NATIONAL OPEN UNIVERSITY OF NIGERIA

SCHOOL OF SCIENCE AND TECHNOLOGY

COURSE CODE: PHY 405

COURSE TITLE: ELECTRONICS III

Course Code	PHY 405
Course Title	ELECTRONICS III
Course Developer	DR. A. B. Adeloye
	Dr. M. A. Chendo

PHYSICS DEPARTMENT
UNIVERSITY OF LAGOS

Programme Leader	Dr. Ajibola S. O.
	National Open University of Nigeria
	Lagos

DIGITAL ELECTRONICS

Course Introduction

In the previous two previous courses: Network Analysis and Devices; and Electronics I, we dealt with analogue electronics in which the inputs and output are analogue (continuously varying) signals. In this course, we will be studying digital circuits where the signals are discrete.

The origin of digit was in the caves, thousands of years before written-history, when man learnt to count on the fingers (digits). The basic number names, are therefore, known as digits. There are different numbering systems followed in digital electronics. In Unit 1 you will be introduced to some of the important number systems used. We will learn how to convert numbers from one system to another. We will discuss binary number and some mathematical operations using them.
In Unit 2 we will introduce some of the circuits that are able to operate on binary numbers to perform a logical function. These circuits are called electronic gates. Also you will be familiarised with Boolean algebra which is used in digital systems. After having learned about different types of gates you will be introduced to the flip-flop, which can be built using gates.

In Unit 3 we will study counters which are used for counting the digital pulses and registers which are used to store binary information. Many digital systems include some form of memory, where data can be held on a permanent or a temporary basis. There are different types of memories used in a digital system. We will learn about semiconductor memories in this Unit. Data from the physical world are usually analogue in form and continuous in time. The digital computer or processor operates with numbers and discontinuous data. To utilize the digital processor in the solution or control of physical problems it requires devices to sample the analogue data and code it in digital form or to perform reverse processing and decoding in conversion of processed information back to analogue form. Therefore, in Unit 3 we have discussed analogue to digital converter and their counterpart digital to analogue converter.

In Unit 4 we will come across many testing, measuring and indicating instruments like the CRO, electronic voltmeter, power meter etc. It would help the students to familiarise themselves with these instruments.

UNIT 1 NUMBER SYSTEM AND CODES

Structure

1.1 INTRODUCTION

The aim of any number system is to deal with certain quantities, which can be measured, monitored, recorded, manipulated arithmetically, observed and utilised. Each quantity has to be represented by its value as efficiently and as accurately as is necessary for any application. The numerical value of a quantity can be basically expressed in either analogue (continuous) or digital (step by step) method of representation.

In analogue method, a quantity is expressed by another quantity which is proportional to the first. For example, the voltage output of an amplifier is measured by a voltmeter. The angular position of the needle of the voltmeter is proportional to the voltage output of the amplifier. Yet another example is of a thermometer. The height to which the mercury rises is proportional to the temperature. In both these examples, the value of voltage and temperature can be anywhere between zero and the maximum limit.

In digital method, the value of a quantity is expressed by some symbols, which are called digits, and not by a quantity which is proportional to the first. In a digital watch, the time, which changes continuously, is expressed by digits which do not change continuously. The hour-digits change every hour, and the minute-digits change every minute. But there is no measurement of time lapsed between two successive minute-digits. If we want to measure time more accurately then use can be made of watches which have second-digits as well. The second-digits change every second. The time passed between two seconds is not measured. If this is to be measured, we have to use sports watches where the time is measured up to 2 decimal places. Thus the time can be expressed by digits which change step by step (discrete). This step, which is an interval of time, in this example, can be made by us as small as necessary. Hence, the analogue, quantities like time can be represented as digital approximations (e.g. 10 hour 40 minutes, or more accurately 10 hour 39 minutes 50 seconds). As is clear from the examples above, the accuracy of the value of an analogue quantity generally depends upon the judgement of the observer.

Many number systems are being used in digital technology. Most common amongst them are decimal, binary, octal, and hexadecimal systems. We are most familiar with the decimal number system, because we use it everyday. In this unit we shall describe these number systems, the conversion of a number from one system to another, and finally binary arithmetic. This unit is intended to provide the first step in our understanding of digital electronics.

In the next unit you will be introduced to some of the gates, which are fundamental in digital electronics. There you will be familiarised with Boolean algebra which is a mathematical method used in the design of digital systems.

Objectives
After studying this unit, you should be able to:

- write binary number and convert it into its decimal equivalent and a decimal number into its binary equivalent,
- explain octal number system, understand octal counting, convert an octal number into its decimal and binary equivalents and decimal and binary numbers into their octal equivalents,
- explain hexadecimal number system, understand hex counting, convert hex number into its decimal, binary and octal equivalents and decimal, binary and octal numbers into their hex equivalents,
- write BCD code and convert a decimal number into its equivalent BCD code and vice versa,
- understand ASCII code,
- learn addition, subtraction, multiplication and division using ;binary numbers.

1.2 BINARY NUMBER SYSTEM

First let us consider the familiar decimal system. In this system there are ten distinct and different digits (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9). For magnitudes greater than 9 the convention is to arrange digits in rows starting with the most significant on the left and concluding with the least significant on the right. The significance is determined by what is called the 'weighting' of a digit. Thus arises the concept of 'tens', 'hundreds', 'thousands', etc. For example 3458 = (3×10^3) + (4×10^2) + (5×10^1) + (8×10^0). Each digit is one of the symbols 0 to 9 and is multiplied by a power of ten, depending upon the position of the digit. Thus, decimal numbers are said to have a base of ten and the multiplying powers 10^0, 10^1, $10^2, 10^3$, etc. are called 'weight' or 'positional values'.

In the binary number system (base of 2), there are only two digits: 0 and 1 and the place values are 2°, 2^1, 2^2, 2^3 etc. Binary digits are abbreviated as bits. For example 1101 is a binary number of 4 bits (i.e., it is a binary number containing four binary digits.)

A binary number may have any number of bits. Consider the number 11001.011. Note the binary point (counterpart of decimal point in decimal number system) in this number. The bit on the extreme right is called least significant bit (LSB) and the bit on the extreme left is called most significant bit (MSB). Each bit has its positional value as shown in Fig. 1.1.

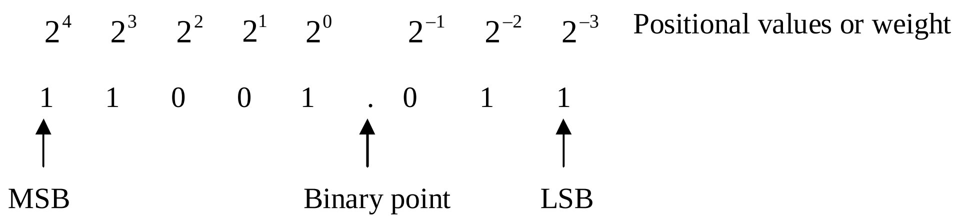


Fig. 1.1 Binary number: showing positional values (weight) of each bit

The bits on the left of the binary point are positive powers of 2 and bits on the right of binary point are negative powers of 2. The decimal equivalent of this number is found by summing the products of each bit and its positional value as follows:

$$\begin{aligned} \text{I}1001.011_2 &= (1\times 2^4) + (1\times 2^3) + (0\times 2^2) + (0\times 2') + (1\times 2^0) + (0\times ^{-1}) + (1\times 2^{-2}) + (1\times 2^{-3}) \\ &= 16 + 8 + 0 + 0+1+0 + 0.250 + 0.125 = 25.375_{10}. \end{aligned}$$

Note that to avoid confusion the subscripts 2 and 10 are written with the numbers to indicate the base of the appropriate number system in which the number is expressed.

Any number can be expressed in binary form in the usual way as shown in Table 1.1

Table 1.1 Counting in Binary System

2^3	2^2	2^1	2°	Binary Number	Decimal Number
0	0	0	0	0000	0
0	0	0	1	0001	1
0	0	1	0	0010	2
0	0	1	1	0011	3
0	1	0	0	0100	4
0	1	0	I	0101	5
0	1	I	0	0110	6
0	1	I	1	0111	7
1	0	0	0	1000	8
1	0	0	1	1001	9
1	0	1	0	1010	10
1	0	1	1	1011	11
1	1	0	0	1100	12
1	1	0	1	1101	13
1	1	1	0	1110	14
1	1	1	1	1111	15

From this Table, note that 4 binary digits are required to do counting up to 15_{10}. Thus if the number of bits is n, then we can go up to 2^n counts and the largest decimal number represented will be $2^n - 1$. For example, in the above case, $n = 4$. Therefore, the largest decimal number represented is $2^4 - 1 = 15_{10}$. To write the next higher number in Table 1.1, we need an additional column for the next power of the base, i.e., 2^4.

SAQ 1
What is the largest decimal number that can be represented using 10 bits?

The advantage of binary system is that it has made the job of designing the digital circuitry very easy because only two distinct states or levels of voltages have to be handled. For example, 'ON' state of a bulb may be represented by the bit ‘1’ and 'OFF' state by '0'. In terms of voltages, 0 V or a 'LOW voltage may represent bit '0' and 5V or a 'HIGH' voltage may represent bit '1'. Actually, it is not necessary also to have precise voltages assigned to each bit. In analogue system the exact value of voltage is very important which makes the design of accurate analogue circuitry very difficult. However, in digital systems exact value of voltage is not important because a voltage of 3.9 V means the same thing as a voltage of 4.4 V or 5 V. This aspect will be dealt with in Unit 3.

Let us now see how binary numbers can be converted into equivalent decimal form and vice-versa.

1.2.1 Binary to Decimal Conversion

From the example discussed above it is clear that a binary number can be converted into its decimal equivalent by simply adding the weights of various positions in the binary number which have bit 1. For example, consider the conversion of 100011.101_2.

1 0 0 0 1 1. 1 0 1

$2^5 + 0 + 0 + 0 + 2^1 + 2^0 + 2^{-1} + 0 + 2^{-3}$

$= 32 + 2 + 1 + 0.5 + 0.125 = 35.625_{10}$

Let us take up another example of conversion of 11100111.0101_3.
1 1 1 0 0 1 1 1. 0 1 0 1

$2^7 + 2^6 + 2^s + 0 + 0 + 2^2 + 2^1 + 2^0 + 0 + 2^{-2} + 0 + 2^{-4}$

$= 128 + 64 + 32 + 4 + 2 + 1 + 0.250 + 0.0625 = 231.3125_{10}$.

Consider the following examples.

1111.00 = 15
11110.0 = 30
111100.0 = 60

From these examples it is clear that if the binary point is shifted towards the right side, then the value of the number is doubled.

Now consider the following examples.

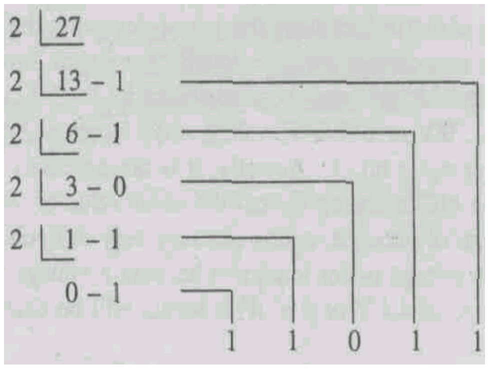
111.100 = 7.5
11.1100 = 3.75
1.11100= 1.875

From these examples it is clear that if the binary point is shifted towards the left side, then the value of the number is halved.

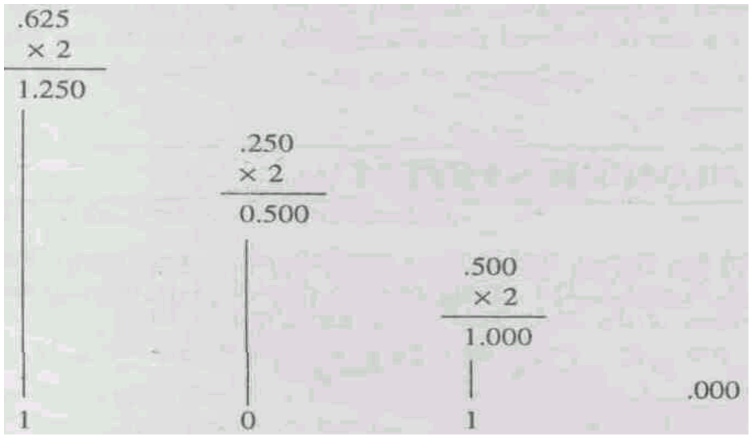
SAQ2
Convert 1011.101 into its decimal equivalent.

1.2.2 Decimal to Binary Conversion

A decimal number is converted into its binary equivalent by its repeated divisions by 2. The division is continued till we get a quotient of 0. Then all the remainders are arranged sequentially with first remainder taking the position of LSB and the last one taking the position of MSB. Consider the conversion of 27 into its binary equivalent as follows.

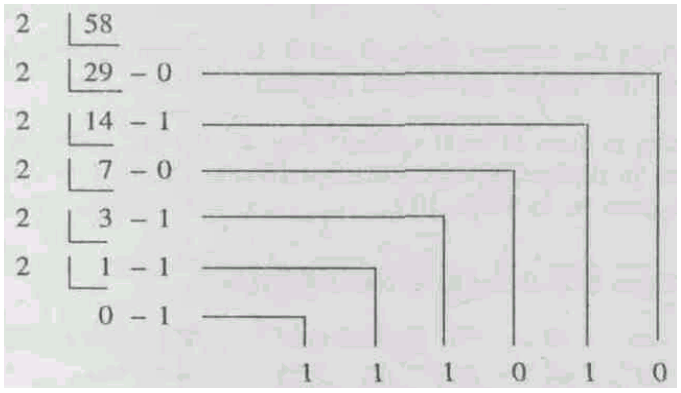


If the number also has some figures on the right of the decimal point, then this part of the number is to be treated separately. Multiply this part repeatedly by 2. After first multiplication by 2, either 1 or 0 will appear on the left of the decimal point. Keep this 1 or 0 separately and do not multiply it by 2 subsequently. This should be followed for every multiplication. Continue multiplication by 2 till you get all 0s after the decimal point or up to the level of the accuracy desired. This will be clear from the following example. Consider the conversion of $27.625_{]0}$ into its binary equivalent. We have already converted 27 into its binary equivalent, which is 11011_2. Now for the conversion of 0.625, multiply it by 2 repeatedly as follows:

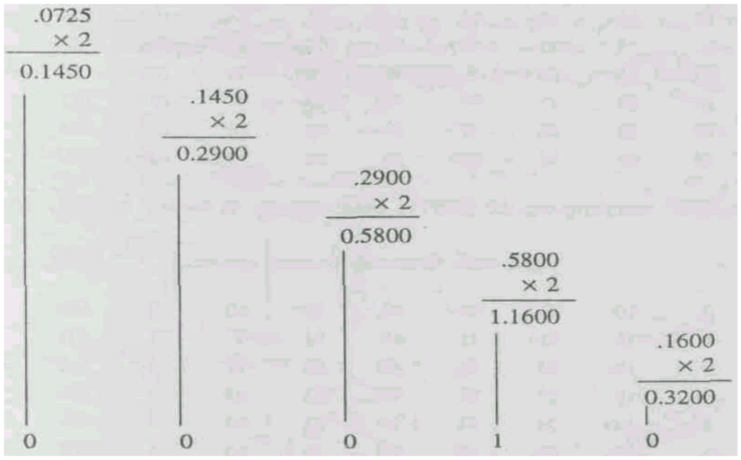


Thus $27.625_{10} = 11011.101_2$.

Let us try another example, conversion of 58.0725_{10} into binary. Split this number in two parts, i.e. 58 and .0725 and convert them into binary separately as described above.



Now take up the conversion of .0725



Thus $58.0725,_{0} = 111010.00010_2$

SAQ3
What is the binary equivalent of 37.75_{10}?

Representing numbers in binary is very tedious since binary numbers often consist of a large chain of 0's and 1's. Imagine the length of the binary equivalent of a 10 digit decimal number! So, convenient shorthand forms for representing the binary numbers are developed such as octal system and hexadecimal system. With these number systems long strings of 0's and 1's can be reduced to a manageable form. Let us see what these systems are.

10.3 OCTAL NUMBER SYSTEM

The octal number system has base-8, that is there are 8 digits in this system. These digits are 0, 1,2, 3, 4, 5, 6, and 7. The weight of each octal digit is some power of 8 depending upon the position of the digit. This is explained in Fig. 1.2.

8^4	8^3	8^2	8^1	8^0		8^{-1}	8^{-2}	Weights
1	0	6	2	7	.	4	5	Octal Number
↑					↑		↑	
MSB					Octal point		LSB	

Fig. 1.2 Octal number: showing positional values (weight) of each digit

Octal number does not include the decimal digits 8 and 9. If any number includes decimal digits 8 and 9, then the number cannot be an octal number.

Now let us see how counting is done in octal system. You are familiar with the counting in decimal system. In decimal system there are 10 digits from 1 to 9 hence the counting in such system is done as in Table 1.2.

Table 1.2: Counting in decimal system

0	10	20	30	40	50	60	7	100	100	…	170
1	11	21	31	41	51	61	7	101	111		
2	12	22	32	42	*52*	62	7	102	112		
3	13	23	33	43	53	63	7	103	113		
4	14	24	34	44	54	64	7	104	114		
5	15	25	35	45	55	65	7	105	115		
6	16	26	36	46	56	66	7	106	116		
7	17	27	37	47	57	67	7	107	117		
8	18	28	38	48	58	68	7	108	118		
9	19	29	39	49	59	69	7	109	119	…	179

In the same style, counting can be done in octal system as shown in Table 1.3

Table 1.3: Counting in octal system

0	10	20	30	40	50	60	70	100
1	11	21	31	41	51	61	71	101
2	12	22	32	42	52	62	72	102
3	13	23	33	43	53	63	73	103
4	14	24	34	44	54	64	74	104
5	15	25	35	45	55	65	75	105
6	16	26	36	46	56	66	76	106
7	17	27	37	47	57	67	77	107

In the octal counting, if n is the number of digits then the total number of counts is 8^n. The largest decimal number represented by an octal number having n digits is $8^n - 1$. Thus with n = 4, the total number of counts is 8^4 = 4096 and the largest decimal number represented is 4096 - 1 = 4095_{10}.

SAQ 4
Can the number 128.96 be an octal number?

SAQ 5
What is the largest decimal number that can be represented by a three digit octal number?

1.3.1 Octal to Decimal Conversion

As has been done in the case of binary numbers, an octal number can be converted into its decimal equivalent by multiplying the octal digit by its positional value. For example,

$$\begin{aligned}126.25_8 &= (1\times8^3) + (2\times8^1) + (6\times8^0) + (2\times8^{-1}) + (5\times8^{-2})\\ &= 64 + 16 + 6 + 0.25 + 0.078\\ &= 86.328_{10}\end{aligned}$$

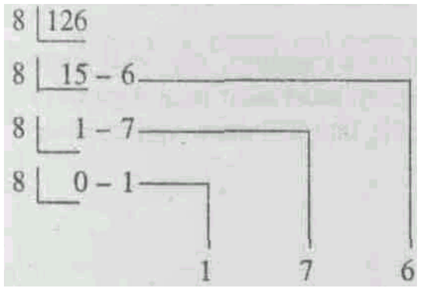
Let us convert 36.48 into decimal number.

$$\begin{aligned}36.4_8 &= 3\times8^1 + 6\times8^\circ + 4\times8^{-1}\\ &= 24 + 6 + 0.5\\ &= 30.5_{10}\end{aligned}$$

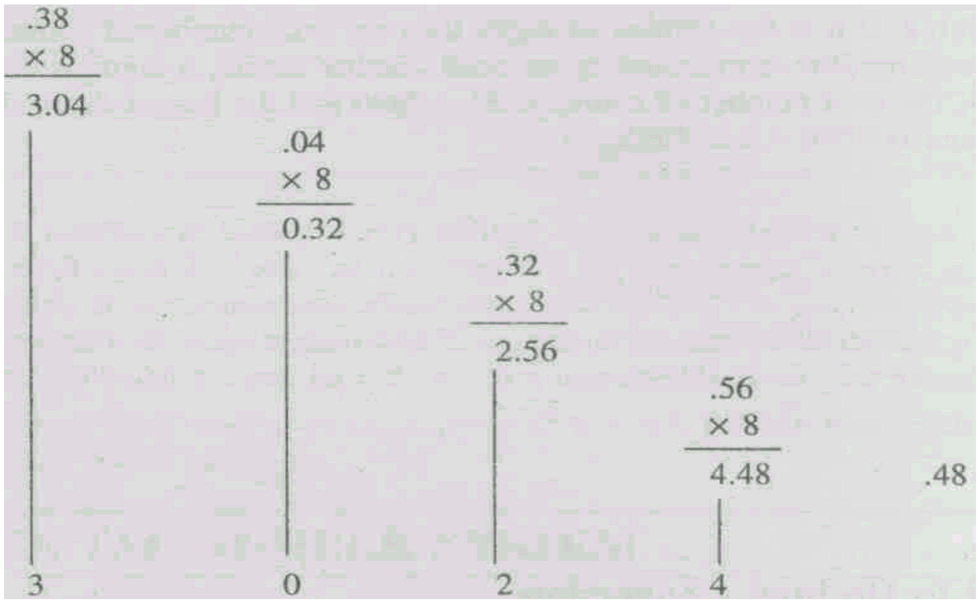
S4Q 6
What is the decimal equivalent of 37.2_8?

1.3.2 Decimal to Octal Conversion

A decimal number can be converted by repeated division by 8 into the equivalent octal number. This method is similar to that adopted in decimal to binary conversion. If the decimal number has some digits on the right of the decimal point, then this part of the number is converted into its octal equivalent by repeatedly multiplying it by 8. The process is same as has been followed in the binary number system. Consider the conversion of 126.38_{10} into its decimal equivalent. Split it into two parts, that is 126 and .38



Now the conversion of .38 is as follows:



Thus $126.38_{10} = 176.3024_8$

SAQ 7
What is the octal equivalent of 15.250_{10}?

1.3.3 Octal to Binary Conversion

In the octal number system the highest octal digit, i.e., 7 can be expressed as a 3-bit binary number. Therefore, all the octal digits have to be represented by a 3-bit binary number. The binary equivalent of each octal digit is shown in Table 1.4. The main advantage of the octal number system is the easiness with which any octal number can be converted into its binary equivalent.

Table 1.4: Binary equivalent of each octal digit

Octal digit	**3-bit binary equivalent**
0	000
1	001
2	010
3	01l
4	100
5	101
6	110
7	111

Using this conversion of octal digit into 3-bit binary number, any octal number can be converted into its binary equivalent by simply replacing each octal digit by a 3-bit binary number. For example, conversion of 567_8 into its binary equivalent is:

$567_8 = 101\ 110\ 111$

$= 101110111_2$

Thus $567_8 = 101110111_2$

Another example:

Conversion of 672.27_8 into its binary equivalent.

$672.27_8 = 110\ 111\ 010.010\ 111$

$= 110111010.010111$

SAQ 8
Represent 10027.12_8 in binary number.

1.3.4 Binary to Octal Conversion

A binary number can be converted into its octal equivalent by first making groups of 3-bits starting from the LSB side. If the MSB side does not have 3 bits, then add 0s to make the last group of 3 bits. Then by replacing each group of 3 bits by its octal equivalent, a binary number can be converted into its binary equivalent. For example, consider the conversion of 1100011001_2 into its octal equivalent as follows:

1100011001_2 = 1 100 011 001

= 001 100 011 001 [As the MSB side does not have 3 bits, we have added two 0's to make the last group of 3 bits]

= 1 4 3 1

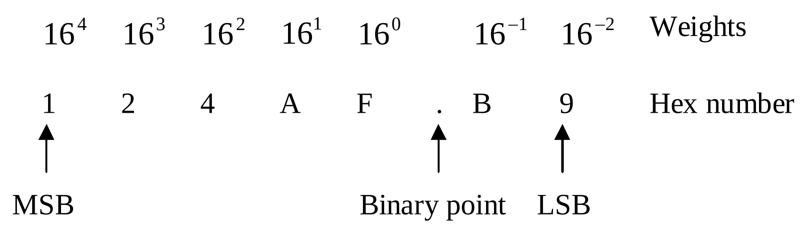
Thus $1100011001_2 = 1431_8$

[As the MSB side does not have 3 bits, we have added two 0's to make the last group of 3 bits]

SAQ 9
What is the octal equivalent of 10010_2?

1.4 HEXADECIMAL NUMBER SYSTEM

The hexadecimal number system has base-16, that is it has 16 digits (Hexadecimal means '16'). These digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F The digits A, B, C, D, E, and F have equivalent decimal values 10, 11, 12, 13, 14, and 15 respectively. Each Hex (hexadecimal is popularly known as hex) digit in a hex number has a positional value that is some power of 16 depending upon its position in the number. This is illustrated in Fig. 1.3.



16^4 16^3 16^2 16^1 16^0 16^{-1} 16^{-2} Weights
1 2 4 A F . B 9 Hex number
MSB
Binary point
LSB

Fig. 1.3: Hexadecimal number: showing positional values (weight) of each digit

The relationship of hex digits with decimal and binary numbers is given in Table 1.5. Note that to represent the largest hex digit we require four binary bits. Therefore, the binary equivalent of all the hex digits have to be written in 4-bit numbers.

Table 1.5: Binary and Decimal equivalent of each Hex Digit

Hex digit	**Decimal equivalent**	**4-bit Binary equivalent**
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001

A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

While doing counting in hex number system if n is the number of hex digits then counting can be done up to 16^n counts and the largest decimal number represented by a hex number is $16^n - 1$. The hex counting is shown in Table 1.6.

Table 1.6: Counting in Hexadecimal system

0	10	20	10	40	…	90	A0	B0	C0	D0	E0	F0	100
1	11	21	31	41	…	91	Al	Bl	C1	D1	E1	F1	
2	12	22	32	42	…	92	A2	B2	C2	D2	E2	F2	
3	13	23	33	43	…	93	A3	B3	C3	D3	E3	F3	
:	:	:	:	:	:	:	:	:	:	:	:	:	
9	19	29	39	49	…	99	A9	B9	C9	D9	E9	F9	
A	1A	2A	3A	4A	…	9A	AA	BA	CA	DA	EA	FA	
B	IB	2B	3B	4B	…	9B	AB	BB	CB	DB	EB	FB	
C	1C	2C	3C	4C	…	9C	AC	BC	CC	DC	EC	FC	
D	ID	2D	3D	4D	…	9D	AD	BD	CD	DD	ED	FD	
E	IE	2E	3E	4E	…	9E	AE	BE	CE	DE	EE	FE	
F	IF	2F	3F	4F	…	9F	AF	BF	CF	DF	EF	FF	

SAQ 10
What is the number next to $835F_{16}$?

SAQ 11
What octal number represented by a 3-digit hex number?

1.4.1 Hex to Decimal Conversion

Hex to decimal conversion is done in the same way as in the cases of binary and octal to decimal conversions. A hex number is converted into its equivalent decimal number by summing the products of the weights of each digit and their values. This is clear from the example of conversion of $514.AF_{16}$ into its decimal equivalent.

$$\begin{aligned} 514.AF_{16} &= 5\times16^2 + 1\times16^1 + 4\times16^0 + 10\times16^{-1} + 15\times16^{-2} \\ &= 1280 + 16 + 4 + 0.625 + 0.0586 \\ &= 1300.6836_{10} \end{aligned}$$

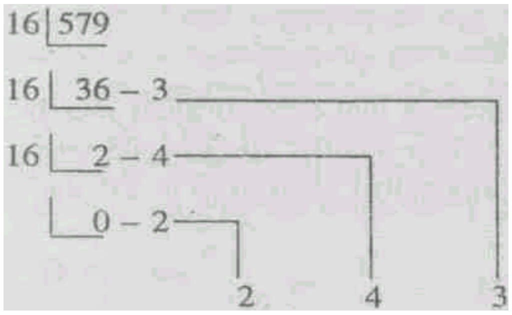
Another example:

$$\begin{aligned} 3BE.1A_{16} &= 3\times16^2 + 11\times16^1 + 14\times16^0 + 1\times16^{-1} + 10\times16^{-2} \\ &= 768 + 176 + 14 + 0.0625 + 0.0391 \\ &= 958.1016_{10} \end{aligned}$$

SAQ 12
What is decimal equivalent of $1BE2_{16}$?

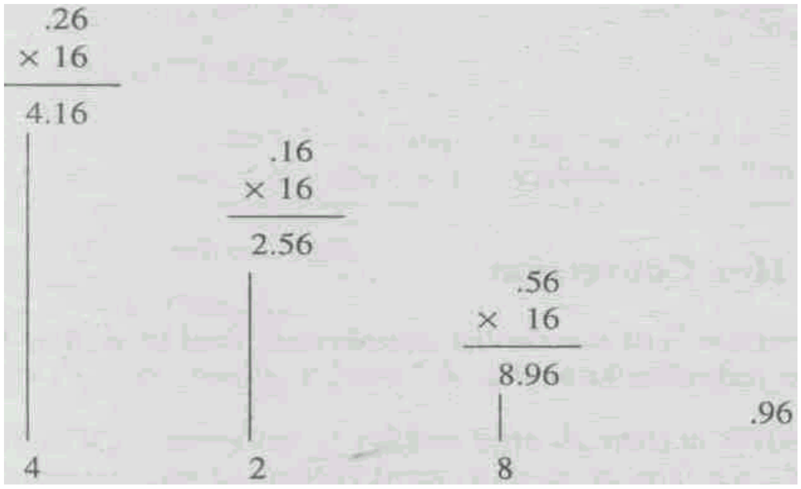
1.4.2 Decimal to Hex Conversion

A decimal number is converted into hex number in the same way as a decimal number is converted into its equivalent binary and octal numbers. The part of the number on the left of the decimal point is to be divided repeatedly by 16 and the part on the right of the decimal point is to be repeatedly multiplied by 16. This will be clear from the examples of conversion of 579.26_{10} into hex equivalent. Split the number into two parts 579 and .26.



Thus $579_{10} = 243_{16}$.

Now .26 is converted into hex number as follows:



Thus $579.26_{IO} = 243.428_{16}$.

SAQ 13
What is the hex equivalent of 37_{10}?

1.4.3 Hex to Binary Conversion

As in octal number system, a hex number is converted into its binary equivalent by replacing each hex digit by its equivalent 4-bit binary number. This is clear from the following example:

$BA6_{16}$ = B A 6

= 1011 1010 0110

= 101110100110_2

SAQ 14
What is the binary equivalent of $6F10_{16}$?

1.4.4 Binary to Hex Conversion

By a process that is the reverse of the process described in section 1.4.4 above, a binary number can be converted into its hex equivalent. Starting from the LSB side, group the binary number bits into groups of four bits. If towards the MSB side, the number of bits is less than four then add zeros on the left of the MSB so that the group of four is complete. Replace each group by its equivalent hex digit. This is clear from the following example:

1001101110_2 = 0010 0110 1110

= 2 6 E

= $26E_{16}$

SAQ 15
What is the hex equivalent of 110010101001111_2?

1.4.5 Hex to Octal Conversion

Each digit of the hex number is first converted into its equivalent four bit binary number. Then the bits of the equivalent binary number are grouped into groups of three bits. Then each group is replaced by its equivalent octal digit to get the octal number. For example,

```
5AF16  = 0101    1010    1111
       = 010110101111
       = 010     110     101     111
       =  2       6       5       7
       = 26578
```

SAQ 16
What is the octal equivalent of $5A9_{16}$?

1.4.6 Octal to Hex Conversion

For octal to hex conversion, just reverse the process described in section 1.4.6 above. This is clear from the following example:

5457_8 = 101 100 101 111

= 1011 0010 1111

= B 2 F

= $B2F_{16}$

This method can also be applied to hex to decimal and decimal to hex conversions. For example consider the conversion of $3C_{I6}$ into its decimal equivalent:

$3C_{16}$ = 0011 1100

$= 111\ 100_2$

Check the conversion.

$3C_{16} = 3\times l6^l + C\times l6^0$

$= 3\times 16' + 12\times 16^0$

$= 48 + 12$

$= 60_{10}$

$111100_2 = 2^s + 2^4 + 2^3 + 2^2$

$= 32+16 + 8 + 4$

Thus $3C_{16} = 111100_2 = 60_{10}$

SAQ 17
What is the hex equivalent of 327_8?

1.5 CODES
So far you have learnt about binary, octal and hexadecimal number system. For any number system with a base B and digits N_0 (LSB), N_1, N_2, ... N_m (MSB), the decimal equivalent N_{10} is given by

$$N_{10} = N_m \times B^m + ... + N_3 \times B^3 + N_2 \times B^2 + N_1 \times B^1 + N_0 \times B^0\ N_3 \times B^3 \qquad (1.1)$$

You have also observed that a number in any system can be written in the binary form. A number code is a relationship between the binary digits and the number represented. Thus, all number systems are codes and the decimal equivalent is given by Eq, (1.1). But there are other relationships or codes that relate decimal numbers and groups of binary digits that do not obey Eq. (1.1) These relationships are called codes. We will now discuss some of the important codes used in digital work.

10.5.1 BCD Code
In BCD (BCD stands for binary coded decimal) code, each digit of a decimal number is converted into its four bit binary equivalent. The largest decimal digit is 9; therefore the largest binary equivalent is 1001. This is illustrated as follows:

951_{10} = 1001 0101 0001

= $10010I010001_{BCD}$

Remember that the conversion of a decimal number into its binary equivalent and BCD equivalent leads to two different numbers. For example:

158_{10} = 0001 0101 1000

= $10I011000_{BCD}$

158_{10} = 10011110_2 (obtained by repeated division method).

Thus we see that it is quite easy to convert from decimal to BCD and from BCD to decimal. It is much easier to convert from BCD to decimal than from straight binary to decimal, because we only have to count up to 9 in binary to do so. However, it takes more bits to represent a number in BCD than in binary.
A BCD number is converted into its decimal equivalent by the reverse process. For example:

$10101011100IO_{BCD}$ = 0001 0101 0111 0010

= 1 5 7 2

= 1572_{10}

Although the main function of a computer is to perform arithmetic operations, it also processes messages and information in a language that uses letters of the alphabet (e.g. English) and data of other kinds. Computers operate by coding letters of the alphabet, other symbols, and data into binary form. The code used for this purpose is the ASCII code, which you will study now.

1.5.2 ASCII Code

The word ASCII is an acronym of American Standard Code for Information Interchange. This is the alphanumeric code most widely used in computers. The alphanumeric code is one that represents alphabets, numerical numbers, punctuation marks and other special characters recognised by a computer. The ASCII code is a 7-bit code representing 26 English alphabets, 0 through 9 digits, punctuation marks, etc. A 7-bit code has 2^7 = 128 possible code groups which are quite sufficient. A partial ASCII code listing is shown in Table 10.6.

Table 1.6: Some of the ASCII codes for numbers, alphabets and other common symbols

$A_6A_5A_4$						$A_3A_2A_1A_0$
010	011	100	101	110	111	
SP	0	@	p		p	0000
1	1	A	Q	a	q	0001
"	2	B	R	b	r	0010
' #	3	C	S	c	s	0011
\$	4	D	T	d	t	0100
%	5	E	U	e	u	0101
&	6	F	V	f	v	0110
'	7	G	w	g	w	0111
- (	8	H	X	h	x	1000
)	9	I	Y	i	y	1001
*		J	Z	j	z	1010
+	;	K		k		1011
,	<	L		l		iioo
-	=	M		m		1!01
	>	N	-	n		1110
/	9	O		o		1111

The code is $A_6A_5A_4A_3A_2A_1A_0$. For example, A has $A_6A_5A_4$ of 100 and an $A_3A_2A_1A_0$ of 0001. Therefore, its ASCII code is

100   0001 = A

The ASCII code for a is 110 0001.

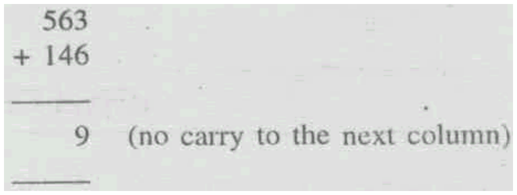
SAQ 18
What is the ASCII code of SHARM?

1.6 BINARY ARITHMETIC

Digital computers can perform arithmetic operations using only binary numbers. We will learn how to add, subtract, multiply and divide binary numbers. We will first review this in the familiar decimal system and apply the same ideas to the binary system.

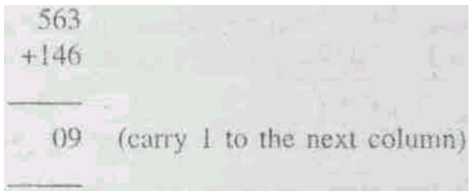
1.6.1 Addition

Let us recall the addition in decimal numbers. Suppose we want to add 563 and 146. We start adding the digits in the least significant column. We get,



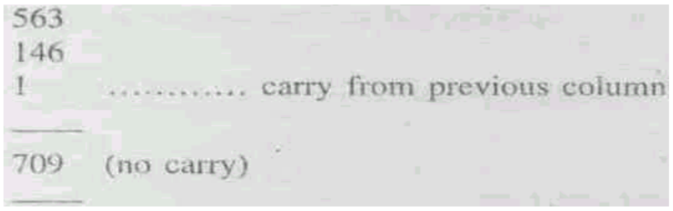
```
  563
+ 146
 ____
    9   (no carry to the next column)
 ____
```

Next, the digits of the second column are added and we get,



```
  563
 +146
 ____
   09   (carry 1 to the next column)
 ____
```

In this case 6 + 4 gives 0, with a carry 1 to the next column. Then the digits of the last column and the 'carry' from the previous column are added. We get,



```
563
146
1       ............ carry from previous column
____
709   (no carry)
____
```

The addition of binary numbers can be carried out in a similar way by the column method. But before we do this, we need to discuss four simple cases. We known in the decimal number system, 3 + 6 = 9 symbolizes the combining of ... with ...... to get a total of .... Let us now discuss the four simple cases.

Case 1: When nothing is combined with nothing, we get nothing. The binary representation of this is 0 + 0 = 0.

Case 2: When nothing is combined with ., we get . Using binary numbers to denote this gives 0 + 1 = 1.

Case 3: Combining . with nothing gives. The binary equivalent of this is 1 + 0 = 1.
Case 4: When we combine . with ., the result is .. Using binary numbers, we symbolize
1 + 1 = 10.

The last result is sometimes confusing because of our long time association with decimal numbers. But it is correct and makes sense because we are using binary numbers. Binary number 10 stands for.. and not for .... (ten).

To summarize our results for binary addition,

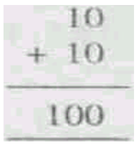
0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

To add large binary numbers, carry into higher-order columns as is done with decimal numbers. As an example, add 10 to 10 as follows



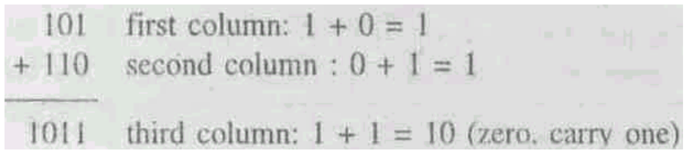
```
   10
 + 10
 ----
  100
```

In the first column, 0 plus 0 is 0. In the second column, 1 plus 1 is 0, carry a 1. As another example, take 1+1 + 1. Add two of the 1's to get 10 + 1.

Adding again gives 11 as follows:

1 + 1 + 1 = 10 + 1 = 11

See another example:



```
  101   first column: 1 + 0 = 1
+ 110   second column : 0 + 1 = 1
-----
 1011   third column: 1 + 1 = 10 (zero, carry one)
```

Further examples are

$$\begin{array}{r} 110 \\ +\ 111 \\ \hline 1101 \\ \hline \end{array} \qquad\qquad \begin{array}{r} 101.011 \\ +\ 111.110 \\ \hline 1101.001 \\ \hline \end{array}$$

In all digital networks or computers only two binary numbers are added at a time. To add more than two numbers, first two numbers are added, then to their sum the third number is added, and so on. Therefore, we should not worry about the addition of more than two numbers. The computer can add numbers in a few microseconds or even less. You will see that the multiplication, division and subtractions are actually done by the computers by way of addition.

SAQ 19
Add the following:

(a) 1010 and 1101

(b) 1011 and 1010

1.6.2 Subtraction

Binary subtraction is done in the same way as in decimal system. Let us recall the decimal subtraction, for example.

$$\begin{array}{r} 111 \\ -\ 101 \\ \hline 010 \end{array} \qquad \begin{array}{ll} \text{first column :} & 1 - 1 = 0 \\ \text{second column:} & 1 - 0 = 1 \\ \text{third column:} & 1 - 1 = 0 \end{array}$$

In this example, a 1 is borrowed from the ten's position giving 16 in the LSD. Then 16-9 = 7. Borrowing a 1 from the ten's position leaves 4 in place of 5. Then 4 - 4 = 0. In the same way the binary subtraction can be done.

To subtract binary numbers, we first need to discuss four simple cases.

Case 1 0 – 0 = 0

Case 2 1 – 0 = 1

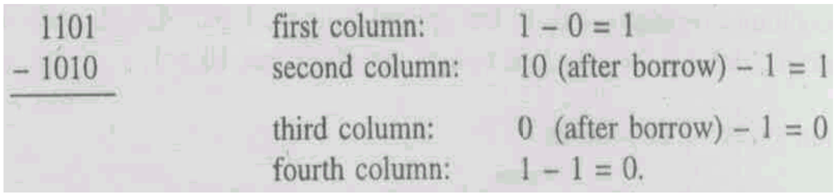
Case 3 1 – 1 = 0

Case 4 10 – 1 = 1

The last result represents .. –. = . which makes sense. To subtract large binary numbers, subtract column by column, borrowing from the adjacent column when necessary. For example, in subtracting 101 from 111, we proceed as follows:

$$\begin{array}{r} 111 \\ -\ 101 \\ \hline 010 \end{array} \qquad \begin{array}{ll} \text{first column :} & 1 - 1 = 0 \\ \text{second column:} & 1 - 0 = 1 \\ \text{third column:} & 1 - 1 = 0 \end{array}$$

Here is another example: subtract 1010 from 1101.



```
  1101     first column:     1 – 0 = 1
– 1010     second column:    10 (after borrow) – 1 = 1
———————
           third column:     0  (after borrow) – 1 = 0
           fourth column:    1 – 1 = 0.
```

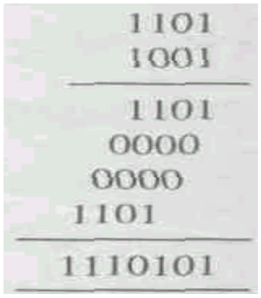
SAQ 20
Subtract binary 100011 from 110011.

1.6.3 Multiplication and Division

The multiplication of binary numbers is also done in the same manner as in decimal system. It is rather easier, because the multiplication table for binary has only four cases.

Case 1	$0 \times 0 = 0$
Case 2	$0 \times 1 = 0$
Case 3	$1 \times 0 = 0$
Case 4	$1 \times 1 = 1$

For example, in multiplying 1101 by 1001, we proceed as follows:



```
      1101
      1001
   ———————
      1101
     0000
    0000
   1101
———————————
   1110101
———————————
```

In the beginning the first partial product is written. Subsequently, each partial product is written below the previous one by shifting one place towards the left relative to the previous place. However, the digital circuits or computers add only two binary numbers at a time. Therefore, to the sum of first two partial products is added the third partial product. To this sum is added the third partial product to give the final sum.

The process of dividing a binary number is once again the same as followed in the decimal system. To divide 1100 by 10, we proceed as follows.

$$\begin{array}{rl}
 & 110 \\ \cline{2-2}
10 \big) & 1100 \\
 & \underline{10} \\
 & 10 \\
 & \underline{10} \\
 & 00
\end{array}$$

SAQ 21
Multiply 10110 by 110.

10.7 SUMMARY

There are mainly four number systems namely, binary, octal, decimal and hexadecimal, which have 2, 8, 10 and 16 digits respectively. But it is the ease in applications that decides which kind of number system should be defined and used. Every computer uses two or more of the above mentioned number systems simultaneously.

The binary number system has only two digits: 0 and 1. A binary digit is called bit. A binary number can be converted into its equivalent octal, decimal and hex numbers as described in the text. And also octal, decimal and hex numbers can be converted into equivalent binary numbers.
The octal number system has 8 digits: 0 through 7. An octal number can be converted into its equivalent binary, decimal and hex numbers and vice versa as described in the text.

The hex number system has 16 digits: 0 through 9, A (10) through F (15). As in the other systems, the hex numbers can be converted as described in the text into their binary, octal and, decimal equivalents and vice versa.

It is possible to arrange sets of binary digits to represent numbers, letters of the alphabet or other information by using a given code. Some of the important codes are the BCD and the ASCII codes.

In the BCD code, each decimal digit is replaced by its 4-bit binary equivalent. The conversion of BCD code into its decimal equivalent and vice versa is quite easy. Therefore, it is quite often used in computers.

The ASCII code is the most widely used alphanumeric code. It is a 7-bit binary number and has $2^7 = 128$ possible 7-bit binary numbers, which are quite sufficient to describe the capital and small letters of the alphabet, digits, punctuation marks, and other symbols.

The fundamental arithmetic of binary addition is contained in four rules:

1. $0 + 0 = 0$

2. $0 + 1 = 1$

3. $1 + 0 = 1$

4. $1 + 1 = 0$ but 1 must be carried over to next higher (more significant) bit.

The fundamental arithmetic of binary subtraction is contained in four rules:

1. $0 – 0 = 0$

2. $0 - 1 = 1$ and borrow 1 from the next more significant bit

3. $1 - 0 = 1$

4. $1 - 1 = 0$

The four rules for binary multiplication are:

1 $0 \times 0 = 0$

2 $0 \times 1 = 0$

3 $1 \times 0 = 0$

4 $1 \times 1 = 1$

1.8 TERMINAL QUESTIONS

(1) In the binary sequence, what is number that follows 10111?

(2) What is the largest decimal number that can be expressed by 6 bits?

(3) Convert 11011011010.1101_2 into its decimal equivalent.

(4) Convert $372.125,_0$ into its binary equivalent.

(5) Convert 89.875_{10} into its binary equivalent.

(6) What is the largest decimal number represented by a five digit octal number?

(7) Convert 7777_8 into its decimal equivalent.

(8) Convert 6789_{10} into its octal equivalent.

(9) Convert 23401_g into its binary equivalent.

(10) Convert $110011011100I010_2$ into its octal equivalent.

(11) Add the following binary numbers 1110001 and 1010101

(12) Multiply 101.1 by 11.01

(13) Divide 11011 by 100

1.9 SOLUTIONS AND ANSWERS

SAQs

1. Largest decimal number $= 2^n - 1$. With $n = 10$, $2^{10} - 1 = 1024 - 1 = 1023_{10}$
2. 11.625_{10}
3. 100101.11
4. No. Octal numbers do not have digits 8 and 9
5. The largest decimal number is $8^3 - 1 = 512 - 1 = 511_{10}$
6. 31.250_{10}
7. 17.2_8

8. 0010000000I0111.001010$_2$
9. 22$_8$
10. 8360$_{16}$
11. Largest decimal number = $16^3 - 1 = 4096 - 1 = 4095_{10}$.
12. $1BE2_{16} = 1\times16^3 + 11\times16^2 + 14\times16^1 + 2\times16^0 = 4096 + 2816 + 224 + 2$
13. 25$_{16}$
14. $6F10_{16}$ = 0110 1111 0001 0000 = 110111100010000_2
15. 110010101001111_2 = 0110 0101 0100 1111 = $654F_{16}$
16. $5A9_{16}$ = 0101 1010 1001
    = 010 110 101 001
    = 2651_8
17. 3278 = 011 010 111
    = 0 1101 0111
    = $D7_{16}$
18. SHARM = 1010011 1001000 1000001 1010010 1001101
19. (a) 10111 (b) 10101
20. 10000
21. 10000100

TQs

(1) 11000_2
(2) 63$_{I0}$
(3) 1754.8125$_{I0}$
(4) 101110100.001_2
(5) 1011001.111_2
(6) 32767_{10}
(7) 4095$_{I0}$
(8) 15205_8
(9) 100 111 00000001$_2$
(10) 1100110111001010_2 = 001 100 110 111 001 010
    = 1 4 6 7 1 2
    = 146712_8
(11) 11000110
(12) 10001.111
(13) 110.11

UNIT 2 FUNDAMENTALS OF BOOLEAN ALGEBRA AND FLIP FLOPS

Structure

2.1 INTRODUCTION

A digital circuit is designed for a desired application by a combination of several logic gates. This application involving several logic gates may be a simple or complex one. Different users may design digital circuits by using different combinations of logic gates for the same application. In selecting one of these digital circuits for that application, it is necessary to keep in mind that the chosen digital circuit should have a minimum number of logic gates. By seeing a digital circuit, it is not obvious that a circuit is minimal or certain gates may be removed from the circuit without changing its operation. Boolean algebra provides a means by which logic circuitry may be expressed symbolically, manipulated and reduced.

In this Unit we shall learn about three basic logic gates: AND, OR, NOT and their various combinations. All digital (logic) circuits operate in the binary mode, where all the inputs and outputs are predefined voltages representing binary digit either 1 or 0. It is this characteristics of the logic circuits that enables us to use Boolean algebra for designing and analysing digital systems. This area of digital circuitry is known as combinational (or combinatorial) logic, where the relationship between the inputs and outputs can be precisely defined by the logic summarised in a truth table.

In the combinational logic circuit, there is no memory, i.e., the output of the digital circuit does not depend upon the occurrence of a previous event. But it is very essential for more advanced digital circuits meant for storing and manipulating information to have memory. The basic memory element is a flip-flop which is obtained by using NAND or NOR gates. In this Unit we shall learn about various kinds of flip-flops and their operation. This area of digital circuitry is known as sequential circuits.

Objectives

- After studying this unit, you should be able to
- describe the operation of AND, OR and NOT Gates and write their truth tables,
- describe the combination of gates and write the truth tables of NAND and NOR gates,
- explain as to how a timing diagram of the output of all the logic circuits is obtained,
- explain how the operation of three basic logic gates leads us to various theorems or rules used in Boolean algebra,
- write Boolean theorems and use algebraic method for combinational logic,
- obtain a truth table from a give Boolean expression,
- describe the operation of exclusive-OR and exclusive-NOR gates,
- design a half adder and describe its operation,
- design a full adder and describe its operation,
- design logic circuits using only NAND gates,
- describe the construction and explain the operation of the RS flip-flop,
- describe the construction and explain the operation of clocked RS flip-flop, D flip-flop, and JK flip-flop,   .
- obtain the timing diagrams of the outputs of flip-flops.

2.2 LOGIC GATES

A logic gate is a digital circuit which has logical relationship between input and output voltages. There are three basic gates: AND, OR and NOT (also called inverter) gates. We shall now learn these gates one by one.

2.2.1 AND Gate

The AND gate can be understood by the circuit given in Fig. 2.1. In this circuit switch (s) is input and the bulb is output. Let us assign 0 to the event when the switch is open and 1 to the event when the switch is closed. Similarly when the bulb does not glow we call it 0 and when the bulb glows we call it 1. With both the switches (A and B) off, the bulb (Y) does not glow.

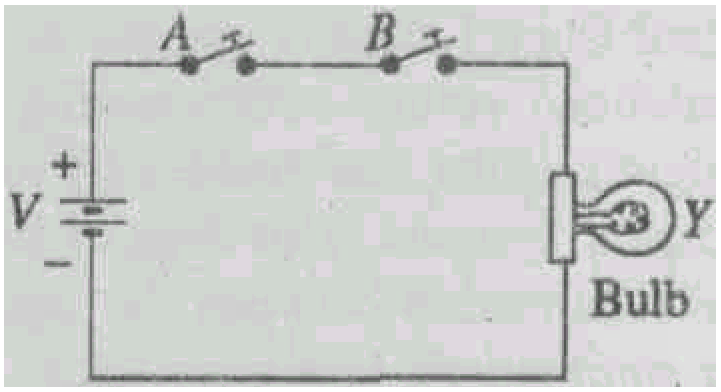


Fig. 2.1 AND gate using switches

With one of the switches off and another switch on, once again the bulb (Y) does not glow. However, with both the switches (A and B) on, the bulb (Y) glows. Thus there are four events which can be summarised in the form of a table which is called the truth table of this circuit. This is given in Table 2.1. The switches A and B, which control the input voltage are usually called the input of the truth table and Y as the output.

Table 2.1: Truth Table of AND Gate

Inputs		**Output**
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Fundamentals of Boolean Algebra and Flip flops
From this table it is clear that the bulb glows (1) only when both the switches (A and B) are on (1). Stated in a different way, the output is 1 when both the inputs A and B are 1. This state of the circuit is distinct from the other three states. This circuit is known as the AND gate. The symbol of AND gate is given in Fig. 2.2. It is clear from the Fig. 2.1 that if the circuit has any number of switches in series, then the output will be 1 if and only if all inputs are 1. Now for all times to come, you must remember that **for an AND gate the output is 1 if and only if all the inputs are 1.**

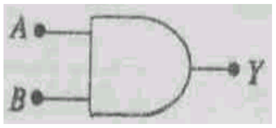


Fig. 2.2 Symbol of AND gate

Electronically the AND gate can be realised by using two pn junction diodes as shown in the circuit of Fig. 2.3. The resistor *R* is used to control the current passing through the diodes. As slated above, a 0 bit is assigned 0V and a 1 bit is assigned 5V. However, such accurate values of voltage will not always be available at the output in electronic circuits. Therefore, a 0 bit is assigned a voltage range of 0 to 0.8V and a 1 bit is assigned to a voltage range of 2.8 to 5.0V. Quite often these voltage ranges are referred to a LOW and a HIGH respectively. The voltages greater than 0.8V and less than 2.8 V are indeterminate and hence not used.

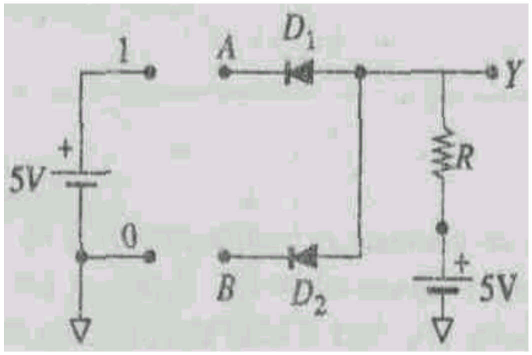


Fig. 2.3 Realisation of AND Gate using diodes

In the circuit of Fig. 2.3 when the inputs A and B are 0, i.e. when they are connected to the 0V or ground terminal, both the diodes are forward biased with a voltage drop of 0.7V across each diode if the diodes are of Si or of 0.3V if the diodes are of Ge. Hence the output voltage is a LOW or a 0 bit. If the input A is 0 and B is 1 (i.e. 5V), the diode A is forward biased with 0.7V drop across it (assuming diode to be of Si) while the diode B is not biased (because both p and n sides of the diode are at the same voltage, 5V). Therefore the output voltage is 0.7V, i.e. a LOW or a 0 bit. Similarly, if the input A is 1 and input B is 0, the output is a 0. However, if both inputs are 1, i.e. connected to 5V, then both the sides of the diodes are at the same voltage and hence not conducting. Therefore, the output voltage is nothing but the battery voltage which is 5V, i.e. a HIGH or a 1 bit. These four cases satisfy the truth table of Table 2.1. For more input AND gate, the number of diodes may be more. The input output relationship of the AND gate is written as A.B = Y and is read as A AND B equal to Y.

Example 2.1
If the inputs A and B to the AND gate are as shown in Fig. 2.4, trace the output Y.

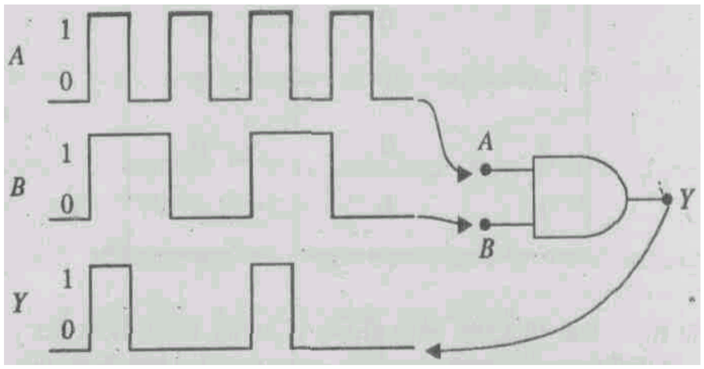


Fig. 2.4

Solution
Recall that the output of an AND gate is 1 when all the inputs are 1. If any of the inputs is 0, then the output is 0. With this understanding, the output comes out to be as shown in the trace for Y.

SAQ 1
Trace the output of an AND gate, if the inputs A and B are as shown in Fig. 2.5.

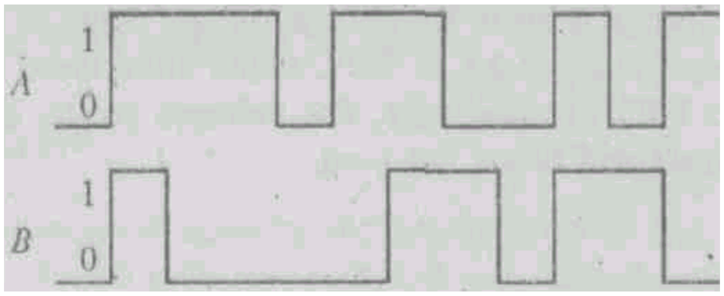


Fig. 2.5

2.2.2 OR Gate
The OR gate operation can be understood by the circuit of Fig. 2.4. If both the switches are off, (0), the bulb does not glow (0). If one of the switches is on (1) and other is off (0), the bulb glows (1). And if both the switches are on (1), then also the bulb glows (1). These events are summarised in the truth table given in Table 2.2.

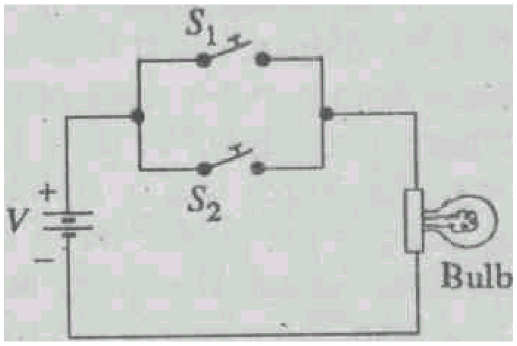


Fig. 2.6 OR gate using switches

Table 2.2: Truth Table of OR Gate

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

It is clear from the truth table that the output of an OR gate is 0 if both the inputs are 0 and the output is 1 if any one of the inputs or both the inputs are 1. If a larger number of switches are used in parallel in the circuit, then the bulb does not glow if all the switches are off, and the bulb glows if any one of the switches is on. The symbol of OR gate is given in Fig. 2.7. The OR gate operation is expressed as A + B = Y and is read as A OR B = Y.

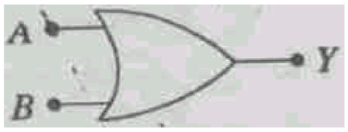


Fig. 2.7 Symbol of OR gate

Electronically OR gate can be realised by using two pn junction diodes as shown in the circuit of Fig. 2.8. If both the inputs are 0, that is connected to ground, then the diodes are not biased and hence no current flows through the diodes. The output is zero or a 0 bit. If the input to diode A is 0 and B is 1 (i.e. 5V), then the diode A is not -biased and thus does not conduct, but the diode B is forward biased with a 0.7V drop across it and 4.3V drop across the resistor. Thus the output is a HIGH or a 1 bit. Similarly, if the inputs to the diode A is 1 and diode B is 0, the output is 1. When the inputs to both the diodes A and B are 1, both the diodes are forward biased, the voltage drop across the resistor R continues to be 4.3V. Hence, the output is a 1 bit. All these four cases satisfy the truth table of OR gate. A more input OR gate is obtained by using more diodes in the circuit. Analysing the truth table of OR gate, we learn that the output is 0 **if both or all the inputs are 0, and the output is 1 if at least one of the inputs is 1.**

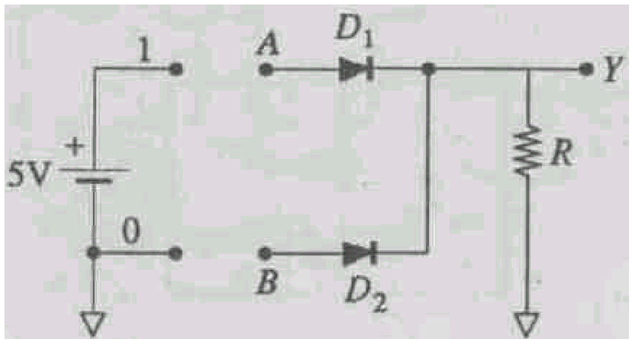


Fig. 2.8 Realisation of OR gate using diodes

Example 2.2
If the inputs A and B to OR gate are as shown in Fig. 2.9, trace the output Y.

Recall that the output of an OR gate is 1 if any of the input is 1, and the output is 0 if all the inputs are 0. With this understanding, the output comes out to be as shown in the trace for Y.

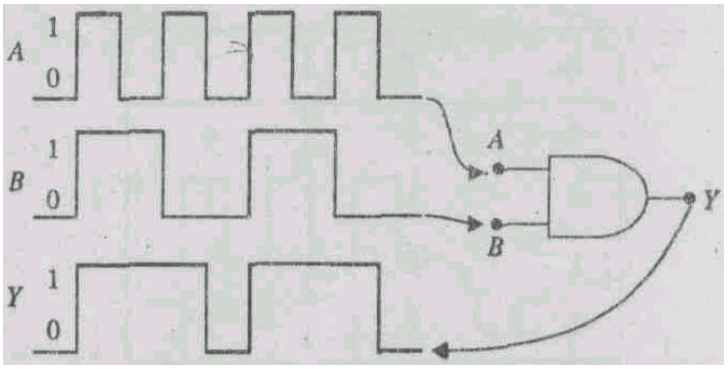


Fig. 2.9

SAQ 2
Trace the output of an OR gate if the inputs A and B are as shown in Fig. 2.10.

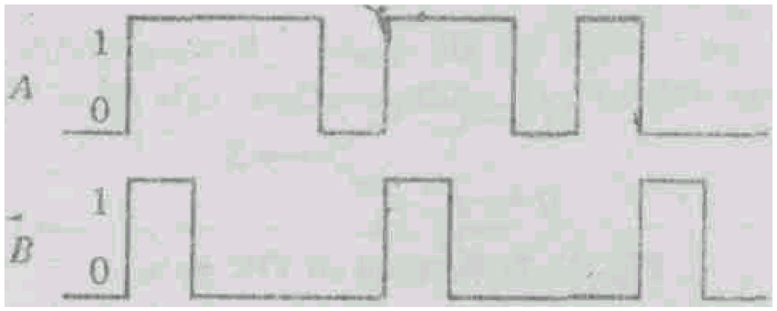


Fig. 2.10

2.2.3 NOT Gate

The NOT gate can be understood by considering the electrical circuit shown in Fig. 2.11, Let us assign a 0 bit to the event when bulb does not glow and 1 bit to the event when bulb glows, and a

0 bit to switch off and 1 bit to switch closed. In Fig. 2.11, when switch is closed, no current will pass through the bulb and the bulb will not glow. This is because the current always flows through the path of least resistance. Similarly, when the switch is open then the **whole** current will flow through the bulb making it glow.

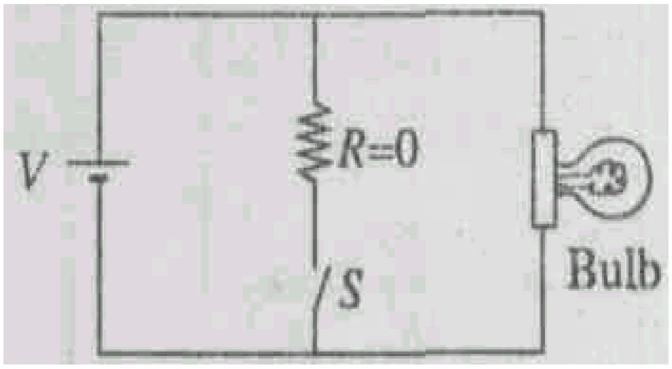


Fig. 2.11 NOT gate using a switch

If input to the circuit is 1, the output is 0 and if the input is 0 then the output is 1. This is the NOT gate operation which is summarised in the truth table given in Table 2.3.

Table 11.3: Truth table for NOT gate

A	Y
0	1
1	0

The NOT gate is also known as the INVERTER. It has only one input. Its symbol is given in Fig. 2.12. The input-output relationship is expressed as A = Y.

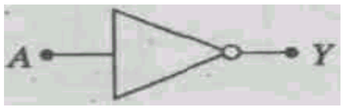


Fig. 2.12 Symbol of NOT gate

The NOT gate can be realised using the circuit given in Fig. 2.13. The circuit uses the cut-off and saturation modes of the transistor. When the input to the circuit is a 0 bit, i.e. zero volt, no base current, I_B. flows. This means the collector current, I_C is zero. This is cut-off mode of the transistor.

Therefore, the output voltage is the bias voltage of 5V indicating the output to be a 1 bit. When the input to the circuit is a 1 bit, i.e. 5V, very large I_B flows resulting in very large I_C, in fact $I_{C\ sat}$. This is the saturation mode of the transistor. This indicates that most of the bias voltage is dropped across R_C, with output to be a 0 bit.

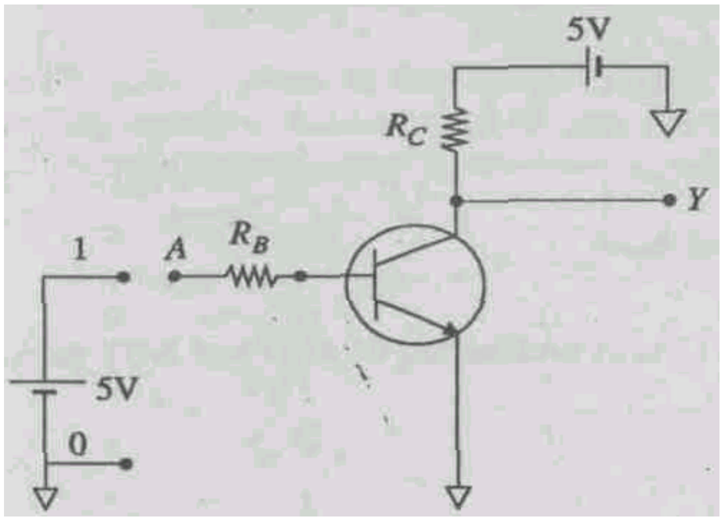


Fig. 2.13 Realisation of NOT gate using a transistor

Example 2.3
If the input A to NOT gate is as shown in Fig. 2.14, trace the output Y.

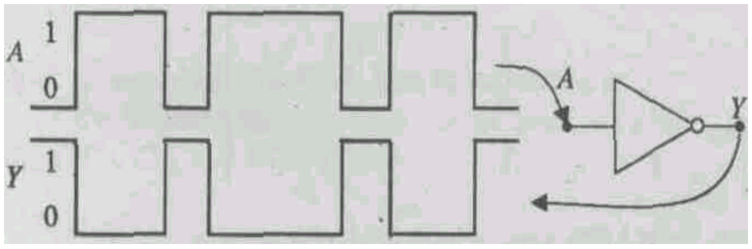


Fig. 2.14

Solution
Recall that the output of a NOT gate is 1 if the input is 0, and the output is 0 if the input is 1. With this understanding, the output comes out to be as shown in the trace for Y in Fig. 2.14.

SAQ 3
Trace the output of a NOT gate if the input is as shown in Fig. 2.15.

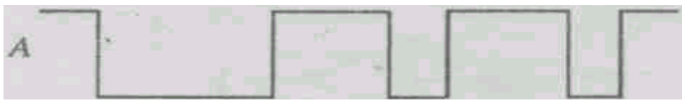


Fig. 11.15

2.2.4 Combination of Logic Gates

The AND, OR and NOT gates are the fundamental gates for all digital circuits. These gates can be combined with each other for a particular application. However, two types of combinations are very important as you will learn now.

(i) NAND Gate

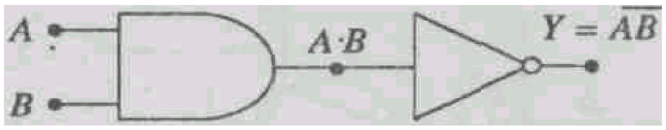


Fig. 2.16 Combination of AND and NOT gate

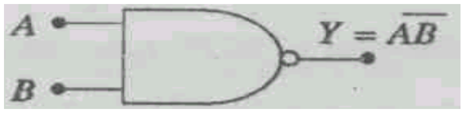


Fig. 2.17 Symbol of NAND gate

If the output of an AND gate is given to the input of a NOT gate, as shown in Fig. 2.16, the resulting circuit is known as NAND gate, the symbol of which is shown in Fig. 2.17. The truth table of this gate is obtained as follows:

A	B	Y' (AB)	Y
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Thus the truth table of NAND gate is shown in Table 2.4.

Table 2.4: Truth table for NAND gate

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

The input-output relationship of a NAND gate is expressed as A.B = Y. The NAND gate is known as the building block for the digital circuits because using NAND gates, one can obtain AND, OR and NOT gates. This aspect will be explained later.

(ii) NOR gate

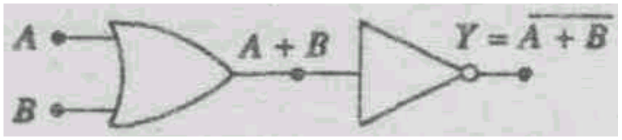


Fig. 2.18 Combination of OR and NOT gate

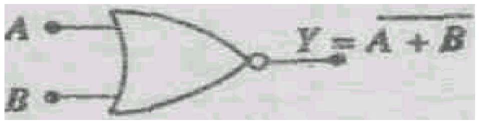


Fig. 2.19 Symbol of NOR gate

If the output of an OR gate is given to the input of a NOT gate, as shown in Fig. 2.18, the resulting circuit is known as NOR gate, the symbol for which is shown in Fig. 2.19. The truth table of this gate is obtained as follows

A	B	Y'(A + B)	Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

The truth table of a NOR gate is shown in Table 2.5.

Table 2.5 Truth table for NOR gate

A	B	Y

The input-output relationship of a NOR gate is expressed as A + B = Y, The NOR is also known as the building block for the digital circuits because using NOR gates one can obtain AND, OR and NOT gates.

Example 11.4
If the inputs A and B to NAND gate are as shown in Fig. 2.20, trace the output Y.

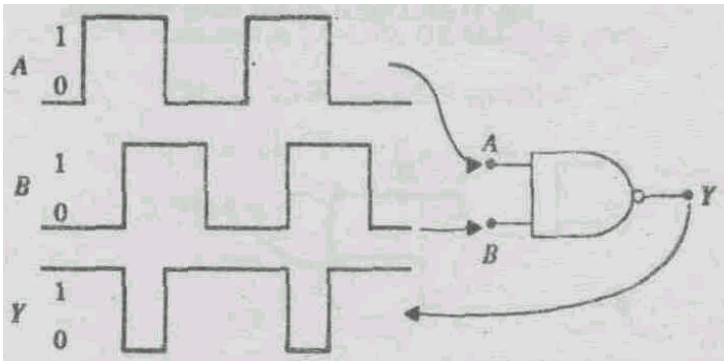


Fig. 2.20

Solution
Recall that the output of a NAND gate is 0 only when all the inputs are 1, and its output is 1 if any or all of the inputs is/are 0. With this understanding, the output comes out to be as shown in the trace for Y in Fig. 2.20.

SAQ 4
If the inputs A and B to a NOR gate are as shown in Fig. 2.21, trace its output Y. (Hint. Apply truth table 2.5).

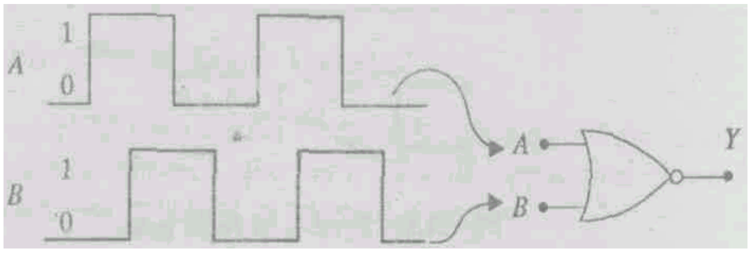


Fig. 2.21

2.3 BOOLEAN ALGEBRA
In this section we shall learn about the Boolean algebra, which provides the methodology for reducing a complex digital circuit into a simple one. This methodology includes the following:

(1) The logic operations are written in the form of a Boolean expression.

(2) From the given truth table, a Boolean expression can be obtained which may not represent a simple circuit having minimum number of gates.

(3) The Boolean expression may then be simplified to get a digital circuit having minimum number of gates.

Consider the digital circuit given in Fig. 2.22. It has five logic gates of three types -

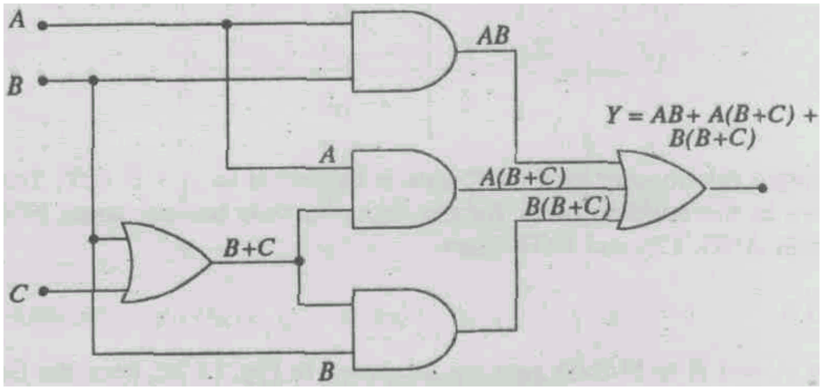


Fig. 2.22: Digital circuit using five gates

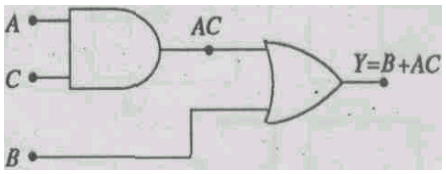


Fig. 2.23: Digital circuit having the same operation as that of the circuit given in Fig. 2.22

three 2-input AND gates, one 2-input OR gate and one 3-input OR gate. Its logic table is given in Table 2.6. This circuit can be reduced to the one shown in Fig. 2.23, which has only two logic gates and is considerably cheaper and simple. It fully satisfies the logic Table 2.6.

Table 2.6

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	I
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	I

The root of its initial assumptions, known as Boolean postulates, lies in the truth tables of the logic gates described in the previous section: Let us recall that the AND operation has been described by the sign of multiplication (.), that is, logical multiplication. Most often we do not use this sign (.), e.g. A.B = AB. Similarly the OR Operation has been described by the sign of addition (+), that is, logical addition. And the NOT operation has been described as a bar $(-)$ over the variable, that is, logical inversion or complementation. These three operations are the basic Boolean operations based upon which we shall develop the Boolean algebra.

Since the number of bits used in binary system is only two, i.e. 0 and 1, there could be only four possible combination of inputs A and B to 2-input AND and OR gates, and two possible inputs to NOT gate. The logical tables of AND, OR and NOT gates are rewritten in Table 2.7.

Table 2.7: Truth tables of AND, OR and NOT gates

AND

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

NOT

0	1
1	0

These logic tables lead to ten postulates of the Boolean algebra, each of which describes the input-output relationship of the concerned logic gate in the form of Boolean expression and is one of the truth table entries for AND, OR, NOT functions. These are:

Table 2.8: Boolean expression for AND, OR and NOT gates

AND operation	OR operation	NOT operation
$0 \cdot 0 = 0$	$0 + 0 = 0$	$0 = 1$
$0 \cdot 1 = 0$	$0 + 1 = 1$	$1 = 0$
$1 \cdot 0 = 0$	$1 + 0 = 1$	
$1 \cdot 1 = 1$	$1 + 1 = 1$	

It is quite clear from these equations that all the four Boolean equations using AND operation satisfy the binary multiplication using bits 0 and 1. However, in the case of OR operation, the first three Boolean equation satisfy binary addition, but the last equation 1 + 1 = 1 does not. It is because in binary arithmetic 1 + 1 = 10. Despite this contradiction between Boolean and binary additions which will be settled later, the Boolean operations are very helpful in digital circuits. The Table 2.8 will lead us to various Boolean theorems, which will be described in the following section.

For the moment let us see how Boolean equations are written and used for a digital circuit. Consider the circuit of Fig. 2.24 in which A and B are the inputs to AND-gate

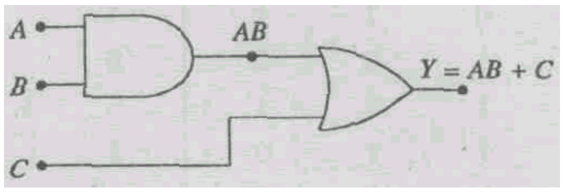


Fig. 2.24 Digital circuit for Y = A-B + C

while C is one of the inputs to OR gate. Another input to OR gate is the output of AND gate, i.e., AB. The output pf this combination is Y, which is

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

Let us find Y if, say, A = 0, B = 1, and C = 1.

$$Y = 0 \cdot 1 + 1$$

From Table 2.8, $0 \cdot 1 = 0$, so

$$Y = 0 + 1$$

From Table 2.8.

$$0 + 1 = 1$$

Hence,

Y = 1

Let us now convert a given Boolean expression into a logic circuit. Say, Y = ($\overline{A} \cdot B$) + ($A \cdot \overline{B}$). The equation means that Y is the output of a 2-input OR gate the inputs t0 which are $\overline{A} \cdot B$ and

$A \cdot \overline{B}$ which in turn are the outputs of two AND gates. The inputs to these AND gates are $\overline{A}$ and B and A and $\overline{B}$ respectively. The whole of this exercise is summarised in the Fig. 2.25.

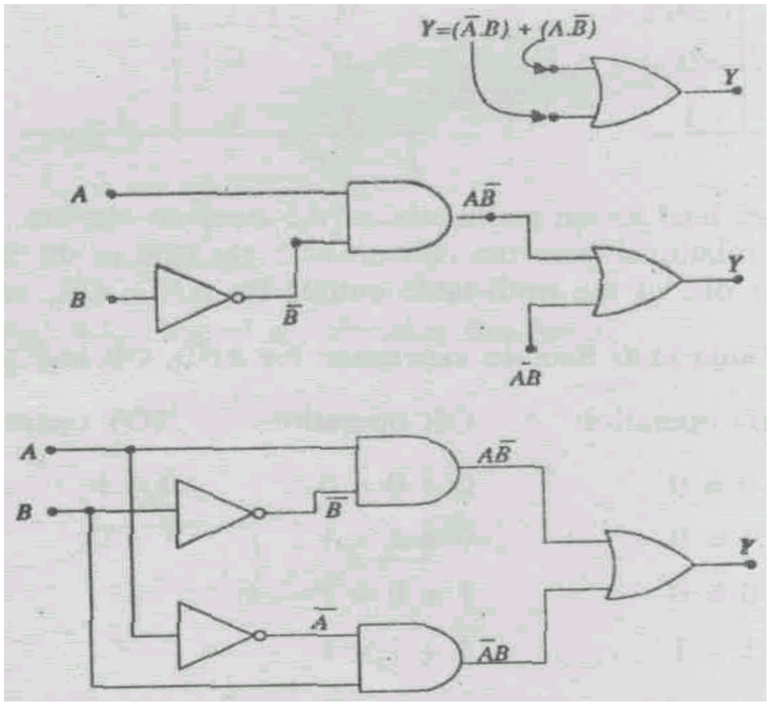


Fig. 2.25 Conversion of a boolean expression Y = $\overline{A}B + A\overline{B}$ a digital circuit

2.3.1 Boolean Theorems

Recalling Table 2.8 we can now write several identities or theorems which are used in Boolean algebra. It is also worthwhile to recall that

A. (i) The output of an AND gate is 1 only when all the inputs are 1.

(ii) The output of an AND gate is 0 when all or any of the inputs is 0.

B. (i) The output of an OR gate is 0 when all the inputs are 0.

(ii) The output of an OR gate is 1 when either of the inputs or all the inputs are 1.

C. The output of a NOT gate is the inversion of its input.

From these conclusions and postulates, we derive the following properties or rules/law/ theorems: From AND function,

1. $X \cdot 0 = 0$

2. $0 \cdot X = 0$

3. $X \cdot 1 = X$

4. $1 \cdot X = X$

From OR functions,

5. $X + 0 = X$

6. $0 + X = X$

7. $X + 1 = 1$

8. $1 + X = 1$

Combination variable with itself or its complement,

9. $X \cdot X = X$

10. $X \cdot \overline{X} = 0$

11. $X + X = X$

12. $X + \overline{X} = 1$

From double complementation,

13. $\overline{\overline{X}} = X$

Commutative laws for multiplication and addition. These laws show that the order in which two variables are ORed or ANDed together makes no difference.

14. $X \cdot Y = Y \cdot X$

15. $X + Y = Y + X$

Associative laws for addition and multiplication. These laws show that while ORing or ANDing several variables, it makes no difference in what order the variables are grouped.

16. $X + (Y + Z) \cdot (X + Y) + Z = X + Y + Z$

17. $X (YZ) = (XY) Z = XYZ$

Distributive laws.

18. $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$

19. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

20. $(W + X) \cdot (Y + Z) = WY + XY + WZ + XZ$

Note here that commutative, associative and distributive laws are similar to ordinary algebra.
Absorption laws. These have no counterpart in ordinary algebra.

21. $X + X \cdot Y = X$

22. $X \cdot (X + Y) = X$

23. $X + XY = X + Y$

24. $X \cdot (\overline{X} + Y) = XY$

De Morgan's theorems. The first theorem says that the complement of a sum is equal to the product of complements:

25. $\overline{X + Y} = \overline{X} + \overline{Y}$

The second theorem says that the complement of a product is equal to the sum of the complements.

26. $\overline{X \cdot Y} = \overline{X} \cdot \overline{Y}$

These theorems are valid even when the variables are expressions. There is no algebraic proof of these theorems. However, each theorem/law can be proved by putting the values (0 or 1) of variables and applying Boolean postulates given in Table 2.8.

2.3.2 Algebraic Method for Combinational Logic

We have now known that a logic circuit can be expressed in the form of Boolean expression which, in turn, can be simplified using Boolean laws. We have also known that a Boolean expression can also be transformed into an equivalent logic circuit.

Before we learn the simplification method and other techniques, let us understand the meaning of combinational logic. Whenever a logic circuit is explicitly defined by its truth table to provide a fixed, invariant relationship between input and output, the circuit is called the combinational circuit. A combinational circuit does not have a memory. It always operates in accordance with its truth table regardless of any prior input which may have been given to the circuit. This will be further understood after we have taken up some examples.

A Boolean expression can be simplified in either of the two forms - (a) Sum of Product (SOP), and (b) Product of Sum (POS). We shall limit ourselves to only SOP form, which is most commonly used. The object of simplification is to minimise the number of variables or occurrences of a variable in an expression. This means minimising operation symbols and hence the number of gates to be used in the circuit. Many a times we get more than one simplified form of an expression, each being equivalent in number of gates and variables to be used. In the final analysis, we shall use the Minimum Sum of Product (MSP) form which is written without brackets. Consider the reduced expression A (B + C) which is written in MSP form AB + AC. While the reduced expression requires one AND gate and one OR gate, the MSP expression requires one AND gates and one OR gate. Thus in this case MSP expression is not the simplest. The fundamental rule is that the expression must be (a) reduced as much as possible, and (b) written without brackets. For the simplification of Boolean expression, Boolean operations should be carried out in the following order:

(1) Inversion of single variables.
(2) All operations with brackets.

(3) AND operations before OR operations.
(4) OR operations.
(5) If an expression is with a bar, then before inverting, perform all operations.

Example 11.5

(1) Find the MSP expression for

Y	$= (\overline{A}+\overline{B})\overline{C} + \overline{AB}$	
	$= (\overline{A}+\overline{B})\overline{C} + (\overline{A}+\overline{B})$	Using De Morgan's theorem Th. 26
	$= (\overline{A}+\overline{B})(\overline{C}+1)$	Taking $(\overline{A}+\overline{B})$ common
	$= (\overline{A}+\overline{B})\cdot 1$	Using Th. 7
	$= (\overline{A}+\overline{B})$	Using Th. 3
	= MSP expression	

The logic circuits for the given and the MSP expressions are shown in Figs. 2.26 and 2.27 respectively.

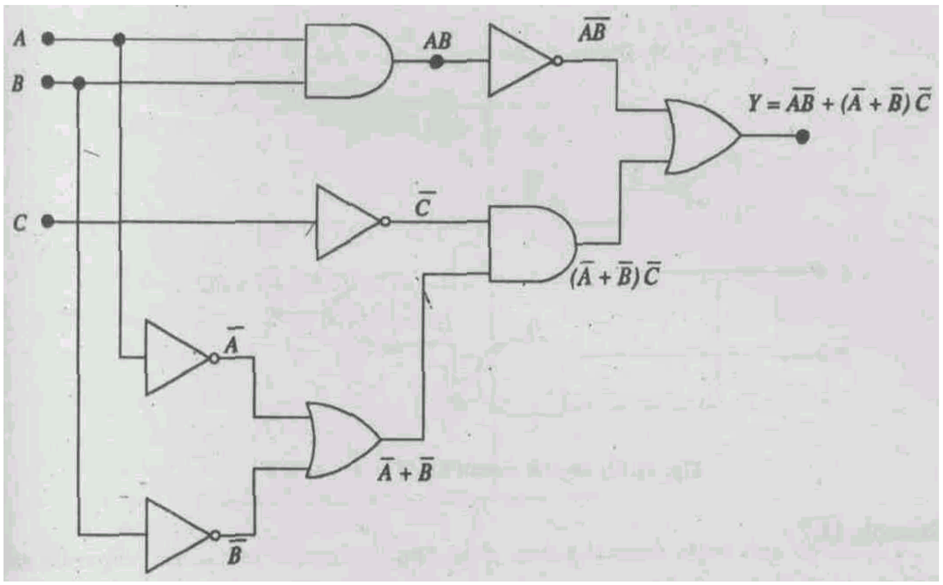

A
B
C
AB
$\overline{AB}$
$Y = \overline{AB} + (\overline{A}+\overline{B})\overline{C}$
$\overline{C}$
$(\overline{A}+\overline{B})\overline{C}$
$\overline{A}$
$\overline{A}+\overline{B}$
$\overline{B}$

Fig. 2.26 Digital circuit for $Y = (\overline{A}+\overline{B})\overline{C} + \overline{AB}$

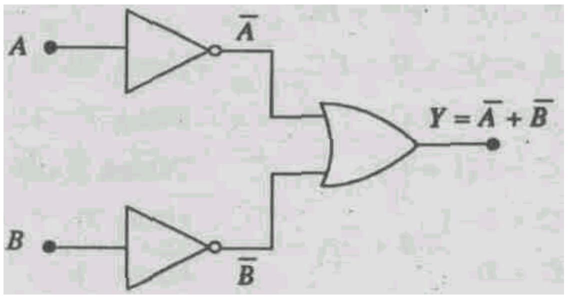


Fig. 2.27 Digital circuit for $Y = \overline{A}+\overline{B}$

Example 2.6

Find the MSP expression for $Y = \overline{A}C + AB(\overline{B} + C)$

Y	$= \overline{A}C + AB(\overline{B} + C)$	
	$= \overline{A}C + AB\overline{B} + ABC$	
	$= \overline{A}C + A \cdot 0 + ABC$	Using Th. 10
	$= \overline{A}C + ABC$	Using Th. 1
	$= (\overline{A} + AB)C$	Taking C common
	$= (\overline{A} + B)C$	Using Th. 23
	$= \overline{A}C + BC$	
	= MSP expression	

Using Th. 10 Using Th. 1 Taking C common Using Th. ?*

The logic circuits for the given expression and the MSP expressions are shown in Figs. 2.28 and 2.29 respectively.

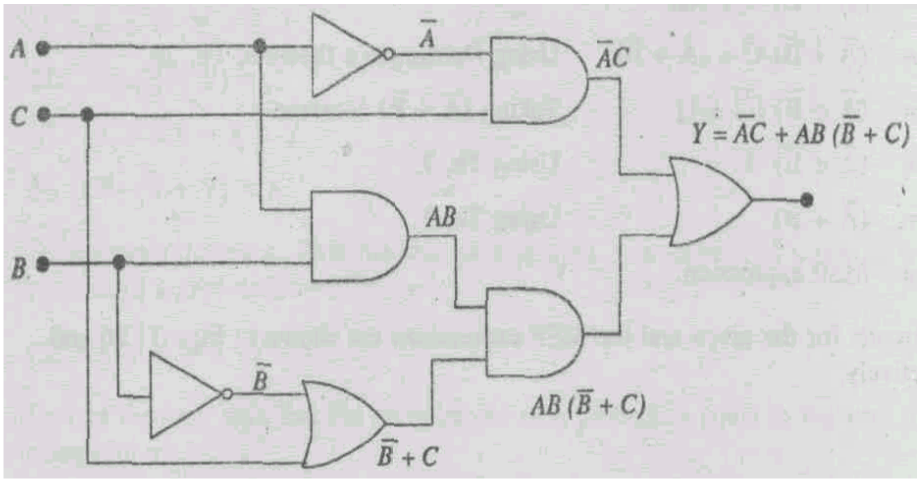


Fig. 2.28 Digital circuit for $Y = \overline{A}C + AB(\overline{B} + C)$

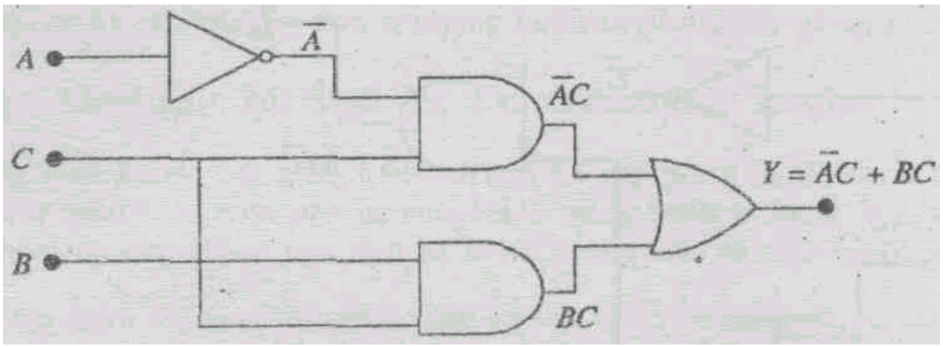


Fig. 2.29 Digital circuit for $Y = \overline{A}C + BC$

Example 11.7

Find the MSP expression for

Y	= AB + A (B + C) + B (3 + C)	
	= AB + AB +AC + BB + BC	
	= AB + AB + AC + B + BC	Using Th. 9
	= AB + AC + B + BC	Using Th. 11
	= AB + AC + B(1 + C)	Taking B common
	= AB + AC + B $\cdot$ 1	Using Th. 8
	= AB + AC + B	Using Th. 3
	= (A + 1) B + AC	
	= 1 $\cdot$ B + AC	Using Th. 7
	= B + AC	Using Th. 4
	= MSP expression	

The logic circuits for the given expression and the MSP expressions are shown in Figs. 2.22 and 2.23 respectively.

SAQ5

Find the MSP expression for $Y = A\overline{B}\,\overline{C} + AB\overline{C} + ABC$.

11.3.3 Obtaining a Truth Table from a Boolean Expression

A simple method of obtaining the truth table from a Boolean expression has already been mentioned. That is, substitute the values of variables in each possible combinations of values in the expression. Perform all the logic operations and get the result for each combination. For example,

Y = AB + A (B + C) + B (B + C)

In this expression, say, A = 1, B = 0, and C = 0, then

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

Similarly, find Y for all combinations of values for A, B, and C, and complete the truth table which is given in Table 11.9.

Table 2.9: Truth table for Y **=** AB + A (B + C) + B (B + C)

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The alternative method of obtaining a truth table from a Boolean expression involves reasoning. Ask yourself:

When shall the output of the expression be 1. Consider the expression

$Y = A\overline{C} + BC$ = MSP expression

This expression is 1 so long as either $A\overline{C}$ or BC is 1. Therefore, put Y = 1 for all entries of $A\overline{C}$ = 1 (i.e., entries 5 and 7). Then put Y = 1 for all entries of BC = 1 (i.e., entries 4 and 8). Now, Y for all other entries is 0. Table 2.10 is thus the truth table for the given expression.

Table 2.10: Truth table for $Y = A\overline{C} + BC$

	A	B	C	Y
1.	0	0	0	0
2.	0	0	1	0
3.	0	1	0	0
4.	0	1	1	1
5.	1	0	0	1
6.	1	0	1	0
7.	1	1	0	1
8.	1	1	1	1

Hence, it is better to use the method of reasoning for obtaining the truth table. This method involves just two steps:
(1) Obtain the MSP form of the given Boolean expression, and
(2) Reason out which of the truth table entries should be 1 for each product in MSP form.

Example 11.8
Obtain the truth table for the Boolean expression Y = A + AB + BCD.

Y = A + AB + BCD
= A (1 + B) + BCD
= A $\cdot$ 1 + BCD
= A + BCD
= MSP expression

Reasoning out we find that Y = 1 whenever A = 1 or the product BCD = 1. Therefore, in the truth table for this expression, put Y = 1 for all entries of A = 1, (i.e., entries 9 to 16) and put Y = 1 for all entries of product BCD = 1 (i.e. entries 8 and 16). For all other entries put Y = 0 (i.e. entries 1 to 7). The complete truth table is given in Table 2.11.

Table 2.11: Truth table for Y = A + AB + BCD

	A	B	C	D	Y
1.	0	0	0	0	0
2.	0	0	0	1	0
3.	0	0	1	0	0

4.	0	0	1	1	0
5.	0	1	0	0	0
6.	0	1	0	I	0
7.	0	1	1	0	0
8.	0	1	1	1	1
9.	1	0	0	0	1
10.	1	0	0	1	1
11.	1	0	1	0	1
12.	1	0	I	1	1
13.	1	1	0	0	1
14.	1	1	0	1	1
15.	1	1	1	0	1
16.	1	1	1	1	1

SAQ 6
Obtain the truth table for Y = AB + BC + CA

2.3.4 Obtaining a Boolean Expression from a Truth Table

Consider the truth table given in Table 2.12.

Table 2.12: Given truth table

	A	B	C	Y
1.	0	0	0	0
2.	0	0	1	0
3.	0	1	0	0
4.	0	1	i	0
5.	1	0	0	1
6.	1	0	1	0
7.	1	1	0	1
8.	1	1	1	1

Note, that the entries 5, 7, and 8 contribute a logic 1 to the operation while all other entries give a logic 0. To obtain the Boolean expression, we need only write a product term for each entry that contribute a logic 1, and then assemble the operations by connecting all the products with a logic OR. Do as follows:

Entry 5: Y = 1 for A = 1, B = 0, C = 0
= $A\overline{B}\overline{C}$

because the output of an AND gate will be 1 only if all the inputs are 1. Similarly,

Entry 7: Y = 1 for A = 1, B = 1, C = 0
= ABC

Entry 8: Y = 1 for A = 1, B = 1, C = 1
= ABC

Now connect all the three products with an OR logic. Hence Y $= A\overline{B}\,\overline{C} + AB\overline{C} + ABC$ (Sum of Product), which can be simplified as

$$\begin{aligned} Y &= A\overline{B}\,\overline{C} + AB(\overline{C} + C) \\ &= A\overline{B}\,\overline{C} + AB \\ &= A(\overline{B}\,\overline{C} + B) \\ &= A(B + \overline{C}) \\ &= AB + A\overline{C} \end{aligned}$$

The procedure can be summarised as follows:

(1) Combine with an AND operation all the input variables for the entries that contribute a logic 1.

(2) Select for each variable in the product an overbar or no overbar so that when the input values of the entries are substituted, the product gives a logic 1. These products are also known as fundamental products.

(3) The products are assembled with an OR operation.
(4) The sum of product expression thus obtained may not be minimal. Use Boolean algebra to bring an SP expression in an MSP form.

SAQ 7
Obtain the Boolean expression for the truth table given below:

A	B	C	Y
0	0	0	0
0	0	1	1
0	0	1	0
0	1	1	0
1	0	0	1
1	1	0	0
1	1	1	1

2.3.5 Exclusive- OR (XOR) Gate
An XOR gate gives a high output (i.e. 1) when an odd number of inputs is high. A two-input exclusive - OR gate has its output 1 if one of the two inputs is 1 and the other is 0, and if both the inputs are same then the output is 0. The truth table of an XOR gate is given in Table 2.13.

Table 2.13: Truth table for XOR gate

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Its Boolean expression is obtained from the entries 2 and 3, that is $Y = \overline{A}B + A\overline{B}$. This expression is in MSP form because it cannot be simplified further. Thus Y is the output of an OR gate the inputs to which are $\overline{A}B$ and $A\overline{B}$, which in turn are the outputs of two AND gates. The circuit thus obtained for an XOR gate is given in Fig. 2.30 and is represented by the symbol shown in Fig. 2.31. The XOR operation is expressed by $\oplus$.

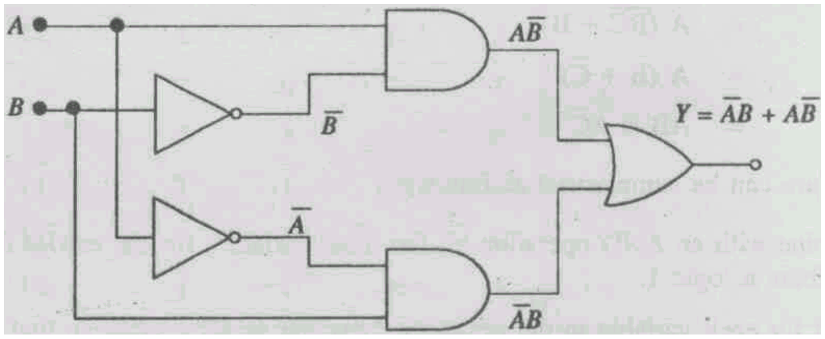


Fig. 2.30 Exclusive -OR (XOR) gate

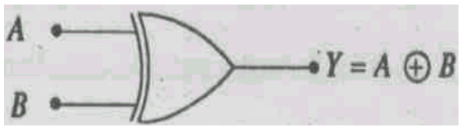


Fig. 2.31 Symbol of XOR gate

2.3.6 Exclusive -NOR (XNOR) Gate

An exclusive-NOR gate has its output 1 if both the inputs are same, and if both the inputs are different then the output is 0. The truth table of an XNOR gate is given in Table 2.14.

Table 2.14: Truth table for XNOR gate

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Its Boolean expression is obtained from the entries 1 and 4, that is,

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

This expression is in MSP form because it cannot be simplified further. Thus Y is the output of an OR gate, the inputs to which are $\overline{A}\overline{B}$ and AB, which in turn are the outputs of two AND gates.

The circuit thus obtained for an XNOR gate is given in Fig. 2.32 and is represented by the symbol shown in Fig 2.33.

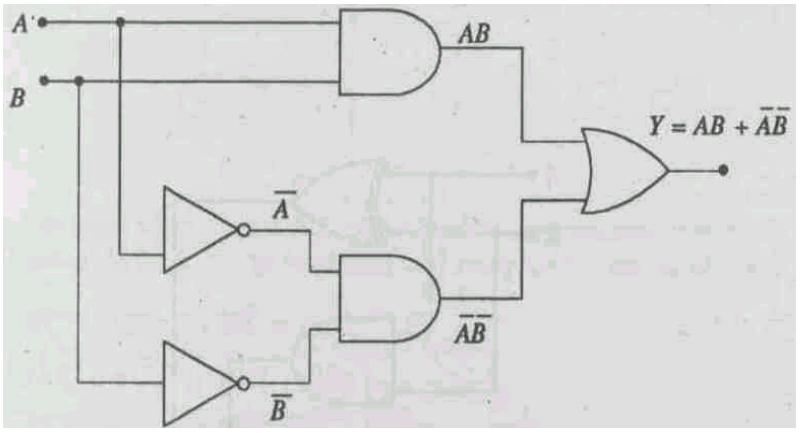


Fig. 2.32 Exclusive-NOR (XNOR) gate

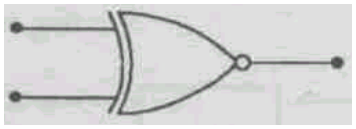


Fig. 2.33: Symbol of XNOR gate

11.3.7 Addition of Two One Bit Binary Numbers (Half Adder)

Recall the binary addition learnt in Unit 10. The binary addition of two single bit binary numbers is as follows.

$$\begin{array}{rrrr} 0 & 0 & 1 & 1 \\ +0 & +1 & +0 & +1 \\ \hline 00 & 01 & 01 & 10 \\ \hline \end{array}$$

In this example of addition, the bit on the right hand side is sum while the bit on the left hand side is carry. This can be put in a truth table as shown in Table 2.15.

Table 2.15: Truth table for half adder

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

This application has two outputs, one for 'sum' and another for 'carry'. Therefore, we have to obtain two Boolean expressions for the two outputs.

The expression for carry is

Carry = AB

that is, it is the output of an AND gate.

The expression for sum is

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

that is, it is the output of an XOR gate described in the previous section. These two circuits are connected together as shown in Fig. 2.34. This circuit is known as half adder and its symbol is given in Fig. 2.35.

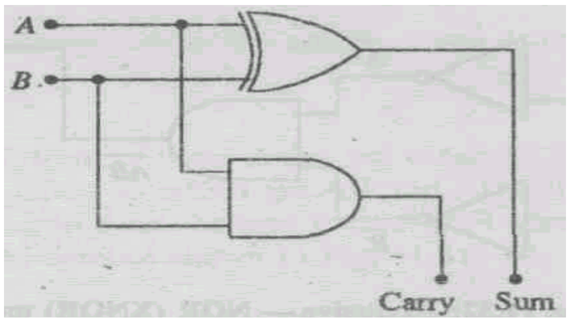


Fig. 2.34 Half adder circuit

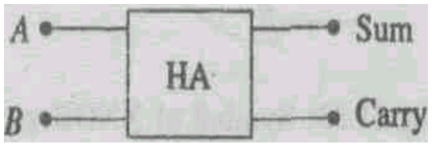
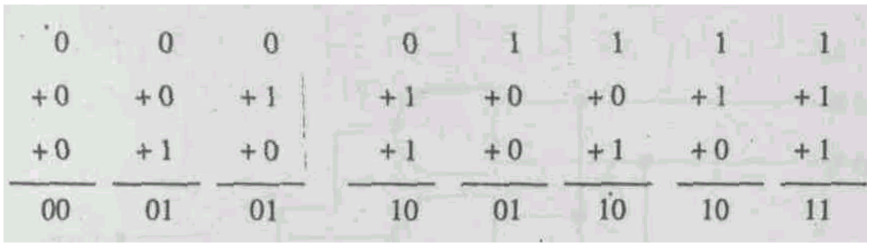


Fig. 2.35 Symbol of half adder

Recall the contradiction pointed out while describing addition by an OR gate. While it could justify addition in the case of its first three entries of inputs, it could not give correct result of addition of binary numbers in its last entry of inputs, i.e. it gave 1 + 1 = 1 (Boolean addition) rather than 1 + 1 = 10 (binary addition). This contradiction is now taken care of by the design of half adder. We can now say that the binary addition should be done using half adder or circuits described later in the Unit. But as far as Boolean postulates, including those based on OR gate, are concerned, they are helpful in designing circuits for binary arithmetic.

2.3.8 Addition of Three One Bit Binary Numbers (Full Adder)

The full adder can add three single-bit binary numbers. The binary addition three single-bit binary numbers is as follows:



0	0	0	0	1	1	1	1
+0	+0	+1	+1	+0	+0	+1	+1
+0	+1	+0	+1	+0	+1	+0	+1
00	01	01	10	01	10	10	11

The right hand bits of these additions represent the sum and the left hand bits represent the carry. These eight possible combinations of three single-bit binary numbers can be presented in the form of a truth table given in Table 2.16.

Table 2.16: Truth table for full adder

A	B	C	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

In order to design the logic circuit for a full adder, Boolean expressions have to be written and simplified in MSP form for both sum and carry, which are as follows:

Sum $= \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$ ABC + ABC + ABC + ABC

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

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

$= \overline{A}X + A\overline{X}$ where $X = B \oplus C$

$= A \oplus B \oplus C$

= MSP expression.

This is the output of a 3-input XOR gate.

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

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

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

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

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

$= BC + AC + AB\overline{C}$

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

$= BC + A(C + B)$

$= BC + AC + AB$

= MSP expression.

From these two MSP expressions, the logic circuit for a full adder can be obtained as described earlier. This circuit is given in Fig. 2.36 and its symbol is given in Fig. 2.37.

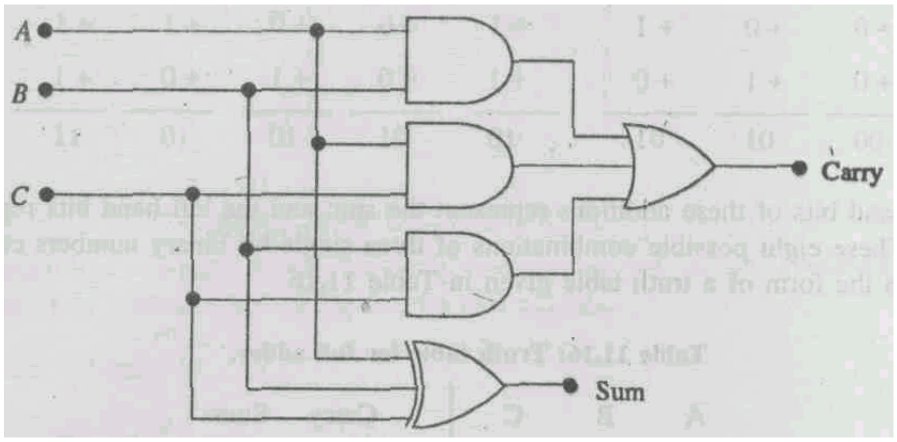


Fig. 2.36: Full adder circuit

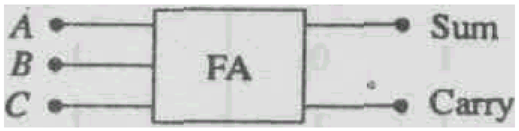


Fig. 2.37: Symbol of full adder

You would recall that a computer or a digital circuit can add only two binary numbers at a time. If a digital circuit has to add more than two binary numbers, as would mostly be the case, the circuit will add first two binary numbers and to the sum of these two numbers it will add the third binary number, and so on. But while adding two bits a carry is likely to appear as shown above. Therefore if the two binary numbers to be added are having more than one bit, then after the addition of first bits of the numbers the addition of second bits will also require the addition of any carry which appears from the addition of first bits. Thus the addition of first bits can be carried out by the half adder which has two inputs, but the addition of second bits require a 3-input adder which is realised by the full adder. There are eight entries to the truth table of a full adder, half of which are satisfied by the truth table of half adder ignoring carry bit (because the addition of first bits of two numbers do not have a carry to be added). For this reason, the adder described in the previous section is called the half adder and the one described in this section is called the full adder.

Example 2.8

Addition of two 4-bit binary numbers. Let us say the numbers are $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$.This addition requires one half adder to add A_0 and B_0 and three full adders to add the rest of the bits as shown in the circuit of Fig. 2.30. The outputs of the half adder are sum (S_0) and carry. The carry output of the half adder is given as the third input to the first full adder which has a carry output and a sum (S_1) output. The carry output of the first full adder is given to the second full adder, and so on. Thus for addition of two 4-bit binary numbers, we require one half adder and three full adders. For each additional bit in the numbers to be added, we require one more full adder.

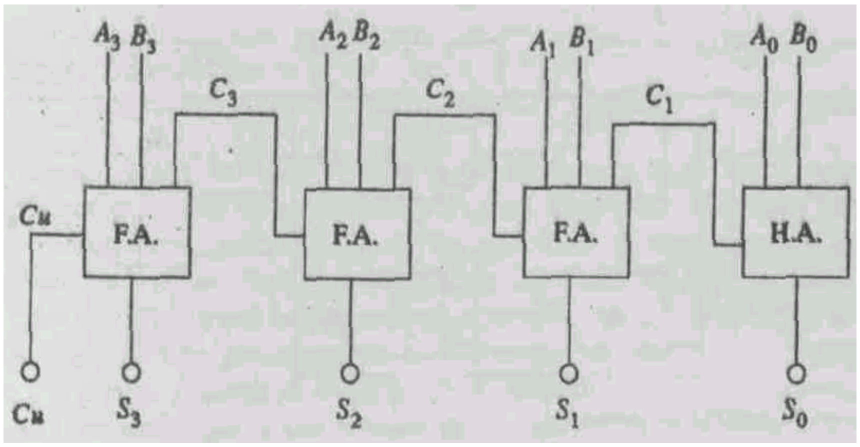


Fig. 2.38: A 4-bit binary adder

SAQ 8
Draw a digital circuit for a 2-bit binary adder.

2.3.9 Designing Circuits Using NAND Gates Only

Quite often it is required that only NAND gates should be used in designing digital circuits. The NAND gate being universal can be used to realise AND, OR and NOT gates. Therefore, wherever these gates are appearing, the equivalent NAND circuit is used. The realisation of AND, OR and NOT gates from NAND gates is shown in Fig. 2.39.

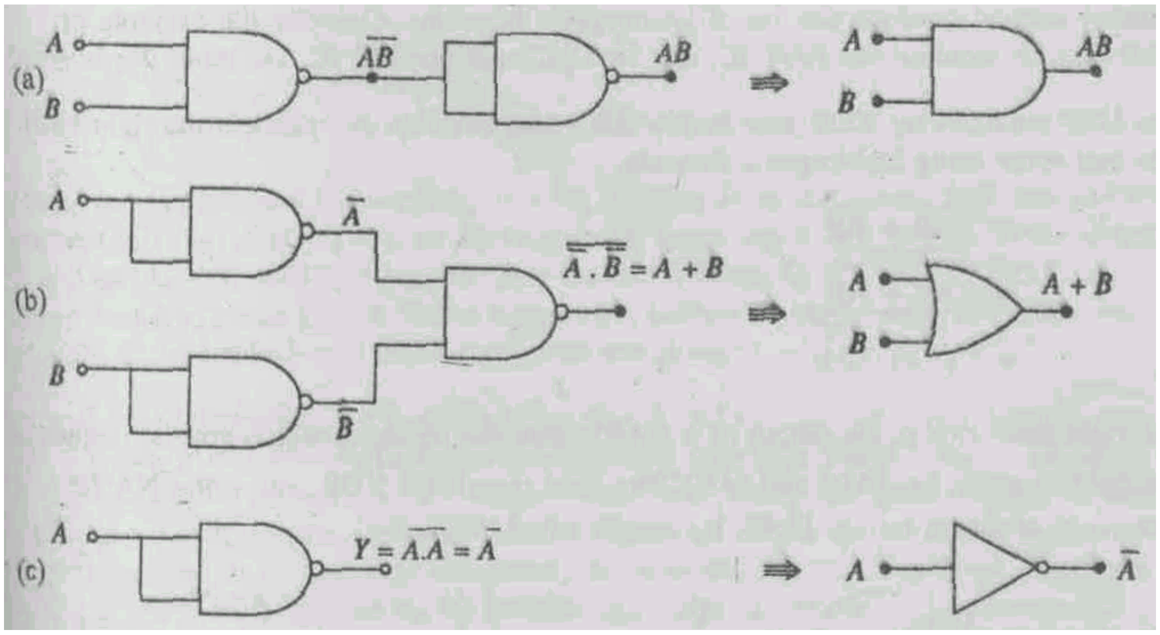


Fig. 2.39: Realisation of (a) AND, (b) OR, and (c) NOT gates using NAND gates

Example 2.9
Design a circuit for Y = AB + CD using NAND gates only.

The circuit for Y = AB + CD using AND and OR gates is shown in Fig. 2.40.

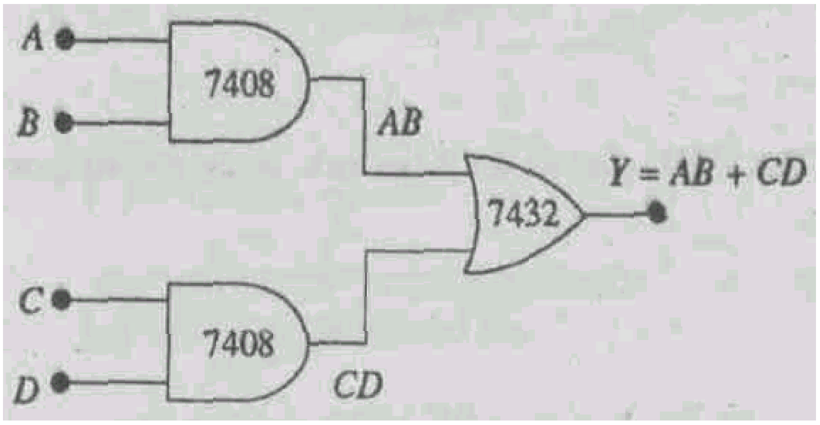


Fig. 2.40 Digital circuit for Y = AB + CD

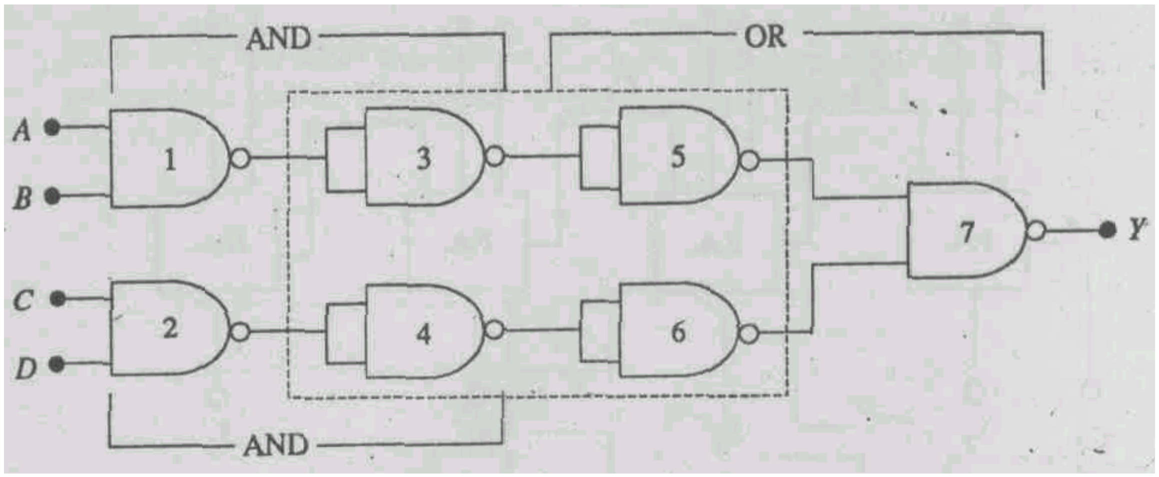


Fig. 2.41 AND and OR gates in the circuit given in Fig. 2.40 replaced by their equivalents

The AND gates and OR gate in Fig. 2.40 are replaced by equivalent NAND gate circuits from Fig. 2.39 as shown in Fig. 2.41. It requires two NAND ICs, Since the input and output of a combination shown as dotted of a NOT gate followed by another NOT gate are same, therefore such a combination is useless and it is hence eliminated. The final circuit after such elimination is shown in Fig. 2.42.

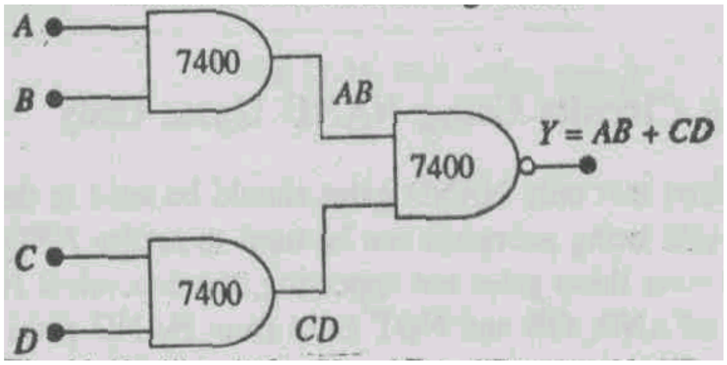


Fig. 2.42 Circuit for Y = AB + CD using NAND gates

Another method involves the use of De Morgan's theorems. Consider the example of XOR gate. It requires one NOT IC, one AND IC and one OR IC, i.e. three ICs in total.

The MSP equation for XOR gate is $Y = \overline{A}B + A\overline{B}$. Double complement the right hand side and solve using De Morgan's theorem.

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

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

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

The right hand side is the output of a NAND gate the inputs to which are the outputs of two NAND gates, i.e. $\overline{(\overline{A}B)} + \overline{(A\overline{B})}$. The final circuit for XOR gate using NAND gates only is shown in Fig. 2.43. It requires two NAND ICs.

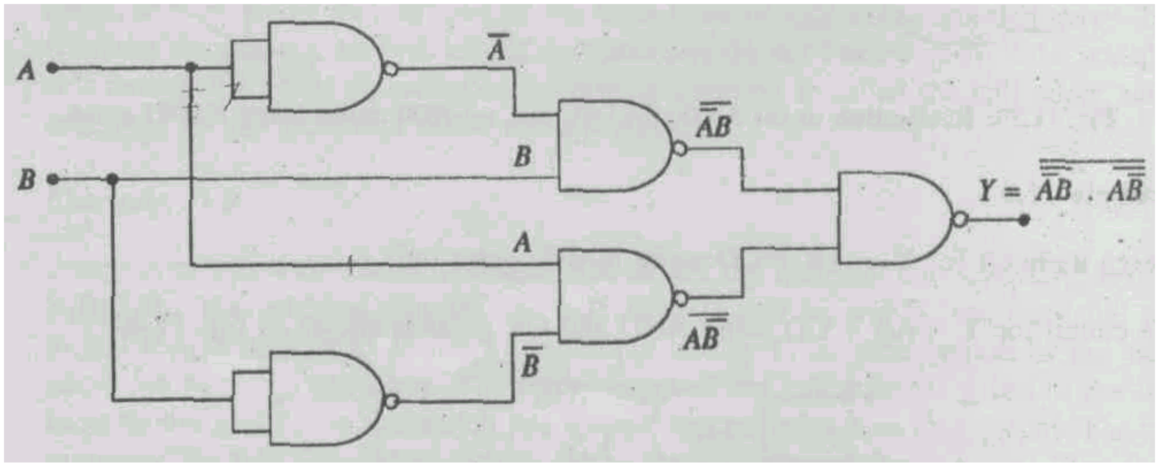

A
B
$\overline{A}$
$\overline{\overline{A}B}$
B
$Y = \overline{\overline{A}B} \cdot \overline{A\overline{B}}$
A
$\overline{A\overline{B}}$
$\overline{B}$

Fig. 2.43 Circuit for XOR gate using NAND gates only

SAQ9
Design *a* digital circuit for Y = A + BC using NAND gates only.

2.4 FLIPFLOPS

We have learnt combinational logic circuits in the previous section. The combinational logic circuits operate strictly in accordance with their truth table. However, there are logic circuits which have feedback path and the operation of which is not strictly defined by their truth tables. Such circuits operate differently for a given input condition depending upon the prior input sequence applied to the circuit. Such circuits are known as sequential logic circuits. These circuits have memory element also. In addition to the logic gates, a computer requires memory element. The simplest memory element is a flip-flop. It has two stable states and remains in any one of these two stable states until triggered into the other state. Quite often the flip-flop is also known as a latch.

2.4.1 RS Flip-flop

The most basic flip-flop circuit is constructed using two NAND gates or two NOR gates. In NAND gate flip-flop, two NAND gates are cross-coupled as shown in Fig. 2.44. It has two latched outputs Q and $\overline{Q}$. It has two inputs: SET (S)' and RESET (R) or CLEAR (C). The input names signify their actions as well. For the input names such a flip-flop is known as RS flip-flop.

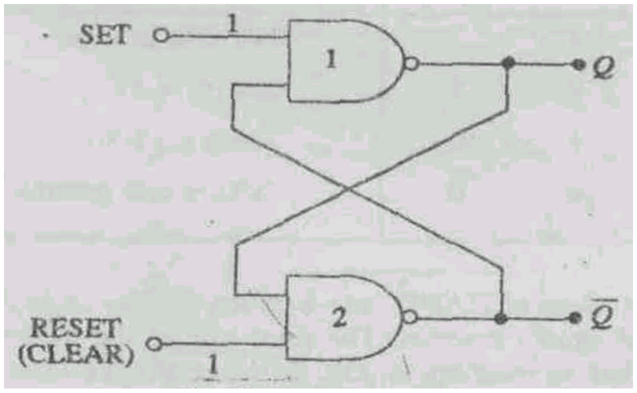


Fig. 2.44 RS flip-flop

Let us now understand the working of a RS flip-flop. Both the inputs, SET and RESET, are kept HIGH, i.e. at logic 1. In the beginning, let us say S = R = 1. With the outputs Q = 0 and $\overline{Q}$ = 1, NAND-1 has the inputs 1 and 1 hence Q = 0, and NAND-2 has inputs 1 and 0, hence $\overline{Q}$ = 1. These outputs are latched or stuck with each other and continue to be latched until input conditions are changed.

Second possibility with S = R = 1 is when Q= 1 and $\overline{Q}$ = 0. The NAND-1 will have 1 and 0 inputs giving Q= 1. Likewise the NAND-2 will have 1 and 1 inputs giving $\overline{Q}$ = 0. Once again the two outputs are latched together and they will continue to be latched until input conditions are changed. S and R both high means the two sets of possible outputs remains in its last state indefinitely because of the internal latching action. Thus, a high S and a high R gives us the inactive state; the circuit stores or remembers. When we want to change the flip-flop output one of the inputs will be pulsed LOW (i.e. logic 0).

Setting the Flip-flop

Let us say that the SET is momentarily pulsed LOW (i.e. S = 0 for a moment) while RESET continues to be 1. Now if Q= 0 and $\overline{Q}$ = 1 prior to the occurrence of a LOW pulse at SET, Q goes 1 which in turn forces $\overline{Q}$ to a 0. Thus when SET returns to 1, the NAND- 1 output remains HIGH which in turn keeps the NAND-2 output at 0.

If prior to, the application of SET pulse, Q= 1 and $\overline{Q}$ = 0, then a LOW pulse at SET will not change anything because $\overline{Q}$ = 0 is already keeping the NAND-1 output to 1. Thus, when SET returns to 1, the outputs are still Q= 1 and $\overline{Q}$ = 0.

Thus a LOW on the SET input will always cause the flip-flop to end up in Q = 1 state. Hence, this operation is called setting the flip-flop, and Q = 1 state is known as SET state.

Resetting or Clearing the Flip-flop

The SET is kept at 1 and RESET is momentarily pulsed LOW (i.e. 0). Let us say that prior to the pulse, $Q = 0$ and $\overline{Q} = 1$. Since $Q = 0$ is already keeping the NAND-2 output at 1, therefore the application of a LOW pulse at RESET will not change the situation. However, if prior to the application of a LOW pulse, $Q = 1$ and $\overline{Q} = 0$, then a LOW pulse at RESET will give NAND-2 output as 1, which in turn forces the NAND-1 output to a 0. Thus a LOW at RESET always ends up in $Q = 0$. This operation is called clearing or resetting operation. And the $Q = 0$ state is known as CLEAR (or RESET) state.

When SET and CLEAR are simultaneously pulsed LOW, it produces 1 at both the outputs. There is a race to come to a 1 state. This is an undesired state-because Q and $\overline{Q}$ are inverse of each other. When R and S return to 1, the race among the two will give unpredictable results. Therefore, R = S = 0 is not used. However, as described above, R = S = 1 produces no change in the outputs. The entire operation of the RS flip-flop is summarised in the truth table given in table 2.17.

Table 2.17: Truth table for RS flip-flop

S	R	Output
1	1	NC (No change)
0	1	Set (Q = 1)
1	0	Reset (Q = 0)
0	0	*(Race and invalid)

The De Morgan equivalent of NAND gate is given in Fig. 2.45. Fig. 2.45 (a) represents the left side of De Morgan's theorem. The right side of the theorem implies that the inputs are inverted before reaching an OR gate (see Fig. 2.45b). This combination is used so often that the abbreviated symbol shown in Fig. 2.45c has come into use. This symbol is called a bubbled OR gate. Fig. 2.45d is a graphic summary of De Morgan's theorem which shows that a NAND gate and a bubbled OR gate are equivalent. Therefore, we can replace one with the other whenever desired.

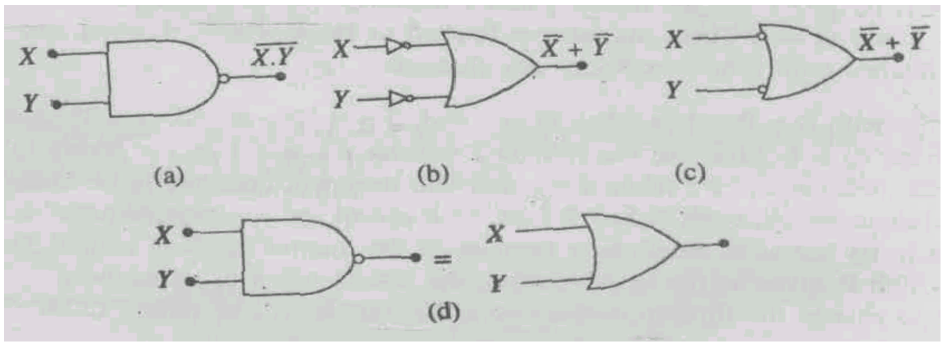


Fig. 2.45: De Morgan equivalent of NAND gate

Using De Morgan equivalent of a NAND gate, the NAND gate flip-flop can also be represented by the circuit shown in Fig. 2.46. The symbol of this flip-flop is shown in Fig. 2.47. The bubble at the S and R inputs indicate that the flip-flop can be set or reset by giving a LOW pulse.

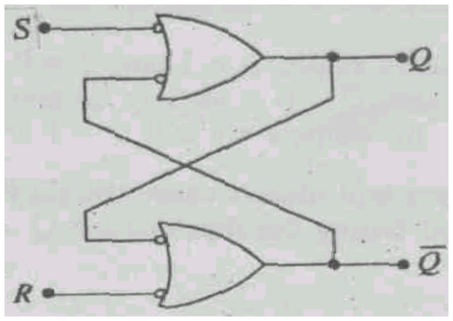


Fig. 2.46 De Morgan equivalent of NAND gate RS flip-flop

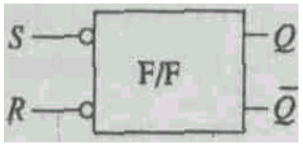


Fig. 2.47 Symbol of RS flip-flop

Example 2.10
If the train of pulses given to the S and R inputs of RS flip-flop are as shown in Fig. 2.48(a) and (b) respectively, then trace its Q output. Initial value of Q is given to be 0.

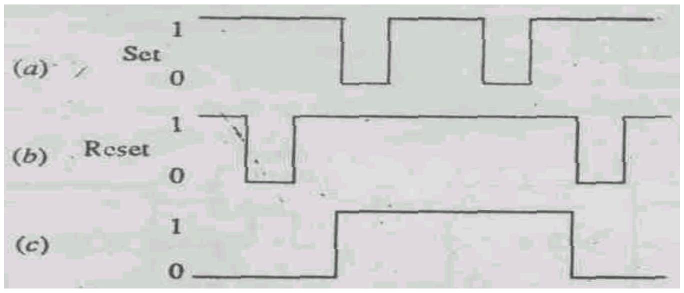


Fig. 2.48 Set/Reset pulses and the output

Solution
Using Table 2.17, the Q output of the RS flip-flop is as shown in Fig. 2.48(c).

SAQ 10
What is the shape of the Q output of RS flip-flop if the S and R inputs are as shown in Fig. 2.49? Initial value of Q is given to be 1.

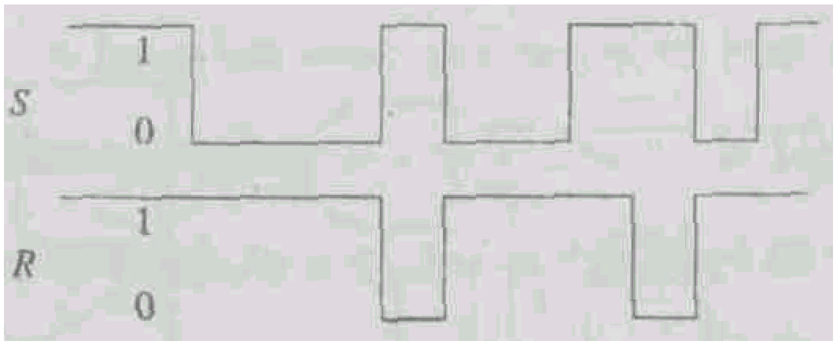


Fig. 2.49

2.4.2 Clocked RS Flip-flop

Computers use thousands of flip-flops. To coordinate the overall action, a square wave signal called the **clock** is sent to each flip-flop. The clock is applied to all flip-flops simultaneously; this ensures that they all change states in unison. This synchronization is essential in many digital systems.

In most of the synchronous systems the output can change only when the clock signal is making a transition from 0 to 1, i.e. positive going transition (PGT) or 1 to 0, i.e. negative going transition (NGT). These systems are known as edge triggered. The PGT and NGT are shown in Fig. 2.50. The symbols of edge triggered RS flip-flop which work with PGT and NGT are shown symbolically in Figs. 2.51(a) and (b) respectively,

Note the difference in symbol of clock activated by a PGT and NOT. The change in the control inputs R and S to the flip-flop will not effect a change in the Q output until an active clock (CLK) transition, i.e. a PGT in case of Fig. 2.51 (a) and a NOT in case of Fig. 2.51(b), occurs. The control inputs keep the flip-flop ready to change and the active clock transition at the CLK input actually triggers the change. To ensure that a clocked flip-flop responds properly when the active clock transition occurs, the inputs must he stable, i.e., unchanging.

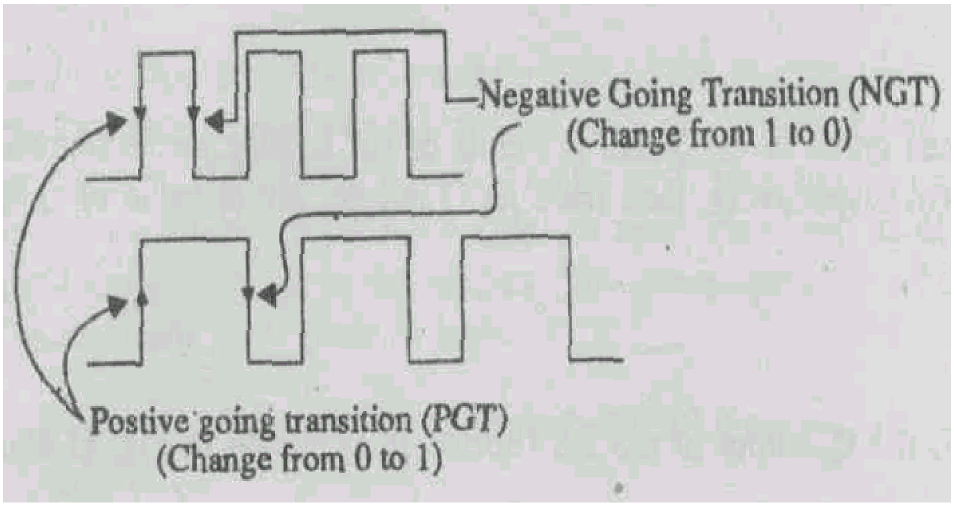

Negative Going Transition (NGT)
(Change from 1 to 0)
Postive going transition (PGT)
(Change from 0 to 1)

Fig. 2.50 Positive and negative going transitions

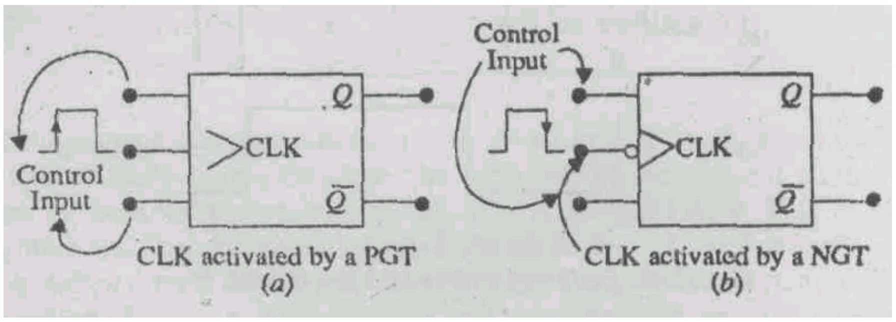


Fig. 2.51 Symbol of edge triggered flip-flop activated by a (a) PGT, and (b) NGT

Consider the circuit given in Fig. 2.52 in which two additional NAND gates are used as the clock pulse steering circuit and is triggered by a PGT. A LOW (i.e. 0) clock CLK prevents S and R from controlling the flip-flop, because with whatever values of S and R the outputs of the NAND-1 and NAND-2 will be 1 which will not produce any change in the Q output of the flip-flop. However, when the CLK is HIGH (i.e. 1) and S = R = 0, the outputs of the two NAND gates will be 1 and there would be no change in the Q output.

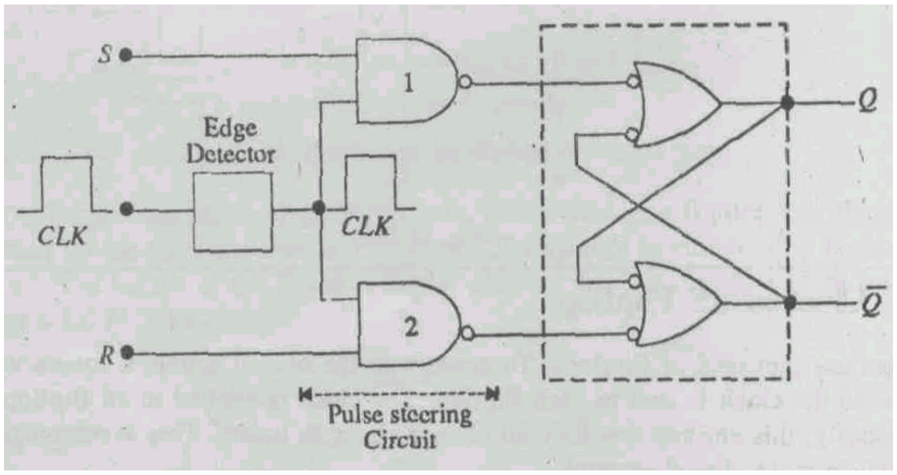


Fig. 2.52 Circuit of edge triggered RS flip-flop

Table 2.18 shows the truth table for a positive edge triggered RS flip-flop. The $Q = Q_0$ is the output level before the arrival of the PGT of the CLK. The arrow directed upward $(\uparrow)$ indicates that a PGT is required at the CLK.

Table 2.18: Truth table for a positive edge triggered RS flip-flop

Inputs			Output
R	S	CLK	Q
0	0	$\uparrow$	Q_0 (No change
0	1	$\uparrow$	1
1	0	$\uparrow$	0
1	1	$\uparrow$	*Race

The inputs S and R, and corresponding Q output, assuming the initial value of Q, i.e., Q_0, equal to 0, are as shown in Fig. 2.53. It is clear that at the arrival of the first clock transition both R and S are 0, therefore there is no change in the Q output, which continues to be 0. But at the arrival of the second clock transition S is 1 and R is 0, this sets the flip-flop with Q = 1 which does not change till the third clock transition. At the time of the third clock transition R is 1 and S = 0 which resets the flip-flop with Q= 0. This is how the Q output is traced. Note that between two PGTs of the CLK, the Q output does not change. It must be remembered, that whenever tracing a Q output corresponding to the inputs, you have to look for the active clock, note the values of inputs nand then decide the value of the Q output.

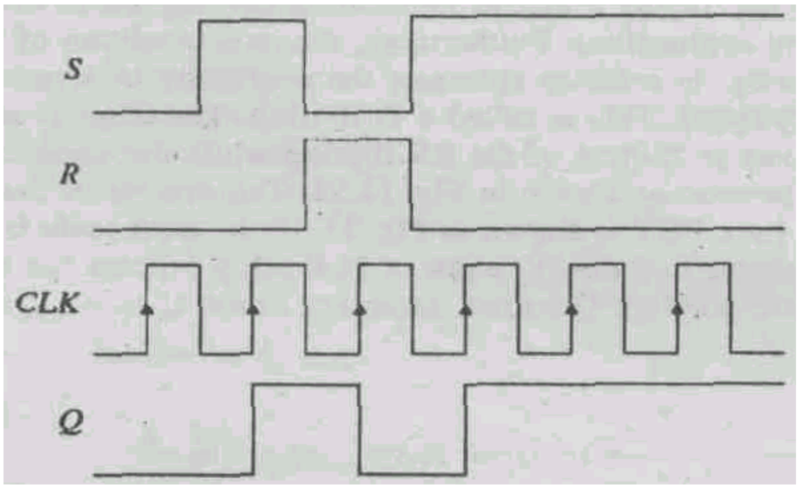


Fig. 2.53: Inputs and output of a clocked RS flip-flop

The truth table of an RS flip-flop triggered by a NGT is shown in Table 2.19.

Table 2.19: Truth table for a negative edge triggered RS flip-flop

Inputs			Output
R	S	CLK	Q
0	0	↓	Q_0 (No change)
0	1	↓	1
1	0	↓	0
1	1	↓	*Race

The PGT or NOT can be obtained by using a combination of gates or a differentiating circuit consisting of a capacitor and a resistor.

SAQ 11
If the train of pulses to S and R inputs of a clocked RS flip-flop are as shown in Fig. 2.54, and if the initial value of Q is 0, trace its Q output.

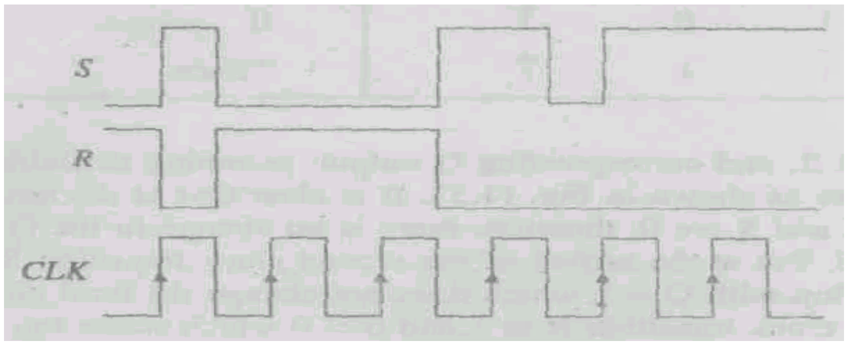


Fig. 2.54

2.4.3 Clocked D Flip-flop

The RS flip-flop has two inputs S and R. Generating two signals to drive a flip-flop is a disadvantage in many applications. Furthermore, the race condition of both S and R low may occur inadvertently. In order to eliminate the possibility of a race condition a new kind of flip-flop is designed. This is called a D flip-flop. The letter D stands for the data. The data input is given to S-input of the RS flip-flop while the same input goes to its R-input through an inverter as shown in Fig. 2.55. This symbol of the edge triggered D flip-flop activated by a PGT is shown in Fig. 2.56. Its truth table is given in Table 2.20 which shows that the Q output of D flip-flop follows the input data D. The D input and corresponding Q output, assuming initial Q to be 1, are shown in Fig. 2.57.

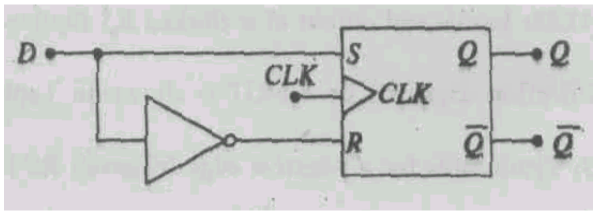


Fig. 2.55 Circuit for D flip-flop

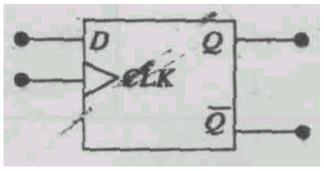


Fig. 2.56: Symbol of D flip-flop

Table 11.20: Truth table for a positive edge triggered D flip-flop

D	CLK	Q
0	↑	0
1	↑	1

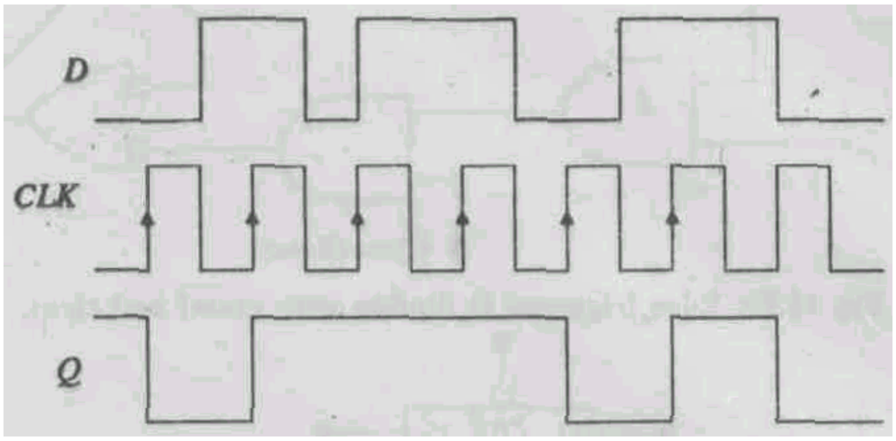


Fig. 2.57 Input and output of a D flip-flop

D Latch
Sometimes edge trigger detecting circuit (like RC combination) for D flip-flop is not used. In this case the D flip-flop functions slightly differently and is known as a D latch. Instead of edge triggering, level clock or an ENABLE (abbreviated as EN) signal is used as shown in Fig. 2.58. When EN/CLK is 1, D will produce a 0 at either SET or CLEAR inputs of the NAND latch to give a Q output to be at the same level of D.

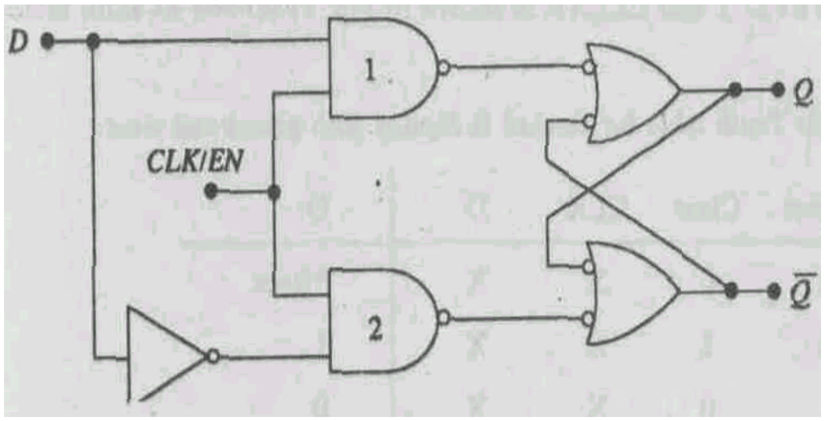


Fig. 2.58 Circuit for D latch

When EN/CLK is 1, if D changes, Q will follow changes exactly like D as the Q output does not have to wait for the clock transition to respond to changes in D. The D latch is thus 'transparent' to the input in this mode. When EN/CLK is at 0, D is inhibited from affecting NAND latch because the outputs of both steering NAND gates will be 1. Thus Q and $\overline{Q}$ continue to stay wherever they were before EN/CLK became 0. In other words, the outputs are latched to their current level and cannot change during the period EN/CLK is 0, even if D changes. The truth table of D latch is given in Table 2.21.

Table 2.21: Truth table for D latch

D	EN/CLK	Q
X	0	NC
0	1	0
1	1	1

Quite often two AND gates are introduced between the pulse steering circuit and the NAND latch as shown in Fig. 2.59. One input each of these AND gates are known as RESET (direct SET) and CLEAR (direct RESET) and are kept at 1 so as to allow the output of the pulse steering circuit to pass through. However, if we want to set the flip-flop irrespective of the value of the D input, then we give a 0 to PRESET, which will set the flip-flop. Similarly, by giving a 0 to CLEAR will directly reset the flip-flop. The symbol for D flip-flop with PRESET and CLEAR is shown in Fig. 2.60 and its truth table is given in Table 2.22.

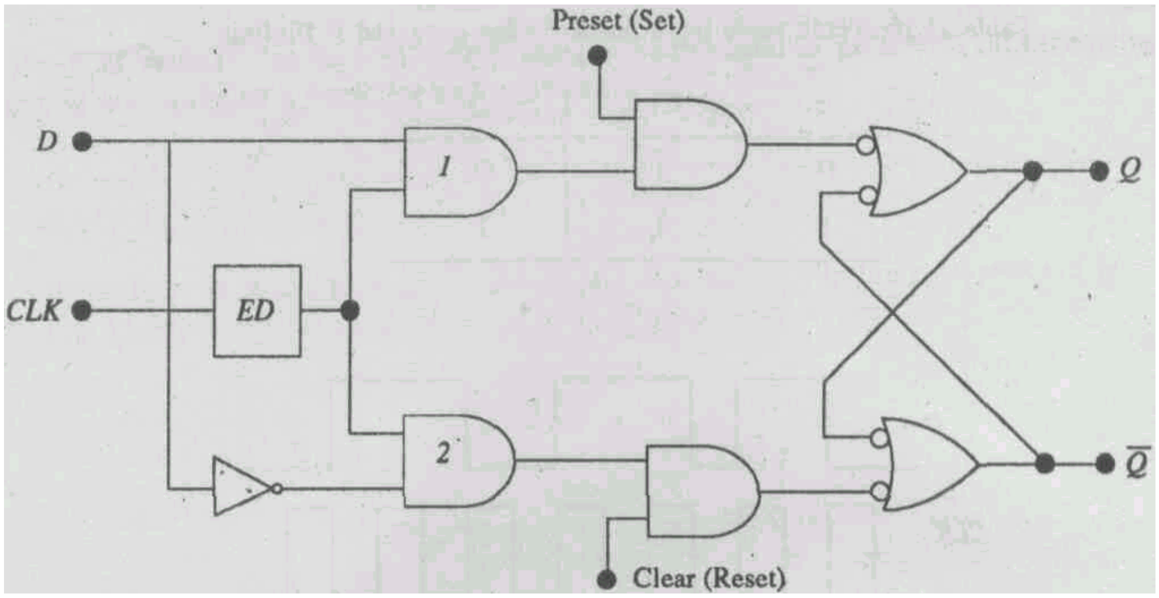


Fig. 2.59 Edge triggered D flip-flop with preset and clear

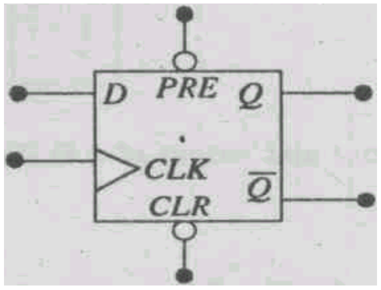


Fig. 2.60 Symbol of edge triggered D flip-flop with preset and clear

Table 2.22: Truth table for clocked D flip-flop with preset and clear

Preset	Clear	CLK	D	Q
0	0	X	X	*Race
0	1	X	X	1
1	0	X	X	0
1	1	0	X	NC
1	1	1	X	NC
1	1	↑	X	NC
1	1	↑	0	0
1	1	↑	1	1

SAQ12
The D input to a positive edge triggered D flip-flop is as shown in Fig. 2.61. Trace the Q output.

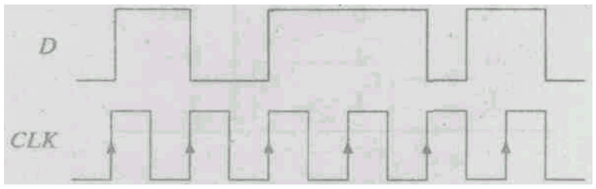


Fig. 2.61

2.4.4 Clocked JK Flip-flop

In the next unit, we show you how to build a counter, a circuit that counts the number of positive or negative clock edges driving its clock input. When it comes to circuits that count, JK flip-flop is the ideal element to use. Therefore before ending this unit we will study about JK flip-flop. The circuit for an edge triggered JK flip-flop is shown in Fig. 2.62 and its symbol is shown in Fig. 11.63. The working of JK flip-flop is same as that of RS flip-flop

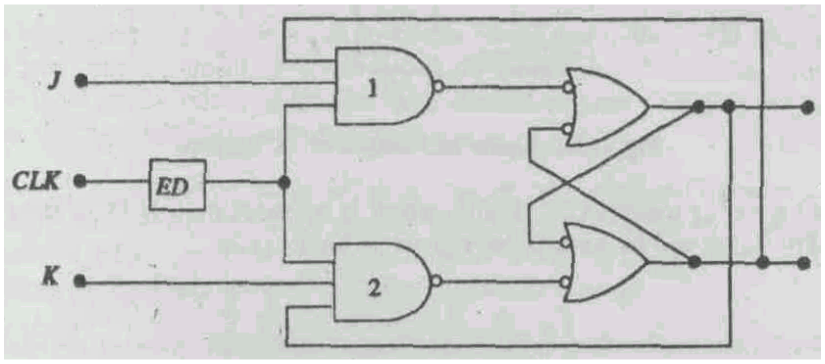


Fig. 2.62 Circuit for edge triggered JK flip-flop

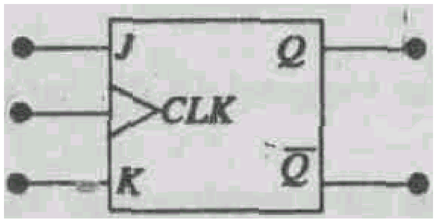


Fig. 2.63 Symbol of edge triggered JK flip-flop

except that race condition is absent. That is, there is no ambiguous result. The outputs Q and $\overline{Q}$ of the NAND latch are fed back to NAND-2 and NAND-1 respectively of the pulse steering circuit which gives toggle operation. With J = K = 1, assume that Q is 0 when clock transition arrives. With Q = 0 and $\overline{Q}$ = 1, NAND-1 will steer PGT to set the NAND latch to give Q = 1. If we assume Q = 1 when PGT of the clock appears, NAND-2 will steer PGT to clear the NAND latch to produce Q = 0. Thus Q always ends up in opposite state. This is known as the toggle mode of operation. If both J and K are left to a state of 1, the flip-flop will change state for each clock transition. The Q output equal to Q_0 means that the new value of Q will be inverse of the value it had prior to the PGT. The truth table of this flip-flop is given in Table 2.23. Fig. 2.64 shows J and K inputs and the corresponding Q output.

Table 2.23: Truth table for a positive edge triggered JK flip-flop

J	K	CLK	Q
0	0	↑	Q_0 (No change)
1	0	↑	1
0	1	↑	0
1	I	↑	Q_0 (Toggle)

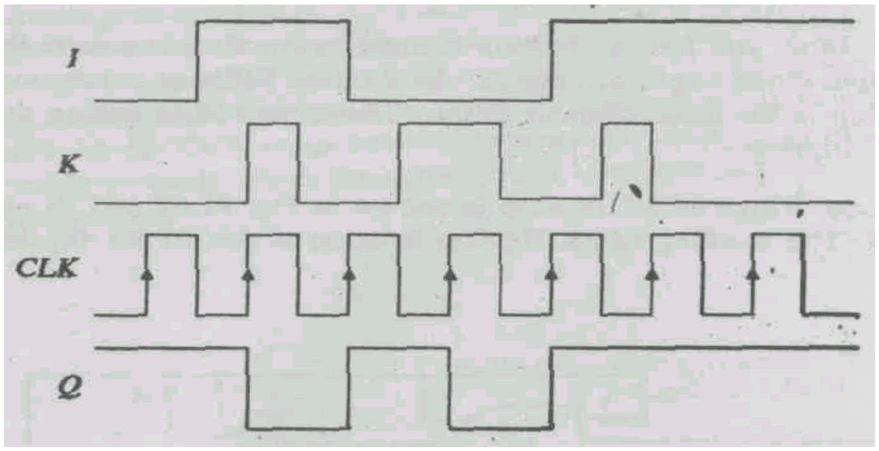


Fig. 2.64 Inputs and output of JK flip-flop

The symbol for edge triggered JK flip-flop which is activated by an NGT of the clock is shown in Fig. 2.65 and its truth table is given in Table 2.24.

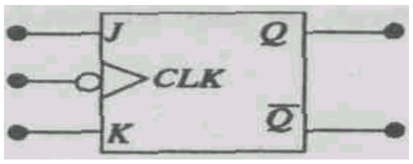


Fig. 2.65 Symbol of edge triggered JK flip-flop activated by an NGT

Table 2.24: Truth table for a negative edge triggered JK flip-flop

J	K	CLK	Q
0	0	↑	Q_0 (No change)
1	0	↑	1
0	I	↑	0
1	1	↑	Q_0 (Toggle)

SAQ 13

The J and K inputs to a JK flip-flop are as shown in Fig. 2.66. If the initial value of Q output is 0, trace the Q output.

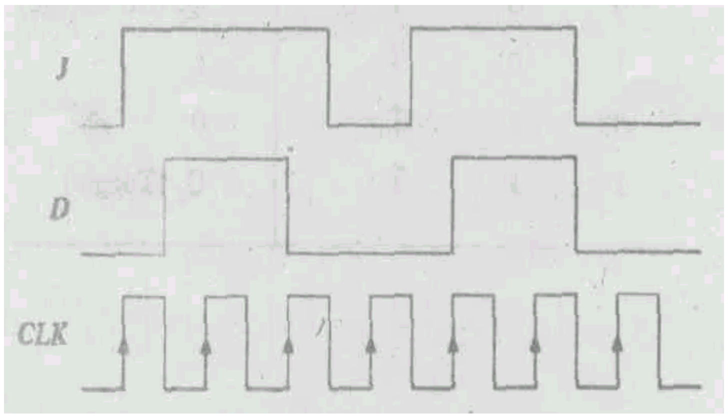


Fig. 2.66

2.5 SUMMARY

- There are three basic logic gates - AND, OR and NOT. The output of an AND gate is 1 only when all the inputs are 1. The output of an OR gate is 0 only when all the inputs are 0. The output of a NOT gate is complement of the input.
- The AND and NOT gates are combined to get the NAND gate and OR and NOT gates are combined to get NOR gate. The NAND and NOR gates are known as building blocks in digital circuitry because AND, OR and NOT gates can be obtained using NAND and NOR gates only.
- All logic gates and circuits work in binary mode, that is the inputs and outputs can have values either 1 or 0. Therefore, Boolean algebra is used to describe their input-output relationships, the basic Boolean rules or theorems are obtained from the truth tables of the three basic gates.
- A digital circuit can be expressed as a Boolean expression and likewise a logic circuit can be obtained from a Boolean expression. A Boolean expression can be simplified, which gives us a simplified digital circuit. In all applications, first a Boolean expression is simplified to give a simpler circuit.
- A Boolean expression can also be obtained from a truth table. And a truth table can be obtained from a Boolean expression without reference to its logic circuit. The Boolean expression is written in the Sum-of-the-Product (SOP) form, which is simplified to get the Minimum-Sum-of-the-Product (MSP) form. The MSP expression is used to write the final digital circuit.
- Exclusive-OR and exclusive-NOR gates are obtained by the combinations of three basic gates. The output of the XOR gate is 0 if both the inputs are same and is 1 if both the inputs are different. The output of the XNOR gate is 1 if both the inputs are same and is 0 if both the inputs are different.
- A half adder adds two bits binary numbers while a full adder adds three bits. The half and full adders are combined to add two multi-bit binary numbers.
- The combinational logic circuits do no have memory, that is, the outputs of such circuits do not depend on the previous occurrence of an event. The input-output relationship of the circuit is precisely defined by its truth table.

- The RS flip-flop is the basic element which has memory, that is, its output depends on the previous occurrences of an event. The input of a RS flip-flop can also be triggered by a clock by using a pulse steering circuit. The other flip-flops are D and JK flip-flops. The output of the D flip-flop follows the input. The race condition of RS flip-flop is avoided in JK flip-flops.
- The RS, D and JK flip-flop can be triggered by a positive going transition (PGT) or a negative going transition (NGT). These flip-flops are used as memory devices.

2.6 TERMINAL QUESTIONS

1. Simplify the expression $Y = A\overline{B}D + A\overline{B}\overline{D}$
2. Simplify the expression $Y = BCD + A\overline{B}CD$ and find its MSP form.
3. Simplify the expression $Y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$.
4. Simplify the expression $Y = \overline{(A+BC)\cdot(D+FG)}$ Y = (A + BC) • (D + FG)
5. Write the Boolean expression for the truth table given in Table 2.25.

Table 2.25

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	l	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

6. Write Boolean expression for the truth table given in Table 2.26

Table 2.26

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

7. Write Boolean expression for the truth table given in Table 2.27

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	I
1	1	1	0

8. Write Boolean expression for the truth table given in table 2.28.

Table 2.8

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

9. Write the truth table for the expression obtained in question No. 2 above.

10. Write the truth table for the expression obtained in question No. 3 above.

11. Write the truth table for the expression obtained in question No. 5 above.

12. Write the truth table for the expression obtained in question No. 8 above.

13. Draw a digital circuit for a 5-bit binary adder.

14. Design a digital circuit for $Y = \overline{A}C + A\overline{D}$.

15. Design a digital circuit for expression of question No. 14 using NAND gates only.

2.7 SOLUTIONS AND ANSWERS

SAQs

1.

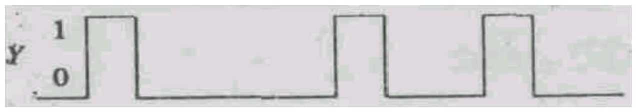


Fig. 2.67

2.

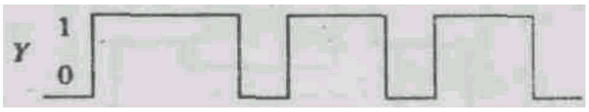


Fig. 2.68

3.

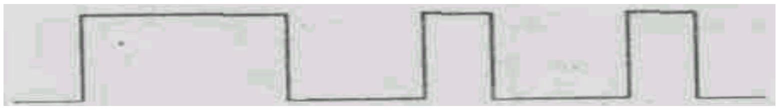


Fig. 2.69

4.

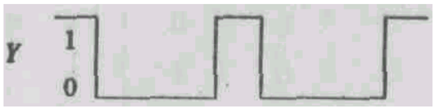


Fig. 2.70

5. $$\begin{aligned} Y &= A\overline{B}\,\overline{C} + AB\overline{C} + ABC \\ &= A\overline{B}\,\overline{C} + AB(\overline{C} + C) \\ &= A\overline{B}\,\overline{C} + AB \\ &= A(\overline{B}\,\overline{C} + B) \\ &= A(B + \overline{C}) \\ &= AB + A\overline{C} \end{aligned}$$

6. Using the reasoning method, Y = 1 when either or all of AB, BC, and CA are 1. Thus we get the truth table as follows:

Table 2.29: ABC

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

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

8.

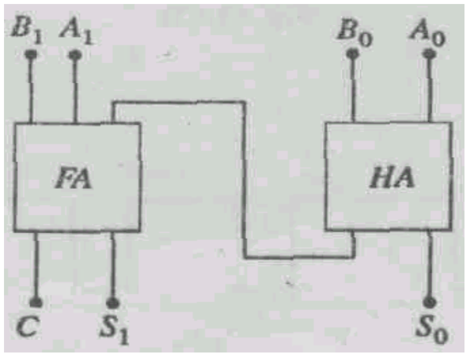


Fig. 2.71: A 2-bit binary adder

9. Digital circuit for Y = A + BC is as shown in Fig. 2.72.

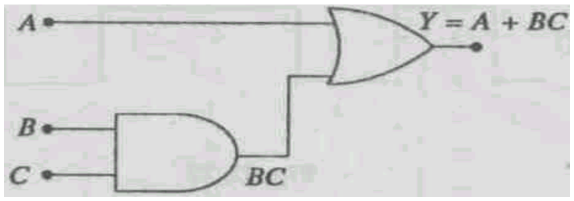


Fig. 2.72: Now replace OR and AND gates by their NAND equivalents as shown in Fig. 2.73

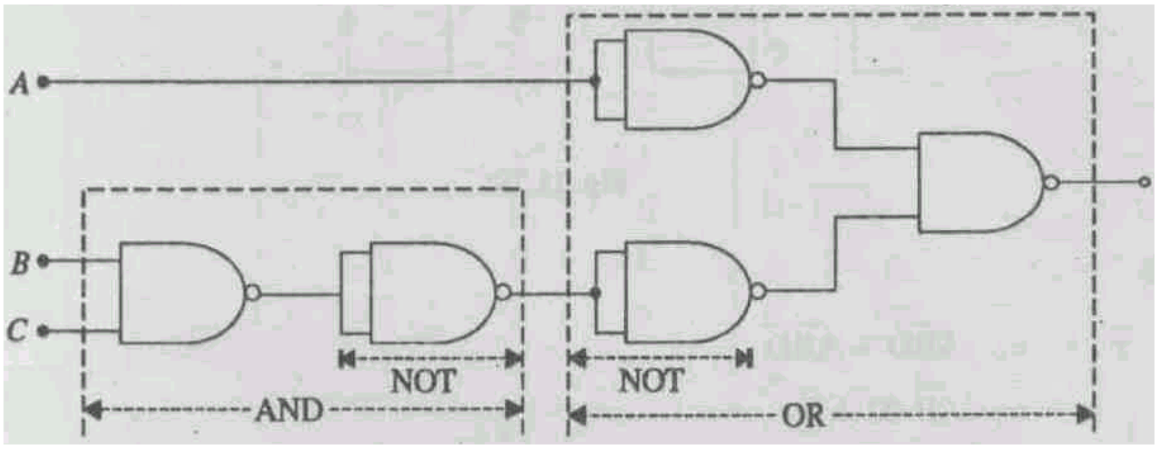


Fig. 2.73

Removing the combination of a NOT gate followed by a NOT gate, we get the circuit as shown in Fig. 2.74.

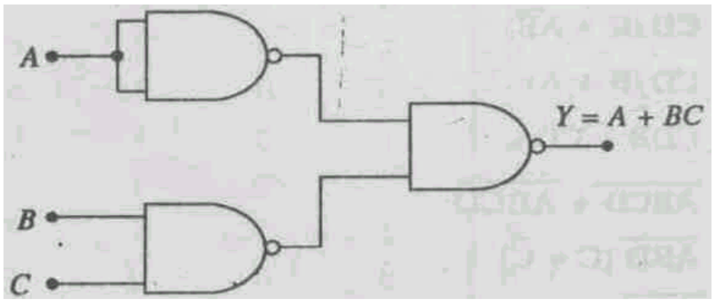


Fig. 2.74

Alternatively, simplify the expression using De Morgan's theorem as follows:

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

This equation gives the circuit already obtained in Fig. 2.74.

10.

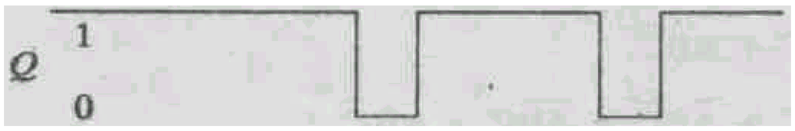


Fig. 2.75

11.

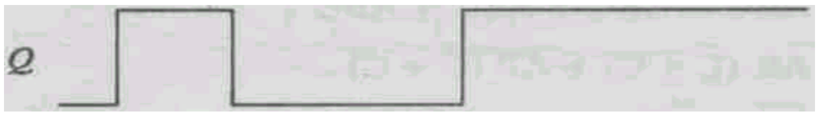


Fig. 2.76

12.

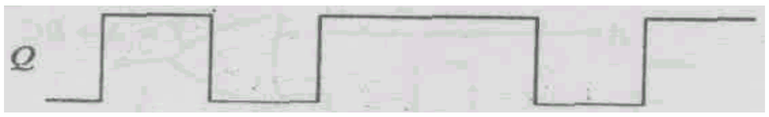


Fig. 2.77

13.

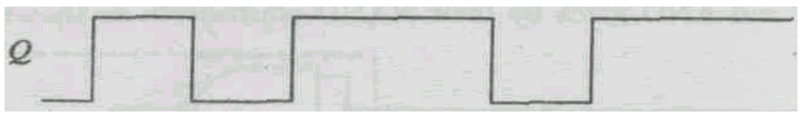


Fig. 2.78

TQs

1. $Y = A\overline{B}D + A\overline{B}\,\overline{D}$

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

2. Y
$$= BCD + A\overline{B}CD$$
$$= CD(B + A\overline{B})$$
$$= CD(B + A)$$
$$= CDB + CDA$$

3. Y
$$= \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$
$$= \overline{A}\overline{B}\,\overline{D}(C + \overline{C})$$
$$= \overline{A}\,\overline{B}\,\overline{D}$$

4. Y
$$= \overline{(A + BC)\cdot(D + FG)}\,\mathrm{Y}$$
$$= \overline{A + BC} + \overline{D + FG}$$
$$= \overline{A}\cdot\overline{BC} + \overline{D}\cdot\overline{FG}$$
$$= \overline{A}\cdot(\overline{B} + \overline{C}) + \overline{D}\cdot(\overline{F} + \overline{G})$$
$$= \overline{AB} + \overline{AC} + \overline{DF} + \overline{DG}$$

5. Y
$$= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}B\overline{C} + ABC$$
$$= \overline{A}\,\overline{C}(\overline{B} + B) + AC(\overline{B} + B)$$
$$= \overline{A}\,\overline{C} + AC$$

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

7. Y
$$= \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}C + AB\overline{C}$$
$$= \overline{A}(\overline{B}C + B\overline{C}) + A(\overline{B}C + B\overline{C})$$
$$= (\overline{A} + A)(\overline{B}C + B\overline{C})$$
$$= \overline{B}C + B\overline{C}$$

8. Y
$$= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC$$
$$= \overline{A}\,\overline{B}(\overline{C} + C) + \overline{A}B(\overline{C} + C)$$
$$= \overline{A}\,\overline{B} + \overline{A}B$$
$$= \overline{A}(\overline{B} + B)$$
$$= \overline{A}$$

9.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0

0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
I	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

10.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	I	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	I	0	0	0
1	1	0	I	1
1	1	1	0	0
1	1	1	1	1

11.

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	I
1	I	0	0
1	1	1	1

12.

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

13.

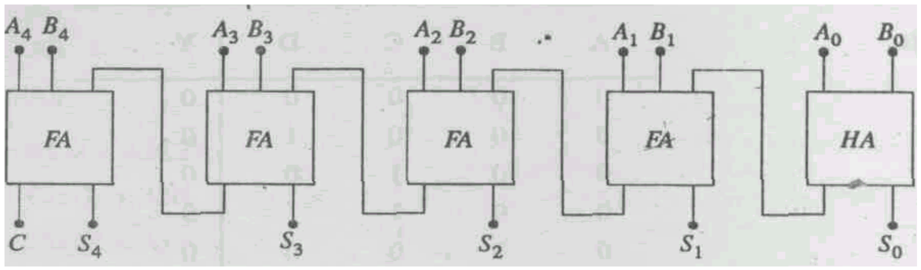


Fig. 2.79 A 5-bit binary adder

14.

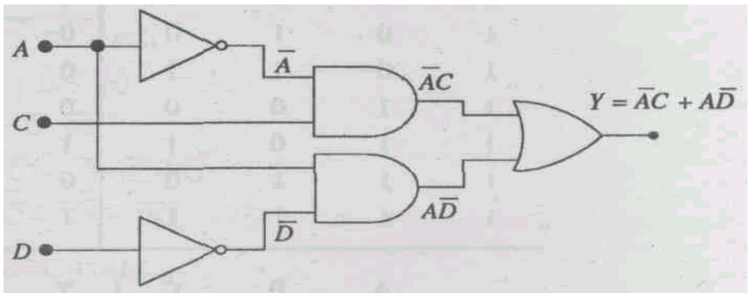


Fig. 2.80 Digital Circuit for $Y = \overline{A}C + A\overline{D}$

15.

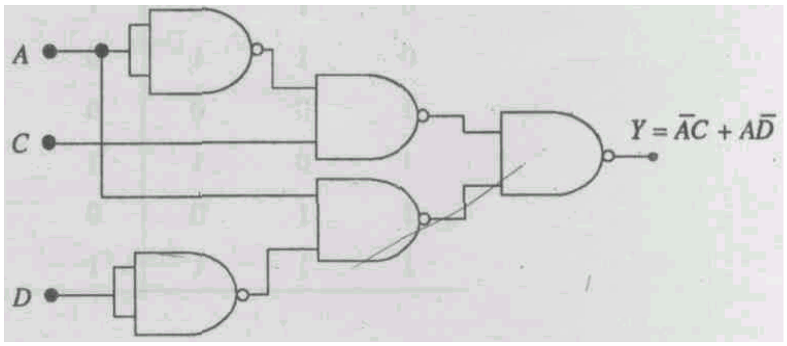


Fig. 2.81: Digital circuit for $Y = \overline{A}C + A\overline{D}$ using NAND gates only

UNIT 12 REGISTERS, COUNTERS, MEMORY CIRCUITS AND ALOGUE/DIGITAL **CONVERTERS**

Structure

3.1 INTRODUCTION

In Unit 2, you learnt about the combinational logic circuits and adders. The operation of all such circuits is faithfully defined by the respective truth table and their outputs do not depend upon the previous input or output conditions. Hence they do not have memory. However, since the output of flip-flops depend upon the previous input or output conditions or sequence of input or output, therefore these circuits, called sequential circuits, give us a basic memory element.

The registers and counters are combinations of several flip-flops and their use in digital circuits is very important. A register is a group of memory elements which stores a binary word and it may modify the stored word in a particular fashion as is desired by the application in which it is used. It is capable of shifting the stored binary word a step or more towards left or right. A counter is basically a register which counts the number of clock (CLK) pulses arriving at the input. In this unit you will learn about several types of registers and counters.

In a digital computer, the 'memory' is the key device. You often come across the terms RAM, ROM, floppy, and hard disk. The registers are memory devices. They are connected in different ways and are available as integrated circuits in the market. You know that the whole world is analogue. To use a digital circuit or computer, you have to convert an analogue quantity into a digital one so that the quantity can be acted upon or manipulated by the digital circuit or computer as per requirement. The output of the digital circuit is also in digital form, which cannot be perceived by you. For this purpose, you have to convert the digital output into analogue form. Therefore, it is necessary to have circuits which will convert an analogue quantity (such as voltage) into digital form and vice versa. Such circuits are analogue-to-digital (AD) and digital-to-analogue (DA) converters. In this unit, you will learn different kinds of memories, and AD and DA Converters also.

Objectives
After studying this unit, you should be able to:

- explain the functioning of buffer and controlled buffer registers,
- describe the functioning of the shift register,
- describe the functioning of the shift left and shift right registers,
- explain the functioning of the controlled shift register,
- explain the construction and functioning of an asynchronous (ripple) counter,
- describe the functioning of ring and mod 10 (decade) counters,
- explain several memory terms used in digital circuits,
- explain the capacity of memory and specify how many bits can be stored in a memory device,
- describe general memory operation,
- explain and distinguish between RAM and ROM,
- describe the functioning of a digital-to-analogue and an analogue-to-digital converters.

3.2 REGISTERS

A register is a group of memory elements which stores a binary word and it may modify the stored word in a particular fashion as is desired by the application in which it is used. It is capable of shifting the stored binary word a step or more towards left or right. In this section you will learn about registers.

3.2.1 Buffer Register

The simplest kind of register is a buffer register, which stores a binary word. It is made up of several D flip-flops, the number of which depends on the number of bits present in a binary word. A buffer register for storing a 4-bit word, $X_3X_2X_1X_0$ with $Q_3Q_2Q_1Q_0$ as its output word is shown in Fig. 3.1.

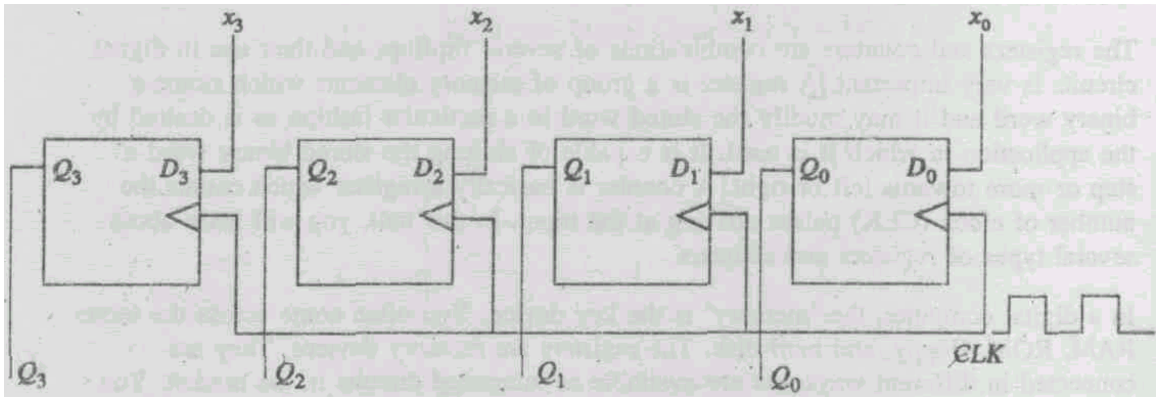


Fig. 3.1 Buffer Register

Each flip-flop is positive edge triggered. At every clock the output, Q of each flip-flop is the same as the input X. For this 4-bit register, we can write

$$Q_3Q_2Q_1Q_0 = X_3X_2X_1X_0$$

In chunked notation, this expression is written as

$$Q = X$$

This circuit is very basic. We should have some method to hold the input word till such time we are ready to store it. This is achieved by a controlled buffer register.

3.2.2 Controlled Buffer Register

A controlled buffer register is shown in Fig. 3.2. All flip-flops are with CLEAR which resets flip-flops when HIGH. The CLEAR is inactive when LOW. The control LOAD terminal when HIGH allows input X to reach the flip-flop and does not allow when LOW. When CLEAR is HIGH, all flip-flops reset and the stored word is

$$Q = 0000$$

When CLR returns. LOW, the register is ready for desired action.

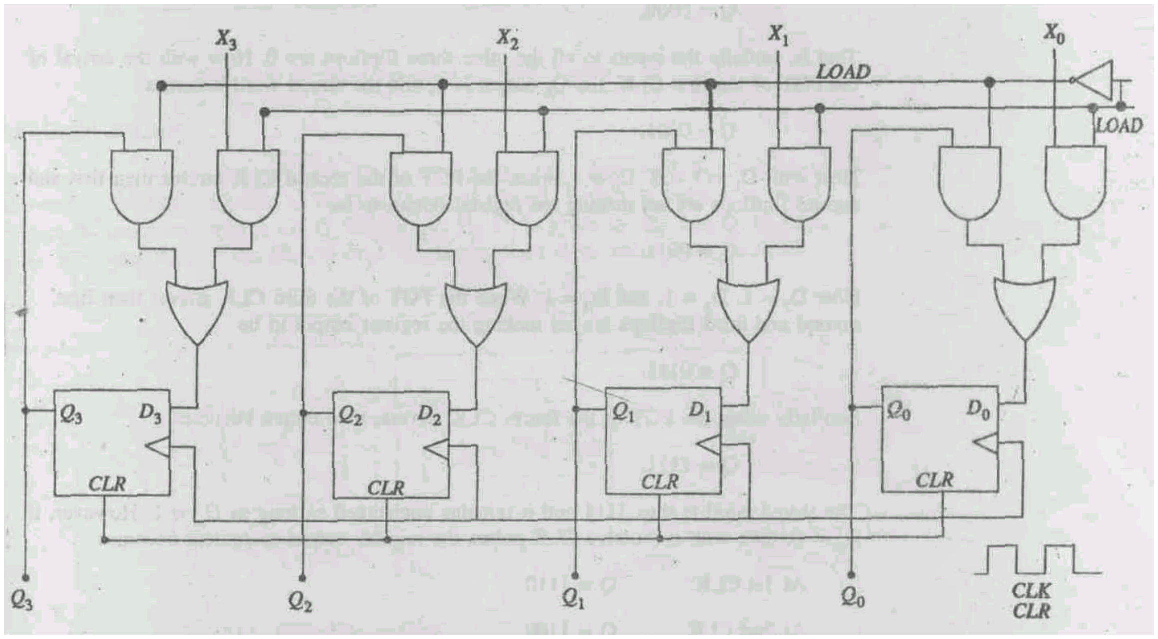


Fig. 3.2 Controlled buffer register

The control terminal LOAD determines the circuit function. When LOAD is HIGH, the data X is allowed to reach the flip-flop.

However when LOAD is LOW, $\overline{\text{LOAD}}$ is HIGH, which allows the Q outputs to go to D inputs. It means that so long as the LOAD is LOW, the input data X is circulated or retained at the PGT of each CLK. That is, the contents of the register continue to remain unchanged so long as LOAD is LOW.

When the LOAD is made HIGH, the word or data X is transmitted to the D inputs and the flip-flops are ready to change. When the PGT of the CLK arrives, the X input is loaded and is available at Q outputs, and

$$Q_3Q_2Q_1Q_0 = X_3X_2X_1X_0$$

With LOAD returning to LOW, the input word is stored. That is, so long as the LOAD remains LOW, it is not affected even when X input is changed. In this kind of register, as is seen from the circuit, the input is given to all the flip-flops simultaneously and the output is also obtained from all the flip-flops simultaneously. This is quite often referred to as parallel-in/parallel-out register.

3.2.3 Shift Registers

The shift registers move the stored word towards left or right. Therefore there are two types of shift registers – Shift-left and shift-right registers. Shifting of bits of the stored word towards left or right is essential in arithmetical operations.

Shift Left Register

A register which shifts the bits of the stored word towards left, called shift-left register, is shown in Fig. 3.3. As is clear from the circuit, the data input D_{jn} sets up the first flip-flop, and the Q_0 output of this flip-flop sets up the second flip-flop; Q_1 sets up the third and Q_2 sets up the fourth; Sincere data is given to the input of the first flip-flop, i.e., D_{in} and the output is obtained simultaneously from all the flip-flops, the circuit is known as serial-in/parallel-out.

The working of shift-left registers can be understood by the following example:
Consider that the input data D_{in} is 1, i.e., the input to flip-flop-1, D_0 = 1 and the initial output

$$Q = 0000$$

That is, initially the inputs to all the other three flip-flops are 0. Now with the arrival of
the PGT of the first CLK, the Q_0 output is 1, and the stored word becomes

$$Q = 0001$$

Now with D_1 = 1 and D_0 = 1, when the PGT of the second CLK arrives then the first and the second flip-flops are set, making the register output to be

$$Q = 0011$$

Now D_2 = 1, D_1 = 1, and D_0 = 1. When the PGT of the third CLK arrives then the first, the second and the third flip-flops are set making the register output to be

$$Q = 0111$$

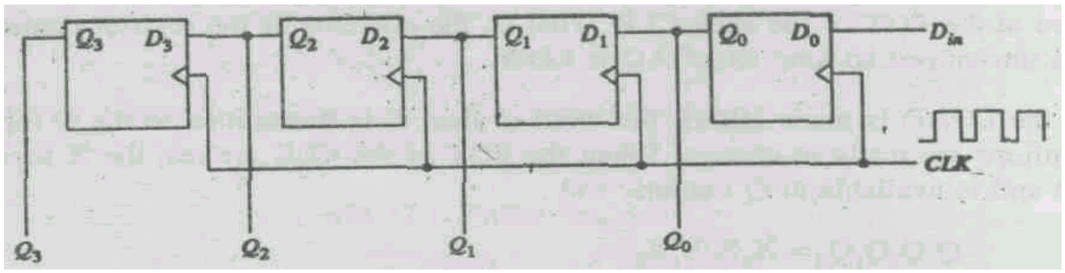
Similarly when the PGT of the fourth CLK arrives, then output becomes

$$Q = 1111$$

The stored word is thus 1111 and it remains unchanged so long as $D_{in} = 1$. However, if $D_{in} = 0$, then with successive CLK pulses the register output or content becomes

At 1st CLK	$Q = 1110$
At 2nd CLK	$Q = 1100$
At 3rd CLK	$Q = 1000$
At 4th CLK	$Q = 0000$

This word 0000 remains stored so long as $D_{in} = 0$. The entire operation of the shift-left register in terms of its tuning diagram is shown in Fig. 3.4.



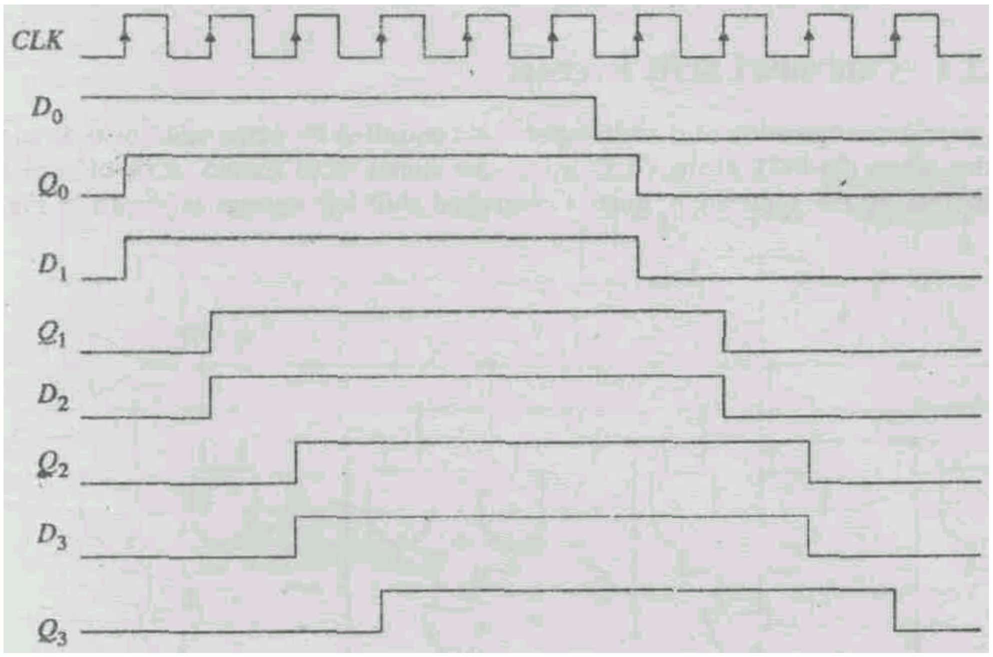


Fig. 3.4 Timing diagram of shift-left register

Shift Right Register

The circuit for a shift-right register is shown in Fig. 3.5. The data input, D_{in} is given to the input of the fourth flip-flop as D_3. The Q output of each flip-flop is fed back to the D input of the previous flip-flop, i.e. Q_3 is given to D_2, Q_2 is given to D_1 and Q_1 is given to D_0. When the PGT of the CLK arrives, the stored word shifts one step to its right.

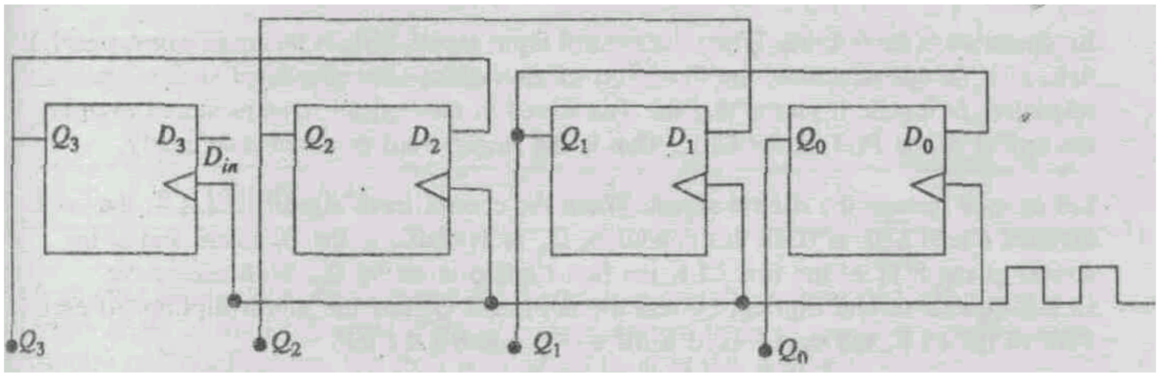


Fig. 3.5 Shift-right register

The operation of the shift-right register can be understood as follows. Consider that in the beginning D_{in} = 1, and

Q = 0000

At the arrival of the PGT of the first CLK, D_3 = 1, and all other D inputs are 0, Therefore, the fourth flip-flop is set and the stored word is

Q = 1000

Now D_3 = 1 and D_2 = 1. When the PGT of the second CLK arrives, the third and the fourth flip-flops are set, and the stored word becomes

Q = 1100

Similarly, with the arrival of the PGT of the third CLK, the stored word becomes

Q = 1110

And with the arrival of the PGT of the fourth CLK, the stored word becomes

Q = 1111

3.2.4 Controlled Shift Register

In general the operation of a shift register is controlled by some additional arrangement so that when the PGT of the CLK arrives the stored word should or should not change as desired by the application. Such a controlled shift-left register is shown in Fig. 3.6.

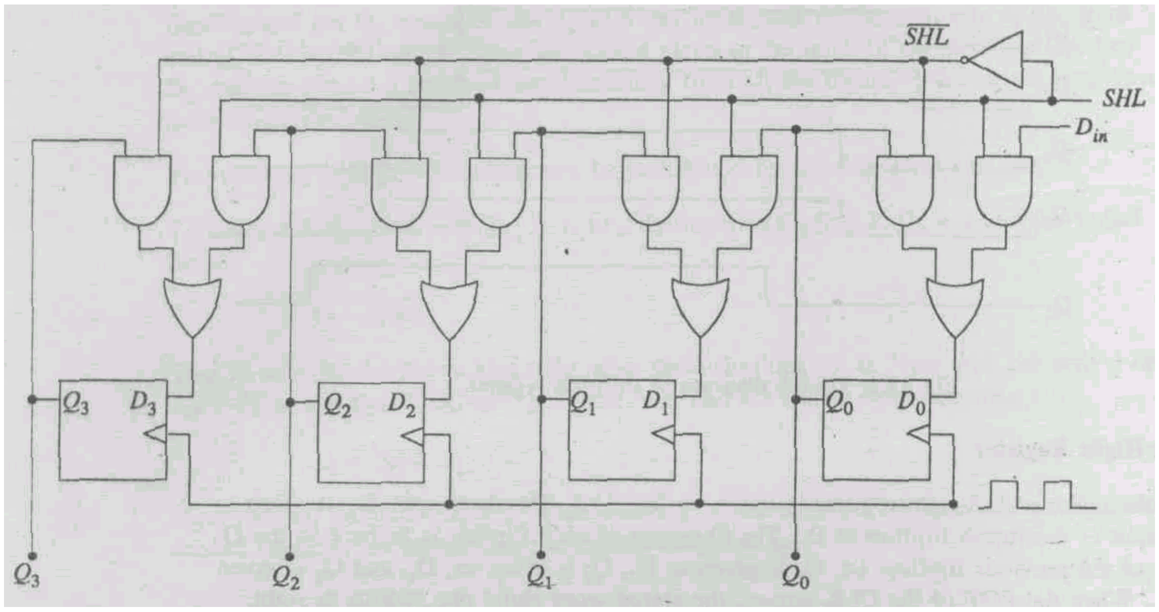


Fig. 3.6 Controlled Shift-left register

Its operation is as follows: When the control input signal SHL is 0, the inverted signal $\overline{SHL}$ is 1. In this condition, the Q outputs of the flip-flops are circulated back to their respective D inputs. It means that the data stored in the register remains stored even at the arrival of the PGT of the CLK. That is, the stored word is stored indefinitely.

Let us now reverse the control signal. When the control input signal SHL is 1, the inverted signal $\overline{SHL}$ is 0. In this condition, D_{in} is available at the D_0 input, and at the arrival of the PGT of the first CLK the first flip-flop is set by D_0. With successive CLKs, Q_0 sets the second flip-flop, Q_1 sets the third, and Q_2 sets the fourth flip-flop. At each PGT of the CLK, the stored word shifts a step towards the left.

The loading of the word to be stored in this kind of register is done serially, that is the word is loaded by entering one bit per CLK. To store a 4-bit word we require four CLK pulses. For example, X = 1001 is loaded serially as follows:
Keep SHL = 1, and make D_{in} = 1. At the first CLK

Q = 0001

Now keeping SHL = 1, make D_{in} = 0. At the second CLK

Q = 0010

At the third CLK

Q = 0100

Now keeping SHL = 1, make D_{in} = 1. At the fourth CLK

$Q = 1001$

The data is thus entered serially and the stored word is available parallel from all the Q outputs.

All the bits of a word can, however, be loaded simultaneously and it needs only one CLK pulse as is done in a buffer register. The circuit for this kind of loading is given in Fig. 3.7, which can be used for serial as well as parallel loading of a word to be stored.

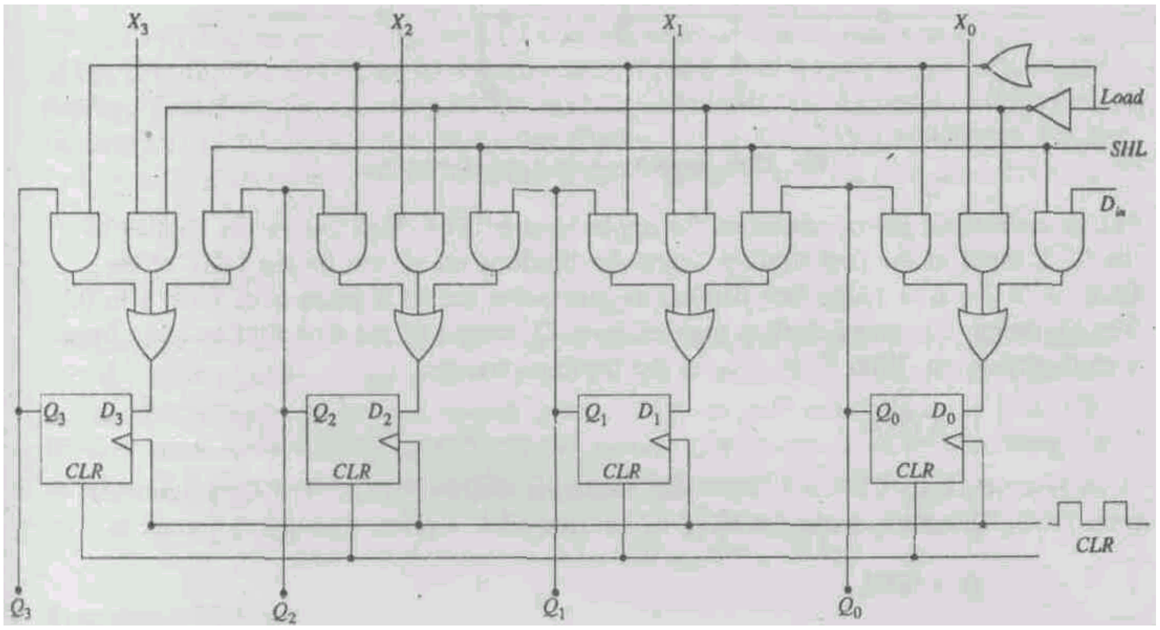


Fig. 3.7 Controlled shift register with parallel as well as serial loading arrangement

If LOAD and SHL are 0, the output of the NOR gate is 1. With this condition the Q outputs are circulated back to their respective D inputs. The previously stored word continues to be stored. The register, in this state, is known as inactive register.

If the LOAD is 0 and SHL is 1, the register is used for serial loading as is done in the case of register shown in Fig. 3.6. If Load is 1 and SHL is 0, then X bits set the D inputs simultaneously on the first CLK itself. This is the case of parallel loading.

For a word of more bits to be stored, more flip-flops are required. Actually you require the same number of flip-flops as is the number of bits in the word to be stored.

3.3 COUNTERS

A counter is an equivalent of binary odometer. It counts the number of CLK pulses which arrive at the CLK input. Basically, there are two types of counters, asynchronous (ripple) and synchronous. We shall learn about them in this section.

3.3.1 Asynchronous (Ripple) Counter

Fig. 3.8 shows a 4-bit binary counter circuit which is made by using JK flip-flops. All the JK inputs are kept at 1. The CLK signal is given to the CLK input of the first flip-flop. The Q_0

output is given to the CLK input of the second flip-flop, Q_1 output is given to the CLK input of the third, and so on. The CLR input is activated when it is made 0. All CLR inputs have been joined together so that all the flip-flops could be reset simultaneously. Such a counter where each flip-flop output serves as the CLK input for the next flip-flop is known as asynchronous counter. This name is given because all the flip-flops do not change state in exact synchronism with the CLK pulses. Only the first flip-flop responds to the CLK pulse, while all others wait for the previous flip-flops to change state. Therefore, there is a delay between the responses of consecutive flip-flops. This type of counter is also known as ripple counter.

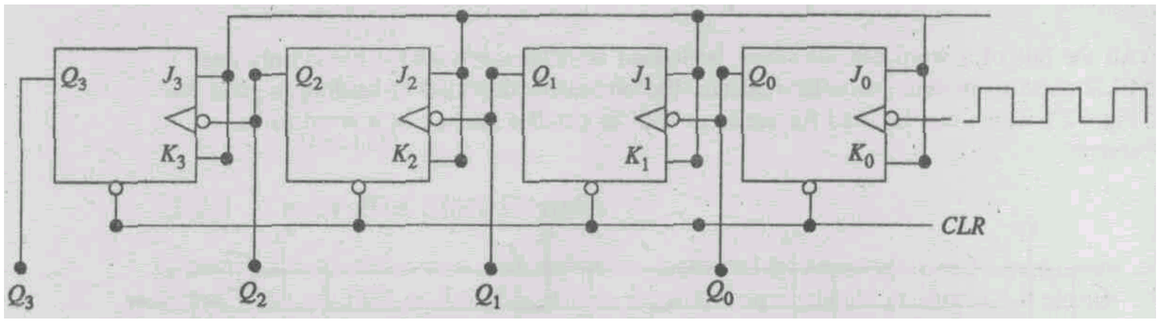


Fig. 3.8 Asynchronous (ripple) counter

Let us understand the operation of the ripple counter. The clock pulses are applied to the CLK input of the first flip-flop. Since the flip-flops are driven by the NGT of the CLK, with J = K = 1, the first flip-flop toggles when the CLK pulse goes form 1 to 0. The Q_1 output of second flip-flop toggles when Q_0 output of the first flip-flop goes from 1 to 0, and so on. With CLR = 0, all the flip-flops are reset to

$Q = 0000$

After resetting keep CLR = 1. Now the counter is ready to count. The Q_0 toggles for each NGT. Therefore, when the NGT of the first CLK arrives, then the Q output is

$Q = 0001$

At the second CLK, Q_0 toggles from 1 to 0 which acts as a NGT for the CLK input of the second flip-flop, the Q_1 output of which toggles to 1. Therefore,

$Q = 0010$

At the third CLK, Q_0 toggles from 0 to 1, and there is no change in Q_1. Therefore,

$Q = 0011$

At the fourth CLK, Q_0 toggles from 1 to 0, resulting in toggling of Q_1 from 1 to 0. The Q_1 going from 1 to 0 acts as a NGT for the CLK input of the third flip-flop, the Q_2 output of which toggles from 0 to 1. Therefore,

$Q = 0100$

The Q output of the counter at each CLK is summarised in Table 3.1.

Table 3.1: No. of CLK pulses

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Next CLK resets alt the flip-flops and the Q outputs on successive CLK would be

16	0000 (recycles)
17	0001
18	0010
..	….
..	….

While analysing the Q outputs, we find that whenever a flip-flop resets to 0, the output of the next flip-flop is 1. That is, the resetting of a flip-flop sends a carry to the next higher flip-flop. Therefore, the counter acts like a binary odometer. The Q_0 output of the first flip-flop acts as a LSB and that of the last flip-flop as the MSB. This would now be clear as to why asynchronous counter is called a ripple counter. It is because the carry in the output moves like a ripple on water.

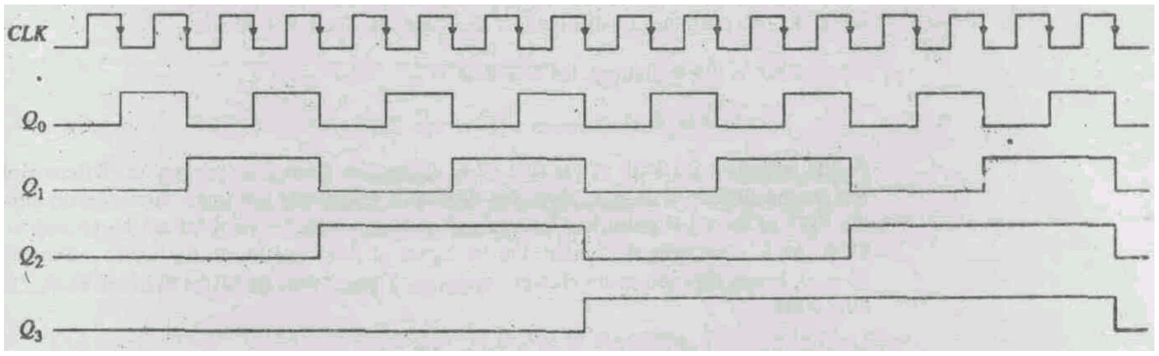
Mod of a Counter

The counter described above has 16 distinct states or outputs (0000 to 1111). It is said that the Mod number of this counter is 16. The Mod number of a counter is equal to the number of states which the counter goes through in each complete-cycle before it recycles back to its starting state. The Mod number can be increased by increasing the number of flip-flops. If *n* is the number of flip flops used in a counter, then

Mod Number = 2^n

Frequency Division

The output of each flip-flop and the CLK are shown in Fig. 3.9. It is clear that the frequency of Q_0 output is half the frequency of the CLK. The Q_0 output acts as a CLK to the second flip-flop, and the frequency of its Q_1 output is half the frequency of Q_0 or one-fourth the frequency of the CLK.



First flip-flop divides by	2
Second flipflop divides by	4
Third flipflop divides by	8
Fourth flipflop divides by	16
mth flipflop divides by	2^n

SAQ 1

If the frequency is 100 kHz, what will be the output frequency of the third flip-flop of a ripple counter?

3.3.2 Synchronous Counter

As stated in the previous section, there is a lot of time delay in a ripple counter because a carry has to pass through n flip-flops. Therefore, the ripple counters are very slow. If Tpd is time delay for one flip-flop, then for n flip-flops the time delay is nTpd. Therefore, there is a need for synchronous counter in which all the flip-flops respond on each CLK pulse simultaneously. The circuit for a synchronous counter is given in Fig. 3.10.

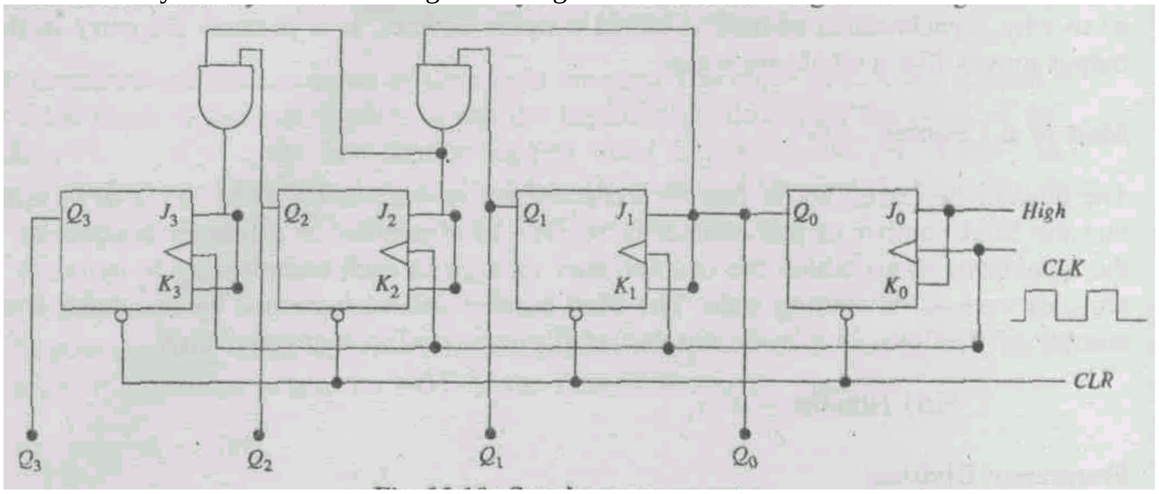


Fig. 3.10 Synchronous counter

The CLK inputs of all the flip-flops are connected with each other so that the CLK signal reaches them simultaneously. Similarly, the CLR inputs of all the flip-flops are connected with each other so that they can be reset simultaneously by making CLR = 0. All J and K inputs have not been connected to each other as is the ease in ripple counter. The JK inputs of the first flip-flop are always kept at 1. The flip-flops toggle at the arrival of the PGT of the clock pulse at their CLK inputs provided their JK inputs are at 1. The operation of this counter can be understood as follows:

When reset in the beginning, the Q output is

$Q = 0000$

At the arrival of the PGT of the first CLK, Q_0 toggles from 0 to 1 bringing JK inputs of the second flip-flop also to 1. Now this flip-flop is also ready to toggle. However, by now the PGT of the CLK pulse has disappeared. It has to wait for the PGT of the second CLK. As is clear from the circuit, the JK inputs of third and fourth Flip-flops continue to be at 0, hence they are in no change condition. Thus, at the arrival of the first CLK,

$Q = 0001$

Now at the PGT of the second CLK, Q_0 toggles from 1 to 0 and Q_1 toggles from 0 to 1. However, the JK inputs of the third flip-flop continue to be at 0 because the inputs to the AND gate (the output of which is connected to these JK inputs) are $Q_1 = 1$ and $Q_0 = 0$. Therefore, it is in no change condition and hence Q_2 continues to be at 0. Similarly, the JK inputs of the fourth flip-flop are at 0 because $Q_2 = 0$ and therefore Q_3 continues to be at 0. Thus, at the arrival of the PGT of the second CLK,

$Q = 0010$

At the arrival of the PGT of the third CLK, the JK inputs of second, third and fourth flip-flops are at 0; therefore they are in no change condition. Only the first flip-flop is ready to toggle from 0 to 1. Thus at the third CLK,

$Q = 0011$

Now since Q_1 and Q_0 are at 1, therefore JK inputs of the third flip-flop are at 1. However, the JK inputs of the fourth flip-flop are still at 0. Hence the first three flip-flops are ready to toggle at the arrival of the PGT of the fourth CLK. Thus Q_0 and Q_1 toggle from 1 to 0, and Q_2 toggles from 0 to 1. Therefore, at the fourth CLK,

$Q = 0100$

Successive, Q outputs are 0101, 0110, and 0111. At the arrival of the eighth CLK, the JK inputs of all flip-flops are at 1. The Q outputs of all the flip-flops toggle, and we have

$Q = 1000$

The successive CLK pulses change the Q outputs in the same way as described above. The Q output at each CLK is summarised in Table. 3.2.

Table 3.2

No. of CLK pulses	Q
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

At the arrival of the PGT of the next CLK, the counter resets to Q = 0000.

A counter of any length can be built by adding more number of flip-flops. The advantage of synchronous counter is that it requires only one propagation delay time in getting the Q output. The Mod of this counter is also 16 $(= 2^4)$.

3.3.3 Controlled Synchronous Counter

The circuit of the controlled synchronous counter is shown in Fig. 3.11. The COUNT is the control input. When the COUNT is at 0, the JK input of all the flip-flops are at 0 keeping the flip-flops in no change condition. When the COUNT is at 1, the circuit is the synchronous counter which works exactly in the same way as the counter of

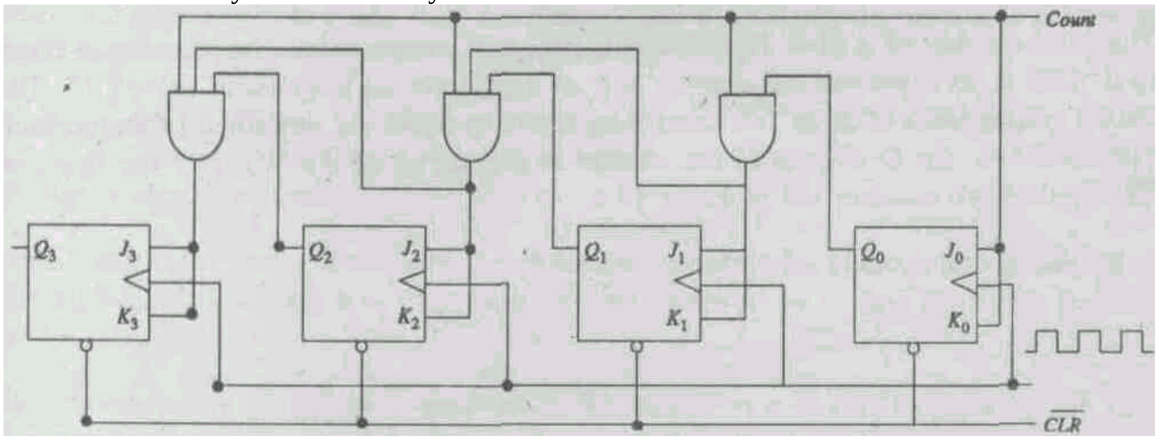


Fig. 3.11 Controlled synchronous counter

3.3.4 Ring Counter

The ring counter does not count the binary number. The Q output of this counter has only a single 1 bit and all other bits are 0. At each CLK the bit 1 shifts a step to its left. The digital circuit of the ring counter is shown in Fig. 3.12. It is made up of D flip-flops. Note that the CLR inputs of second, third and fourth flip-flops are connected with the PRESET input of the first flip-flop. It means that when CLR is brought to 0, it presets the Q_0 to 1, and resets Q_1, Q_2 and Q_3 outputs to 0.

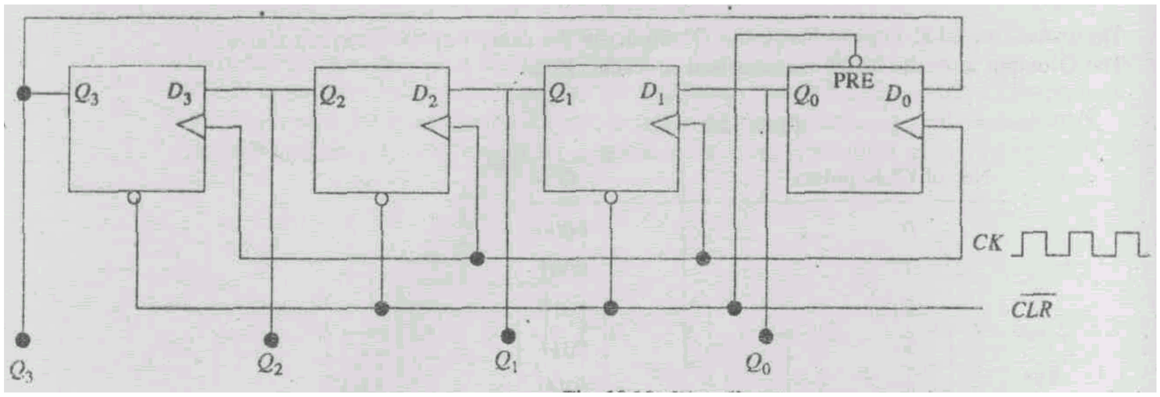


Fig. 3.12 King Counter

The working of the ring counter can be understood as follows. When the CLR is made active, i.e., when it is made 0, the first flip-flop is set and all others are reset. Therefore, Q output is

$$Q = 0001$$

Now $Q_3 = 0$ is fed back to D_0 input of the first flip-flop. Therefore, at the arrival of the PGT of the first CLK, Q_0 is 0 and Q_1 is 1, while Q_2 and Q_3 continue to be 0. Thus, at the first CLK,

$$Q = 0010$$

At the time the PGT of the second CLK arrives, D_0, D_1 and D_3 are at 0, and D_2 is at 1. Therefore, the Q output is,

$$Q = 0100$$

Similarly at the arrival of the PGT of the third CLK, .the Q output becomes,

$$Q = 1000$$

The PGT of the fourth CLK starts the cycle again, and

$$Q = 0001$$

Thus we find that bit 1 shifts a step to its left and it rotates back to its initial position, and so on. It is because of this effect that it is known as ring counter. Instead of a ring of four bits, if you want a bigger ring then add more flip-flops.

3.3.5 Mod 10 (Decade) Counter

The Mod number of a Mod 10 counter is 10, i.e., it counts from 0 to 9 and then resets to 0. This is an asynchronous counter and its digital circuit is given in Fig. 3.13. The circuit counts from 0000 to 1001 and then resets to 0000. As described in the section on ripple counter, the Q outputs of the counter at the arrival of the NGTs of the first nine CLK pulses are summarised in Table 3.3.

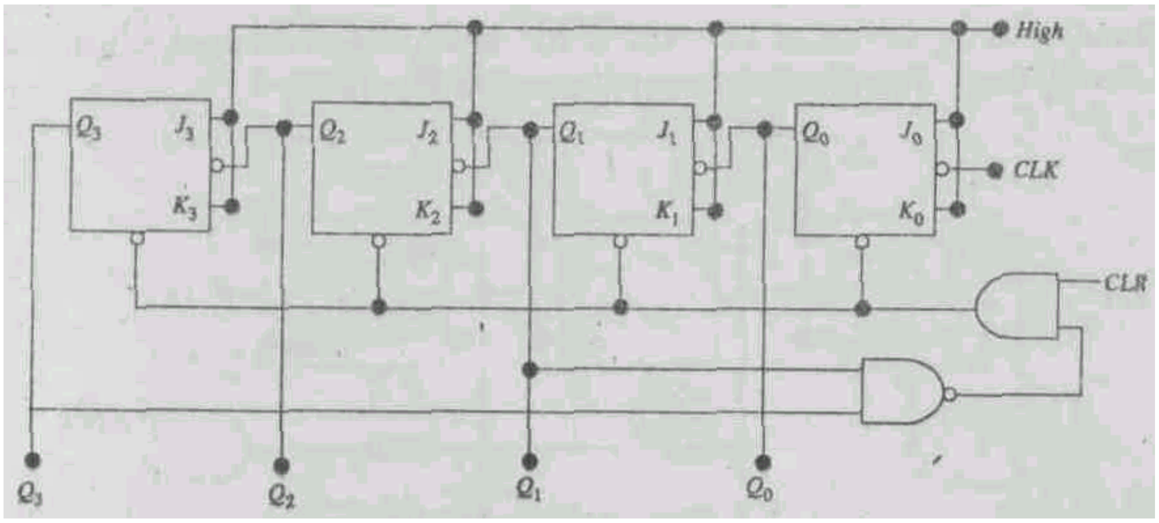


Fig. 3.13: Mod-10 (Decade) counter

Table 3.3

No. of CLK pulses	Q	Decimal Equivalent
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	0000	0 (resets)

The circuit skips the states from 10 to 15, i.e., from 1010 to 1111. The circuit is made to skip these states by the combination of NAND and AND gates present in the circuit. The idea is that when Q = 1010 is expected at the tenth CLK, the flip-flops should be cleared to be reset to 0000 which is done by bringing CLR input to 0. This is achieved by connecting Q_1 and Q_3 to the inputs of a NAND gate which gives output 0 when its inputs are 1 (which is the case when Q =

1010 is expected at the tenth CLK). The output 0 of the NAND gate makes the AND gate output to be 0. This makes the CLR active and the flip-flops reset to

$$Q = 0000$$

When the CLR is a made inactive, i.e., when CLR = 1, the counter becomes ready to count once again.

Since it takes 10 CLK pulses to reset the counter, the frequency of the Q_3 output is one-tenth of that of the CLK. It is therefore called a divide-by-10 circuit. It is used in BCD applications and frequency counters.

Example 3.1

Design a Mod 5 counter.

Recall that in a Mod 10 counter, the expected output of the counter on the arrival of the 10th CLK, i.e. 1010, was used to reset the counter so as to skip the states from 1010 to 1111.

In this example a mod 5 counter which will count from 0 to 4 is to be designed. It means that at the arrival of the 5th CLK, the counter should reset. For counting up to 4, not more than three bits are required. Therefore, we require three JK flip-flops. The expected output at the arrival of 5th CLK is 101 which should be used to activate CLR to reset the flip-flops. The required circuit is given in Fig. 3.14.

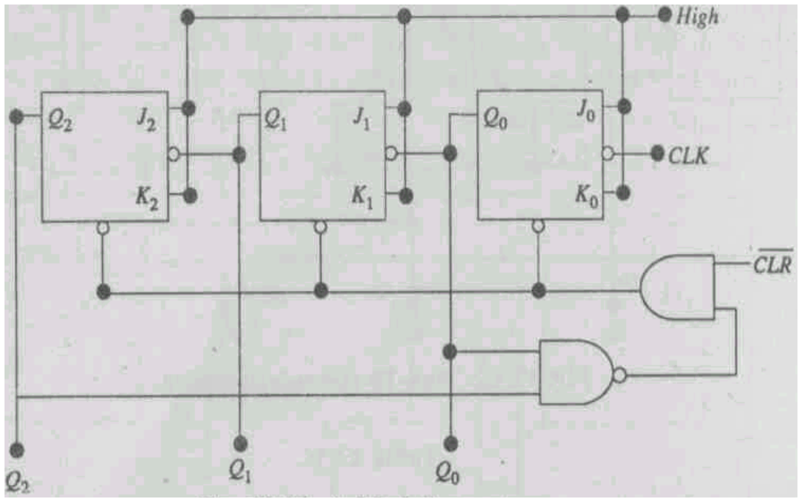


Fig. 3.14 A Mod 5 counter

SAQ 2
What is the Mod of a counter which consists of six flip-flops?

3.4 SEMICONDUCTOR MEMORIES

The advantage of digital systems over analogue systems is their ability to store information for short as well as long periods which makes them versatile. A digital computer has a minimum amount of memory with the help of which it is able to manipulate information or data as desired by us. It also has memory which makes it capable of storing this information as long as we want and make it available to us whenever we want.

We have already studied the basic memory element which stores a single bit, that is the flip-flop. We have also learnt about registers which store a word of any number of bits. The registers are very high-speed memory elements and are used extensively in the internal operation of a digital computer. With the advent of integrated circuit technology and its further advancement in LSI (Large Scale Integration) and VLSI (Very Large Scale Integration), a large number of registers can be obtained on a single chip.
The cost of these semiconductor devices is also decreasing. However, the cost of these devices per bit of storage is very high, which prohibits their use as mass storage devices. A computer has internal memory which is constantly in communication with the central processing unit of the computer as a program of instructions is being executed. The program and any other information or data used by the program are also stored in the internal memory.

The mass storage memory devices are external to the computer and are capable of storing millions of bits even without requiring any electrical power. The mass storage memory is generally very slow compared to the internal memory and the information stored is the one which is not currently required by the computer. It is supplied to the computer only when required. The mass storage memory devices are floppies, magnetic tapes and disks, etc. The cost of per bit storage of these devices is much less compared to the internal memory.

3.4.1 What a Memory is!

A 'memory' is simply an array of registers, each register storing a word. Every register has an address number which identifies the location of a word in a memory. The location of a word is nothing but the register that stores the word to be identified. The address of each location is unique and is described by a binary number. To illustrate, let us consider that we have a memory which consists of eight registers. It is clear that this memory has eight memory locations. The unique addresses of the memory locations are given in Table 3.4.

Table 3.4

Address	Location
000	word 0
001	word 1
010	word 2
011	word 3
100	word 4
101	word 5
110	word 6
111	word 7

Each word in the memory is thus identified by an address. By a read operation, the binary word stored in a memory location is sensed and, if desired, it can be transferred to another device. For example, if we have to read word 6, then we have to do read operation on address 110. By a write operation, a new word can be placed or stored on a particular memory location.

The memories are volatile and non-volatile. A memory is volatile if it requires electrical power to store information and if the power is removed then the stored information is lost. Many types of semiconductor memories are volatile. The non-volatile memory retains the stored information even when electrical power is removed. The mass storage memory devices fall into this category. The other types of memories like Random-Access Memory (RAM) and Read Only Memory (ROM) will be described in later sections.

3.4.2 Capacity of Memory

Before understanding the meaning of capacity of memory let us know some of the memory terms. A device, such as a flip-flop, which can store a single bit (0 or 1) is called a memory cell. In a memory, a group of bits or cells which represents instructions or data is known as a memory word. A register consisting of four flip-flops is a memory which can store a 4-bit word. Similarly, a register having eight flip-flops is a memory which can store a 8-bit word. The size of the word in modern computers range from 4 to 64 bits. A 4-bit word is called a nybble and 8-bit word is called a byte. A byte is the most commonly used word size.

The capacity of a memory is a term used to express how many bits can be scored in a particular memory device or in a complete memory system. For example, let us say that we have a memory which can store 2048 eight-bit words. This memory can store $2048 \times 8 = 16384$ bits and we say that this memory can store 16384 bits. Another way to express this capacity is as 2048×8. This kind of expression of memory means that there are 2048 words and the size of the word is 8 bits. The number of words in a memory is generally a multiple of 1024. The figure of $1024 = 2^{10}$ is commonly represented as 'IK'. Thus memory capacity of 2048×8 is also expressed as $2K \times 8$. For larger memories, '1M' or '1 meg' is used for $2^{20} = 1{,}048{,}576$. Therefore, a $4M \times 8$ memory has a capacity of $4{,}194{,}304 \times 8$ or alternatively of 33,554,432 bits.

Example 3.2
A user has two memory devices. One of these stores 10M words of 8-bit size, while the other stores 2M words of 16-bit size. Which of the two stores most bits?

Solution
The two memories are of $10M \times 8$ and $2M \times 16$.

$$10M \times 8 = 10 \times 1{,}048{,}576 \times 8 = 83{,}886{,}080 \text{ bits}$$

$$2M \times 16 = 2 \times 1{,}048{,}576 \times 16 = 33{,}554{,}432 \text{ bits.}$$

Therefore, the memory of $10M \times 8$ stores more bits.

SAQ 3
A certain memory is specified as $32K \times 8$.

(a) What is the size of the word?

(b) What is the total number of bits stored by the memory?

3.4.3 Random-Access Memory (RAM)

A Random-Access Memory (RAM) is also known as read-write memory. It is a group of registers which have their unique addresses, and using an appropriate address the stored word on a memory location can be read and new contents, if desired, can be written on this location. Actual physical location of a stored word in RAM does not make any difference because the access time (which is the speed of a memory device, that is the time required to perform the read operation) is same for any address in the memory. The semiconductor RAMs are volatile, because when the electrical power is turned off the stored data is lost.

While working with a computer when a user is giving instructions or doing some calculations using a program, it is the RAM that is being continuously used to read the stored information and write the new information. You might have heard the term being used that a particular computer has 1 or 4 MB (Mega Byte) RAM or so.

3.4.4 General Memory Operation

Despite the fact that the internal operation of each type of memory is different, the general memory operation remains the same for all. Every memory system will have terminals for data input, data output, address input, selecting read or write operation and for enabling or disabling the memory operation.

In a general memory operation, first the address of a memory location is selected where the read or write operation is to be performed. Decide whether you want to perform read or write operation. If you want to write, then perform the write operation and supply the data to the memory. If you want to read, then perform the read operation and hold the output data coming from the memory. If you want that the memory should respond to the address and read/write operation, then enable the memory and if you do not want the memory to respond then disable the memory.

To illustrate the aforesaid operation consider a 16×4 memory device shown in Fig. 3.15. Since the word size is 4 bit, it has four data input lines and four output data lines. It will also have four address lines because the given memory device has 16 memory locations which can be expressed by 4-bit addresses. It has one read/ write command terminal; it will read if kept at 1 and write if at 0. It has one enable/ disable terminal. In the diagram shown, if this terminal is kept at 1, it enables the memory and it disables if kept at 0. A virtual arrangement of memory cells into 4-bit words is shown in Fig. 3.16 along with their address.

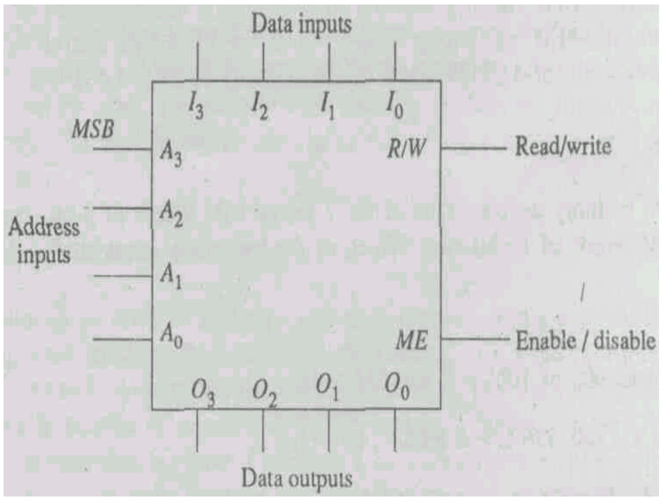


Fig. 3.15 A 16×4 memory

Memory cells				Addresses
0	0	1	1	0000
1	0	0	1	0001
0	0	0	0	0010
0	1	1	1	0011
1	0	1	0	0100
1	0	0	0	0101
1	1	1	0	0110
0	0	0	1	0111
1	0	1	1	1000
0	0	0	0	1001
1	1	0	0	1010
0	1	0	1	1011
1	0	0	1	1100
1	1	1	1	1101
0	0	0	1	1110
1	0	1	0	1111

Fig. 12.16 Virtual arrangement of memory cells into sixteen 4-bit words

As a further illustration, let us say that you want to change the word 1111 stored in the fourteenth location to 0101. To do so, choose the address 1101, keep read/write terminal at 1 so that write operation is chosen, and then feed the desired word 0101 to the data input. Thus the new word is stored in place of the old one.

3.4.5 Read Only Memory (ROM)

The Read-Only Memory (ROM) is a broad class of semiconductor memories which are designed for those kinds of applications where only read operation is required. These memories hold the data permanently. In general, no new data can be written on ROM but it can be read. The data to be stored permanently in ROM is selected and built in by the manufacturer at the time of 1C fabrication. However, there are some varieties of ROM in which data can be entered electrically once only. The process of entering the data is known as programming or burning the ROM. Such ROMs are called PROMs (Programmable-ROM). In some other ROMs the data stored can be erased and the ROM can be reprogrammed. Such ROMs are called EPROMs (Erasable-PROM). All ROMs are nonvolatile, that is, they keep storing the data even when electrical power is removed.

A typical block diagram of a 16×8 memory is shown in Fig. 12.17. It has four address lines, eight terminals for data output and one terminal called chip select (CS) which enables or disables the memory. To read the data, say, at the location with address 1010, we have to apply $A_3A_2A_1A_0 =$ 1010 to the address inputs and then select the chip select so as to enable the memory. The data output terminals will show the actual word stored in that location.

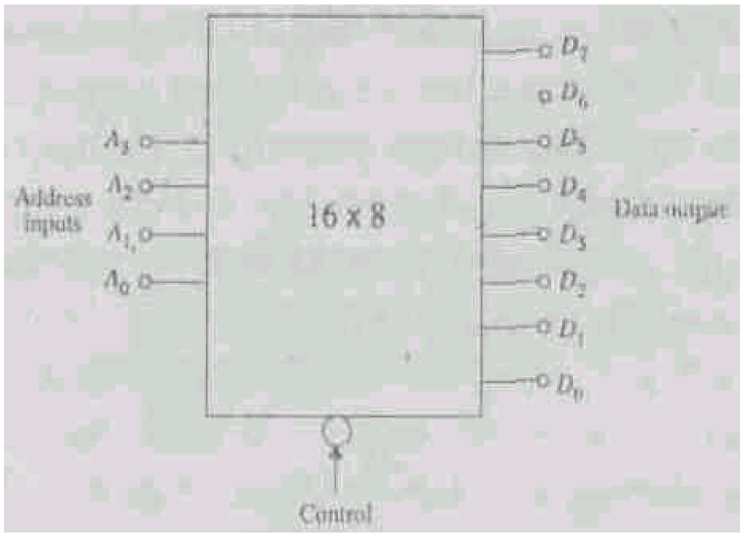
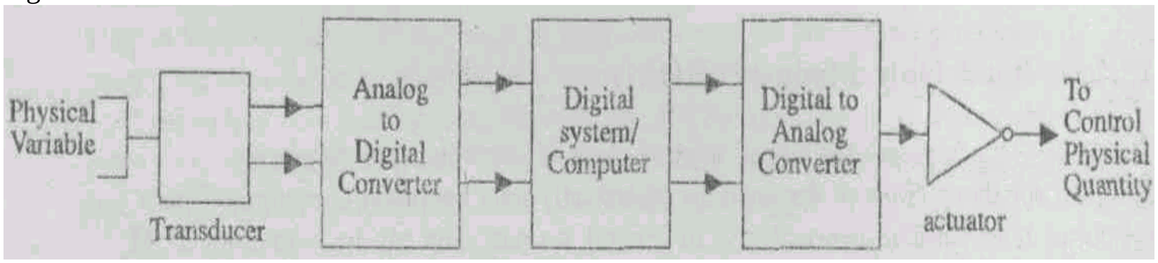


Fig. 3.17 Block diagram of a 16 x 8 ROM

3.5 A/D AND D/A CONVERTERS

As pointed out in the introduction of this unit, digital systems or computers perform all of their functions and internal operations using digital circuits which require digital inputs. A digital quantity will have a value either 0 or 1, while an analogue quantity can take any value over a continuous range of values and its exact value is significant. Most physical variables are analogue in nature, such as temperature, pressure, light intensity, audio signals, position, speed, etc. Therefore, it is essential to put an analogue quantity to be analysed using a digital system first in a digital form. The analogue-to-digital (A/D or ADC) converter is a digital circuit which converts an analogue quantity into digital form consisting of a number of bits that represents the value of the analogue input. This circuit is used as an interface between the digital system or computer and the analogue system of the input stage. The output of a digital system is digital and has to be converted back into analogue quantity. The digital-to-analogue (D/A or DAC) converter serves this purpose and its output is a proportional analogue voltage or current corresponding to an analogue quantity. This is used as an interface between the digital system or computer and the analogue system of the output stage.

Pictorially this is summarised in Fig. 12.18. Let us say that in a physical system a quantity, such as temperature, is to be controlled using a computer. This physical quantity is first converted into a corresponding voltage or current with the use of a transducer. A transducer is a device which converts a physical variable into an electrical signal. Thermistors, bolometers, photocells, thermocouples are some of the commonly available transducers. Actuator used in this illustration is a device that controls the physical quantity, temperature, in a computer-controlled system. In this section, we shall learn about design and working of digital-to-analogue and analogue-to-digital converters.



Physical Variable
Transducer
Analog to Digital Converter
Digital system/ Computer
Digital to Analog Converter
actuator
To Control Physical Quantity

Fig. 3.18 ADC and DAC used as interfaces

3.5.1 Digital-to-Analogue Converter

We are first treating Digital-to-Analogue Converter (DAC) because the Analogue-to-Digital Converter (ADC) requires the use of DAC. The circuit for DAC takes the BCD or binary input and converts it to a voltage or current that is proportional to the digital value. The digital input is generally derived from an output register of the digital system which can theoretically be of any number of bits. In general, the registers used are 8-bit registers. For the purpose of an illustration, let us consider that the digital output from the digital system is of four bits. Therefore, we require a DAC that can convert a 4-bit digital output to a proportional analogue value.

A block diagram of such a DAC is shown in Fig. 3.19. It has four binary input lines representing $A_3A_2A_1A_0$ and one output line representing corresponding proportional analogue quantity. Each 4-bit input has unique proportional output voltage. There are $2^4 = 16$ states that the binary input can have. Let us say that each input specifies a decimal number. Let us designate 1V output equivalent to decimal number 1, 2V as number 2, and so on.

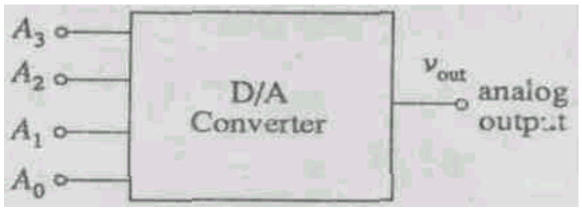


Fig. 3.19 Block diagram of DAC

The digital input and the corresponding proportional voltage as the analogue output is summarised in Table 3.5 In this example, the analogue output voltage is equal in volts to the binary number. The output voltage could be twice the binary number or any multiple. We can, therefore, write

$$\text{Analogue output} = k \times \text{digital input} \tag{3.1}$$

where k is proportionality factor, a constant for a given DAC.

The value of k in the given example is 1V, therefore V_{out} is 1V times the digital input. For $0110_2 = 6_{10}$, we get

$$V_{out} = 1\text{ V} \times 6 = 6\text{ V}$$

Table 3.5

A_3	A_2	A,	AO	V_{out}
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6

0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Analogue Output
The DAC output is technically not an analogue quantity. It can have only specific values. In the above example, it can have values only from 0 to 15 in steps of 1, that is 1, 2, 3, ..., 15. Therefore, strictly speaking it is digital. By increasing the number of input bits, the number of possible output values can be increased and the difference between successive values decreased. Thus the output can be made more or less analogue. For the time being we can only say that the DAC output is pseudo analogue.

Let us analyse the inputs and outputs given in Table 3.5 and consider only those digital inputs where one of the four bits is 1 and other three bits are 0. Such inputs and corresponding outputs are rewritten in Table 3.6.

Table 3.6

A_3	A_2	A_1	A_0	V_{out}
0	0	0	1	1
0	0	1	0	2
0	1	0	0	4-
1	0	0	0	8

It is clear from the entries included in Table 3.6 that the contributions of 1 are weighted according to their position in binary number. The bit A_3 has weight of 8, A_2 has weight of 4, A_1 has weight of 2 and A_0 has weight of 1. Thus the weight of the LSB is the smallest change. To check,

$$1001 = 8 + 0 + 0 + 1 = 9$$

Example 3.3
A 5-bit DAC produces 0.5V for 00001. Find V_{out} for 11010.

Solution
In the example the smallest change is 0.5V. Therefore,

$$\begin{aligned} 11010 &= 16\times0.5 + 8\times0.5 + 0 + 2\times0.5 + 0 = 8 + 4 + 1 \\ &= 13\text{V} \end{aligned}$$

Example 3.4
A 5-bit DAC produces a 10 mV output for a digital input of 10100. What will V_{out} be for a digital input of 11101?

Solution

$10100_2 = 20_{10}$

$V_{out} = k \times \text{digital input}$

$10 \text{ mV} = k \times 20_{10}$

$k = 0.5 \text{ mV}$

Now V_{out} for 11101 is obtained as follows:

$10100_2 = 29_{10}$

$V_{out} = 0.5\text{mV} \times 29$

$= 14.5 \text{ mV}$

SAQ4

The smallest change in a 4-bit DAC is 0.25V. What is V_{out} for a DAC input 1110?

Resolution (Step Size)

The resolution is the smallest change that can take place in the analogue output as a result of a change in the digital input. In the example of Table 3.5, the smallest change is 1 V. Therefore, the resolution in that example is 1 V. The resolution is also known as the step size. In the said example, the voltage rises in step of 1 V and goes up from 0 to 15 in 15 steps. Pictorially, this can be represented as shown in Fig. 3.20.

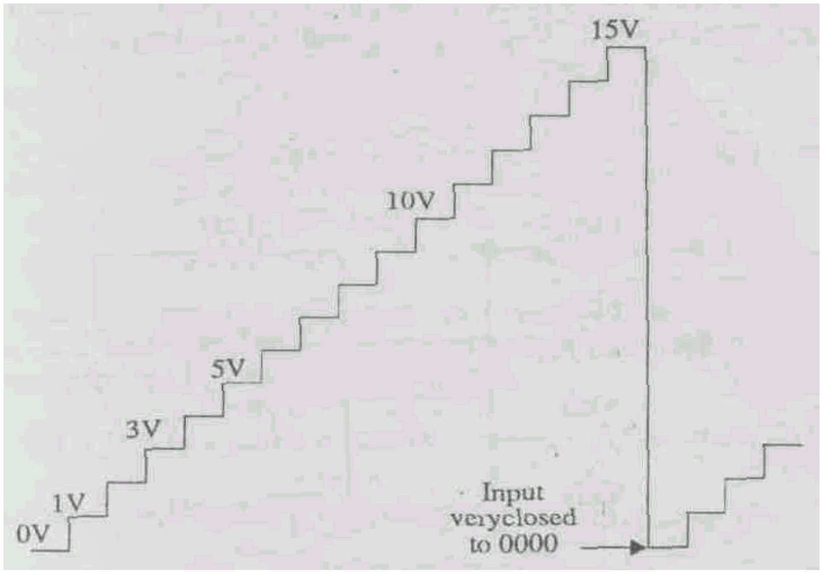


Fig. 3.20 Pictorial representation of V_{out} of the example given in Table 3.5.

It can easily be seen that there are 16 levels from 0 to 15V, but there are only 15 jumps. That is, the number of steps between 0 and 16 is 15. The number of steps in general can be calculated as

Number of steps = $2^n - 1$.

The resolution or step size is actually the constant k in equation (3.1). The percentage resolution is defined as

$$\% \text{ resolution} = \frac{\text{step size}}{\text{full scale (FS)}} \times 100 \qquad (12.2)$$

Example 3.5

What is resolution (step size) of the DAC of Example 3.4? Describe the staircase signal out of this DAC.

Solution

The LSB for this converter has a weight of 0.5 mV. This is the resolution (step size). A staircase waveform can be generated by connecting a 5-bit counter to the DAC inputs. The staircase will have $2^8 = 32$ levels from 0 mV up to a full scale output (V_{out} for input 11111 = $0.5\times16 + 0.5\times8 + 0.5\times4 + 0.5\times2 + 0.5 = 8 + 4 + 2 + 1 + 0.5 = 15.5$mV and 31 steps of 0.5 mV each.

SAQ 5

What is the percentage resolution of the DAC of Example 3.5?

DAC circuit

There are several methods and circuits for digital to analogue conversion which need not be known. A basic DAC circuit is obtained using an op-amp as a summing amplifier. A 4-bit DAC circuit is shown in Fig. 3.21. The input resistors are binary weighted, that is, they are in the ratio of 1 : 2 : 4 : 8. The output voltage of this circuit is given as

$$V_{out} = -\left(V_{A3} + \frac{1}{2}\cdot V_{A2} + \frac{4}{4}\cdot V_{A1} + \frac{1}{8}\cdot V_{A0}\right)$$

Negative sign indicates that it is an inverting amplifier. Note that the digital input bits can be either 0 or 1, therefore V_{A3}, V_{A2}, V_{A1}, V_{A0}. V_{A0} will have values either 0 or 5V.

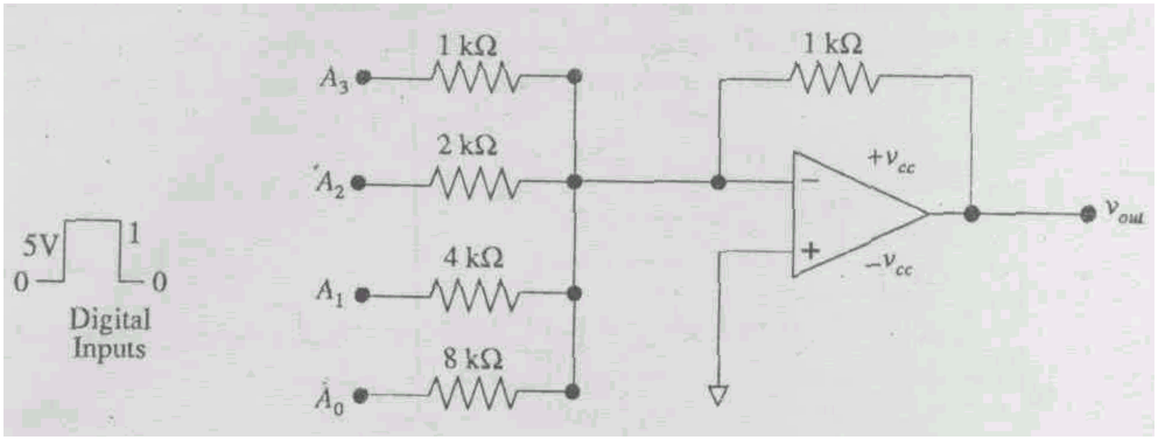


Fig. 3.21 A 4-bit DAC

Therefore, V_{out} for 0001 or LSB would be one-eighth of 5V, i.e. 0.625V. And this is the step size of this converter. Sixteen levels of the V_{out} are shown in Table 3.7.

Table 12.7: Ideal values of V_{out} for a 4-bit DAC

A_3	A_2	A_1	A_0	V_{out}
0	0	0	0	0
0	0	0	1	- 0.625 LSB
0	0	1	0	- 1.250
0	0	1	1	-1.875
0	1	0	0	- 2.500
0	1	0	1	-3.125
0	1	1	0	-3.750
0	1	1	1	-4.375
1	0	0	0	-5.000
1	0	0	1	- 5.625
1	0	1	0	- 6.250
1	0	1	1	-6.875
1	1	0	0	- 7.500
1	1	0	1	-8.125
1	1	1	0	-8.750
1	1	1	1	- 9.375 MSB Full Scale

These values are ideal values. However, the actual values may not be same. There may be some error due to fluctuations in the voltages or inaccurate resistors. The error in a DAC is specified by a term called full scale error which is the maximum deviation of the DAC's output from its expected ideal value expressed as the percentage of the full scale (FS). Let us say that a DAC has an error of + 0.01 % FS in the example considered above. It means that error is 0.01 % of 9.375V, i.e., + 0.9375 mV.

SAQ 6
What are the weights of each input bit of Fig. 3.21?

Example 3.6
If in the DAC circuit of Fig. 3.21, R_F is reduced to half, i.e. $500\,\Omega$, then what will V_{out} be for 1001?

Solution
The MSB passes with gain 0.5. Therefore, its weight is reduced to half of the previous case. That is, it is now 2.5V. Thus each input weight is the half of the previous case, i.e., 1.25, 0.625 and 0.312V. The V_{out} for 1001 is

$$2.5 + 0 + 0 + 0.312 = 2.812 \text{ V}$$

3.5.2 Analogue-to-Digital Converter

The circuit of a counter type (or digital ramp) Analogue-to-Digital Converter (ADC) is shown in Fig. 3.22. It consists of an op amp as a comparator, a DAC, counter and a 3-input AND gate. The functioning of this type of ADC is as follows:

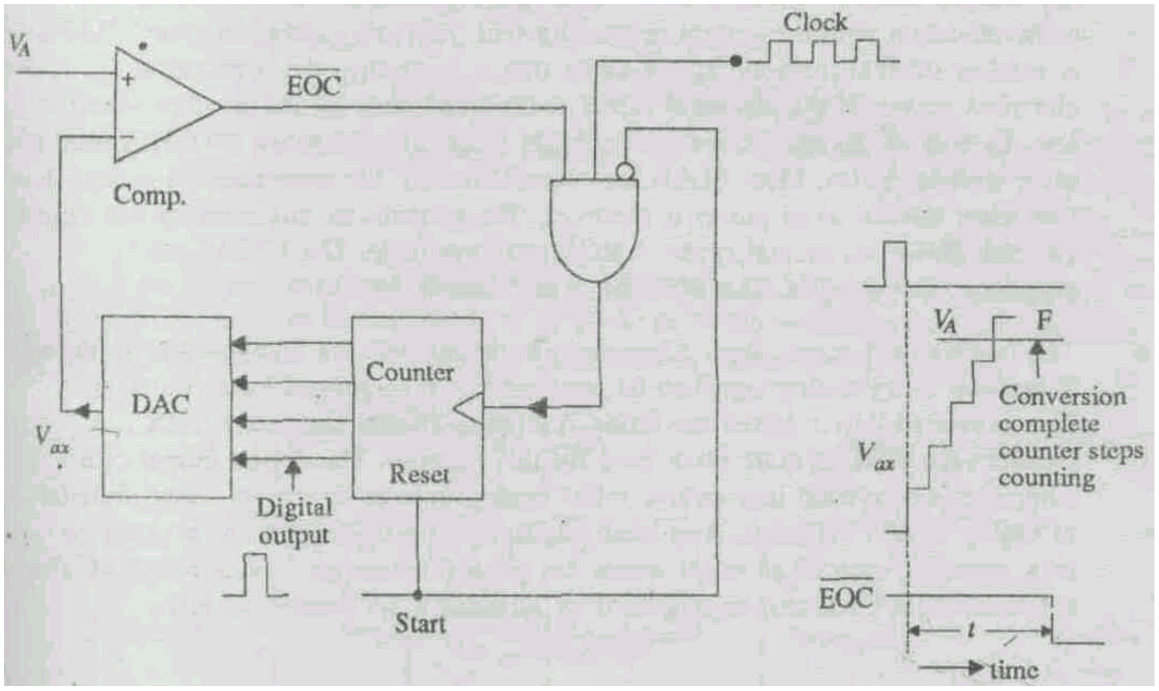


Fig. 3.22 Counter type ADC

Apply start pulse, i.e. make START input equal to 1. This resets the counter to 0 output. With 1 at START input, the AND gate is inhibited which does not allow the CLK from passing through the AND gate. The counter output is the input to the DAC. With counter reset, the DAC output V_{ax} = 0. V_A is the analogue input to be converted into its digital equivalent. Since $V_{ax} < V_A$, the op amp comparator output EOC is HIGH, i.e., 1. When the start pulse returns to 0, the AND gate allows the CLK to pass through and the CLK reaches the counter which starts counting. As the counter advances, the DAC output V_{ax} advances step by step as shown in the figure. When V_{ax} reaches a step that exceeds V_A, EOC goes low, i.e., 0, disabling the AND gate. Therefore, the CLK cannot pass through and the counter stops advancing further. The conversion of analogue input into its digital equivalent is complete. The contents of the counter are the digital representative of V_A. The counter holds the output until the next start pulse to initiate a new conversion is supplied.

3.6 SUMMARY

A register is a combination of D flip-flops and stores as many bits as the number of flip-flops. The register is the most commonly used memory device. The controlled register can retain the contents of the register as long as we want. It can shift the contents towards left (shift left register) or right (shift right register). The data can be supplied to and the output can be obtained from the register either serially or parallel fashion. The registers are, sometimes, classified as serial-in serial-out, serial-in-parallel-out, parallel-in-parallel-out.

A counter is like a register made of JK flip-flops and counts the number of clock pulses arriving at CLK input of the counter. The CLK input to asynchronous counter or ripple counter is given to the first flip-flop. The CLK signal to the other flip-flops is the output of the preceding flip-flop. Therefore, it is very slow because of high propagation delay. In synchronous counter, the CLK input is supplied to all the flip-flops simultaneously. The synchronous counter is fast and there is only one propagation delay.

The Modulus of a counter is the number of states of the output and it is 2^n, where n is the number of flip-flops used. A 4-bit counter having four flip-flops has a modulus of 16. A counter can be designed for a particular modulus. Mod 10 or decade counter has ten states of the output, i.e. it counts from 0 to 9 (or 0000 to 1001).

A memory is a device made up of several registers. The contents of each register can be read or new contents can be stored in it. The computer or a digital system has internal memory which is used while entering data, etc. and is in constant communication with the central processing unit. Random Access Memory (RAM) is used as internal memory. It is volatile, that is, it requires the application of electrical power. If the electrical power is removed, then all information stored is lost. Its cost of storage per bit is very high. Read Only Memory (ROM) is used as mass storage device. Most ROMs are nonvolatile, i.e., the information stored is not lost when the electrical power is removed. The contents of this memory can only be read. However, several types of ROMs are available. The PROM can be programmed only once. The EPROM is an Erasable-PROM.
The input to and output from a computer is digital. But the whole world is analogue. Therefore, every analogue signal to be processed or manipulated by a computer is first converted into a digital one using Analogue-to-Digital Converter (ADC). A counter type ADC is quite often used for this purpose. The digital output of a computer is converted into an equivalent analogue voltage or current using a digital-to-analogue (DAC) converter. The basic circuit of an ADC is like an inverting op amp amplifier used as an adder where the input resistors are binary weighted. The resolution of a DAC can be increased by increasing the number of bits.

3.7 TERMINAL QUESTIONS

(1) Design a Mod 12 counter.

(2) A computer X has memory $1\text{M}\times 8$ and computer Y has memory $500\text{K}\times 16$. What are the word sizes of the memories of the two computers? Which of the two computers can store more bits?

(3) A 4-bit DAC produces an output of 7V for 1110. What is the smallest change in its output voltage? Find the output voltage for 1001.

(4) What is the largest value of output voltage from an 8-bit DAC that produces 1 V for a digital input of 00110010?

(5) If the values of R_1s in the DAC circuit of Fig. 3.21 are reduced to half, then (a) what is the resolution, and (b) what is the output voltage for 1101?

3.8 SOLUTIONS AND ANSWERS

SAQs

1. The third flip-flop divides the CLK frequency by 8. Therefore, the output frequency will be 12.5 kHz.
2. The Mod of a counter is 2^n, where n is the number of flip-flops used. Therefore, the Mod of a counter consisting of six flip-flops is $2^6 = 64$.

3. (a) The size of the word is 8-bit.

(b) The total number of bits that the memory stores is $32\times1024\times8$ = 262,144 bits.

4. $8\times0.25 + 4\times0.25 + 2\times0.25 + 0 = 2 + 1 + 0.5$

$$= 3.5 \text{ V}$$

5. % resolution = $\dfrac{0.5mV}{15.5mV}\times100\% = 3.23\%$

6. The MSB passes with gain = 1, and so its weight is 5V. Thus,

MSB	- 5V
2nd MSB	- 2.5V
3rd MSB	- 1.25V
4th MSB, i.e., LSB	- 0.625V

TQs

(1) A Mod 12 counter counts from 0000 to 1011. When 1100 appears the counter should reset. Therefore, a circuit is to be made which will clear the flip-flops when 1100 appears. The circuit for Mod 12 counter is shown in Fig. 3.23.

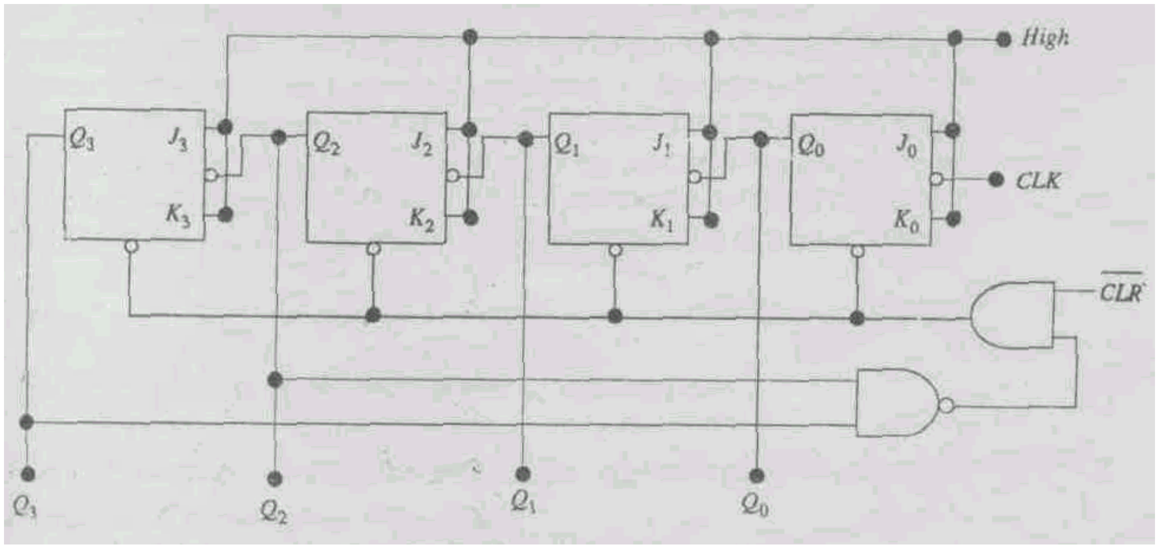


Fig. 3.23: The Mod 12 counter.

(2) The word size of the computer X is 8 bits and that of computer Y is 16 bits.

For computer X

$1M\times8 = 1\times1{,}048{,}576\times8$ = 8,388,608 bits

For computer Y

$500\times1024\times16$ = 8, 192,000 bits

Therefore, computer X can store more bits.

(3) Follow Example 3.4. Smallest change in the output voltage is 0.5V. The output voltage for 1001 is 4.5V.

(4) $00110010_2 = 50_{10}$
$1\ \text{V} = \text{K}\times 50$
Therefore, K = 20 mV, The largest output will occur for $11111111_2 = 255_{10}$.

$$V_{out(\max)} = 20\ \text{mV} \times 255$$
$$= 5.10\text{V}$$

5) Follow the example 3.6. The resolution is 0.312V and the output voltage for 1101 is 4.062V.

UNIT 13 ELECTRONIC INSTRUMENTS

Structure

4.1 INTRODUCTION

As you are aware, in practical applications of a system, we typically encounter a situation as shown in Fig. 4.1. In the study of Signal Processing Circuits we assume that we have the desired electrical signal and do not worry about input sensor. The performance of a circuit *is* displayed on an instrument which can be seen by us. The signals of different shapes and time duration are provided by signal generators and a general purpose oscilloscope is used to display them.

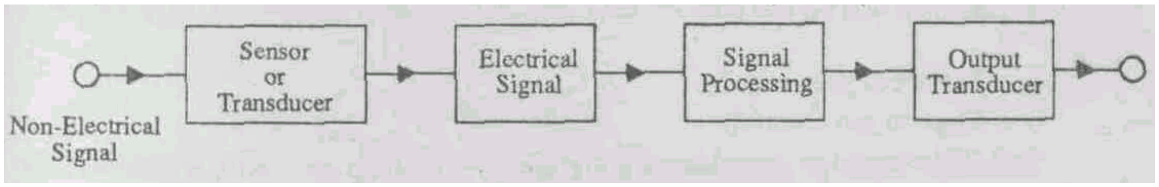


Fig. 4.1 Situation encountered in practical applications of a system

We know that all circuits are made up of some active components like transistors, FET, MOSFET etc. and passive components like resistors, inductors & capacitors. To measure values of passive components, we use the multimeter, bridges (for L&C) etc. In this unit, we will be studying Electronic Voltmeter (EVM), which is a more sensitive and hence, more accurate instrument as compared to the Multimeter. EVM can also be used for very low current measurements by using a standard resistance. The power consumed by these circuits is of vital importance and hence we will also study the power meter. While studying the construction of power meter, we will see that the necessary torque required for meter movement is generated with the help of interaction of magnetic field and current and hence we will also discuss the art of measurement of magnetic field.

Objectives

After going through this unit, you will be able to understand

- basic construction, working and some of the applications of Oscilloscope,
- generation of various shapes of signals,
- accurate measurement of voltage using Electronic voltmeter,
- measurement of power, and
- measurement of magnetic field.

4.2 CATHODE RAY OSCILLOSCOPE

The cathode ray oscilloscope, generally referred to as the oscilloscope or simple "scope", is probably the most versatile electrical measuring instrument available. Some of the electrical parameters that can be observed with the oscilloscope are ac or dc voltage, indirect measurement of ac or dc current, time, phase relationships, frequency, and a wide range of waveform evaluations such as rise time, fall time, ringing, and overshoot. The oscilloscope consists of the following major subsystems:

- Cathode ray tube or CRT
- Vertical amplifier
- Horizontal amplifier
- Sweep generator
- Trigger circuit
- Associated power supplies

The heart of the instrument is the cathode ray tube. The remaining sub-systems are necessary for signal conditioning so that a visual representation of the input signal will be displayed on the face of the CRT.

4.2.1 Cathode Ray Tube (CRT)

The cathode ray tube used in an oscilloscope is very similar to the picture tube in a television set. A cross sectional representation of a CRT is shown in Fig.13.2. Major components of a general purpose CRT are:

- Evacuated glass envelope
- Electron gun assembly
- Deflection plate assembly
- Phosphor coated screen

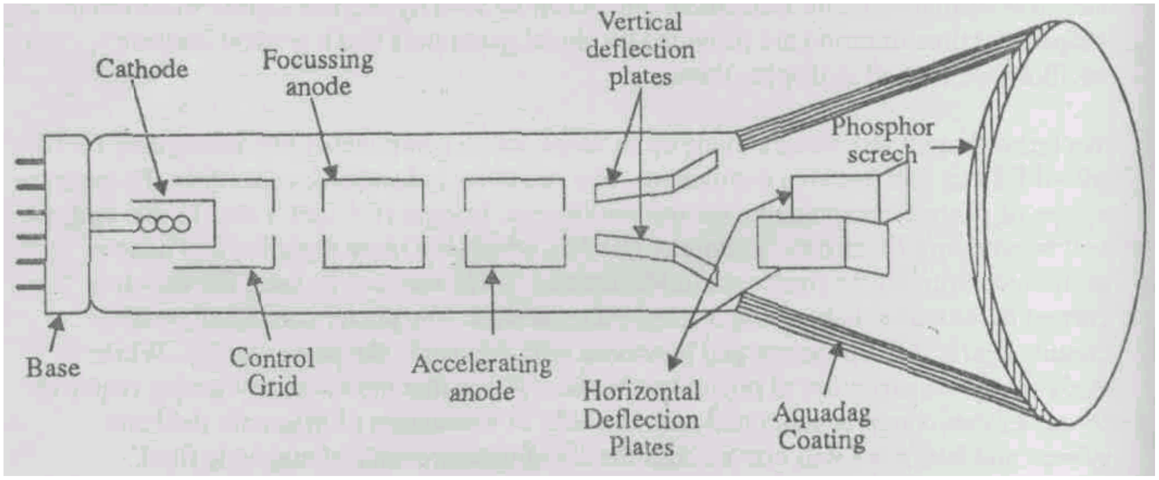


Fig. 4.2 Cathode ray tube with major components identified

The glass envelope is evacuated to a fairly high vacuum to permit the electron beam to traverse the tube easily. Most laboratory quality oscilloscopes use a CRT, which has a circular screen approximately 5 inch in diameter. All electrical connections except the high-voltage connection are made through the base of the CRT.

The electron gun assembly consists of an indirectly heated cathode and the necessary heater, a control grid, focussing anode and accelerating anode. The purpose of the electron gun assembly is to provide a source of electrons, converged and focussed into a well-defined beam, which is accelerated towards the fluorescent screen. The electrons that make up the beam are given off by thermionic emission from the heated cathode. The cathode is surrounded by a cylindrical cap that is at a negative potential. This acts as a control grid. Because the control grid is at negative potential, electrons are repelled away from the cylinder walls and, therefore, stream through the hole where they move into the electric field of the focussing and accelerating anodes. The magnitude of the accelerating field is given by

$$E = \frac{V_a}{d}$$

where, V_a = accelerating anode voltage and d = distance between the cathode and the second anode, measured in meters. When electrons enter the electric field, which is assumed to be of uniform intensity, a force will be exerted on the electrons that will accelerate them along the axis of the tube. The magnitude of the force is given by

$$F = EQ = ma \;\Rightarrow\; a = \frac{EQ}{m}$$

where E = electric field intensity and Q = electronic charge = 1.6×10^{-19} C, m = mass of electron, a = acceleration produced due to electric field. Using the expression for electric field in above equation, we obtain

$$a = \frac{V_a Q}{dm}$$

During the period of acceleration, the electrons are gaining kinetic energy as they gain velocities. If v is the velocity acquired then,

$$\frac{1}{2}mv^2 = V_a Q \quad \Rightarrow \quad v = \sqrt{\frac{2V_a Q}{m}}$$

After the electrons leave the electron gun assembly at a speed given by the above-mentioned equation, they enter and pass through a region controlled by the deflection plates. One pair of plates control the vertical motion of the beam while the other pair controls the longitudinal component of the electron velocity. The deflection plates are described by two geometric parameters of length (L) of the plates and the plate separation (d). The deflecting action of the plates is dependent on the intensity of the electric field (E_d) between the plates given by

$$E_d = \frac{V_d}{d}$$

where V_d = magnitude of the deflecting voltage. This field will exert a force = E_dQ on the electrons, deviating the beam from a straight line trajectory.

$$F_d = E_dQ = \frac{V_d}{d}Q = ma_y$$

$\Rightarrow$ $\qquad a_y = \dfrac{V_dQ}{md}$ = acceleration along the y -axis

It can be shown that the lateral distance travelled by an electron is given by

$$h = \frac{V_dQt^2}{2dm}$$

where t = time required for electrons to pass through the plates is given by

$$t = \frac{L}{v}$$

Here, v is the velocity of electrons when it comes out of electron gun assembly. Combining these equations, we get

Combining these equations, we get

$$h = \frac{L^2V_d}{4V_ad}$$

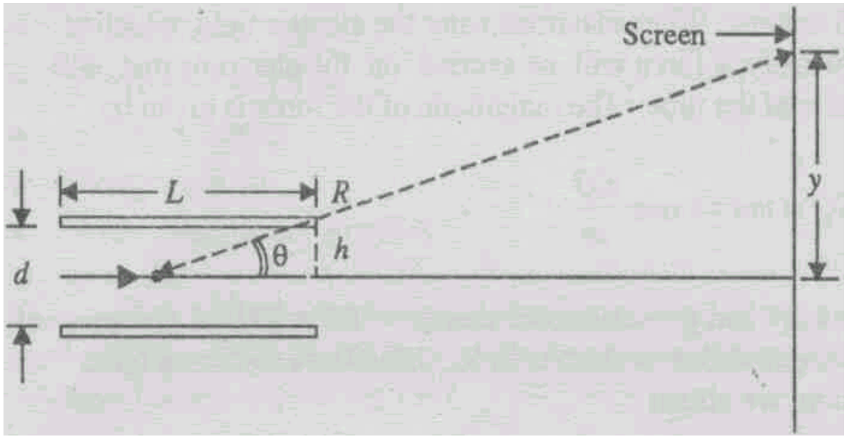


Fig. 4.3 Deflection of electron beam in CRT

From Fig. 4.3,

$$\theta = \frac{h}{L/2} = \frac{2h}{L} = \frac{y}{R}$$

$$\Rightarrow \quad y = \frac{2hR}{L}$$

$$= \frac{RLV_d}{2V_a d}$$

$$\Rightarrow \quad \frac{V_d}{y} = \frac{2V_a f}{RL}$$

The term $\frac{V_d}{y}$ is referred to as "deflection sensitivity" and is defined as the voltage required per unit deflection. When the electron beam strikes the phosphor-coated face of the CRT, a spot of light is produced due to "fluorescence" as phosphor is a florescent material. The high velocity electrons that strike the phosphor-coated face of the CRT are either repelled by the collision or cause secondary emission of electrons to maintain electrical the equilibrium of the screen. To provide a return path to ground for these electrons, the inside surface of the CRT is coated with graphite called "aquadag."

4.2.2 The Basic Oscilloscope

The CRT and the associated controls for accelerating, deflecting and focussing the electron beam, permit us to obtain a lighted spot on the screen. To be of practical use as a measuring instrument, we must connect additional electronic circuitry to the CRT to provide a means of very fast deflection and control of the electron beam. The purpose of the electronic circuits is to cause the beam to trace on the CRT screen a reproduction of the signal we apply to the input terminals of the oscilloscope. A block diagram of a basic oscilloscope is shown in Fig. 4.4(a).

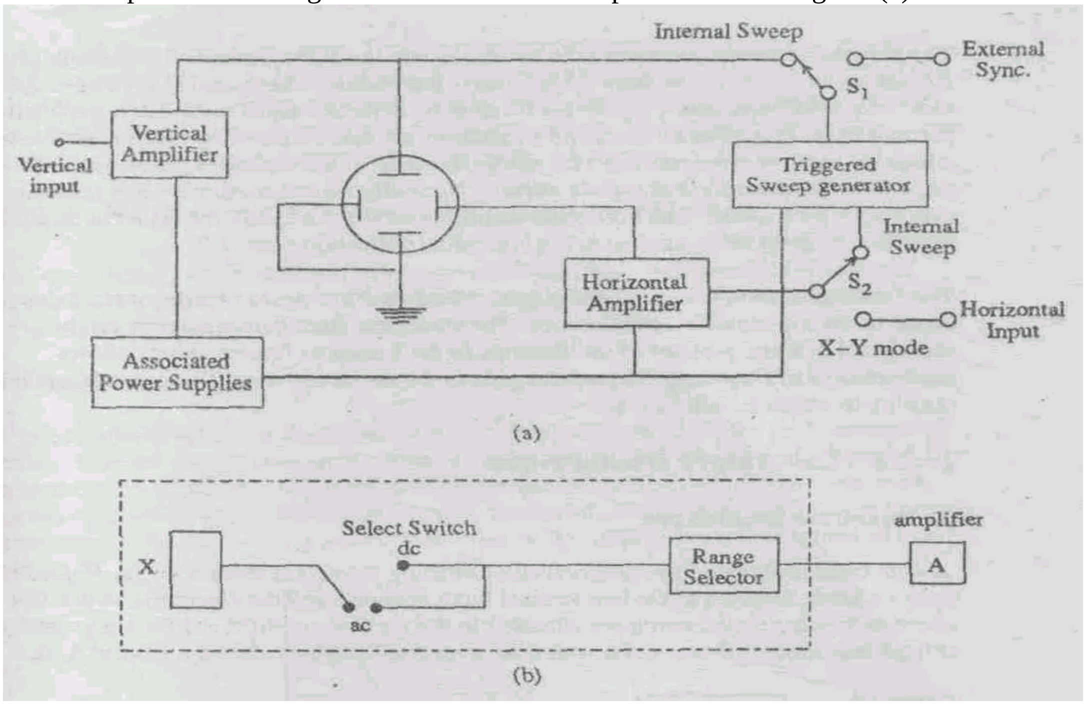


Fig. 4.4 (a) Block diagram of a basic cathode ray oscilloscope (b) Input to amplifier of vertical Plate

A signal to be displayed on the CRT screen is applied to the vertical input terminal where it is fed into the vertical amplifier. The signal is amplified and applied to the vertical deflection plates, which causes the beam to be deflected in the vertical plane.

Input to the Amplifier of Vertical Plate

The external signal is applied to the terminal marked x-input as shown in Fig. 4.4 (b). The select switch is put on the position ac or dc depending on the signal we are measuring. The input amplifier to the y-plate is normally calibrated for a standard input range of amplifier A. For higher voltage measurement we have a range selector, which basically attenuates the signal to the desired input at A (we have already studied filters and attenuators).

Input to the Amplifier of Horizontal Plate

As can be seen in Fig. 4.4 (a), the output of the vertical amplifier is connected to the internal sync position of switch S_1. With the switch set to internal sync, as it is for normal operation of the oscilloscope, the output of the vertical amplifier is applied to the sweep generator. The input voltage waveform irrespective of shape at a particular value triggers the switch, which creates pulses and these pulses are then fed to the Sawtooth generator circuit, which provides the ramp signal. This signal triggers the sweep generator as shown in Fig. 4.5.

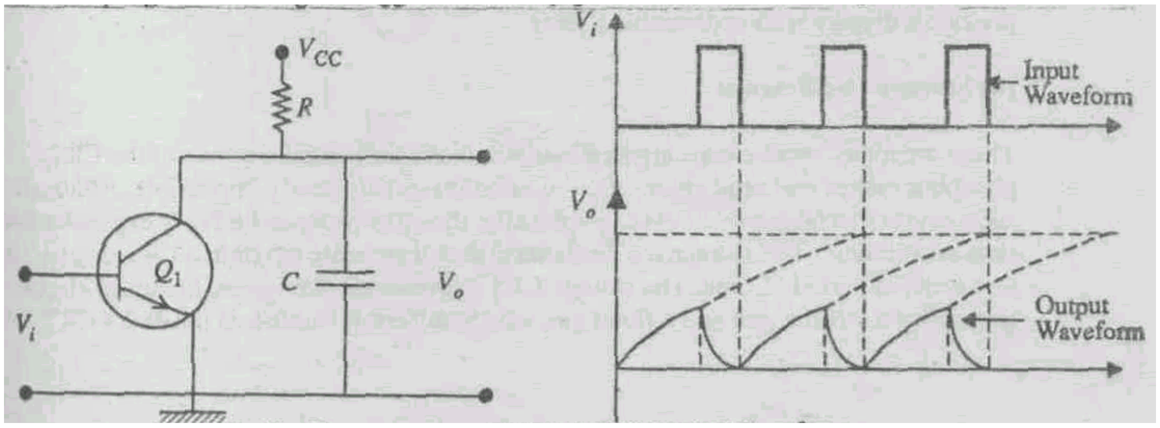


Fig. 4.5 Simple sawtooth generator and associated waveforms

The purpose of the sweep generator is to develop a voltage at the horizontal deflection plate that increases linearly with time. This linearly increasing voltage, called a 'ramp-stage' or a 'Sawtooth waveform', causes the beam to be deflected equal distances horizontally per unit time. The pulse for sawtooth generation can also be given by an external source to which the input is synchronized (the sawtooth signal at the x-plate is generated at the same time the wave form at y-plate starts). Normally we use an oscilloscope in internal synchronization mode. The horizontal amplifier serves to amplify the signal at its input prior to the signal being applied to the horizontal deflection plate.

The function of switch S_2 is to either generate sawtooth wave in the x-plate, or put a direct signal to the x-plate of the oscilloscope. The sine wave from two oscillators can be introduced in the x- and the y-plates of oscilloscope to get Lissajous figures, which allows measurement of frequency. The input signal to the horizontal amplifier depends on the position to which switch S_2 is set.

4.2.3 Laboratory Oscilloscopes

(i) Dual-trace Oscilloscope

A dual trace is obtained by electronically switching the single electron beam. Fig. 4.6 shows a block diagram of the two vertical input channels and the electronic switch that alternately connects the two input channels to the vertical amplifier. There are generally at least four modes of operation with dual-trace oscilloscopes; they are labelled A, B, alternate, and chopped.

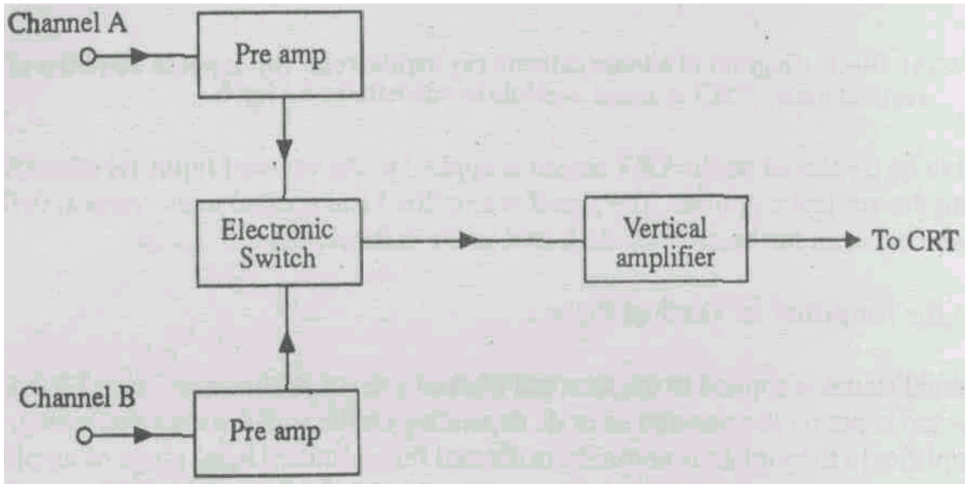


Fig. 4.6 Block diagram of the input channels of a dual trace oscilloscope

When set to A or B, only the input at that channel is displayed. In the alternate mode the inputs are displayed on alternate traces. Since the switching rate is synchronised with the sweep generator, switching occurs at the same rate as the output of the sweep generator. The "alternate mode" of operation is generally preferred when displaying relatively high-frequency signals. In the "chopped mode," electronic switching occurs at a rate completely independent of the sweep rate, and therefore each display has portions missing during which time the other signals is being displayed. The chopped mode is normally used at low sweep rates when the alternate mode would provide a display with appreciable flicker.

(ii) Storage Oscilloscope

There are many oscilloscope applications where the limited persistence of the CRT phosphor makes real time observation of one-time events nearly impossible. Although such events can he recorded photographically, this may prove to be fairly expensive and time consuming. The storage oscilloscope makes it possible to retain a CRT display for an extended period of time. The storage CRT uses two electron guns, the usual electron gun called a writing gun and a flood gun, which uniformly bombards the entire CRT screen with low-energy electrons. The phosphor particles struck by these low energy electrons takes on a fairly low-level charge; however, unenergised particles remain in a no-change condition. When a trace is to be recorded, the writing gun is turned on and high-energy electrons strike the screen, forming an image. The screen is erased by grounding the phosphor screen, which removes excess charge.

4.2.4 Measurement of Voltage, Current and Time

The range of applications of oscilloscopes varies from basic voltage, time, frequency measurements and waveform observations to highly specialised applications in all areas of science, engineering and technology.

Voltage measurements

The most direct voltage measurement made with an oscilloscope is the peak-to-peak value. The rms value of the voltage can easily be calculated from the peak-to-peak measurements if desired. To arrive at a voltage value from the CRT display, one must observe the setting of the vertical attenuator, expressed in volts/div, and the peak-to-peak deflection of the beam. The peak-to-peak value of the voltage is then computed as (see Fig. 4.7)

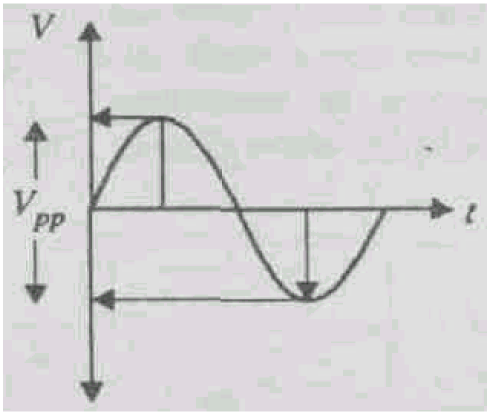


Fig. 4.7 Voltage measurement

$$V_{pp} = \left(\frac{\text{volts}}{\text{div}}\right) \times \text{no.of division}$$

This can be easily explained with the example 4.1.

Example 4.1

Let the waveform shown in Fig. 4.8 be observed on the screen of an oscilloscope. If the vertical attenuator is set to 0.5 volts/div, find the peak-to-peak amplitude of the signal.

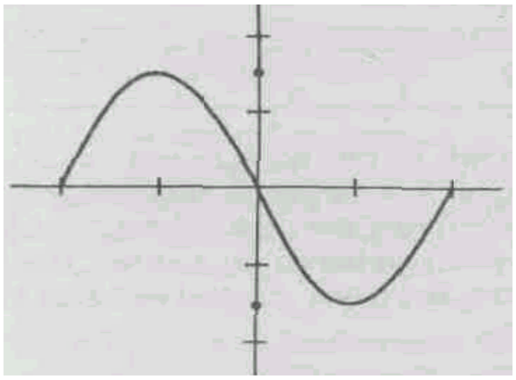


Fig. 4.8

Solution:

$$V_{pp} = \left(\frac{\text{volts}}{\text{div}}\right) \times \text{no.of division}$$

$$= \frac{0.5V}{\text{div}} \times 3\,\text{div}$$

Period and Frequency Measurements

The period and frequency of periodic signals are easily measured with the oscilloscope. The waveform must be displayed in such a manner that one complete cycle is displayed on the CRT screen. Accuracy is generally improved if the single cycle displayed fills as much of the horizontal distance across the screen as possible. The period is calculated as follows:

$$T = \left(\frac{\text{time}}{\text{div}}\right)\left(\frac{\text{no. of div}}{\text{cycle}}\right)$$

The frequency is then computed as the reciprocal of the period.

4.2.5 Digital and Storage Oscilloscope

The storage oscilloscopes described in the earlier section are quite expensive and are now being replaced by digital oscilloscopes. In these oscilloscope the signal on the screen is sampled and digitised. The amplitude and time base per cm are displayed in numbers at a corner of the screen. The digitised signal can be put into a memory (like computer memory) and recalled (D/A converter) to display when desired. Thus they also serve as storage oscilloscopes

.

SAQ 1

Draw a pictorial representation of a general purpose CRT and label the components by name.

SAQ 2

Describe the basic principle of operation of dual-trace/storage oscilloscope.

SAQ 3

If the time/div control is set to 2 μ s/div and the displayed signal covers 4 div on the horizontal scale of the CRT screen, determine the frequency of the signal.

SAQ 4

Explain the principle of the Digital Oscilloscope.

4.3 SIGNAL GENERATORS

A signal source is a vital component of a test set up. Signal sources provide a variety of waveforms for testing electronic circuits, usually at low power. A function generator is an instrument that provides a variety of output waveforms over a wide range of frequency. The most common output waveforms are sine, pulse, triangular and ramp**.** The frequency range generally extends from a fraction of a Hertz to at least several hundred kilohertz. The different wave shapes are given in Fig. 4.9.

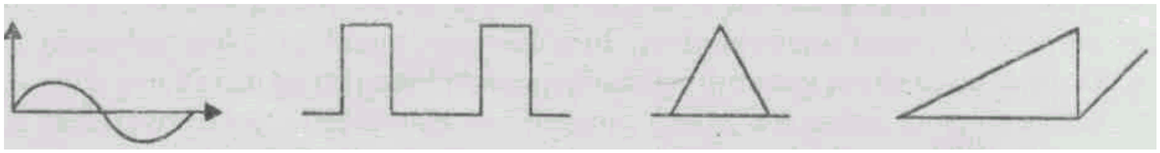


Fig. 4.9 Different shapes of wave form, (a) Sinusoidal (b) Rectangular (e) Triangular (d) Ramp

Definition of rise time (T_r): The time taken by the signal to rise from 10% to 90% of the maximum value of the signal is called rise time.

Fall time (T_f): The time taken by the signal to fall from 90% to 10% of the maximum value of the signal is called fall time.

There are several circuits to provide such waveforms individually. For example, you are aware that an LC oscillator can provide sine wave while a multivibrator can provide pulses. However, by starting from any particular waveform we can, with proper circuitry, generate other waveforms. In a function generator, a simple instrument is capable of providing different types of waveform. The most commonly used circuit is described below.

Function generator:
The primary waveform in the circuit shown is a square wave. This is because some square wave generator circuits offer significantly better amplitude and frequency stability characteristics with simpler circuits than sine wave generating circuits.

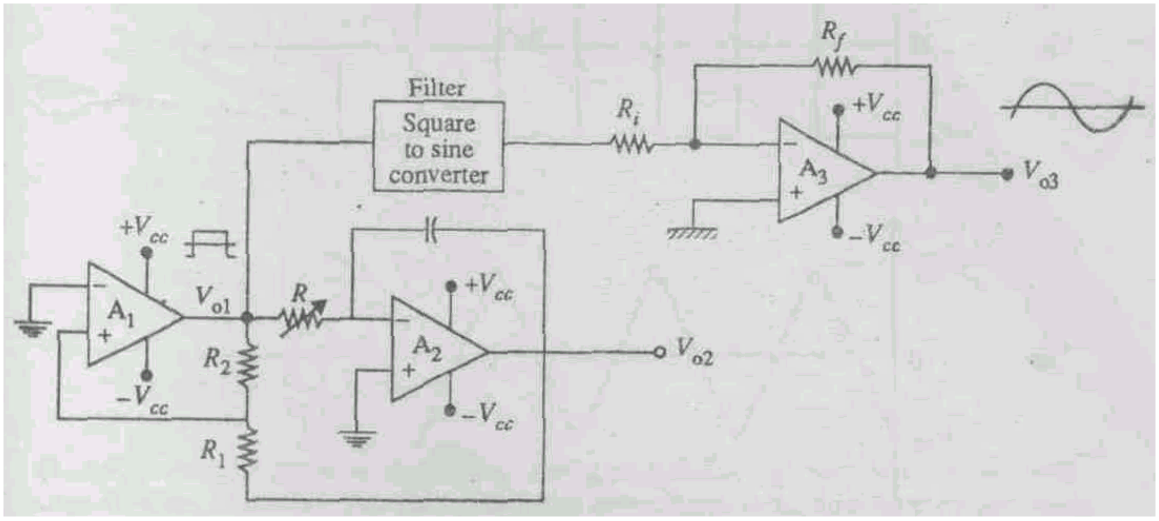


Fig. 4.10 Circuit of a basic function generator

Working of the Circuit
The first stage A_1, which is a voltage comparator, generates a square wave output. The output of A_1 is driven to saturation; therefore the square wave is either at $+V_{CC}$ or $-V_{CC}$ as shown in Fig. 4.11. The second stage, A_2, is an integrator that generates a triangular output at V_{02} as discussed later.

The square wave is also applied to a square-to-sine wave converter that filters out the odd harmonics making up the square-wave while passing only the fundamental sine wave. You will learn later that the square waves are produced by the combination of several sine waves, and by differentiation and integration we can convert pulses to triangular waves and vice-versa.

The operation of the circuit can be analysed by starting at the output of the comparator, which is either $+V_{CC}$ or $-V_{CC}$. Consider V_{01} to be at $-V_{CC}$. The voltage V_{01} will remain at $-V_{CC}$ until the voltage at the inverting input of A$_1$ exceeds the voltage at the non-inverting input, which in this case is at zero volt. The non-inverting input voltage, V_x, is due, in part, to the voltage V_{01} and, in part, to the voltage V_{02} and is given by

$$V_x = -V_{CC}\left(\frac{R_1}{R_1+R_2}\right) + V_{02}\left(\frac{R_2}{R_1+R_2}\right)$$

The output V_{01} changes state when V_x = 0. Therefore

$$0 = -V_{CC}\left(\frac{R_1}{R_1+R_2}\right) + V_{02}\left(\frac{R_2}{R_1+R_2}\right)$$

$$\Rightarrow \qquad V_{CC}R_1 = V_{02}R_2$$

$$\Rightarrow \qquad V_{02} = V_{CC}\left(\frac{R_1}{R_2}\right)$$

The above expression determines the maximum amplitude of the triangular output, V_{02}. When output V_{02} reaches the peak value, the output of the comparator changes states and the triangular wave begins to decrease linearly. The waveforms at V_{01}, V_{02} and V_x are shown in Fig. 4.11 for the case where $R_1 = R_2$.

The frequency of the circuit is controlled by the *RC* time constant of the integrator. To obtain an expression for the frequency, we begin with the expression relating capacitor current:

$$q = i_c t$$

$$\Rightarrow \qquad dq = i_c dt \quad \Rightarrow \quad i_c = \frac{dq}{dt}$$

Also, $\quad q = CV_{02}$

$$\therefore \qquad i_c = \frac{d}{dt}(CV_{02}) = C\frac{dV_{02}}{dt}$$

Since the input resistance of the operational amplifier is very high, the current through resistor *R* is approximately equal to the charging current of the capacitor, therefore, we can write

$$i_R \approx i_c = C\frac{dV_{02}}{dt}$$

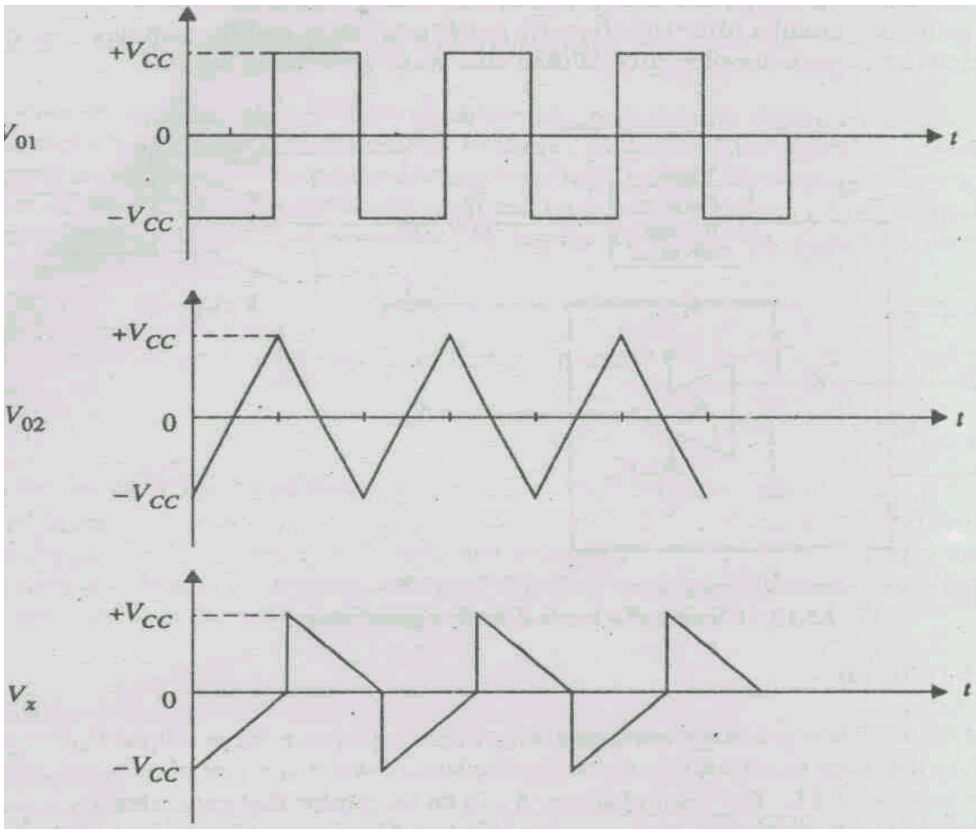


Fig. 4.11 Output waveforms for function generator

Also, since the voltage gain of the operational amplifier is very high, the voltage at the input of the amplifier is very nearly zero, therefore,

$$i_R = \frac{V_{01} - 0}{R} = C\frac{dV_{02}}{dt}$$

$$\Rightarrow \quad dV_{02} = \frac{1}{RC}V_{01}dt$$

Integrating both sides,

$$\int dV_{02} = \frac{1}{RC}\int V_{01}dt = \frac{V_{01}}{RC} \times t$$

$$\Rightarrow \quad V_{02} = \frac{V_{01} \times t}{RC}$$

We know,

$$V_{02} = V_{CC}\left(\frac{R_1}{R_2}\right)$$

$$\therefore \qquad V_{CC}\frac{R_1}{R_2} = \frac{V_{01}t}{RC}$$

$$\Rightarrow \qquad t \qquad = RC\frac{R_1}{R_2} \text{ as } V_{01} = V_{CC}$$

The above equation has been deduced assuming no initial charge and therefore no initial voltage on the capacitor. Therefore, the time t given above is the time for the capacitor to change from 0V until switching occurs, which is 1/4 cycle. Since $t = T/4$.

Pulse Generators
Pulse generators are instruments that produce a rectangular waveform similar to a square wave but of different duty cycle. Duty cycle is defined as the ratio of the pulse width to the pulse period, expressed in percent.

$$\text{Duty cycle} = \frac{\text{Pulse width}}{\text{Pulse period}} \times 100$$

The duty cycle of a square wave is 50% whereas the duty cycle of a pulse is generally from approximately 5 to 95%.

The output of a stable multivibrator is a square wave. The duty cycle of the square wave can be varied by changing values of R and C.

SAQ 5
Describe the function generator.

SAQ6
What is difference between a square wave and a pulse?

SAQ7
Compute the frequency and the peak amplitude of the triangular output of the circuit shown in Fig 4.12.

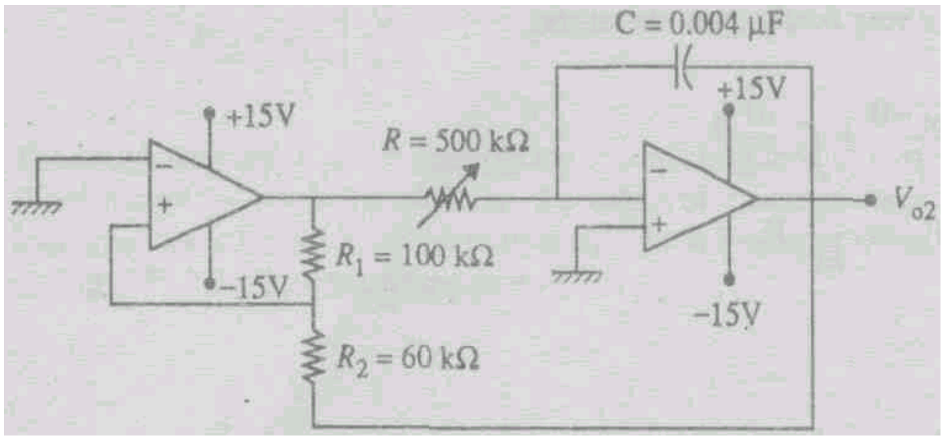


Fig. 4.12

4.4 ELECTRONIC VOLTMETER

The volt-ohm-milliameter (VOM) is a rugged and accurate instrument, but suffers from certain disadvantages. The main problem is that it lacks both sensitivity and high input resistance. (A sensitivity of 20,000 Ω/V with a 0 to 0.5 V range has an input impedance of only $0.5 \times 20{,}000$ = 10 KΩ. The electronic voltmeter (EVM), on the other hand, can have an input resistance ranging from 10 to 100 MΩ, and the input resistance will remain constant over all ranges instead of being different on each range as in the VOM. The EVM presents less loading to a circuit under test than the VOM. The original EVMs used vacuum tubes, so they were called vacuum tube voltmeters (VTVM). With the introduction of the transistor and other semiconductor devices, vacuum tubes are no longer used in these instruments. We will discuss below in detail the differential amplifier type of EVM.

The Differential-Amplifier type of EVM

The field effect transistors (FET) can be used to increase the input resistance of a dc voltmeter. Fig. 4.13 shows the schematic of a difference amplifier using field-effect transistors.

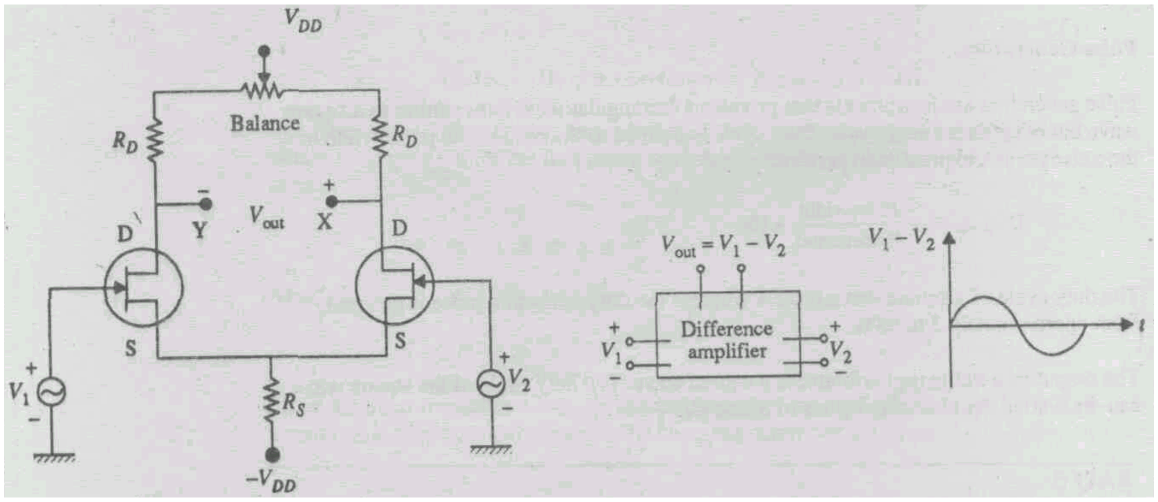


Fig. 4.13 Difference amplifier with balance adjustment

This circuit also applies to a difference amplifier with bipolar junction transistors (BJTs). The circuit shown here consists of two FETs that should he reasonably matched for current gain to ensure thermal stability of the circuit. Therefore, an increase in source current in one FET is offset by a corresponding decrease in the source current of the other FET, The two FETs form the lower arms of the bridge circuit. Drain resistors R_D together form the upper arms. The meter movement is connected across the drain terminals of the FETs, representing two opposite corners of the bridge.

The circuit is balanced when identical FETs are used so that for a zero input there is no current through the ammeter. If a positive dc voltage is applied to the gate of the left FET, a current will flow through the ammeter in the direction shown in Fig. 4.14.

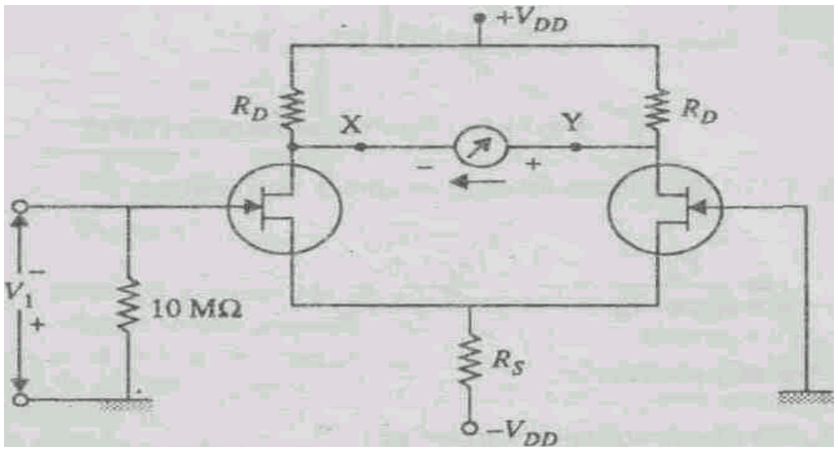


Fig. 4.14 The difference amplifier type EVM

The size of the current depends on the magnitude of the input voltage. By properly designing the circuit, the ammeter current will be directly proportional to the dc voltage across the input. Thus, the ammeter can be calibrated in volts to indicate the input voltage.

By using Thevenin's theorem, we can find the relation between the ammeter current and the input dc voltage, where the ammeter is considered as the load. To determine V_{th}, we remove the ammeter and the output voltage is the voltage gain of a single FET times the difference of V_1 and V_2. Since V_2 is zero, the output voltage under open circuit condition is

$$V_{out} = g_m\left(\frac{r_d R_D}{r_d + R_D}\right)V_1 = g_m(r_d \parallel R_D)V_1$$

where r_d is the ac drain resistance, g_m= transconductance. To find the Thevenin resistance at terminals *XY*, we first set V_1 and V_{DD} equal to zero. Under this condition, both the FETs have a resistance of r_d as shown in Fig. 4.15. Assuming R_S to be relatively large,

$$\begin{aligned} R_{th} &= 2r_d \parallel 2R_D = 2(r_d \parallel R_D) \\ &= 2\frac{r_d R_D}{r_d + R_D} \end{aligned}$$

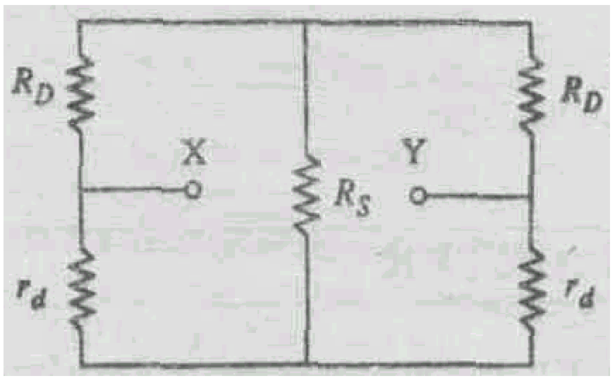


Fig. 4.15 Setting all voltages equal to zero to find R_{th} of EVM

The Thevenin equivalent circuit with ammeter connected as a load is shown in Fig. 4.16.

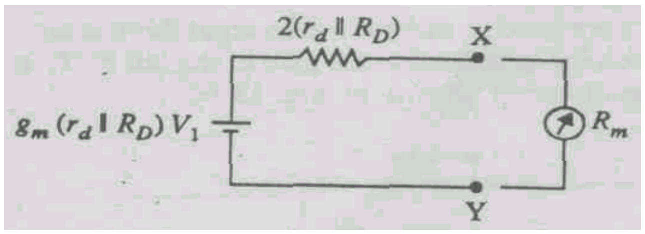


Fig. 4.16 Equivalent circuit of EVM

From Fig. 4.16, the current through ammeter is found as:

$$i = \frac{V_{out}}{R_{Th} + R_m} = \frac{g_m(r_d \parallel R_D)}{2(r_d \parallel R_D) + R_m} V_1$$

where R_m = meter resistance.

If $R_D << r_d$, the above equation simplifies to

$$i = \frac{g_m R_D}{2R_D + R_m} V_1$$

This equation relates ammeter current to the input dc voltage.

SAQ8
How does FET EVM differ from the VOM?

SAQ9
Give the circuit for the difference-amplifier type of EVM.

SAQ10
Given a difference amplifier type of FET voltmeter, find the ammeter current under the following conditions:

V_1	$= 1$ V	R_D	$= 10$ kΩ
r_d	$= 100$ kΩ	R_m	$= 50$ MΩ
g_m	$= 0.005$ Siemens		

4.5 POWERMETER

The wattmeter is an instrument used to measure the power or rate of consumption of electricity in a circuit in watts. The most commonly used powermeter is the Siemen's wattmeter shown in Fig. 4.17.

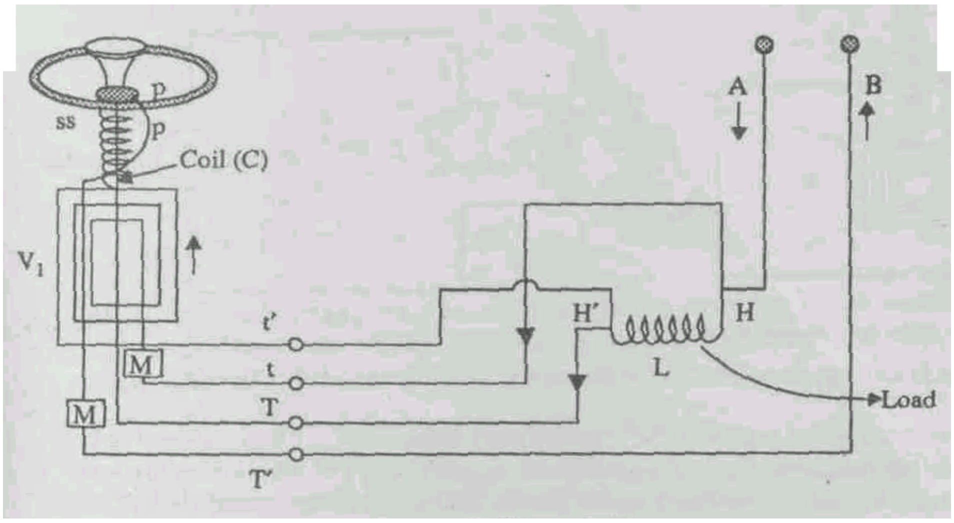


Fig. 4.17 Siemen's Wattmeter

The Siemen's Wattmeter is identical in principle with Siemen's electrodynamometer. It consists of two coils at right angles to each another. One coil C is movable and the other, V is fixed. The moving coil C is of low resistance and is inserted in the main circuit. The high resistance fixed coil V, is joined as a shunt (i.e., in parallel) to that part of the circuit for which the power consumption is required. In Fig. 4.17, this part is an electric lamp (L). On closing the circuit, the main current i passes through the moving coil and a small current, proportional to the voltage E across the lamp terminals, passes through V, The turning moment is proportional to the product of these two, i.e., proportional to $E\,i$ or the Watts used in L. When the moving coil is brought back to its normal position by turning the torsion head and its pointer through an angle say θ, the turning moment is balanced by the torsional moment. Since torsional moment is proportional to θ

$$E\,i \propto \theta$$

or Watt expended in L = $K\theta$

where the constant of proportionality K is a constant of the instrument and must be determined experimentally.

4.6 MAGNETIC FIELD METER

There are several techniques for the measurement of magnetic field. These are based on the change in the resistance of a material (magneto-resistance) under the application of magnetic field, or the voltage developed across a semiconductor under magnetic field (Hall effect). In this section we shall discuss a method which is based on electromagnetic induction or the voltage developed across a coil when flux changes through a coil. The change of flux can be produced by moving the coil across a magnetic field. This method is often called determination of magnetic field by search coil.

A fluxmeter, an important instrument for measuring magnetic field strengths, has the same principle as that of a ballistic galvanometer. A Fluxmeter consists of a moving coil suspended by a single silk fibre without torsion, the upper end of the fibre being connected to a fixed flat spring as shown in Fig. 4.18 (a).

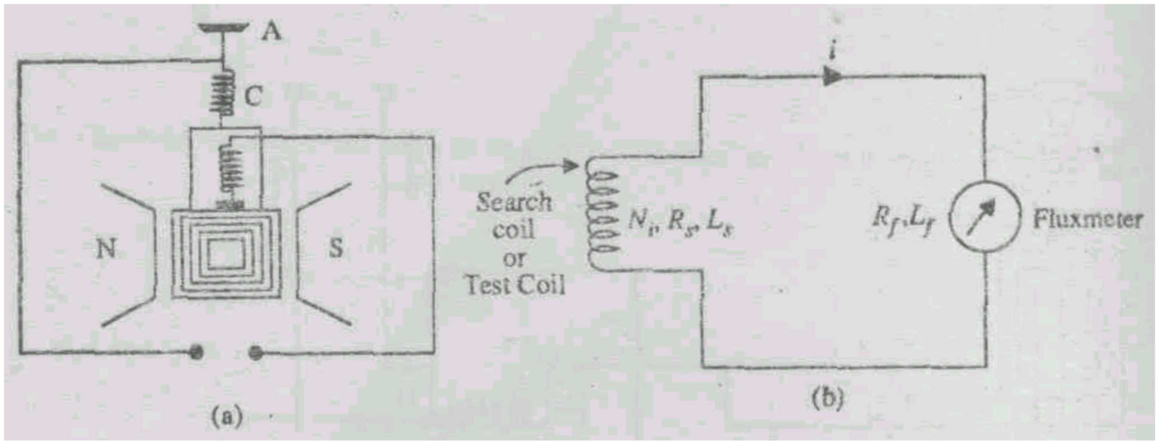


Fig 4.18 (a) Construction of Fluxmetcr (b) Fluxmeter in a circuit

The coil is connected to the terminal X,X through two spirals C,C of thin silvered coil and is suspended in magnetic field of a permanent magnet NS. For direct measurement of the flux, a search coil is provided, which can be connected to the terminals X,X as shown in Fig. 4.18 (b).

The expression for the change in magnetic flux of a fluxmeter can be derived as follows: Let,

R_f	= Resistance of fluxmeter
L_f	= Sell Inductance of fluxmeter
R_s	= Resistance of search coil
N_i	= No. of turns in search coil
L_s	= Self Inductance of search coil

The emf induced in the search coil is $-N_1\dfrac{d\phi}{dt}$, where $\dfrac{d\phi}{dt}$ is the rate of change of flux in the search coil and the emf in the fluxmeter coil is $G\dfrac{d\theta}{dt}$, where $\dfrac{d\theta}{dt}$ is the angular velocity of the fluxmeter coil and $G = NAB$ is a constant depending on the construction of the fluxmeter. In addition, the emf produced in the circuit due to self inductances is $(L_f + L_s)\frac{di}{dt}$ or $L\frac{di}{dt}$, where L is the total inductance of the circuit. The potential drop in the resistance is $(R_f + R_s)i$ or Ri, R being the total resistance of the circuit. Using Kirchoffs law, we get

$$-N_1\frac{d\phi}{dt} + G\frac{d\theta}{dt} + L\frac{di}{dt} + Ri = 0$$

$$\Rightarrow \qquad N_1\frac{d\phi}{dt} = G\frac{d\theta}{dt} + L\frac{di}{dt} + Ri$$

In practical applications, the potential drop in resistances $(= Ri)$ is small and can be neglected in comparisons to other terms, giving

$$N_1\frac{d\phi}{dt} = G\frac{d\theta}{dt} + L\frac{di}{dt}$$

Integrating over time t, during which the flux change occurs,

$$N_1\int_0^t \frac{d\phi}{dt} = G\int_0^t \frac{d\theta}{dt} + L\int_0^t \frac{di}{dt}$$

$$\Rightarrow \qquad N_1\int_{\phi_1}^{\phi_2} d\phi = G\int_{\theta_1}^{\theta_2} d\theta + L\int_{i_1}^{i_2} di$$

$$\Rightarrow \qquad N_1(\phi_2 - \phi_1) = G(\theta_2 - \theta_1) + L(i_2 - i_1)$$

Now, if we assume that the period in which the flux is changing is completely contained within the period $(0 - t)$ over which the integration is carried, both the initial and final currents are zero, giving

$$N_1(\phi_2 - \phi_1) = G(\theta_2 - \theta_1)$$

$$N_1\Delta\phi = G\Delta\theta \quad \Rightarrow \qquad \Delta\theta = \frac{N_1}{G}\Delta\phi$$

$$\Rightarrow \qquad \Delta\theta \propto \Delta\phi$$

which suggests that the deflection in the fluxmeter accurately follows any change in flux in the coil.

4.7 SUMMARY

- A cathode ray oscilloscope is used for the measurement of electrical parameters like, ac and dc voltage, ac and dc current, time-phase relationship, frequency and for observing various waveforms.
- Laboratory oscilloscopes can be classified into two categories: (i) Dual trace oscilloscope and (ii) Storage oscilloscope.

- A signal generator provides a variety of output waveforms over a wide range of frequency. The most common output waveforms are: sine, pulse, square, triangular and ramp.
- An electronic voltmeter is characterised by high input resistance.
- A power meter is used to measure the power or rate of consumption of electricity in a circuit.
- A magnetic field meter is an instrument for measuring magnetic field strengths.

4.8 TERMINAL QUESTIONS

(1) Explain the functioning of a general purpose CRO, giving the block diagram.
(2) Explain the basic principle involved in the dual trace CRO.
(3) How does storage CRO work? Explain.
(4) Describe in detail the functions of a function generator.
(5) Describe in detail, the functioning of the differential-amplifier type electronic voltmeter.
(6) How does a Siemens power meter work? Explain.
(7) Give in detail, how magnetic field can be measured with the help of a fluxmeter.

4.9 SOLUTIONS AND ANSWERS

SAQs

1. See text
2. See text
3. $T = \frac{2\mu\text{sec}}{\text{div}} \times \frac{4\text{div}}{\text{cyc}} = \frac{8\mu\text{sec}}{\text{cyc}}$

   $\therefore \quad f = \frac{1}{T} = \frac{1}{8\mu\text{sec/cyc}}$ = 125 KHz
4. See text
5. See text
6. See text
7. We know

$$f = \frac{1}{4RC}\frac{R_2}{R_1}$$

$$= \frac{1}{4\times 500\times 10^3 \times 0.004\times 10^{-16}}\frac{100\times 10^3}{60\times 10^3}$$

= 208 Hz

Also, $V_{02} = V_{CC}\frac{R_1}{R_2}$

$$= 15\left(\frac{60\times 10^3}{100\times 10^3}\right) = 9\text{ V}$$

8. See text
9. See text
10. $i = \frac{g_m(r_d \parallel R_D)}{2(r_d \parallel R_D) + g_m}$

Substituting all the values, we get

$$i = 2.5 \text{ mA}$$

TQs

1. See text
2. See text
3. See text
4. See text
5. See text
6. See text
7. See text