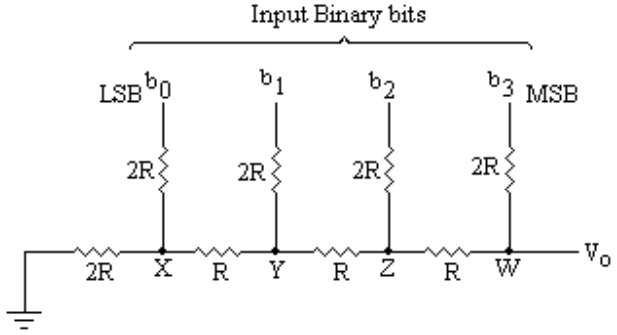


Fig. 11.5

To examine the behaviour of this network, it is assumed that the bit b_3 is at logic 1 (or V_{REF} potential) as shown in figure 11.6(a). The output voltage corresponding to MSB may be calculated as follows. The equivalent resistance at the point X is the parallel combination of two resistances each having the value of 2R. So the equivalent resistance looking at point X and ground is R as shown in figure 11.6(b). At the point Y again there is a parallel combination of two 2R resistances; the equivalent resistance looking at the point Y and ground is R as shown in figure 11.6(c). Similarly, one can find the equivalent resistance looking at the point Z and ground is R as shown in figure 11.6(d).

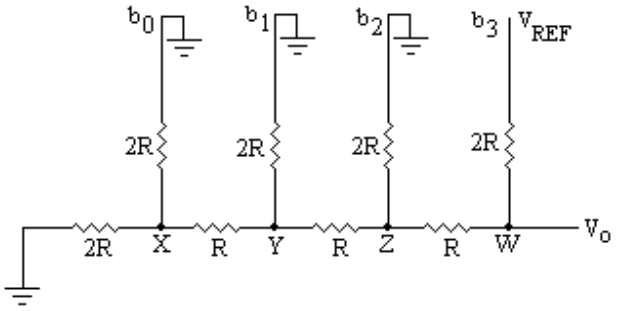
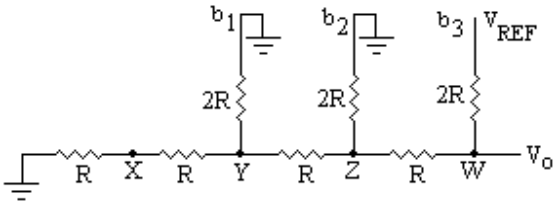
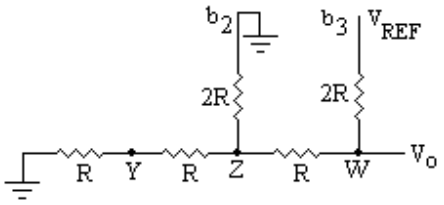


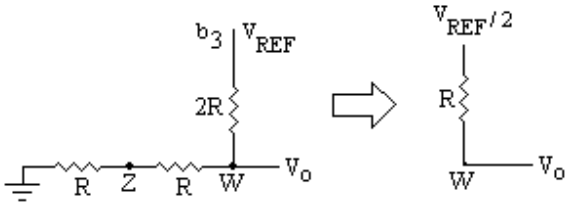
Fig. 11.6(a)



(b)



(c)

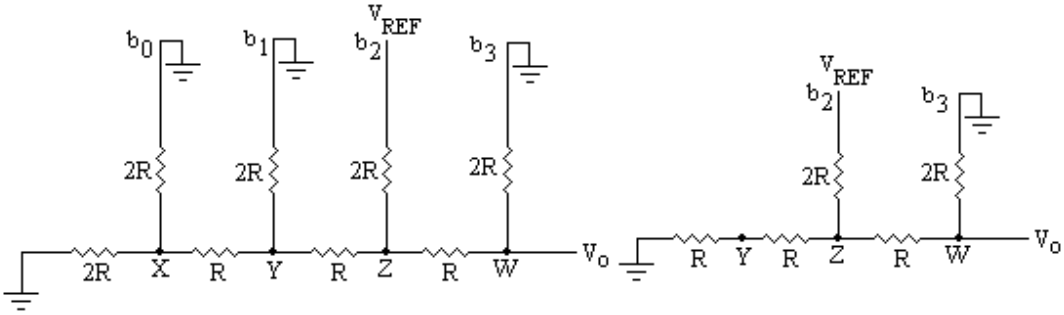


(d)

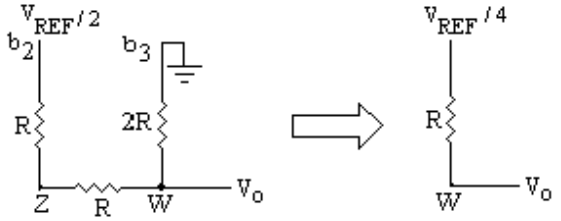
From figure 11.6(d) it is clear that the resistance looking at the point W and ground is 2R, and the resistance looking towards the bit b_3 is also 2R. Thus the output voltage at the point W due to bit b_3 (MSB) assumed at V_{REF} potential is given by:

$$V_0 = \frac{V_{REF}\, x 2R}{(2R + 2R)} = \frac{V_{REF}}{2}$$

The output voltage V_0 due to the binary input 1000 (only MSB is high) is half of the reference voltage having Thevenis's resistance R in series with it. Similarly one can calculate the output voltage due to the binary input 0100 (i.e. second MSB); the network for this case is shown in figure 11.7(a). The resistance looking at the point Y and ground is R as shown in figure 11.7(b). The resistance between the point Z and ground is 2R. The voltage at point Z and ground is (V_{REF}/ 2) have a Thevenin's resistance R, as shown in figure 11.7(c).



(a) (b)



(c) **Fig. 11.7**

From this figure the output voltage V_0 at the point W is given by:

$$V_0 = \frac{(V_{REF}/2)x2R}{(2R+2R)} = \frac{V_{REF}}{4}$$

So the output voltage due to second MSB (or for binary input 0100) is $\frac{V_{REF}}{4}$ with Thevenin's resistance R in series with it.

It can further be shown that the output due to third MSB (for binary input 0010) is $\frac{V_{REF}}{8}$. And for LSB (0001 binary input) the output is $\frac{V_{REF}}{16}$. Each voltage source will have Thevenin's resistance R in series with the source. The total output voltage in analog form, due to all the inputs as 1 (for 1111) can easily be found by adding the outputs obtained for each bit as given below:

$$V_0 = \frac{V_{REF}}{2} + \frac{V_{REF}}{4} + \frac{V_{REF}}{8} + \frac{V_{REF}}{16}$$

It may be noted that $\frac{V_{REF}}{2}$ is the voltage due to MSB, $\frac{V_{REF}}{4}$ due to second MSB, $\frac{V_{REF}}{8}$ for third MSB and $\frac{V_{REF}}{16}$ for LSB. So to distinguish these voltages it is useful to write the bit positions along with V_{REF} as given below. So if the bit is 0 the voltage corresponding to that bit will be zero otherwise the voltage as discussed above.

$$V_0 = \frac{V_{REF}xb_3}{2} + \frac{V_{REF}xb_2}{4} + \frac{V_{REF}xb_1}{8} + \frac{V_{REF}xb_0}{16}$$

$$= \frac{V_{REF}}{16}\left[8xb_3 + 4xb_2 + 2xb_1 + 1xb_0\right]$$

$$= \frac{V_{REF}}{2^4}\left[2^0 xb_0 + 2^1 xb_1 + 2^2 xb_2 + 2^3 xb_3\right] \quad \ldots(11.4)$$

The equation (11.4) is the equation for voltage at the output of 4 bit binary ladder network. A general equation for the output of n-bit binary data can be given as follows:

$$V_0 = \frac{V_{REF}}{2^n}\left[2^0 xb_0 + 2^1 xb_1 + 2^2 xb_2 + 2^3 xb_3 + ... + 2^{n-1} xb_{n-1}\right] \quad \ldots(11.5)$$

The output of this network is proportional to the input binary data. So using this R-2R ladder network, a D/A converter (called binary ladder D/A converter) can be

designed as given below. The schematic diagram of 4-bit D/A converter is shown in figure 11.8. It consists of the following major parts.

(i) n switches, one for each bit applied to the input,

(ii) A binary ladder network which changes each of the digital level into equivalent binary weighted voltage or current.

(iii) A reference voltage source V_{REF}.

(iv) A summing amplifier that adds the currents flowing in the resistors of the network to develop a signal that is proportional to the digital input.

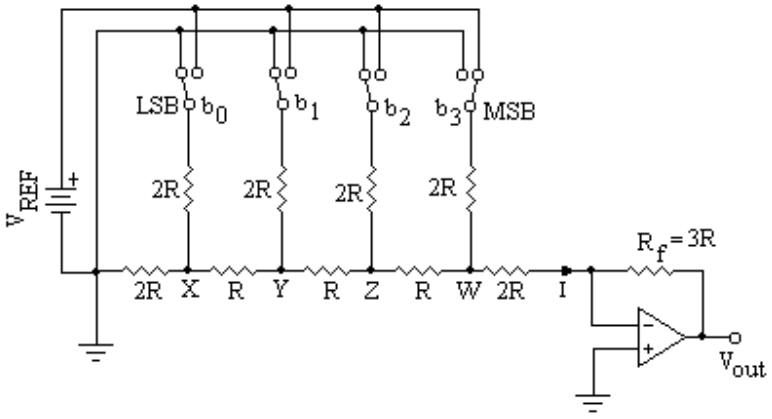


Fig. 11.8

The output voltage V_{out} of this D/A converter due to MSB (1000 binary input) will be calculated as given below:

The voltage at the point W due to MSB is $V_{REF} / 2$ having a Thevenin's resistance R in series with it as discussed above and is shown in figure 11.9

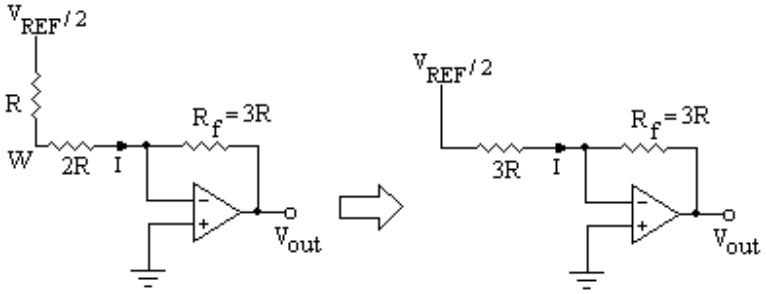


Fig. 11.9

From this figure, the current I is given by:

$$I = \frac{V_{REF}}{2}(\frac{1}{3R})$$

and the output voltage V_{out} is given by:

$$Vout = -I.R_f$$

$$= -\frac{V_{REF}}{2}(\frac{1}{3R}).3R = -\frac{V_{REF}}{2}$$

The output voltage is the same as calculated in equation 11.5, with the difference that it has a negative value because the operational amplifier is used in inverting configuration.

Note that the resistors in the ladder network are either R or 2R. It is the ratio of resistances matters rather than the absolute value of resistances. Further the resistors do not cover a wide range of magnitude; it is therefore practically possible to get the precision in the ratio of their magnitudes. The temperature coefficients of these resistances can easily match. Because of these advantages, the ladder network is widely used in D/A converters.

Example 11.3 For a five-bit binary ladder D/A converter the input levels are 0 = 0 volt and 1 = + 10 volts, find

(i) the output voltages caused by each bit
(ii) the output voltage corresponding to an input of 10110
(iii) the full scale output voltage of the ladder.

Solution:

(i) The output voltage caused by MSB $= -\frac{V_{REF}}{2} = -\frac{10}{2} = -5 volts$

The output voltage caused by 2nd MSB $= -\frac{V_{REF}}{4} = -\frac{10}{4} = -2.5 volts$

The output voltage caused by 3rd MSB $= -\frac{V_{REF}}{8} = -\frac{10}{8} = -1.25 volts$

The output voltage caused by 4th MSB $= -\frac{V_{REF}}{16} = -\frac{10}{16} = -0.625 volt$

The output voltage caused by LSB $= -\frac{V_{REF}}{32} = -\frac{10}{32} = -0.3125 volt$

(ii) The output voltage corresponding to an input of 10110 is

$$V_0 = -\frac{V_{REF}}{2^5}\left[2^0 xb_0 + 2^1 xb_1 + 2^2 xb_2 + 2^3 xb_3 + 2^4 xb_4\right]$$

$$= -\frac{10}{32}\left[2^0 x0 + 2^1 x1 + 2^2 x1 + 2^3 x0 + 2^4 x1\right]$$

$$= -\frac{10}{32}\left[2^0 x0 + 2^1 x1 + 2^2 x1 + 2^3 x0 + 2^4 x1\right]$$

$$= -\frac{10}{32}\left[2 + 4 + 16\right] = -\frac{220}{32} = -6.875 volts$$

(iii) The full scale output voltage is given by:

$$= -\frac{10}{32}\left[2^0 x1 + 2^1 x1 + 2^2 x1 + 2^3 x1 + 2^4 x1\right]$$

$$= -\frac{10}{32}\left[1 + 2 + 4 + 8 + 16\right] = -\frac{310}{32} = -9.6875 volts$$

11.2 PERFORMANCE CRITERIA FOR D/A CONVERTER

The D/A converters are available in the form of ICs with different specifications for their performances. So before discussing D/A converter ICs it will be better to discuss first the

characteristics of the converters specified by the manufacturers. These specifications include:

1. Resolution
2. Accuracy
3. Monotonicity
4. Settling time

1. **Resolution:** As discussed above, the analog output of D/A converter is proportional to the digital input (binary data), so a perfect staircase is obtained if there is an LSB increment. The resolution is, therefore, a measure of quality of D/A converter, which is defined as the ratio of the LSB increment to the maximum output. For an n-bit D/A converter the resolution is given by:

The change in output due to LSB increment for n-bit digital input (Step size)

= Full scale output / No. of steps

$$= \frac{\text{Full scale output}}{2^n - 1}$$

where $(2^n - 1)$ is the number of steps for n-bit D/A converter.

$$\text{Percentage Resolution} = \frac{\text{Full scale output} / (2^n - 1)}{\text{Full scale output}} \text{ x } 100\%$$

$$= \frac{1}{2^n - 1} \text{ x } 100\%$$

The step size for a 10 bit D/A converter, having full scale output voltage as 10 volts, is given by $= \frac{10}{2^{10} - 1} = \frac{10}{1023} = 9.8mV$

And % Resolution = 0.0978%

2. Accuracy: Accuracy of a D/A converter is the closeness of the output analog voltage to the expected theoretical output. In a linear variation of analog output with digital input, the relative accuracy is the maximum deviation of the D/A output compared with the linear behaviour. It is expressed as a percent of a full-scale or maximum output voltage. For example, if a converter has a full scale output of 10 V and the accuracy is $\pm 0.1\%$, then the maximum error for any output voltage is (10V)(0.001) = 10 mV. Ideally, the accuracy should be at most ± ½ of an LSB.

For an 8 bit D/A converter, one LSB is $\frac{1}{256} = 0.0039 = 0.39\%$ of full scale. The accuracy should be approximately $\pm 0.2\%$.

3. Monotonicity: A D/A converter is said to be monotonic if it gives an analog output voltage which increases regularly and linearly with increase in input digital signal. Such a quality of the converter is called as monotonicity. In order to demonstrate monotonicity of a D/A converter, a counter output is given as digital input to a D/A converter and the analog output is displayed on the CRO. Monotonicity then requires that the output waveform should be a perfect staircase waveform with steps equally spaced and of same amplitude. If the steps are missing or have varying amplitude, the D/A converter is defective.

4. Settling Time: After the application of digital input to a D/A converter, it takes about few nanoseconds to microseconds to produce the correct output. So the settling time is defined as the time the converter takes to give an output to settle within ± ½ LSB of its final value. For example, if a D/A converter has a resolution of 10mV, the settling time is the measure of the time the converter takes to settle with in ±5mV of its final value. Figure 11.10 illustrates the settling time in a D/A converter. The settling time is important because it places a limit on how fast one can change the digital input. The settling time depends on the stray capacitance, saturation delay time, and other factors.

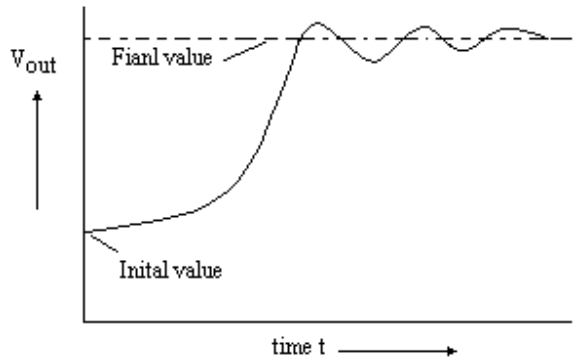


Fig. 11.10

Example 11.4 What is the step size (or resolution in volts) of a 12 bit D/A converter, if the full scale output is +10 volts? Find the percentage resolution also.

Solution: Here n = 12

So step size is given by

$$\text{Step size} = \frac{\text{Full scale output}}{2^n - 1}$$

$$= \frac{10}{2^{12} - 1} = \frac{10}{4095} = 2.44mV$$

$$\text{Percentage Resolution} = \frac{\text{Full scale output} / (2^n - 1)}{\text{Full scale output}} \text{ x } 100\%$$

$$= \frac{1}{2^{12} - 1} x100 = \frac{100}{4095} = 0.0244\%$$

Example 11.5 How many bits are required at the input of a D/A converter to achieve a resolution of 10mV, if the full scale output is 10 volts?

Solution:

$$\text{Resolution} = \frac{\text{Full scale output}}{2^n - 1}$$

$$10mV = \frac{10}{2^n - 1}$$

$$2^n - 1 = \frac{10}{10x10^{-3}} = 1000$$

$$2^n = 1001 \cong 2^{10}$$

$$n = 10$$

So the number of bits required = 10

11.3 D/A CONVERTER IC 0808

There are many commercially available D/A converter ICs. The IC 0808 is the most popular, inexpensive and widely used 8 bit D/A converter. It contains a reference current source, an R-2R binary ladder network and 8 transistor switches to steer the binary currents to the network. Figure 11.11 shows the pin configuration of this D/A converter IC 0808.

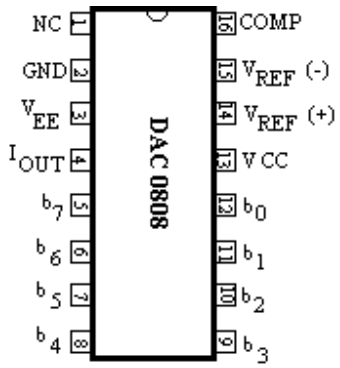


Fig. 11.11

In this IC pin 5 through 12 are the 8 bit input data so should be connected to input data bits. Pin 15 is to be connected to ground through a resistance. Pin 13 is to be connected to +5 volt supply. Pin 3 (VCC) is to be connected to – 15 volts. Pin 4 is the output current of the ladder network should be connected to the operational amplifier. Pin 2 is the ground pin. The pin 16 is the frequency compensation pin, a capacitor between pin 16 and 3 is to be connected for this purpose.

A circuit diagram to get the analog output voltage corresponding to 8 bit digital input is shown in figure 11.12. A +5V supply sets up a reference current of 2mA for the ladder. The output current I_{out} drives the operational amplifier to give final output between 0 and 2 volts (approximately) for the 8 bit digital input.

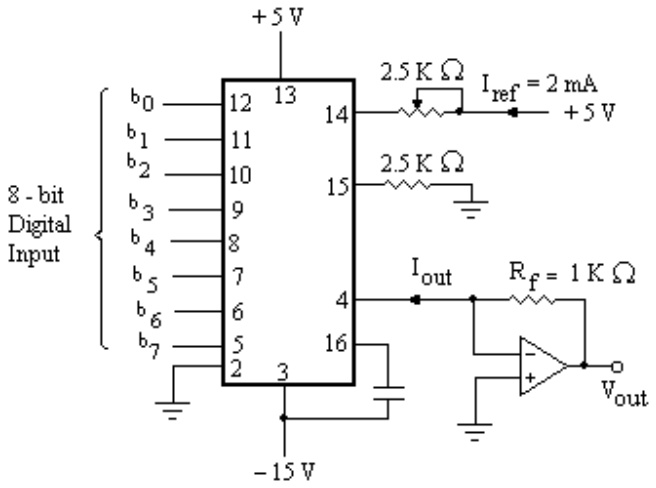


Fig. 11.12

11.4 ANALOG TO DIGITAL CONVERTER

Generally the information to be processed by the digital systems is in the analog form. So before applying such signals to the digital systems it is necessary to convert the

signal into its equivalent digital form. The method with the help of which the analog signal may be converted to digital form is known as digital to analog (D/A) converter. The A/D converter is more complex and difficult than the D/A converter. Followings are the different methods for A/D converter, which will be discussed in the next sections.

(i) Simultaneous A/D converter
(ii) Successive approximation D/A converter
(iii) Counter or Digital Ramp type A/ D converter
(iv) Single slope D/A converter
(v) Dual slope D/A converter

11.5 SIMULTANEOUS A/D CONVERTER

This is the fastest and simplest method of converting an analog signal to digital signal. It utilizes the parallel differential comparators; the input analog voltage is compared by these comparators with known voltages called as reference voltages. The comparators gives the low output (logic 0) when the input is less than the reference voltage and gives the high output (logic 1) when the input analog voltage exceeds the reference voltage. This method of conversion is also called as Flash or parallel type A/D converter.

For the conversion of analog voltage ranging between 0 to V volts into two bit digital output, three comparators (in general $2^n - 1$ comparators where n in the number of bits) are required. The input analog voltage is converted to the 4 (in general 2^n) equal regions as shown in figure 11.13. If the analog voltage is lying in the first region, then the

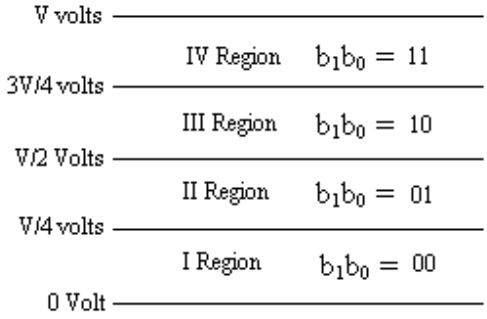


Fig. 11.13

binary bits (b_1, b_0) are 00, similarly to second, third and forth regions the binary bits are 01, 10 and 11 respectively. The reference voltages to the three comparators C_0, C_1, C_2 should be V/4, V/2, 3V/4 respectively as shown in figure 11.14. The output of the three comparators should be connected to the logic gates to produce the desired binary output. The read gates and output registers are used to read the digital output.

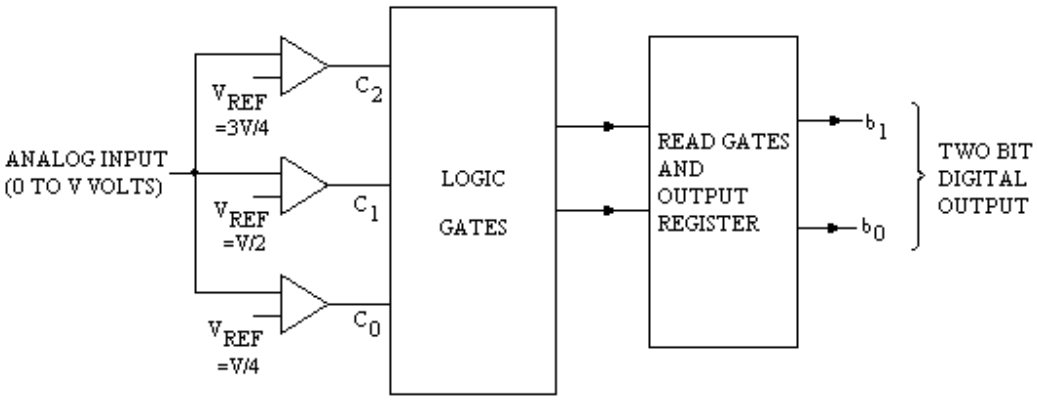


Fig. 11.14

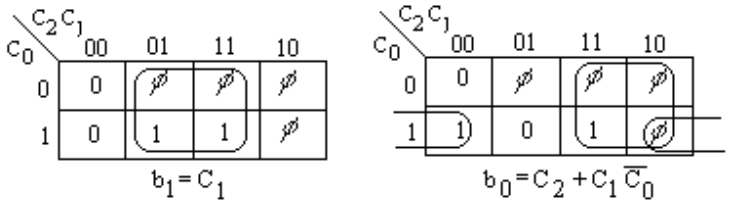
Referring to figures 11.13 and 11.14, if the input analog voltage exceeds the reference voltage to any comparator, the comparator gives high output (logic 1); if on the other hand if the input analog voltage is less than the reference voltage of the comparator, it gives low output (logic 0). In this way if all the comparators give low output, the analog input voltage must be between 0 and V/4 volts (I region) and digital binary output should be 00. If the C_0 is high and C_1 and C_2 are low, the input must be between V/4 and V/2 volts (II region) and digital binary output should be 01. If C_0 and C_1 are high and C_2 is low, the input must be between V/2 and 3V/4 volts (III region) and digital binary output should be 10. Finally if all the comparators give high outputs, the input must lies 3V/4 and V volts (IV region) and digital binary output should be 11. Table 11.2 summarizes outputs of the comparators.

Table 11.2

Input voltage	Comparator output			Binary output	
	C_2	C_1	C_0	b_1	b_0
0 to V/4	0	0	0	0	0
V/4 to V/2	0	0	1	0	1
V/2 to 3V/4	0	1	1	1	0
3V/4 to V	1	1	1	1	1

By drawing the K –maps (figure 11.15), the expressions for b_0 and b_1 are obtained as:

$$b_1 = C_1 \qquad \text{and} \qquad b_0 = C_1 \cdot \overline{C}_0$$



(a) **(b)**

Fig. 11.15

These expressions may be realized using the gates as shown in figure 11.16. The output may be reset by applying high signal to the reset line and to read the data a high signal is applied to the read line.

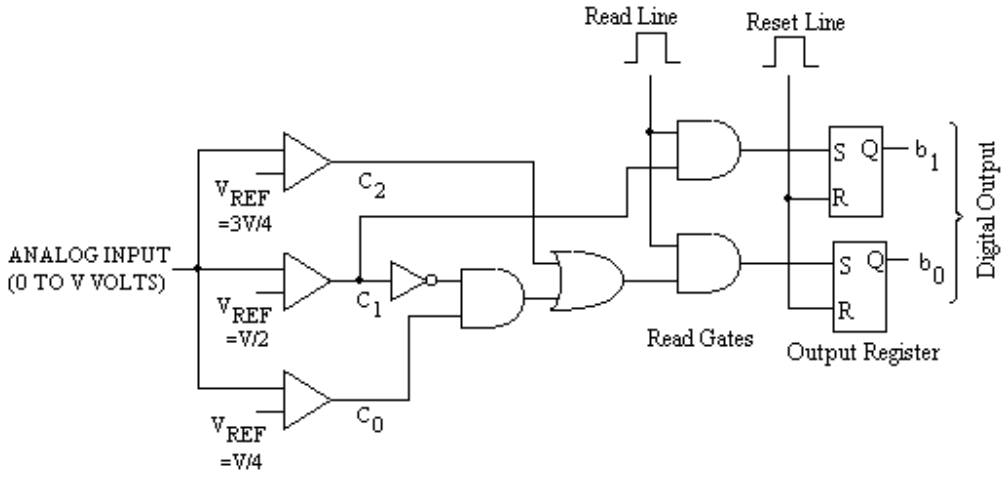


Fig. 11.16

For the conversion of analog input voltage (0 to V volts) into three bit binary output we proceed in the similar method as for the two bits output. For the three bits outputs the input voltages are divided into 8 (as $2^3 = 8$) equal regions and 7 (as $2^3-1 = 7$) comparators are to be used. So the logic circuit to be designed should have seven inputs (output of the seven comparators) and three outputs. The output of comparators and the corresponding binary output are shown in table 11.3.

Table 11.3

Input voltage	Comparator output							Binary output		
	C_6	C_5	C_4	C_3	C_2	C_1	C_0	b_2	b_1	b_0
0 to V/8	0	0	0	0	0	0	0	0	0	0
V/8 to V/4	0	0	0	0	0	0	1	0	0	1
V/4 to 3V/8	0	0	0	0	0	1	1	0	1	0
3V/8 to V/2	0	0	0	0	1	1	1	0	1	1
V/2 to 5V/8	0	0	0	1	1	1	1	1	0	0
5V/8 to 3V/4	0	0	1	1	1	1	1	1	0	1
3V/4 to 7V/8	0	1	1	1	1	1	1	1	1	0
7V/8 to V	1	1	1	1	1	1	1	1	1	1

The expressions of the output bits can easily be obtained by examining the table 11.3.

The bit b_2 gives the high output whenever the output of the comparator C_3 is high. So $b_2 = C_3$.

The bit b_1 is high whenever the output of comparator C_1 is high and output of C_3 is low, or whenever the output of comparator C_5 is high. So $b_1 = C_1 \cdot \overline{C}_3 + C_5$.

Similarly, the expression for bit b_0 can be obtained as:

$$b_0 = C_0 \cdot \overline{C}_1 + C_2 \cdot \overline{C}_3 + C_4 \cdot \overline{C}_5 + C_6$$

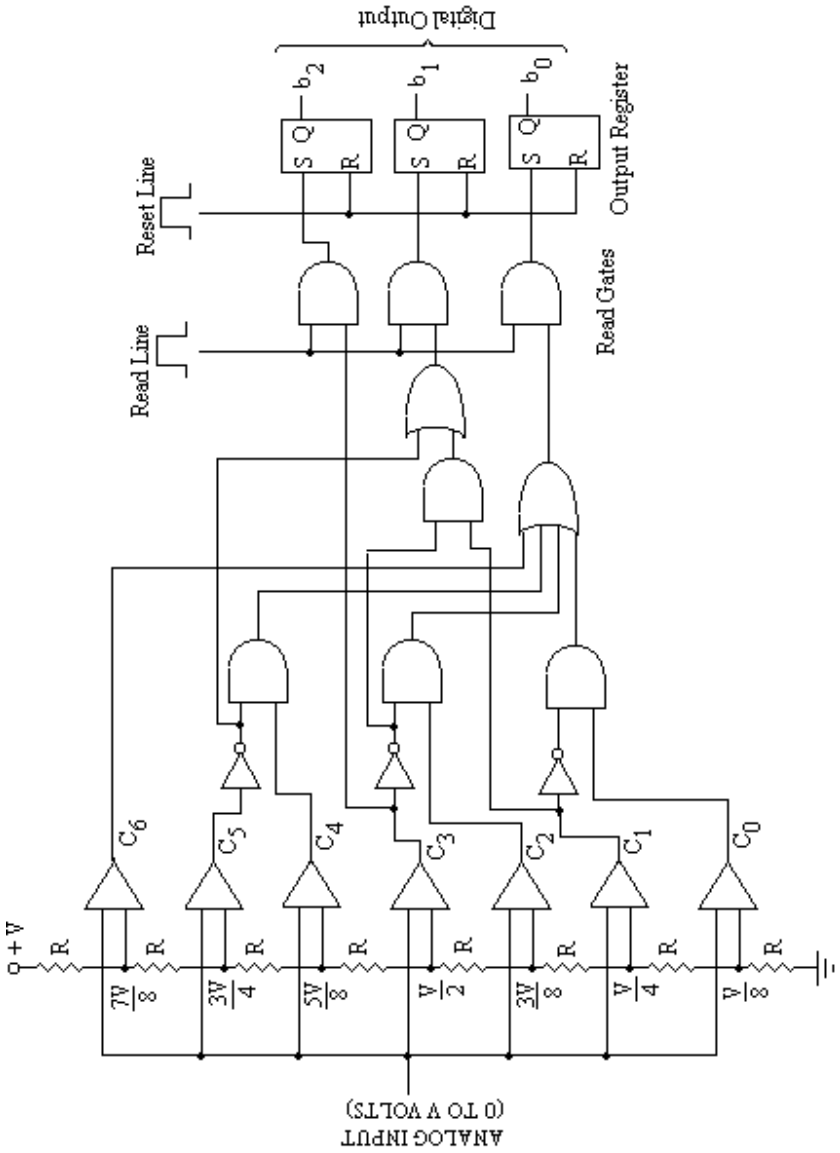


Fig. 11.17

These expressions may be realized using the gates as shown in figure 11.17. The output may be reset by applying high signal to the reset line and to read the data a high signal is applied to the read line.

The design of a simultaneous A/D converter is quite straight forward and relatively easy to understand. However, the design becomes complicated as the number of bits is increased, since the number of comparators to be used increases drastically. This method has highest speed of conversion.

11.6 SUCCESSIVE APPROXIMATION A/D CONVERTER

Simultaneous A/D converter has the very fast conversion time but becomes unwieldy when the required digital bits are more. The successive approximation method is most useful and commonly used method. The block diagram four bit successive approximation A/D converter is shown in figure 11.18. It consists of a D/A converter,

successive approximation register (SAR) and a comparator. The basic principle of this A/D converter is as follows:

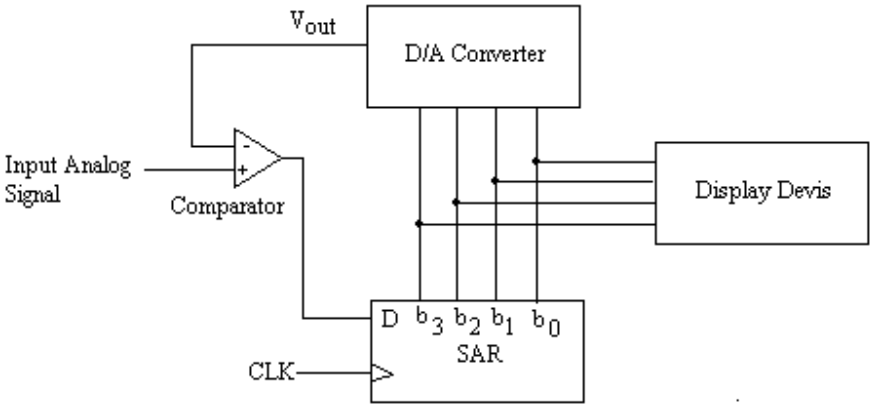


Fig. 11.18

In this type of converter, the bits of D/A converter are enabled one by one, starting with the most significant bit (MSB). The analog output of the D/A converter corresponding to the enabled bit is compared with the input analog voltage. The comparator gives the output low if the input analog voltage is less than the output of the D/A converter and it gives the high out if the input analog voltage is more than the output of the D/A converter. The low output of the comparator resets the corresponding bit of SAR, on the other hand if the comparator's output is high then that bit is retained in SAR. In this way the output of D/A converter are compared with the input voltage for all the bits starting with the most significant bit.

Thus the successive approximation method is the process of approximating the analog voltage bit by bit starting with MSB. This process is shown in figure 11.19.

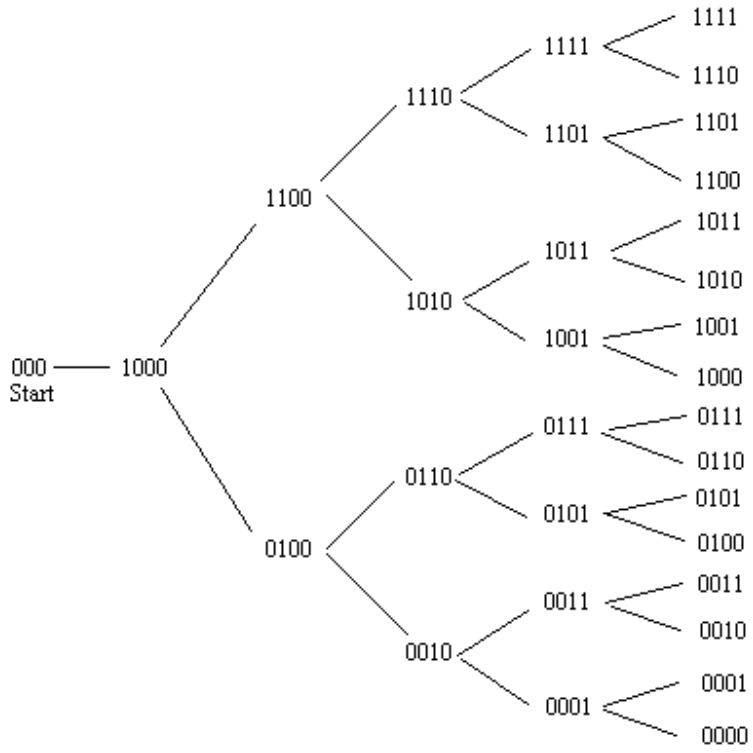


Fig. 11.19

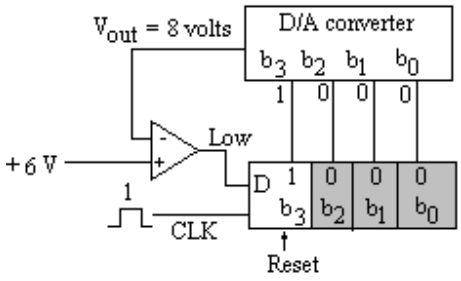
In order to understand the operation of this type of A/D converter, we will take a specific example of a four-bit conversion. Figure 11.20 (a through d) shows the step –by-step conversion of a given analog input voltage (say 6 volts). It is further assumed that D/A converter has the following output characteristics:

V_{out} = 8 volts for bit 3 (MSB or b_3)

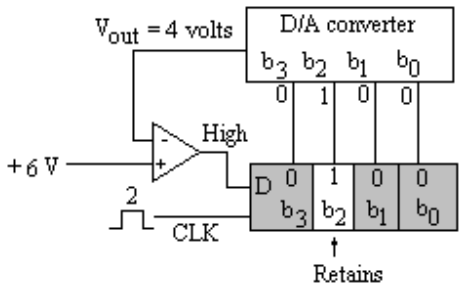
V_{out}= 4 volts for bit 2 (2^{nd} MSB or b_2)

V_{out}= 2 volts for bit 1 (3^{rd} MSB or b_1)

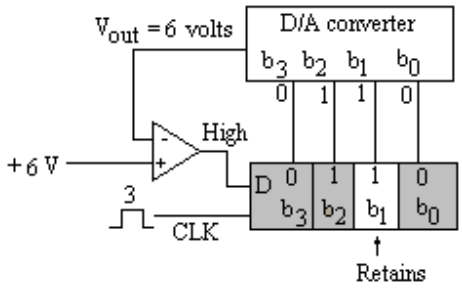
V_{out}=1 volt for bit 0 (LSB or b_0)



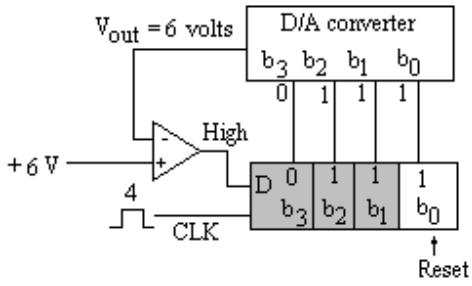
(a)



(b)



(c)



(d) **Fig. 11.20**

It is clear from these figures that after completing the conversion cycle. The binary code 0110 is retained in SAR, which is binary value of the input voltage (6Volts). It is finally displayed on the display devices.

11.7 COUNTER OR DIGITAL RAMP TYPE A/D CONVERTER

Another method of converting the analog signal to digital one is the counter or digital ramp type A/D converter which utilizes a binary counter to count a continuous pulse of standard width and height from a clock. The standard clock pulses are passed through a gate which is open for some time to allow these pulses to go to the input of counter. Normally the gate is closed and as soon as the start signal is applied a stair case voltage is initiated. This voltage is increased linearly with the increase of the binary counts in the counter. The gate remains open for the time the linear stair case voltage becomes equal to the input analog voltage. The counter records the number of clock pulses which is proportional to the input analog voltage.

Figure 11.21 shows the schematic diagram of this type of A/D converter. The analog signal, to be converted to its equivalent digital output, is applied to one input of an

operational amplifier being used as a comparator. When a start of conversion pulse is applied to the control unit it resets the binary counter and opens the gate. The counter

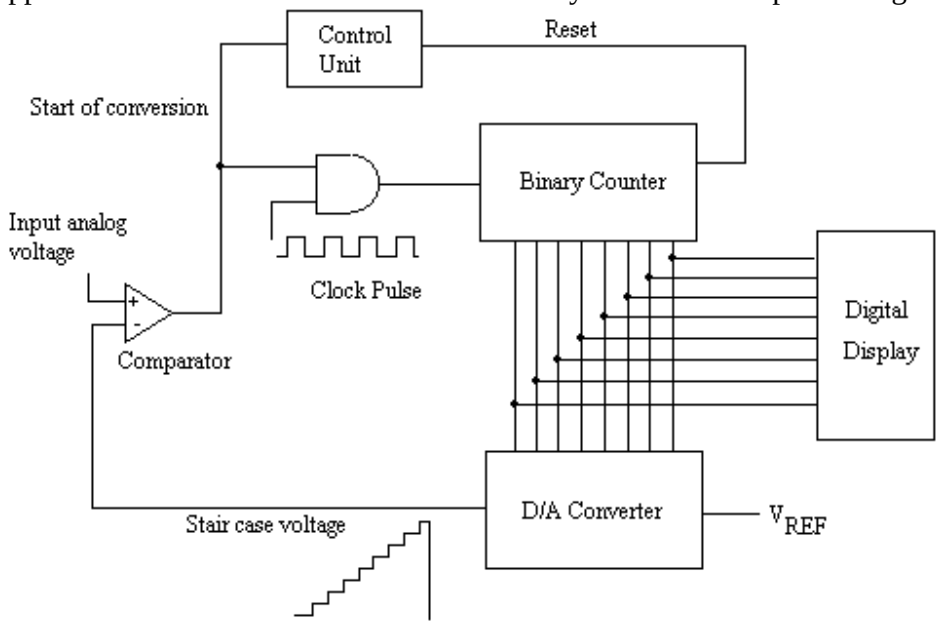


Fig. 11.21

starts counting the clock pulses which are of standard width and height. The output of the counter is fed to a D/A converter which produces an analog output (stair case voltage) in response to the digital signal (output of the counter) as its input. This analog output voltage is fed to the reference input of the comparator. So long as the input analog signal is greater than the stair case voltage the comparator provides the high output to the gate, the gate remains open and the clock pulses are allowed to reach to the input of the counter. These pulses are counted by the counter thus continuously increasing the digital output. The moment the analog output of D/A converter (stair case voltage) exceeds the input analog voltage, the comparator provides a low output disabling the gate and the counter stops counting. The binary number stored in the counter represents the digital output voltage corresponding to the input analog voltage. The digital output is displayed on the display devices.

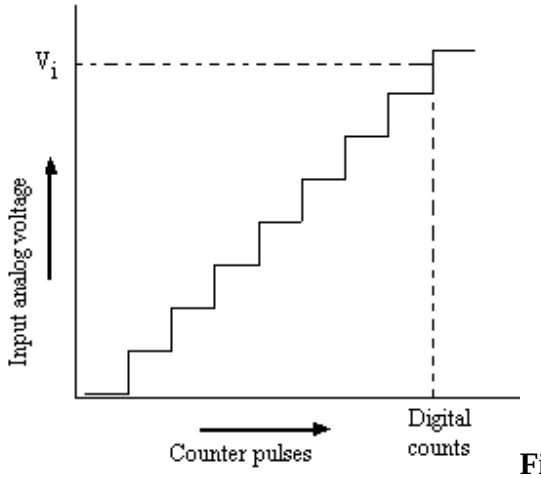


Fig. 11.22

For a steady input the digital output is as shown in figure 11.22. The output is represented by the number of clock pulses counted by the counter till the stair case voltage becomes equal to the input voltage. This method of conversion is slow; as for maximum input, the counter has to count from zero to maximum number of states for the comparison. For each conversion cycle the counter is to be reset and counting starts from beginning. The time of conversion is not important in d.c. or slow varying signals as the output waveform gives a good representation from which the input waveform can be constructed as shown in figure 11.23. But if the conversion time and the signal transient time are comparable the reconstructed digital output will not be correct. In this case it is necessary to reduce the conversion time by using faster D/A converter.

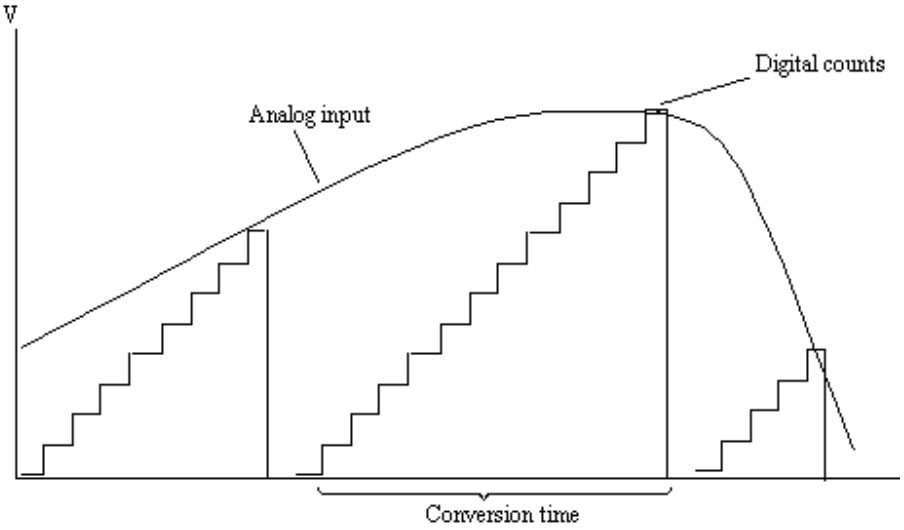


Fig. 11.23

A modification to this converter is possible if the resetting of the counter is avoided each time. For this purpose an up/down counter may be used in place of up counter. The circuit shown in figure 11.24 illustrates this modification in which an

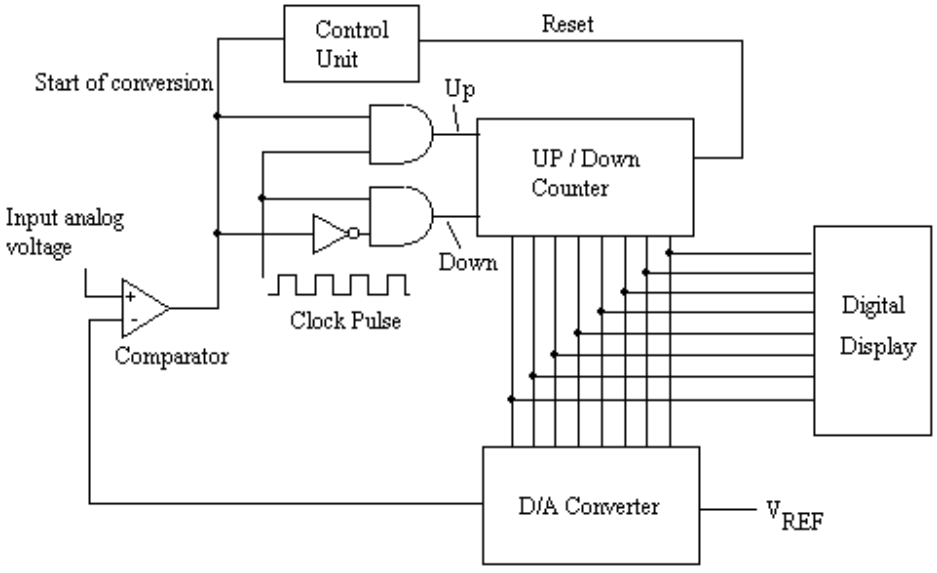


Fig. 11.24

up/down binary counter is used and the converter proceeds without resetting. The circuit is almost the same as the counter or digital ramp type A/D converter. The up/Down counter is operated by up or down signals from the control unit. The digital to analog converter output controls the output of the comparator. Till the D/A converter output is less than the analog input voltage, the up signal is enabled and the counter counts in forward direction. When the analog input falls, the down signal is enabled and the counter starts reverse counting giving an output corresponding to new analog input as shown in figure 11.25.

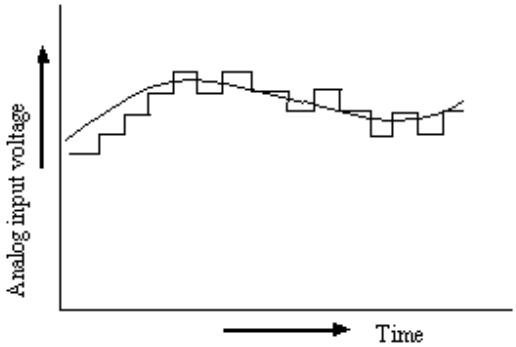


Fig. 11.25

11.8 SINGLE SLOPE A/D CONVERTER

This type of method is similar to counter or digital ramp type A/D converter. In this type of A/D converter also, a gate whose period is proportional to the amplitude of the analog sample is generated. For the generation of gate, the input analog voltage is compared with the output of an integrator. The output of integrator is a ramp voltage of constant slope. The standard clock pulses are passed through the gate and are counted by the counter. The gate remains open for the time proportional to the input analog signal. The recorded number of pulses is, therefore, the required digital output of the analog signal.

The schematic block diagram of such an analog to digital converter is shown in figure 11.26. Initially a reset pulse is applied which clears the counter and resets the integrator. The integrator produces a linearly rising ramp voltage, whose slope will depend on the values of the resistance R and capacitor C. The input analog voltage is compared by a comparator with the ramp voltage. As long as the integrator output is smaller than the input analog voltage, the comparator output is high. This high output enables the AND gate. The standard clock pulses are, therefore, allowed to pass through the gate which will be counted by the counter. When the ramp voltage becomes greater than the input analog voltage, the comparator changes the state thereby disabling the AND gate. The counter stops counting. It can easily be seen that the gate duration is linearly related to the magnitude of the input analog signal. Hence the count accumulated in the counter is a digital representation of the input analog voltage.

It may be mentioned here that the precision in the proportionality between the gate duration and the magnitude of the input analog signal depends upon the linearity of the ramp voltage obtained at the output of the operational amplifier. So the overall accuracy will depend upon the stability of reference source, the off-set of the operational amplifier, the frequency stability of the clock as well as the values of resistance R and capacitance C.

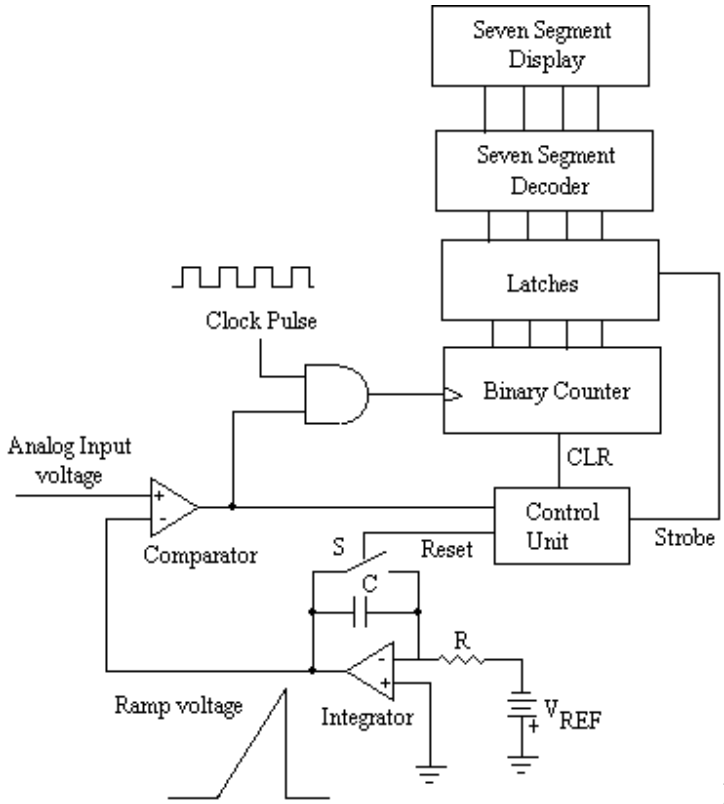


Fig. 11.26

11.9 DUAL SLOPE A/D CONVERTER

In single slope A/D converter, the accuracy of the converter depends on the linearity of the ramp voltage generated by the integrator. The linearity of ramp voltage, however, depends on the accuracy of the values of Resistance R and capacitance C of the integrator, whose values may vary with time and temperature. The dual slope analog to digital converter utilizes two different ramps, one for fixed time and other for fixed slope. It is very popular and widely used D/A converter because it has the slowest conversion time and relatively low cost. This method offers good accuracy, good linearity, and very good noise rejection characteristics.

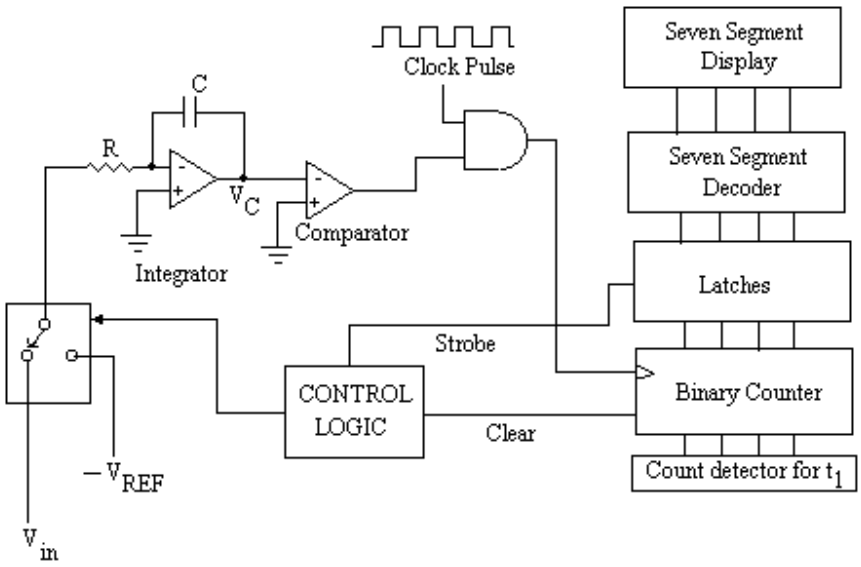


Fig. 11.27

The logic diagram of the dual slope A/D converter is given in figure 11.27. This converter is similar to that of the single slope A/D converter. In this converter, the integrator forms two different ramps, one for fixed time and other for fixed slope. The capacitor of the integrator is first charged with constant current obtained from input analog voltage for fixed time then the capacitor is discharged for fixed slope through other constant current obtained from a reference voltage source. The basic operation of this converter can be understood as follows:

This converter consists of standard clock pulses applied to the gate. The gate allows the pulses to the input of the counter which counts these pulses. Initially all the counters are reset to 0's and ramp too is reset to zero. Now the control logic allows switch S to connect the input analog voltage V_{in} to the integrator circuit. A constant current equal to $\frac{V_{in}}{R}$ flows through the capacitor C as the inverting input of the operational amplifier of the integrator is at virtual ground. The capacitor C will charge linearly with this constant current. This results a negative going ramp at the output of the integrator. The comparator's output will be positive which allows the clock pulse to pass through the AND gate to the input of the counter. This ramp is allowed for fixed time say t_1. The actual time t_1 is determined by the count detector. The voltage V_C at the output of the integrator is given by:

$$V_C = -\frac{1}{R.C}\int_0^{t_1} V_{in}.dt$$

$$= -\frac{1}{R.C}.V_{in}.t_1 \qquad \ldots (11.6)$$

The counter when reaches the fixed count at t_1, the control logic generate a pulse to clear the counter to zero and the switch S connects the integrator input to a negative reference voltage ($-V_{REF}$). The capacitor C of the integrator starts discharging linearly due to the constant current from $-V_{REF.}$ The integrator thus produces a positive going

ramp beginning at $-V_C$ and increases steadily till it reaches to 0 volt as shown in figure 11.28. At this time the counter is counting. The conversion cycle ends when $V_C = 0$ volt; the comparator produces the low state, which disables the gate and counter stops counting.

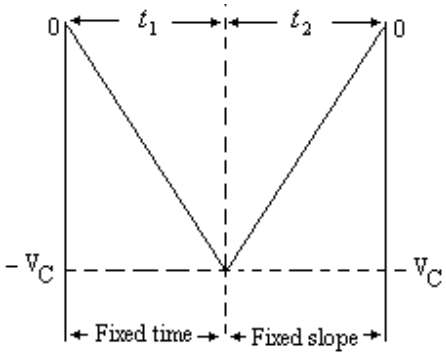


Fig. 11.28

Let t_2 is the time when the output of integrator becomes zero, so the output of the integrator is given by:

$$V_C = \frac{V_{REF}}{R.C}.t_2 \qquad \ldots (11.7)$$

Since the integrator's output beginning at 0 volt and integrates down to $-V_C$ and then integrate back to 0 volt, so the equations (11.6) and (11.7) may be equated as:

$$\frac{V_{in}}{R.C}.t_1 = \frac{V_{REF}}{R.C}.t_2$$

or
$$V_{in} = V_{REF}.\frac{t_2}{t_1} \qquad \ldots (11.8)$$

In this equation V_{REF} and time t_1 are constants, so

$$V_{in} \propto t_2 \qquad \ldots (11.9)$$

This equation is independent of the values of resistance R and capacitance C. Further at the end of conversion cycle, the counts measured by the counter are proportional to the input analog signal are latched to display on the display devices.

11.10 A / D CONVERTER IC 0801

There are many commercially available A/D converter ICs. The IC 0801 is the most popular, inexpensive and widely used 8 bit A/D converter. This IC uses successive approximation method to convert an analog input varying between 0 to 5 volts to an 8-bit digital equivalent. It has an on-chip clock generator for which external pins are provided to connect a resistance and capacitance. It is a 20 pin IC and operates on +5 volts supply. It has an optimum conversion time of approximately 100 μs. Figure 11.29 shows the pin configuration of this A/D converter IC 0801.

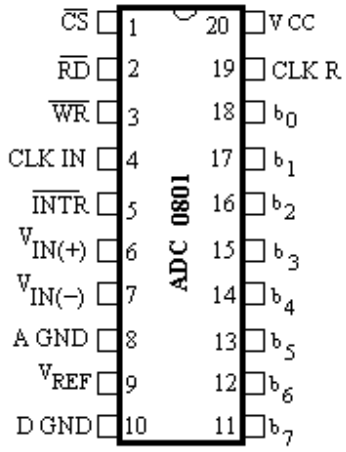


Fig. 11.29

The 20 pins of this IC are defined as:

Pin 1	$\overline{CS}$	–	Chip select terminal which is active low.
2	$\overline{RD}$	–	Output enable terminal which is active low.
3	$\overline{WR}$	–	Start of conversion which is also active low.
4	CLK IN	–	Capacitor is to be connected between this point and ground for internal clock.
5	$\overline{INTR}$	–	End of conversion which is active low.
6	$V_{IN(+)}$	–	Analog input pin (positive terminal)
7	$V_{IN(-)}$	–	Analog input pin (negative terminal)
8	A GND	–	Analog ground
9	V_{REF}	–	Reference voltage
10	D GND	–	Digital ground
11 – 18		–	Output bits b_7 to b_0 respectively.
19	CLK R	–	A resistance is to be connected between this pin and CLK IN for internal clock.
20	V_{CC}	–	+ 5 volts supply.

The frequency of the internal clock is given by the expression:

$$f = \frac{1}{1.1R.C}$$

The clock frequency of this converter IC should be in the range of 100 to 800 KHz. The outputs (bits b0 to b7) are tri-state outputs. If $\overline{CS}$ or $\overline{RD}$ is high the output pins float. The digital output appears on the output lines when $\overline{CS}$ and $\overline{RD}$ are both low.

PROBLEMS

1. Discuss the resistive divider D/A converter. Find the general expression for the output voltage of a resistive divider network.

2. Using the resistive divider network draw the circuit of a 6 bit D/A converter and explain its operation. What are the drawbacks of this D/A converter?
3. Show that the outputs of a binary weighted resistor network are directly proportional to the binary inputs.
4. Draw the schematic diagram of a resistive divider D/A converter. Explain its operation. Mention the drawbacks of this converter.
5. A 5 bit resistive divider network has 0 volts full scale output, find the output voltage for a binary input 10101. (Ans. 6.774 V)
6. A 6 bit resistive divider D/A converter has resistance of 100 KΩ in MSB branch. The reference voltage is 15 V. The resistance in the feed back path of the operational amplifier is 39 KΩ. What is the output voltage for the binary input 101101? (Ans. – 8.22V)
7. For a 6 bit resistive divider network, the reference voltage is 10 V, find the following:
   (i) Full Scale output voltage.
   (ii) The analog output voltage for a digital input of 010011.
   (iii) The output voltage change due to least significant bit.
   (Ans.:10V, 3.02 V, 0.16 V)
8. Draw the schematic diagram of a binary ladder D/A converter. Explain its operation. Mention its merits and demerits.
9. Find the expressions for the output due to MSB and second MSB of a 4-bit binary ladder network.
10. Discuss the binary ladder D/A converter. Find the general expression for the output voltage of a binary ladder network.
11. What are the performance criteria for the D/A converter? Discuss their importance while selecting a D/A converter.
12. For a 6-bit binary ladder D/A converter the input levels are 0 = 0 V and 1 = 10 V, find
    (i) The output voltages caused by each bit.
    (ii) The output voltage corresponding to an input of 101101.
    (iii) The full scale output voltage of the ladder.
    (Ans. (i) -5V, -2.5 V, -1.25 V, -0.625 V, -0.3125 V, -0.15625 V (ii) – 7.03125 V (iii) – 9.84 V)
13. For a 5-bit binary ladder D/A converter the input levels are 0 = 0 V and 1 = 10 V, find the output voltage corresponding to binary input of (i) 10111 (ii) 01101. (Ans. – 7.1875 V, – 4.0625 V)
14. What is the step size (or resolution in volts) of a 10 bit D/A converter, if the full scale output is +10 volts? Find the percentage resolution also. (Ans. 9.78 mV, 0.0978%)
15. How many bits are required at the input of a D/A converter to achieve a resolution of 15mV, if the full scale output is 15 volts? (Ans. 10 bits)
16. Give the details of D/A converter IC 0808. Using this IC draw a circuit diagram to get the analog output voltage corresponding to 8 bit digital input.

17. Discuss the simultaneous A/D converter to convert 0 to V volts analog voltage to 3 bit digital output. Draw the logic diagram also. What are the disadvantages of this type of A/D converter?
18. Draw a logic diagram to convert 0 to V volt analog voltage to its equivalent 2 bit digital output using simultaneous A/D converter.
19. Describe the successive approximation method for A/D conversion.
20. Draw a schematic diagram of counter or digital ramp type A/D converter. Explain its operation.
21. Describe the modified counter or digital ramp type A/D converter with neat diagram.
22. Draw a schematic diagram of a D/A converter. Explain its operation.
23. Describe single slope A/D converter with it logic diagram. Mention its merits and demerits.
24. Describe Dual slope A/D converter with it logic diagram.
25. Give the details of A/D converter IC 0801.

12

Digital Memories

In digital systems memories are used for the storage of binary information or data. It is well known that a flip-flop can be used to store the binary bit (0 or 1). So the flip-flops can be organized to form storage registers. The storage registers also called as memory registers are normally used for temporary storage of a few bits of information. These registers are combined to form a memory unit which is capable of storing large data. So the information to be stored in the digital system is transferred to these registers, where this information is retained and can be retrieved whenever required for processing in the digital systems. In this chapter both semiconductor and magnetic memories and their applications will be discussed.

12.1 MEMORY PARAMETERS

The memory unit is the important part of the digital systems or digital computers as the binary information necessary for processing in the system can be stored in or retrieved form this unit. The devices used in the memory unit can either be semiconductor devices or magnetic devices. A device or electronic circuit used to store a single bit is known as binary memory cell which include a flip-flop, a charged capacitor, a single spot on magnetic tape or disc. The characteristics of the device used as a binary memory cell should be as:

1. The device must have two stable states to represent the binary information 0 or 1. When the binary memory cell is one of the two stable states, it should not consume any power, if it does consume some power it must be small enough so that the total power dissipation must not be very large.
2. The cost and size of each cell should be very small so that the physical size occupied by the memory unit and its total cost are not too large.
3. The time taken to read the information from a group of binary memory cells or for storing the information in them should be very small.

The memory unit can be used to store a large number of binary words. A binary word is a combination of binary bits. The word length is different for different digital system or computers; typically it ranges from 8 to 128 bits. A binary memory cell is used to store a binary bit. If the length of the word in a system is of 8 bits then eight binary memory cells are combined to store a word. Each word stored in the memory unit will have different memory locations. The word will always be treated as an entity and can be

stored in and retrieved form the memory as a unit controlled by the control signals. The location of the memory unit where a word is to be stored or written is called the address of the word. So the address of the location is to be specified where the word is to be stored or retrieved from the memory unit. The word to be stored in the memory unit is first entered in the memory buffer register (MBR) also called as memory data register (MDR). The address where the word is to be stored is given in the Memory Address Register (MAR) and a WRITE signal is initiated by the control unit and the particular word will be written or stored in the specified memory location or the address. The length of the MBR is equal to the word length of the system. The length of MAR will, however, depend on the capacity of the memory locations. If a memory unit has the capacity to store *m* words (each of *k* bits), the length of MAR will be of *n* bits such that $2^n = m$ i.e. the length of MAR will be of 12 bits if the memory unit has the capacity to store 4096 words as $2^{12} = 4096$. In order to read or retrieved the stored word, the address of the location from where the word is to be read is given in MAR and then read signal is initiated by the control unit. Thus the stored content will be available in MBR. Figure 12.1 shows the block diagram of a memory system.

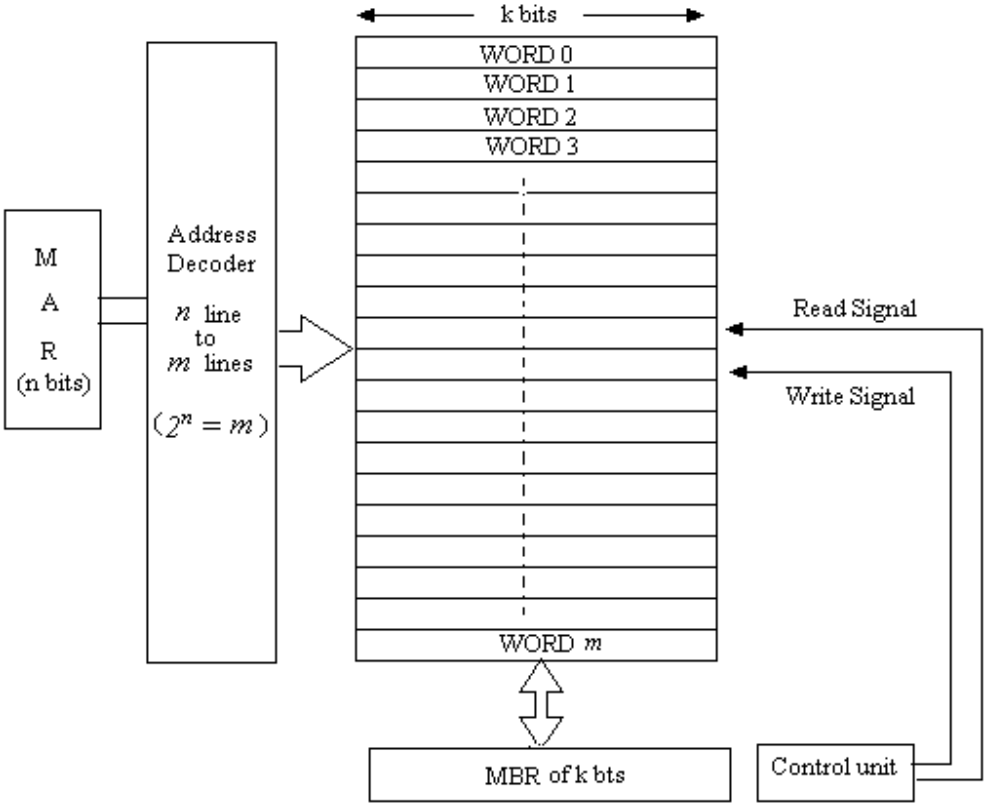


Fig. 12.1

There are some important terms related to memory unit which will now be discussed.

(i) **Destructive and Non-destructive Read Out**: As discussed above to read the stored content in some memory location the addresses of the location is given in MAR and when the read signal is initiated the stored content is copied into MBR. In this process if the copying process leaves the content in the corresponding

location undisturbed, then the read out process is known as non-destructive read out. If on the other hand, the stored content is lost during the reading process then the read out process is known as destructive read out. The read out process in the flip-flop binary cells are non-destructive while read out process in the binary cells made with magnetic cores is destructive.

(ii) Access Time of Memory: The time interval between the initiation of the READ signal and the availability of the stored content from the required memory location is known as the access time of memory.

(ii) Write Time of Memory and Memory Cycle Time: The time interval between the initiation of the WRITE signal and the storing of the content in the specified memory location is known as the write time of memory. In the destructive memory, during the read out process once the stored content is available in the MBR, the stored content is lost from the memory location. So in the destructive memory once the content is read from the memory location it is rewritten back in the same memory location. The time taken for reading the content and rewriting back in the same memory location is known as memory cycle time.

(iv) Volatile and Non-volatile Memories: The memory unit in which the stored content is lost when the power is turned off is known as volatile memory. The memory units consisting of flip-flop binary memory cells are the volatile memories as data is lost when the power is turned off. The memory unit consisting of binary cells made with magnetic cores is known as non-volatile as the stored data is not lost when the power is turned off.

(v) Memory Capacity: The number of bits that can be stored in a particular memory device or unit is known as the memory capacity. Suppose a memory unit can store 2048 twenty-bit words so it has a capacity of 40960 bits as 2048 x 20 = 40960. Further, 8 bit is known as byte so the capacity of 40960 bits memory is 5120 bytes or 5 K bytes as $1024 = 2^{10} = 1$ Kilo.

The larger memory may be represented by mega and 1M (1mega) = 2^{20} = 1024 x 1024 = 1048576.

Example 12.1 What are the sizes of MAR and MBR for a 16K x 32 bit memory?

Solution: The memory has the capacity to store 16 K words and each word is of 32 bits. So the size of MBR is 32 bits as it equal to size of the word.
The size of MBR is 14 bits as 2^{14} = 16 x 1024 = 16 K.

Example 12.2 How many words can be stored in 8K x 20 memory unit? How many bits can be stored with this memory unit? What are the sizes of MAR and MBR?

Solution: It can store 8K = 8 X 1024 = 8192 word and each word is of 20 bits.

It can store 8K x 20 = 8 x 1024 x 20 = 163840 bits.

Size of MBR = 20 bits

Size of MAR = 13 as 2^{13} = 8192.

12.2 SEMICONDUCTOR MEMORIES:

The Read Only Memory (ROM) and the Random Access Memory (RAM) are the two basic types of semiconductor memories. ROMs are those in which information or the data is permanently stored. The information can be read but fresh information cannot be written into it. These are nonvolatile memories. The other semiconductor memory RAM has both read write facilities. So the RAMs are also called as read write (R/W) memories. These are volatile memories.

12.3 READ ONLY MEMORIES:

As discussed above the read only memory is used to read the stored information or data but the fresh information can not be written into it. A block diagram of a read only memory is shown in figure 12.2, in which 8 words are stored and each

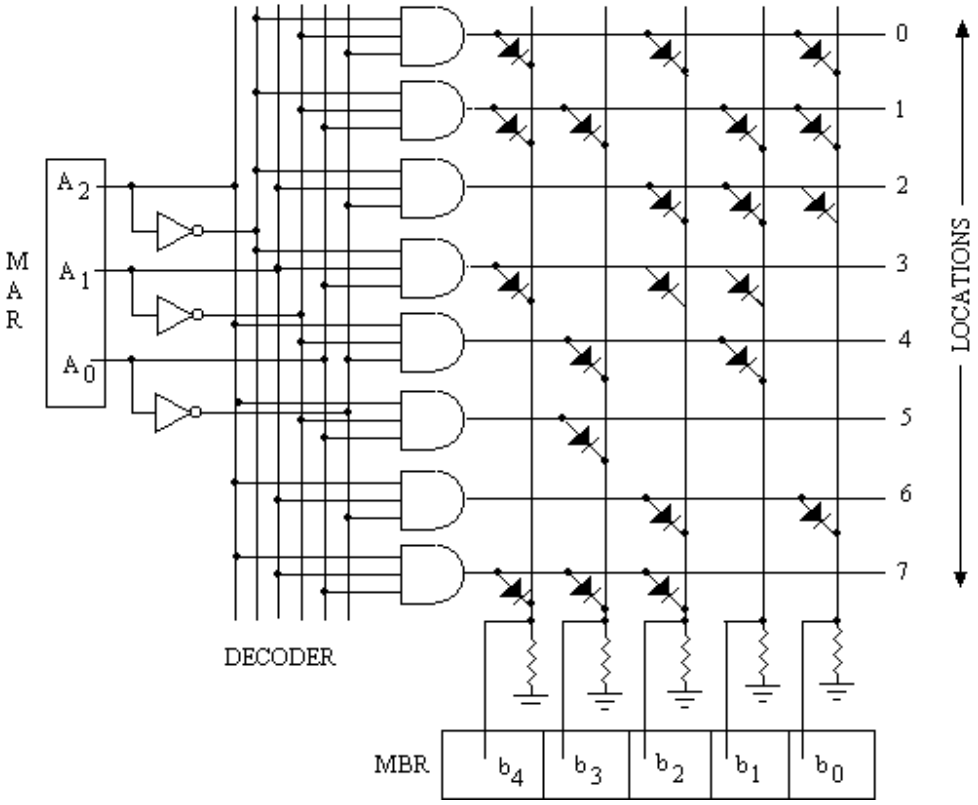


Fig. 12.2

word is of five bit long. This memory is organized as a two dimensional grid of 8x5 wires. It has eight horizontal wires and 5 vertical wires. At each intersection point in the grid, diode is either present or absent. If a diode is present at the intersecting point then that bit of the word is a 1 else it is a 0. This grid connected with diodes at some intersecting points is known as diode matrix ROM. The 8 words stored in 8 locations of this diode matrix ROM is given in table 12.1. This table indicates that at the zero location word 10101 is stored. To read the stored content the address of the location is given into MAR, the decoder circuit will activate the corresponding address line and diodes connected to that horizontal line conduct. The conducting diodes give rise the current

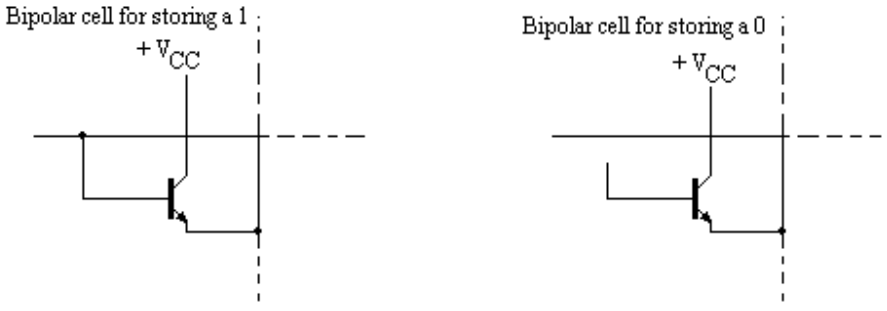
flow in the corresponding vertical wires and a high voltage is developed across the resistance connected to the vertical line (logic 1). If no diode connected to that horizontal line no current will flow to the corresponding vertical line and no voltage (zero voltage) is developed across the resistance connected to the vertical line (logic 0). Thus bits b_4 through b_0 will be available in the MBR as per the data available or stored in the addressed location.

Table 12.1

Words	Data bits				
	b_4	b_3	b_2	b_1	b_0
0	1	0	1	0	1
1	1	1	0	1	1
2	0	0	1	1	1
3	1	0	1	1	0
4	0	1	0	1	0
5	0	1	0	0	0
6	0	0	1	0	1
7	1	1	1	0	0

The diodes connected at the intersecting points are fixed by the manufacturer at the time of manufacturing according to the data supplied by the users. The data once fixed by the manufacturer in the ROM can not be altered.

Most ROMs available in the market are made with bipolar transistors or MOS transistors instead of diodes. At the intersecting points bipolar transistors or MOS transistors are connected. A bipolar cell for storing a 1 is shown in figure 12.3(a) in which the base of the transistor is connected to the row wire while the emitter is connected to the column line. When base is high the transistor conducts and a current flows through the column wire. A bipolar cell for storing a 0 is shown in figure 12.3(b) in which base connection is left open which result no current to flow through column wire.



(a) (b)

Fig. 12.3

A block diagram of 8x5 bipolar ROM matrix is shown in figure 12.4 and the data stored in different locations are shown in table 12.2.

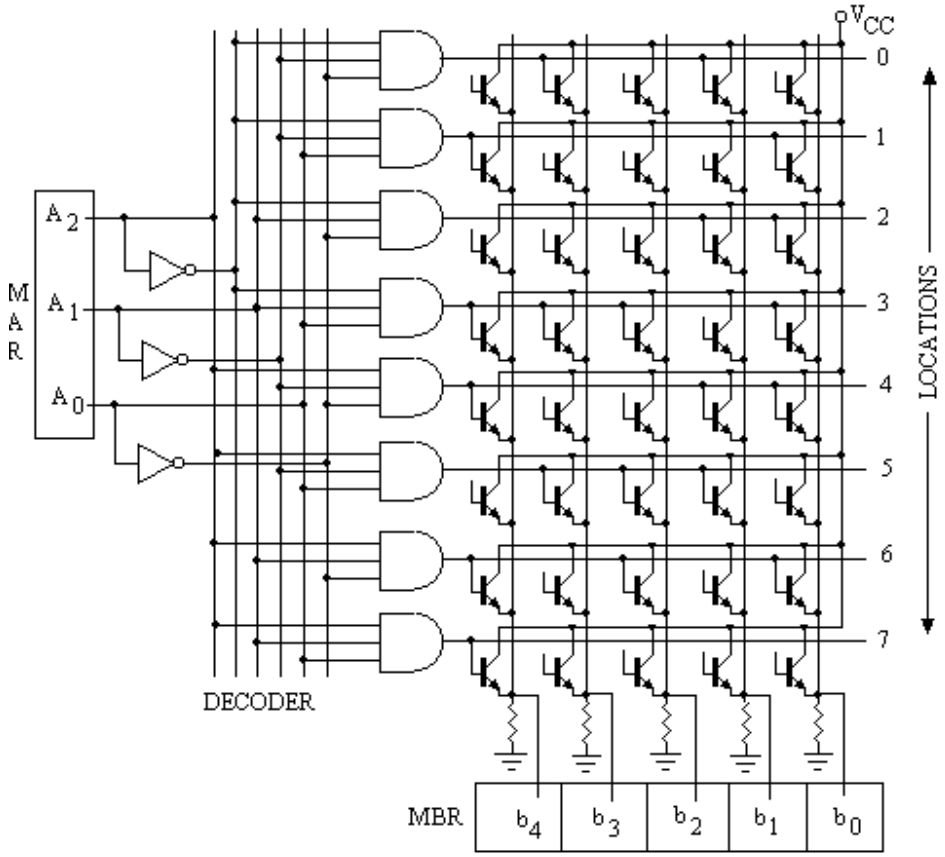


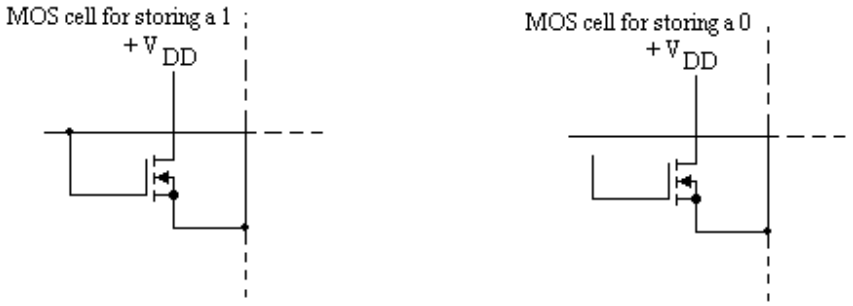
Fig. 12.4

Table 12.2

Words	Data bits				
	b_4	b_3	b_2	b_1	b_0
0	0	1	0	1	0
1	1	0	0	0	1
2	0	0	0	1	1
3	1	1	1	0	1
4	1	0	0	1	1
5	0	1	1	1	0
6	1	0	1	0	1
7	1	0	0	0	0

A MOS cell for storing a 1 is shown in figure 12.5(a) in which the gate of MOSFET is connected to the row wire while the source of the MOS is connected to the column line. A MOS cell for storing a 0 is shown in figure 12.5(b) in which gate

connection is left open. A block diagram of 8x5 MOS ROM matrix is shown in figure 12.6 and the data stored in different locations are shown in table 12.3.



(a) **(b)**

Fig. 12.5

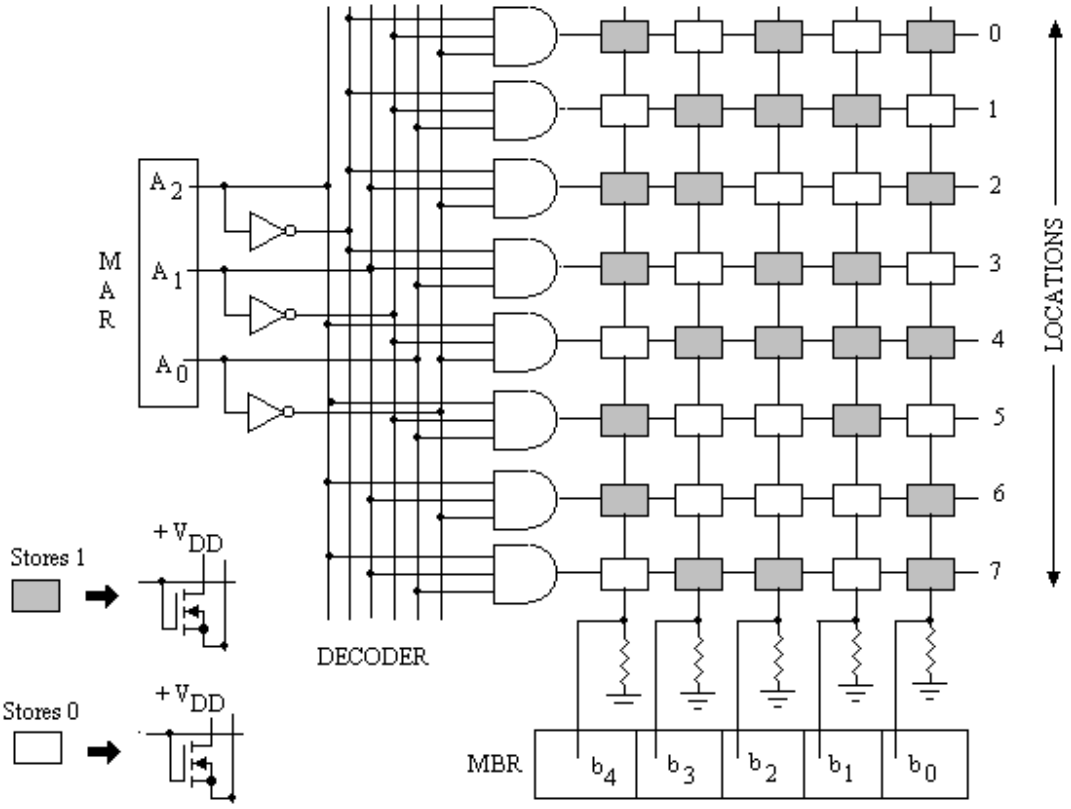


Fig. 12.6

Table 12.3

Words	Data bits				
	b_4	b_3	b_2	b_1	b_0
0	1	0	1	0	1
1	0	1	1	1	0
2	1	1	0	0	1
3	1	0	1	1	0
4	0	1	1	1	1
5	1	0	0	1	0
6	1	0	0	0	1
7	0	1	1	0	1

12.3.1 PROGRAMMABLE READ ONLY MEMORY (PROM):

In ROM's the data is fixed at the time of manufacture and the user can simply read the stored content. PROM's are also basically the same but the users can store the data as per their requirement. It is programmed by the user only once. It can not be reprogrammed. PROM's are available both in bipolar and MOS technologies. In PROM's bipolar or MOS transistors are connected to each joint and fusible links are provided to these transistors. Figure 12.7 illustrates a bipolar PROM array with fusible links provided at the emitter of each transistor connected at the joints. The fusible links may be burnt to store a bit 0; and a bit 1 is stored to keep the link intact. The user can burn the necessary links to store the desired data. For this a special device called PROM programmer is used for its programming.

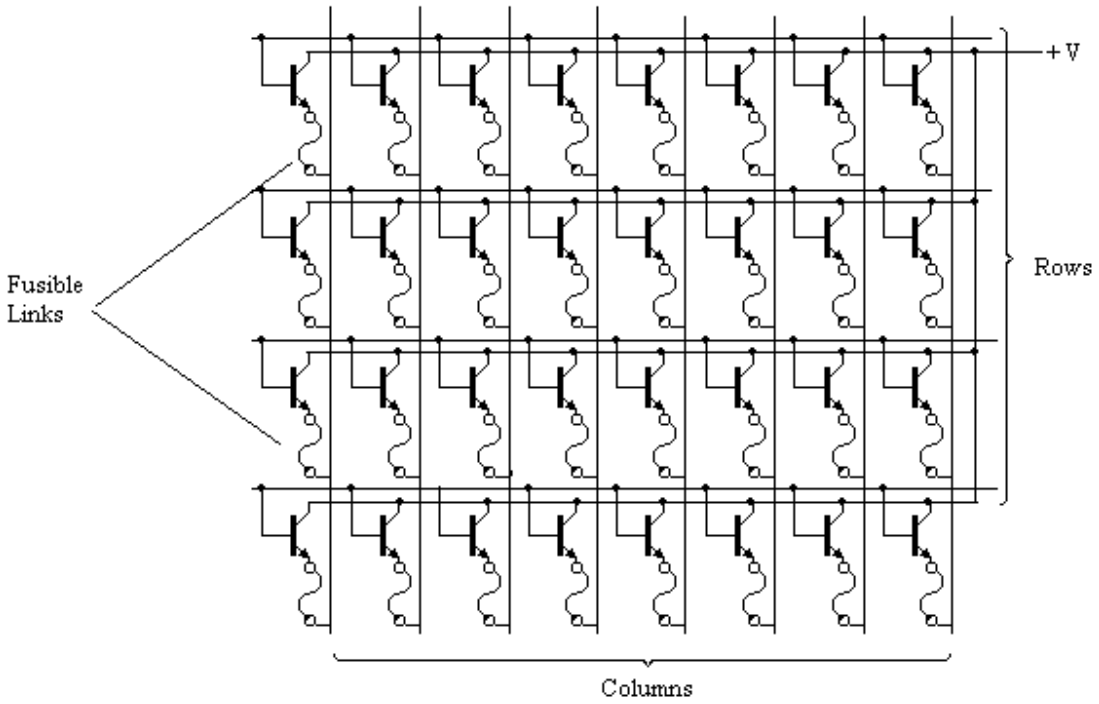


Fig. 12.7

12.3.2 Erasable Programmable Read Only Memory (EPROM):

The information or data once stored in ROM or in PROM can not be altered but in EPROM's the data can be erased and reprogrammed. Once programmed, the EPROM is non-volatile and the stored data will be retained indefinitely. Each binary cell in EPROM is formed with MOS transistor having a floating gate. The floating gate is surrounded by silicon dioxide which works as an insulator. If a sufficiently high voltage programming pulse is applied to the transistor, the high energy electrons are injected into the floating gate. Even after the termination of the programming pulse the electrons are trapped into the gate. Because the gate is completely isolated the charges can not leak very rapidly. It loses nearly 30% of its charge in a decade. Once the charges are stored on the gate the transistor becomes permanently on and the binary cell stores a 0. The cells which are not programmed store 1. So by proper programming of the memory the required data may be stored in desired memory locations.

The data can be erased if EPROM chip is exposed to the ultraviolet (UV) light. A quartz window on the chip is provided for the exposure of ultraviolet light. The ultraviolet light removes the stored charges on the floating gates of the MOS transistors. This in turn brings the EPROM chip back to the unprogrammed state. The erasing process usually takes 25 to 30 minutes. Erased chip may further be programmed with fresh data. The programmed chip may be protected from stray radiations by placing an opaque label on the quartz window of the chip.

The various EPROM chips are available with different storing capacities. The current popular series of these chips are 27XX, where XX indicates the capacity of the memory in kilo – bits. For example 2716 has the capacity (2 K x 8) to store 2 K words and each word is of 8 bit. It will have l1 address lines. Similarly, 2732 has (4 K x 8) capacity (4 K words, each of 8 bits).

12.3.3 Electrically Erasable Programmable Read Only Memory (EEPROM):

Electrically erasable Programmable Read Only Memories (EEPROMs) are also available as an improvement over EPROM. In EEPROMs also known as E^2PROMs, individual word in the memory can be electrically erased and reprogrammed. This facility is not available in EPROMs. In EPROM's complete memory contents are to be erased and reprogramming of the complete memory chip is to be required even if one or two words of the memory are to be altered. Another advantage of E^2PROMs is that the programming of this chip can be done when connected in the circuit without the use of ultra violet source and special PROM programmer unit. The memory cells of E^2PROMs utilize MOS transistors with floating gate structure similar to EPROMs, with the addition of very thin oxide region above the drain. This modification allows the cell to be electrically erased. By applying a high voltage (21 V) between the gate and drain of MOS transistors, where it will remain even when the power is removed. The application of reversed voltage removes the trapped charges from the floating gate and thus erases the cell. E^2PROMs can be erased in negligibly small time of 10 msec.

12.4 APPLICATIONS OF ROMs:

Read Only Memories are used in variety of tasks in the digital systems. Following are the common applications of ROMs:

Implementation of Logic Functions: ROMs can be used as the direct substitute of any logic function. For this consider the following example.

Example 12.3: Use a 32 x 8 bipolar PROM to form the following functions of five variables:

$$f_1 = \sum (1,2,6,8,9,13,16,21,29)$$
$$f_2 = \sum (0,3,8,10,14,15,16,22,25,30)$$
$$f_3 = \sum (5,8,9,11,19,20,21,25,29,31)$$
$$f_4 = \sum (1,5,6,9,13,16,28,29)$$

Solution: The PROM has the capacity to store 32 words of 8 bit long, so for getting four output functions f_1 through f_4 the output bits are assigned as:

$$f_1 = b_0 \qquad f_2 = b_1 \qquad f_3 = b_2 \qquad f_4 = b_3$$

The remaining output data bits are left open. List of all locations of PROM is prepared as shown in table 12.4. Each minterm of the given functions will represent its own address. The output bits will have logic 1 for the locations in the table for which the minterms is present in the function. For example, bit b_0 in the PROM will have logic 1

Table 12.4

Locations	*Contents*	*Locations*	*Contents*
00	0000 0010	16	0000 1011
01	0000 1001	17	0000 0000
02	0000 0001	18	0000 0000
03	0000 0010	19	0000 0100
04	0000 0000	20	0000 0100
05	0000 1100	21	0000 0101
06	0000 1001	22	0000 0010
07	0000 0000	23	0000 0000
08	0000 0111	24	0000 0000
09	0000 1101	25	0000 0110
10	0000 0010	26	0000 0000
11	0000 0100	27	0000 0000
12	0000 0000	28	0000 1000
13	0000 1001	29	0000 1101
14	0000 0010	30	0000 0010
15	0000 0010	31	0000 0100

for the locations corresponding to minterms given in f_1 i.e. bit b_0 will have logic 1 for the locations $1,2,6,8,9,13,16,21,29$ as illustrated in table 12.4. The logic diagram for the same is shown in figure 12.8.

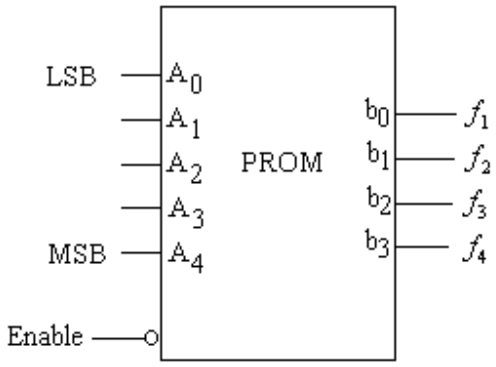


Fig. 12.8

Look-up tables: It is a usual practice to use ROMs as look-up tables for routine calculations in a computer. Trigonometric functions, logarithms, exponentials and square root etc are programmed as look-up tables in ROMs and used in lengthy calculations. It is economical to use look-up tables, rather than to use subroutine or a software program to perform the calculations for these functions. For example the look-up table for $y = \sin x$ can be formed with 128 x 8 ROM. This ROM will have 8 address lines and 8 output data lines. The address input should represent the angle in increment of desired accuracy and the output data lines will represent the approximate sine of the angle.

Code Converters: The ROMs can be used as code converter circuits. The data expressed in one type of code can be produced in other type of code. For this address lines of the appropriate ROM can be used as the representation of the given code and the output lines gives the equivalent data in the required code.

Example 12.4: Draw a diode matrix ROM that converts the four bit binary numbers to gray code.

Solution: Diode matrix ROM for the conversion of binary number to gray code is shown in figure 12.9, in which the address lines are used to represent the four bit binary numbers

Table 12.5

Binay Input				Gray Code			
A	B	C	D	X	Y	Z	W
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

and the output gives the gray codes. The data may be verified from the table 12.5.

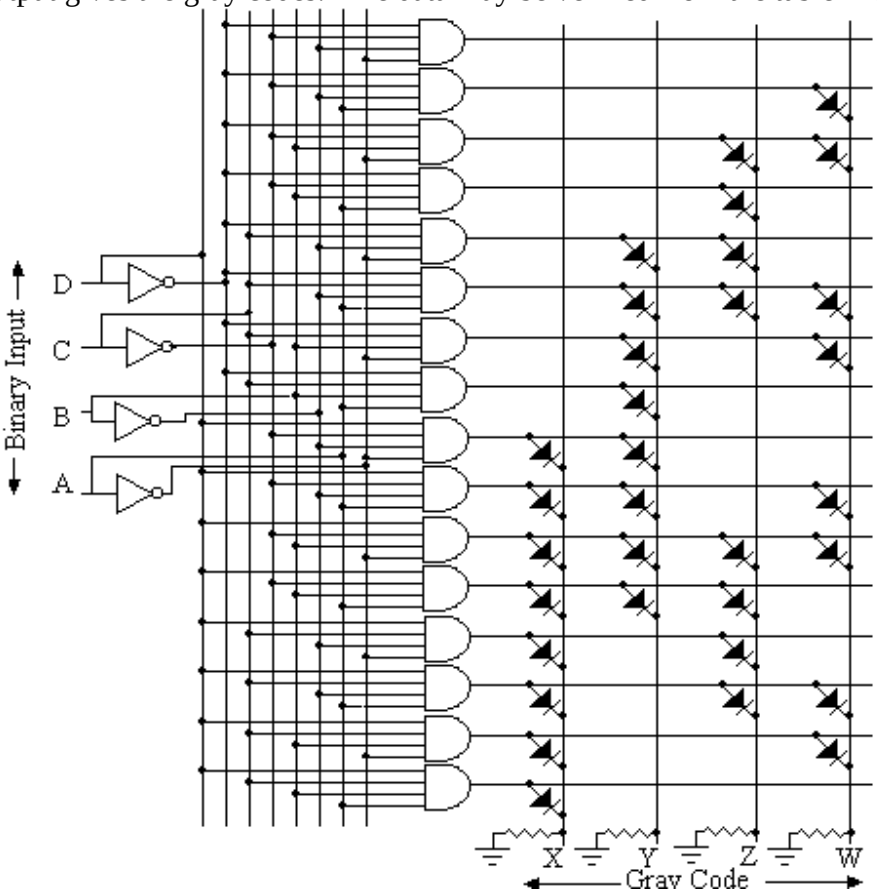


Fig. 12.9

Example 12.4: Draw a diode matrix ROM that implements the square of decimal numbers ranging from 0 to 15.

Solution: Table 12.6 shows the square data of the decimal number from 0 to 15. The binary equivalent of the decimal numbers represents the address of the location. This will need the 4 bit address line. It requires eight data lines as the square of 15 is 225 whose binary equivalent is 11100001. Figure 12.10 shows the diode matrix ROM that implements the square of decimal number ranging from 0 to 15. One can verify the data given in table 12.6 and ROM matrix.

Table 12.6

Decimal numbers	Bianry numbers as address of ROM				Square data output							
	A	B	C	D	a	b	c	d	e	f	g	h
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	1	0	0
3	0	0	1	1	0	0	0	0	1	0	0	1
4	0	1	0	0	0	0	0	1	0	0	0	0
5	0	1	0	1	0	0	0	1	1	0	0	1
6	0	1	1	0	0	0	1	0	0	1	0	0
7	0	1	1	1	0	0	1	1	0	0	0	1
8	1	0	0	0	0	1	0	0	0	0	0	0
9	1	0	0	1	0	1	0	1	0	0	0	1
10	1	0	1	0	0	1	1	0	0	1	0	0
11	1	0	1	1	0	1	1	1	1	0	0	1
12	1	1	0	0	1	0	0	1	0	0	0	0
13	1	1	0	1	1	0	1	0	1	0	0	1
14	1	1	1	0	1	1	0	0	0	1	0	0
15	1	1	1	1	1	1	1	1	0	0	0	1

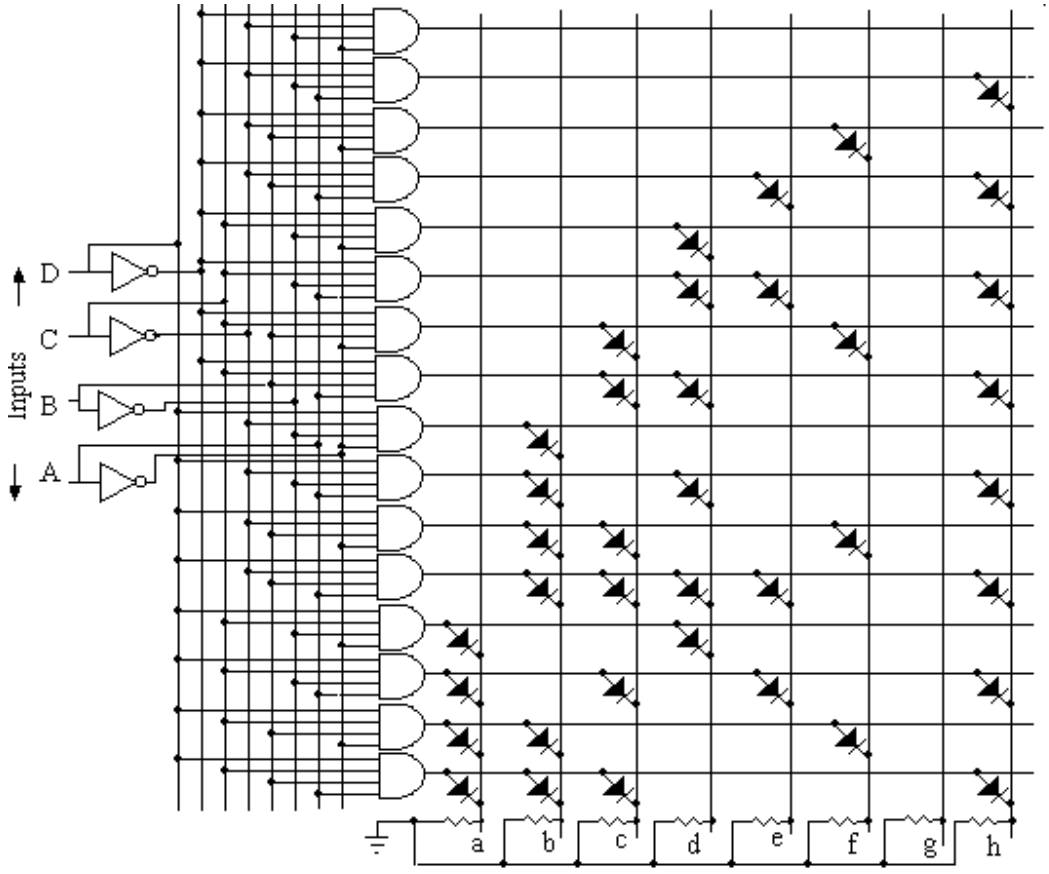
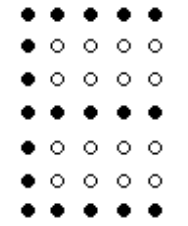


Fig. 12.10

Character Generators: ROMs can also be used to generate alphanumeric characters (dot patterns) on the video screen of computer monitor. There are many character formats that can be designed into ROM character generators. The 5 x 7 dot-matrix format (fig. 12.11) is generally used for display systems. The letter E is represented by this dot matrix. The solid dots in the letter E are lamps which are ON, and open dots are off light sources. A character generator ROM stores the dot pattern codes for each character at an address corresponding to the ASCII code for that character. For example, the dot pattern for the letter E would be stored at address 1000101, where 1000101 is the ASCII for E.



5 x 7 dot matrix

Fig. 12.11

Function Generator: The function generator produces sine, saw tooth, triangular and square waveforms. ROM can be used to produce such waves. Figure 12.12 illustrates how ROM look-up table is used to produce sine wave. The output lines of ROM are connected to a digital to analog converter. The ROM stores 256 different 8 bit values. The values stored at the different locations of ROM are the values of different voltage points of the sine wave. The eight address lines of ROM are connected to an 8-bit counter. The 8 bit counter sequentially excites the address lines of ROM with the application of clock pulse to the counter. The D/A converter gives analog output voltage corresponding to the data points of the required waveform. A low pass filter may be used at the output of the D/A converter to produce the smooth sine wave.

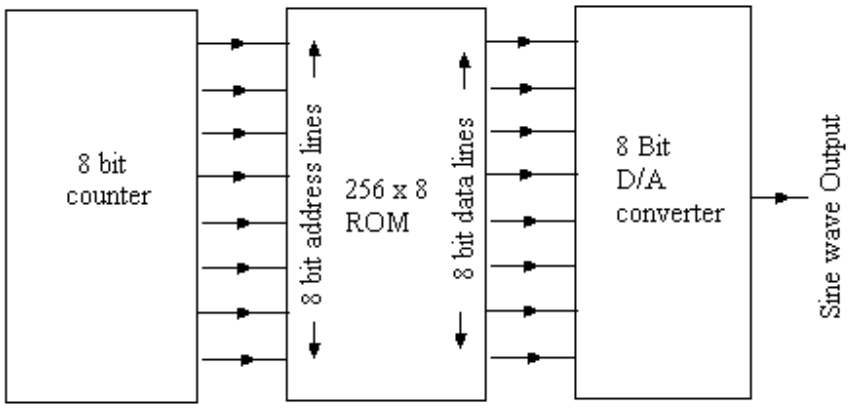


Fig. 12.12

12.5 RANDOM ACCESS MEMORIES:

Random Access Memory (RAM) is also known as Read/write memory. The Data can be written in to the memory location and can be read /retrieved from the memory locations. In this memory every memory cell can be addressed directly without the need

of any other previous cell being addressed first. In other words one may say that the contents of any memory location can be accessed randomly. RAM is volatile memory that is all the stored contents are lost if power is switched off. Basically RAMs are of three types Bipolar RAM and Static MOS RAM and Dynamic MOS RAM.

Before discussing the details of the different types of RAM, it becomes necessary to discuss first the different methods of memory cell addressing. There are two methods of memory cell addressing namely Linear Selection and Coincident Selection (or X-Y Selection).

Linear Selection: One method of addressing a RAM is known Linear Selection. In linear selection a memory cell can be approached by exciting the address lines appropriately. Suppose a random access memory can store 16 words of 4 bit each. In this method of selection it will have 64 cells which are arranged into 16 rows and four columns. One row will be for each word and one column for each bit in a word. Figure 12.13 illustrates linear addressing of 16 x 4 ROM cells. By four select inputs and 4 to 16 line decoder desired row from 16 rows can approached.

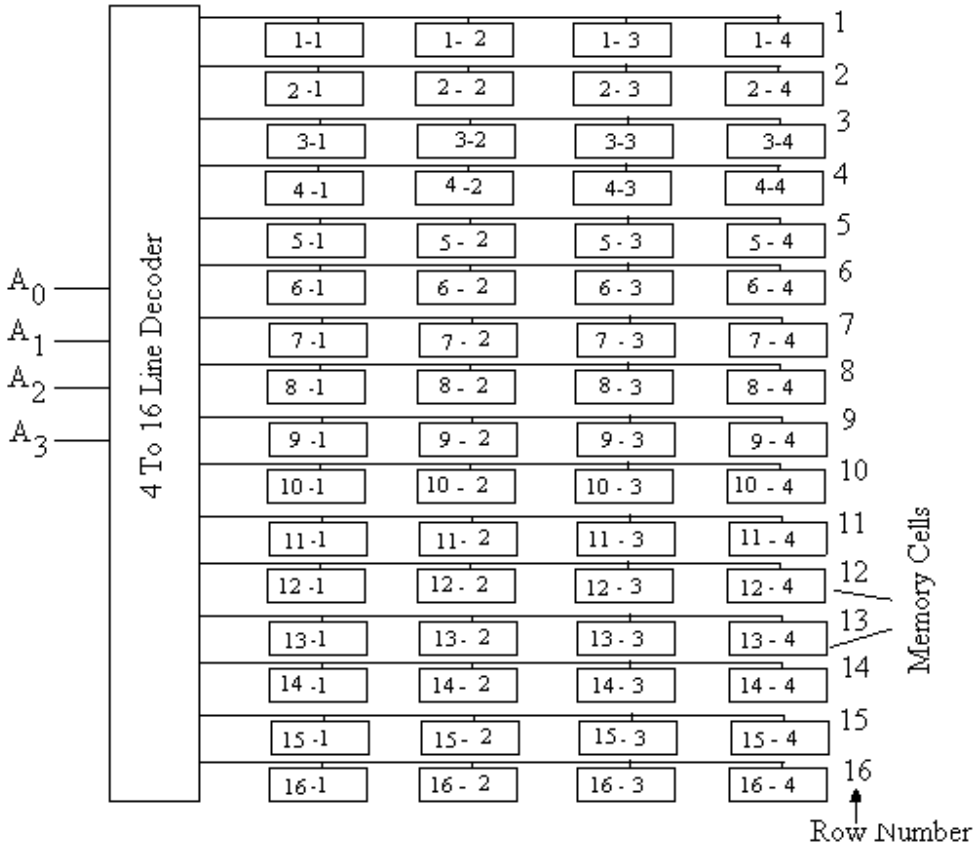


Fig. 12.13

Coincident Selection or X-Y Selection: The other addressing system known as coincident selection or X – Y selection shown in Fig.12.14. In this figure 64 memory elements are arranged in an 8 x 8 matrix for each word bit. A 64 word 256 bit RAM will need four 8 x 8 matrix arrays, one for each of the 4 bits in every word.

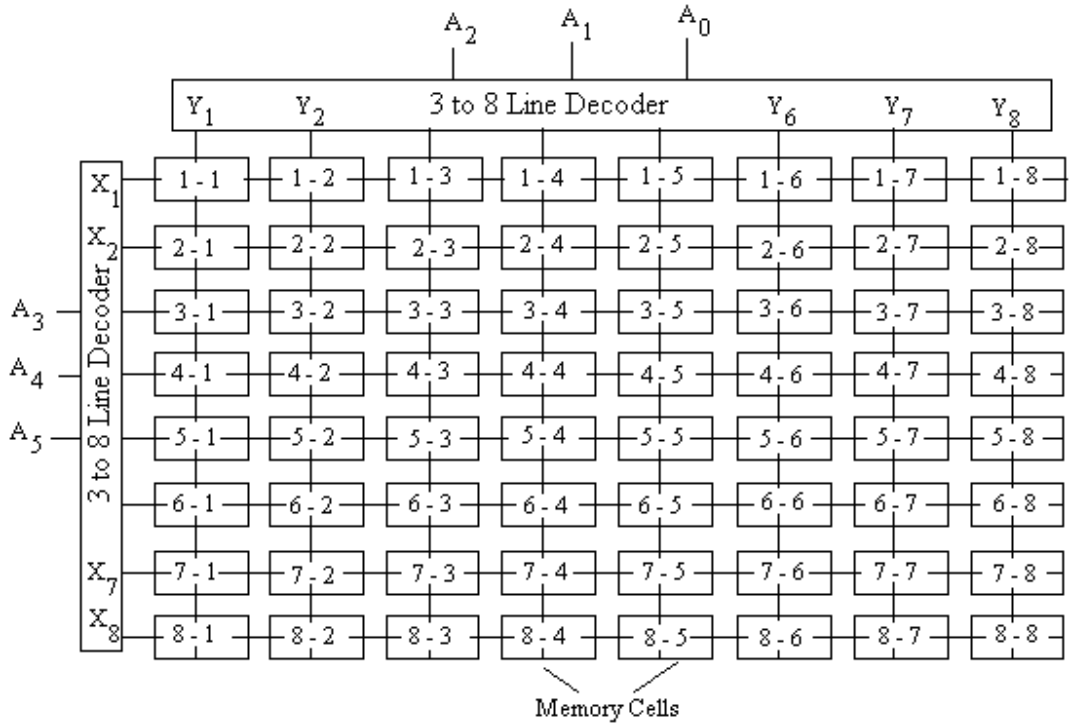


Fig. 12.14

12.5.1 Bipolar RAM: The cells of RAM make use of flip-flops which are designed using bipolar transistors. There are two types of RAM cells are designed using bipolar transistors shown in figure 12.15. The first type (fig. 12.15 a) is made using two dual emitter transistors. These types of RAM cells are used for linear selection. Triple emitter transistors are used in second type of RAM cells (fig. 12.15 b). These types of cells are used for coincident selection or X – Y selection

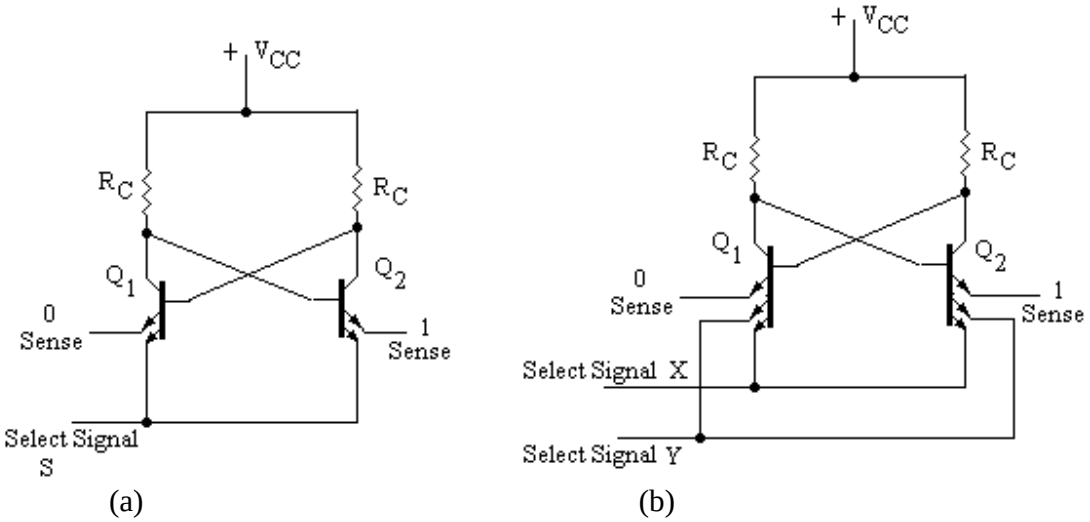


Fig. 12.15

In the first type (fig. 12.15 a) one emitter of each of transistors Q_1 and Q_2 are connected together to signal S. The second emitter of transistor Q_1 serves to sense or write a logic 0 (Q_1 ON). Similarly the second emitter of transistor Q_2 serves to sense or

write a logic 1 (Q_2 conducting). The sense and select terminals provides the low resistance path between the emitter and the round thus cell works as a flip-flop. Normally the select terminal S is kept low and the current form the conducting transistor flows out of this select terminal. For read operation, select line S is kept high. The conducting transistor will not conduct through select line but will conduct through sense '0' or sense '1' line depending upon whether logic '0' or logic '1'is stored in the cell. For writing or storing operation, the select line S is kept high. For storing logic '0' in the cell, the sense line '0' is kept low and sense line '1' is kept high. The transistor Q_1 now conducts to store logic 0. For storing logic '1' in the cell, sense line '0' is kept high and sense line '1' is kept low thus making the transistor Q_2 to conduct. Thus the cell stores logic 1.

In the second type bipolar RAM cell (fig. 12.15 b) two select terminals X and Y are obtained for connecting them to X and Y lines of coincident selection. The working of this triple emitter RAM cell is similar to dual emitter RAM cell. Normally both X and Y select terminals are kept low and the current from the conducting transistors flow out of these select lines. For read operation, select terminals X and Y are kept high; and thus no current flow through these select lines. The conducting transistor will conduct through sense '0' or sense '1' line depending upon whether logic '0' or logic '1'is stored in the cell. For writing or storing operation, the select lines X and Yare kept high. Similarly one can explain that for storing logic '0' in the cell the transistor Q_1 conducts and Q_2 becomes off; and for storing logic '1' in the cell, the transistor Q_2 conducts and Q_1 becomes off.

12.5.2 Static MOS RAM Cell: A static MOS RAM cell also known as SRAM cell is shown in figure 12.16. It consists of a flip-flop formed by n-channel MOS transistors. Here the MOS transistors Q_3 and Q_4 work as active load and MOS transistors Q_1 and Q_2 work as two NOT gates. The cross coupled NOT gates with active loads work as a flip-flop. It stays in the given state and retains the data indefinitely as long as power is applied to the flip-flop. The MOS transistors Q_5 and Q_6 provide the '1' sense line and '0' sense line respectively. The gates of these two transistors are connected together to form a select terminal for linear selection.

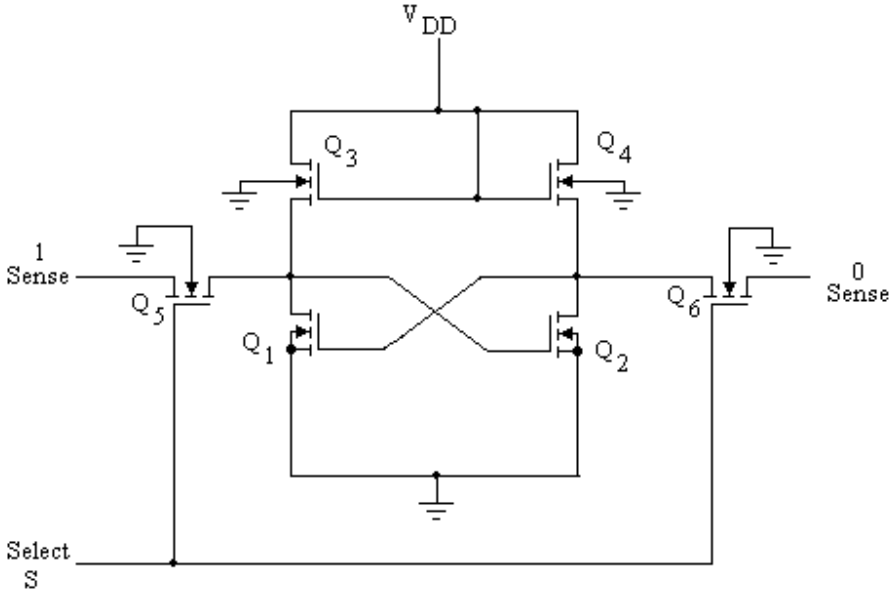


Fig. 12.16

Normally the select terminal S is kept low and for read operation, select line S is kept high. The transistors Q5 and Q6 will conduct through select line. In order to read or sense the state of the flip-flop suppose Q1 in ON and Q2 is OFF. Then the current flows through '1' sense line while no current flows through '0' sense line as Q2 is OFF. Similarly through the select line, the flip-flop can be set to logic '1' or logic '0' by using sense line as data input.

12.5.3 Dynamic MOS RAM Cell: Figure 12.17 illustrates a dynamic MOS RAM cell also called DRAM cell. It consists of a MOS transistor and a capacitor. The charging of the capacitor is controlled by the MOS transistor. The capacitor can hold a very small charge when it is charged. The MOS transistor is connected to an address line and a bit/sense line. This transistor works as pass transistor. To write a bit '1' on the cell the address line is kept high, a high voltage is applied to the bit/sense line. The transistor is switched ON and the capacitor is charged. The logic '1' is stored in the cell. However to write a bit '0', 0 volt is applied to the sense line and the capacitor is discharged and 0 is stored. Though the capacitor has a very large leakage resistance yet it is not an ideal capacitor. Thus the charge stored on the capacitor (when logic 1 is stored) discharges very slowly and the will be lost. It is therefore necessary to rewrite or refresh the data periodically.

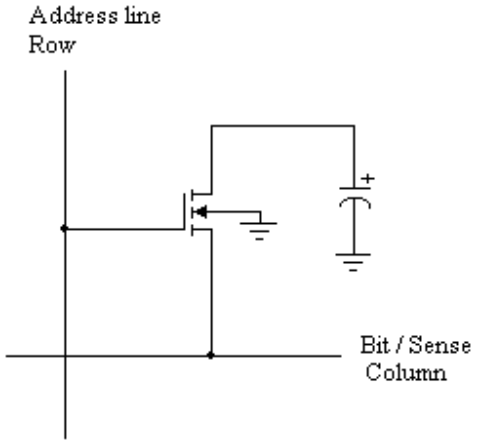


Fig. 12.17

To read the stored data in the cell high voltage is again applied to the address line. This switches ON the transistor and the capacitor voltage appears on the bit/sense line. If a '1' is stored in the cell, the voltage of the bit/sense line will tend to go up to the high voltage; and if a '0' is stored in the cell, the voltage of the bit bit/sense line will go down to 0 volt. The reading operation of this type of cell is destructive so a write operation should immediately be followed.

The dynamic RAMs are much cheaper than SRAMs as they allow high packing density (bits/chip) due to the simple structure of DRAM cells. The power consumption of DRAMs is very small as compared to the SRAMs. The dynamic RAMs are however slower in speed than Static RAM. Dynamic RAMs also require refreshing operation after regular intervals whereas SRAMs do not require this operation of refreshing.

12.6 RAM ICs: Figure 12.18 illustrates RAM IC 7489 of 16 x 4 memory. It is capable of storing 16 words of 4 bits. Data can be stored into memory by applying the address to

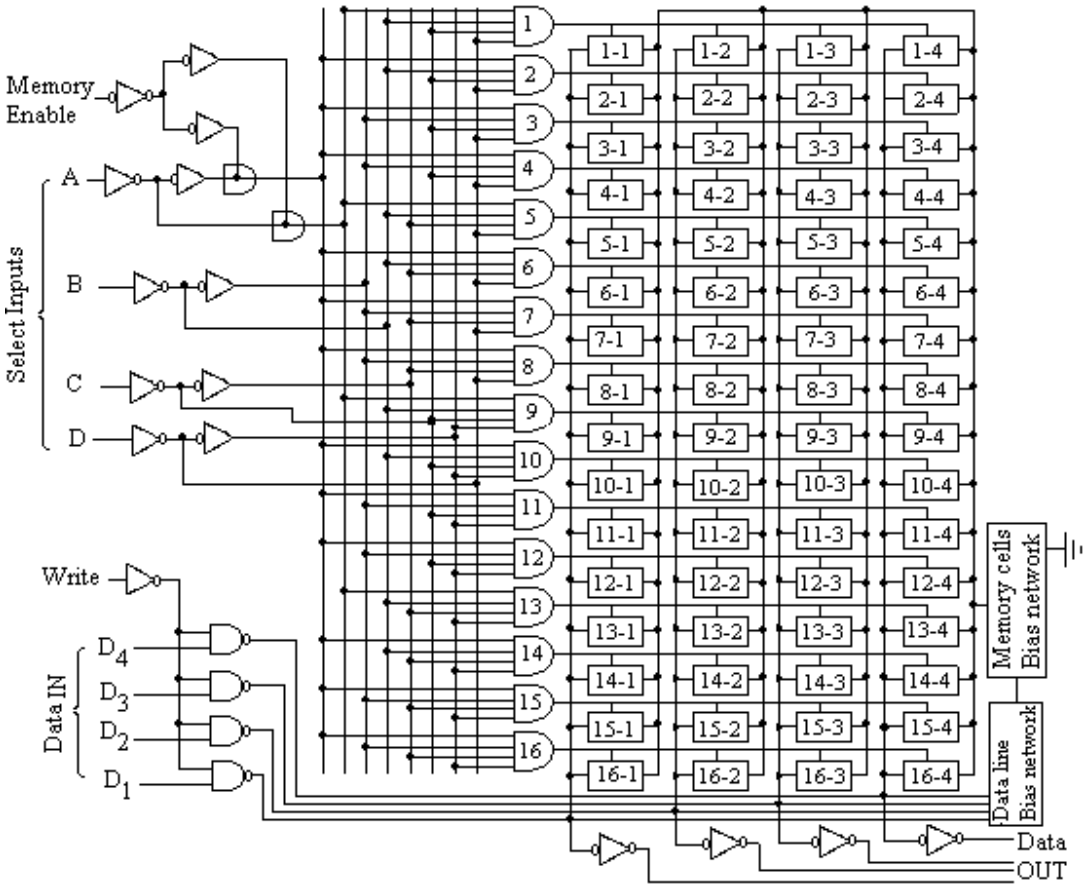


Fig. 12.18

the Select input and by providing low voltage to Memory enable ($\overline{ME}$) and write signal. However to read the stored content the address is given to the Select input and memory enable and read signal are applied low voltages. The data in complemented will appear at the Data out terminals. The functional block diagram of this IC is shown in figure 12.19.

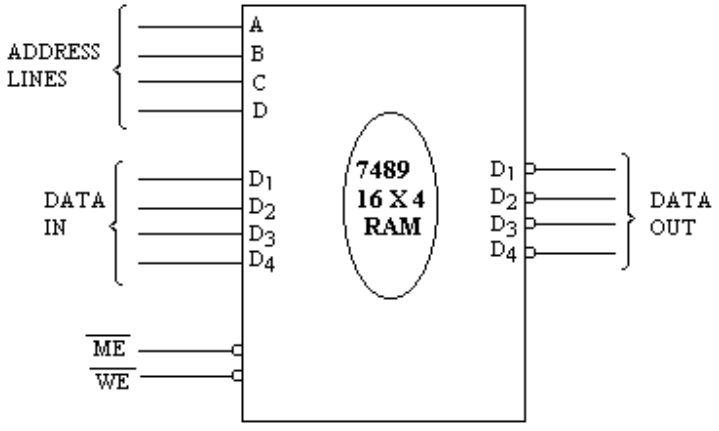


Fig. 12.19

RAM ICs can be connected in parallel to increase the word size. Two ICs 7489 (16 x4) are connected in parallel which is used as 16 x 8 memory. This is illustrated in figure 12.20.

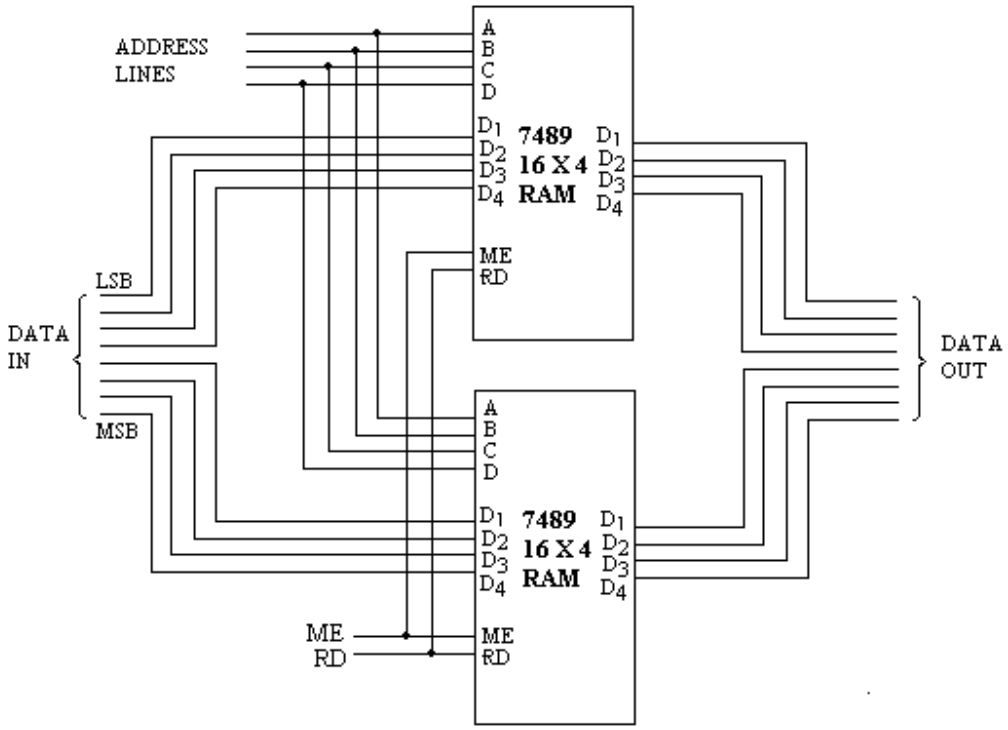


Fig. 12.20

Figure 12.21 shows an IC 2147 which is a Static MOS RAM of capacity 4K x 1. It contains separate terminals for DATA in (D_{in}) and DATA out (D_{out}). The chip select terminal ($\overline{CS}$) should be low to activate the chip. The bit may be written or stored in the RAM if write signal ($\overline{WE}$) is made low, of course the chip select terminal should also be activated. Data out (D_{out}) terminal remains isolated with the rest of the circuit during the write operation.

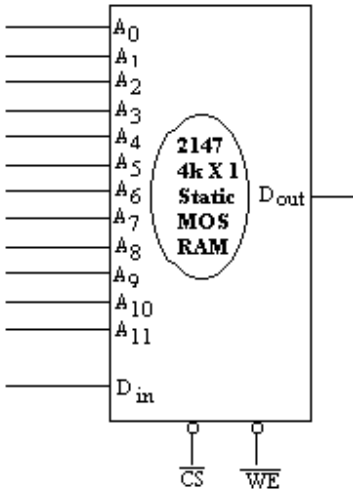


Fig. 12.21

Figure 12.22 shows Dynamic MOS RAM chip 4164 of capacity 64K x 1. It has 8 bit address line. However for 64K memory it should have 16 bit address line, as $2^{16} = 64K = 64x1024 = 65536$. For this the memory is arranged into 256 rows and 256 columns as 256 x 256 = 65536. It contains ROW ADDRESS STROBE ($\overline{RAS}$) and COLUMN ADDRESS STROBE ($\overline{CAS}$) pins for selecting row and column of address. The memory arrangement for 256 x 256 is shown in figure 12.23.

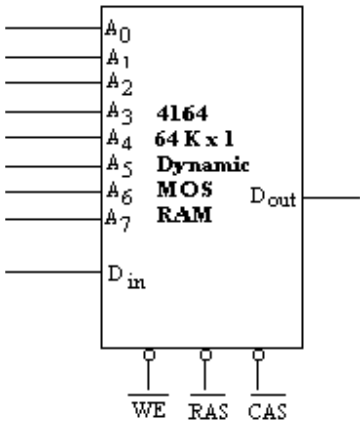


Fig. 12.22

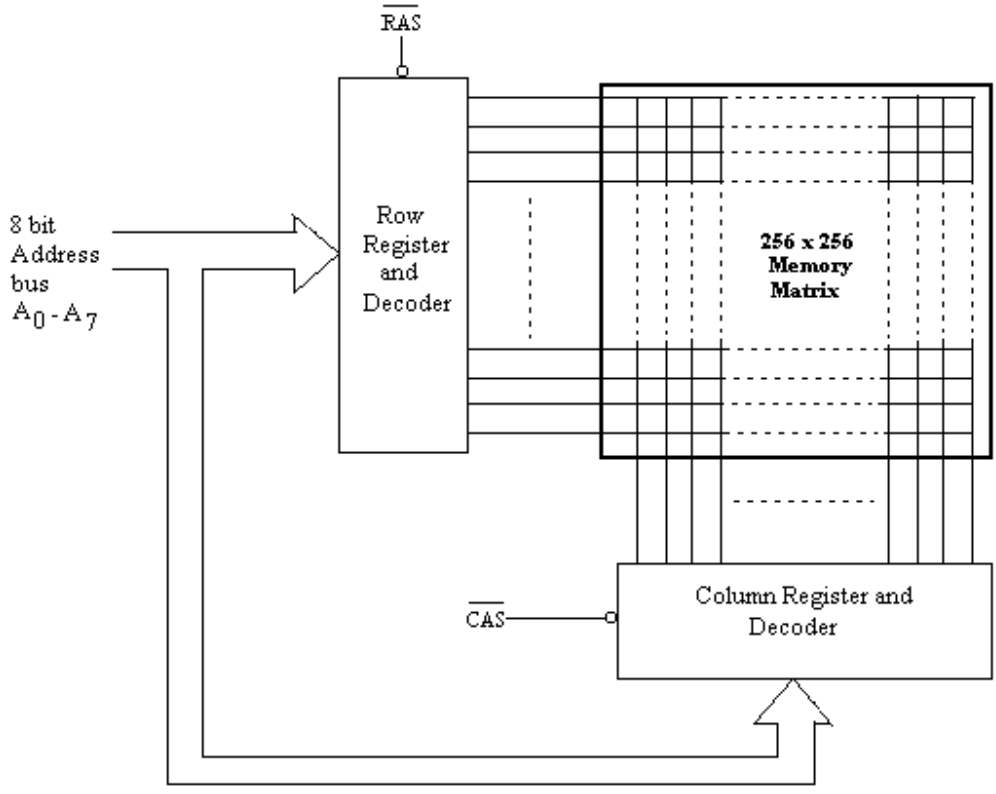


Fig. 12.23

The combination of 8 PROMs (1K x 8) to produce a total capacity of 4 K X 8 is illustrated in figure 12.24. This arrangement can very be understood.

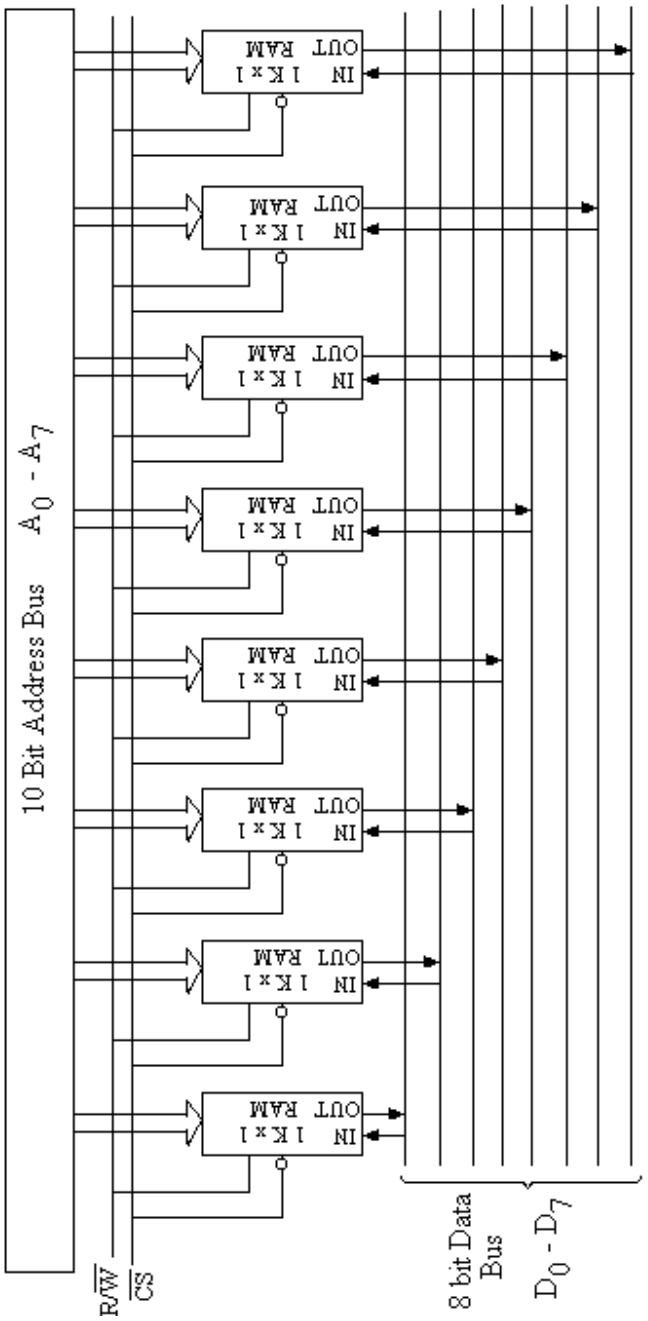


Fig. 12.24

Figure 12.25 illustrates the construction of 4K x 8 memory using 4 PROMs (1K x 8). The PROM 1K x 8 has 10 address lines. However, for 4K memory, it requires 12 address lines. Two extra address lines in combination with 2 to 4 line decoder are used to select the particular chip. This is clearly specified in the figure 12.25.

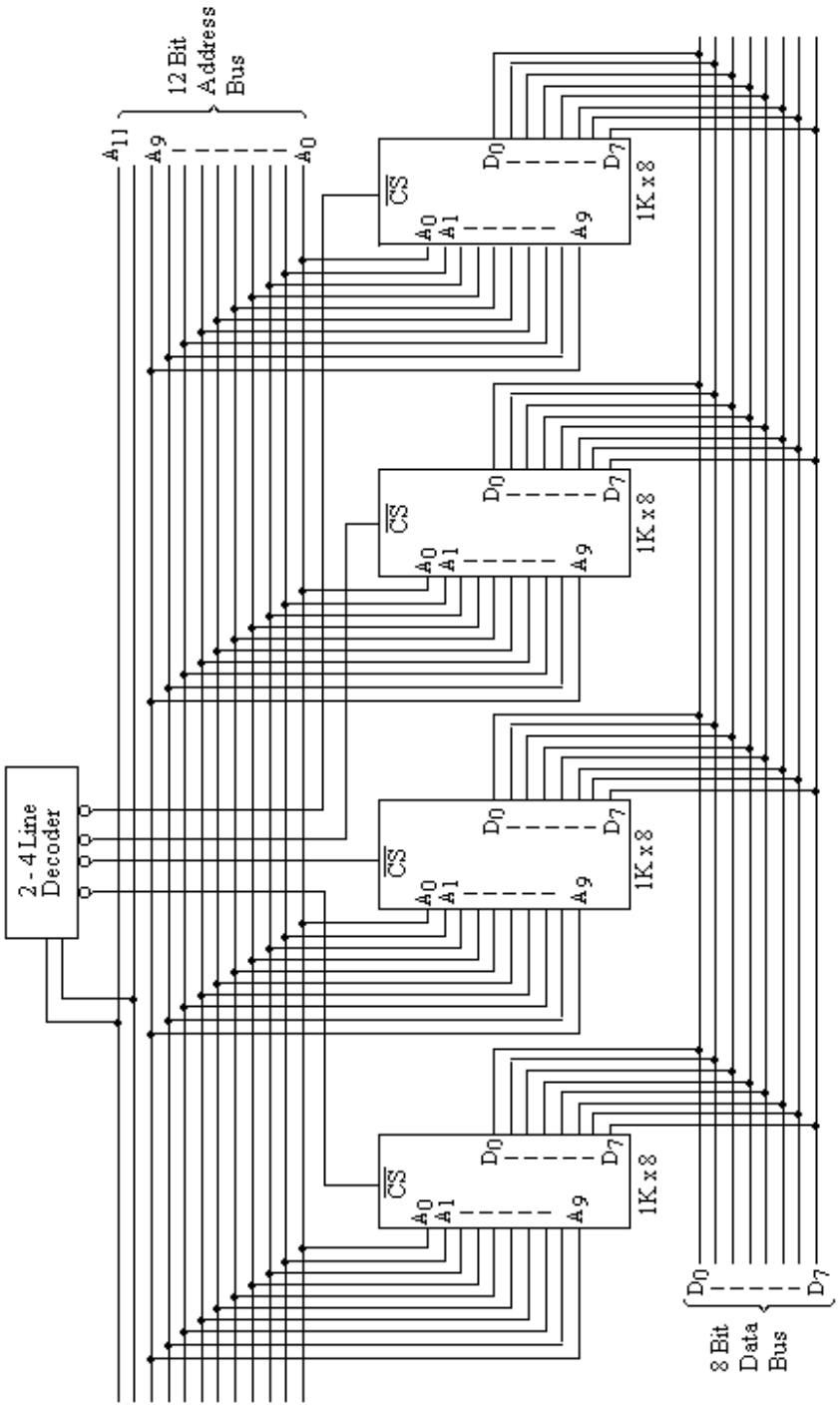


Fig. 12.25

12.7 MAGNETIC MEMORIES: In the forgoing sections of this chapter semiconductor memories have been discussed which utilizes the memory cells based on electrical charge or voltage. Magnetic memories are based on the principle that a ferromagnetic material can be magnetized by passing a current through it. The direction of magnetization depends on the direction of current. The magnetic materials were found to be inexpensive and everlasting materials; therefore, it became an ideal choice as the storage devices. Magnetic core, magnetic tape and disk, floppy disk etc. are some commonly used memory devices.

12.7.1 Magnetic Core Memory: In the magnetic core memory a core of a ferromagnetic material is used as a storage element. The core is usually toroidal is shape as shown in figure 12.26. When a current *i* is passed in the direction indicated in the figure 12.26(a) through the winding on the magnetic core, magnetic flux ϕ is set up in the clockwise direction. The variation of the magnetic flux ϕ with current *i* is shown in figure 12.26(b).

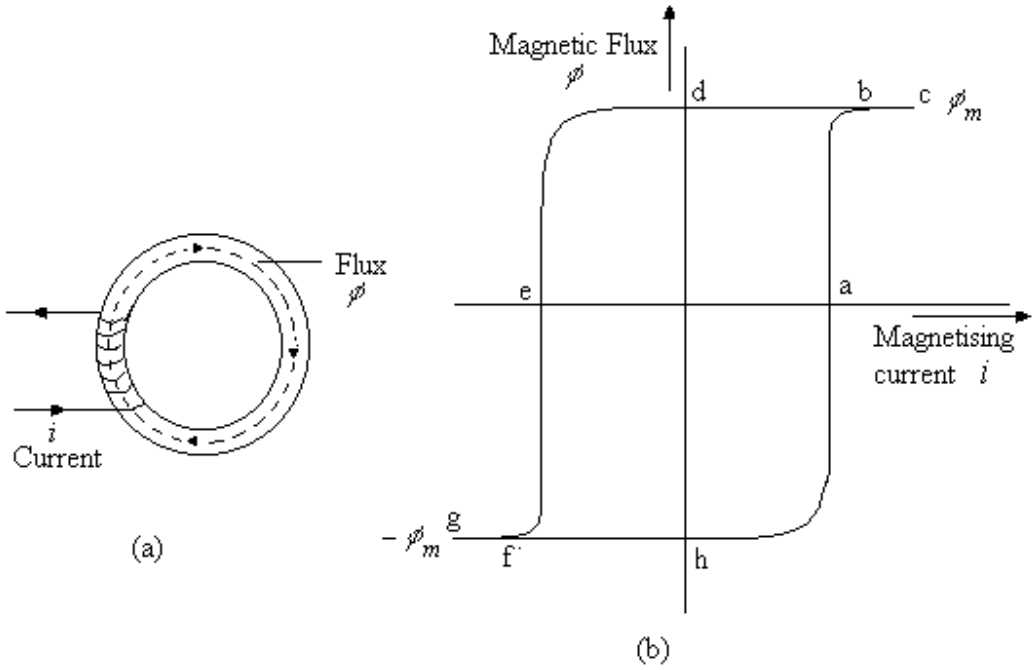


Fig. 12.26

This curve is known as hysterisis curve. This hysterisis curve is almost rectangular in shape; in fact for the magnetic core memory such a magnetic material is used whose hyterisis curve is rectangular in shape. From this curve it is clear that when the core is magnetized with the positive direction of current *i* (curve a b c), the magnetic flux gets the saturated value ϕ_m at point c and further increase in the magnetizing current will not increase the magnetic flux induced in the core. Now when the current is decreased the flux changes according to the curve c b d and stays at the point d where the magnetizing current becomes zero. This state is called positive remnant flux. The core remains in that state for indefinite period even without supplying any energy. Now if the direction of magnetizing current is reversed curve follows the path d e f g and gets the negative saturated value of flux $-\phi_m$. Now if the magnitude of the current is increased then the

core attains the negative remnant flux at the point h. It is therefore clear that the core can be magnetized and it attains either the positive remnant flux or negative remnant flux. In other words the core remains in either of the two states without any external energy. The energy is required only to change the state. One state may be represented by logic '1' and other state by logic '0'.

A similar situation arises when a current is passed through a wire which passes through the axis of the core. The current *i* following upwards (fig. 12.27) in the wire will lead a magnetic flux in the counter clockwise direction and the state attained by the core may be represented by a logic '1'. Similarly the current *–i* flowing in the wire (in the down ward direction) gives the state represented by logic '0' by setting the flux in the clockwise direction.

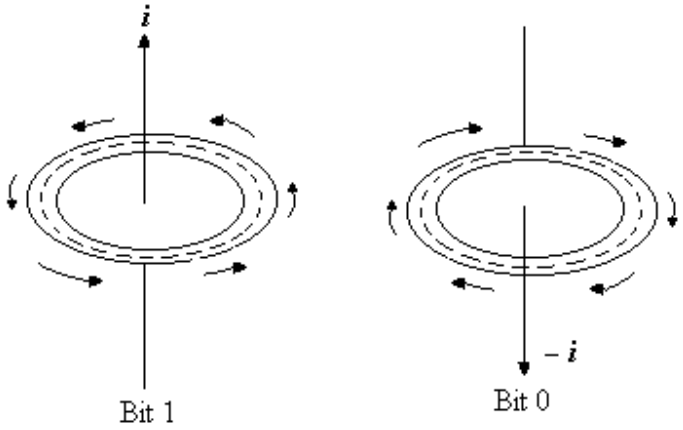


Fig. 12.27

In order to read or sense the bit present in the core, it is necessary to have a sense coil or sense winding as shown in figure 12.28. For reading the bit present in the core a current (- *i*) is passed through its one winding (read /write winding) and the voltage induced across the output or sense winding is detected. Now if a '0' bit is stored in the

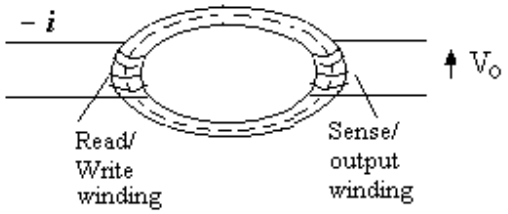


Fig. 12.28

core, then for reading this bit the voltage induced in the sense coil will be very small , as the read current (- *i*) will not cause significant change in its state. Similarly if a '1' bit is stored in the core, read current (- *i*) will induce a significant change un the sense coil. The change in the voltage induced across the sense coil of the core memory will indicate a '0' or '1' bit is stored in the core memory cell. This is illustrated in figure 12.29.

Magnetic core memories are non-volatile read / write memories and were used in main frame computers.

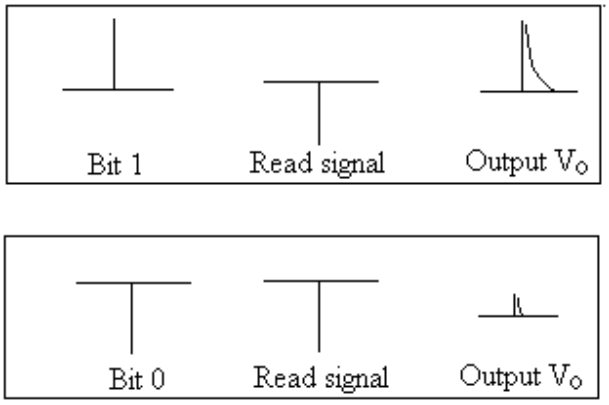


Fig. 12.29

12.7.2 Magnetic Disk Memory:

Magnetic disk storage devices include floppy disk or Hard disk and are used as auxiliary memory in the computers. These devices are less expensive compared with the semiconductor memories. The access time for these devices is very fast. These devices make use of magnetic surfaces. A conducting coil named as Read / write Head is used for writing the data on the magnetic surface and retrieving the data from it. The head remains stationary while the disk rotates below it for reading or writing operation. Figure 12.30 shows the read /write operation on the magnetic surface. To write a data bit on the magnetic surface a current pulse is applied through the coil of the write head (fig. 12.30 a). By the application of this current pulse a small segment of the moving magnetic surface gets magnetized. The direction of the magnetic flux is controlled by the current pulse as per the bit to be stored. Binary '1' is represented by one magnetized polarity of the spot of magnetic surface. Similarly, the other polarity of the magnetized spot of the magnetic surface represents the binary bit '0'. Once a spot of the magnetic surface is magnetized by the write head, it will not be changed until changed again by the write head. The magnetized spot on the magnetic surface produces an induced voltage in the windings of the read head when the surface is passed on the read head. The direction of the output induced voltage pulse will be according to the polarity of the magnetized spot of the magnetic surface. This is the procedure to read the stored content. (fig. 12.30 b). The read and write head are usually combined into a single unit as shown in figure 12.20(c).

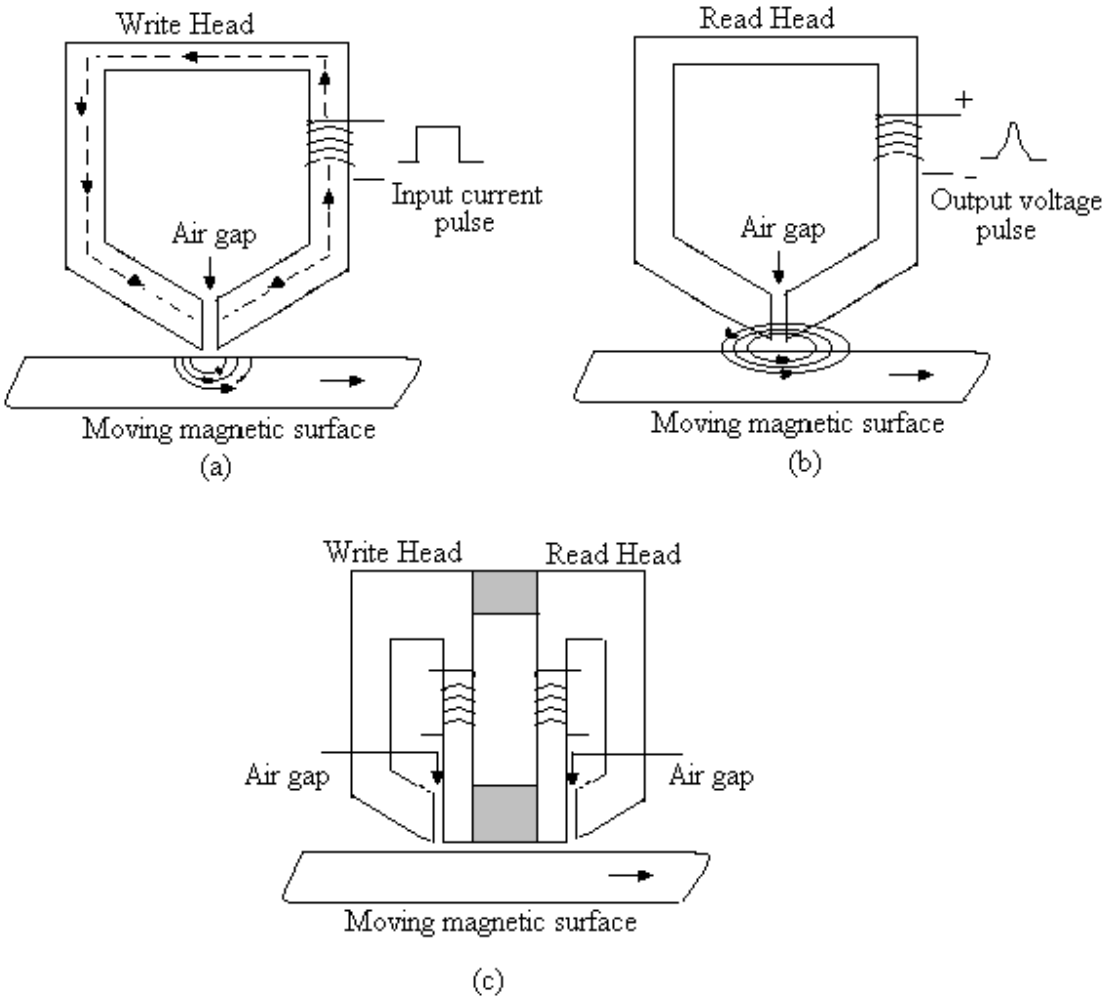


Fig. 12.30

There are several ways to represent the digital data (binary bits 0 or 1) on the magnetic surface. They include: Return to zero (RZ), Non-return to zero (NRZ), Bi-phase, Manchester and Kansas city standards.

Figure 12.31 illustrates a Return to Zero (RZ) waveform. From this figure it is clear that pulse always return to zero after a '1' occurs. For a '0' no pulse occurs during the entire bit time.

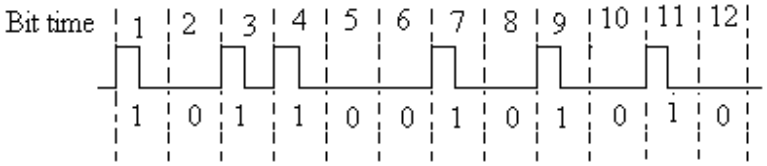


Fig. 12.31

Non return to zero (NRZ) waveform is shown in figure 12.32. It is clear from this figure that pulse remains high or low during the entire bit level for representing a '1' or '0' respectively. In this case, waveform does not return to the 0 level until a 0 occurs.

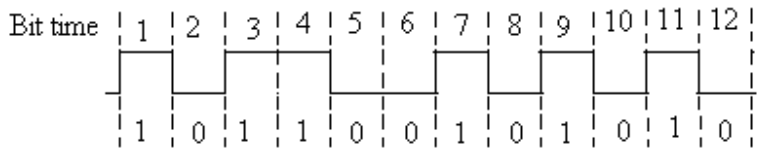


Fig. 12.32

Biphase waveform is illustrated in figure 12.33. For a '1', high level is for the first half of the bit time and low level for second half of the bit time. Similarly, for a '0', low level is for the first half of the bit time and low level for the second half of the bit time. So low to high or high to low transition occurs in the middle of the bit time.

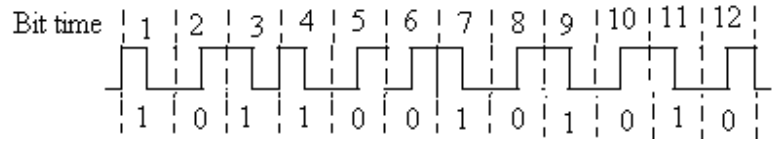


Fig. 12.33

In the Manchester's wave form, at the start of a bit time transition from high to low represents a '0'. If there is no transition it represents a '1'. Manchester's waveform is shown in figure 12.34.

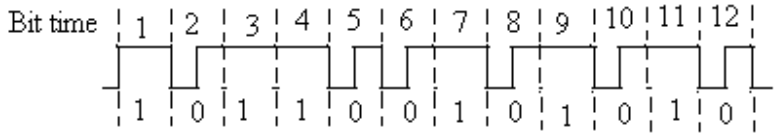


Fig. 12.34

Two different frequencies are used to represent 0s or 1s in the Kansas City method illustrated in figure 12.35. The standard eight cycles of 2.4 KHz frequency are used to represent a '1', and four cycles of 1.2 KHz frequency are used to represent a '0'.

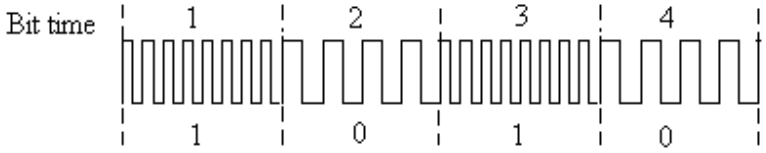


Fig. 12.35

12.7.3 Floppy Disk

The floppy disk is smaller, simpler and cheaper disk unit. It is a flexible Mylar plastic diskette coated with thin film of magnetic material (figure 12.36). This is housed in a square plastic jacket which provides handling protection. A small hole called the index hole in the floppy is used for referencing all the tracks. Through the access window Read / Write head makes contact with the rotating disk. The write protect notch is also provided that can prevent new data to be written on the floppy, for which write protect notch can be covered with a piece of tape. If this notch is not covered with a piece of tape then the fresh data can be written on the floppy several times.

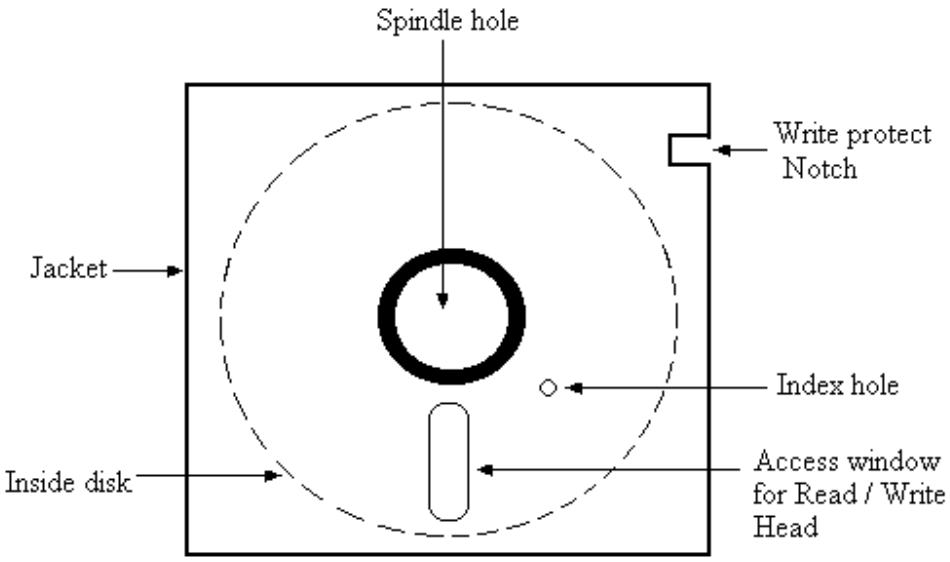


Fig.12.36

The floppies of different sizes are available; 5.25 inch floppy is very popular (figure 12.37). It is organized into 77 tracks and each track is divided into 26 sectors of equal sizes. Each of these sectors can store 128 bytes of data. Total capacity of the disk will therefore be:

$$77(tracks)\,x\,26(\text{sec}\,tors\,/\,track)\,x\,128(bytes\,/\,\text{sec}\,tor) = 256256\,bytes \cong 256\,Kbytes$$

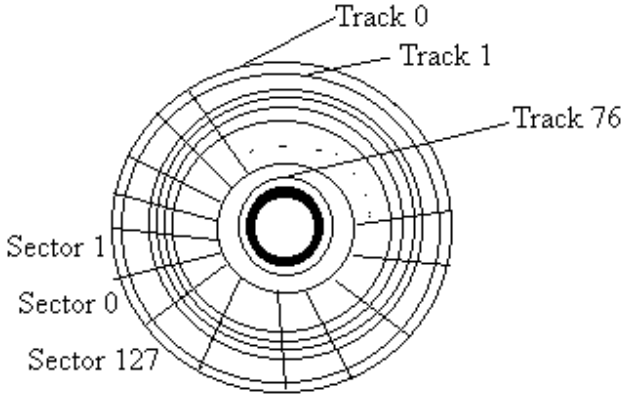


Fig. 12.37

The format writing the data on each sector is divided into different fields as shown in figure 12.38. As per the sequence of rotation as the address mark passes the read/write head, it identifies the up coming areas of the sector as ID field. The ID field identifies the data field by sector and track number. The data mark indicates if contains the good record. The 128 bytes of data can be stored in the data mark which is the part of the sector. The average accessing time of a sector is about 500 ms which slower than the semiconductor memories. The floppy disks are less expensive and portable. The capacity of the floppy disk is very small. However, double density floppy disks are also available which can store 256 bytes per sector with a total capacity of 512512 bytes.

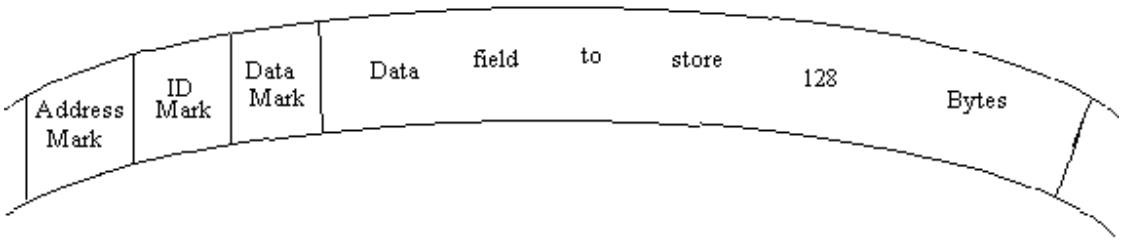


Fig. 12.38

12.7.4 Hard Disk System:

In hard disk system, magnetic disks of smooth metal plates coated on both sides with a thin film of magnetic material are fixed to a rotating shaft. These plates are stacked as illustrated in figure 12.39. The disk pack is mounted on a disk drive. The disk drive consists of a motor to rotate the disk pack about its axis at a speed of about 3600 to 54oo revolutions per minute. The disk pack and a set of magnetic heads mounted on arms are sealed in an enclosure. The access arm assembly is capable of moving in and out in a radial direction. The hard disk data transfer rates are 1M to 10M bits /sec. The hard disk are physically large and bulky so it is quite costly.

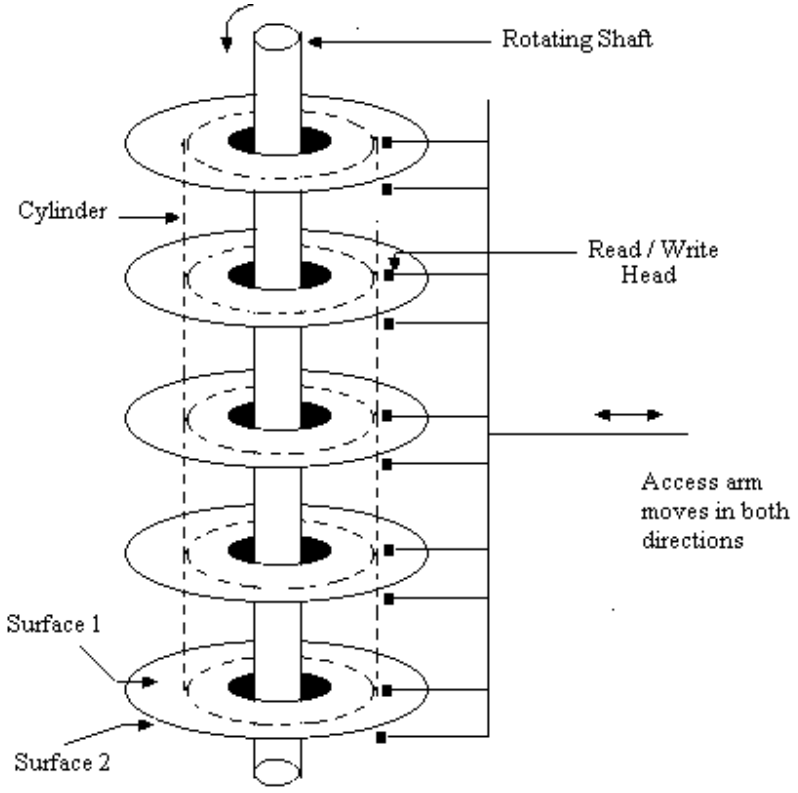


Fig. 12.39

12.8 MAGNETIC BUBBLE MEMORIES :

In some magnetic materials such as garnets on applying magnetic field certain cylindrically shaped domains called magnetic bubbles are created. The direction of magnetization is opposite to that of magnetic field. The diameters of these bubbles are found to be in the range of few micrometers. These bubbles can be moved at high speed by applying parallel magnetic field to the surface of magnetic materials. Thus the rotating field can be generated by an electromagnetic field and no mechanical motion is required. Soft magnetic material is also deposited on this device which forms a pre-determined path called tracks. Magnetic bubbles are forced to move continuously in a fixed direction of these tracks. The presence of a bubble is considered a '1' state, whereas the absence of the bubble is considered a '0' state. For writing data into a cell bubble generator is used to introduce bubbles and a bubble annihilator removes the bubbles. Read operation is performed by a bubble detector.

Magnetic bubble memories having capacities of 1 M or more bits per chip have been manufactured. The cost and performance of these memories fall between semiconductor RAMs and magnetic disks. These memories are non-volatile in contrast semiconductor RAMs. In addition these memories are more reliable than magnetic disks as there are mo moving parts. These memories are difficult to interface with conventional processors. These memories are used in specialized applications where extremely high reliability is required.

12.9 CHARGE COUPLED DEVICES (CCDS):

The charge coupled devices (CCDs) are used to store the data. In these devices the data are stored in the form of charges on capacitors. They have arrays of cells which can hold charge packets of electron. The storage cells do not include transistors like dynamic RAMs. A word is represented by a set of charge packets, the presence of each charge packet represent the bit value '1'. The charge packets do not remain stationary and the cells pass the charge to the neighbouring cells with next clock pulse. As the dynamic RAMs are to be refreshed periodically, the charges in CCDs must also be refreshed periodically. The access time to these devices is not very high. At present this technology is used only in specific applications and commercial products are not available.

12.10 COMPACT DISK READ ONLY MEMORY (CDROM):

The compact disk read only memory falls in the category of optical memories. It is a direct extension of audio CD. This optical technology is the mass storage device capable of storing large data. It can store around 650 Mbytes of data, which is equivalent to 2,50,000 pages of printed text. The CD-ROM disk is normally formed from a resin named polycarbonate which is coated with aluminum to form a highly reflective surface. The data on CD ROM is stored as a series of microscopic pits on this reflective surface. A high intensity laser beam is focused to create pits on the master disk. The circular pit of around 5×10^4 mm sizes is created whenever a 1 is to be written and no pit (also called a land) if a zero is to be written. The master copy of the information is first prepared and from the master disk many copies can be reproduced by a process called stamping a disk. A top coat of clear lacquer is applied on the CD ROMs surface to protect from dust and scratches. The data stored on CD ROMs can be retrieved by a CD ROM reader which uses low powered laser beam. The CD ROM disk is rotated by a motor at a speed of 360

r.p.m. The laser head moves in and out to specified position. As the disk rotates the head senses pits and land. This is converted to 1's and 0s.

PROBLEMS:

1. Define the following terms relating to memory unit.
   Memory address register, memory buffer register, access time of memory, write time of memory, memory cycle time, destructive and non-destructive memory, and volatile memory.
2. What is a memory unit? Explain with block diagram the concept of memory using registers connected to memory unit.
3. What are the sizes of MAR and MBR for a 64K x 8 bit memory?
4. How many words can be stored in 16K x 10 memory unit? How many bits can be stored with this memory unit? What are the sizes of MAR and MBR?
5. A computer memory is to have 8192 words with 16 bits per word. Find how many bits are required for MAR and MBR.
6. Define Read-only-memory. Explain the organization of a diode matrix. ROM.
7. Explain bipolar ROM cell. Draw the block diagram of 8 x 3 ROM.
8. Discuss MOS ROM cell, Draw the block of 8 x 5 MOS ROM matrix.
9. Describe Programmable Read only memory (PROM) using bipolar ROM cells.
10. Describe Programmable Read only memory (PROM) using MOS ROM cells.
11. Describe EPROM and EEPROM. What is the difference between the two.
12. List the application of various types of ROM.
13. Describe the applications of ROM as code converter.
14. Use a 32 x 8 bipolar PROM to form the following functions of five variables:
$$X = \sum (1,2,16,18,19,23,26,27,29)$$
$$Y = \sum (0,3,8,10,18,20,23,24,25,31)$$
$$Z = \sum (3,8,9,10,15,19,21,25,27,28)$$
$$W = \sum (4,5,6,7,11,22,26,29)$$
15. Draw the diode matrix ROM for the conversion of binary number to gray codes.
16. Draw the diode matrix ROM that can implement the cubes of decimal numbers ranging from 0 to 8.
17. How ROM can be used as function generator?
18. Explain how MOS RAM is programmed.
19. Explain the linear selection in a random access memory.
20. Explain the coincident selection in a random access memory.

21. Discuss static bipolar RAM cell for linear selection as well as for coincident selection.
22. Discuss a static MOS RAM cell.
23. Discuss a dynamic MOS RAM cell.
24. What are Random Access Memories? Explain the difference between the bipolar RAMs and MOS RAMs.
25. Discuss the relative merits and demerits of a dynamic RAM cell over static RAM.
26. Draw the block schematic diagram of RAM IC 7489 of 16 x 4 memory.
27. Explain how two 16 x 4 Rams can be connected to use 16 x 8 RAMs.
28. Give the details of dynamic MOS RAM chip 4164 of capacity 64K x 1.
29. Give the combination of 8 PROMs (1K x 8) to produce a total capacity of $K x 8.
30. Give the combination of 4 PROMs (1K x 8) to produce a total capacity of 4K x 8.
31. Discuss the principle and working of magnetic core memory.
32. Discuss the principle and working of magnetic disk memory. Mention different ways to represent digital data on the magnetic surface.
33. Discuss the principle and working of floppy disk.
34. Discuss the principle and working of Hard disk.
35. Write short note on the following:
    (i) Charge coupled Devices (CCDS**)**
    (ii) Compact Disk Read Only Memory (CDROM):

Appendix – I

COMMONLY USED TTL ICs

Number	Description
7400	Quad two-input NAND gates
7401	Quad two-input NAND gates with open collector
7402	Quad two-input NOR gates
7403	Quad two-input NOR gates with open collector
7404	Hex Inverter
7405	Hex Inverter with open collector
7406	Hex inverter Buffer/driver
7407	Hex buffer drivers open collector
7408	Quad two-input AND gates
7409	Quad two-input NAND gates with collector
7410	Triple three-input NAND gates
7411	Triple three-input AND gates
7412	Triple three-input NAND gates with open collector
7413	Dual four-input Schmitt trigger NAND gates
7414	Hex Schmitt trigger inverters
7416	Hex inverter buffer /driver with open collector high voltage output
7417	Hex buffer /driver with open collector high voltage output
7420	Dual four-input NAND gates
7421	Dual four-input AND gates
7427	Triple three-input NOR gates
7430	Single eight-input NAND gate
7431	Quad two-input OR gates
7440	Dual four-input NAND buffer
7441	1-of-10 line decoder/driver
7442	1-of-10 line decoder/driver
7446	BCD to seven segment decoder/drivers (active low outputs
7447	BCD to seven segment decoder/drivers (active low outputs)
7448	BCD to seven segment decoder/drivers (active high outputs)
7470	Edge triggered J K flip-flop
7472	Master slave J K flip-flop with AND inputs

7473	Dual master slave J K flip-flops with separate clears and clocks
7474	Dual edge triggered D-type flip-flops
7475	Four bit latch
7476	Dual master slave J K flip-flops with separate presets, clears and clocks
7483	Four bit full adder
7485	Four bit magnitude comparator
7486	Quad Ex-OR gates
7489	16 x 4 bit RAM
7490	Decade counter
7491	Eight bit serial shift register
7492	Divide by twelve counter
7493	Four bit binary counter
7494	Four bit shift register
7495	Four bit Right - Left shift register
74107	Dual J K master slave flip-flops
74121	Monostable multivibrator
74141	BCD to decimal decoder/driver
74145	1- of -10 line decoder/driver
74150	16 - input multiplexer
74151	8 – input multiplexer
74152	8 – input multiplexer
74153	Dual 4 - input multiplexer
74154	4 – to – 16 line decoder/demultiplexer
74164	8 – bit serial to parallel converter
74165	8 – bit parallel to serial converter
74176	BCD decade counter
74177	Binary counter
74180	8 - bit parity generator/checker
74181	4 – bit ALU
74190	Synchronous Up/Down decade counter
74191	Synchronous Up/Down binary counter
74192	Synchronous Up/Down BCD counter
74195	4 – bit parallel shift register
74196	Decade counter (presettable)
74198	8-bit shift register
74246	BCD –to- seven segment decoder/driver
74290	Decade counter
74293	4 – bit binary counter
74393	Dual 4 – bit binary counter

Appendix – II

COMMONLY USED CMOS ICs

Number	Description
4000	Dual 3 – input NOR gare + inverter
4001	Quad 2 – input NOR gate
4002	Dual 4 – input NOR gate
4006	16 bit Static shift register
4008	4 – bit Full Adder
4009	Hex inverter/buffer
4010	Hex buffer
4011	Quad 2 – input NAND gate
4012	Dual 4 – input NAND gate
4013	Dual D-type flip-flop
4014	8 –bit static shift register, synchronous
4015	Dual 4 –bit static shift register
4016	Quad analog switch/analog multiplexer
4017	Decade counter
4018	Presettable divide – by – n counter
4019	Quad AND/OR gate
4020	14-bit binary counter
4021	8 –bit static shift register, Asynchronous
4022	Octal counter/divider
4023	Triple 3-input NAND gate
4024	7 – stage ripple counter
4025	Triple 3 – input NOR gate
4026	Decade counter/7 segment decoder
4027	Dual J K flip-flop
4028	BCD – to – decimal decoder
4029	4 bit presettable Up/down counter
4030	Quad Ex – OR gate
4031	64 – bit static shift register
4032	Triple serial adder
4033	Decade counter/7 segment decoder with ripple blanking
4034	8 – bit universal bus register
4035	4 bit shift register
4036	4 x 8 bit static RAM
4037	Triple AND/OR gate
4038	Triple serial adder
4040	12 – bit binary counter
4041	Quad true/complement buffer
4042	Quad latch
4043	Quad NOR R S latch

4044	Quad NAND R S latch
4045	21 – stage counter
4046	Phase locked loop
4047	Monostable /Astable multivibrator
4048	8 – input multi function gate
4049	Hex inverter / buffer
4050	Hex buffer
4051	8 – channel analog multiplexer
4052	Dual 4 – channel analog multiplexer
4053	Triple 2 – channel analog multiplexer
4054	4 – segment liquid crystal display driver
4055	BCD – to – 7 segment decoder for multiplexed display
4056	BCD – to – 7 segment decoder / latch
4059	Programmable divide – by - counter
4060	14 stage counter/divider/oscillator
4066	Quad analog switch
4068	8 – input NAND gate
4069	Hex inverter
4070	Quad Ex – OR gate
4071	Quad 2 –input OR gate
4072	Dual 4 –input OR gate
4073	Triple 3 – input AND gate
4075	Triple 3 – input OR gate
4076	Quad D-type register
4077	Quad Ex – NOR gate
4078	8 – input NOR gate
4081	Quad 2 – input AND gate
4082	Dual 4 – input AND gate
4085	Dual AND/OR inverter gate
4086	Dual AND/OR inverter gate
4089	Binary multiplexer
4095	J K master slave flip-flop
4096	J K master slave flip-flop
4097	8 - channel multiplexer/demultiplexer
4098	Dual Monostable multivibrator
4099	8 – bit addressable latch
4510	BCD up/down counter
4511	BCD – to – 7 segment latch/decoder/driver
4512	8-channel data selector
4513	BCD – to – 7 segment latch/decoder/driver with ripple blanking
4514	4 – to – 16 line decoder with latch
4515	4 – to – 16 line decoder with latch
4516	Binary up/down counter
4518	Dual BCD counter

4520	Dual binary counter
4521	24-stage frequency divider
4532	8 – bit priority encoder
4537	256 x 1 bit RAM
4543	BCD – to – 7 segment latch/decoder/driver
4544	BCD – to – 7 segment latch/decoder/driver with ripple blanking
4547	BCD – to – 7 segment latch/decoder/driver
4552	64 x 4 bit static RAM
4556	Dual 2 – to – 4 demultiplexer
4559	Successive approximation register
4560	NBCD adder
4581	4 - bit ALU
4585	4 – bit magnitude comparator
4720	256 x 1 bit RAM
40160	Synchronous programmable decade counter
40161	Synchronous programmable binary counter
40162	Synchronous programmable decade counter
40192	Programmable up/down decade counter
40193	Programmable up/down binary counter
40194	4 – bit bidirectional universal shift register
40195	4 – bit universal shift register
40373	Octal transparent latch
40374	Octal D-type flip-flop