3. Encoders and decoders (using the principle...

4. Adder and subtractions, in these laboratory students will construct a full adder and subtract or using basic design principle.

5. RS, D-Type, clocked D and master slave. In this laboratory students will design and verify the concepts of different flip-flops based on basic logic gates.

6. Design of counters (decade counters and binary counters). Students will design decade and binary counters verify the concepts suing the CAD tools.

7. Design of shit registers (serial in serial out and parallel in parallel out)

Reference Books:

1. Malvino: *Digital Computer Electronics*
2. Morries Mano: *Digital Logic and Computer Design*
3. Frederic J. Mowle: *A systematic approach to digital logic design*

Contents

CHAPTER 1
INTRODUCTION

* * * * *

* * * * *

1.1 DIGITAL NUMBER SYSTEM

The study of number system is important from the viewpoint of understanding how data are represented before they can be processed by any digital system including a digital computer. There are two basic ways of representing the numerical values of the various physical quantities with which we constantly deal in our day-to-day lives. One of the ways, referred to as analogue, is to express the numerical values of the quantity as a continuous range of the values of the two expected extreme values. The underlying concept in this mode of representation is that variation in the numerical value of the quantity is continuous and could have any of the infinite theoretically possible values between the two extremes.

The other possible way, referred to as digital, represents the numerical value of the quantity in steps of discrete values. The numerical values are mostly represented using binary numbers. To summarize, while an analogue representation gives a continuous output, a digital representation produces a discrete output. Analog system contain devices that process or work on various physical quantities represented in analog form. Digital system contains devices that process the physical quantities

represented in digital form. Digital techniques and systems have the advantages of being relatively much easier to design and having higher accuracy, programmability, noise immunity, easier storage of data and ease of fabrication in integrated circuit form, leading to availability of more complex functions in a smaller size.

1.2 DIGITAL AND ANALOG SYSTEM

Digital systems represents information using a binary system where data can assume one of only two possible values: 0 or 1. Digital systems are designed to store, process and communicate information in digital form. They are found in a wide range of applications, including process control, communication systems, digital instruments and consumer products. The digital computer, more commonly called the computer, is an example of a typical digital system. A digital system is a system that stores data in a discrete way. The opposite is an analogue system, which stores the data in a continuous way. Example of digital system are calculator, digital voltmeters, general purpose computers etc.

System based on continuous forms of information are called analog system or devices. A watch that displays time with hour, minute and second hands is an example of an analog device whereas a watch that displays the time in decimal digits is a digital device. Analog computers and other analog systems were in use long before digital device were perfected. Why then have digital systems supplanted analog systems in most application areas? There are several reasons.

- In general, digital techniques offer more flexibility than do analog techniques in that they can more easily programmed to perform any desired algorithm.
- Digital circuits provide for more powerful processing capabilities in terms of speed.
- Numeric information can be represented digitally with greater precision and range than it can with analog signals.
- Information storage and retrieval functions are much easier to implement in digital form than in analog.
- Digital techniques allow use of built-in error detection and correction mechanisms.
- Digital systems lend themselves to miniaturization more than do analog systems.

1.3 INTEGRATED CIRCUIT (IC)

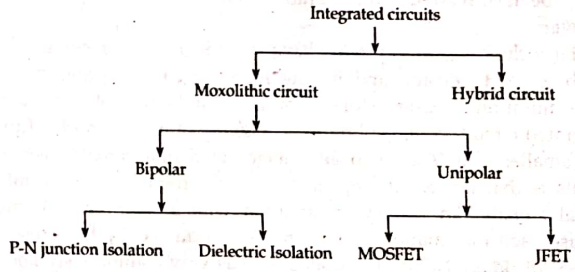
An Integrated circuit (IC) or monolithic integrated circuit or microchip is a set of electronic circuits on one small flat piece of semiconductor material that is normally silicon. The integration of large numbers of tiny MOS transistors into a small chip results in circuits that are orders of magnitude electronic components. The IC's mass production capability,

reliability and building-block approach to circuit design has ensured the rapid adoption of standardized ICs in place of designs using discrete transistors. An integrated circuit is defined as a circuit in which all or some of the circuit elements are inseparably associated and electrically interconnected so that it is considered to be indivisible for the purpose of construction and commerce.

The integrated circuit or IC is a miniature, low cost electronic circuit consisting of active and passive components that are irreparably joined together on a single crusted chip of silicon. Most of the components used in ICS are not similar to conventional components in appearance although they perform similar electrical functions. Integrated circuits offer a wide range of applications and could be broadly classified as:

i) Digital ICs
ii) Linear ICs

Based upon the requirements, distinctly different IC technology namely. Monolithic technology and Hybrid technology have been developed. In monolithic integrated circuits, all circuit component both active and passive elements and their interconnections are manufactured into or on top of a single chip of silicon. The monolithic circuit is ideal for applications where identical circuits are required in very large quantities and hence provides lowest per unit cost and highest order of reliability. In hybrid circuits, separate component parts are attached to a ceramic substrate and interconnected by means of either metallization pattern or wire bounds. This technology is more adaptable to small quantity custom circuits. Based upon the active devices used, ICs can be classified as bipolar (using BJT) and unipolar (using FET).



EXAMINATION QUESTION SOLUTIONS

1. Compare analog and digital system. [2011/F, 2011/S, 2012/S, 2012/F, 2014/F, 2015/S, 2015/F, 2017/F, 2017/S, 2018/S, 2019/S]

Answer:

S.N.	Analog system	Digital system
a)	System based on continuous forms of information are called analog system.	Digital systems represents information using binary system where data can assume one of only two possible value *i.e.*, 0 or 1.
b)	An analog system is more prone to distortion.	A digital system is less prone to distortion.
c)	An analog system transmit data in the form of wave.	A digital system carries data in the binary form *i.e.*, 0 or 1.
d)	Analog system draws large power.	Digital system draws only negligible power.
e)	It is expensive system.	It is less expensive system.
f)	Analog systems are less adjustable for a range of use.	Digital systems are more adjustable for a range of use.
g)	The data storage of an analog system is in the wave signal form.	Digital system stores the data in the binary form.
h)	*Example*: Analog watch, Analog voltmeter, analog thermometer etc.	*Example*: Digital watch, digital voltmeter, digital calculator etc.
i)	It is more accurate system but are less flexible.	It is less accurate system but are more flexible.

2. Describe in brief why most of system in the present days has been converted to digital rather than analog. [2013/S, 2014/F]

Answer:

Digital techniques and systems have the advantages of being relatively much easier to design and having higher accuracy, programmability, noise immunity, easier storage of data and ease of fabrication in integrated circuit form, leading to availability of more complex functions in a smaller size. The main advantages of digital signals over analog signals is that the precise signal level of the digital signal is not vital. Digital signals can convey information with greater noise immunity, because each information component is determined by the presence or absence of data bit (0 or 1). Analog signals vary continuously and their value is affected by all level of noise.

Digital signals do not get corrupted by noise. Digital signals can be processed by digital circuit components which are cheap and easily produced in many components on a single chip. Again, noise propagation through the demodulation system is minimized with digital techniques. Digital signals can be encrypted. Digital signals typically use less bandwidth. Digital transmission is more secure than analog transmission.

Also, it enables multi-directional transmission simultaneously. Thus, most of the system in the present days has been converted to digital rather than analog.

3. Define analog and digital system. [2013/F, 2016/F, 2016/S, 2018/F]

Answer:

A signal is considered analog if it is defined for all points in time and if it can take any real magnitude value within its range. An analog system is a system that represents data using a direct conversion from one form to another. In other words, an analog system is a system that is continuous in both time and magnitude.

A signal or system is considered digital if it is both discrete time and quantized. In other words, a digital system is a system that stores data in a discrete way and it deals with digital signal.

4. "Digital circuits are easier to design than analog circuit". Do you agree with this statement? Give reason to support your answer. [2014/S]

Answer:

A digital circuit is a circuit where the signal must be one of the two discrete levels. Each level is interpreted as one of two different states (*For example*; on/off, 0/1, true/false). Digital circuits use transistors to create logic gates in order to perform Boolean logic. Digital circuits are less susceptible to noise or degradation in quality than analog circuits. It is also easier to perform error detection and correction with digital signals. Digital techniques are helpful because it is lot easier to get an electronic device to switch into one of a number of known states than to accurately reproduce a continuous range of values. In a digital system, a more precise representation of a signal can be obtained by using more binary digits to represent it. While this requires more digital circuits to process the signals, each digit is handled by the same kind of hardware, resulting in an easily scalable system. In an analog system, additional resolution requires fundamental improvements in the linearity and noise characteristics of each step of the signal chain.

With computer controlled digital systems, new functions to be added through software revision and no hardware changes. Often this can be done outside of the factory by updating the products software. Information storage can be easier in digital system than in analog ones. In some cases, digital circuits use more energy than analog circuits to accomplish the same tasks, thus producing more heat which increases the complexity of the circuits such as the inclusion of heat sinks. A digital circuit called logic gates that can be used to create combinational logic. Each logic when acting on logic signals. A logic gate is generally created from one or more electrically controlled switches, usually transistors. The output of a logic gate can be, in turn control or feed into more logic gates. Thus, digital circuits are easier to design than analog circuits.

5. How can we represent the information? [2015/S]

Answer:

The signals in most present day's electronic digital system use just two values and are therefore said to be binary. The two discrete values used are often called 0 and 1, the digits for the binary number system. Information is represented using signals. Electrical signals such as voltage and currents are the most common. There are two types of signals: Analog and digital signal. Analog signals represent information that is continuous in time and value. Digital signals represent information that is discrete in time and value.

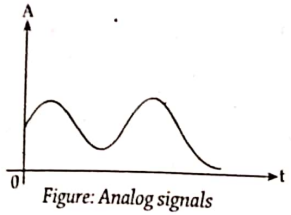


Figure: Analog signals

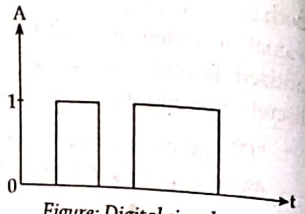


Figure: Digital signals

6. List out the advantages of digital system over the analog system. [2016/F, 2019/F]

OR, Why digital systems are preferred over analog system. [2011/F, 2013/F, 2016/S]

Answer:

Advantages of digital system over the analog system are;

a) Reproducibility of the results and accuracy
b) Flexibility and functionality
c) Programmability
d) Ease of design.
e) High speed
f) Economy.
g) Maintainability
h) Process more efficiently and reliably
i) Does not affect by noise
j) Compact storage
k) Simple circuit
l) Low power consumption
m) Small in size

7. Which system is more efficient for logical computation? [2015/F]

Answer: Digital system is more efficient for logical computation.

8. Explain the digital number system. [2017/S]

Answer: See the topic 1.1.

9. How does the logic system express data during computation?

Answer: [2018/S]

Logic system expresses data during computation by using current. If current is flowing, then it is expressed as binary 1 and if there is absence of current, then it is expressed as binary 0. Logic system can also be expressed by voltage with presence and absence of voltage representing 1 and 0 respectively.

CHAPTER 2

NUMBER SYSTEM AND CODES

*****

*****

2.1 BINARY TO DECIMAL AND DECIMAL TO BINARY CONVERSIONS

2.1.1 Binary Number System

The binary number system is a radix-2 number system with '0' and '1' as the independent digits. In the binary system, the base is 2, so any number can be expressed by an array of numbers representing the coefficients of power 2. In a binary system, the weight of each successively higher position (to the left) is an increasing power of 2. Some terms like bit, nibble and byte are used in binary number system. Bit is used for a single binary digit. A binary number with four bits is called a nibble. When binary number has eight bits, then it is called a byte.

The first 16 numbers in the binary number system would be 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, and 1111. The next number after 1111 is 10000 which is the lowest binary number with five digits.

2.1.2 Decimal Number System

The decimal number system has a base of 10 and is a position-value system. The statement 'the decimal number system has a base of 10' implies that it contains ten unique symbols (or digits) *i.e.*, 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. The ten digits do not limit us to express only ten different quantities because we use the various digits in appropriate positions within a number to indicate the magnitude of the quantity. For expressing quantities exceeding nine, two or more digits are used and the position of each digit within the number indicates the magnitude it represents.

A. Binary to Decimal Conversions

The binary number system is a positional system where each binary digit (bit) carries a certain weight based on its position relative to the least significant bit (LSB). The right most bit is the LSB in a binary number and has a weight of $2^0 = 1$. Any number can be converted to its decimal equivalent simply by adding the products of each bit and its weight.

For example;

$$10011 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$
$$= 16 + 0 + 0 + 2 + 1 = (19)_{10}$$

Example 1

Find the decimal equivalent of binary number 11010.

Solution:

Binary weight	2^4	2^3	2^2	2^1	2^0
Weight value	16	8	4	2	1
Binary number	1	1	0	1	0

so, $(11010)_2 = 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 = (26)_{10}$

Example 2

Find the decimal equivalent of $(11011000)_2$

Solution:

$$(11011000)_2 = 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$
$$= 128 + 64 + 0 + 16 + 8 + 0 + 0 + 0 = (216)_{10}$$

Fractional number can also be represented in binary by placing bits to the right of the binary point, just as fractional decimal digits are placed to the right of the decimal point. The column weights of a binary number are,

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

(Binary point)

This indicates that all the bits to the left of the binary point have weights that are positive powers of two, but all bits to the right of the binary points have weights that are negative power of two or fractional weights $\left(2^{-1} = \frac{1}{2^1} \text{ etc}\right)$.

Binary fractional number can also be converted into decimal fractional numbers. Mixed binary numbers can be converted into a decimal number by converting integer binary number and fractional binary number individually.

Example 3

Convert the following number into decimal $(1001010.0101)_2$

Solution:

$$1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$$
$$= 1 \times 64 + 0 + 0 + 1 \times 8 + 0 + 2 + 0 + 0 + 1 \times 0.25 + 0 \times 0.125 + 1 \times 0.0625 = (74.3125)_{10}$$

Example 4

Convert $(11101.01)_2$ to decimal.

Solution:

$$= 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$$
$$= (29.25)_{10}$$

B. Decimal to Binary Conversion

For the conversion of a decimal number into a binary number, decimal number is divided by 2 progressively and remainder is written down after each division. The remainder is then taken in a reverse order and it is the binary equivalent. This method of conversion is usually called the double dabble method.

Example 5

Convert $(1245)_{10}$ to binary equivalent.

Solution:

2	1245	1
2	622	0
2	311	1
2	155	1
2	77	1
2	38	0
2	19	1
2	9	1
2	4	0
2	2	0
	1	

Now writing down all the remainders in reverse order, we get the binary equivalent of decimal number 1245. Thus,

$(1245)_{10} = (10011011101)_2$

Fractional decimal numbers are converted into binary numbers, multiplying it by 2. Carry in the integer position of the result is recorded and carry of the decimal position is again multiplied by 2. Again the process is repeated. The carries of integer position recorded are written downward which gives fractional binary number equivalent of fractional decimal number.

Example 6

Convert $(201.32)_{10}$ into its equivalent binary number.

Solution:

Integer 201

2	201	1
2	100	0
2	50	0
2	25	1
2	12	0
2	6	0
2	3	1
	1	

Fraction 0.32

$0.32 \times 2 = 0.64$ with carry 0
$0.64 \times 2 = 0.28$ with carry 1
$0.28 \times 2 = 0.56$ with carry 0
$0.56 \times 2 = 0.12$ with carry 1
$0.12 \times 2 = 0.24$ with carry 0
$0.24 \times 2 = 0.48$ with carry 0
$0.48 \times 2 = 0.96$ with carry 0
$0.96 \times 2 = 0.96$ with carry 1
$0.92 \times 2 = 0.84$ with carry 1

Thus, $(201.32)_{10} = (11001001.010100011)_2$

Example 7

Convert the following $(0.0625)_{10} = (?)_2$

Solution:

$0.0625 \times = 0.125$ with a carry of 0
$0.125 \times 2 = 0.25$ with a carry of 0

$0.25 \times 2 = 0.50$ with a carry of 0
$0.5 \times 1 = 0.00$ with a carry of 1

i.e., $(0.0625)_{10} = (0.0001)_2$

2.2 OCTAL, HEXADECIMAL NUMBER SYSTEM AND CONVERSIONS

2.2.1 Octal Number System

The number system with base or radix eight is known as the octal number system. Although we can use any eight digits, it is customary to use the first eight decimal digits 0, 1, 2, 3, 4, 5, 6, 7. There is no 8 or 9 in octal number system.

Conversion from binary to octal and octal to binary is also quick and simple. In fact, octal numbers are used to represent binary numbers because if ease of conversion and compactness. Since digital circuits can process only zeros and ones, the octal numbers have to be converted into binary form employing special circuits known as octal to binary converters before being processed by the digital circuits.

2.2.2 Hexadecimal Number System

The binary number system forms the natural choice for the two state system. But in this system, the numbers tend to get short rather long. Hence to reduce the length of a given number, it is quite common to use hexadecimal system. The hexadecimal system has a base of 16: that is, it is composed of 16 digits and characters. It uses the digits 0 through 9 plus the letters A, B, C, D, E and F. Since numeric digits and alphabets both are used to represent the digits in the hexadecimal number system, this is alphanumeric number system. Table below shows the relationship between hexadecimal, decimal, octal and binary number system.

Hexadecimal	Decimal	Octal	Binary
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	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

A. Octal to decimal conversion

An octal number can be easily converted to its decimal equivalent by multiplying each octal digit by its positional weight.

Example 8

Convert octal 756 to decimal

Solution:

$$(756)_8 = 7 \times 8^2 + 5 \times 8^1 + 6 \times 8^0 = (494)_{10}$$

Example 9

Convert the following number $(36.125)_8 = (\quad)_{10}$

Solution:

$$\begin{aligned}(36.125)_8 &= 3 \times 8^1 + 6 \times 8^0 + 1 \times 8^{-1} + 2 \times 8^{-2} + 5 \times 8^{-3} \\ &= 24 + 6 + 0.125 + 0.03125 + 0.009765625 \\ &= (30.16601563)_{10}\end{aligned}$$

B. Decimal to Octal Conversion

A decimal integer can be converted to octal by using the same repeated division method called the double dabble method, that we used in the decimal to binary conversion; but with a division factor of 8 rather than 2.

Example 10

Convert $(540)_{10}$ to $(\ ?\ )_8$

Solution:

8	540	4
8	67	3
8	8	0
	1	

$\therefore \quad (540)_{10} = (1034)_8$

Example 11

Convert $(5621.125)_{10}$ to octal

Solution:

Integer

8	5621	5
8	702	6
8	87	7
8	10	2
	1	

Fraction

$0.125 \times 8 = 0$ with carry of 1

Hence, $(5621.125)_{10} = (12765.1)_8$

C. Octal to Binary Conversion

Because 8 is the third power of 2, the conversion from octal to binary can be performed by converting each octal digit to its 3-bit binary equivalent. The eight possible digits are converted as indicated below in table.

Octal digit	0	1	2	3	4	5	6	7
Binary equivalent	000	001	010	011	100	101	110	111

Using these conversions, any octal number can be converted to binary by individually converting each digit.

Example 12

Convert $(628)_8$ to $(\quad)_2$

Solution:

$$(628)_8 = (630)_8 = \begin{matrix} 6 & 3 & 0 \\ \downarrow & \downarrow & \downarrow \\ 110 & 011 & 000 \end{matrix}$$

so, $(628)_8 = (110011000)_2$

Example 13

Convert $(375.37)_8 = (\quad)_2$

Solution:

$$\begin{matrix} 3 & 7 & 5 & \cdot & 3 & 7 \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow \\ 011 & 111 & 101 & \cdot & 011 & 111 \end{matrix}$$

Thus, $(375.37)_8 = (011111101.011111)_2$

D. Binary to Octal Conversion

Conversion of a binary number to an octal number is simply the reverse of the fore-going process. The bits of the binary number are grouped into groups of three bits starting from LSB towards MSB for integer part and then each group of three bits is replaced by its octal equivalent. Zeros are added as required to complete a 3-bit group. For the fractional part, the above procedure is repeated from the bit next to the binary point.

Example 14

Convert $(111000)_2$ to octal

Solution:

$$(111000)_2 = \begin{matrix} 111 & 000 \\ \downarrow & \downarrow \\ 7 & 0 \end{matrix}$$

Hence, $(111000)_2 = (70)_8$

Example 15

Convert the following $(101010011.1101)_2$ to $(\quad)_8$.

Solution:

$$(101010011.1101)_2 = \begin{matrix} 101 & 010 & 011 & \cdot & 110 & 100 \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow \\ 5 & 2 & 3 & \cdot & 6 & 4 \end{matrix}$$

Hence, $(101010011.1101)_2 = (523.64)_8$

E. Hexadecimal to Decimal Conversion

A hexadecimal number can be converted to its decimal equivalent b multiplying each hex digit by its weight and then taking the sum of the products. The weights of hex number are increasing powers of 16 (fro right to left). For a four digit hex number, the weights are as follows.

16^3	16^2	16^1	16^0
4096	256	16	1

Example 16

Convert $(1A53)_{16}$ to decimal

Solution:

$$(1A53)_{16} = 1 \times 16^3 + 10 \times 16^2 + 5 \times 16^1 + 3 \times 16^0 = (6739)_{10}$$

Example 17

Convert $(131.F_2)_{16} = (?)_{10}$

Solution:

$$(131.F_2)_{16} = 1 \times 16^2 + 3 \times 16^2 + 1 \times 16^0 + 15 \times 16^{-1} + 2 \times 16^{-2}$$

$$= 256 + 48 + 1 + \frac{15}{16} + \frac{2}{256}$$

$$= (305.9453125)_{10}$$

F. Decimal to Hexadecimal Conversion

Repeated division of a decimal number by 16 will produce the equivale hex number formed by the remainder of each division. This is similar t the repeated division by 2 for decimal to binary conversion and repeate division by 8 for decimal to octal conversion.

Example 18

Convert $(374.37)_{10} = (\quad)_{16}$

Solution:

Integer 374

16	374	6
16	23	7
	1	

Fraction 0.37

$0.37 \times 16 = 0.92$ with carry of 5

$0.92 \times 16 = 0.72$ with carry of 14 or E

$0.72 \times 16 = 0.52$ with carry of 11 or B

$0.52 \times 16 = 0.32$ with carry of 8

Thus, $(374.37)_{10} = (176.5EB8)_{16}$

G. Hexadecimal to Binary Conversion

Hex numbers can be converted into equivalent binary numbers by replacing each hex digit by its equivalent 4-bit binary number.

Example 19

Convert $(1684)_{16}$ to $(\quad)_2$

Solution:

$(1684)_{16} =$ 1 6 8 4

↓ ↓ ↓ ↓

0001 0110 1000 0100

Hence, $(1684)_{16} = (1011010000100)_2$

Example 20

Convert $(A6B{\cdot}F5)_{16} = (?)_2$

Solution:

$(A6B{\cdot}F5)_{16} =$ A 6 B · F S

↓ ↓ ↓ ↓ ↓

1010 0110 1011 · 1111 0101

Hence, $(A6B{\cdot}F5)_{16} = (101001101011.11110101)_2$

H. Hexadecimal to Octal Conversion

Hexadecimal number can be converted to equivalent octal numbers and octal number can be converted to equivalent hex number by converting the hex octal numbers to equivalent binary and then to octal/hex respectively.

Example 21

Convert $(F2A4)_{16}$ into $(\quad)_8$

Solution:

$(F2A4) =$ 1111 0010 1010 0100

↓

001 111 001 010 100 100

↓ ↓ ↓ ↓ ↓ ↓

1 7 1 2 4 4

Hence, $(F2A4)_{16} = (171244)_8$

Example 22

Convert $(2715)_8$ to $(\quad)_{16}$

Solution:

$(2715)_8 =$ 2 7 1 5

↓ ↓ ↓ ↓

010 111 001 101

↓

0101 1100 1101

↓ ↓ ↓

5 12 13

↓ ↓ ↓

5 C D

Example 23

Convert $(C3A{\cdot}47)_H$ to ( )$_8$

Solution:

C	3	A	.	4	7
↓	↓	↓		↓	↓
1100	0011	1010	.	0100	0111

$(110000111010.01000111)_2$

110	000	111	010	.	010	001	110
↓	↓	↓	↓		↓	↓	↓
6	0	7	2	.	2	1	6

and,

so, $(C3A{\cdot}47)_H = (6072.216)_8 = (110000111010.01000111)_2$

2.3 BINARY ARITHMETIC 1'S COMPLEMENT AND 2'S COMPLEMENT

In binary number system, negative binary numbers are represented in three ways; signed magnitude representation, 1's complement representation and 2's complement representation.

2.3.1 1's Complement Subtraction

Subtraction of binary numbers can be accomplished by using 1's complement method, which allows us to subtract using only addition. This method of subtraction is particularly useful in arithmetic logic circuits because subtraction can be accomplished with an adder. Also the 1's complement of a number is easily obtained by inverting each bit in the number.

Binary subtraction can be performed by adding 1's complement of the subtrahend to the minuend. If a carry is generated, remove the carry, add it to the result. This carry is called the end around carry. Now if the subtrahend is larger than the minuend, then no carry is generated. The answer obtained is 1's complement of the true result and opposite in sign.

Example 24

Subtract $(1001)_2$ from $(1101)_2$ using the 1's Complement method. Also subtract using the direct method and compare.

Solution:

Direction subtraction

$$\begin{array}{r} 1101 \\ -1001 \\ \hline 0100 \end{array}$$

1's complete method

$$\begin{array}{rr} & 1101 \\ \text{1's complement of 1001} \rightarrow & +0110 \\ \hline \text{Carry} \rightarrow & \textcircled{1}0011 \\ \text{Add carry} \rightarrow & +1 \\ \hline & 0100 \end{array}$$

Example 25

Perform the subtraction for following binary number using 1's complement.

a) 11010 – 1101

Solution:

$$\begin{array}{rl} 11010 & \\ +1101 & \text{1's complement of 1101} \\ \hline \textcircled{1}0110 & \\ \quad\rightarrow 1 & \text{Add end - around carry} \\ \hline 1101 & \end{array}$$

b) 100 – 110000

Solution:

$$\begin{array}{rl} 100 & \\ +001111 & \text{1's complement of 110000} \\ \hline 010011 & \text{1's complement result} \\ -101100 & \end{array}$$

Final result (No carry indicates that the answer is negative and in complement form).

2.3.2 2's Complement Subtraction

The 2's complement of a binary number is obtained by adding 1 to its 1 complement *i.e.*, 2's complement = 1's complement + 1

Example 26

Find the 2's complement of $(110110)_2$

Solution:

= 2's complement of $(110110)_2$

= 1's complement of $(110110)_2 + 1$

= $001001 + 1 = (001010)_2$

For subtracting a smaller number from a larger one by 2's complement subtraction method, the procedure applied is as follows;

i) Determine the 2's complement of the smaller number.

ii) Add the 2's complement of the smaller number to the larger one.

iii) Discard the carry (three is always a carry in this case).

For subtracting a larger number from smaller one by 2's complement subtraction method, the procedure applied is as follows;

i) Determine the 2's complement of the larger number.

ii) Add 2's complement of the larger number to the smaller one.

iii) If there is no carry, the result is in 2's complement and is negative.

iv) For having result in true form, take 2's complement and change the sign.

Example 27

Perform 11100.101 – 101.101 using 2's complement.

Solution:

$$\begin{array}{r} 11100.101 \\ +\ 11010.110 \quad \text{2's complement of } 101.01 \\ \hline 110111.011 \\ 10111.011 \quad \text{Dropping end - around carry} \end{array}$$

so, $(11100.101)_2 - (101.01)_2 = (10111.011)_2$

Example 28

Perform the operation $(12_{10} - 35_{10})$ using 2's complement.

Solution:

$(12)_{10} = (001100)_2$

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

$$\begin{array}{r} 001100 \\ +\ 011101 \quad \text{2's complement of } (100011)_2 \\ \hline 101001 \end{array}$$

No carry indicates that result is negative and is 2's complement form.

i.e. $(-0101001)_2 = (-23)_{10}$

Example 29

If X = 111.101 and Y = 101.110

Calculate X + Y and X – Y.

Solution:

$$\begin{array}{rr} X = & 111.101 \\ Y = & 101.110 \\ \hline X + Y = & 1101.011 \end{array}$$

$$\begin{array}{rr} \text{and, } X = & 111.101 \\ \text{+ 2's complement of } Y = & +\ 010.010 \\ \hline & 1001.111 \end{array}$$

X – Y = X + 2's complement of Y

= 1001.111

= 001.111 Discarding end around carry

= 1.111

2.4 GRAY CODE

This code belongs to a class of codes called minimum-change code in which only one bit in the code group changes when going from one step to the next. This is an unweighted code which means that there are no specific weights assigned to the bit position. Because of this, the gray code is not suited for arithmetic operations but finds applications in input/output devices and some types of analog to digital converters. The advantage of the gray code over binary numbers is that only one bit in the code group changes when going from one number to the next.

The gray code representation for the decimal numbers 0 through 15 together with the straight binary code is shown in table below.

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

The gray code is used in applications where the normal sequence of binary numbers may produce an error or ambiguity during the transition from one number to the next. If binary numbers are used, a change from 0111 to 1000 may produce an intermediate erroneous number 1001 if the rightmost bit takes more time to change than the other three bits. The gray code eliminates this problem since only one bit changes in values during any transition between two numbers.

A typical application of gray code occurs when analog data are represented by continuous change of shaft position. The shaft is partitioned into segments and each segment is assigned a number. If adjacent segments are made to correspond with the gray-code sequence, ambiguity is eliminated when detection is sensed in the line that separates any two segments.

Binary-To-Gray Conversion

For conversion of binary number into a gray code number, following rules are applied.

i) The most significant digit (Left-most digit) in the gray cod is the same as the corresponding digit in the binary number.

ii) Going from left to right, add each adjacent pair of binary digits to get the next Gray code digit. Discard carries if any.

Example 30

Obtain the following conversion $(111011)_2$ to gray code

Solution:

Step 1: The left-most gray digit is the same as the left most binary digit

	1	1	1	0	1	1	Binary code
1							Gray code

Step 2: Add the left most binary digit to the adjacent one

	1 + 1	1	0	1	1	Binary code
1	↓ 0					Gray code

Step 3: Add the next adjacent binary number

1	1 + 1	0	1	1		Binary code
1	0 ↓ 0					Gray code

Step 4: Add the next adjacent binary number

1	1	1 0	1	1		Binary code
1	0	0 ↓ 1				Gray code

Step 5: Add the next adjacent binary number

1	1	1	0 1	1		Binary code
1	0	0	1 ↓ 1			Gray code

Step 6: Add the next adjacent binary number

1	1	1	0	1 1		Binary code
1	0	0	1	1 ↓ 0		Gray code

Hence, $(111011)_2 = (100110)_{Gray}$

Gray to Binary Conversion

For conversion of gray code number to binary code number, the applicable rules are as follow:

i) The most significant digit (left-most) in the binary code is the same as the corresponding digit in the gray code.

ii) Add each binary digit generated to the gray digit in the next adjacent position. Discard carries if any.

Example 31

Convert the gray code 11011 to equivalent binary code.

Solution:

Step 1: The left-most digits are the same

	1	1	0	1	1	Gray code
1						Binary code

Step 2: Add the last binary digit just generated to the gray digit in the next position. Discard carry.

1	1	0	1	1	
	+ ↓				
1	0				

Step 3: Add the last binary digit generated to the next gray digit.

1	1	0	1	1	
		+ ↓			
1	0	0			

Step 4: Add the last binary digit generated to the next gray digit.

1	1	0	1	1	
			+ ↓		
1	0	0	1		

Step 5: Add the last binary generated to the next gray digit.

1	1	0	1	1	Gray code
				+ ↓	
1	0	0	1	0	Binary code

Thus, $(11011)_{Gray} = (10010)_2$

2.5 INSTRUCTION CODES

An instruction code is a group of bits that tells the computer to perform a specific operation part. An instruction must include one or more operands which indicate the registers and/or memory addresses from which data is taken or to which data is deposited. Instructions are encoded as a binary instruction codes. Each instruction code contains of a operation code or opcode which designates the overall purpose of the instruction (*For example*; add, subtract, move, input etc). In addition to the opcode, many instructions also contain one or more operands, which indicate where in registers or memory the data required for the operation is located. The opcode and operands are most often encoded as unsigned binary numbers in order to minimize the number of bits used to store them.

Instruction codes, together with data, are stored in memory. The control reads each instruction from memory and places it in a control register. The control then interprets the instruction and proceeds to execute it by issuing a sequence of control functions. Every general purpose computer has its own unique instructions, the stored program concept, is the most important property of a general purpose computer.

An instruction code is usually divided into parts each having its own particular interpretation. The most basic part of an instruction code is its operation part. The operation code of an instruction is a group of bits that define an operation such as add, subtract, multiply, shift and complement. The number of bits required for the operation part of the

instruction code is a function of the total number of operations used. It must consist of atleast n bits for a given 2^n (or less) distinct operation. The designer assigns a bit combination (a code) to each operation.

The operation part of an instruction code specifies the operation to be performed. This operation must be executed on some data, usually stored in computer registers. An instruction code, therefore, must specify not only the operation but also the registers where the operands are to be found as well as the register where the result is to be stored. These registers may be specified in an instruction code in two ways. A register is said to be specified explicitly if the instruction code contains special bits for its identification.

Instruction-code formats

The format of an instruction is usually depicted in a rectangular box symbolizing the bits of the instruction as they appear in memory words in a control register. The bits of the instruction are sometimes divided into groups that subdivide the instruction into parts. Each group is assigned a symbolic name, such as operation-code or address part. The various parts specify different functions for the instruction and when shown together they constitute the instruction-code format.

Operation code	(a) Implied
Operation code | Operand	(b) Immediate operand
Operation code | Address of operand	(c) direct address

Figure: Three possible instruction formats

2.6 ALPHANUMERIC CHARACTERS

An alphanumeric character sets is a set of elements that includes the 10 decimal digits, the 26 letters of the alphabet, and a number of special characters. Such a set contains between 36 and 64 elements if only capital letters are included or between 64 and 128 elements if both uppercase and lowercase letters are included. In the first case, we need a binary code of seven bits.

2.6.1 ASCII Code

The standard binary code for the alphanumeric characters is ASCII (American standard code for information interchange). It uses seven bits to code 128 characters. ASCII is a 7-bit code, but most computers manipulate an 8-bit quantity as a single unit called byte. Therefore, ASCII characters most often are stored one per byte. The extra bit is sometimes used for other purposes, depending on the application. Most of these codes, however also represent symbols and various instructions necessary for conveying intelligible information.

ASCII, pronounced "askee" is perhaps the most widely used alphanumeric code. The ASCII code contain 94 graphic characters that can be printed and 34 non printing characters used for various control functions. The graphic characters consist of the 26 upper-case letters (A through Z), the lowercase letters (a through z), the numerals (0 through 9) and 32 special printable characters such as %, *, and $.

Table: ASCII Code

Binary	Hex.	Dec.	ASCII Symbol	Explanation	Group
0000000	0	0	NUL	The null character prompts the device to do nothing	Control Character
0000001	1	1	SOH	Initiates a header (Start of Heading)	Control Character
0000010	2	2	STX	Ends the header and marks the beginning of a message. (Start of text)	Control Character
0000011	3	3	ETX	Indicates the end of the message (End of text)	Control Character
0000100	4	4	EOT	Marks the end of a completes transmission (End of Transmission)	Control Character
0000101	5	5	ENQ	A request that requires a response (Enquiry)	Control Character
0000110	6	6	ACK	Gives a positive answer to the request (Acknowledge)	Control Character
0000111	7	7	BEL	Triggers a beep (Bell)	Control Character
0001000	8	8	BS	Lets the cursor move back one step (Backspace)	Control Character
0001001	9	9	TAB (HT)	A horizontal tab that moves the cursor within a row to the next predefined position (Horizontal Tab)	Control Character
0001010	A	10	LF	Causes the cursor to jump to the next line (Line Feed)	Control Character
0001011	B	11	VT	The vertical tab lets the cursor jump to a predefined line (Vertical Tab)	Control Character
0001100	C	12	FF	Requests a page break (Form Feed)	Control Character
0001101	D	13	CR	Moves the cursor back to the first position of the line (Carriage Return)	Control Character
0001110	E	14	SO	Switches to a special presentation (Shift Out)	Control Character
0001111	F	15	SI	Switches the display back to the normal state (Shift In)	Control Character

Binary	Hex.	Dec.	ASCII Symbol	Explanation	Group
0010000	10	16	DLE	Changes the meaning of the following characters (Data Link Escape)	Control Character
0010001	11	17	DC_1	Control characters assigned depending on the device used (Device Control)	Control Character
0010010	12	18	DC_2	Control characters assigned depending on the device used (Device Control)	Control Character
0010011	13	19	DC_3	Control characters assigned depending on the device used (Device Control)	Control Character
0010100	14	20	DC_4	Control characters assigned depending on the device used (Device Control)	Control Character
0010101	15	21	NAK	Negative response to a request (Negative Acknowledge)	Control Character
0010110	16	22	SYN	Synchronizes a data transfer, even if no signals are transmitted(Synchronous Idle)	Control Character
0010111	17	23	ETB	Marks the end of a transmission block (End of Transmission Block)	Control Character
0011000	18	24	CAN	Makes it clear that a transmission was faulty and the data must be discarded (Cancel)	Control Character
0011001	19	25	EM	Indicates the end of the storage medium (End of Medium)	Control Character
0011010	1A	26	SUB	Replacement for a faulty sign (Substitute)	Control Character
0011011	1B	27	ESC	Initiates an escape sequence and thus gives the following characters a special meaning (Escape)	Control Character
0011100	1C	28	FS	Marks the separation of logical data blocks and is hierarchically ordered: file as the largest unit, file as the smallest unit.(File Separator, Group Separator, Record Separator, Unit Separator)	Control Character

Binary	Hex.	Dec.	ASCII Symbol	Explanation	Group
0011101	1D	29	GS	Marks the separation of logical data blocks and is hierarchically ordered: file as the largest unit, file as the smallest unit.(File Separator, Group Separator, Record Separator, Unit Separator)	Control Character
0011110	1E	30	RS	Marks the separation of logical data blocks and is hierarchically ordered: file as the largest unit, file as the smallest unit.(File Separator, Group Separator, Record Separator, Unit Separator)	Control Character
0011111	1F	31	US	Marks the separation of logical data blocks and is hierarchically ordered: file as the largest unit, file as the smallest unit.(File Separator, Group Separator, Record Separator, Unit Separator)	Control Character
0100000	20	32	SP	Blank space (Space)	Special character
0100001	21	33	!	Exclamation mark	Special character
0100010	22	34	"	Only quotes above	Special character
0100011	23	35	#	Pound sign	Special character
0100100	24	36	$	Dollar sign	Special character
0100101	25	37	%	Percentage sign	Special character
0100110	26	38	&	Commercial and	Special character
0100111	27	39	'	Apostrophe	Special character
0101000	28	40	(	Left bracket	Special character
0101001	29	41	)	Right bracket	Special character
0101010	2A	42	*	Asterisk	Special character
0101011	2B	43	+	Plus symbol	Special character
0101100	2C	44	,	Comma	Special character
0101101	2D	45	-	Dash	Special character
0101110	2E	46	.	Full stop	Special character
0101111	2F	47	/	Forward slash	Special character
0110000	30	48	0		Numbers
0110001	31	49	1		Numbers
0110010	32	50	2		Numbers
0110011	33	51	3		Numbers
0110100	34	52	4		Numbers
0110101	35	53	5		Numbers

Binary	Hex.	Dec.	ASCII Symbol	Explanation	Group
0110110	36	54	6		Numbers
0110111	37	55	7		Numbers
0111000	38	56	8		Numbers
0111001	39	57	9		Numbers
0111010	3A	58	:	Colon	Special character
0111011	3B	59	;	Semicolon	Special character
0111100	3C	60	<	Small than bracket	Special character
0111101	3D	61	=	Equals sign	Special character
0111110	3E	62	>	Bigger than symbol	Special character
0111111	3F	63	?	Question mark	Special character
1000000	40	64	@	At symbol	Special character
1000001	41	65	A		Capital letters
1000010	42	66	B		Capital letters
1000011	43	67	C		Capital letters
1000100	44	68	D		Capital letters
1000101	45	69	E		Capital letters
1000110	46	70	F		Capital letters
1000111	47	71	G		Capital letters
1001000	48	72	H		Capital letters
1001001	49	73	I		Capital letters
1001010	4A	74	J		Capital letters
1001011	4B	75	K		Capital letters
1001100	4C	76	L		Capital letters
1001101	4D	77	M		Capital letters
1001110	4E	78	N		Capital letters
1001111	4F	79	O		Capital letters
1010000	50	80	P		Capital letters
1010001	51	81	Q		Capital letters
1010010	52	82	R		Capital letters
1010011	53	83	S		Capital letters
1010100	54	84	T		Capital letters
1010101	55	85	U		Capital letters
1010110	56	86	V		Capital letters
1010111	57	87	W		Capital letters
1011000	58	88	X		Capital letters
1011001	59	89	Y		Capital letters
1011010	5A	90	Z		Capital letters
1011011	5B	91	[	Left square bracket	Special character
1011100	5C	92	\	Inverse/backward slash	Special character
1011101	5D	93	]	Right square bracket	Special character
1011110	5E	94	^	Circumflex	Special character
1011111	5F	95	_	Underscore	Special character

Binary	Hex.	Dec.	ASCII Symbol	Explanation	Group
1100000	60	96	`	Gravis (backtick)	Special character
1100001	61	97	a		Lowercase letters
1100010	62	98	b		Lowercase letters
1100011	63	99	c		Lowercase letters
1100100	64	100	d		Lowercase letters
1100101	65	101	e		Lowercase letters
1100110	66	102	f		Lowercase letters
1100111	67	103	g		Lowercase letters
1101000	68	104	h		Lowercase letters
1101001	69	105	i		Lowercase letters
1101010	6A	106	j		Lowercase letters
1101011	6B	107	k		Lowercase letters
1101100	6C	108	l		Lowercase letters
1101101	6D	109	m		Lowercase letters
1101110	6E	110	n		Lowercase letters
1101111	6F	111	o		Lowercase letters
1110000	70	112	p		Lowercase letters
1110001	71	113	q		Lowercase letters
1110010	72	114	r		Lowercase letters
1110011	73	115	s		Lowercase letters
1110100	74	116	t		Lowercase letters
1110101	75	117	u		Lowercase letters
1110110	76	118	v		Lowercase letters
1110111	77	119	w		Lowercase letters
1111000	78	120	x		Lowercase letters
1111001	79	121	y		Lowercase letters
1111010	7A	122	z		Lowercase letters
1111011	7B	123	{	Left curly bracket	Special characters
1111100	7C	124	|	Vertical line	Special characters
1111101	7D	125	}	Right curly brackets	Special characters
1111110	7E	126	~	Tilde	Special characters
1111111	7F	127	DEL	Deletes a character. Since this control character consists of the same number on all positions, during the typewriter era it was possible to invalidate another character by punching out all the positions (Delete)	Control characters

The seven bits of the code are designated by b_1 through b_7 being the MSB. The letter A, for example, is represented in ASCII as 1000001 (column 100, row 0001).

Another alphanumeric code used in IBM equipment is the extended binary coded decimal interchange code. It uses eight bits for each character. It has same character symbol as ASCII but the bit assignment for characters is different. As the name implies, the binary code for the letters and numerals is an extension of the binary coded decimal (BCD) code. This means that the last four bits the code range from 0000 through 1001 as in BCD.

2.6.2 EBCDIC Code

The full form of EBCDIC is "Extended Binary coded decimal interchange code. It is also an alphanumeric code generally used in IBM equipment and in large computers for communicating alphanumeric data. For the different alphanumeric characters the code grouping in this code is different from the ASCII code. It is actually an 8-bit code and a ninth bit is added as the parity bit. EBCDIC is an eight bit code in which the decimal digits are represented by the 8421 code preceded by 111.

The EBCDIC, pronounced 'ed-si-dik' is widely used alphanumeric code mainly popular with larger systems. The code was created by IBM to extend the binary coded decimal that existed at that time. All IBM mainframe computer peripherals and operating systems use EBCDIC code and their operating system provide ASCII and Unicode modes to allow translation between different encodings. It may be mentioned here that EBCDIC offers no technical advantage over the ASCII code and its variant 150-8859 or Unicode. Its importance in the earlier days lay in the fact that it made it relatively easier to enter data into larger machines with punch cards. Since, punch cards are not used on mainframes any more, the code is used in contemporary mainframes machines solely for backwards compatibility.

A single byte in EBCDIC is divided into two four bit groups called nibbles. The first four bit, called the 'Zone' represents the category of the character, while the second group, called the 'digit' identifies the specified character.

Table: EBCDIC code

Decimal	Hex	Binary	Code	Code description
0	00	0000 0000	NUL	Null character
1	01	0000 0001	SOH	Start of header
2	02	0000 0010	STX	Start of text
3	03	0000 0011	ETX	End of text
4	04	0000 0100	PF	Punch off
5	05	0000 0101	HT	Horizontal tab
6	06	0000 0110	LC	Lower case
7		0000 1000		
8		0000 1001		

Decimal	Hex	Binary	Code	Code description
10	0A	0000 1010	SMM	Start of manual message
11	0B	0000 1011	VT	Vertical tab
12	0C	0000 1100	FF	Form feed
13	0D	0000 1101	CR	Carriage return
14	0E	0000 1110	SO	Shift out
15	0F	0000 1111	SI	Shift in
16	10	0001 0000	DLE	Data link escape
17	11	0001 0001	DC1	Device control 1
18	12	0001 0010	DC2	Device control 2
19	13	0001 0011	TM	Tape mark
20	14	0001 0100	RES	Restore
21	15	0001 0101	NL	New line
22	16	0001 0110	BS	Backspace
23	17	0001 0111	IL	Idle
24	18	0001 1000	CAN	Cancel
25	19	0001 1001	EM	End of medium
26	1A	0001 1010	CC	Cursor control
27	1B	0001 1011	CU1	Customer use 1
28	1C	0001 1100	IFS	Interchange file separator
29	1D	0001 1101	IGS	Interchange group separator
30	1E	0001 1110	IRS	Interchange record separator
31	1F	0001 1111	IUS	Interchange unit separator
32	20	0010 0000	DS	Digit select
33	21	0010 0001	SOS	Start of significance
34	22	0010 0010	FS	Field separator
35	23	0010 0011		
36	24	0010 0100	BYP	Bypass
37	25	0010 0101	LF	Line feed
38	26	0010 0110	ETB	End of transmission block
39	27	0010 0111	ESC	Escape
40	28	0010 1000		
41	29	0010 1001		
42	2A	0010 1010	SM	Set mode
43	2B	0010 1011	CU2	Customer use 2
44	2C	0010 1100		
45	2D	0010 1101	ENQ	Enquiry
46	2E	0010 1110	ACK	Acknowledge
47	2F	0010 1111	BEL	Bell
48	30	0011 0000		

Decimal	Hex	Binary	Code	Code description
49	31	0011 0001		
50	32	0011 0010	SYN	Synchronous idle
51	33	0011 0011		
52	34	0011 0100	PN	Punch on
53	35	0011 0101	RS	Reader stop
54	36	0011 0110	UC	Upper case
55	37	0011 0111	EOT	End of transmission
56	38	0011 1000		
57	39	0011 1001		
58	3A	0011 1010		
59	3B	0011 1011	CU3	Customer use 3
60	3C	0011 1100	DC4	Device control 4
61	3D	0011 1101	NAK	Negative acknowledge
62	3E	0011 1110		
63	3F	0011 1111	SUB	Substitute
64	40	0100 0000	SP	Space
65	41	0100 0001		
66	42	0100 0010		
67	43	0100 0011		
68	44	0100 0100		
69	45	0100 0101		
70	46	0100 0110		
71	47	0100 0111		
72	48	0100 1000		
73	48	0100 1001		
74	4A	0100 1010	¢	Cent sign
75	4B	0100 1011	.	Period, decimal point
76	4C	0100 1100	<	Less-than sign
77	4D	0100 1101	(	Left parenthesis
78	4E	0100 1110	+	Plus sign
79	4F	0100 1111	|	Logical OR
80	50	0101 0000	&	Ampersand
81	51	0101 0001		
82	52	0101 0010		
83	53	0101 0011		
84	54	0101 0100		
85	55	0101 0101		
86	56	0101 0110		
87	57	0101 0111		
88	58	0101 1000		
89	59	0101 1001		
90	5A	0101 1010	!	Exclamation point
91	5B	0101 1011	$	Dollar sign
92	5C	0101 1100	*	Asterisk
93	5D	0101 1101	)	Right parenthesis
94	5E	0101 1110	;	Semicolon
95	5F	0101 1111	^	Logical NOT
96	60	0110 0000	-	Hyphen, minus sign
97	61	0110 0001	/	Slash, virgule
98	62	0110 0010		
99	63	0110 0011		
100	64	0110 0100		
101	65	0110 0101		
102	66	0110 0110		
103	67	0110 0111		
104	68	0110 1000		
105	69	0110 1001		
106	6A	0110 1010		
107	6B	0110 1011	,	Comma
108	6C	0110 1100	%	Percent
109	6D	0110 1101	_	Underline, underscore
110	6E	0110 1110	>	Greater-than sign
111	6F	0110 1111	?	Question mark
112	70	0111 0000		
113	71	0111 0001		
114	72	0111 0010		
115	73	0111 0011		
116	74	0111 0100		
117	75	0111 0101		
118	76	0111 0110		
119	77	0111 0111		
120	78	0111 1000		
121	79	0111 1001	`	Grave accent
122	7A	0111 1010	:	Colon
123	7B	0111 1011	#	Number sign, octothorp, pound
124	7C	0111 1100	@	At sign
125	7D	0111 1101	'	Apostrophe, prime
126	7E	0111 1110	=	Equals sign

Decimal	Hex	Binary	Code	Code description
127	7F	0111 1111	"	Quotation mark
128	80	1000 0000		
129	81	1000 1001	a	a
130	82	1000 1010	b	b
131	83	1000 1011	c	c
132	84	1000 1100	d	d
133	85	1000 0101	e	e
134	86	1000 0110	f	f
135	87	1000 0111	g	g
136	88	1000 1000	h	h
137	89	1000 1001	i	i
138	8A	1000 1010		
139	8B	1000 1011		
140	8C	1000 1100		
141	8D	1000 1101		
142	8E	1000 1110		
143	8F	1000 1111		
144	90	1001 0000		
145	91	1001 0001	j	j
146	92	1001 0010	k	k
147	93	1001 0011	l	l
148	94	1001 0100	m	m
149	95	1001 0101	n	n
150	96	1001 0110	o	o
151	97	1001 0111	p	p
152	98	1001 1000	q	q
153	99	1001 1001	r	r
154	9A	1001 1010		
155	9B	1001 1011		
156	9C	1001 1100		
157	9D	1001 1101		
158	9E	1001 1110		
159	9F	1001 1111		
160	A0	1010 0000		
161	A1	1010 0001	~	Tilde
162	A2	1010 0010	s	s
163	A3	1010 0011	t	t
164	A4	1010 0100	u	u
165	A5	1010 0101	v	v

Decimal	Hex	Binary	Code	Code description
166	A6	1010 0110	w	w
167	A7	1010 0111	x	x
168	A8	1010 1000	y	y
169	A9	1010 1001	z	z
170	AA	1010 1010		
171	AB	1010 1011		
172	AC	1010 1100		
173	AD	1010 1101		
174	AE	1010 1110		
175	AF	1010 1111		
176	B0	1011 0000		
177	B1	1011 0001		
178	B2	1011 0010		
179	B3	1011 0011		
180	B4	1011 0100		
181	B5	1011 0101		
182	B6	1011 0110		
183	B7	1011 0111		
184	B8	1011 1000		
185	B9	1011 1001		
186	BA	1011 1010		
187	BB	1011 1011		
188	BC	1011 1100		
189	BD	1011 1101		
190	BE	1011 1110		
191	BF	1011 1111		
192	C0	1100 0000	{	Opening brace
193	C1	1100 0001	A	A
194	C2	1100 0010	B	B
195	C3	1100 0011	C	C
196	C4	1100 0100	D	D
197	C5	1100 0101	E	E
198	C6	1100 0110	F	F
199	C7	1100 0111	G	G
200	C8	1100 1000	H	H
201	C9	1100 1001	I	I
202	CA	1100 1010		
203	CB	1100 1011		
204	CC	1100 1100		

Decimal	Hex	Binary	Code	Code description
205	CD	1100 1101		
206	CE	1100 1110		
207	CF	1100 1111		
208	D0	1101 0000	}	Closing brace
209	D1	1101 0001	J	J
210	D2	1101 0010	K	K
211	D3	1101 0011	L	L
212	D4	1101 0100	M	M
213	D5	1101 0101	N	N
214	D6	1101 0110	O	O
215	D7	1101 0111	P	P
216	D8	1101 1000	Q	Q
217	D9	1101 1001	R	R
218	DA	1101 1010		
219	DB	1101 1011		
220	DC	1101 1100		
221	DD	1101 1101		
222	DE	1101 1110		
223	DF	1101 1111		
224	E0	1110 0000	\	Reverse slant
225	E1	1110 0001		
226	E2	1110 0010	S	S
227	E3	1110 0011	T	T
228	E4	1110 0100	U	U
229	E5	1110 0101	V	V
230	E6	1110 0110	W	W
231	E7	1110 0111	X	X
232	E8	1110 1000	Y	Y
233	E9	1110 1001	Z	Z
234	EA	1110 1010		
235	EB	1110 1011		
236	EC	1110 1100		
237	ED	1110 1101		
238	EE	1110 1110		
239	EF	1110 1111		
240	F0	1111 0000	0	0
241	F1	1111 0001	1	1
242	F2	1111 0010	2	2
243	F3	1111 0011	3	3
244	F4	1111 0100	4	4
245	F5	1111 0101	5	5
246	F6	1111 0110	6	6
247	F7	1111 0111	7	7
248	F8	1111 1000	8	8
249	F9	1111 1001	9	9
250	FA	1111 1010	|	
251	FB	1111 1011		
252	FC	1111 1100		
253	FD	1111 1101		
254	FE	1111 1110		
255	FF	1111 1111	eo	

The main difference between the ASCII code and EBCDIC code is the number of bits that they use to represent each character. EBCDIC uses 8 bits per character while the ASCII standard only used 7, due to concerns that using 8 bits for characters that can be represented with 7 is much less efficient.

2.7 MODULO 2 SYSTEM

In computing, the modulo operation finds the remainder after division of one number by another (called the modulus of the operation). The modulo operation commonly expressed as a "%" operator, is a useful operation in data coding. Modulo is the remainder of a division operation between two numbers. In division, we typically have the equation,

$$S = n \times d + r$$

where, S = dividend
d = divisor
n = quotient
r = remainder

From this equation, we have two equations,

$$S \div d = n$$
$$S \% d = r$$

Note that r must be smaller than d.

Modulo-2 arithmetic is an arithmetic system where every result is taken modulo-2. The result of the operation is 1 if the result is odd and 0 if the result is even. Since we are dealing with individual bits in data coding, most of the operands will be a 1 or 0. In modulo 2 arithmetic, 2 is the base (modulus) and there is no numbers other than 0 and 1. Any higher number mod 2 is obtained by dividing it by 2 and taking the remainder.

Mod-2 addition

+	0	1
0	0	1
1	1	0

Mod-2 multiplication

X	0	1
0	0	0
1	0	1

Addition and subtraction in binary arithmetic modulo 2 is equivalent t XOR.

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Mod-2 division

```
1011 ) 1001011 ( 1010 ← Quotient
       1011
       ------
       001001
         1011
         ----
         00101  ← Remainder
```

2.8 BINARY CODED DECIMAL AND EXCESS-3 CODE

2.8.1 Binary Coded Decimal (BCD)

The 8-4-2-1 code or simply the BCD code is composed of four b representing the decimal digits 0 through 9, 8, 4, 2 and 1 are weights the four bits (2^3, 2^2, 2^1, 2^0) of each decimal digit similar to straight bina number system. The ease of conversion between 8-4-2-1 code number a familiar decimal numbers is the main advantage of this code. The BC code means that each decimal digit is represented by a binary code of fo bits. Clearly only the 4-bit binary numbers from 0000 through 1001 a used. The BCD code does not use the numbers 1010, 1011, 1100, 1101, 11 and 1111. In other words, only 10 of the sixteen (2^4) possible 4-bit bina code groups are used. If any of the forbidden 4 bit number ever occurs a machine using the BCD code, it is usually an indication that an error h occurred.

Any decimal number can be expressed BCD code by replacing ea decimal digit by the appropriate 4-bit combinations. Ten bina combinations that represent the ten decimal digits are shown in table below

BCD code	Decimal number
0000	0
0001	1
0010	2
0011	3
0100	4

BCD code	Decimal number
0101	5
0110	6
0111	7
1000	8
1001	9

Conversely, a BCD number can be easily converted into a decimal by dividing the coded number into groups of 4-bits (starting with LSB) and then writing down the decimal digit represented by each four bit group. It is very important to realize that BCD is not another number system like binary, octal, decimal and hexadecimal. It is, in fact, the decimal system with each decimal digit encoded in its binary equivalent. It is also important to note that a BCD number is not the same number as a straight binary number. A straight binary code takes the complete decimal number and represents it in binary while the BCD code converts each decimal digit to binary individually.

BCD Addition

BCD code is a numerical code, and many applications need that arithmetic operations be carried out. Addition is the most important operations because the other three operations (subtraction, multiplication and division) can be accomplished using addition. For adding two BCD numbers.

i) Add the two numbers using the rules for binary addition.

ii) If a four-bit is equal to 9 or less than 9, it is a valid BCD number.

iii) If a four bit sum exceeds 9 or if a carry out of the group is generated, it is an invalid result. Add 6 $(0110)_2$ to the four bit sum. If a carry results when $(0110)_2$ is added, simply add the next four bit group.

BCD subtraction

For subtracting the two numbers:

i) Find the 9's complement of the decimal subtrahend and encode that number in BCD code.

ii) The resulting BCD number is added as described above.

Example 32

Convert the 256 decimal number to its BCD equivalent.

Solution:

$(256)_{10} = (001001010110)_{BCD}$

Example 33

Convert $(99.9)_{10}$ to BCD equivalent.

Solution:

$99.9 = (10011001.1001)_{BCD}$

Example 34

Add 96 and 56 BCD numbers.

Solution:

$(96)_{10} = (10010110)_{BCD}$

$(56)_{10} = (0101\ 0110)_{BCD}$

```
  1001  0110
+ 0101  0110
------------
  1110  1100
  0110  0110
------------
0001  0101  0010
  1     5     2
```

Both groups are invalid BCD numbers being > 9 and therefore add 6(0110) to both groups.

Thus, sum of $(96)_{10}$ and $(56)_{10} = (152)_{10}$ or $(0001\ 0101\ 0010)_{BCD}$

Example 35

Subtract 748 from 983.

Solution:

983 = $(1001\ 1000\ 0011)_{BCD}$

-748 = 251 (9's complement of 748) = $(0010\ 0101\ 0001)_{BCD}$

```
  1001  1000  0011
+ 0010  0101  0001
------------------
  1011  1101  0100
+ 0110  0110
------------------
①0010  0011  0100
  └──────────→ 1
------------------
 0010  0011  0101
   2     3     5
```

Left two groups are invalid BCD numbers being > 9

Add carry 0001 to next group and add end around entry.

Thus, $(983)_{10} - (748)_{10} = (235)_{10}$ or $(0010\ 0011\ 0101)_{BCD}$.

Example 36

Add the following BCD numbers

a) 0100 and 0010

Solution:

```
 0100
+0010
------
 0110
```

b) 0101 0110 and 0011 0010

Solution:

```
 0101  0110
+0011  0010
-----------
 1000  1000
```

c) 1001 and 0101

Solution:

```
  1001
+ 0101
------
  1110   Invalid as BCD number exceeds 9
+ 0110   Add 6
------
0001 0100
```


d) 0011 0110 and 0001 0101

Solution:

```
  0011  0110
+ 0001  0101
------------
  0100  1011
+ 0110
------------
  0101  0001
```

Right group is invalid as BCD number exceeds 9

Add 6 in invalid group (add carry 0001 to next group)

2.8.2 Excess-3 Code

The excess-3 code, abbreviated as XS-3, is an important 4-bit code sometimes used with binary coded decimal numbers. It possesses advantages in certain arithmetic operations. The excess-3 code for a decimal number can be obtained in the same manner as BCD except that 3 is added to each decimal digit before encoding it in binary. For example, to encode the decimal digit 5 into excess-3 code, we must first add 3 to obtain 8. The digit 8 is encoded in its equivalent 4-bit binary code 1000. Since no definite weights can be assigned to the four digit positions, excess-3 is an unweighted code. Excess-3 codes for decimal digit 0 through 9 are given in table below.

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

The noteworthy point from the table is that both codes (BCD and excess-3) use only 10 of the 16 possible 4-bit code groups. The excess-3 code however, does not use the same code groups. For excess-3 code, the invalid code groups are 0000, 0001, 0010, 1101, 1110 and 1111.

The key feature of the excess-3 code is that it is self-complementing code. It means that 1's complement of the coded number yields 9's complement of the number itself. *For example;* excess-3 code of decimal 5 is 1000, its 1's complement is 0111, which is excess-3 code for decimal 4, which is 9's complement of 5. It should be noted that the 1's complement is easily produced with digital logic circuits by simply inverting each bit. The self complementing property makes the excess-3 code useful in some arithmetic operations because subtraction can be performed using the 9's complement method.

Example 37

Encode $(2345)_{10}$ in BCD and excess-3 code.

Solution:

Decimal number:	2	3	4	5
	↓	↓	↓	↓
BCD code:	0010	0011	0100	0101
	+ 0011	+ 0011	+ 0011	+ 0011
	(Add 3 or 0011 in each BCD)			
Excess-3 code:	0101	0110	0111	1000

Example 38

Encode $(1236)_{10}$ to excess-3 code

Solution:

Decimal number:	1	2	3	6
	+ 3	+ 3	+ 3	+ 3
	4	5	6	9
Excess-3 code:	0100	0101	0110	1001

Example 39

What is the decimal equivalent of the binary number (1000 1011 011 represented in excess-3 code?

Solution:

Excess-3 code:	1000	1011	0111	
	− 0011	0011	0011	Subtract 0011 in each excess-3 code
	0101	0101	0110	
	↓	↓	↓	
Decimal number:	5	8	4	

Hence, $(1000\ 1011\ 0111)_{\text{Excess-3 code}} = (584)_{10}$

Example 40

Convert $(58.43)_{10}$ to excess-3 code

Solution:

Decimal number:	5	8 .	4	3
Add 3 to each bit:	+3	+3 .	+3	+3
	8	11 .	7	6

Convert above sun into 4-bit binary equivalent, 4 bit binary equivalent:

1000 1011 · 0111 0110

Hence, Excess-3 code of $(367)_{10}$ = (10001011.01110110)

2.9 PARITY METHOD FOR ERROR DETECTION

Binary information may be transmitted through some form o communication medium such as wires or radio waves or optic cables et Any external noise introduced into a physical communication mediu changes bit values from 0 to 1 or vice versa. An error detection code ca

be used to detect errors during transmission. The detected error cannot be corrected but its presence is indicated. One of the most common ways to achieve error detection is by means of parity bit. A parity bit is an extra bit included with a message to make the total number of 1's transmitted either odd or even. A message of four bits and a parity is adopted, the P bit is chosen such that the total constitute the message and P. If the even parity is adopted, the P bit is chosen so that the total number of 1's in the five bits is even. In a particular situation, one or the other parity is adopted, with even parity being more parity is adopted, with even parity being more common.

Table: Parity Bit

Odd Parity	
Message	**P**
0000	1
0001	0
0010	0
0011	1
0100	0
0101	1
0110	1
0111	0
1000	0
1001	1
1010	1
1011	0
1100	1
1101	0
1110	0
1111	1

Even Parity	
Message	**P**
0000	0
0001	1
0010	1
0011	0
0100	1
0101	0
0110	0
0111	1
1000	1
1001	0
1010	0
1011	1
1100	0
1101	1
1110	1
1111	0

The parity bit is helpful in detecting errors during the transmission of information from one location to another. This is done in the following manner. An even parity bit is generated in the sending end for each message transmission. The message, together with the parity bit is transmitted to its destination. The parity of the received data is checked in the receiving end. If the parity of the received information is not even, it means that atleast one bit has changed value during the transmission. The method detects one, three or any odd combination of errors in each message that is transmitted. An even combination of errors is undetected. Additional error detection schemes may be needed to take care of an even combination of errors.

What is done after an error is detected depends on the particular application. One possibility is to request retransmission of the message on

the assumption that the error was random and will not occur again. Thu
if the receiver detects a parity error, it sends back an acknowledge messa
The sending end will respond to a previous error by transmitting t
message again until the correct parity is received. If, after a number
attempts, the transmission is still in error, a message can be sent to
human operator to check for malfunctions in the transmission path.

2.10 NINE'S (9'S) AND TEN'S (10'S) COMPLEMENTS

The 9's complement of a decimal number is obtained by subtracting ea
digit of that decimal number form 9. The 10's complement of a decim
number is obtained by adding a 1 to its 9's complement.

To perform decimal subtraction using the 9's complement method, obta
the 9's complement of the subtrahend and add it to the minuend. Call t
indicates that the answer is positive. Add the carry is called the e
around carry. If there is no carry, it indicates that the answer is negati
and the intermediate result is its 9's complement. Take the 9's compleme
of this result and place a negative sign in front to get the answer.

To perform decimal subtraction using the 10's complement metho
obtain the 10's complement of the subtrahend and add it to the minue
If there is a carry, ignore it. The presence of the carry indicates that t
answer is positive; the result obtained is itself the answer. If there is
carry, indicates that the answer is negative and the result obtained to
10's complement. Obtain the 10's complement of the result and place
negative sign in front get the answer.

Example 41

Find the 9's complement of the following decimal numbers.

a) 3465 b) 782.54 c) 4526.075

Solution:

a)
$$\begin{array}{r} 9999 \\ -3465 \\ \hline 6534 \end{array}$$ (9's complement of 3465)

b)
$$\begin{array}{r} 999.99 \\ -782.54 \\ \hline 217.45 \end{array}$$ (9's complement of 782.54)

c)
$$\begin{array}{r} 9999.999 \\ -4526.075 \\ \hline 5473.924 \end{array}$$ (9's complement of 4526.075)

Example 42

Find the 10's complement of the following decimal numbers.

a) 4069 b) 1056.074

Solution:

a)
$$\begin{array}{r} 9999 \\ -4069 \\ \hline 5930 \\ +1 \\ \hline 5931 \end{array}$$ (9's complement of 4069) (add 1) (10's complement of 4069)

b)
$$\begin{array}{r} 9999.999 \\ -1056.074 \\ \hline 8943.925 \\ +1 \\ \hline 8943.926 \end{array}$$ (9's complement of 1056.074) (add 1) (10's complement of 1056.074)

Example 43

Subtract the following numbers using the 9's complement method.

a) 745.81 – 436.62 b) 436.62 – 745.81

Solution:

a)
$$\begin{array}{r} 745.81 \\ -436.62 \\ \hline 309.19 \end{array} \Rightarrow \begin{array}{r} 745.81 \\ +563.37 \\ \hline 1309.18 \\ +1 \\ \hline 309.19 \end{array}$$ (9's complement) (Intermediate result) (End arround carry) (∴ Answer)

b)
$$\begin{array}{r} 436.62 \\ -745.81 \\ \hline -309.19 \end{array} \Rightarrow \begin{array}{r} 436.62 \\ +254.16 \\ \hline 690.80 \end{array}$$ (9's complement) (Intermediate result with no carry)

The 9's complement of 690.80 is 309.19.

Hence answer is - 309.19

Example 44

Subtract the following using the 10's complement method.

a) 2928.54 – 416.73 b) 416.73 – 2928.54

Solution:

a)
$$\begin{array}{r} 2928.54 \\ -0416.73 \\ \hline 2511.81 \end{array} \Rightarrow \begin{array}{r} 2928.54 \\ +9583.27 \\ \hline \textcircled{1}2511.81 \end{array}$$ (10's complement) (Ignore carry)

b)
$$\begin{array}{r} 0416.73 \\ -2928.54 \\ \hline -2511.81 \end{array} \Rightarrow \begin{array}{r} 0416.73 \\ +7071.46 \\ \hline 7488.19 \end{array}$$ (10's complement) (No carry)

The 10's complement of 7488.19 is 2511.81

Hence answer is -2511.81

Example 50

Convert the following: $(5621.125)_{10}$ to ($)_8$

Solution:

Here;

Integer **Fraction**

8	5621	5
8	702	6
8	87	7
8	10	2
	1	

$0.125 \times 8 = 0$ with a carry 1

$\therefore \quad (5621.125)_{10} = (12765.1)_8$

Example 51

Convert the following: $(375.37)_8 = (\ )_2$

Solution:

Here;

$$(375.37)_8 = \underset{\downarrow}{3}\ \underset{\downarrow}{7}\ \underset{\downarrow}{5} \cdot \underset{\downarrow}{3}\ \underset{\downarrow}{7}$$

$$011\ 111\ 101 \quad 011\ 111$$

$\therefore \quad (375.37)_8 = (011111101.011111)_2$

Example 52

Convert the following: $(101010011.1101)_2$ to ($)_8$

Solution:

Here;

$$(101010011.1101)_2 = \underset{\downarrow}{101}\ \underset{\downarrow}{010}\ \underset{\downarrow}{011} \cdot \underset{\downarrow}{110}\ \underset{\downarrow}{100}$$

$$5\ 2\ 3 \quad 6\ 4$$

$\therefore \quad (101010011.1101)_2 = (523.64)_8$

Example 53

Convert the following: $(131.F2)_{16}$ to ($)_{10}$

Solution:

Here;

$$(131.F2)_{16} = 1 \times 16^2 + 3 \times 16^1 + 1 \times 16^0 + 15 \times 16^{-1} + 2 \times 16^{-2}$$

$$= 256 + 48 + 1 + \frac{15}{16} + \frac{2}{256}$$

$$= 305 + 0.9375 + 0.0078125 = (305.9453125)_{10}$$

Example 54

Convert the following: $(374.37)_{10}$ to ($)_{16}$

Solution:

Here;

Integer 374

16	374	6
16	23	7
	1	

Fraction 0.37

$0.37 \times 16 = 5.92 = 0.92$ with a carry 5
$0.92 \times 16 = 14.72 = 0.72$ with a carry 14
$0.72 \times 16 = 11.52 = 0.52$ with a carry 11
$0.52 \times 16 = 8.32 = 0.32$ with a carry 8

so, $(374.37)_{10} = (176.5EB)_{16}$

Example 55

Convert the following: $(11011.011)_{10}$ to $(\ )_8$

Solution:

Here;

Integer

16	11011	3
16	688	0
16	43	11
	2	

Fraction

$0.011 \times 16 = 0.176$ with a carry 0
$0.176 \times 16 = 0.816$ with a carry 2
$0.816 \times 16 = 0.056$ with a carry 13
$0.056 \times 16 = 0.896$ with a carry 0
$0.896 \times 16 = 0.336$ with a carry 14

so, $(11011.011)_{10} = (2B03.02D0E)_{16}$

Example 56

Convert the following: $(A6B.F5)_{16}$ to $(\ )_2$

Solution:

Here;

$$(A6B.F5)_{16} = \underset{1010}{A}\ \underset{0110}{6}\ \underset{1011}{B}\ .\ \underset{1111}{F}\ \underset{0101}{5}$$

$\therefore (A6B.F5)_{16} = (101001101011.11110101)_2$

Example 57

Convert the following: $(F2A4)_{16}$ to $(\ )_8$

Solution:

Here;

$$(F2A4)_{16} = \underset{1111}{F}\ \underset{0010}{2}\ \underset{1010}{A}\ \underset{0100}{4}$$

$$\underset{1}{001}\ \underset{7}{111}\ \underset{1}{001}\ \underset{2}{010}\ \underset{4}{100}\ \underset{4}{100}$$

so, $(F2A4)_{16} = (171244)_8$

Example 58

Convert the following: $(1E9C)_{16}$ to $(\ )_8$

Solution:

Here;

$$(1E9C)_{16} = \underset{0001}{1}\ \underset{1110}{E}\ \underset{1001}{9}\ \underset{1100}{C}$$

$$\underset{1}{001}\ \underset{7}{111}\ \underset{2}{010}\ \underset{3}{011}\ \underset{4}{100}$$

so, $(1E9C)_{16} = (17234)_8$

Example 59

Convert the following: $(C3A.47)_H$ to $(\ )_8$

Solution:

Converting to binary; we have,

$$\underset{1100}{C}\ \underset{0011}{3}\ \underset{1010}{A}\ .\ \underset{0100}{4}\ \underset{0111}{7}$$

(1100 0011 1010 . 0100 0111)

$$\underset{6}{110}\ \underset{0}{000}\ \underset{7}{111}\ \underset{2}{010}\ .\ \underset{2}{010}\ \underset{1}{001}\ \underset{6}{110}$$

so, $(C3A.47)_H \rightarrow (6072.216)_8 \rightarrow (110000111010.01000111)_2$

Example 60

Convert the following: $(2715)_8$ to $(\ )_{16}$

Solution:

Here;

$$(2715)_8 = \underset{010}{2}\ \underset{111}{7}\ \underset{001}{1}\ \underset{101}{5}$$

$$\underset{5}{0101}\ \underset{12}{1100}\ \underset{13}{1101}$$

$$\underset{C}{12}\ \underset{D}{13}$$

so, $(2715)_8 = (5CD)_{16}$

Example 61

Convert the following: $(8.3)_9$ to $(\ )_{10}$

Solution:

Here;

$$(8.3)_9 = 8 \times 9^0 + 3 \times 9^{-1} = (8.333)_{10}$$

EXAMINATION QUESTION SOLUTIONS

1. **Describe the party method of error detection with a suitable four bit system example. Also, down its limitations.** **[2011/S]**

Solution: See the topic 2.9.

The limitations of parity method of error detection is that it cannot detect an error when an even number of bits change in the same data unit *i.e.*, it can only detect errors that change an odd number of bits.

2. **Discuss in brief about gray code.** **[2012/S, 2013/F]**

Solution: See the topic 2.4.

3. **Explain the parity method of error detection with a suitable example.** **[2012/S, 2012/F, 2017/S, 2016/S]**

Solution: See the topic 2.9.

4. **"Excess-3 code is self complementary code. Verify this statement.** **[2011/F, 2017/S, 2019/F]**

Solution:

A member in the excess-3 number system is obtained by simply adding 3 to a number in the ordinary decimal number system and writing the corresponding binary number. For example, the number 0 is represented in excess-3 as 3. This is obtained by adding 3 to 0.

Let us consider the excess-3 number 0100. This corresponds to decimal 1. The 9's complement of 1 is 8. The excess-3 equivalent of 8 is 1011. Complementing 0100 in the usual way of replacing 1's with 0's and 0's with 1's also results in 1011. We therefore conclude that direct complementing of the numbers in the excess-3 family results in the 9's complementing operation also. So, excess-3 is said to be a self-complementing code *i.e.*, members of this family complement on themselves.

5. **Define alphanumeric codes.** **[2011/S]**

Solution:

Alphanumeric codes, also called characters code are binary codes used to represent alphanumeric data. A complete alphanumeric code would include the 26 lower case letters, 26 uppercase letter, 10 numeric digits, 7 punctuation marks and anywhere from 20 to 40 other characters such as +, /, *, # and so on.

6. **Write short notes on self-complementing code.** **[2015/F, 2018/S, 2019/S]**

Solution:

A code is said to be self complementing if the code word of the 9's complement of 9-N can be obtained from the code word of N by interchanging all the 0's 1's. Therefore, in a self-complementing code, the code for 9 is the complement of the code for 0, the code for 8 is the complement of the code for 1 and so on. The 2421, 5211, 642-3, 84-2-1 and excess-3 are self complementing codes. The self complementing property is desirable in a code when the 9's complement must be found such as in

9's complement subtraction. Self complementing codes have an advantage that their logical complement is the same as the arithmetic complement. For a code to be self-complementing, the sum of all its weights must be 9. This is because whatever may be the weights, 0 is to be represented by 0000 and since in a self-complementing code, the code for 9 is the complement of the code for 0, 9 has to be represented by 1111.

Table below depicts a self-complementing code.

N	Code for N 2 4 2 1	Code for 9-N 2 4 2 1	9-N
0	0 0 0 0	1 1 1 1	9
1	0 0 0 1	1 1 1 0	8
2	0 0 1 0	1 1 0 1	7
3	0 0 1 1	1 1 0 0	6
4	0 1 0 0	1 0 1 1	5
5	1 0 1 1	0 1 0 0	4
6	1 1 0 0	0 0 1 1	3
7	1 1 0 1	0 0 1 0	2
8	1 1 1 0	0 0 0 1	1
9	1 1 1 1	0 0 0 0	0

To be self-complementing code, following conditions must be satisfied.

i) The complement of a number should be obtained from that number by replacing 1's with 0's and 0's with 1's.

ii) The sum of the number and its complement should be equal to decimal 9.

7. **"84-2-1 code is self complementing code". Justify this statement.** **[2013/S]**

Solution:

"84-2-1 code is self complementing code because the complement of this number is obtained by replacing 1's with 0's and 0's with 1's. Also the sum of the number and its complement is equal to decimal 9.

N	Code for N 8 4 2 1	Code for 9-N 8 4 2 1	9-N
0	0 0 0 0	1 1 1 1	9
1	0 1 1 1	1 0 0 0	8
2	0 1 1 0	1 0 0 1	7
3	0 1 0 1	1 0 1 0	6
4	0 1 0 0	1 0 1 1	5
5	1 0 1 1	0 1 0 0	4
6	1 0 1 0	0 1 0 1	3
7	1 0 0 1	0 1 1 0	2
8	1 0 0 0	0 1 1 1	1
9	1 1 1 1	0 0 0 0	0

Let us consider the 84-2-1 number 0111. This corresponds to decimal
The 9's complement of 1 is 8. The 84-2-1 equivalent of 8 is 100
Complementing 0111 in the usual way of replacing 1's with 0's and
with 1's also results in 1000. We therefore conclude that dire
complementing of the numbers in the 84-2-1 family results in the
complementing operation also. So, 84-2-1 code is said to be a se
complementing code.

8. What are the different types of binary codes? Explain each them in brief. **[2017/**

Solution:

Different types of binary codes are;

i) Weighted binary codes
ii) Non-weighted binary codes
iii) Error-detection codes
iv) Error correcting codes
v) Alphanumeric codes

1. Weighted binary codes

If each position of a number represents a specific weight then the codi scheme is called weighted binary code. In such coding, the bits a multiplied by their corresponding individual weight and then the sum these weighted bits gives the equivalent decimal digit.

Types of weighted binary codes

a) BCD code or 8421 code
b) 84-2-1 code
c) 2421 code

2. Non-weighted code

These codes are not positionally weighted. It basically means that ea position of the binary number is not assigned a fixed value.

Types of Non-weighted code

a) Excess-3 code
b) Gray code

3. Error detection code

Error detecting codes are usually block codes, where the message divided into fixed sized blocks of bits to which redundant bits are adde for error detection. Error detection involves checking whether any err has occurred or not. The number of error bits and the type of error doe not matter.

4. Error correcting code

An error correction code or forward error correction code is a process adding redundant data or parity data to a message such that it can recovered by a receiver even when a number of errors were introduce either during the process of transmission or on storage. For correction errors, hamming code are used.

5. Alphanumeric codes

An alphanumeric code is a binary code of a group of elements consisting of ten decimal digits, the 26 letters of the alphabet (both in uppercase and lowercase) and a certain number of special symbols such as #, /, % etc. The total number of elements is an alphanumeric code is greater than 36. Therefore, it must be coded with a minimum number of 6 bits ($2^6 = 64$ but $2^5 = 32$ is in sufficient). It is used in many computers to represent alphanumeric characters and symbols internally and hence can be called internal code. Frequently there is a need to represent more than 64 characters, including the lowercase letters and special control characters. For this reason, ASCII, EBCDIC and Hollerith codes are normally used.

9. How can you find the r's complement using (r – 1)'s complement? Explain with example. **[2017/S]**

Solution:

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulations. There are two types of complements for each number system of base -r: the r's complement and the (r – 1)'s complement.

If a positive number N is given in base r with an integer part of n digits, the r's complement of N is given as $r^n - N$ for $N \neq 0$ and 0 for $N = 0$. If a positive number N is given in base r with an integer part of n digits and a-fraction part of m digits, then the (r – 1)'s complement of N is given as $(r^n - r^{-m} - N)$ for $N \neq 0$ and 0 for calculated (r – 1)'s complement.

For example;

Find the 7's and 8's complement of the number $(563)_8$.

Solution:

Here, base or radix (r) = 8

Since 7 is the largest digit in the number system, subtract each digit of given number from 7, *i.e.*, if it's a three digit number, subtract from 777.

$$\begin{array}{r} 7\ 7\ 7 \\ -\ 5\ 6\ 3 \\ \hline 2\ 1\ 4 \end{array}$$

$\therefore (214)_8$ is the 7's complement of $(563)_8$. Now to find r's complement *i.e.*, 8's complement add '1' to the result of 7's complement number.

$$\begin{array}{r} 2\ 1\ 4 \\ +\ 1 \\ \hline 2\ 1\ 5 \end{array}$$

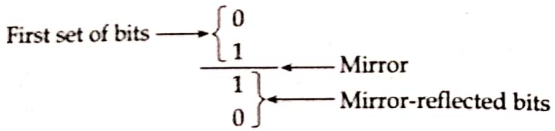
$\therefore (215)_8$ is the 8's complement of $(563)_8$. In this way, r's complement can be found using (r – 1)'s complement.

10. 'Gray code is called as reflected code. 'Justify you answer with appropriate illustration. **[2017/S]**

Solution: See the topic 2.4.

The gray code (also known as the mirror code) is a binary code whose adjacent members differ only in one position from each other. For decimal numbers 0 and 1, the gray code numbers differ in the least significant bit position. Similarly, gray codes for decimals 1 and 2 differ in the second bit position from right and for 2 and 3, the difference occurs in the least significant bit position. Thus, gray code has the property that any two adjacent numbers of the code differ in one bit position only.

Gray code can be constructed by its mirror reflection property. For example, a two bit gray code can be constructed as illustrated below. First, we write the bit 0 and immediately below it write 1. Then we draw a horizontal line below it and write the bits 0 and 1 below this line, assuming the horizontal line to act as a mirror. Thus, we get the structure as shown below;



By adding one 0 to the bits above the horizontal line and one 1 each to the bits below the horizontal line, we get the final form of the 2-bit gray code structure as under:

0's added to the bits above the mirror $\begin{matrix}0 & 0\\ 0 & 1\end{matrix}$

1's added to the bits below the mirror $\begin{matrix}1 & 1\\ 1 & 0\end{matrix}$

Now we have the numbers 00, 01, 11 and 10 in the gray code for the decimal numbers 0, 1, 2 and 3 respectively. This process can be continued to construct any number in the gray code.

11. Explain (r – 1)'s complement with example. **[2019/F]**

Solution:

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulations. If a positive number N is given in given in base r with an integer part of n digits and a fraction part of m digits, then the (r – 1)'s complement of N is given as $(r^n - r^{-m} - N)$ for $N \neq 0$ and 0 for N = 0. The following examples will clarify the definition:

The 9's complement of $(23450)_{10}$ is $10^5 - 10^0 - 23450 = 76549$

Since there is no fraction part, $10^{-m} = 10^0 = 1$

The 9's complement of $(0.3245)_{10}$ is $10^0 - 10^{-4} - 0.3245 = 0.6754$

Since there is no integer part, $10^n = 10^0 = 1$

The 9's complement of $(23.324)_{10}$ is $10^2 - 10^{-3} - 23.324 = 76.675$

Now,

If we consider a binary number system, then r = 2, *i.e.*, (r – 1) = 1

The 1's complement of $(10110)_2$ is $(2^5 - 1)_{10} - (10110)_2 = 01001$

The 1's complement of $(0.1011)_2$ is $(1 - 2^{-4})_{10} - (0.1011)_2 = 0.0100$

Now,

If we consider an octal number system, then r = 8 *i.e.*, (r – 1) = 7

The 7's complement of $(2350)_8$ is $8^4 - 8^0 - 2350_8 = 4095_{10} - 1256_{10}$

$= 2839_{10} = 5427_8$

From the above example, it is clear that to find the 9's complement of a decimal number each of the digits can be separately subtracted from 9. The 1's complement of a binary number can be obtained by changing 1's into 0's and 0's into 1's. Similarly, to find the 7's complement of a decimal number, each of the digits can be separately subtracted from 7. Again, to find the 15's complement of a decimal number each of the digital can be separately subtracted from 15.

12. Perform the operation as indicated. **[2011/S]**

i) $(1010)_{Gray} = (.....)_2$

ii) $(2EB.7A)_{16} = (.....)_{10}$

iii) $(225.225)_{10} = (.....)_2$

iv) $(237.5)_8 = (.....)_{16}$

Solution:

Here,

Step -1: The LSB is same.

1 0 1 0 Gray code

↓

1 Binary code

Step-2: Add the last binary digit just generated to the gray digit in the next position. Discard carry.

1 0 1 0

+ ↓

1 1

Step 3: Add the last binary digit generated to the next gray digit.

1 0 1 0

+ ↓

1 1 0

Step 4: Add the last binary digit generated to the next gray digit.

1 0 1 0

+ ↓

1 1 0 0

Thus conversion is complete and result is $(1100)_2$

ii) Here, $(2EB{\cdot}7A)_{16}$
$= 2 \times 16^2 + 14 \times 16^1 + 11 \times 16^0 + 7 \times 16^{-1} + 10 \times 16^{-2}$
$= 512 + 224 + 11 + 0.4375 + 0.03906$
$= (748.47656)_{10}$

iii) Here, $(225.225)_{10}$

Integer 225

2	225	1
2	112	0
2	56	0
2	28	0
2	14	0
2	7	1
2	3	1
	1	

Fraction 0.225

$0.225 \times 2 = 0.45$ with carry 0
$0.45 \times 2 = 0.9$ with carry 0
$0.9 \times 2 = 0.8$ with carry 1
$0.8 \times 2 = 0.6$ with carry 1
$0.6 \times 2 = 0.2$ with carry 1
$0.2 \times 2 = 0.4$ with carry 0
$0.4 \times 2 = 0.8$ with carry 0
$0.8 \times 2 = 0.6$ with carry 1

Thus, $(225.225)_{10}$ $(11100001.00111001)_2$

iv) Here, $(237.5)_8$ =

2	3	7	·	5
↓	↓	↓		↓
010	011	111	·	101
	↓			
1001	1111	·	1010	
↓	↓		↓	
9	15	·	10	
↓	↓		↓	
9	F	·	A	

$\therefore$ $(237.5)_8 = (9F{\cdot}A)_{16}$

13. Compute any four of the following as indicated. [2012/S]

i) $(11101)_{Gray} = (\quad)_2$
ii) $(706)_8 = (\quad)_{16}$
iii) $(512)_{10} = (\quad)_8$
iv) $(110110.101)_2 = (\quad)_{10}$
v) $(3FAFE{\cdot}1BA)_{16} = (\quad)_{10}$

Solution:

i) Here, $(11101)_{Gray}$

1	1	1	0	1
↓ +	↓ +	↓ +	↓ +	↓
1	0	1	1	0

Gray code

Binary code

Hence, $(11101)_{Gray} = (10110)_2$

ii) Here, $(706)_8$ =

7	0	6
↓	↓	↓
111	000	110
	↓	
0001	1100	0110
↓	↓	↓
1	12	6

$\therefore$ $(706)_8 = (1C6)_{16}$

iii) Here, $(512)_{10}$

8	512	0
8	64	0
8	8	0
	1	

$\therefore$ $(512)_{10} = (1000)_8$

iv) Here, $(110110.101)_2$
$= 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1}$
$+ 0 \times 2^{-2} + 1 \times 2^{-3}$
$= 32 + 16 + 0 + 4 + 2 + 0 + 0.5 + 0 + 0.125$
$= (54.625)_{10}$

v) Here, $(3FAFE.1BA)_{16}$
$= 3 \times 16^4 + 15 \times 16^3 + 10 \times 16^2 + 15 \times 16^1 + 14 \times 16^0 + 1 \times 16^{-1}$
$+ 11 \times 16^{-2} + 10 \times 16^{-3}$
$= 196608 + 61440 + 2560 + 240 + 14 + 0.0625 + 0.04296$
$+ 0.002441$
$= (260862.1079)_{10}$

14. Determine the value of base x if $(321)_x = (57)_8$. [2013/S]

Solution:

$(57)_8 = 5 \times 8^1 + 7 \times 8^0 = 40 + 7 = (47)_{10}$

Converting $(321)_x$ into decimal number, we have,

or, $3 \times x^2 + 2 \times x^1 + 1 \times x^0 = 47$

or, $3x^2 + 2x + 1 = 47$

or, $3x^2 + 2x - 46 = 0$

$\therefore$ $x = 3.596$ or, -4.263

Discarding negative value of x, we have $x = 3.596$

15. Find the value of x. [2014/S, 2016/F]

i) $(777)_x = (212)_8$
ii) $(DEC)_H = (0101)_x$
iii) $(563)_7 = (x)_3$
iv) $(100111)_{Gray} = (x)_{Binary}$

Solution:

i) Here;

$(212)_8 = 2 \times 8^2 + 1 \times 8^1 + 2 \times 8^0 = 128 + 8 + 2 = (138)_{10}$

Converting $(777)_8$ into decimal number,

or, $7 \times x^2 + 7 \times x^0 = 138$

or, $7x^2 + 7x + 7 - 138 = 0$

or, $7x^2 + 7x - 131 = 0$

$\therefore$ $x = 3.854$ or -4.854

Discarding negative value, we have $x = 3.854$

ii) Here;

$(DEC)_H = (13\ \ 14\ \ 12)_{16}$

$= 13 \times 16^2 + 14 \times 16^1 + 12 \times 16^0$

$= 3328 + 224 + 12$

$= (3564)_{10}$

Converting $(0101)_x$ into decimal number,

or, $0 \times x^3 + 1 \times x^2 + 0 \times x^1 + 1 \times x^0 = 3564$

or, $x^2 + 1 = 3564$

or, $x^2 = 3563$

$\therefore$ $x = 59.690870$

Thus value of $x = 59.690870$

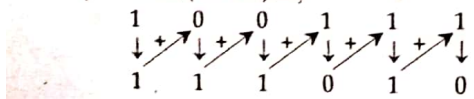
iii) Here, $(563)_7 = 5 \times 7^2 + 6 \times 7^1 + 3 \times 7^0 = 245 + 42 + 3 = (290)_{10}$

Converting to base 3,

3	290	2
3	96	0
3	32	2
3	10	1
3	3	0
	1	

i.e., value of $x = (101202)$

iv) Here, $(100\ 111)_{Grary}$



Gray code

Binary code

Hence, Value of $x = (111010)_{Binary}$

i.e., $(100111)_{Gray} = (111010)_{Binary}$

16. Perform the following conversion [2015/S]

i) $(573)_{10} = (\ \ )_{Excess-3}$

ii) $(842)_{10} = (\ \ )_{2421}$

iii) $(10101111)_{Gray} = (\ \ )_2$

iv) $(FAB)_{16} = (\ \ )_2$

Solution:

Here, $(573)_{10}$

i) **Decimal number:**

5	7	3
+3	+3	+3
8	10	6
↓	↓	↓

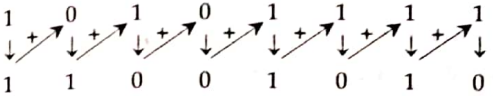
Excess-3 code: 1000 1010 0110

ii) Here, $(842)_{10}$ =

		2	4	2	1
8	:	1	1	1	0
4	:	0	1	0	0
2	:	0	0	1	0

$\therefore$ $(842)_{10} = (1110\ 0100\ 0010)_{2421}$

iii) Here, $(10101111)_{Gray}$



Gray code

Binary code

Hence, $(10101111)_{Gray} = (11001010)_2$

iv) Here, $(FAB)_{16}$

Converting each hexadecimal digit to binary number of bit, we have,

F	A	B
↓	↓	↓
1111	1010	1011

i.e., $(FAB)_{16} = (1111\ 1010\ 1011)_2$

17. Perform the conversion as indicated. [2015/F, 2016/F, 2019/S (similar)]

i) $(243)_6 = (\ ?\ )_{Excess-3}$

ii) $(816)_{10} = (\ ?\ )_{2421}$

iii) $(BE)_{16} = (\ ?\ )_2$

Solution:

i) Here, $(243)_6 = 2 \times 6^2 + 4 \times 6^1 + 3 \times 6^0 = (99)_{10}$

9	9
↓	↓
+3	+3
12	12
1100	1100

$\therefore$ $(243)_6 = (11001100)_{Excess-3}$

ii) Here, $(816)_{10} = (111000010110)_{2421}$ (Proceed as Q.N. 16 (ii))

iii) Here, $(BE)_{16}$

Converting to binary

B E

↓ ↓

1011 1110

∴ $(BE)_{16} = (10111110)_2$

18. Determine the value of base x if $(211)_x = (152)_8$. [2017]

Solution:

Here, $(152)_8 = 1 \times 8^2 + 5 \times 8^1 + 2 \times 8^0 = 64 + 40 = (106)_{10}$

Converting $(211)_x$ into decimal number,

$2 \times x^2 + 1 \times x^1 + 1 \times x^0 = 106$

$2x^2 + x + 1 - 106 = 0$

or, $2x^2 + x - 105 = 0$

∴ $x = 7$ or, -7.50

Discarding negative value, we have $x = 7$

19. Convert the following conversions. [2015]

i) $(101001.101)_2 = (?)_{10}$

ii) $(ABD)_{16} = (?)_8$

iii) $(10101101)_2 = (?)_{Gray}$

iv) $(17.351)_8 = (?)_{16}$

Solution:

i) Here, $(101001.101)_2$

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

$= 32 + 0 + 8 + 0 + 0 + 1 + 0.5 + 0 + 0.125$

$= (41.625)_{10}$

ii) Here, $(ABD)_{16}$

A B D

↓ ↓ ↓

1010 1011 1101

↓

101 010 111 101

↓ ↓ ↓ ↓

5 2 7 5

∴ $(ABD)_{16} = (5275)_8$

iii) Here, $(10101101)_2$

The left most gray digit is the same as the left-most binary digit.

1 0 1 0 1 1 0 1 : Binar

↓

1

Add the left-most digit to the adjacent one and discard carry.

1 + 0 1 0 1 1 0 1

↓

1 1

Add next adjacent pair.

1 0 + 1 0 1 1 0 1

↓

1 1 1

Add next adjacent pair.

1 0 1 + 0 1 1 0 1

↓

1 1 1 1

Add next adjacent pair.

1 0 1 0 + 1 1 0 1

↓

1 1 1 1 1

Add next adjacent pair.

1 0 1 0 1 + 1 0 1

↓

1 1 1 1 1 0

Add next adjacent par.

1 0 1 0 1 1 + 0 1

↓

1 1 1 1 1 0 1

Add next adjacent pair,

1 0 1 0 1 1 0 + 1

↓

1 1 1 1 1 0 1 1

∴ $(10101101)_2 = (11111011)_{Gray}$

OR,

1 0 1 0 1 1 0 1 : Binary

↓ + ↓ + ↓ + ↓ + ↓ + ↓ + ↓ + ↓

1 1 1 1 1 0 1 1 : Grey

∴ $(10101101)_2 = (11111011)_{Gray}$

iv) Here, $(175.351)_8$

1 7 5 · 3 5 1

↓ ↓ ↓ ↓ ↓ ↓

001 111 101 · 011 101 001

↓

0111 1101 · 0111 0100 1000

↓ ↓ ↓ ↓ ↓

7 13 · 7 4 8

∴ $(175.351)_8 = (7D \cdot 748)_{16}$

20. Perform the conversion as indicated. [2018/S]

i) $(235)_6 = (\quad)_{Excess-3}$

ii) $(369)_{10} = (\quad)_{2421}$

iii) $(BCA)_{16} = (\quad)_2$

Solution:

i) Here, $(235)_6 = 2 \times 6^2 + 3 \times 6^1 + 5 \times 6^0 = (95)_{10}$

so,

9	5
+ 3	+ 3
↓	↓
12	8
↓	↓
1100	1000

$\therefore \quad (235)_6 = (1100\ 1000)_{Excess-3}$

ii) Here, $(369)_{10} = (001101101111)_{2421}$

iii) Here, $(BCA)_{16} =$

B	C	A
↓	↓	↓
1011	1100	1010

$\therefore \quad (BCA)_{16} = (101111001010)_2$

21. Use 2's complement to subtract the following [2011/

i) $(100)_2 - (110000)_2$

ii) $(7850)_{10} - (7650)_{10}$

Solution:

i) Here, $(100)_2 - (110000)_2$

Making number of bits in both minuend and subtrahend equal,

$M = (000100)_2$

$S = (110000)_2$

2's complement of $S = (010000)_2$

Now,

$$\begin{array}{r} 000100 \\ +\ 010000 \\ \hline 010100 \end{array}$$

There is no end carry, so the answer is 2's complement of (01010

so, 2's complement of $(010100)_2 = (101100)_2$

$\therefore \quad (100)_2 - (110000)_2 = (101100)_2$

NOTE: 2's complement = 1's complement + 1

ii) Here, $(7850)_{10} - (7650)_{10}$

Converting base 10 into binary form,

$(7850)_{10} = (1111010101010)_2 = M$

$(7650)_{10} = (1110111100010)_2 = S$

2's complement of $S = (0001000011110)_2$

Now,

$$\begin{array}{r} 1111010101010 \\ +\ 0001000011110 \\ \hline \textcircled{1}0000011001000 \end{array}$$

There is end carry, so we neglect it and remaining is answer,

$\therefore \quad (7850)_{10} - (7650)_{10} = (0000011001000)_2 = (200)_{10}$

22. Perform the following operation using BCD method. [2012/S]

i) **689 – 354**

ii) **496 + 825**

Solution:

i) Here, 689 – 354

To subtract two numbers using BCD method, convert two numbers into BCD. But first 9's complement of subtrahend.

9's complement of 354 = 999 – 354 = 645

Now, convert it to BCD,

$(689)_{10} = (0110\ \ 1000\ \ 1001)_{BCD}$

$(645)_{10} = (0110\ \ 0100\ \ 0101)_{BCD}$

Thus,

$$\begin{array}{r} 0110\ \ 1000\ \ 1001 \\ +\ 0110\ \ 0100\ \ 0101 \\ \hline 1100\ \ 1100\ \ 1110 \end{array}$$

Since three groups are invalid BCD numbers being > 9 add 6 (0110) to them.

$$\begin{array}{r} 1100\ \ 1100\ \ 1110 \\ +\ 0110\ \ 0110\ \ 0110 \\ \hline \textcircled{1}0011\ \ 0011\ \ 0100 \end{array}$$

The end carry is added as,

$$\begin{array}{r} 0011\ \ 0011\ \ 0100 \\ +\ \ \ \ \ \ \ \ 0001 \\ \hline 0011\ \ 0011\ \ 0101 \end{array}$$

0011	0011	0101
↓	↓	↓
3	3	5

Thus $(689)_{10} - (350)_{10} = (0011\ 0011\ 0101)_{BCD} = (335)_{10}$

ii) Here, 496 + 825

Converting into BCD numbers,

$(496)_{10} = (0100\ 1001\ 0110)_{BCD}$

$(825)_{10} = (1000\ 0010\ 0101)_{BCD}$

Now,

$$\begin{array}{r} 0100\ \ 1001\ \ 0110 \\ +\ 1000\ \ 0010\ \ 0101 \\ \hline 1100\ \ 1011\ \ 1011 \end{array}$$

Here, all groups have sum exceeding 9, so adding 6 (0110) in all,

$$\begin{array}{r} 1100\ \ 1011\ \ 1011 \\ +\ 0110\ \ 0110\ \ 0110 \\ \hline (1)0011\ \ 0010\ \ 0001 \end{array}$$

Hence, $(496)_{10} + (825)_{10} = (0001\ 0011\ 0010\ 0001)_{BCD} = (1321)_{10}$

23. Use complement to subtract the following:
i) $(00101101)_2 - (10110101)_2$ [2012
ii) $(835701)_{10} - (569812)_{10}$

Solution:

i) Here, $(00101101)_2 - (10110101)_2$

$M = (00101101)_2$

$S = (00101101)_2$

2's complement of $S = (01001011)_2$

Now, adding M and 2's complement of S

$$\begin{array}{r} 00101101 \\ +\ 01001011 \\ \hline 01111000 \end{array}$$

There is no end carry, so the result is 2's complement of ab result with a minus sign.

i.e., $-(10001000)_2$

$\therefore\ (00101101)_2 - (10110101)_2 = -(10001000)_2$

ii) Here, $(835701)_{10} - (569812)_{10}$

10's complement of $(569812)_{10}$

$= (1000000 - 569812) = 430188$

so,

$$\begin{array}{r} 835701 \\ +\ 430188 \\ \hline 1265889 \end{array}$$

Since there is end carry, so after neglecting it, remaining is answer. Thus, $(835701)_{10} - (569812)_{10} = (265889)_{10}$

24. Perform the following subtraction $(34)_8 - (27)_8$. Convert the gi number into binary and perform subtraction using complement. [201

Solution:

Here, $(34)_8 - (27)_8$

$$\begin{array}{r} 34 \\ -\ 27 \\ \hline (05)_8 \end{array}$$

Now, $(34)_8 = (011100)_2$ = minuend (m)

$(27)_8 = (010111)_2$ = subtrahend (S)

2's complement of subtrahend = $(101001)_2$

so,

$$\begin{array}{r} 011\ \ 100 \\ +\ 101\ \ 001 \\ \hline (1)000\ \ 101 \end{array}$$

There is end carry, so neglect it and remaining is answer.

Hence, $(34)_8 - (27)_8 = (000101)_2 = (5)_8$.

25. Subtract $(100101)_2$ from $(1111101)_2$ using 2's complement. [2013/F]

Solution:

Here, $M = (0100101)_2$

$S = (1111101)_2$

2's complement of $S = (0000011)_2$

so,

$$\begin{array}{r} 0100101 \\ +\ 0000011 \\ \hline 0101000 \end{array}$$

Since, there is no end carry answer is 2's complement of $(0101000)_2$ with negative sign.

i.e., $-(1011000)_2$

$\therefore\ (100101)_2 - (1111101)_2 = -(1011000)_2$

26. Subtract $(7729)_{10} - (842.4)_{10}$ using 9's complement. [2013/F]

Solution:

Here, 9's complement of 0842.4 is

$9999.9 - 0842.4 = 9157.5$

so,

$$\begin{array}{r} 7729 \\ +\ 9157.5 \\ \hline 16886.5 \end{array}$$

Since there is end carry, so remove the end carry and add with last significant bit.

Thus,

$$\begin{array}{r} 6886.5 \\ +\ 1 \\ \hline (6886.6)_{10} \end{array}$$

$\therefore\ (7729)_{10} - (842.4)_{10} = (6886.6)_{10}$

27. Use 2's complement to subtract the following: [2015/F]
i) $(101)_2 - (10100)_2$
ii) $(3950)_{10} - (876)_2$
iii) $(378)_{BCD} - (256)_{BCD}$

Solution:

i) Here, $(101)_2 - (10100)_2$

$M = (00101)_2$

$S = (10100)_2$

2's complement of S = $(01100)_2$

$$\begin{array}{r} 00101 \\ +\ 01100 \\ \hline 10001 \end{array}$$

Since there is no end carry, answer is 2's complement of result with negative sign *i.e.*, $-(01111)_2$

$\therefore$ $(101)_2 - (10100)_2 = -(01111)_2$

ii) Here, $(3950)_{10} - (876)_{10}$

$M = (3950)_{10} = (111101101110)_2$

$S = (876)_{10} = (001101101100)_2$

2's complement of S = $(110010010100)_2$

Now,
$$\begin{array}{r} 111101101110 \\ +\ 110010010100 \\ \hline 1110000000010 \end{array}$$

There is end carry, so neglect it and remaining is answer

$\therefore$ $(3950)_{10} - (876)_{10} = (110000000010)_2 = (3074)_2$

iii) $(378)_{BCD} - (256)_{BCD}$

$M = (378)_{BCD} = (0.011\ 0111\ 1000)_2$

$S = (256)_{BCD}\ (0010\ 0101\ 0110)_2$

2's complement of S,

$= (1101\ 1010\ 1010)_2$

Now,
$$\begin{array}{r} 001101111000 \\ +\ 110110101010 \\ \hline 1000100100010 \end{array}$$

Since there is an end carry, neglect the end carry and remaining is answer.

so, $(378)_{BCD} - (256)_{BCD} = (000100100010)_2 = (122)_{BCD}$

28. Perform the subtraction with the following binary number using 1's complement. **[2016/S]**

i) 1010100 – 1000100

ii) 1000100 – 1010100

Solution:

i) Here, $M = (1010100)_2$

$S = (1000100)_2$

1's complement of S = 0111011

$$\begin{array}{r} 1010100 \\ +\ 0111011 \\ \hline 10001111 \end{array}$$

Since there is an end carry, so add it to remaining answer

$$\begin{array}{r} 0001111 \\ +\ 1 \\ \hline 0010000 \end{array}$$

$\therefore$ $(1010100)_2 - (100100)_2 = (0010000)_2$

ii) Here, $(100100)_2 - (1010100)_2$

$M = 1000100$

$S = 1010100$

1's complement of S = 0101011

$$\begin{array}{r} 1000100 \\ +\ 0101011 \\ \hline 1101111 \end{array}$$

Since there no end carry, so answer is complement of above result with negative sign, *i.e.*, $-(001000)_2$

$\therefore$ $(1000100)_2 - (1010100)_2 = -(00110000)_2$

29. Perform the following subtraction using 2's complement method. **[2017/S]**

i) $(11010)_2 - (10000)_2$

ii) $(1000100)_2 - (1010100)_2$

Solution:

Here,

$M = (11010)_2$

$S = (10000)_2$

2's complement of S = 10000

$$\begin{array}{r} 11010 \\ +\ 10000 \\ \hline 101010 \end{array}$$

Since there is end carry, neglect it and remaining is answer.

$\therefore$ $(11010)_2 - (10000)_2 = (01010)_2$

ii) Here, $(1000100)_2 - (1010100)_2$

$M = (1000100)_2$

$S = (1010100)_2$

2's complement of S = $(0101100)_2$

Now,
$$\begin{array}{r} 1000100 \\ +\ 0101000 \\ \hline 1110000 \end{array}$$

Since there is no end carry, answer is 2's complement of 1110000 *i.e.*, $-(0010000)_2$

$\therefore$ $(1000100)_2 - (1010100)_2 = -(0010000)_2$

30. Use 2's complement to subtract the following; [2018/S, 2019/S(similar)]

i) $(1010)_2 - (10100)_2$

ii) $(957)_{10} - (876)_{10}$

iii) $(378)_{BCD} - (256)_{BCD}$

Solution:

i) Here, $(1010)_2 - (10100)_2$

M = 01010

S = 10100

2's complement of S = 01100

Now,
$$\begin{array}{r} 01010 \\ +\ 01100 \\ \hline 10110 \end{array}$$

Since there is no end carry, answer is $(01010)_2$

$\therefore$ $(1010)_2 - (10100)_2 = -(01010)_2$

ii) Here, $(957)_{10} - (876)_{10}$

$M = (957)_{10} = (1110111101)_2$

$S = (876)_{10} = (1101101100)_2$

2's complement of S = $(0010010100)_2$

Now,
$$\begin{array}{r} 1110111101 \\ +\ 0010010100 \\ \hline 10001010001 \end{array}$$

$\therefore$ $(957)_{10} - (876)_{10} = (0001010001)_2 = (81)_{10}$

There is end carry, so neglect it and remaining is answer.

iii) Here, $(378)_{BCD} - (256)_{BCD}$

Same as Q.N. 27. (iii)

31. Perform $(45)_{10} - (99)_{10}$ using r's complement. [2019/

Solution:

$M = (45)_{10}$

$S = (99)_{10}$

10's complement of S = 100 – 99 = 1

Now,
$$\begin{array}{r} 45 \\ +\ 1 \\ \hline 46 \end{array}$$

Since there is no end carry, answer is 10's complement of given answe with negative sign. *i.e.*, $-(100 - 46) = -(54)_{10}$

$\therefore$ $(45)_{10} - (99)_{10} = -(54)_{10}$

32. Explain in brief about modulo 2 systems. [2013/

Solution: See the topic 2.7.

CHAPTER **3**

BOOLEAN ALGEBRA AND LOGIC GATES

* * * * *

* * * * *

3.1 BASIC DEFINITION

Boolean Algebra, like any other deductive mathematical system, may be defined with a set of elements, a set of operators, and a number of assumptions and postulates. A set of elements means any collection of objects having common properties. If S denotes a set, and X and Y are certain objects, then $X \in S$ denotes X is an object of set S. A binary operator defined on a set S of elements is a rule that assigns to each pair of elements from S a unique element from S.

The postulates of a mathematical system are based on the basic assumptions, which make possible to deduce the rules, theorems and properties of the system. Various algebraic structures are formulated on the basis of the most common postulates which are described as follows;

a) Closure

A set is closed with respect to a binary operator if, for every pair of elements of S, the binary operator specifies a rule for obtaining a unique element of S. For example, the set of natural numbers N = {1, 2, 3, 4, ...} is said to be closed with respect to the binary operator plus (+) by the rules of arithmetic addition, since for any X, Y∈N we obtain a unique element Z∈N by the operation X + Y = Z

However, note the set of natural numbers is not closed with respect to the binary operator minus (–) by the rules of arithmetic subtraction because for 1 – 2 = –1 where –1 is not of the set of natural numbers.

b) Associative law

A binary operator X on a set S is said to be associated whenever

$(A \times B) \times C = A \times (B \times C)$ for all A, B, C∈S

c) Commutative law

A binary operator × on a set S is said to be commutative whenever

$A \times B = B \times A$ for all A, B∈S

d) Identify Element

A set S is to have an identity element with respect to a binary operation on S, if there exists an element E∈S with the property,

$E \times A = A \times E = A$

e) Inverse

If a set S has the identity element E with respect to a binary operator there exists an element B∈S, which is called the inverse for every A∈S such that $A \times B = E$

Example: In the set of integers I with E = 0, the inverse of an element A is (–A) since A + (–A) = 0

f) Distributive law

If × and (·) are two binary operators on a set S, × is said to be distributive over (·), whenever

$A \times (B \cdot C) = (A \times B) \cdot (A \times C)$

If summarized, for the field of real numbers, the operators and postulates have the following meanings.

- The binary operators + defines addition.
- The additive identify is 0.
- The additive inverse defines subtraction.
- The binary operator (·) defines multiplication.
- The multiplication inverse of A is $\frac{1}{A}$ defines division

  i.e., $A \cdot \frac{1}{A} = 1$
- The only distributive law applicable is that of (·) over +

  $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$
- The multiplication identity is 1.

3.1.1 Definition of Boolean Algebra

In 1854 George Boole introduced a systematic treatment of logic and developed for this purpose an algebraic system now called Boolean algebra. In 1938 C·E Shannon developed a two valued Boolean algebra called switching algebra and demonstrated that the properties of two valued or bitable electrical switching circuits can be represented by this algebra. The postulates formed by E·V Huntington in 1904 are employed for the formal definition of Boolean algebra. However, Huntington postulates are not unique for defining Boolean Algebra and other unique for defining Boolean algebra and other postulates are also used. The following Huntington postulates are satisfied for the definition of Boolean algebra on a set of elements S together with two binary operators (+) and (·).

1. a) Closure with respect to the operator (+)
   b) Closure with respect to the operator (·)
2. a) An identity element with respect to + is designated by 0
   i.e., $A + 0 = 0 + A = A$
   b) An identity element with respect to (·) is designated by 1
   i.e., $A \cdot 1 = 1 \cdot A = A$
3. a) Commutative with respect to (+) *i.e.*, $A + B = B + A$
   b) Commutative with respect to (·) *i.e.*, $A \cdot B = B \cdot A$
4. a) (.) is distributive over (+), *i.e.*, $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$
   b) (+) is distributive over (·) *i.e.*, $A + (B \cdot C) = (A + B) \cdot (A + C)$
5. For every element A∈S, there exists an element A'∈S (called the complement of A) such that $A + A' = 1$ and $A \cdot A' = 0$
6. There exists atleast two elements A, B∈S such that A is not equal to B.

Comparing Boolean Algebra with arithmetic and ordinary algebra (the field of real numbers), the following differences are observed.

1. Huntington postulates do not include the associative law. However, Boolean Algebra follows the law and can be derived from the other postulates for both operations.
2. The distributive law of (+) over (·) *i.e.*, $A + (B \cdot C) = (A + B) \cdot (A + C)$ is valid for Boolean algebra but not for ordinary algebra.
3. Boolean Algebra does not have additive or multiplicative inverses. So, there are no subtraction or division operations.
4. Postulates 5 define an operator called complement which is not available in ordinary algebra.
5. Ordinary algebra deals with real numbers which consists of an infinite set of elements. Boolean Algebra deals with the as yet undefined set of elements S but in the two valued Boolean algebra, the set S consists of only two elements -0 and 1.

Boolean algebra is very much similar to ordinary algebra in some respects. The symbols (+) and (·) are chosen intentionally to facilitate

Boolean algebraic manipulations by persons already familiar to ordinary algebra. Although one can use some knowledge from ordinary algebra to deal with Boolean algebra.

3.2 BASIC PROPERTIES AND THEOREM OF BOOLEAN ALGEBRA

3.2.1 Principle of Duality

From Huntington postulates, it is evident that they are grouped in pairs (a) and (b) and every algebraic expression deductible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged. This means one expression can be obtained from the other in each pair by interchanging every element *i.e.*, every 0 with 1, every 1 with 0 as well as interchanging the operators *i.e.*, every (+) with (·) and every (·) with (+). This important property of Boolean algebra is called principle of duality.

3.2.2 Basic Theorems

Theorem 1(a) A + A = A

A + A = (A + A)·1	By postulate 2(b)
= (A + A)·(A + A')	By postulate 5
= A + A·A'	
= A + 0	By postulate 4
= A	By postulate 2(a)

Theorem 1(b): A·A = A

A·A = (A·A) + 0	By postulate 2(a)
= (A·A) + (A·A')	By postulate 5
= A (A + A') = A·1 = A	

Theorem 2(a) A + 1 = 1

A + 1 = 1·(A + 1)	By postulate 2(b)
= (A + A') (A + 1)	
= A + A'·1	
= A + A'	
= 1	

Theorem 2(b): A·0 = 0

A·0 = 0	By duality

Theorem 3(a): A + A·B = A

A + A·B = A·1 + A·B	By postulate 2(b)
= A (1 + B)	By postulate 4(a)
= A·1	By postulate 2(a)
= A	By postulate 2(b)

Theorem 3(b): A (A + B) = A By duality

The following is the complete list of postulates and theorems useful for two valued Boolean algebra.

Postulate 2	a) A + 0 = A	b) A·1 = A
Postulate 5	a) A + A' = 1	b) A·A' = 0
Theorem 1	a) A + A = 1	b) A·A = A
Theorem 2	a) A + 1 = 1	b) A·0 = 0
Theorem 3, Involution	(A')' = A	
Theorem 3, Commutative	a) A + B = B + A	b) A·B = B·A
Theorem 4, Associative	a) A + (B + C) = (A + B) + C	b) A·(B·C) = (A·B)·C
Theorem 4, Distributive	a) A (B + C) = A·B + A·C	b) A + B·C = (A + B)·(A + C)
Theorem 5, De- Morgan	a) (A + B)' = A'·B'	b) (A·B)' = A' + B'
Theorem 6, Absorption	a) A + A·B = A	b) A·(A + B) = A

3.3 DE-MORGAN'S THEOREM

Two theorems that were proposed by De-Morgan plays important role in Boolean algebra. It defines the uniformity between the gate with same inverted input and output.

a) De-Morgan's First Theorem

The first theorem states that the complement of sum is equal to the product of the complements. That is, if the variables are A and B, then, (A + B)' = A'·B'.

Truth table;

Inputs			Output 1			Output 2
A	B	A + B	(A + B)'	A'	B'	A'·B'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Conclusion: Comparing the values of (A + B)' and A'·B' from truth table both are equal, hence proved.

De-Morgan's theorem are applicable to any number of variables. For three variables A, B and C, the equations are,

(A·B·C)' = A' + B' + C' and (A + B + C)' = A'·B'·C'

b) De-Morgan's Second Theorem

The second theorem states that the complement of a product is equal to sum of the complements. That is, if the variables are A and B, then,

(A·B)' = A' + B'

It can be verified using truth table as,

Inputs		Output 1				Output 2
A	B	A·B	(A·B)'	A'	B'	A' + B'
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Conclusion: Comparing the values of (A·B)' and A' + B' from truth table both are equal, hence proved.

3.4 LOGIC GATES AND TRUTH TABLES

Logic Gates are electronic circuits that can be used to implement the most elementary logic expressions also known as Boolean expressions. The logic gate is the most basic building block of combinational logic. There are three basic logic gates namely the OR gate, the AND gate and the NOT gate other logic gates that are derived from these basic gates are the NAND gate, the NOR gate, the Exclusive - OR gate and the Exclusive NOR gate.

A truth table lists all possible combinations of input binary variables and the corresponding outputs of a logic system.

A. OR Gate

An OR gate performs an ORing operation on two or more than two logic variables. The OR operation on two independent logic variables A and B is written as Y = A + B and and reads as Y equals A or B and not as A plus B. An OR gate is a logic circuit with two or more inputs and one output. The output of an OR gate is low only when all of its inputs are LOW. For all other possible input combinations, the output is HIGH. The operation of a two input OR gate is explained by the logic expression Y = A + B.

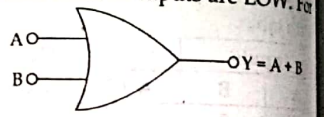


Figure: Circuit symbol of two input OR gate

Truth Table for two input

A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

If we have four logic variables and we want to know the logical output of (A + B + C + D), then it would be the output of a four input OR gate with A, B, C and D as its inputs.

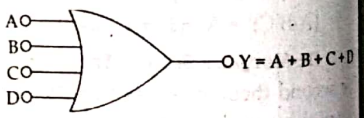


Figure: Four input OR gate

Truth Table for four input

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

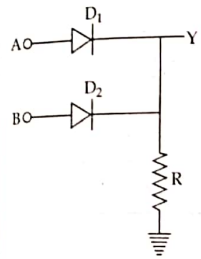


Figure: Two input diode OR gate

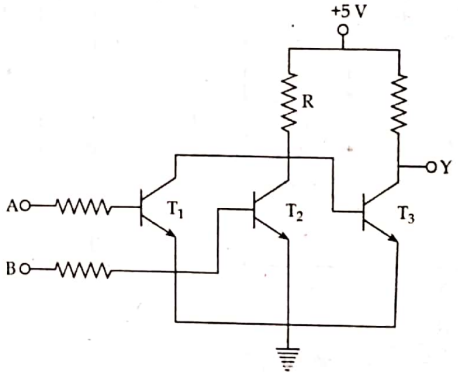


Figure: Two input transistor OR gate

Figure: Dicrete OR gate

B. AND Gate

An AND gate is logical circuit having two or more inputs and one output. The output of an AND gate is HIGH only when all of its inputs are in the HIGH state. In all other cases, the output is LOW. The AND operation on two independent logic variables A and B is written as Y = A·B and reads as Y equals A AND B and not as A multiplied by B. Here, A and B are input logic variables and Y is the output.

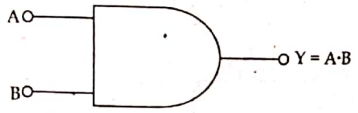


Figure: Logic symbol for two input

Truth Table for two input

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

For a two input AND gate, Y = A·B

For a three input AND gate, Y = A·B·C

For a four input AND gate Y = A·B·C·D

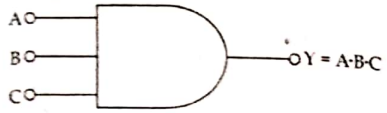


Figure: Logic symbol for three inputs

Truth Table for three inputs

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

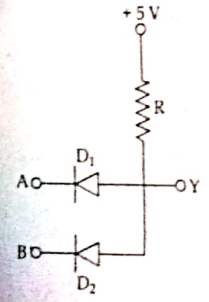


Figure: Two-input diode AND gate

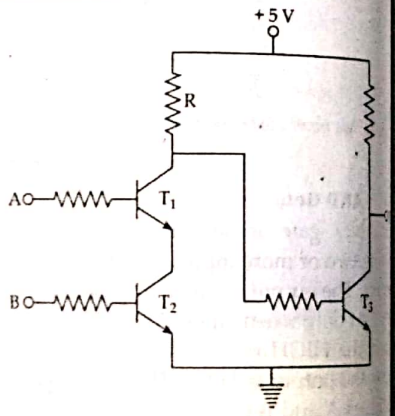


Figure: Two-input transitor AND gate

Figure: Discrete AND gate

C. NOT Gate

A NOT gate is a one input, one output logic circuit whose output is always the complement of the input. That is, a low input produces a HIGH output and vice-versa. The NOT operation on a logic variable X is denoted as $\overline{X}$ or X'. That is, if X is the input to a NOT circuit, then its output Y is given by $Y = \overline{X}$ or X' and reads as Y equals NOT X. Thus, if X = 0, Y = 1 and if X = 1, Y = 0

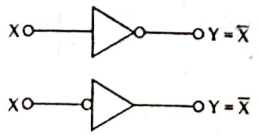


Figure: NOT gate circuit symbol

Truth Table

X	Y
0	1
1	0

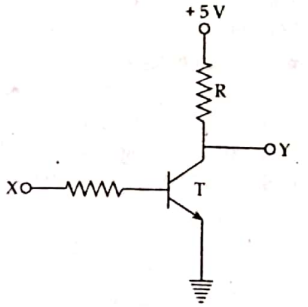


Figure: Transitor inverter

D. Exclusive–OR Gate or X–OR Gate

The Exclusive–OR gate, commonly written as EX–OR gate is a two input, one-output gate. The output of an EX–OR gate is a logic '1' when the inputs are unlike and a logic '0' when the inputs are like. Although EX–OR gates are available in integrated circuit form only as two input gates, unlike other gates which are available in multiple input also, multiple input EX–OR logic functions can be implemented using more than one two-input gates. The output of a two-input EX–OR gate is expressed by,

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

Truth Table For two inputs

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

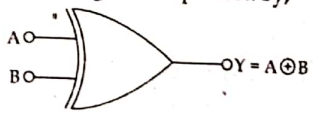


Figure: Logic symbol

E. Exclusive–NOR Gate or XNOR Gate

Exclusive NOR, (commonly written as EX–NOR) means NOT of EX–OR *i.e.*, the logic gate that we get by complementing the output of EX–OR gate. The truth table of an EX–NOR gate is obtained from the truth table of an EX–OR gate by complementing the output entries. Logically,

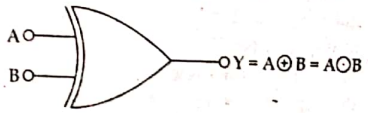


Figure: Logic symbol

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

Truth Table

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

The output of two input EX-NOR gate is a logic '1' when the inputs a
like and a logic '0' when they are unlike.

F. NAND Gate

NAND stands for NOT AND. An AND gate followed by a NOT circ
makes it a NAND gate. The output of a NAND gate is a logic '0' when
its input are a logic '1'. For all other input combinations, the output is
logic '1' NAND gate operation is logically expressed as, $Y = \overline{A \cdot B}$

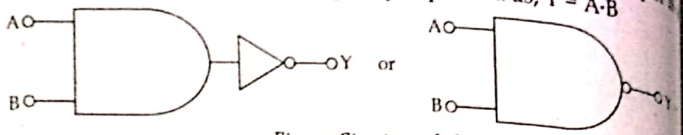


Figure: Circuit symbol

Truth Table

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

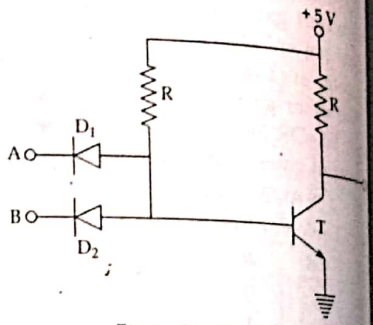


Figure: Discrete NAND gate

G. NOR Gate

NOR stands for NOT OR. An OR gate followed by a NOT circuit makes it a NOR. The output of a NOR gate is a logic '1' when all its inputs are logic '0'. For all other input combinations, the output is a logic '0'. The output of a input NOR gate is logically expressed as,

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

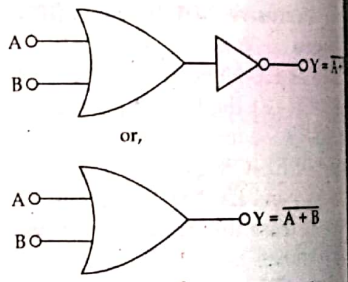


Figure: Circuit symbol

Truth Table for two inputs

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

In general, the Boolean expression for a NOR gate with more than two inputs can be written as,

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

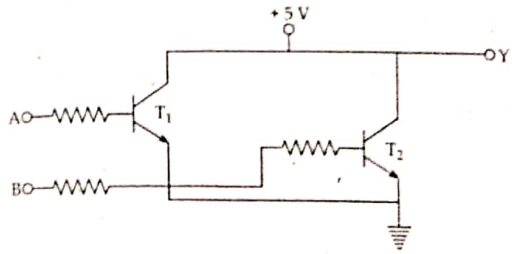


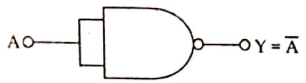
Figure: Discrete NOR gate

3.5 UNIVERSALITY OF NAND AND NOR GATE

NOR and NAND gates have the property that they individually can be used to hardware implement a logic circuit corresponding to any given Boolean expression. That is, it is possible to use use either only NAND gates or only NOR gates to implement any Boolean expression. This is so because a combination of NAND gates or a combination of NOR gates can be used to perform functions of any of the basic logic gates. It is for this reason that NAND and NOR gates are universal gates.

A. Implementation of basic logic gates using only NAND gates

i)



is equivalent to

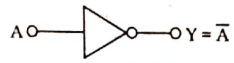
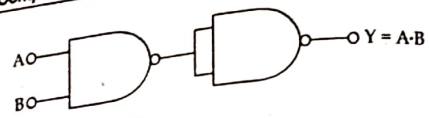


Figure: NOT gate

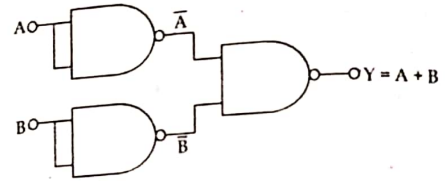
ii)



is equivalent to

Figure: AND gate

iii)



is equivalent to

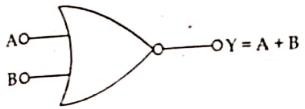
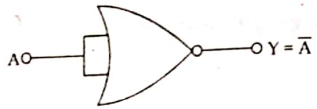


Figure: OR gate

B. Implementation of basic logic gates using only NOR gates

i)



is equivalent to

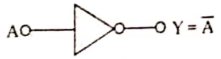
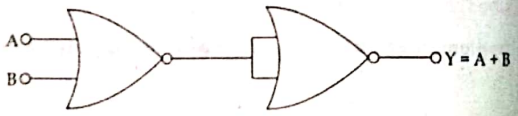


Figure: NOT gate

ii)



is equivalent to

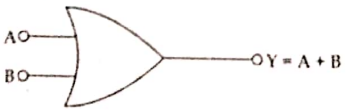
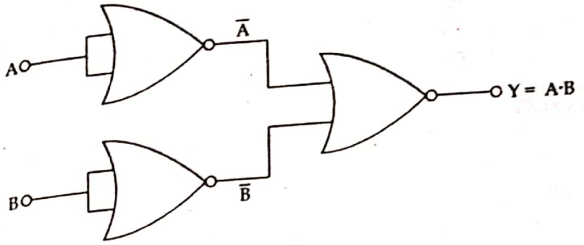


Figure: OR gate

iii)



is equivalent to

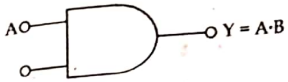


Figure: AND gate

3.6 TRISTATE LOGIC OR THREE-STATE LOGIC

Tristate logic gates have three possible output states *i.e.*, logic '1' state, the logic '0' state and a high impedance state. The high impedance state is controlled by an external ENABLE input. The ENABLE input decides whether the gate is active or in the high in the impedance state. When active, it can be '0' or '1' depending upon input conditions. One of the advantages of these gates is that their inputs and outputs can be connected in parallel to a common bus line. When tristate devices are paralled, only one of them is enabled at a time.

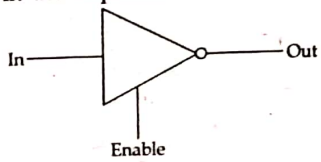


Figure: Tristate logic

Truth Table

Enable	In	Out
0	0	High Z
0	1	High Z
1	0	1
1	1	0

Three state output are implemented in many registers, bus drivers and flip flops in the 7400 and 4000 series as well as in other types, but also internally in many integrated circuits other typical uses are internal and external buses in microprocessors, computer memory and peripherals.

The basic concept of the third state, high impedance is to effectively remove the device's influence from the rest of the circuit. If more than one device is electrically connected to another device, putting an output into the High impedance state is often used to prevent short circuits, or one device driving high (logical 1) against another device driving low (logical

with large numbers of inputs. They are essential for the operation of efficient multiplexers especially those shared electronic bus. They can reduce the number of wires needed to drive a set of LEDS.

Example 1

How would you hardware implement a for input OR gate using two input OR gates only?

Solution:

Figure (a) shows one possible arrangement of two input OR gates that simulates a four input OR gate. A, B, C and D are logic inputs and Y is the output.

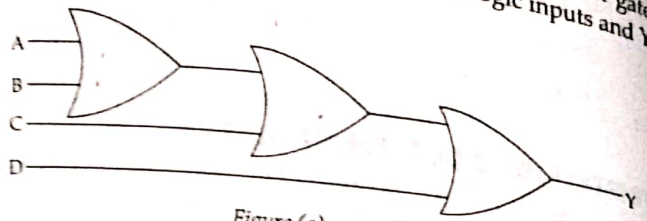


Figure (a)

Figure (b) shows another possible arrangement

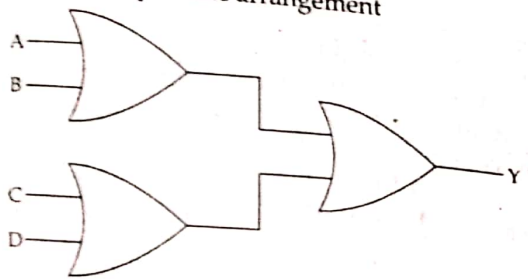


Figure (b)

$\therefore \quad Y = A + B + C + D$

Example 2

How do you implement three input and four input EX–OR logic functions with the help of two input EX–OR gates?

Solution:

In figure (a), the output Y_1 is given by $A \oplus B$. The final output Y is given by

$$Y = (Y_1 \oplus C) = (A \oplus B) \oplus C = A \oplus B \oplus C$$

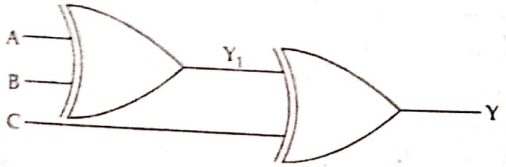


Figure (a)

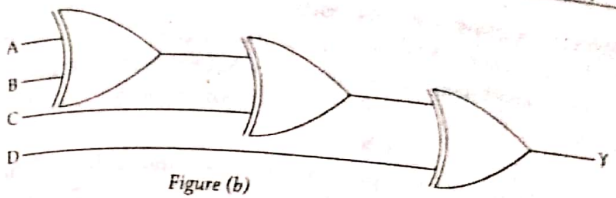


Figure (b)

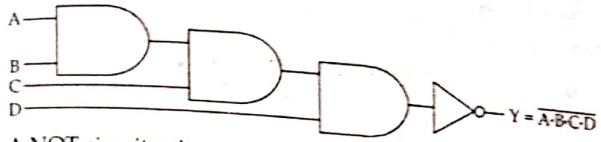
Figure: Three input EX–OR gate and (b) a four input EX–OR gate

Example 3

Show the logic arrangements for implementing

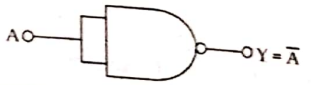
a) A four input NAND gate using two input AND and NOT gates

Solution:



b) A NOT circuit using a two input NAND gate.

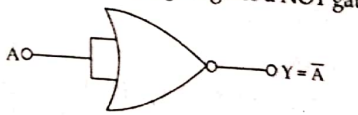
Solution:



c) A NOT circuit using a two input NOR gate.

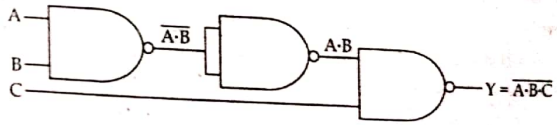
Solution:

Shorting the inputs of a NOR gate gives a NOT gate.



d) A three input NAND gate using two input NAND gates

Solution:



EXAMINATION QUESTION SOLUTIONS

1. Explain the NAND gate as an universal gate. [2011/S

Write short notes in universality of NAND and NOR gate. [2011/F, 2015/S, 2015/F, 2018/F, 2018/S

OR,

Discuss the universal property of NAND and realize all the basic gates using NAND gate only. [2012/S, 2017/S

OR,

Realize all the basic gates using NOR gate only. [2012/F, 2019/S

OR,

Explain the universal property of NOR gate with appropriate logic gates. [2013/S

Solution: See the topic 3.5.

2. State and prove De-Morgan's Theorem. [2011/S, 2014/S, 2016/S, 2018/S, 2019/S

Solution: See the topic 3.3

3. What are the universal gates? Why they are called so? [2014/S, 2017/S

Solution:

NAND gates and NOR gates are the universal gates. They are so called because any type of gates or logic functions can be implemented by these gates.

4. Define logic gates. [2013/F

Solution:

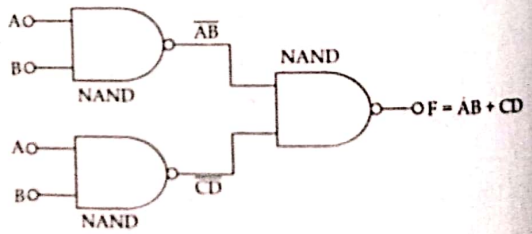
Logic gates are the basic blocks of any digital system. It is an electronic circuit having one or more than one input and only one output. It is an idealized or physical device implementing a Boolean function.

5. Construct F = AB + CD using universal gates. [2013/S

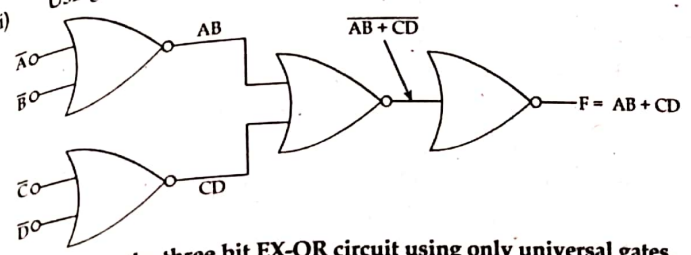
Solution:

F = AB + CD

i) Using NAND gate only,



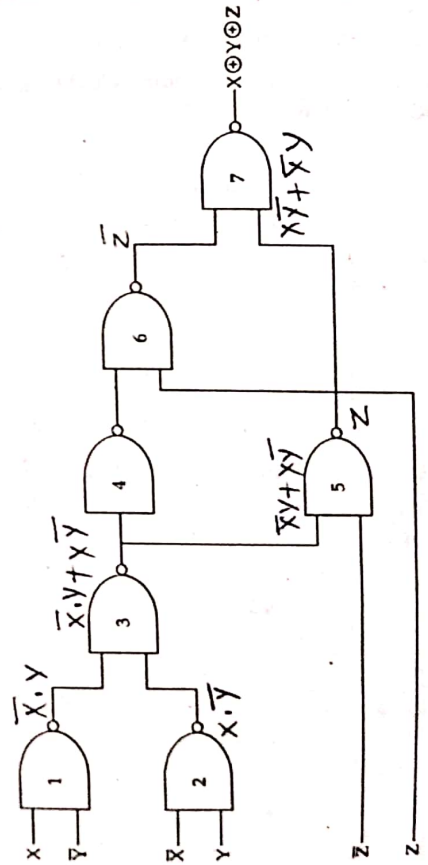
ii) Using NOR gate only,



6. Design the three bit EX-OR circuit using only universal gates. [2014/S]

Solution:

Let, X, Y and Z be three bit inputs. Taking NAND gate as universal gates.



Gate number	Output
1.	$\overline{(X\overline{Y})} = \overline{X} + Y$
2.	$\overline{(\overline{X}Y)} = X + \overline{Y}$
3.	$[(\overline{X} + Y)(X + \overline{Y})]$
	$= (\overline{X} + \overline{Y}) + \overline{(X + \overline{Y})} = X \oplus Y$
4.	$\overline{(X \oplus Y)}$
5.	$[\overline{(X \oplus Y)} \cdot \overline{Z}] = [\overline{(X \oplus Y)} + Z]$
6.	$[\overline{(X \oplus Y)} \cdot Z] = [(\overline{X \oplus Y}) + \overline{Z}]$
7.	$[\overline{\{(X \oplus Y) + \overline{Z}\}} \cdot \overline{\{(X \oplus Y) + \overline{Z}\}}]$
	$= [\overline{(X \oplus Y) + \overline{Z}}] + [\overline{(X \oplus Y) + \overline{Z}}]$
	$= (\overline{X \oplus Y}) \cdot Z + (X \oplus Y) \cdot \overline{Z}$
	$= X \oplus Y \oplus Z$

Hence, output is $X \oplus Y \oplus Z$ *i.e.*, three bit X-OR gate.

Implement XNOR gate using only NAND gates and XOR gate using NOR gate only. **[2016/F]**

Solution:

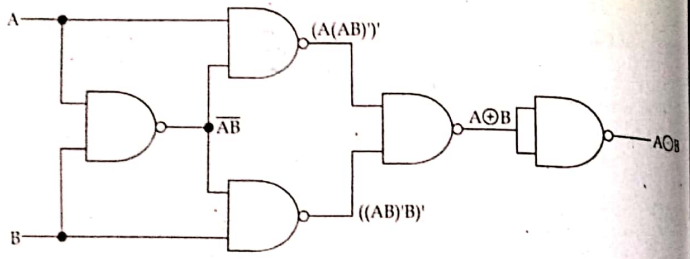
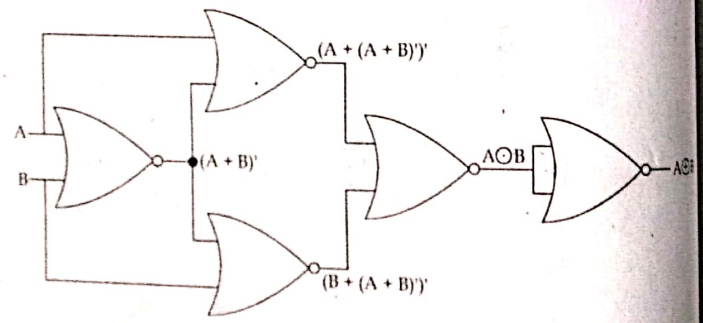


Figure: X-NOR gate using only NAND gates.

X-OR gate is actually X-NOR followed by NOT gate. So, give the output of X-NOR gate to a NOT gate, overall output is that of an X-OR gate.

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



8. **Define positive and negative logic system.** **[2014/S]**

Solution:

In positive logic systems, high voltage is associated with a logic 1 and a low voltage with a logic 0.

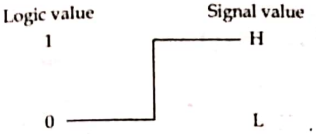


Figure: Positive logic

In negative logic systems, a high voltage is associated with a logic 0 and low voltage with a logic 1.

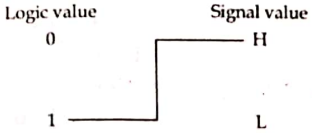


Figure: Negative logic

In most cases, we will assume positive logic. The AND gate positive logic will not behave as AND gate in negative logic.

9. **Write short notes on Tri-state logic.** **[2011/F]**

Solution: See the topic 3.6.

10. **Compare and contrast XOR and XNOR gate.** **[2017/F]**

Solution: See the topic 3.3 D and 3.3 E.

ADDITIONAL QUESTION SOLUTIONS

1. Demonstrate by means of truth tables the validity of the following identities.
   - i) De-Morgan's theorem for three variables: $(xyz)' = x' + y' + z'$
   - ii) Second distributive law: $x + yz = (x + y)(x + z)$
   - iii) $x + (y + z) = (x + y) + z$

Solution:

i)

x	y	z	(xyz)	(xyz)'	x'	y'	z'	x' + y' + z'
0	0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0	1
0	1	0	0	1	1	0	1	1
0	1	1	0	1	1	0	0	1
1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	0	1	1
1	1	1	1	0	0	0	0	0

Hence, $(xyz)' = x' + y' + z'$

ii)

x	y	z	x + yz	x + y	x + z	(x + y) (x + z)
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	1

$\therefore \quad (x + yz) = (x + y)(x + z)$

iii)

x	y	z	x	y + z	x(y + z)	(x + y)	(x + y) + z
0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1
0	1	0	0	1	1	1	1
0	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1
1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1

2. Find the complement of $F = x + yz$ then show that $F \cdot F' = 0$ and $F + F' = 1$.

Solution:

Here,
$F = x + yz$
$\bar{F} = \bar{x} + \bar{y}\,\bar{z} = (x + yz)'$

Now,
$F \cdot \bar{F} = (x + yz) \cdot (x + yz)' = 0 \quad (\because A \cdot \bar{A} = 0)$

and,
$F + \bar{F} = (x + yz) + (x + yz)' = 1 \quad (\because A + \bar{A} = 1)$

3. **Simplify the following expression**
   - i) $xy + x\bar{y}$
   - ii) $(x + y)(x + \bar{y})$
   - iii) $xyz + \bar{x}y + xy\bar{z}$
   - iv) $(A + B)'(A' + B')$
   - v) $xyz' + x'yz + xyz + \bar{x}yz'$
   - vi) $(x + y + z')(x' + y' + z)$

Solution:

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

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

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

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

v) $xy\bar{z} + \bar{x}yz + xyz + \bar{x}\,y\,\bar{z} = xy(z + \bar{z}) + \bar{x}\,y(z + \bar{z}) = xy + \bar{x}y = y$

vi) $(x + y + \bar{z})(\bar{x} + \bar{y} + z) = x\bar{x} + x\bar{y} + xz + \bar{x}y + y\bar{y} + yz + \bar{x}\,\bar{z} + \bar{y}\,\bar{z} + z\bar{z}$
$= x\bar{y} + xz + \bar{x}y + yz + \bar{x}\,\bar{z} + \bar{y}\,\bar{z}$
$= x \oplus y + \overline{(x \oplus z)} + \overline{(y \oplus z)}$

4. **Show that the dual of the exclusive OR is equal to its complement.**

Solution:

$x \oplus y = \bar{x}y + x\bar{y}$ and $\overline{(x \oplus y)} = (x + \bar{y})(\bar{x} + y)$

Dual of $\bar{x}y + x\bar{y} = (\bar{x} + y)(x + \bar{y}) = \overline{(x \oplus y)}$

5. **Show that a positive logic NAND gate is a negative logic NOR gate and vice-versa.**

Solution:

Gate

X	Y	Z
L	L	H
L	H	H
H	L	H
H	H	L

NAND (Positive logic)

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

NOR (Negative logic)

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

Gate

X	Y	Z
L	L	H
L	H	L
H	L	L
H	H	L

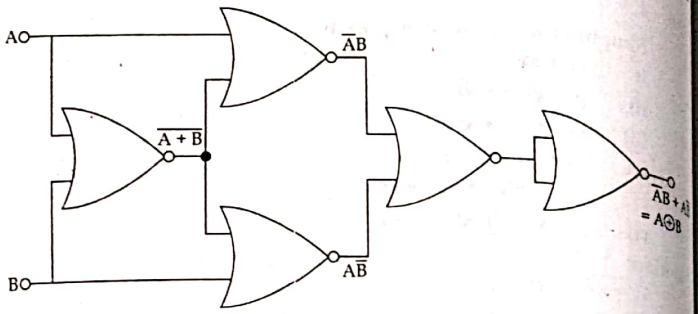
NOR (Positive logic)

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

NAND (Negative logic)

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

6. Implement XOR gate using only NOR gates only.



7. Implement AND gate with OR and NOT gates.

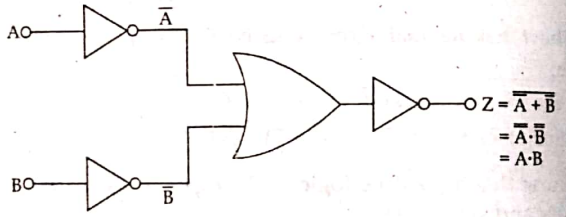
Solution:

Let, $Z = A \cdot B$

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

Taking again complement,

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



8. Implement OR gate with AND gate and NOT gate.

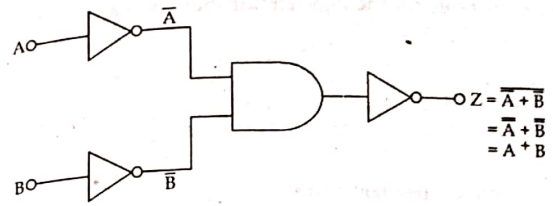
Solution:

Let, $Z = A + B$

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

Taking again complement,

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

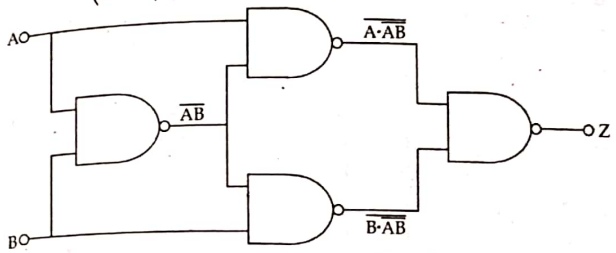


9. Design EX – OR gate using only NAND gate.

Solution:

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

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



10. How can exclusive NOR gate can be obtained using NAND gates only?

Solution:

An EX-NOR gate is represented as,

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

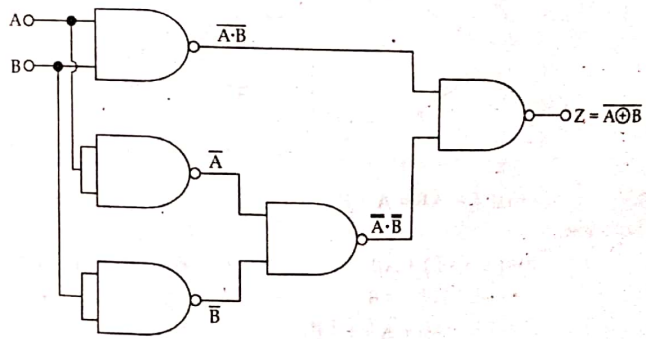
Let, $Z = \overline{(\overline{A \cdot B}) \cdot (\overline{A \cdot B})}$

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

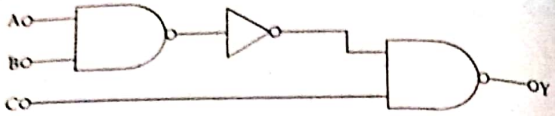
$= A \cdot B + \overline{A} \cdot \overline{B}$

$= \overline{A \oplus B}$

$=$ EX–NOR



11. Draw the table for the logic circuit show below;



Solution:

Boolean equation for the logic circuit is,

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

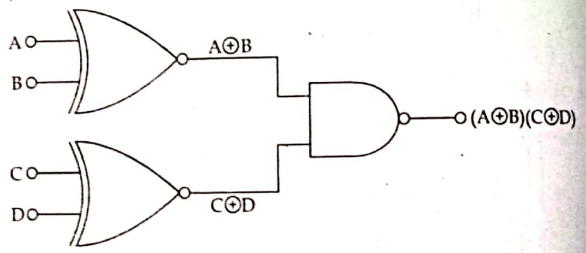
Truth table

Inputs			Outputs
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

12. Implement the following Boolean expression with XOR and AND gates. $F = A\overline{B}C\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + \overline{A}B\overline{C}D$.

Solution:

$$\begin{aligned} F &= A\overline{B}C\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + \overline{A}B\overline{C}D \\ &= A\overline{B}C\overline{D} + A\overline{B}\,\overline{C}D + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D \\ &= A\overline{B}(C\overline{D} + \overline{C}D) + \overline{A}B(C\overline{D} + \overline{C}D) \\ &= (A\overline{B} + \overline{A}B)\,(C\overline{D} + \overline{C}D) \\ &= (A \oplus B)\,(C \oplus D) \end{aligned}$$



13. Prove that $A + \overline{A}B = A + B$

Solution:

$$\begin{aligned} A + \overline{A}B &= (A + AB) + \overline{A}B && [\because A + AB = A] \\ &= AA + AB + \overline{A}B && [\because A \cdot A = A] \\ &= AA + AB + A\overline{A} + \overline{A}B && [A \cdot \overline{A} = 0] \end{aligned}$$

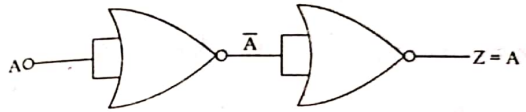
$$\begin{aligned} &= (A + \overline{A}) \cdot (A + B) \\ &= 1 \cdot (A + B) \end{aligned}$$

$\therefore A + \overline{A}B = (A + B)$ proved. $[\because A \cdot \overline{A} = 0]$

14. Simplify the following and realize it using NOR gates only.
$Y = A\overline{B} + AB\overline{C} + ABCD + ABC\overline{D}$

Solution:

$$\begin{aligned} Y &= A\overline{B} + AB\overline{C} + ABCD + ABC\overline{D} \\ &= A\overline{B} + AB\overline{C} + ABC\,(D + \overline{D}) \\ &= A\overline{B} + AB\overline{C} + ABC && (\because D + \overline{D} = 1) \\ &= A\overline{B} + AB\,(\overline{C} + C) \\ &= A\overline{B} + AB && (\because \overline{C} + C = 1) \\ &= A\overline{B} + AB \\ &= A\,(\overline{B} + B) \\ &= A \end{aligned}$$



15. Simplify $f = (A + \overline{B}C) + \overline{(A + \overline{B} \cdot C)}$.

Solution:

$$\begin{aligned} f &= (A + \overline{B}C) + \overline{A + \overline{B}C} \\ &= (A + \overline{B}C) + \overline{A} \cdot \overline{\overline{B}C} \\ &= (A + \overline{B}C) + \overline{A} \cdot (\overline{\overline{B}} + \overline{C}) \\ &= (A + \overline{B}C) + \overline{A}B + \overline{A}\,\overline{C} \\ &= A(B + \overline{B})\,(C + \overline{C}) + \overline{B}C(A + \overline{A}) + \overline{A}B(C + \overline{C}) + \overline{A}\,\overline{C}(B + \overline{B}) \\ &= A(BC + B\overline{C} + \overline{B}C + \overline{B}\,\overline{C}) + A\overline{B}C + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}B\overline{C} \\ &\quad + \overline{A}B\overline{C} + \overline{A}\,\overline{B}\,\overline{C} \\ &= ABC + AB\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}B\overline{C} + \overline{A}\,\overline{B}\,\overline{C} \\ &= AB(C + \overline{C}) + A\overline{B}(C + \overline{C}) + \overline{A}\overline{B}\,(C + \overline{C}) + \overline{A}B(C + \overline{C}) \\ &= AB + A\overline{B} + \overline{A}\overline{B} + \overline{A}B \\ &= A(B + \overline{B}) + \overline{A}(B + \overline{B}) \\ &= (A + \overline{A}) \end{aligned}$$

$\therefore f = 1$

16. Simplify $f(w, x, y, z) = x + xyz + \bar{x}yz + wx + \bar{w}x + \bar{x}y$

Solution:

$$\begin{aligned} f &= x + xyz + \bar{x}yz + wx + \bar{w}x + \bar{x}y \\ &= x + (x + \bar{x})yz + (w + \bar{w})x + \bar{x}y \\ &= x + yz + x + \bar{x}y \\ &= x + \bar{x}y + yz \end{aligned}$$

$= x + y + yz$

$= x + y(1 + z)$

$\therefore \quad f = x + y$

17. Simplify the following Boolean expression which represents the output of a logical decision circuit.

Solution:

$$f(A, B, C, D, E) = (AB + C + D)(\overline{C} + D)(\overline{C} + D + E)$$
$$= (AB + C + D)(\overline{C} + D)(\overline{C} + D + E)$$
$$= [AB\overline{C} + \overline{C}D + ABD + D(1 + C)](\overline{C} + D + E)$$
$$= [AB\overline{C} + \overline{C}D + D + ABD](\overline{C} + D + E)$$
$$= (AB\overline{C} + D)(\overline{C} + D + E)$$
$$= AB\overline{C} + D\overline{C} + ABD\overline{C} + D + AB\overline{C}E + DE$$
$$= AB\overline{C} + D(1 + \overline{C} + ABD + E) + AB\overline{C}E$$
$$= AB\overline{C}(1 + E) + D$$
$$= AB\overline{C} + D$$

18. Prove the following identity $\overline{AB + \overline{A} + AB} = 0$.

Solution:

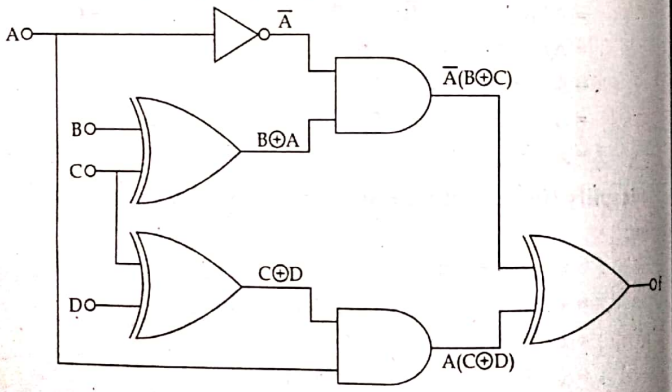
$$\overline{AB + \overline{A} + AB} = \overline{\overline{A} + \overline{B} + \overline{A} + AB}$$
$$= \overline{\overline{A} + \overline{B} + AB} = \overline{A} \cdot \overline{B} \cdot \overline{AB}$$
$$= AB \cdot \overline{AB} = AB(\overline{A} + \overline{B}) = AB\overline{A} + AB\overline{B}$$
$$= 0 \text{ proved.}$$

19. Realize the following expression using EX-OR/EX-NOR gate and basic gates if required.

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

Solution:

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



20. Prove that: $A + \overline{A}B = A + B$

Solution:

$$A + \overline{A}B = (A + AB) + \overline{A}B \qquad [\because A + AB = A]$$
$$= AA + AB + \overline{A}B \qquad [\because A \cdot A = A]$$
$$= AA + AB + A\overline{A} + \overline{A}B \qquad [\because A \cdot \overline{A} = 0]$$
$$= (A + \overline{A}) \cdot (A + B) \qquad [\because A + \overline{A} = 1]$$
$$= 1 \cdot (A + B) = A + B$$

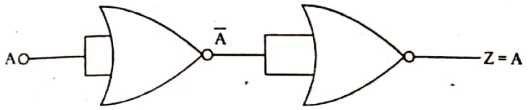
21. Simplify the following and realize it using NOR gates only, $Y = A\overline{B} + AB\overline{C} + ABCD + ABC\overline{D}$

Solution:

Here;

$$Y = A\overline{B} + AB\overline{C} + ABCD + ABC\overline{D}$$
$$= A\overline{B} + AB\overline{C} + ABC(D + \overline{D})$$
$$= A\overline{B} + AB\overline{C} + ABC \qquad [\because D + \overline{D} = 1]$$
$$= A\overline{B} + AB(\overline{C} + C) = A\overline{B} + AB = A(\overline{B} + B) = A$$

Using NOR gates only

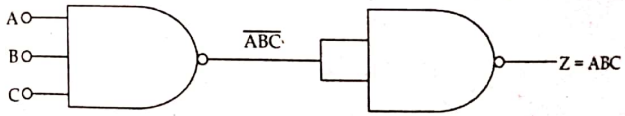


22. Simplify the following Boolean expression and realize it using only NAND gates $Y = ABC[AB + \overline{C}(BC + AC)]$

Solution:

Here;

$$Y = ABC[AB + \overline{C}(BC + AC)]$$
$$= ABC(AB + BC\overline{C} + AC\overline{C})$$
$$= ABC(AB + 0) \qquad [\because C\overline{C} = 0]$$
$$= A^2B^2C = ABC$$



23. Simplify, $f = A \cdot B + ABC + \overline{A}B + A\overline{B}C$

Solution:

Here;

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

24. Prove the following, identity $\overline{\overline{AB} + \overline{A} + AB} = 0$

Solution:

Here;

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

25. Simplify $A\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}D + BC\overline{D} + \overline{A}B + B\overline{C}$

Solution:

Here;

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

26. Simplify $f(A, B, C, D) = (AB + C + D)(\overline{C} + D)(\overline{C} + D + E)$

Solution:

Here;

$$f(A, B, C, D) = (AB + C + D)(\overline{C} + D)(\overline{C} + D + E)$$
$$= (AB\overline{C} + C\overline{C} + \overline{C}D + ABD + CD + D)(\overline{C} + D + E)$$
$$= [AB\overline{C} + \overline{C}D + ABD + D(C + 1)](\overline{C} + D + E)$$
$$= (AB\overline{C} + \overline{C}D + D + ABD)(\overline{C} + D + E)$$
$$= AB\overline{C} + D\overline{C} + AB\overline{C}D + D + AB\overline{C}E + DE$$
$$= AB\overline{C} + D(1 + \overline{C} + AB\overline{C} + E) + AB\overline{C}E$$
$$= AB\overline{C}(1 + E) + D$$
$$= AB\overline{C} + D$$

27. Implement following expression with XOR and AND gate

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

Solution:

Here;

$$F = A\overline{B}C\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + \overline{A}B\overline{C}D$$
$$= A\overline{B}C\overline{D} + A\overline{B}\,\overline{C}D + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D$$
$$= A\overline{B}(C\overline{D} + \overline{C}D) + \overline{A}B(C\overline{D} + \overline{C}D)$$
$$= (A\overline{B} + \overline{A}B)(C\overline{D} + \overline{C}D)$$
$$= [A \oplus B][C \oplus D]$$

CHAPTER **4**

SIMPLIFICATION OF BOOLEAN FUNCTION

4.1 VENN DIAGRAM

A Venn diagram is a helpful illustration to visualize the relationship among the variable of a Boolean expression. The diagram consists of a rectangular inside two overlapping circles are drawn which represents two variables. Each circle is labeled by a variable. A Venn diagram may be used to illustrate the postulates of Boolean algebra or to explain the validity of theorems.

We designate all points inside a circle as belonging to the named variable and all points outside a circle as not belonging to the variable. *For example;* if the variables are A and B, then A = 1 for inside the circle A, A = 0 for outside of circle A, and B = 1 for inside the circle B, B = 0 for outside the circle B. Now for two overlapping circles as shown in figure below, four distinct areas are available inside the rectangle area belonging to A only or $A\overline{B}$, area belonging to B only or $\overline{A}B$, area belonging to both A and B *i.e.*, AB and area belonging to neither A or B *i.e.*, $\overline{A}\,\overline{B}$

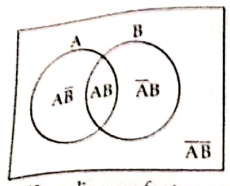


Figure: Venn diagram for two variables

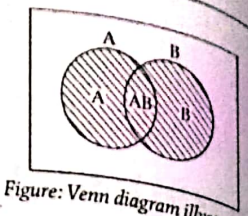


Figure: Venn diagram illystration of A = AB + A

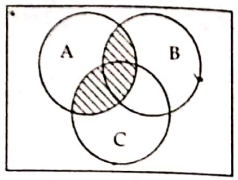


Figure: Venn diagram of the distributive law

It is possible to demonstrate eight distinct areas available for th variables in a Venn diagram.

4.2 KARNAUGH MAPS UPTO FIVE VARIABLES

The map method, first proposed by Veitch and slightly improvised Karnaugh, provides a simple, straight forward procedure for simplification of Boolean functions. The method is called Veitch diagr or Karnaugh map, which may be regarded either as a pictor representation of a pictorial representation of a truth table or as extension of the Venn diagram.

The Karnaugh map provides a systematic method for simplification manipulation of a Boolean expression. The map is a diagram consisting squares of squares. For n variable on a Karnaugh map there are numbers of squares. Each square or cell represents one of the minter Since any Boolean function can be expressed as a sum of minterms, i possible to recognize a Boolean function graphically in the map from t area enclosed by those squares whose minterms appear in the function. is also possible to drive alternative algebraic expressions or simplify t expression with a minimum number of variables or literals and sum products or product of sums terms by analyzing various patterns. In fa the map represents a visual diagram of all possible ways a function can expressed in a standard form and the simplest algebraic expres consisting of a sum of products or product of sums can be selected. N that the expression is not necessarity unique.

A. Two–Variable Karnaugh maps

A two variable Karnaugh map is shown in figure below.

	$\overline{B}$	B
$\overline{A}$	m_0	m_1
A	m_2	m_3

Since a two variable system can form four minterms, the map consists of four cells-one for each minterm. The map has been redrawn below to show the relationship between the squares and the two variables A and B.

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

In the first row, the variable A is complemented, in the second row A is uncomplemented, in the first column variable B is complemented and in the second column B is uncomplemented. The two variable K-map is a useful way to represent any of the Boolean functions of two variables, the squares are marked with 1 whose minterms belong to a certain function. As an example, the function AB has been in shown in figure below. Since the function is equal to the minterms m_3, a 1 is placed in the cell corresponding to m_3.

	$\overline{B}$	B
$\overline{A}$		
A		1

F = AB

Similarly, the function A + B has three minterms of $\overline{A}B$, $A\overline{B}$ and AB as,

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

So, the squares corresponding to $\overline{A}B$, $A\overline{B}$ and AB are marked with 1 as shown in figure below;

	$\overline{B}$	B
$\overline{A}$		1
A	1	1

F = A + B

B. Three variable Karnaugh maps

Since there are eight minterms for three variables, the map consists of eight cells or squares as shown in figure below. It may be noticed that the minterms are arranged not according to the binary sequence, but according to the sequence similar to the reflected code which means between two consecutive rows or columns, only one single variable changes its logic value from 0 to 1 or from 1 to 0.

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	m_0	m_1	m_3	m_2
A	m_4	m_5	m_7	m_6

Figure below shows the relationship between the squares and variables. Two rows are assigned to $\overline{A}$ and A and four columns to $\overline{B}\overline{C}$, $\overline{B}C$, BC and $B\overline{C}$

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

The three-variable K-map can be constructed in other ways too. If variable C is assigned to the rows and variables A and B are assigned along the columns.

$\overline{C}$	m_0	m_2	m_6	m_4
C	m_1	m_3	m_7	m_5

Figure below demonstrates where variable A is along columns and variables B and C are along the rows. Corresponding minterms are shown in figures.

	$\overline{A}$	A
$\overline{B}\overline{C}$	m_0	m_4
$\overline{B}C$	m_1	m_5
BC	m_3	m_7
$B\overline{C}$	m_2	m_6

Example 1

Simplify the Boolean function $F = \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$

Solution:

First, a three-variable K-map is drawn and 1s are placed at the squares according to the minterms of the function. Now two 1s of adjacent squares are grouped together. As in the figure, $\overline{A}BC$ and $\overline{A}B\overline{C}$ are grouped together at the first row and $A\overline{B}\overline{C}$ and $A\overline{B}C$ are grouped together. From the first row, the reduced term of $ABC + \overline{A}B\overline{C}$ is $\overline{A}B$, as C is the variable which changes its form.

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$			1	1
A	1	1		

Similarly, form the second row, $A\overline{B}\overline{C} + A\overline{B}C$ can be simplified to $A\overline{B}$. Now as further simplification is not possible for this particular Boolean function. The simplified sum of the products of the function can be written as, $F = \overline{A}B + A\overline{B}$.

Example 2

Simplify the expression $F = \overline{A}BC + A\overline{B}\overline{C} + ABC + AB\overline{C}$

Solution:

The K-map for this expression is,

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$			1	
A	1		1	1

In the third column, two adjacent squares are grouped together to produce the simplified term BC. The other two 1s are placed at the first column and last column of the same second row. Note that these 1s or minterms can be combined to produce a reduced term. Here the B variable is changing its form, from uncomplemented. After combining these two minterms, we get the reduced term $A\overline{C}$. This can be confirmed by applying the Boolean Algebra, $A\overline{B}\overline{C} + AB\overline{C} = A\overline{C}(B + \overline{B}) = A\overline{C}$

Thus, $F = BC + A\overline{C}$

NOTE:

It is possible to combine four adjacent squares consisting of 1s in the process of simplification of Boolean functions.

Example 3

Simplify the expression $F = \overline{A}\overline{B}C + \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}C + ABC$

Solution:

The K-map is shown below:

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$		1	1	1
A		1	1	

The four adjacent squares comprising the minterms $\overline{A}\overline{B}C$, $\overline{A}BC$, $A\overline{B}C$ and ABC can be combined. Here, it may observe that two of the variables A and B are changing their forms from uncomplemented to complemented. Therefore, these variable can be removed to form the reduced expressions.

Again two adjacent squares comprising the minterms $\overline{A}BC$ and $\overline{A}B\overline{C}$ can be combined to produce the reduced form $\overline{A}B$. So, the final simplified expression of the given function is $F = C + \overline{A}B$

Note that squares that are already considered in one group can be combined with other group or groups.

C. Four-Variable Karnaugh maps

Similar to the method used for two variable and three variable karnaugh maps, four variable K-map may be constructed with 16 squares containing of 16 minterms as shown in figure below.

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	m_0	m_1	m_3	m_2
$\bar{A}B$	m_4	m_5	m_7	m_6
AB	m_{12}	m_{13}	m_{15}	m_{14}
$A\bar{B}$	m_8	m_9	m_{11}	m_{10}

The same is drawn below to show the relationship with the four bina variables. The rows and columns are numbered in a reflected co sequence, where only one variable is changing its form between a adjacent squares.

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
$\bar{A}B$	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
AB	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
$A\bar{B}$	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

Different four variable K-map can be redrawn if the variables a assighed on another way.

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	m_0	m_4	m_{12}	m_8
$\bar{C}D$	m_1	m_5	m_{13}	m_9
CD	m_3	m_7	m_{15}	m_{11}
$C\bar{D}$	m_2	m_6	m_{14}	m_{10}

The minimization of four variable Boolean functions using K-map similar to the method used to minimize three variable functions. Tw four or eight adjacent squares can be combined to reduce the number literals in a function. The squares of the top and bottom rows as well a left-most and rightmost columns may be combined.

Example 4

Simplify the expression $F(A, B, C, D) = m_1 + m_5 + m_{10} + m_{11} + m_{12} + m_{13} + m_{15}$

Solution:

The K-map for above expression is,

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1		
$\bar{A}B$		1		
AB	1	1	1	
$A\bar{B}$			1	1

From the figure, it can be seen that four pairs can be formed. The simplified expression may be written as, $F = \bar{A}\bar{C}D + AB\bar{C} + ACD + A\bar{B}C$.

Note that the reduced expression is not a unique one, because if pairs are formed in different ways, the simplified expression will be different, But both expressions are logically correct.

The simplified expression of the given function as per the K-map is,

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

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1		
$\bar{A}B$		1		
AB	1	1	1	
$A\bar{B}$			1	1

Example 5

Simplify the expression $F(A, B, C, D) = m_7 + m_9 + m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15}$

Solution:

The K-map for above expression is shown below,

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$			1	
AB	1	1	1	1
$A\bar{B}$		1	1	1

Three quads and one pair are formed. The simplified expression of the given function is, $F = AB + AD + AC + BCD$.

Example 6

Plot the logical expression, $F(A, B, C, D) = ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C + AB$ on a four variable K-map obtain the simplified expression.

Solution:

To form a K-map for a logical expression, the function is to be expanded to either canonical SOP form or canonical POS form. The canonical SOP form for the above expression can be obtained as,

$$\begin{aligned} F(A, B, C, D) &= ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C + AB \\ &= ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C(D + \bar{D}) + AB(C + \bar{C})(D + \bar{D}) \\ &= ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} + (ABC + AB\bar{C})(D + \bar{D}) \\ &= ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} + ABCD + AB\bar{C}D + ABC\bar{D} \\ &\quad + AB\bar{C}\bar{D} \end{aligned}$$

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

$= \Sigma(8, 10, 11, 12, 13, 14, 15)$

The K-map for the above expression is shown below,

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$				
AB	1	1	1	1
$A\bar{B}$	1		1	1

Three guads are formed. The simplified expression is

$F = AB + AC + A\bar{D}$

Example 7

Simplify $F(W, X, Y, Z) = \Sigma(3, 4, 5, 7, 9, 13, 14, 15)$

Solution:

The K-map for above expression is,

$\bar{W}\bar{X}$			1	
$\bar{W}X$	1	1	1	
WX		1	1	1
$W\bar{X}$		1		

Four pairs are formed. It may be noted that one quad can also be formed but it is redundant as the squares contained by the quad are already covered by the pairs which are essential. The simplified expression is,

$F = \bar{W}X\bar{Y} + W\bar{Y}Z + WXY + \bar{W}YZ$

D. Five variable Karnaugh map

Karnaugh maps with more than five variable are not simple to use. The number of cells or squares become excessively large and combining the adjacent squares becomes complex. The number of cells or squares is away equal to the number of minterms. A five variable K-map contains or 32 cells which are used to simplify any five variable logic function.

	$\bar{C}\bar{D}\bar{E}$	$\bar{C}\bar{D}E$	$\bar{C}DE$	$\bar{C}D\bar{E}$	$CD\bar{E}$	CDE	$C\bar{D}E$	$C\bar{D}$
$\bar{A}\bar{B}$	m_0	m_1	m_3	m_2	m_6	m_7	m_5	m_4
$\bar{A}B$	m_8	m_9	m_{11}	m_{10}	m_{14}	m_{15}	m_{13}	m_{12}
AB	m_{24}	m_{25}	m_{27}	m_{26}	m_{30}	m_{31}	m_{29}	m_{28}
$A\bar{B}$	m_{16}	m_{17}	m_{19}	m_{18}	m_{22}	m_{23}	m_{21}	m_{20}

Figure below demonstrate five variables Karnaugh maps if the variable are assigned in different ways. The five variable K-maps have

properties similar to the two, three or four variable K-map described earlier *i.e.*, adjacent squares can be grouped together.

	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}BC$	$\bar{A}B\bar{C}$	$AB\bar{C}$	ABC	$A\bar{B}C$	$A\bar{B}\bar{C}$
$\bar{C}\bar{D}$	m_0	m_4	m_{12}	m_8	m_{24}	m_{28}	m_{20}	m_{16}
$\bar{C}D$	m_1	m_5	m_{13}	m_9	m_{25}	m_{29}	m_{21}	m_{17}
CD	m_3	m_7	m_{15}	m_{11}	m_{27}	m_{31}	m_{23}	m_{19}
$C\bar{D}$	m_2	m_6	m_{14}	m_{10}	m_{26}	m_{30}	m_{22}	m_{18}

Also,

	$\bar{D}\bar{E}$	$\bar{D}E$	DE	$D\bar{E}$
$\bar{A}\bar{B}\bar{C}$	m_0	m_1	m_3	m_2
$\bar{A}\bar{B}C$	m_4	m_5	m_7	m_6
$\bar{A}BC$	m_{12}	m_{13}	m_{15}	m_{14}
$\bar{A}B\bar{C}$	m_8	m_9	m_{11}	m_{10}
$AB\bar{C}$	m_{24}	m_{25}	m_{27}	m_{26}
ABC	m_{28}	m_{29}	m_{31}	m_{30}
$A\bar{B}C$	m_{20}	m_{21}	m_{23}	m_{22}
$A\bar{B}\bar{C}$	m_{16}	m_{17}	m_{19}	m_{18}

4.3 CANONICAL AND STANDARD FORMS

Logical functions are generally expressed in terms of different combinations of logical variables with their true form as well as the complement forms. An arbitary logic function can be expressed in terms of following forms;

i) Sum of the products (SOP)

ii) Product of the sums (POS)

1. Sum of the Products (SOP)

The logical sum of two or more logical product term is referred to as a sum of product expression. It is basically an OR operation on AND operated variables. *For example;* $Y = AB + BC + AC$ are sum of products expression.

2. Product of sums (POS)

The logical product of two or more logical sum terms is called a product of sums expression. It is an AND operation on OR operated variables. For example, $Y = (A + B + C)(A + \bar{B} + C)(A + B + \bar{C})$ are POS expressions.

Standard form

The standard form of the Boolean function is when it is expressed in sum of the products or product of the sums fashion. The example stated above like $Y = AB + BC + AC$ is standard form.

However, Boolean functions are also sometimes expressed in non-standard forms like $F = (AB + CD)(\overline{A}\,\overline{B} + \overline{C}\,\overline{D})$ which is neither a SOP form nor POS form. However, the same expression can be converted to standard form with help of various Boolean properties, as

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

4.3.1 Minterm

A product term containing all n variables of the function in either true or complemented form is called the minterm. Each minterm is obtained by an AND operation of the variables in their true form or complemented form. For a two variable function, four different combinations are possible such as $\overline{A}\,\overline{B}$, $\overline{A}\,B$, $A\,\overline{B}$ and AB. These products terms are called as fundamental products or standard products or minterms. In the minterm a variable will possess the value 1 if it is in true or uncomplemented form whereas it contains the value 0 if it is in complemented form. For three variable functions, eight minterms are possible as listed in the following table below;

A	B	C	Minterm
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$
0	0	1	$\overline{A}\,\overline{B}C$
0	1	0	$\overline{A}B\overline{C}$
0	1	1	$\overline{A}BC$
1	0	0	$A\overline{B}\,\overline{C}$
1	0	1	$A\overline{B}C$
1	1	0	$AB\overline{C}$
1	1	1	ABC

So, if the number of variables is n, then the possible number of minterms is 2^n. The main property of a minterm is that it possesses the value of 1 for only one combination of n input variables and the rest of the 2^n - 1 combinations have the logic value of 0. This means, for the above three variable examples, if A = 0, B = 1, C = 1 *i.e.*, for the input combination of 011, there is only one combination $\overline{A}BC$ that has the value 1, the rest of the seven combinations have the value 0.

Canonical Sum of Products Expression

When Boolean function is expressed as the logical sum of all the minterms from the rows of a truth table, for which the value of the function is 1, it is referred to as the conical sum of product expression. The same can be expressed in a compact form by listing the corresponding decimal equivalent codes of the minterms containing a function value of 1. The canonical sum of products form of a logic function can be obtained by using the following procedure.

i) Check each term in the given logic function. Retain if it is a minterm, continue to examine the next term in the same manner.

ii) Examine for the variables that are missing in each product which is not a minterm. If the missing variable in the minterm is X, multiply that minterm with $(X + \overline{X})$

iii) Multiply all the products and discard the redundant terms.

Example 8

Obtain the canonical sum of product form of the following function.

$$F(A, B) = A + B$$

Solution:

The given function contains two variables A and B. The variable B is missing from the first term of the expression and the variable A is missing from the second term of the expression. Therefore, the first term is to be multiplied by $(B + \overline{B})$ and the second term is to be multiplied by $(A + \overline{A})$.

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

Hence, the canonical sum of the product expression of the given function is,

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

Example 9

Obtain the canonical sum of product form of the following function.

$$F(A, B, C, D) = AB + ACD$$

Solution:

$$\begin{aligned} F(A, B, C, D) &= AB + ACD \\ &= AB(C + \overline{C})(D + \overline{D}) + ACD(B + \overline{B}) \\ &= (ABC + AB\overline{C})(D + \overline{D}) + ABCD + A\overline{B}CD \\ &= ABCD + ABC\overline{D} + AB\overline{C}\,\overline{D} + ABCD + A\overline{B}CD \\ &= ABCD + ABC\overline{D} + AB\overline{C}D + AB\overline{C}\,\overline{D} + A\overline{B}CD \end{aligned}$$

4.3.2 Maxterm

A sum term contaning all n variables of the function in either true or complemented form is called the maxterm. Each maxterm is obtained by an OR operation of the variables in their true form or complemented form. Four different combinations are possible for a two variable function, such as, $\overline{A}+\overline{B}$, $\overline{A}+B$, $A+\overline{B}$ and $A+B$. These sum terms are called the standard sums or maxterms. Note that, in the maxterm, a variable will possess the value 0, if it is in true or uncomplemented form whereas it contains the value 1 if it is in complemented form. Like minterms, for a three variable function, eight maxterms are also possible as listed in the following table.

A	B	C	Maxterm
0	0	0	$A + B + C$
0	0	1	$A + B + \overline{C}$
0	1	0	$A + \overline{B} + C$
0	1	1	$A + \overline{B} + \overline{C}$
1	0	0	$\overline{A} + B + C$
1	0	1	$\overline{A} + B + \overline{C}$
1	1	0	$\overline{A} + \overline{B} + C$
1	1	1	$\overline{A} + \overline{B} + \overline{C}$

So, if the number of variables is n, then the possible number of maxterm
The main property of a maxterm is that it possesses the value of 0 fo
one combination of n input variables and the rest of the 2^n -
inations have the logic value of 1.

nical product of sum expression

en a Boolean function is expressed as the logical product of all th
maxterms from the rows of a truth table for which the value of th
function is 0, it is referred to as the canonical product of the sum
expression. The same can be expressed in a compact form by listing th
corresponding decimal equivalent codes of the maxterms containing
function value of 0. The canonical product of sums form of a logic
function can be obtained by using the following procedure.

i) Check each term in the given logic function. Retain it if it is
maxterm, continue to examine the next term in the same manner.

ii) Examine for the variables that are missing in each sum term that
not a maxterm. If the missing variable in the maxterm is X
multiply that amxterm with $(X \cdot \overline{X})$.

iii) Expand the expression using the properties and postulates a
described earlier and discard the redundant terms.

Example 10

Obtain the canonical product of the sum form of the following function
$F(A, B, C) = (A + \overline{B})(B + C)(A + \overline{C})$

Solution:

In the above expression, C is missing from the first term, A is missing from the second term, and B is missing from the third term. Hence, $C\overline{C}$ is to be added with first term, $A\overline{A}$ is to be added with the second term and $B\overline{B}$ is to be added with the third term.

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

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

Conversion between canonical forms

In general, to convert from one canonical form to other canonical form, it is required to interchange the symbol Σ and π, and list the numbers which are missing from the original form. Note that, to find the missing terms, the total 2^n number of minterms or maxterms most be realized, where n is the number of variables in the function.

The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function. This is because the original function is expressed by those minterms that make the function equal to 1, whereas its complement is a 1 for those minterms that the function is a 0. As an example, consider the function,

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

This has a complement that can be expressed as,

$$F'(A, B, C) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$$

Now if we take the complement of F' by Demorgan's theorem, we obtain F in a different form.

$$F = (m_0 + m_2 + m_3)' = m_0' \cdot m_2' \cdot m_3' = M_0 M_2 M_3 = \pi(0, 2, 3)$$

The last conversion follows from the definition of minterms and maxterms. From the minterm and maxterm table, it is clear that the following relation holds true:

$$m_j' = M_j$$

That is, the maxterm with subscript j is a complement of the minterm with the same subscript j and vice-versa.

4.4 DON'T CARE CONDITIONS

In certain digital systems, some input combinations never occur during the process of a normal operation because those input conditions are guranteed never to occur. Such input combinations are called don't care combinations. The function output may be either 1 or 0 and these functions are called incompletely specified functions. These input combinations can be plotted on the Karnaugh map for further simplification of the function. They don't care conditions are represented by d or x or ϕ.

When an incompletely specified function *i.e.*, a function with don't care combinations is simplified to obtain minimal SOP expression, the value 1 can be assigned to the selected don't care combinations. This is done to form groups like pairs, quadoctet etc for further simplification. In each case choice depends only on need to achieve simplification. Similarly, selected don't care combinations may be assumed as 0s to form groups of 0s for obtaining the POS expression.

Example 11

Obtain the minimal sum of the products for the function $F(A, B, C, D) = \Sigma(1, 3, 7, 11, 15) + \phi(0, 2, 5)$.

Solution:
The K-map for the above function is,

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	X	1	1	X
$\bar{A}B$		X	1	
AB			1	
$A\bar{B}$			1	

In the K-map, the minterm m_0 and m_2 *i.e.*, $\bar{A}\bar{B}\bar{C}\bar{D}$ and $\bar{A}\bar{B}C\bar{D}$ are the don't care terms which have been assumed as 1's while making a quad. The simplified SOP expression of above function can be written as, $F = \bar{A}\bar{B} + C\bar{D}$

Example 12

Using the K-map method obtain the minimal sum of the product expression for the function.

$$F(A, B, C, D) = \Sigma(0, 2, 3, 6, 7) + d(8, 10, 11, 15)$$

Solution:
The K-map for the above function is,

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1		1	1
$\bar{A}B$		X	1	1
AB			X	
$A\bar{B}$	X		X	X

The simplified Boolean expression for the function is $F = \bar{A}C + \bar{B}\bar{D}$

Example 13

Using the karnaugh map method, obtain the minimal product of the sum expression for the function. $F(A, B, C, D) = \Sigma(0, 2, 3, 6, 7) + d(8, 10, 11, 15)$

Solution:
To derive the POS expression, the 0s of the K-map are considered as,

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		0		
$\bar{A}B$	0	0		
AB	0	0	X	0
$A\bar{B}$	X	0	X	X

The simplified Boolean expression for the function is $\bar{F} = A + \bar{C}D + B\bar{C}$

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

4.5 LOGIC GATES IMPLEMENTATION

Digital circuits are more frequently constructed with NAND or NOR gates than with AND and OR gates. NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families. Because of the promience of NAND and NOR gates in the design of digital circuits, rules and procedures have been developed for the conversion from Boolean functions given interms of AND, OR and NOT into equivalent NAND and NOR logic diagrams.

A. NAND Implementation

The implementation of a Boolean function with NAND gates requires that the function be simplified in the sum of products form. To see the relationship between a sum of products expression and its equivalent NAND implementation, consider the logic diagram. All three diagrams are equivalent and implement the function

$$F = AB + CD + E$$

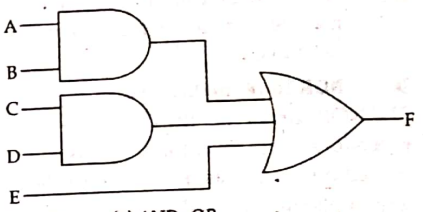


Figure: (a) AND–OR

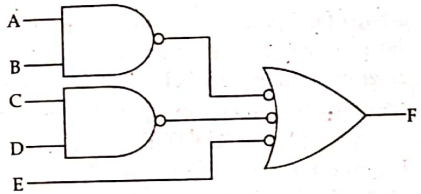


Figure: (b) NAND–NAND

In figure (b), the AND gates are replaced by NAND gates and the OR gate is replaced by a NAND gate with an invert OR symbol. The single variable E is complemented and applied to the second level invert OR gate.

The figure (a) and figure (b) implement the same function and are equivalent. In figure (c) the output NAND gate is redrawn with the conventional symbol. The one input NAND gate complements variable E. It is possible to remove this inverter and apply $\bar{E}$ directly to the input of the second level NAND gate. The diagram in (c) is equivalent to the one in (b) which in turn is equivalent to the diagram in (a).

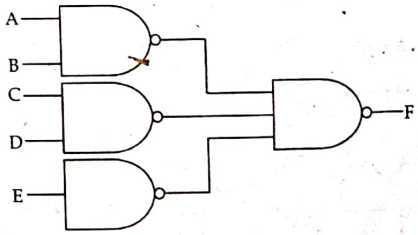


Figure: NAND–NAND

Figure: Three ways to implement F = AB + CD + E

The rule for obtaining the NAND logic diagram from a Boolean fun... is as follows,

i) Simplify the function and express it in SOP

ii) Draw a NAND gate for each product term of the function that ... at least two literals. The inputs to each NAND gate are the lite... of the term. This constitutes a group of first level gates.

iii) Draw a single NAND gate in the second level, with inputs com... from outputs of first level gates.

iv) A term with a single literal requires an inverter in the first leve... may be complemented and applied as an input to the second le... NAND gate.

B. NOR Implementation

The NOR function is the dual of the NAND function. For this reason, ... procedures and rules for NOR logic are the duals of the correspond... procedures and rules developed for NAND logic.

The implementation of a Boolean function with NOR gates requires ... the function be simplified in POS form. A POS expression specifie... group of OR gates for the sum terms followed by an AND gate to prod... the product. The transformation from the OR-AND to the NOR-... diagram is depicted below. It is similar to the NAND transformat... discussed previously, except that now are use the POS expression,

$$F = (A + B)(C + D)E$$

The rule for obtaining the NOR logic diagram from a Boolean funct... can be derived from this transformation.

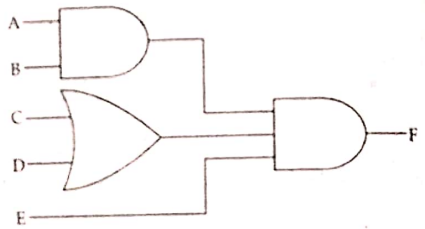


Figure: (a) OR–AND

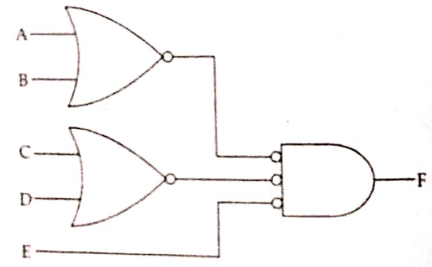


Figure: (b) NAND–NAND

It is similar to the three step NAND rule, except that the simplified expression must be in the POS and the terms for the first level NOR gates are the sum terms. A term with a single literal requires a one-input NOR or inverter gate or may be complemented and applied directly to the second level NOR gate.

A second way to implement a function with NOR gates would be to use the expression for the complement of the function in POS. This will give a two level implementation for $\bar{F}$ and a three level implementation if the normal output F is required.

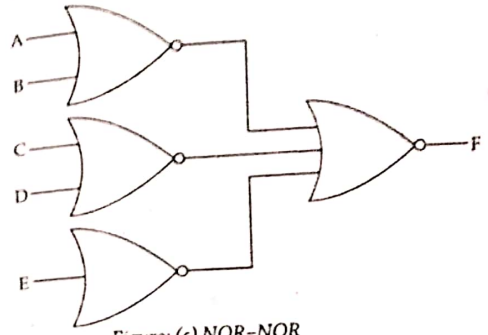


Figure: (c) NOR–NOR

To obtain the simplified product of sums from a map, it is necessary to combine the 0s in the map and then complement the function. To obtain the simplified product of sums expression for the complement of the function.

EXAMINATION QUESTION SOLUTIONS

1. **Write short notes on Venn diagram.** **[2011/S, 2012/S, 2017/**

Solution: See the topic 4.1.

2. **Reduce the expression F(A, B, C, D) = Σ(0, 6, 8, 13, 14) d(A, B, C, D) = Σ(2, 4, 10). Express the simplified function in sum of minterms.** **[2011/**

Solution:

F(A, B, C, D) = Σ(0, 6, 8, 13, 14)

d(A, B, C, D) = Σ(2, 4, 10)

K-map

AB \ CD	00	01	11	10
00	1			X
01	X			1
11		1		1
10	1			X

From K-map,

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

For sum of minterms,

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

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

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

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

$+ AB\bar{C}\bar{D} + ABC\bar{D}$

3. **Simplify the Boolean function F using don't care conditions d = (i) SOP (ii) POS and (iii) draw NAND-NAND equivalent logic. Given that;** **[2011/F, 2015/S, 2015/F, 2019/S (Simila**

$F = A'B'D' + A'CD + A'BC$

$d = A'BC'D + ACD + AB'D'$

Solution:

$F = A'B'D' + A'CD + A'BC$

$d = A'BC'D + ABC + AB'D'$

Expressing terms in minterms,

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

$= A'B'C'D' + A'B'CD' + A'B'CD + A'BCD + A'BCD' + A'BC$

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

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

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

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

i) K-map for SOP

AB \ CD	00	01	11	10
00	1		1	1
01	X		1	1
11			X	
10	X		X	X

From K-map,

$F = \bar{B}\,\bar{D} + BC$

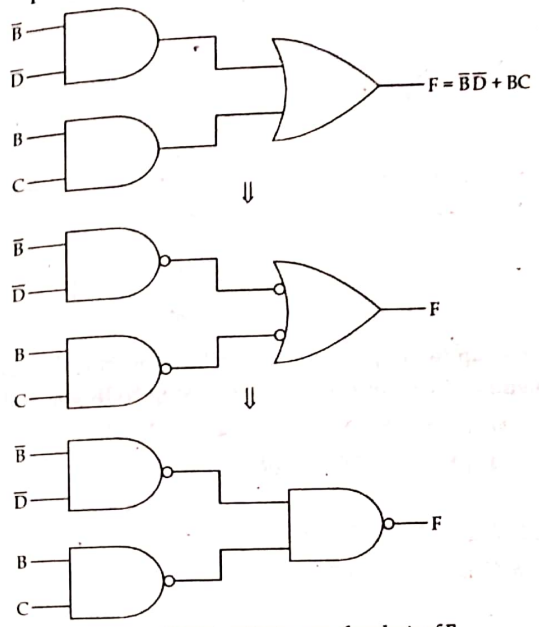


Figure: NAND–NAND equivalent logic of F

ii) K-map for POS

AB \ CD	00	01	11	10
00	0	0		
01	0	X		
11	0	0	X	0
10	X	0	X	X

$\therefore \quad \bar{F} = \bar{C}D + A + B\bar{C}$

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

EXAMINATION QUESTION SOLUTIONS

1. Write short notes on Venn diagram. [2011/S, 2012/S, 2...

Solution: See the topic 4.1.

2. Reduce the expression F(A, B, C, D) = Σ(0, 6, 8, 13, 14) ... d(A, B, C, D) = Σ(2, 4, 10). Express the simplified function in ... of minterms. [2...

Solution:

F(A, B, C, D) = Σ(0, 6, 8, 13, 14)

d(A, B, C, D) = Σ(2, 4, 10)

K-map

AB \ CD	00	01	11	10
00	1			X
01	X			1
11		1		1
10	1			X

From K-map,

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

sum of minterms,

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

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

$+ \overline{A}\,\overline{B}C\overline{D} + A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + AB\overline{C}D$

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

$+ AB\overline{C}D + ABC\overline{D}$

3. Simplify the Boolean function F using don't care conditions ... (i) SOP (ii) POS and (iii) draw NAND-NAND equivalent logic. [2011/F, 2015/S, 2015/F, 2019/S (Simil...

Given that;

F = A'B'D' + A'CD + A'BC

d = A'BC'D + ACD + AB'D'

Solution:

F = A'B'D' + A'CD + A'BC

d = A'BC'D + ABC + AB'D'

Expressing terms in minterms,

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

= A'B'C'D' + A'B'CD' + A'B'CD + A'BCD + A'BCD' + A'B...

$= \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}BCD$

and, $d = \overline{A}BC\overline{D} + ACD(B + \overline{B}) + A\overline{B}\,\overline{D} + (C + \overline{C})$

$= \overline{A}BC\overline{D} + ABCD + A\overline{B}CD + A\overline{B}C\overline{D} + A\overline{B}C\overline{D}$

$= \overline{A}BC\overline{D} + A\overline{B}C\overline{D} + A\overline{B}CD + A\overline{B}CD + ABCD$

i) K-map for SOP

AB \ CD	00	01	11	10
00	1		1	1
01	X		1	1
11			X	
10	X		X	X

From K-map,

$F = \overline{B}\,\overline{D} + BC$

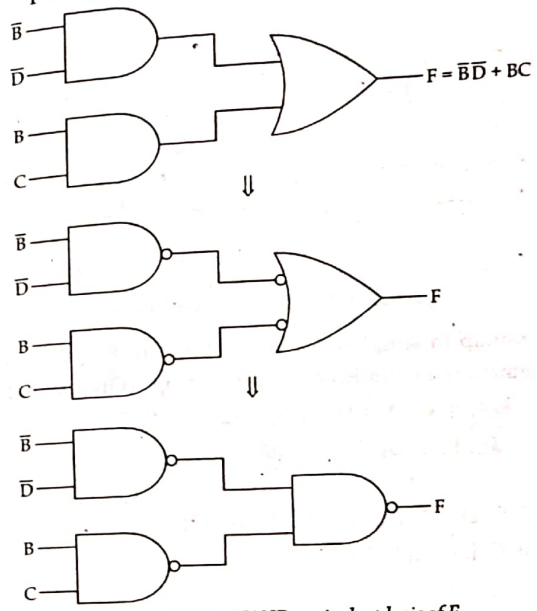


Figure: NAND–NAND equivalent logic of F

ii) K-map for POS

AB \ CD	00	01	11	10
00	0	0		
01	0	X		
11	0	0	X	0
10	X	0	X	X

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

or, $F = (\overline{C}D + A + B\overline{C})'$

or, $F = (C'D)'A'(BC')'$

$\therefore \quad F = (C + D')A'(B' + C)$

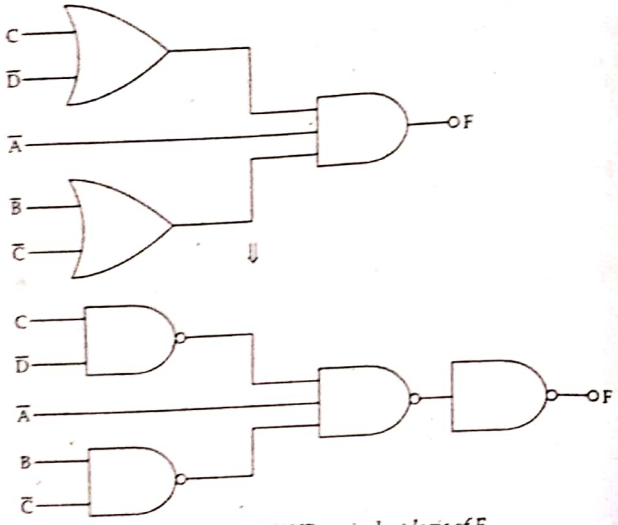


Figure: NAND–NAND equivalent logic of F

4. Use K-map to simplify the given Boolean function in POS and implement the simplified function using NOR gate only. [2012/5]

F(A, B, C, D) = Σ(1, 3, 8, 9, 12, 13, 14, 15)

d(A, B, C, D) = Σ(2, 7, 10)

Solution:

F(A, B, C, D) = Σ(1, 3, 8, 9, 12, 13, 14, 15)

d(A, B, C, D) = Σ(2, 7, 10)

map

AB \ CD	00	01	11	10
00	0	1	1	X
01	0	0	X	0
11	1	1	1	1
10	1	1	0	X

First, take the simplified function with 0's in map. This gives complement of function in sum of product. Then if we complement that function, we get the required function in POS.

From K-map,

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

Now, $F = (\overline{F})' = (A'C'D' + A'B + AB'C)'$

$= (A'C'D')'\,(A'B)'\,(AB'C')$ (Demorgan's law)

$= [(A')' + (C')' + (D')'][(A')' + B'][A' + (B')' + (C')']$

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

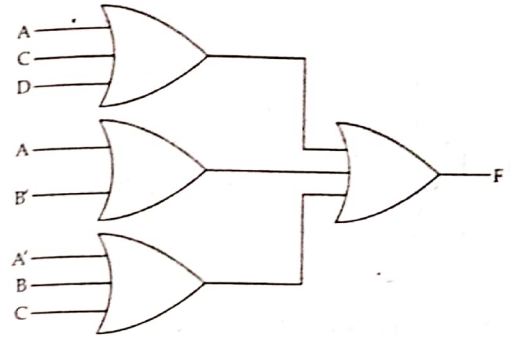


Figure: Basic Implementation

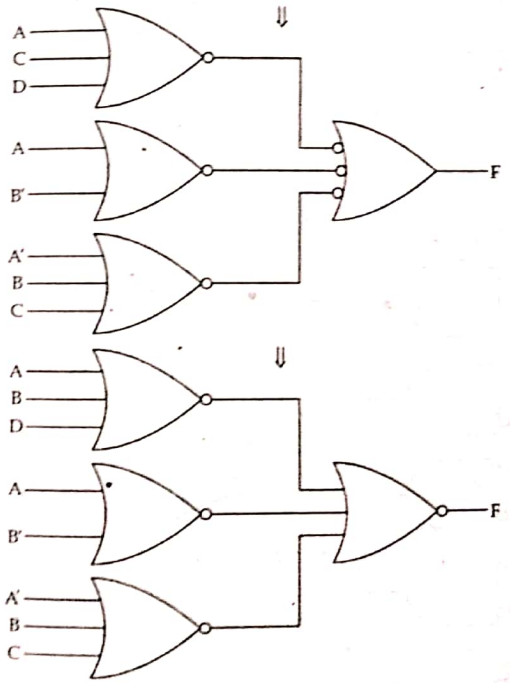


Figure: NOR-NOR implementation of function F

5. **Simplify the following Boolean function $F(W, X, Y, Z) = \Sigma(1, 6, 7, 8, 11, 13)$, $d(W, X, Y, Z) = \Sigma(0, 2, 3, 4, 10, 12)$ and then implement the function using NOR gates only. [2012/F]**

Solution:

$F(W, X, Y, Z) = \Sigma(1, 6, 7, 8, 11, 13)$

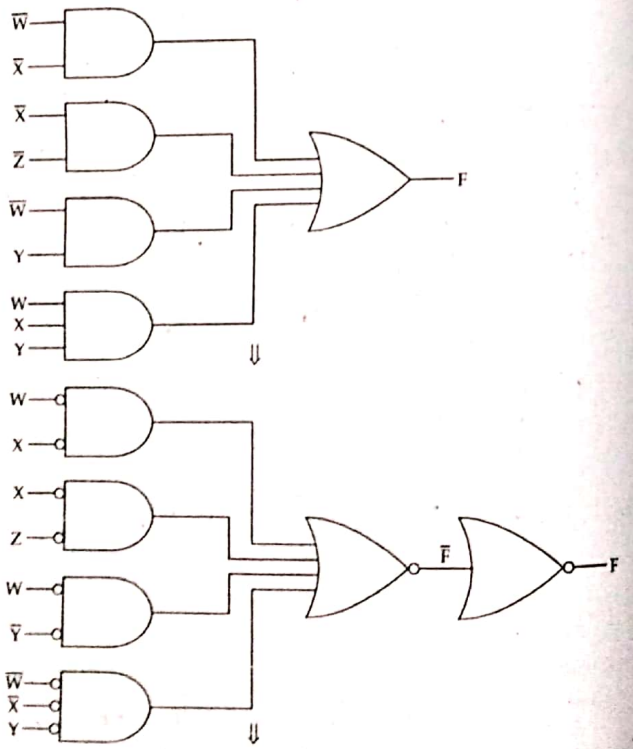
$d(W, X, Y, Z) = \Sigma(0, 2, 3, 4, 10, 12)$

K-map

WX \ YZ	01	01	11	10
00	X	1	X	X
01	X		1	1
11	X	1		
10	1		1	X

From Table,

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



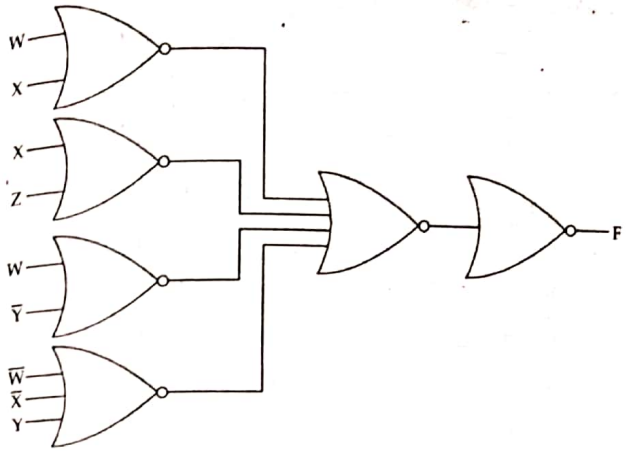


Figure: Implementation of function using NOR gates only.

6. **Reduce the given expression in minimum number of literals using Boolean algebra and derive the truth table and implement in NAND logic $A + B[AC + B[AC + (B + \overline{C})\,D]]$**

Solution:

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

$= A + B\,[AC + B\,[AC + BD + \overline{C}D]]$ [$\because (x(y + z) = xy + xz$]

$= A + B\,[AC + ABC + B\,(BD + \overline{C}D)]$

$= A + B\,[AC + ABC + BD + B\overline{C}D]$ [$\because x \cdot x = x$]

$= A + B\,[AC\,(1 + B) + BD\,(1 + \overline{C})]$

$= A + B\,[AC + BD]$ [$\because 1 + x = 1$ and $1.x = x$]

$= A + ABC + BD$

$= A\,(1 + BC) + BD$ [$\because 1 + x = 1$]

$\therefore \quad F = A + BD$

Truth Table:

Here C is don't care condition

$F = A + BD$

A	B	C	D	F = A + BD
0	0	X	0	0
0	0	X	1	0
0	1	X	0	0
0	1	X	1	1
1	0	X	0	1
1	0	X	1	1
1	1	X	0	1
1	1	X	1	1

Now, Implementation in NAND gate only

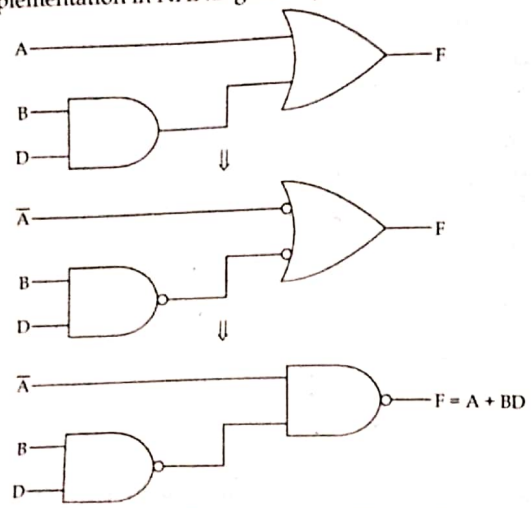


Figure: NAND implementation of function F

A logic circuit implements the following Boolean function $F = \overline{A}C + A\overline{C}\,\overline{D}$. It is found that the circuit input combination A = C = 1 can never occur. Using k-map with proper don't care conditions find a simplified expression and implement it using NAND gates only. [2013/S]

Solution:

$F = \overline{A}C + A\overline{C}\,\overline{D}$ $\qquad$ $d = AC$

Expressing these terms in minterms,

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

and, $d = AC(D + \overline{D}) = ACD + AC\overline{D}$

The 3-variable k-map is,

A \ CD	00	01	11	10
0			1	1
1	1	0	X	X

From K-map

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

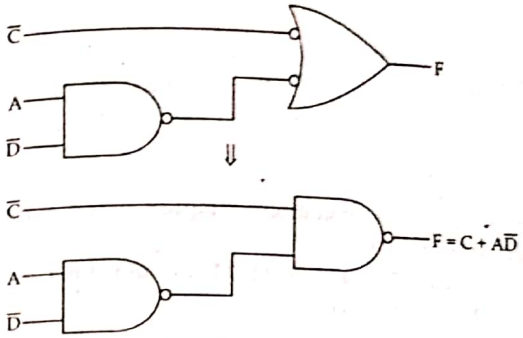


Figure: NAND gates implementation of F

8. **Design a logic circuit to implement the Boolean function. [2013/F]**
F(A, B, C, D) = Σ(1, 3, 7, 11, 15)
D(A, B, C, D) = Σ(0, 2, 5) in the terms of
(i) Sum of products (ii) Implement with NAND-NAND gate only

Solution:

F(A, B, C, D) = Σ(1, 3, 7, 11, 15), $\qquad$ D(A, B, C, D) = Σ(0, 2, 5)

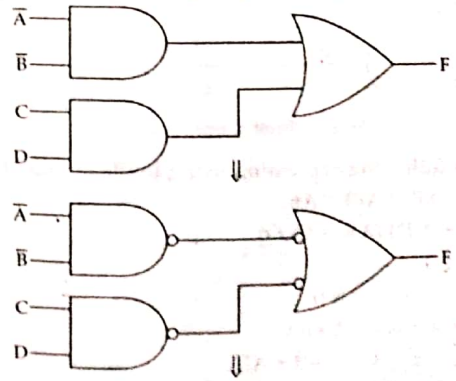
K-map

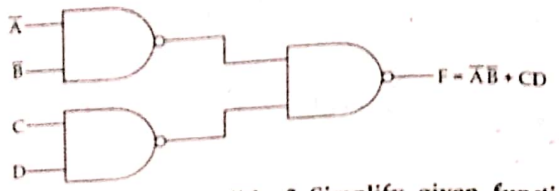
AB \ CD	00	01	11	10
00	X	1	1	X
01		X	1	
11			1	
10			1	

∴ $F = \overline{A}\,\overline{B} + CD$

i) Sum of minterms, $F = \overline{A}\,\overline{B} + CD$

ii) Implementation using only NAND gate.





9. What is don't care condition? Simplify given function using K-map with circuit design.
F(W, X, Y, Z) = Σ(1, 4, 5, 6, 12, 14, 15) and don't are condition D(W, X, Y, Z) = Σ(10, 11) **[2014/S]**

Solution:

For definition of don't care condition, see topic 4.4.

F(W, X, Y, Z) = Σ(1, 4, 5, 6, 12, 14, 15)

D(W, X, Y, Z) = Σ(10, 11)

K-map

WX \ YZ	00	01	11	10
00		1		
01	1	1		1
11	1		1	1
10			X	X

∴ F = $\overline{W}\,\overline{Y}Z + X\overline{Z} + WY$

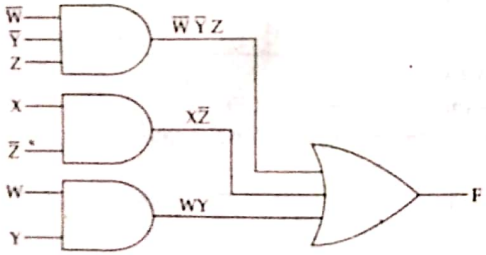


Figure: Circuit design

10. Simplify the following expression using Boolean algebra.[2014/F]

i) (AB′ + ABC)′A(B + AB′)

ii) [(BC″ + A′D) (AB′ + CD′)]′

Solution:

i) (AB′ + ABC)′ + A(B + AB′)

= (AB′)′ (ABCY + AB + AAB′)

= (A′ + B) (A′ + B′ + C′) + AB + AB′

= A′ + B(C + C′) + A(B + B′)

= A′ + BB′ + BC′ + A [∵ B + $\overline{B}$ = 1]

= A + A′ + BC′ [∵ A + $\overline{A}$ = 1]

= BC′

ii) [(BC″ + A′D) (AB′ + C′D′)]′

= (BC′ + A′D)′ + (AB′ + C′D′)′ [∵ (xy)′ = x′ + y′]

= [(BC)′ (A′D)′] + [(AB′)′ (C′ D′)′]

= [(B′ + C′) (A + D′)] + [(A′ + B) (C + D)]

= (B′ + C′)A + (B′ + C′)D′ + (A′ + B)C + (A′ + B)D

[∵ x(y + z) = xy + xz]

= AB′ + AC′ + B′D′ + C′D′ + A′C + BC + A′D + BD

= AB′ + AC′ + A′C + B′D′ + BD

= AB′ + A⊕C + B⊙D

11. Given the function F = xy + x′y′ + y′z **[2014/F]**

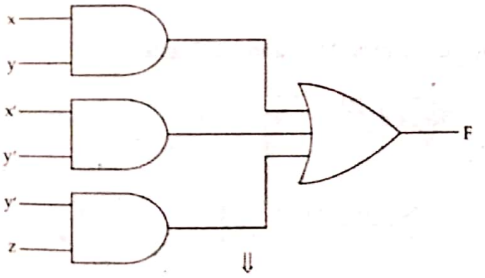
i) Implement it with OR and NOT gate.

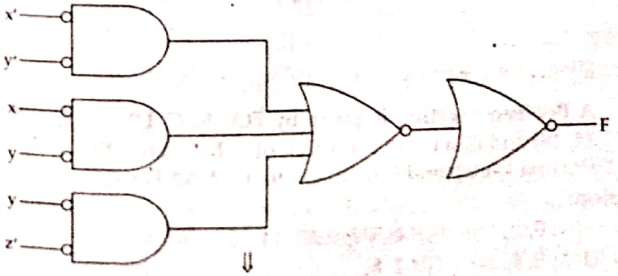
ii) Implement it with AND and NOT gate.

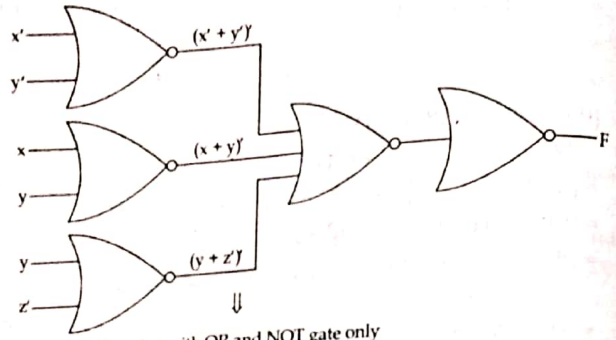
F = xy + x′y′ + y′z

Solution:

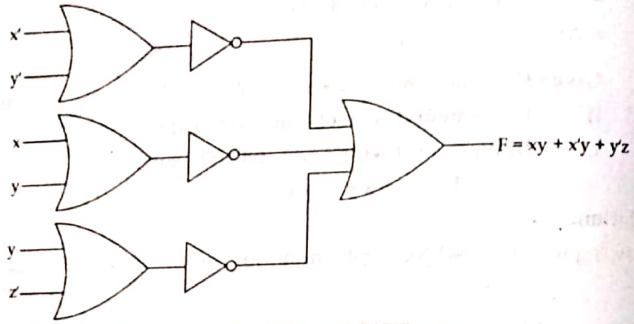
Drawing with OR and NOT gate means implementing with NOR gate



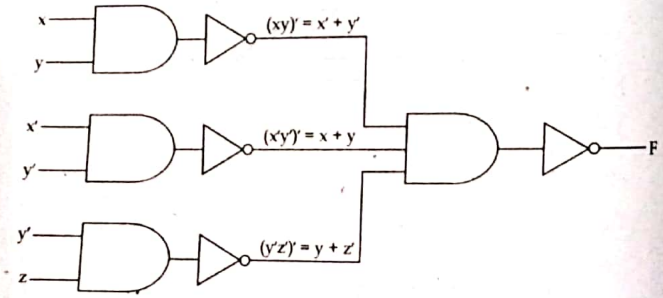




Drawing with OR and NOT gate only



ii) Implementing using only AND and NOT gate



∴ $F = (x' + y')' + (x + y)' + (y + z')' = xy + x'y' + y'z$

12. A Boolean function is given by F(A, B, C, D) = Σ(3, 4, 5, 7, 9, 13, 14, 15) and don't care condition d(A, B, C, D) = Σ(0, 2, 8). Simplify it using k-map and implement using NAND gate only. [2014/F]

Solution:

F(A, B, C, D) = Σ(3, 4, 5, 7, 9, 13, 14, 15)

d(A, B, C, D) = Σ(0, 2, 8)

K-map

AB \ CD	00	01	11	10
00	X		1	X
01	1	1	1	
11		1	1	1
10	X	1		

∴ $F = A'C'D' + AB'C' + A'B'C + ABC + BD$

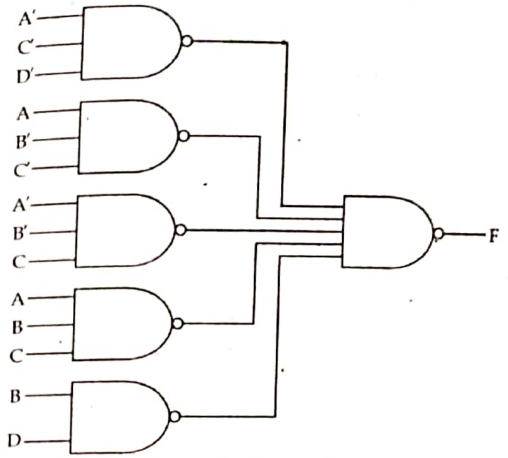


Figure: Implementing F using NAND gate only

13. Simplify the Boolean expression using Boolean algebra. [2015/S]

i) (AB' + AB'C')' + A (B' + AB')

ii) [(BC' + A'D) (AB' + C'D')]'

Solution:

i) (AB' + AB'C')' + A (B' + AB')

= (AB' + AB'C')' + AB' [∵ x + xy = x]

= [(AB')' (AB'C')'] + AB' [∵ (x + y)' = x' y']

= [(A' + B'') (A' + B'' + C'') + AB' [∵ (xy)' = x' + y']

= [(A' + B) (A' + B + C)] + AB' [∵ x(x + y) = x]

= A' + B + AB' [Here, x = (A' + B), y = c]

ii) [(BC' + A' D) (AB' + C' D')]'

See the Q. N 10 (ii).

14. A Boolean function is given by F(A, B, C, D) = Σ(3, 4, 6, 8, 10, 12, 14) and don't care condition d(A, B, C, D) = Σ(0, 28). Simplify it using K-map and amplify it using k-map and implement using NAND gate only. [2015/S, 2015/F]

Solution;

$F(A, B, C, D) = \Sigma(3, 4, 6, 8, 10, 12, 14)$

$d(A, B, C, D) = \Sigma(0, 2, 8)$

K-map

AB \ CD	00	01	11	10
00	X		1	X
01	1			1
11	1			1
10	1			1

$\therefore$ $F = D' + A'B'C$

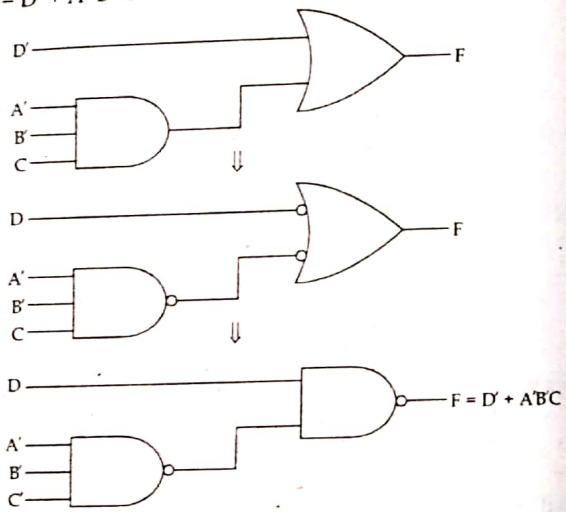


Figure: NAND gate implementation of F

15. Distinguish between minterms and maxterms. [2012F

Solution: See the topic 4.3.1 and 4.3.2.

16. Prove the following Boolean expression. [2015/F, 2019S

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

ii) $X\bar{Y} + Y\bar{Z} + Z\bar{X} = \bar{X}Y + \bar{Y}Z + \bar{Z}X$

Solution:

i) $\bar{A}\bar{B} + BC + \bar{A}B\bar{C}$

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

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

Using K-map,

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
A			1	
$\bar{A}$	1	1	1	1

From K-map,

$F = \bar{A} + BC$

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

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

$= (X\bar{Y}Z + X\bar{Y}\bar{Z} + XY\bar{Z} + \bar{X}Y\bar{Z} + \bar{X}YZ + \bar{X}\bar{Y}Z$

Z \ XY	$\bar{X}\bar{Y}$	$\bar{X}Y$	XY	$X\bar{Y}$
Z		1	1	1
$\bar{Z}$	1	1		1

From K-map, $F = \bar{X}Z + Y\bar{Z} + X\bar{Y}$

Hence, $X\bar{Y} + Y\bar{Z} + Z\bar{X} = \bar{X}Z + Y\bar{Z} + X\bar{Y}$

17. A Boolean function is given: F(W, X, Y, Z) = Σ(1, 4, 5, 6, 12, 14, 15) and don't care condition d(W, X, Y, Z) = Σ(1011). Simplify it using k-map with logic gate implementation. **[2016/F]**

Solution:

$F(W, X, Y, Z) = \Sigma(1, 4, 5, 6, 12, 14, 15)$, $d(W, X, Y, Z) = \Sigma(1011) = \Sigma 11$

K-map

WX \ YZ	00	01	11	10
00		1		
01	1	1		1
11	1		1	1
10			X	

$\therefore$ $F = X\bar{Z} + \bar{W}\bar{Y}Z + WYZ$

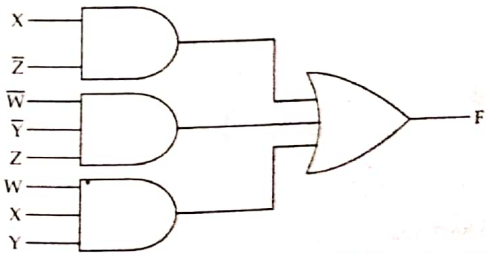


Figure: Logic gate implementation

18. Define literal and term. Find canonical SOP for this expression F = ac + ab + bc. [2016/F]

Solution:

A literal is a primed or unprimed variable. When a Boolean function is implemented with logic gates, each literal in the function designates an input to a gate and each term is implementated with a gate. For example, The function F(A, B, C, D) = A + $\overline{B}\overline{C}$ + AC$\overline{D}$ contains 6 literals.

Given that;

F = ac + ab + bc

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

= abc + a$\overline{b}$c + abc + ab$\overline{c}$ + abc + $\overline{a}$bc

∴ F = $\overline{a}$bc + ab$\overline{c}$ + a$\overline{b}$c + abc

19. What is don't care condition? Simplify given function using K-map and implement it in a circuit. [2016/S]

F(W, X, Y, Z) = Σ(1, 4, 5, 6, 12, 14, 15)

d(W, X, Y, Z) = Σ(11, 13)

Solution:

W, X, Y, Z) = Σ(1, 4, 5, 6, 12, 14, 15)

X, Y, Z) = Σ(11, 13)

WX \ YZ	00	01	11	10
00		1		1
01	1	1		1
11	1	X	1	1
10			X	

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

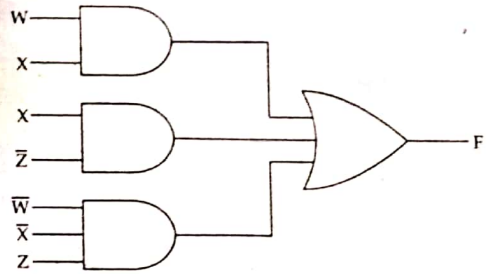


Figure: Logic gate implementation

20. Write short notes on don't care conditions. [2017/F]

Solution: See the topic 4.4.

21. Use K-map to simplify function in SOP form and implement the simplified function using NAND gate only. F(A, B, C, D) = Σ(5, 7, 9, 12, 13, 14, 15) and don't care d(A, B, C, D) = Σ(3, 6, 8). [2017/F]

Solution:

F(A, B, C, D) = Σ(5, 7, 9, 12, 13, 14, 15), d(A, B, C, D) = Σ(3, 6, 8)

K-map

AB \ CD	00	01	11	10
00			X	
01		1	1	1
11	1	1	1	1
10	X	1		

∴ F = A$\overline{C}$ + BD + BC

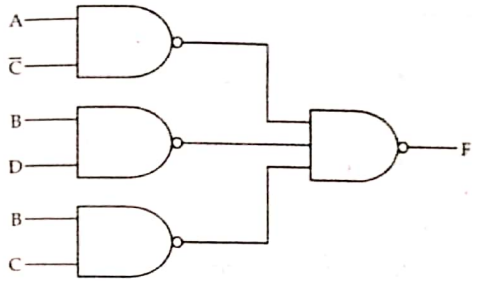


Figure: NAND gate implementation

22. Use K-map to simplify the given Boolean function and once by considering the don't care condition and once by ignoring the don't care condition and realize it using the basic gates. [2017/S]

F(A, B, C, D) = Σ(1, 4, 8, 12, 13, 15)

d(A, B, C, D) = Σ(3, 14)

Solution:

F(A, B, C, D) = Σ(1, 4, 8, 12, 13, 15)

d(A, B, C, D) = Σ(3, 14)

i) With don't care condition

AB \ CD	00	01	11	10
00		1	X	
01	1			
11	1	1	1	X
10	1			

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

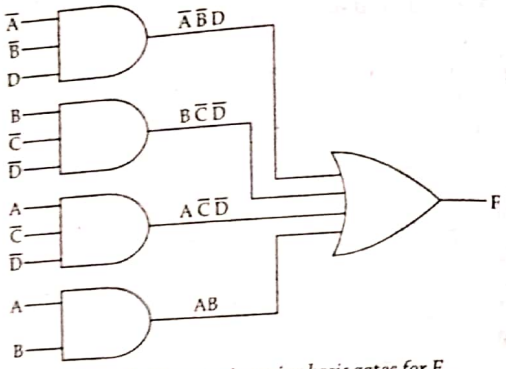


Figure: Implementation using basic gates for F

ii) Without don't care condition

AB \ CD	00	01	11	10
00		(1)		
01	1			
11	1	1	1	
10	1			

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

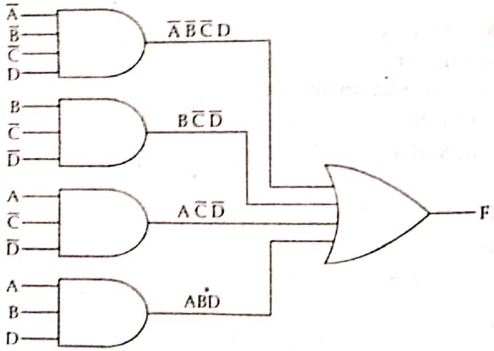


Figure: Implementation using basic gates

23. Use K-map to simplify the given Boolean function with don't care condition and realize it using basic gates only. [2018/F]

$F = \Sigma(1, 4, 8, 12, 13, 15)$

$d = \Sigma(3, 7, 11, 14)$

Solution:

K-map;

AB \ CD	00	01	11	10
00		1	X	
01	1		X	
11	1	1	1	X
10	1		X	

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

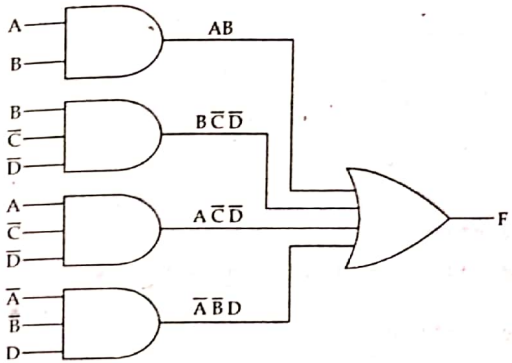


Figure: Implementation of F using basic gates

24. Define K-map simplify the expression using K-map. [2018/S]

$F(A, B, C, D) = \Sigma(1, 3, 7, 10, 13, 15)$

$d(A, B, C, D) = \Sigma(0, 2, 8)$

Also, implement the simplified function using NOR gates only.

Solution:

$F(A, B, C, D) = \Sigma(1, 3, 7, 10, 13, 15)$

$d(A, B, C, D) = \Sigma(0, 2, 8)$

K-map;

AB \ CD	00	01	11	10
00	X			X
01	0	0		0
11	0			0
10	X	0	0	

NOTE

Taking product of sum instead of sum of product to easily implement in NOR gates.

From K-map,

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

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

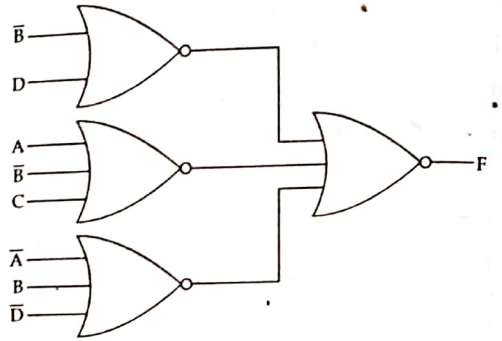


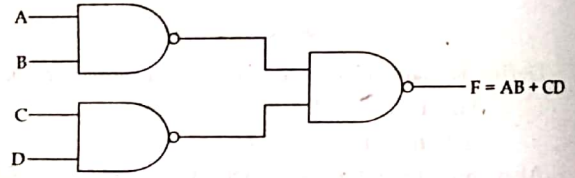
Figure: Implementation of F-using NOR gate only

For definition refer topic 4.2.

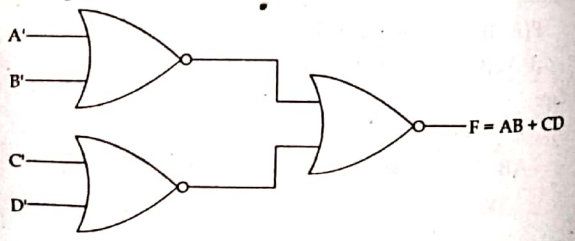
25. Construct F = AB + CD using universal gates. **[2018/S]**

Solution:

i) Using NAND gate



ii) Using NOR gate



26. Simplify the following Boolean expression using K-map and implement using NOR gates only **[19/F]**

F(A, B, C, D) = Σ(0, 2, 4, 6, 12, 15)

d(A, B, C, D) = Σ(8, 10, 14)

Solution:

K-map;

AB \ CD	00	01	11	10
00		0	0	
01		0	0	
11		0		X
10	X	0		X

Using POS instead of sop to implement on NOR gates,

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

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

$\therefore \quad F = (A' + B)(C + \overline{D})(A + \overline{D})$

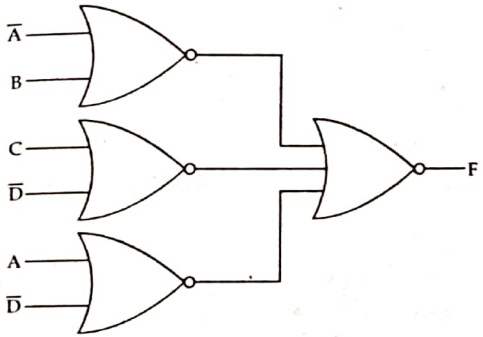


Figure: NOR gate implementation

27. Write short notes on POS and SOP. **[2016/S]**

Solution: See the topic 4.3.

ADDITIONAL QUESTION SOLUTIONS

1. Minimize the K-map shown in figure.

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1		
$\bar{A}B$		1	1	1
AB	1	1	1	
$A\bar{B}$			1	

Solution:

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1		
$\bar{A}B$		1	1	1
AB	1	1	1	
$A\bar{B}$			1	

From K-map

$\therefore \quad Z = AB\bar{C} + \bar{A}BC + \bar{A}\bar{C}D + ACD = A(B\bar{C} + CD) + A(BC + \bar{C}D)$

2. Use the K-map to simplify the expression $x = \bar{A}\bar{B}\bar{C} + \bar{B}C + \bar{A}B$

Solution:

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

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	1	1		
C	1	1		1

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

3. Simplify $F(\bar{a}, b, \bar{c}) = \bar{a}bc + b\bar{c} + ab\bar{c} + a\bar{b}c$ using K-map.

Solution:

$$F = \bar{a}bc + b\bar{c} + ab\bar{c} + a\bar{b}c$$
$$= \bar{a}bc + b\bar{c}(a + \bar{a}) + ab\bar{c} + a\bar{b}c$$
$$= \bar{a}bc + ab\bar{c} + a\bar{b}c$$

K-map;

	$\bar{b}\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$		1	1	1
a	1	1		1

$\therefore \quad F = b\bar{c} + \bar{a}b + a\bar{b}c$

4. Simplify $\bar{a}b + \bar{b}c + \bar{c}a$ using K-map.

Solution:

$$F = \bar{a}b + \bar{b}c + \bar{c}a = \bar{a}b(c + \bar{c}) + \bar{b}c(a + \bar{a}) + \bar{c}a(b + \bar{b})$$
$$= \bar{a}bc + \bar{a}b\bar{c} + a\bar{b}c + \bar{a}\bar{b}c + ab\bar{c} + a\bar{b}\bar{c}$$

K-map;

	$\bar{a}\bar{b}$	$\bar{a}b$	ab	$a\bar{b}$
$\bar{c}$		1	1	1
c	1	1		1

$\therefore \quad F = \bar{a}c + b\bar{c} + a\bar{b}$

5. Given the following Boolean function:

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

i) Express it in sum of minterms.

ii) Find the minimal sum of products expression.

Solution:

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

i) $F = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC$

ii) K-map;

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$		1	1	1
A		1	1	

$\therefore \quad F = \bar{A}B + C.$

6. What is the difference between canonical form and standard form?

Solution:

In canonical from, a Boolean function is expressed as sum of minterms or product of maxterms and is obtained by reading a function from the truth table. Canonical forms may or may not contain the least numbers of literals because by definition each maxterm or minterm must contain all variables, complemented or uncomplmented. In standard from, the terms of function may contain one, two or any number of literals. Standard form is expressed as sum of products or product of sums.

7. Minimize the given K-map

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1		
$\bar{A}B$		1	1	1
AB	1	1	1	
$A\bar{B}$			1	

Solution:

Here;

AB \ CD	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$		1		
$\overline{A}B$		1	1	1
AB	1	1	1	
$A\overline{B}$			1	

From K-map shown above,

$$Z = AB\overline{C} + \overline{A}BC + \overline{A}\,\overline{C}D + ACD$$
$$= A(B\overline{C} + CD) + \overline{A}(BC + \overline{C}D)$$

8. Use the K-map to simplify the expression $x = \overline{A}\,\overline{B}\,\overline{C} + \overline{B}C + \overline{A}B$

Solution:

Here;

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

K-map for the above expression is;

C \ AB	$\overline{A}\,\overline{B}$	$\overline{A}B$	AB	$A\overline{B}$
$\overline{C}$	1	1		
C	1	1		1

From K-map, simplified expression is;

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

9. Simplify $F(a, b, c) = \overline{a}b + \overline{b}c + \overline{c}a$ using K-map.

Solution:

Here;

$$F(a, b, c) = \overline{a}b + \overline{b}c + \overline{c}a$$
$$= \overline{a}b(c + \overline{c}) + \overline{b}c(a + \overline{a}) + \overline{c}a(b + \overline{b})$$
$$= \overline{a}bc + \overline{a}b\overline{c} + a\overline{b}c + \overline{a}\overline{b}c + ab\overline{c} + a\overline{b}\overline{c}$$

c \ ab	$\overline{a}\overline{b}$	$\overline{a}b$	ab	$a\overline{b}$
$\overline{c}$		1	1	1
c	1	1		1

From K-map;

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

10. Minimize $f = \sum(1, 2, 5, 7, 9, 11, 12, 14, 15)$

Solution:

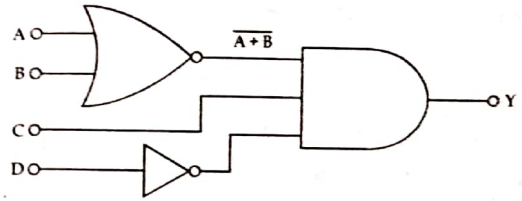
K-map for the expression given is shown below;

AB \ CD	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	0	1	3	1 (2)
$\overline{A}B$	4	1 (5)	1 (7)	6
AB	1 (12)	0 (13)	1 (15)	1 (14)
$A\overline{B}$	8	1 (9)	1 (11)	10

From K-map;

$$F(A, B, C, D) = AB\overline{D} + A\overline{B}D + \overline{A}\,\overline{C}D + \overline{A}\,\overline{B}C\overline{D} + BCD$$

11. Determine output expression for the circuit shown below and simplify it using De-Morgan's theorem.



Solution:

Here;

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

Using De-Morgan's theorem; we have,

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

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

12. Demonstrate by means of truth table the validity of following identities.

i) De-Morgan's theorem for 3 variables: $(XYZ)' = X' + Y' + Z'$

ii) Second distributive law: $X + Y \cdot Z = (X + Y)(X + Z)$

Solution:

Here;

i) De-Morgan's theorem: $(XYZ)' = X' + Y' + Z'$

Truth table

XYZ	X + Y + Z	(X + Y + Z)'	X'	Y'	Z'	X'Y'Z'
000	0	1	1	1	1	1
001	1	0	1	1	0	0
010	1	0	1	0	1	0
011	1	0	1	0	0	0
100	1	0	0	1	1	0
101	1	0	0	1	0	0
110	1	0	0	0	1	0
111	1	0	0	0	0	0

ii) $X + YZ = (X + Y)(X + Z)$

XYZ	X + YZ	X + Y	X + Z	(X + Y)(X + Z)
000	0	0	0	0
001	0	0	1	0
010	0	1	0	0
011	1	1	1	1
100	1	1	1	1
101	1	1	1	1
110	1	1	1	1
111	1	1	1	1

Show that the dual of the exclusive-OR is equal to its complement.

Solution:

Here; $X \oplus Y = X'Y + XY'$

and, $(X \oplus Y)' = (X + Y')(X' + Y)$

∴ Dual of $X'Y + XY' = (X' + Y)(X + Y') = (X \oplus Y)'$

14. Simplify the following Boolean expression using K-map.

i) $F(x, y, z) = \sum(3, 5, 6, 7)$

ii) $F(A, B, C) = \sum(0, 2, 3, 4, 6)$

iii) $F(X, Y, Z) = \sum(0, 1, 5, 7)$

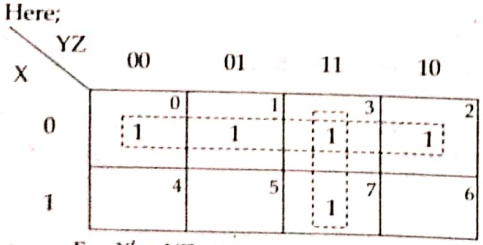
iv) $F(Z, B, C, D) = \sum(4, 6, 7, 15)$

v) $F(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

vi) $F(W, X, Y, Z) = \sum(2, 3, 10, 11, 12, 13, 14, 15)$

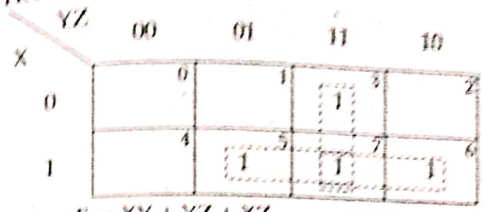
Solution:

i) Here;



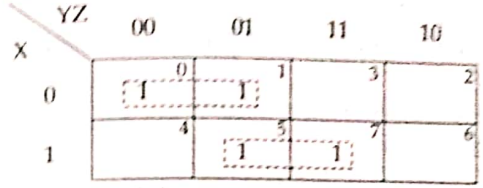
∴ $F = X' + YZ$

ii) Here;



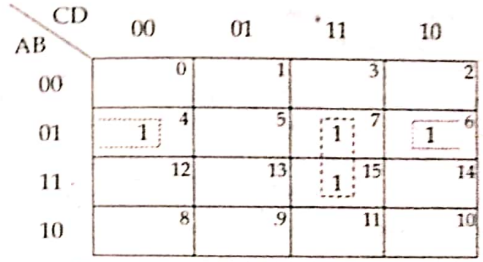
∴ $F = XY + YZ + XZ$

iii) Here;



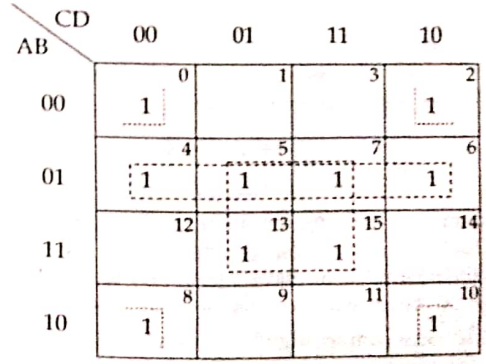
∴ $F = X'Y' + XZ$

iv) Here;



∴ $F = BCD + A'BD'$

v) Here;



∴ $F = BD + A'B + B'D'$

or, $F = BD + BD' + A'D$

vi) Here;

WX \ YZ	00	01	11	10
00			1	1
01				
11	1	1	1	1
10				

∴ $F = W'X'Y + WX$

15. Simplify the expression $F = \overline{A}BC + A\overline{B}\,\overline{C} + ABC + AB\overline{C}$

Solution:

The Karnaugh map for this function is shown below;

	$\overline{B}\,\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$			1	
A	1		1	1

ird column, the adjacent squares are grouped together to produce ified term BC. The other two 1s are placed at the first column and n of the same second row. Note that these 1s or minterms can be to produce a reduced term. Here, the B variable is changing its om un-complemented to complemented. After combining these nterms, we get the reduced term $A\overline{C}$. This can be confirmed by plying the Boolean algebra, $A\overline{B}\,\overline{C} + AB\overline{C} = A\overline{C}(B + \overline{B}) = A\overline{C}$

Hence, the simplified expression can be written as;

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

16. Simplify the expression $F = \overline{A}\,\overline{B}C + \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}C + ABC$

Solution:

The K-map is shown below;

	$\overline{B}\,\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$		1	1	1
A		1	1	

The four adjacent squares comprising the minterms $\overline{A}B\overline{C}$, $\overline{A}BC$, $A\overline{B}C$ and ABC can be combined. Here, it may observed that two of the variables A and B are changing their forms from uncomplemented to complemented. Hence, these variables can be removed to form the reduced expression to C. Again, two adjacent squares comprising the minterms $\overline{A}BC$ and $\overline{A}B\overline{C}$ can be combined to produce the reduced term $\overline{A}B$. So, the final simplified expression of the given function is $F = C + \overline{A}B$.

17. Simplify expression

$F(A, B, C, D) = m_1 + m_5 + m_{10} + m_{11} + m_{12} + m_{13} + m_{15}$

Solution:

The K-map for the given expression is shown below;

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$		1		
$\overline{A}B$		1		
AB	1	1	1	
$A\overline{B}$			1	1

From the figure, it can be seen that four pairs can be formed. The simplified expression may be written as;

$F = \overline{A}\,\overline{C}D + AB\overline{C} + ACD + A\overline{B}C$

	$\overline{C}\,\overline{D}$	CD	$C\overline{D}$	$C\overline{D}$
$\overline{A}\,\overline{B}$		1		
$\overline{A}B$		1		
AB	1	1	1	
$A\overline{B}$			1	1

Note that the reduced expression is not a unique one because if pairs are formed in different ways, the simplified expression will be different. But both the expressions are logically correct. The simplified expression of the given function as per the K-map of the figure below is;

$F = \overline{A}\,\overline{C}D + AB\overline{C} + ABD + A\overline{B}C$

18. Simplify the expression $F(W, X, Y, Z) = m_7 + m_9 + m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15}$

Solution:

The K-map for the given expression is shown below;

	$\overline{Y}\,\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$\overline{W}\,\overline{X}$				
$\overline{W}X$			1	
WX	1	1	1	1
$W\overline{X}$		1	1	1

Three quads and one pair are formed. The simplified expression of the given function is;

$F = WX + WY + WZ + XYZ$

19. Simplify the expression

$F(W, X, Y, Z) = \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$

vi) Here;

WX \ YZ	00	01	11	10
00			1	1
01				
11	1	1	1	1
10				

$\therefore \quad F = W'X'Y + WX$

15. Simplify the expression $F = \overline{A}BC + A\overline{B}\,\overline{C} + ABC + AB\overline{C}$

Solution:

The Karnaugh map for this function is shown below;

	$\overline{B}\,\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$			1	
A	1		1	1

In the third column, the adjacent squares are grouped together to produce the simplified term BC. The other two 1s are placed at the first column and last column of the same second row. Note that these 1s or minterms can be combined to produce a reduced term. Here, the B variable is changing its form, from un-complemented to complemented. After combining these two minterms, we get the reduced term $A\overline{C}$. This can be confirmed by applying the Boolean algebra, $A\overline{B}\,\overline{C} + AB\overline{C} = A\overline{C}(B + \overline{B}) = A\overline{C}$

Hence, the simplified expression can be written as;

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

16. Simplify the expression $F = \overline{A}\,\overline{B}C + \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}C + ABC$

Solution:

The K-map is shown below;

	$\overline{B}\,\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$		1	1	1
A		1	1	

The four adjacent squares comprising the minterms $\overline{A}B\overline{C}$, $\overline{A}BC$, $A\overline{B}C$ and ABC can be combined. Here, it may observed that two of the variables A and B are changing their forms from uncomplemented to complemented. Hence, these variables can be removed to form the reduced expression to C.

Again, two adjacent squares comprising the minterms $\overline{A}BC$ and $\overline{A}B\overline{C}$ can be combined to produce the reduced term $\overline{A}B$. So, the final simplified expression of the given function is $F = C + \overline{A}B$.

17. Simplify expression

$F(A, B, C, D) = m_1 + m_5 + m_{10} + m_{11} + m_{12} + m_{13} + m_{15}$

Solution:

The K-map for the given expression is shown below;

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$		1		
$\overline{A}B$		1		
AB	1	1	1	
$A\overline{B}$			1	1

From the figure, it can be seen that four pairs can be formed. The simplified expression may be written as;

$$F = \overline{A}\,\overline{C}D + AB\overline{C} + ACD + A\overline{B}C$$

	$\overline{C}\,\overline{D}$	CD	$C\overline{D}$	$C\overline{D}$
$\overline{A}\,\overline{B}$		1		
$\overline{A}B$		1		
AB	1	1	1	
$A\overline{B}$			1	1

Note that the reduced expression is not a unique one because if pairs are formed in different ways, the simplified expression will be different. But both the expressions are logically correct. The simplified expression of the given function as per the K-map of the figure below is;

$$F = \overline{A}\,\overline{C}D + AB\overline{C} + ABD + A\overline{B}C$$

18. Simplify the expression $F(W, X, Y, Z) = m_7 + m_9 + m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15}$

Solution:

The K-map for the given expression is shown below;

	$\overline{Y}\,\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$\overline{W}\,\overline{X}$				
$\overline{W}X$			1	
WX	1	1	1	1
$W\overline{X}$		1	1	1

Three quads and one pair are formed. The simplified expression of the given function is;

$$F = WX + WY + WZ + XYZ$$

19. Simplify the expression

$F(W, X, Y, Z) = \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$

Solution:

The K-map for the given expression is shown below;

	$\overline{Y}\,\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$\overline{W}\,\overline{X}$	1	1		1
$\overline{W}X$	1	1		1
WX	1	1		1
$W\overline{X}$	1	1		

One octet and two quads are formed. The simplified expression is;

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

20. Obtain the minimal sum of the products for the function.

$F(A, B, C, D) = \sum(1, 3, 7, 11, 15) + \phi(0, 2, 5)$

Solution:

The K-map for the given expression is shown below;

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1	1	X
$\overline{A}B$			1	
AB			1	
$A\overline{B}$			1	

The minterms m_0 and m_2 *i.e.*, $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ and $\overline{A}\,\overline{B}C\overline{D}$ are the do not care terms which have been assumed as 1s while making a quad. The simplified SOP expression of above function can be written as;

$F = \overline{A}\,\overline{B} + CD$

21. Using the K-map method, obtain the minimal sum of the products expression for the function $F(A, B, C, D) = \sum(0, 2, 3, 6, 7) + d(8, 10, 11, 15)$.

Solution:

The K-map for the given expression is shown below;

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1		1	1
$\overline{A}B$			1	1
AB			X	
$A\overline{B}$	X		X	X

The simplified Boolean expression for the function is;

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

22. Draw the four input AND gate and its truth table.

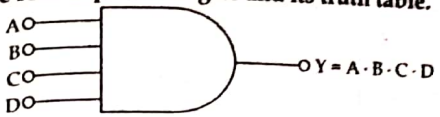


Figure: Four input AND gate symbol

Solution:

Truth table

A	B	C	D	$Y = A \cdot B \cdot C \cdot D$
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	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

23. Write the truth table for a four input exclusive –OR gate.

Solution:

Truth table

A	B	C	D	$Y = A \oplus B \oplus C \oplus D$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

24. Show the logic arrangement for implementing.

i) a four input NAND gate using two input AND gates and NOT gates

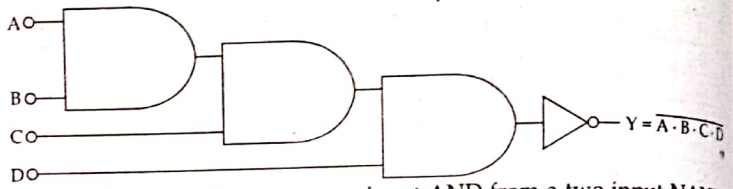
ii) a three input NAND gate using two input NAND gates.

iii) a NOT circuit using a two input NAND gate

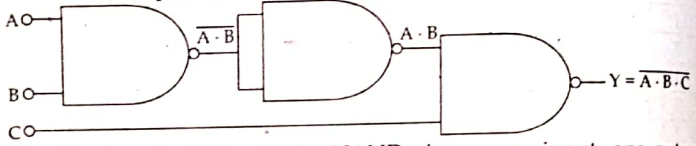
iv) a NOT circuit using a two input NOR gate

Solution:

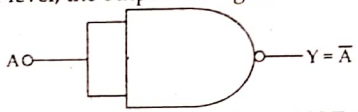
i) The first step is to get a four input AND gate using two input AND gates. The output thus obtained is then complemented using a NOT circuit.



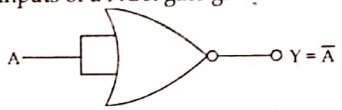
ii) The first step is to get a two input AND from a two input NAND. The output of the two input AND gate and the third input then feed the inputs of another two input NAND to get desired output.



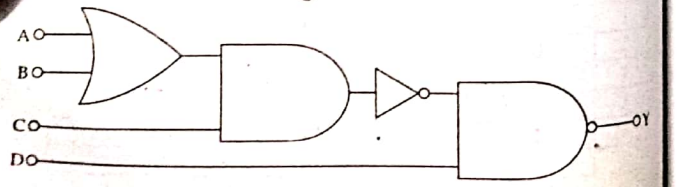
iii) Shorting the inputs of the NAND gives a one input, one output NOT circuit. This is because when all inputs to a NAND are at logic 0 level, the output is a logic 1 and when all inputs to a NAND are at logic 1 level, the output is a logic 0.



iv) Shorting the inputs of a NOR gate gives a NOT circuit.



Draw the truth table of the logic circuit shown below.



Solution:

Truth table

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

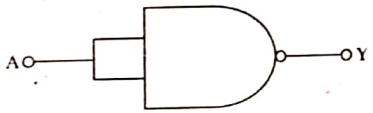
26. Draw logic implementation of an inverter using,

i) two input NAND **ii) two input NOR**

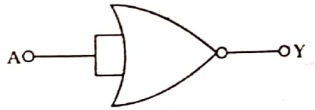
iii) two input EX-OR **iv) two input EX-NOR**

Solution:

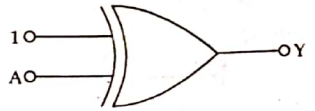
i)



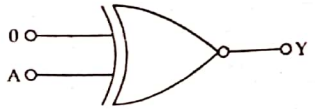
ii)



iii)



iv)



27. Prove $Z \cdot X + Z \cdot \overline{X} \cdot Y = Z \cdot X + Z \cdot Y$ using truth table.

Solution:

Proof

X	Y	Z	ZX	ZY	$Z\overline{X}$	$Z\overline{X}Y$	$ZX + Z\overline{X}Y$	$ZX + ZY$
0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0
0	1	1	0	1	1	1	1	1
1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	1	1
1	1	0	0	0	0	0	0	0
1	1	1	1	1	0	0	1	1

From truth table;

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

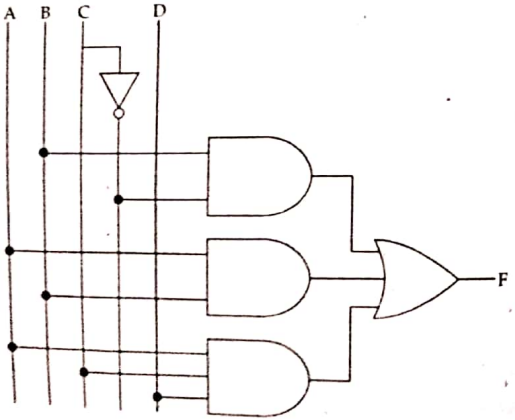
28. Draw the logic diagram corresponding to the following Boolean expressions without simplifying them.

i) $B\overline{C} + AB + ACD$

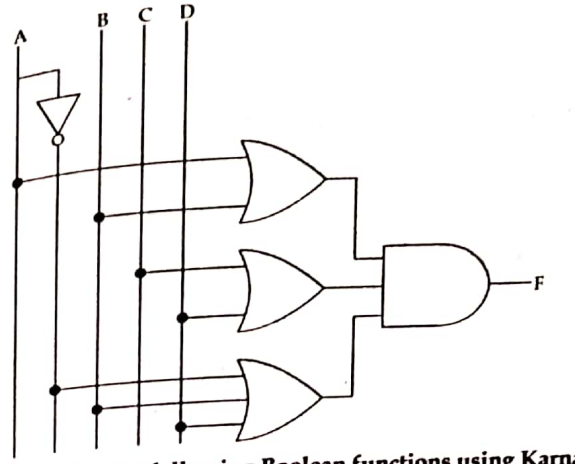
ii) $(A + B)(C + D)(\overline{A} + B + D)$

Solution:

i)



ii)



19. Simplify the following Boolean functions using Karnaugh maps.

i) $F(x, y, z) = \sum(1, 2, 3, 6, 7)$

ii) $xy + \overline{x}\,\overline{y}\,\overline{z} + \overline{x}yz$

iii) $\overline{x}\,\overline{y} + yz + \overline{x}y\overline{z}$

iv) $\overline{x}y + y\overline{z} + \overline{y}\,\overline{z}$

v) $xyz + \overline{x}y\overline{z} + xy\overline{z}$

vi) $F(w, x, y, z) = \sum(2, 3, 12, 13, 14, 15)$

vii) $F(A, B, C, D) = \sum(3, 7, 11, 13, 14, 15)$

viii) $F(W, X, Y, Z) = \sum(1, 4, 5, 6, 12, 14, 15)$

ix) $F(A, B, C, D) = \sum(1, 5, 9, 10, 11, 14, 15)$

x) $F(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

xi) $F(A, B, C, D) = \sum(0, 2, 5, 7, 8, 10, 12, 13)$

xii) $F(W, X, Y, Z) = \sum(1, 3, 4, 5, 9, 10, 11, 12, 13)$

xiii) $F(A, B, C, D) = \sum(1, 2, 3, 5, 7, 9, 10, 11, 13, 15)$

xiv) $F(W, X, Y, Z) = \sum(2, 3, 7, 10, 11, 12, 13, 14, 15)$

xv) $F(A, B, C, D) = \sum(1, 5, 8, 9, 10, 11, 12, 13, 15)$

Solution:

i) Here;

x \ yz	00	01	11	10
0	1		1	1
1	1			1

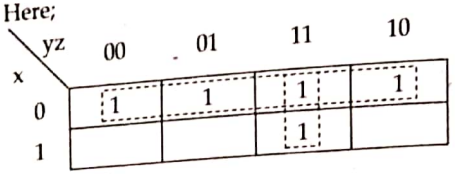
$\therefore F = \overline{z} + \overline{x}y$

ii) Here;

x \ yz	00	01	11	10
0	1			1
1			1	1

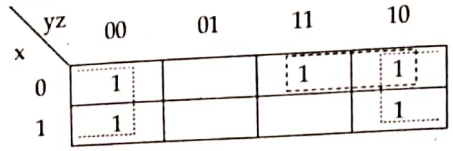
$\therefore F = xy + \overline{x}\,\overline{z}$

iii) Here;



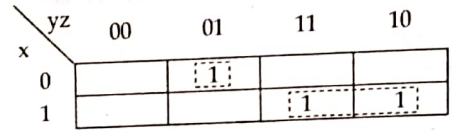
$\therefore$ $F = \bar{x} + yz$

iv) Here;



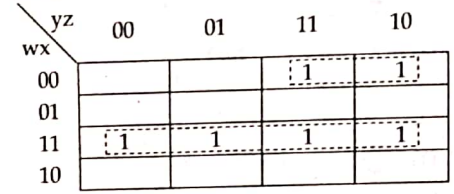
$\therefore$ $F = \bar{x} + y + \bar{z}$

v) Here;



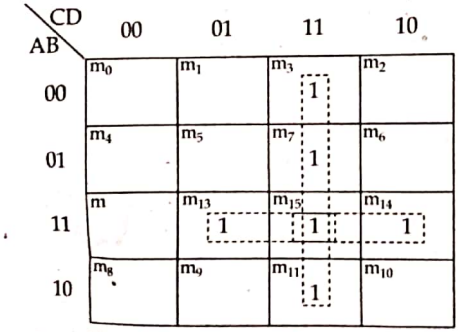
$\therefore$ $F = \bar{x}\,\bar{y}z + xy$

vi) Here;



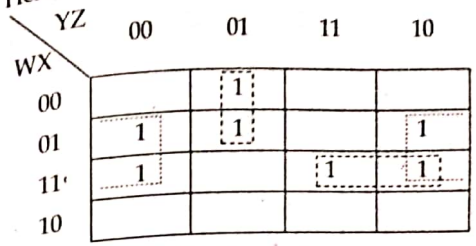
$\therefore$ $F = \bar{w}\,\bar{x}y + wx$

vii) Here;



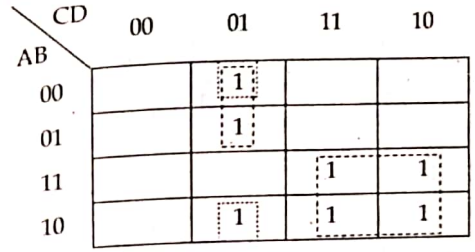
$\therefore$ $F = ABD + CD + ABC$

viii) Here;



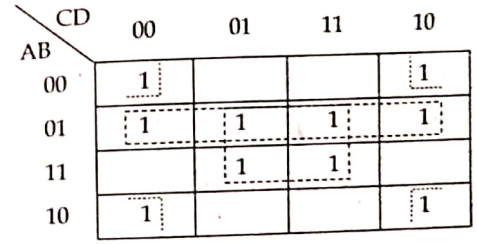
$\therefore$ $F = X\bar{Z} + \bar{W}\,\bar{Y}Z + WXY$

ix) Here;



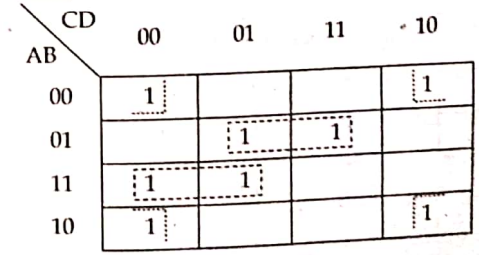
$\therefore$ $F = \bar{A}C + \bar{A}\,\bar{C}D + \bar{B}\,\bar{C}D$

x) Here;



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

xi) Here;



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

xii) Here;

WX \ YZ	00	01	11	10
00		1	1	
01	1	1		
11	1	1		
10		1	1	1

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

xiii) Here;

AB \ CD	00	01	11	10
00		1	1	1
01		1	1	
11		1	1	
10		1	1	1

$\therefore \quad F = Z + \overline{X}Y$

iv) Here;

WX \ YZ	00	01	11	10
00			1	1
01			1	
11	1	1	1	1
10			1	1

$\therefore \quad F = WX + \overline{X}Y + YZ$

xv) Here;

AB \ CD	00	01	11	10
00		1		
01		1		
11	1	1	1	
10	1	1	1	1

$\therefore \quad F = A\overline{C} + AD + CD + A\overline{B}C$

30. Simplify the following Boolean function using K-map and implement if using logic gates.

i) $F(A, B, C, D) = \sum(0, 1, 2, 3, 8, 9, 10, 11, 12, 13, 14, 15)$

ii) $F(A, B, C, D) = \sum(1, 6, 7, 12, 13, 14, 15)$

iii) $F(A, B, C) = \sum(5, 6, 7)$

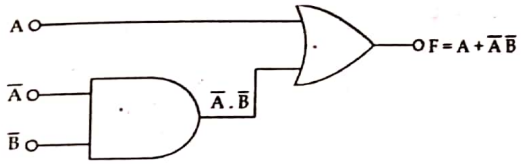
Solution:

i) K-map for the above function is;

AB \ CD	00	01	11	10
00	1	1	1	1
01				
11	1	1	1	1
10	1	1	1	1

$\therefore \quad F = A + \overline{A}\,\overline{B}$

Logic diagram is;

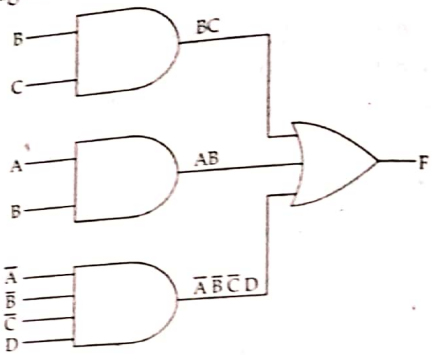


ii) K-map for the above function is;

AB \ CD	00	01	11	10
00		1		
01			1	1
11	1	1	1	1
10				

$\therefore \quad F = BC + AB + \overline{A}\,\overline{B}\,\overline{C}D$

Logic diagram is;

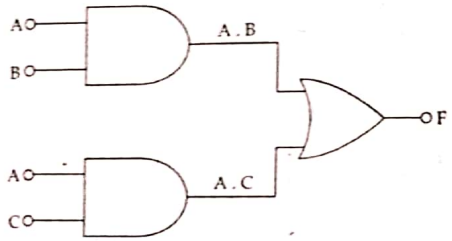


ii) K-map for the above function is;

A \ BC	00	01	11	10
0				
1		1	1	1

∴ $F = AB + AC$

Logic diagram is;



CHAPTER 5
COMBINATION LOGIC

*****

*****

5.1 INTRODUCTION

The digital system consists of two types of circuits, namely,

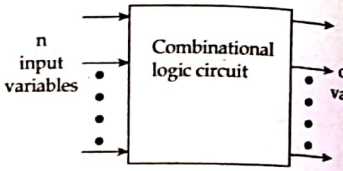
i) Combinational circuits

ii) Sequential circuits

A combinational circuit consists of logic gates where outputs are at any instant and are determined only by the present combination of inputs without regard to previous inputs or previous state of outputs. A combinational circuit performs a specific information-processing operation assigned logically by a set of Boolean function. Sequential circuits contain logic gates as well as memory cells. Their outputs depend

on the present inputs and also on the states of memory elements. Since the outputs of sequential circuits depend not only on the present inputs but also on past inputs, the circuit behaviour must be specified by a time sequence of inputs and memory states.

A combinational circuit consists of input variables, logic gates and output variables. The logic gates accept signals from inputs and output signals are generated according to the logic circuits employed in it. Binary information from the given data transforms to desired output data in this process. Both input and output are obviously the binary signals *i.e.*, both the input and output signals are of two possible states logic 1 and logic 0. Figure shows a block diagram of a combination logic circuit.



There are n numbers of input variables coming from an electric source and m number of output signals go to an external destination. The source and/or destination may consists of memory elements or sequential logic circuit or shift registers located either in the vicinity of the combinational logic circuit or in a remote external location. But the external circuit does not interfere in the behaviour of the combination circuit. For n number of input variables to a combinational circuit, 2^n possible combinations of binary input states are possible. For each possible combination, there is one and only one possible output combination. A combinational logic circuit can be described by m Boolean function and each output can be expressed in terms of n input variables.

5.1 DESIGN PROCEDURE

Any combinational circuit can be designed by the following steps of design procedure;

a) The problem is stated.
b) Identify the input variables and output functions.
c) The input and output variable are assigned letter symbols.
d) The truth table is prepared that completely defines the relationship between the input variables and output functions.
e) The simplified Boolean expression is obtained by any methods of minimization-algebraic method, K-map or tabulation method.
A logic diagram is realized from the simplified expression using logic gates.

ery important that the design problem or the verbal specifications be preted correctly to prepare the truth table. Sometimes, the designer st use his intuition and experience to arrive at the correct interpretation word specification are very seldom exact and complete. Any wrong interpretation results in incorrect truth table and combinational circuit.

A practical design approach should consider constraints like,

i) Minimum number of logic gates
ii) Minimum number of outputs
iii) Minimum propagation time of the signal through a circuit
iv) Minimum number of interconnections
v) Limitations of the driving capabilities of each logic gate

Since the importance of each constraint is dictated by the particular application, it is difficult to make all these criteria satisfied simultaneously and also difficult to make a general statement on the process of achieving an acceptable simplification. However, in most cases, first the simplified Boolean expression at standard form is derive and then other constraints are taken care of as far as possible for a particular application.

5.2 ADDERS AND SUBTRACTORS

5.2.1 Adders

A. Half Adders

A combinational circuit that performs the addition of two bits is called half adders. Such a circuit thus has two inputs that represent the two bits to be added and two outputs with one producing the sum output and the other producing the CARRY. Let the input variables augend and addend be designated as A and B and output functions be designated as S for sum and C for carry. The truth table for the functions is below.

Input variables		Output variables	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From the truth table, it can be seen that the output S and C functions are similar to Exclusive-OR and AND functions respectively.

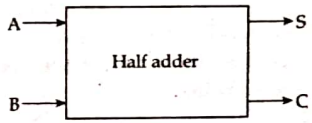
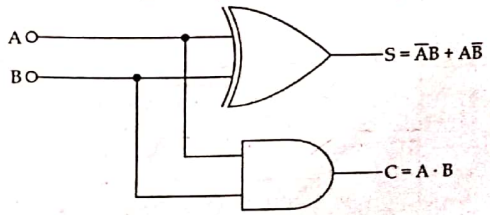


Figure: Half adder

The Boolean expressions are,

$S = \overline{A}B + A\overline{B}$ and $C = AB$

Figure below shows the logic diagram to implement the half adder circuit.



B. Full Adders

A full adder circuit is an arithmetic circuit block that can be used to add three bits to produce a sum and CARRY output. Let us designate the input variable aunged as A, addend as B and previous carry as X, and output sum as S and carry as C. As there are three input variables, eight different input combinations are possible. The truth table is,

Input Variable			Outputs	
X	A	B	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

To derive the simplified Boolean expression from the truth table, the K-map method is adopted as,

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{X}$		1		1
X	1		1	

Figure: (a) K-map for function S

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{X}$			1	1
X		1	1	1

Figure: (b) K-map for function C

The simplified Boolean expression of the outputs are,

$$S = \bar{X}\bar{A}B + \bar{X}A\bar{B} + X\bar{A}\bar{B} + XAB$$

and, $C = AB + BX + AX$

The logic diagram for the above function is shown below;

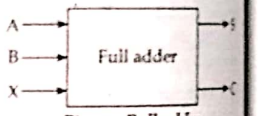


Figure: Full adder

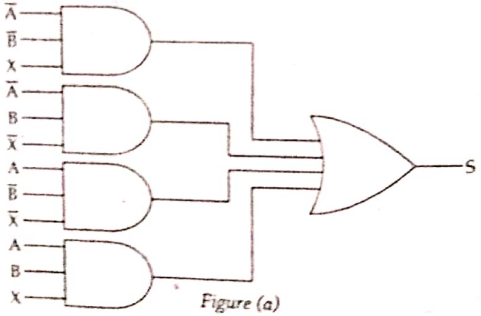


Figure (a)

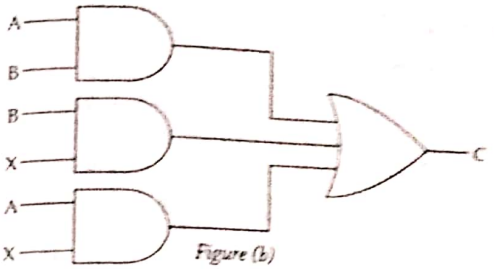


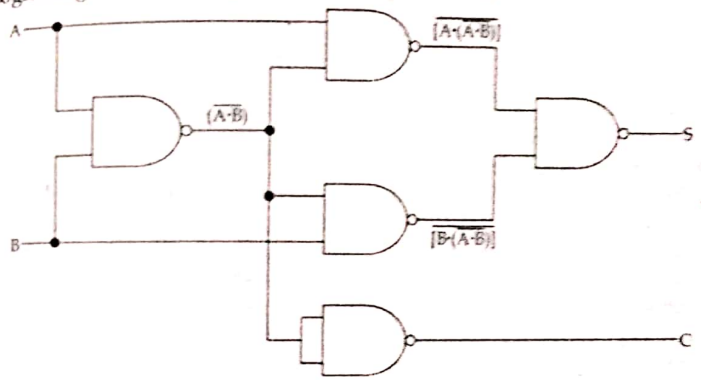
Figure (b)

Figure: Logic circuit diagram of full adder

$$S = \bar{X}\bar{A}B + \bar{X}A\bar{B} + X\bar{A}\bar{B} + XAB$$
$$= \bar{X}(\bar{A}B + A\bar{B}) + X(\bar{A}\bar{B} + AB) = \bar{X}(A \oplus B) + X(\overline{A \oplus B}) = X \oplus A \oplus B$$

and, $C = AB + BX + AX = AB + X(A + B)$
$= AB + X(AB + A\bar{B} + AB + \bar{A}B)$
$= AB + X(AB + A\bar{B} + \bar{A}B)$
$= AB + XAB + X(A\bar{B} + \bar{A}B)$
$= AB + X(A \oplus B)$

Logic diagram according to modified expression is,



This contains a reduced number of gates as well as type of gates. Also if compared with a half adder circuit, the full adder circuit can be formed with two half adders and one OR gate.

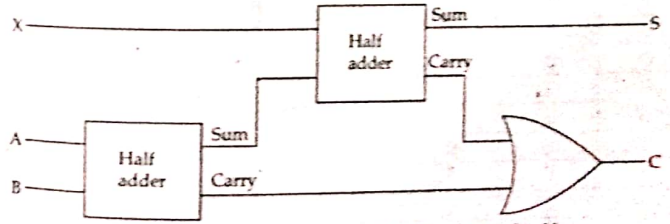


Figure: Logic implementation of a full adder with half adders

5.2.2 Subtractor

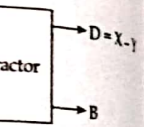
A. Half-subtractor

A half subtractor is a combinational circuit that can be used to subtract one binary digit from another to produce difference output and a Borrow output. The Borrow output here specifies whether a '1' has been borrowed to perform the subtraction. The truth table of a half subtractor is shown below;

Input variables		Output variables	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

By considering the minterms of the truth table, Boolean expressions of the outputs D and B can be written as,

$D = \overline{X}Y + X\overline{Y}$ and $B = \overline{X}Y$



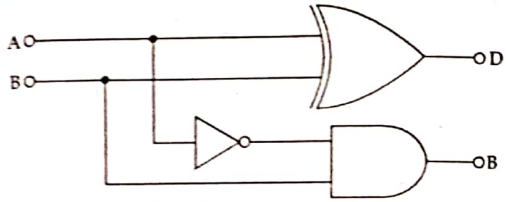


Figure: Logic diagram to realize half subtractor circuit

B. Full Subtractor

A full subtractor, performs subtraction operation on two bits, a minuend and a subtrahend and also takes into consideration whether a '1' has already been borrowed by the previous adjacent lower minuend bit or not. As a result, there are three bits to be handled at the input of a full subtractor, namely the two bits to be subtracted and a borrow bit. let us designate the input variables minuend as X, subtrahend as Y and previous borrow as Z and outputs difference as D and borrow as B. Eight different input combinations are possible for three input variables. The truth table is shown below;

Input variables			Outputs	
X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Karnaugh maps are prepared to derive simplified Boolean expressions of D and B as,

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$\overline{X}$		1		1
X	1		1	

Figure: Map for function D

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$\overline{X}$		1	1	1
X			1	

The simplified Boolean expression of the outputs are,

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

and, $C = \overline{X}Z + \overline{X}Y + YZ$

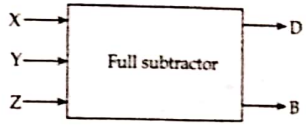


Figure below contains two input and three input NAND gates and three input and four input OR gates.

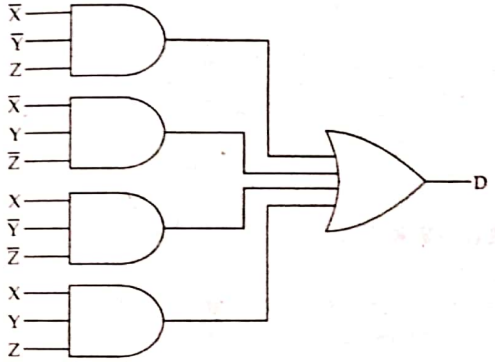
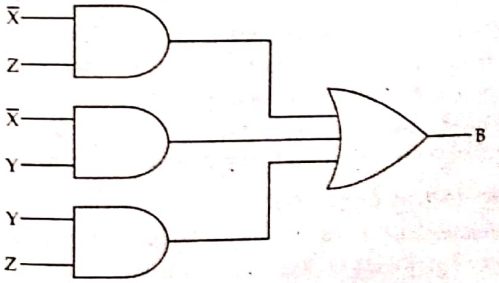


Figure (a)

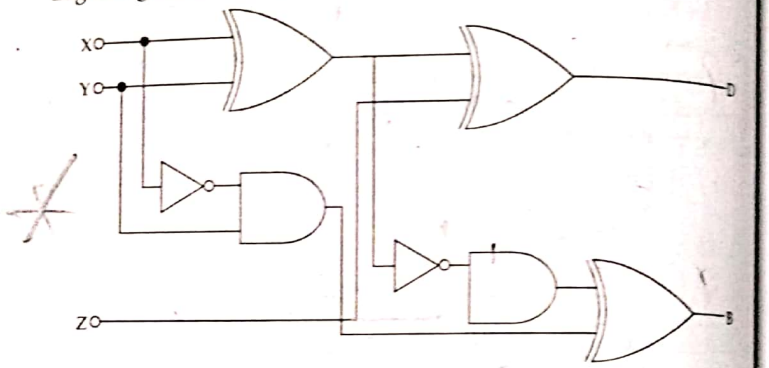


The Boolean expressions of D and B are modified as follows,

$$D = \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ$$
$$= \bar{X}(\bar{Y}Z + Y\bar{Z}) + X(Y\bar{Z} + YZ)$$
$$= \bar{X}(Y \oplus Z) + X(\overline{Y \oplus Z})$$
$$= X \oplus Y \oplus Z$$

and, $B = \bar{X}Z + \bar{X}Y + YZ$
$$= \bar{X}Y + Z(\bar{X} + Y)$$
$$= \bar{X}Y + Z(\bar{X}Y + \bar{X}\bar{Y} + XY + \bar{X}Y)$$
$$= \bar{X}Y + Z(\bar{X}Y + \bar{X}\bar{Y} + XY)$$
$$= \bar{X}Y + \bar{X}YZ + Z(\overline{\bar{X}Y} + XY)$$
$$= \bar{X}Y + Z(\overline{\bar{X} \oplus Y})$$

Logic diagram according to the modified expression is,



5.3 CODE CONVERSION

The availability of a large variety of codes for the same discrete elements of information results in the use of different codes by different digital systems. It is sometimes necessary to use the output of one system as the input to another. A conversion circuit must be inserted between the two systems if each uses different codes for the same information. Thus, a code converter is a circuit that makes the two systems compatible even though each uses a different binary code.

To convert from binary code A to binary code B, the input lines must supply the bit combination of elements as specified by code A and the output lines must generate the corresponding bit combination of code B. A combinational circuit performs this transformation by means of logic gates.

5.3.1 Binary-to-gray Converter

The bit combinations of 4-bit binary code and its equivalent bit combinations of gray code are listed in table below;

Binary				Gray			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

The four bits of binary numbers are designated A, B, C and D and gray code bits are designated as W, X, Y and Z. The K-map are shown as,

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$				
AB	1	1	1	1
$A\bar{B}$	1	1	1	1

Figure: K-map for W

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$	1	1	1	1
AB				
$A\bar{B}$	1	1	1	1

Figure: K-map for X

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$			1	1
$\bar{A}B$	1	1		
AB	1	1		
$A\bar{B}$			1	1

Figure: K-map for Y

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$A\overline{B}$		1		1
$\overline{A}\overline{B}$		1		1
$\overline{A}B$		1		1
AB		1		1
$A\overline{B}$				

Figure: K-map for Z

From the K-map, we get,

$W = A$

$Y = B\overline{C} + \overline{B}C = B \oplus C$

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

$Z = \overline{C}D + C\overline{D} = C \oplus D$

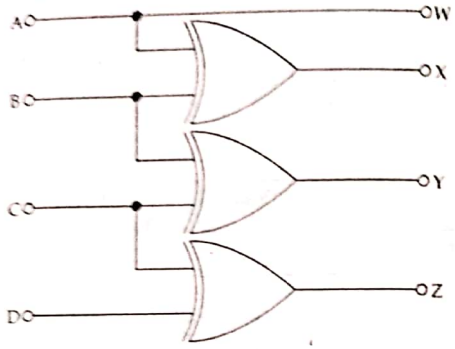


Figure: Circuit diagram

5.3.2 Gray to Binary Converter

Using the same conversion table, the K-maps are formed as,

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$W\overline{X}$				
$\overline{W}\overline{X}$				
WX	1	1	1	1
$W\overline{X}$	1	1	1	1

Figure: K-map for A

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$W\overline{X}$				
$\overline{W}\overline{X}$	1	1	1	1
WX				
$W\overline{X}$	1	1	1	1

Figure: K-map for B

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$W\overline{X}$			1	1
$\overline{W}\overline{X}$	1	1		
WX			1	1
$W\overline{X}$	1	1		

Figure: K-map for C

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$W\overline{X}$		(1)		(1)
$\overline{W}\overline{X}$	(1)		(1)	
WX		(1)		(1)
$W\overline{X}$	(1)		(1)	

Figure: K-map for D

The Boolean expression are,

$A = W$

$B = \overline{W}X + W\overline{X} = W \oplus X$

$C = \overline{W}\overline{X}Y + \overline{W}X\overline{Y} + WXY + W\overline{X}\overline{Y}$

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

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

$= W \oplus X \oplus Y$ or $B \oplus Y$

$D = \overline{W}\overline{X}\overline{Y}Z + \overline{W}\overline{X}Y\overline{Z} + \overline{W}X\overline{Y}\overline{Z} + \overline{W}XYZ + WX\overline{Y}Z + WXY\overline{Z} + W\overline{Y}\overline{Z} + W\overline{X}YZ$

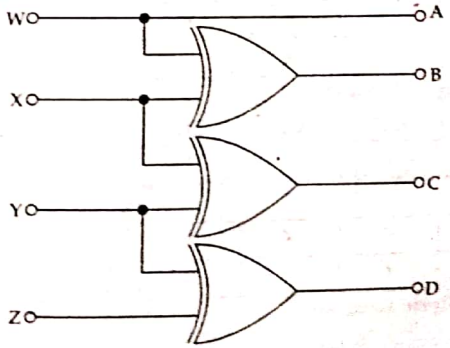
$= \overline{W}\overline{X}(\overline{Y}Z + Y\overline{Z}) + \overline{W}X(\overline{Y}\overline{Z} + YZ) + WX(\overline{Y}Z + Y\overline{Z}) + W\overline{X}(\overline{Y}\overline{Z} + YZ)$

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

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

$= W \oplus X \oplus Y \oplus Z$ or $C \oplus Z$

The circuit diagram of a gray to binary code converter is,



5.3.3 BCD (8-4-2-1) to Excess-3 Code Converter

BCD is an abbreviation for binary-coded-decimal. BCD is a numeric code in which each digit of a decimal number is represented by a separate group of bits. It is called 8-4-2-1 code because the weights associated with 4 bits are 8-4-2-1 from left to right. This means that, bit 3 has weight 8, bit 2 has weight 4, bit 1 has weight 2 and bit 0 has weight 1.

The bit combinations of both BCD and Excess-3 codes represent decimal digits from 0 to 9. Therefore, each of the code systems contains four bits and so there must be four input variables and four output variables. Figure below provides the list of the bits combinations or truth table and equivalent decimal number. The symbols A, B, C and D are designated as the bits of the Excess-3 code system. It may be noted that though 16 combinations are possible from four bits, both code systems use only 10 combinations. The rest of the bit combinations never occur and are treated as don't care conditions.

Decimal equivalent	BCD code				Excess-3 code			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

For the BCD to excess-3 converter, A, B, C, D are input variable and W, X, Y, Z are output variables. The K-map are,

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$		1	1	1
AB	X	X	X	X
$A\bar{B}$	1	1	X	X

Figure: K-map for W

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1	1	1
$\bar{A}B$	1			
AB	X	X	X	X
$A\bar{B}$		1	X	X

Figure: K-map for X

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1		1	
$\bar{A}B$	1		1	
AB	1	X	X	X
$A\bar{B}$	1		1	X

Figure: K-map for Y

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1			1
$\bar{A}B$	1			1
AB	X	X	X	X
$A\bar{B}$	1		X	X

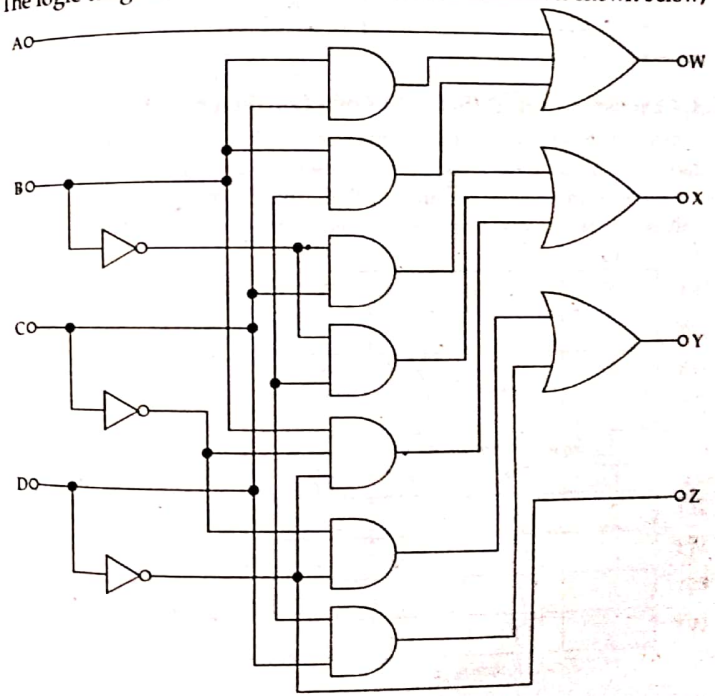
Figure: K-map for Z

The simplified Boolean expression of W, X, Y and Z are,

$W = A + BC + BD$ $\quad X = \bar{B}C + \bar{B}D + B\bar{C}\bar{D}$

$Y = CD + \bar{C}\bar{D}$ $\quad Z = \bar{D}$

The logic diagram of a BCD to Excess-3 converter circuit is shown below,



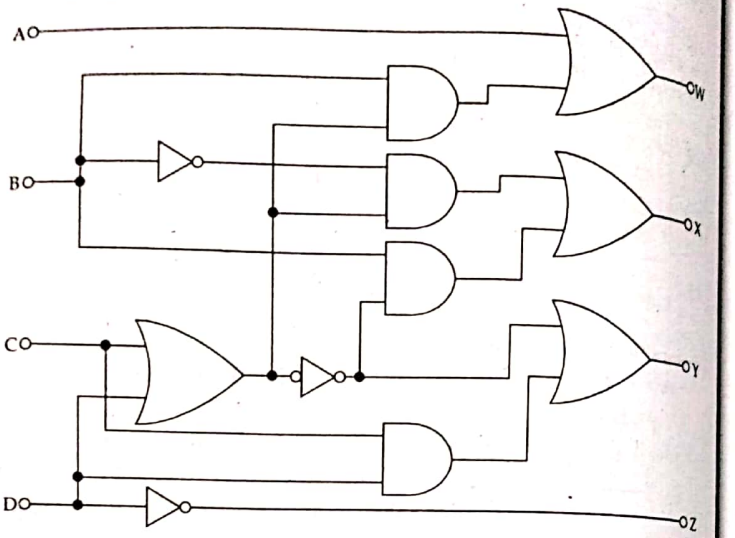
The alternative circuit diagram can be made after the following modification on the Boolean expression.

$W = A + BC + BD + A + B(C + D)$

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

$Y = C\overline{D} + \overline{C}\,\overline{D} = C\overline{D} + \overline{(C + D)}$

$Z = \overline{D}$



5.3.4 Excess-3 to BCD (8-4-2-1) Code Converter

To construct the excess-3 to BCD converter circuit, a similar truth table as in case of BCD to excess-3 code may be used. In this case W, X, Y, Z are considered as input variable and A, B, C and D are termed as output variables. The required K-map are prepared as per figures as shown below;

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$\overline{W}\overline{X}$	X	X		X
$\overline{W}X$				
WX	1	X	X	X
$W\overline{X}$			1	

Figure: K-map for A

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$\overline{W}\overline{X}$	X	X		X
$\overline{W}X$			1	
WX		X	X	X
$W\overline{X}$	1	1		1

Figure: K-map for B

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$\overline{W}\overline{X}$	X	X		X
$\overline{W}X$		1		1
WX	1	X	X	X
$W\overline{X}$		1		1

Figure: K-map for C

	$\overline{Y}\overline{Z}$	$\overline{Y}Z$	YZ	$Y\overline{Z}$
$\overline{W}\overline{X}$	X	X		X
$\overline{W}X$	1			1
WX	1	X	X	X
$W\overline{X}$	1			X

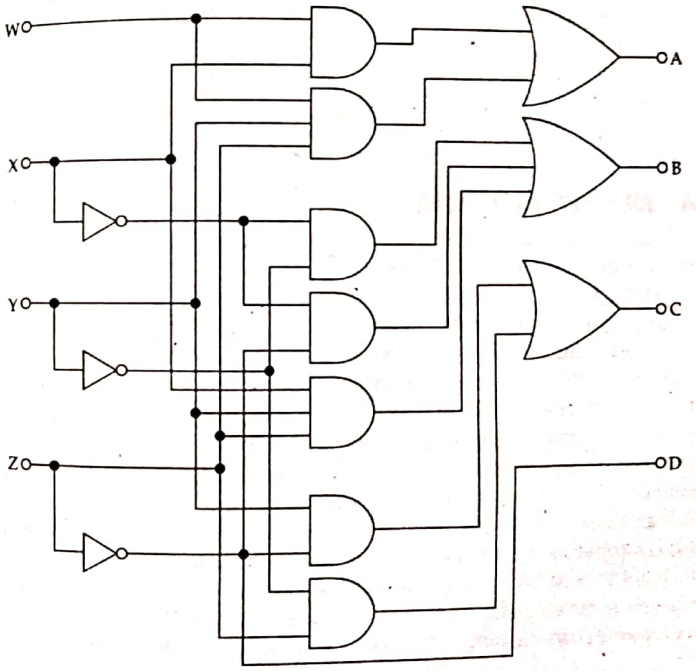
Figure: K-map for D

The Boolean expression of the outputs are

$A = WX + WYZ$ $\qquad$ $B = \overline{X}\overline{Y} + \overline{X}\overline{Z} + XYZ$

$C = \overline{Y}Z + Y\overline{Z}$ $\qquad$ $D = \overline{Z}$

The logic diagram of an Excess-3- to BCD converter is shown below,



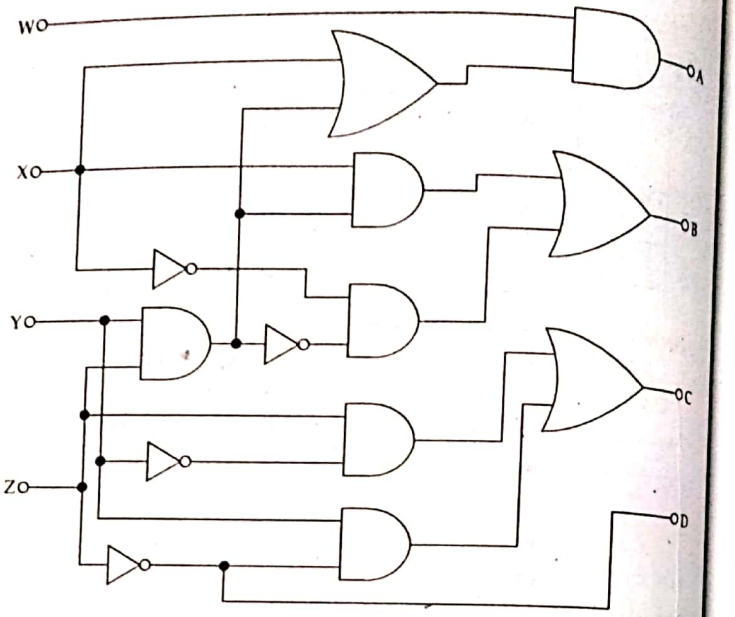
The alternative circuit diagram of above figure can be made after the following modification on the above Boolean expressions.

$A = WX + WYZ = W(X + YZ)$

$B = \bar{X}\bar{Y} + \bar{X}\bar{Z} + XYZ = \bar{X}(\bar{Y} + \bar{Z}) + XYZ = \bar{X}(\bar{Y}Z) + XYZ$

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

$D = \bar{Z}$



5.4 ANALYSIS PROCEDURE

The design of combinational circuit starts from the verbal specifications of a required function and culminates with a set of output Boolean function or logic diagram. The analysis of a combinational circuit is somewhat the reverse process. It starts with a given logic diagram and culminates with a set of Boolean functions, a truth table or a verbal explanation of the circuit operation. If the logic diagram to be analyzed is accompanied by a function name or an explanation of what it is assumed to accomplish then the analysis problem reduces to a verification of the stated function.

The first step in the analysis is to make sure that the given circuit is combinational and not sequential. The diagram of combinational circuit has logic gates with no feedback paths or memory elements. A feedback path is a connection from the output of one gate to the input of a second gate that forms part of the input to the first gate.

Once the logic diagram is verified as a combinational circuit, one can proceed to obtain the output Boolean functions and or the truth table. If

the circuit is accompanied by a verbal explanation of its function, then the Boolean functions or the truth table is sufficient for verification. If the function of the circuit is under investigation, then it is necessary to interpret the operation of the circuit from the derived truth table. The success of such investigation is enchanced if one has previous experience and familiarity with a wide variety of digital circuits. To obtain the output Boolean functions from a logic diagram, proceed as follows;

i) Label with arbitrary symbols all gate outputs that are a function of the input variables. Obtain the Boolean functions for each gate.

ii) Label with other arbitrary symbols those gates that are a function of input variable and or previously labeled gates. Find the Boolean functions for these gates.

iii) Repeat the process outlined in step until the outputs of the circuits are obtained.

iv) By repeated substitution of previously defined functions obtain the output Boolean functions in terms of input variables only.

The derivation of truth table for the circuit is straight forward process once the output Boolean functions are known. To obtain the truth table directly from the logic diagram without going through the derivations of the Boolean functions, proceed as follows;

i) Determine the number of input variables to the circuit. For n inputs, form the 2^n possible input combinations of 1's and 0's by listing the binary number 0 to $2^n - 1$.

ii) Label the output of selected gates with arbitary symbols.

iii) Obtain the truth table for the outputs of those gates that are a function of the input variables only.

iv) Proceed to obtain the truth table for the outputs of those gates that are a function of previously defined values until the columns for all outputs are determined.

5.5 MULTILEVEL NAND AND NOR CIRCUITS

Combinational circuits are more frequently constructed with NAND or NOR gates rather than AND and OR gates. NAND and NOR gates are more common from the hardware point of view because they are readily available in the integrated- circuit form. Because of the promience of NAND and NOR gates in the design of combinational circuits, it is important to be able to recognize the relationships that exists between circuits constructed with AND-OR gates and their equivalent NAND or NOR diagrams.

A. Multilevel NAND Circuits

One possible way to implement a Boolean function with NAND gates is to obtain the simplified Boolean function in terms of Boolean operators and then convert the function to NAND logic. The conversion of an algebraic expression from AND, OR and complement to NAND can be

done by simple circuit manipulation techniques that change AND-OR diagrams to NAND diagrams.

To facilitate the conversion to NAND logic, it is convenient to use the two alternate graphic symbols. The AND-invert graphic symbol consists of an AND graphic symbol followed by a small circle. The invert OR graphic symbol consists of an OR graphic symbol that is preceeded by small circles in all the inputs. Either symbol can be used to represent a NAND gate.

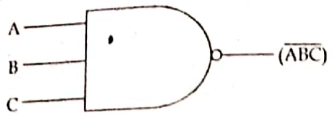


Figure: (a) AND–Invert

Figure: (b) Invert–OR

Figure: Two graphic symbols for a NAND gate

To obtain a multilevel NAND diagram form a Boolean expression, proceed as follows,

i) From the given Boolean expression, draw the logic diagram with AND, OR and inverter gates. Assume that both normal and complement inputs are available.

ii) Convert all AND gates to NAND gates with AND-invert graphic symbols.

iii) Convert all OR gates to NAND gates with invert-OR graphic symbols.

iv) Check all small circles in the diagram. For every small circle that is not compensated by another small circle along the same line, insert an inverter (one input NAND gate) or complement the input variable.

As a example, consider the Boolean function F = (CD + E) (A + $\overline{B}$).

The AND-OR implementation is shown below with three levels of gating.

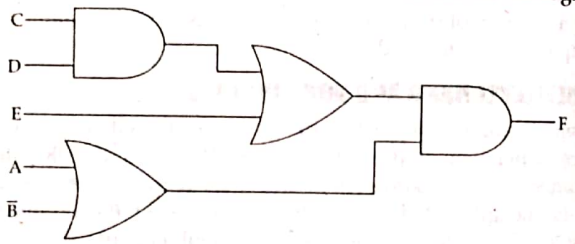


Figure: AND–OR diagram

The conversion into a NAND circuit is presented in below figure. The three additional small circles associated with inputs E, A and $\overline{B}$ causes these three literals to be complemented to $\overline{E}$, $\overline{A}$ and B. The small circle in the last NAND gate complements the output, so we need to insert an inverter gate at the output, so we need to insert an inverter gate at the output in order to complement the signal again and obtain the original value. In general, the number of NAND gates required to implement a

Boolean expression is equal to the number of AND-OR gates except for an occasional inverter. This is true provided both the normal and complement inputs are available, because the conversion forces certain input variables to be complemented.

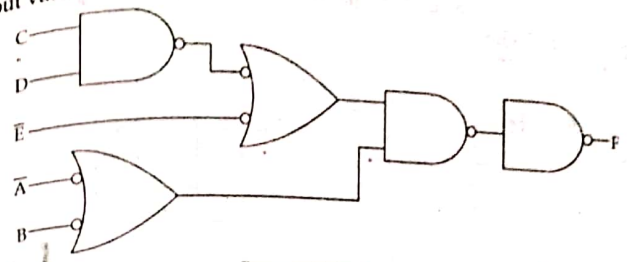


Figure: NAND diagram

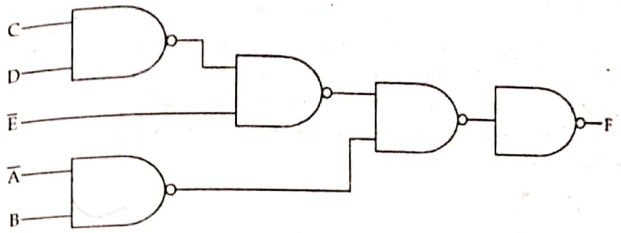


Figure: Alternate NAND diagram

B. Multilevel NOR Circuits

The NOR function is the dual of the NAND. For this reason, all procedures and rules for NOR logic form a dual of the corresponding procedures and rules developed for NAND logic. The two graphic symbols for the NOR gate are shown below. The OR-invert symbol defines the NOR operation as an OR followed by a complement. The invert-AND symbol complements each input and then performs an AND operation. The two symbols designate the same NOR operation and are logically identical because of Demorgans theorem.

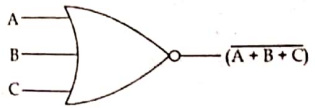


Figure: (a) OR–Invert

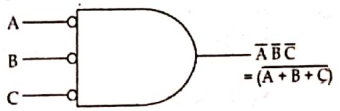


Figure: (b) Invert–AND

The procedure for implementing a Boolean function with NOR gates is similar to the procedure outlined in the previous section for NAND gates.

i) Draw the AND-OR logic diagram from the given algebraic expression. Assume that both the normal and complement inputs are available.

ii) Convert all OR gates to NOR gates with OR-invert graphic symbols.

iii) Convert all AND gates to NOR gates with invert-AND graphic symbols.

iv) Any small circle that is not compensated by another small circle along the same needs an inverter or the compensation of the input variable.

The procedure is illustrated below for the Boolean function, F = (AB + E)(C + D). The AND-OR implementation of the expression is shown in the logic diagram below;

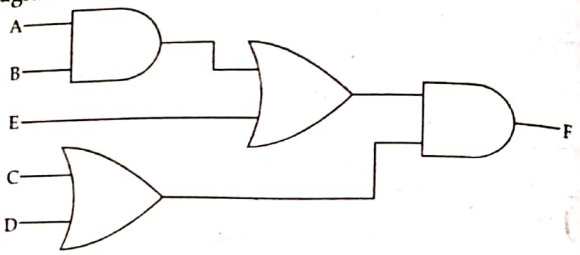


Figure: AND–OR diagram

For each OR gate, we substitute a NOR gate with the OR-invert graphic symbol. For each AND gate, we substitute a NOR gate with the invert-AND graphic symbol. The two small circles associated with inputs A and B cause these two variables to be complemented to $\overline{A}$ and $\overline{B}$ respectively. The NOR diagram is shown below;

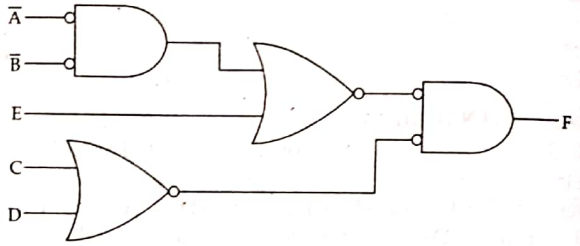


Figure: NOR diagram

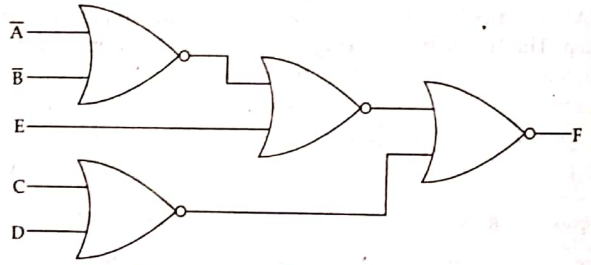


Figure: Alternate NOR diagram

In general, the number of NOR gates required to implement a Boolean function will be the same as the number of gates in the AND-OR diagram. This is true provided both the normal and complement inputs are available because the conversion may require that certain input variables be complemented.

5.6 PARITY GENERATION AND CHECKING

Parity is a very useful tool in information processing in digital computers to indicate any presence of error in bit information. External noise and loss of signal strength cause loss of data bit information while transporting data from one device to other device, located inside the computer or externally. To indicate any occurrence of error, an extra bit is included with the message according to the total number of 1s in a set of data, which is called parity. If the extra bit is considered 0 if the total number of 1s is even and 1 for odd quantities if 1s in a set of data, then it is called even parity, on the other hand, if the extra bit is 1 for even quantities of 1s and 0 for an odd number of 1s, then it is called odd parity.

5.6.1 Parity Generator

A parity generator is a combination logic system to generate the parity bit at the transmitting side. A table in figure illustrates even parity as well as odd parity for a message consisting of four bits.

Four bit message, $D_3\ D_2\ D_1\ D_0$	Even parity (P_e)	Odd parity (P_0)
0000	0	1
0001	1	0
0010	1	0
0011	0	1
0100	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1
1010	0	1
1011	1	0
1100	0	1
1101	1	0
1110	1	0
1111	0	1

If the message bit combination is designated as $D_3D_2D_1D_0$ and P_e and P_0 are the even and odd parity respectively, then it is obvious from the table that the Boolean expressions of even parity and odd parity are,

$$P_e = D_3 \oplus D_2 \oplus D_1 \oplus D_0, \qquad P_0 = \overline{(D_3 \oplus D_2 \oplus D_1 \oplus D_0)}$$

These can be confirmed by Karnaugh maps also. The logic diagram are shown below;

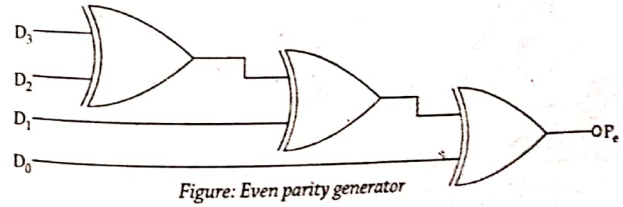


Figure: Even parity generator

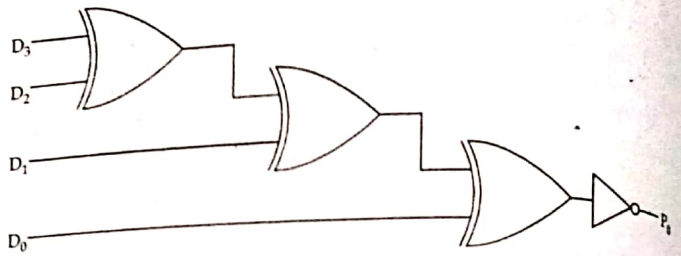


Figure: Odd parity generator

The above illustration is given for a message with four bits of information. However the logic diagrams can be expanded with more XOR gates for any number of bits.

5.6.2 Parity Checker

The message bits with the parity bit are transmitted to their destination where they are applied to a parity checker circuit. The circuit that checks the parity at the receiver side is called the parity checker. The parity checker circuit produces a check bit and is very similar to the parity generator circuit. If the check bit is 1, then it is assumed that the received data is incorrect. The check bit will be 0 if the received data is correct.

4-bit message $D_3D_2D_1D_0$	Even parity (P_e)	Even parity checker (C_e)
0000	0	0
0001	1	0
0010	1	0
0011	0	0
0100	1	0
0101	0	0
0110	0	0
0111	1	0
1000	1	0
1001	0	0
1010	0	0
1011	1	0
1100	0	0
1101	1	0
1110	1	0
1111	0	0

Figure: Even parity checker

4-bit message $D_3D_2D_1D_0$	Even parity (P_0)	Even parity checker (C_0)
0000	1	0
0001	0	0
0010	0	0
0011	1	0
0100	0	0
0101	1	0
0110	1	0
0111	0	0
1000	0	0
1001	1	0
1010	1	0
1011	0	0
1100	1	0
1101	0	0
1110	0	0
1111	1	0

The check bit is 0 for all the bit combinations of correct data. For incorrect data, the parity check bit will be another logic value. Parity checker circuits are the same as parity generator circuits.

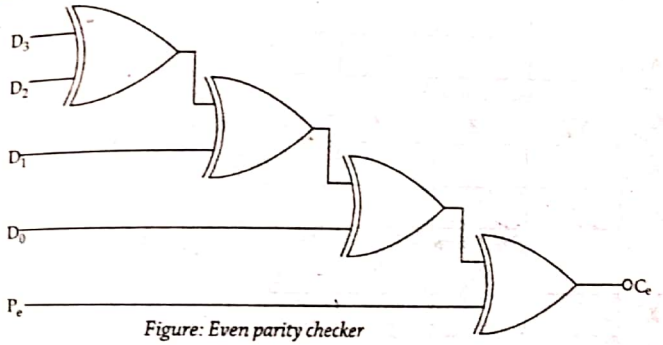


Figure: Even parity checker

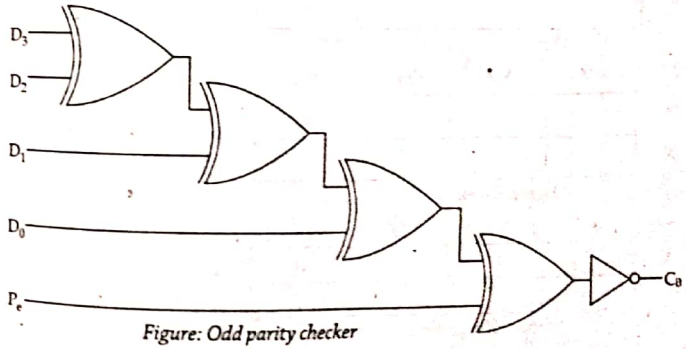


Figure: Odd parity checker

Example 1

Find the squares of 3-bit numbers.

Solution:

With three bits a maximum of eight combinations are possible with decimal equivalents of 0 to 7. By squaring of the decimal numbers the maximum decimal number produced is 49, which can be formed with six bits. A truth table is prepared as,

Input variables				Output variables						
Decimal equivalent	X	Y	Z	Decimal equivalent	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	1	0	4	0	0	0	1	0	0
3	0	1	1	9	0	0	1	0	0	1
4	1	0	0	16	0	1	0	0	0	0
5	1	0	1	25	0	1	1	0	0	1
6	1	1	0	36	1	0	0	1	0	0
7	1	1	1	49	1	1	0	0	0	1

K-maps for each of the output variables are,

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{X}$				
X			1	1

Figure: K-map for A

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{X}$				
X	1	1	1	

Figure: K-map for B

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{X}$			1	
X		1		

Figure: K-map for C

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{X}$				1
X				1

Figure: K-map for D

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{X}$	0	0	0	0
X	0	0	0	0

Figure: K-map for E

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{X}$		1	1	
X		1	1	

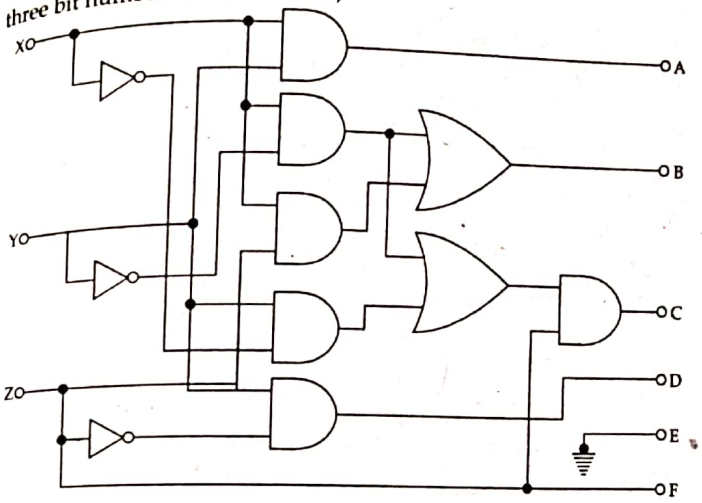
Figure: K-map for F

The Boolean expression of the output variables are,

$A = XY$ $\quad B = X\bar{Y} + XZ$ $\quad C = \bar{X}YZ + X\bar{Y}Z = (\bar{X}Y + X\bar{Y})Z$

$D = Y\bar{Z}$ $\quad E = 0$ $\quad F = Z$

The circuit diagram of the combinational network to obtain squares of three bit numbers is shown below;



Example 2

Design a combinational circuit for converting 2421 code to BCD code.

Solution:

Both the 2421 code and BCD code are 4 bit codes and represent the decimal equivalents 0 to 9. The truth table to design the converter circuit is,

Decimal equivalent	Input variable 2421 code				Output variable BCD code			
	W	X	Y	Z	A	B	C	D
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	1	0	1	1	0	1	0	1
6	1	1	0	0	0	1	1	0
7	1	1	0	1	0	1	1	1
8	1	1	1	0	1	0	0	0
9	1	1	1	1	1	0	0	1

K-maps to obtain the simplified expressions of the output functions are shown below;

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$				
$\bar{W}X$		X	X	X
WX			1	1
$W\bar{X}$	X	X		X

Figure: K-map for A

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$				
$\bar{W}X$	1	X	X	X
WX	1	1		
$W\bar{X}$	X	X	1	X

Figure: K-map for B

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$			1	1
$\bar{W}X$		X	X	X
WX	1	1		
$W\bar{X}$	X	X		X

Figure: K-map for C

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$		1	1	
$\bar{W}X$		X	X	X
WX		1	1	
$W\bar{X}$	X	X	1	X

Figure: K-map for D

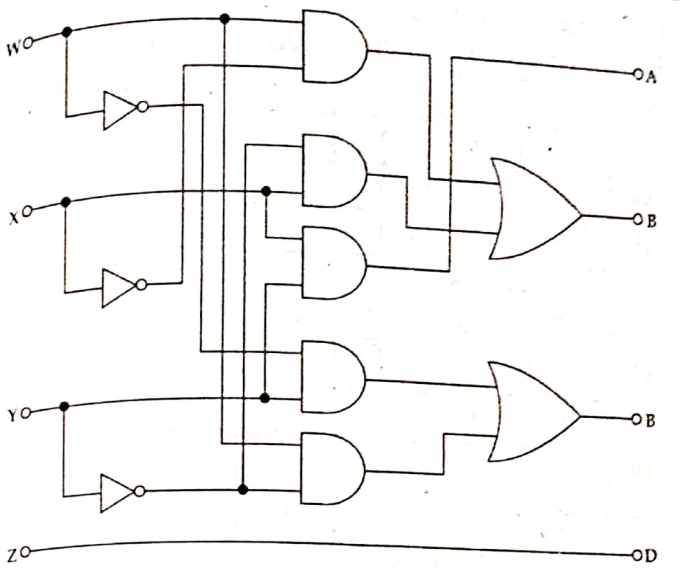
The Boolean expression for the output functions are,

$A = XY$

$B = X\bar{Y} + W\bar{X}$

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

$D = Z$



Example 3

Design a combinational circuit that converts 2421 code to 84-2-1 code and also the converter circuit for 84-2-1 code to 2421 code.

Solution:

Both the codes represent binary codes for decimal digits 0 to 9. Let, A, B, C and D be represented as 2421 code variables and W, X, Y and Z be variables for 84-2-1. The truth table is shown below;

2421 code				84-2-1 code			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	1	0	0	1	1	0
0	0	1	1	0	1	0	1
0	1	0	0	0	1	0	0
1	0	1	1	1	0	1	1
1	1	0	0	1	0	1	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	0	0
1	1	1	1	1	1	1	1

The K-map for W, X, Y, Z are,

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$		X	X	X
AB	1	1	1	1
$A\bar{B}$	X	X	1	X

Figure: K-map for W

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	1
$\bar{A}B$	1	X	X	X
AB			1	
$A\bar{B}$	X	X		X

Figure: K-map for X

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1		1
$\bar{A}B$		X	X	X
AB	1		1	
$A\bar{B}$	X	X	1	X

Figure: K-map for Y

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1	1	
$\bar{A}B$		X	X	X
AB		1	1	
$A\bar{B}$	X	X	1	X

Figure: K-map for Z

The Boolean expressions for a 2421 to 84-2-1 code converter are,

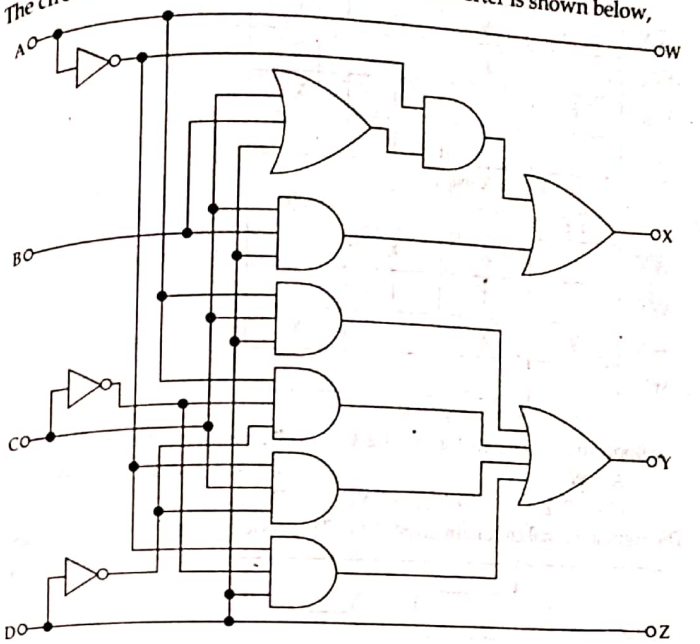
$W = A$

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

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

$Z = D$

The circuit diagram for 2421 to 84-2-1 code converter is shown below,



To design the 84-2-1 code to 2421 code converter, the K-map for the variable A, B, C and D in respect W, X, Y and Z shown below.

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$		X	X	X
$\bar{W}X$				
WX	X	X	1	X
$W\bar{X}$	1	1	1	1

Figure: K-map for A

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$		X	X	X
$\bar{W}X$	1			
WX	X	X	1	X
$W\bar{X}$	1	1		1

Figure: K-map for B

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$		X	X	X
$\bar{W}X$		1		1
WX	X	X	1	X
$W\bar{X}$	1		1	

Figure: K-map for C

	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$		X	X	X
$\bar{W}X$		1	1	
WX	X	X	1	X
$W\bar{X}$		1	1	

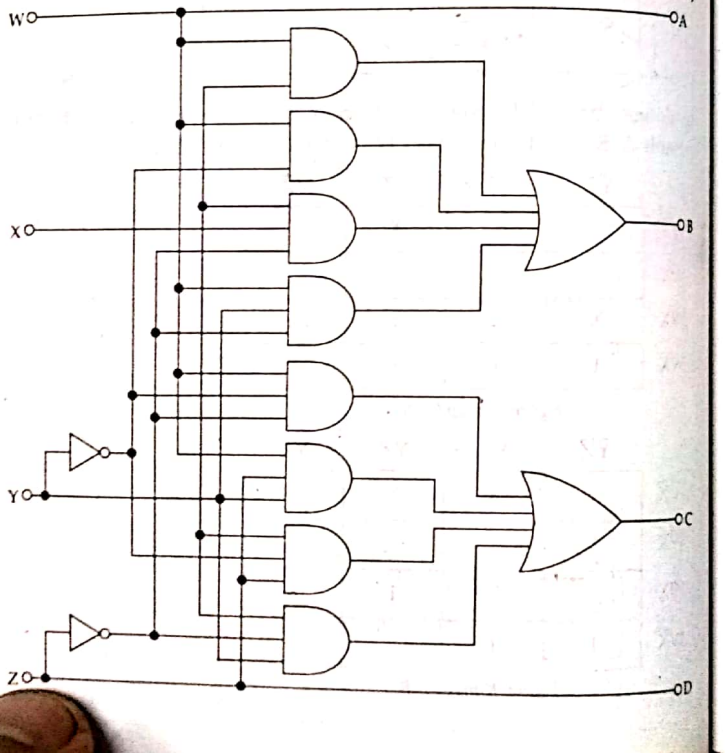
Figure: K-map for D

The Boolean expressions for an 84-2-1 to 2421 code converter are,

$A = W$ $\qquad B = WX + W\bar{Y} + X\bar{Y}\bar{Z} + WY\bar{Z}$

$C = W\bar{Y}\bar{Z} + WYZ + X\bar{Y}Z + XY\bar{Z}$ $\qquad D = Z$

The combinational circuit for an 84-2-1 to 2421 code converter is shown below;



EXAMINATION QUESTION SOLUTIONS

1. **Design a combinational circuit with three inputs and one output. The output is equal to logic 1 when the binary value of the input is less than 3. The output is logic 0 otherwise.** **[2011/S]**

Solution:

Let A, B, C be three input variables and D be he output.

Input			Output
A	B	C	D
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

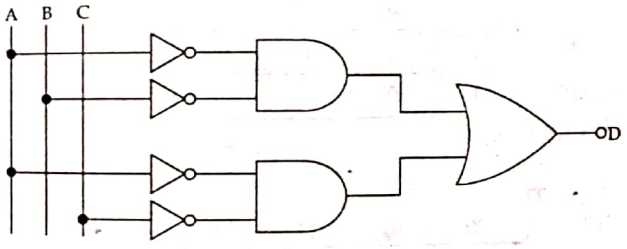
K-map

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1	1	0	1
A	0	0	0	0

For K-map

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

Combinational circuit is,



2. **Design a binary to gray code converter circuit using the minimum number of logic gates. Use the four variable input and four variable output case.** **[2011/S, 2019/F]**

Solution: See the topic 5.3.1.

3. **Design a code converter circuit which converts 84-2-1 code to binary.** **[2011/F, 2012/S]**

Solution:

Let A, B, C, D and W, X, Y, Z represent 84-2-1 code and binary code respectively.

Decimal equivalent	84-2-1				Binary			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1
2	0	1	1	0	0	0	1	0
3	0	1	0	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	1	0	1	1	0	1	0	1
6	1	0	1	0	0	1	1	0
7	1	0	0	1	0	1	1	1
8	1	0	0	0	1	0	0	0
9	1	1	1	1	1	0	0	1
10	1	1	1	0	1	0	0	1
11	1	1	0	1	1	0	1	1
12	1	1	0	0	1	1	0	0

The remaining of A, B, C, D for which 84-2-1 code is not defined are treated as don't care conditions. The don't care conditions are 0001, 0010 and 0011.

K-map;

For W,

AB \ CD	00	01	11	10
00	0	X	X	X
01	0	0	0	0
11	1	1	1	1
10	1	0	0	0

$\therefore \quad W = AB + A\bar{C}\bar{D}$

For X,

AB \ CD	00	01	11	10
00	0	X	X	X
01	1	0	0	0
11	0	0	0	0
10	0	1	1	1

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

For Y,

AB \ CD	00	01	11	10
00	0	X	X	X
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$\therefore \quad Y = \bar{C}D$

For Z,

AB \ CD	00	01	11	10
00	0	X	X	X
01	0	1	1	0
11	0	1	1	1
10	0	1	1	0

$\therefore \quad Z = D$

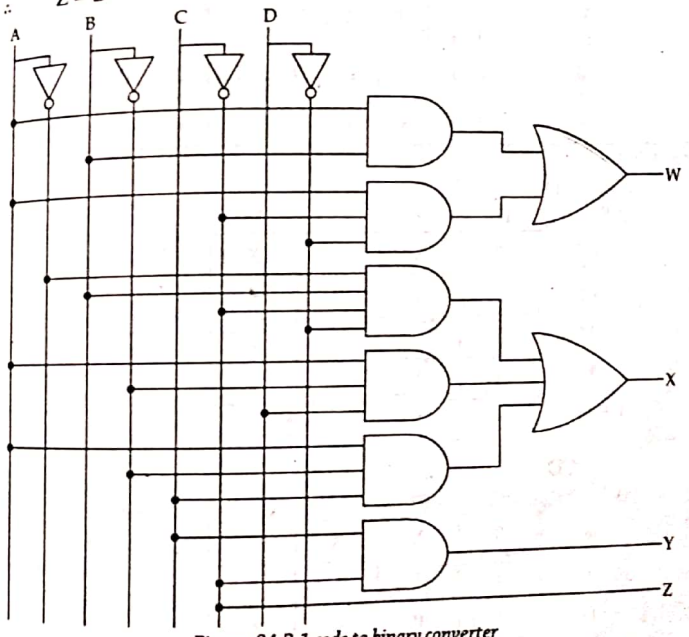


Figure: 84-2-1 code to binary converter

4. **Design a circuit diagram to generate and check the odd parity.** **[2012/S]**

Solution:

See the topic 5.6.1 and 5.6.2 [odd parity generator and checker only].

5. Design a code conversion circuit which converts BCD (8-4-2-1) to Excess-3 code. [2012/F, 2014/F, 2019/S]

Solution: See the topic 5.3.3.

6. Design a code converter circuit to convert 84-2-1 code to gray codes. [2012/F]

Solution:

Let A, B, C, D and W, X, Y, Z denote 84-1-2 and gray code respectively.

Decimal equivalent	84-2-1				Binary (gray)			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	1	0	0
2	0	1	1	0	0	1	0	1
3	0	1	0	1	0	1	1	1
4	0	1	0	0	0	1	1	0
5	1	0	1	1	1	1	1	0
6	1	0	1	0	1	1	1	1
7	1	0	0	1	1	1	0	1
8	1	0	0	0	1	1	0	0
9	1	1	1	1	1	0	0	0
10	1	1	1	0	1	0	0	1
11	1	1	0	1	1	0	1	1
12	1	1	0	0	1	0	1	0

For W;

AB \ CD	00	01	11	10
00	0	X	X	X
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

∴ W = A

For X,

AB \ CD	00	01	11	10
00	0	X	X	X
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

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

For Y,

AB \ CD	00	01	11	10
00	0	X	X	X
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

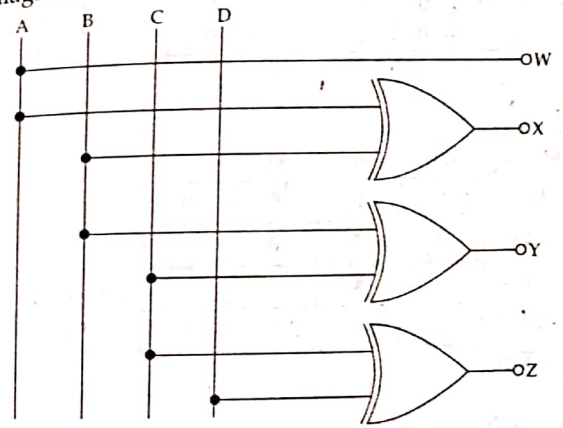
$Y = B\overline{C} + \overline{B}C = B \oplus C$

∴ For Z,

AB \ CD	00	01	11	10
00	0	X	X	X
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

∴ $Z = \overline{C}D + C\overline{D} = C \oplus D$

Circuit diagram;



7. Design even parity generator when a 3 bit message contains cycle code. [2013/S]

Solution:

A cycle code is a block code, where the circular shifts of each code word gives another word that belongs to the code. They are error correcting codes that have algebraic properties that are convenient for efficient error detection and correction.

8. **Design a combinational circuit which takes three input numbers and produces an output equal to square of the input.** [2013/F]

Solution: See the Example 1.

9. **Design a combinational circuit with four input lines that represent a decimal digit in BCD and four output lines that generate the 9's complement of the input digit.** [2013/F]

Soution:

Let, A, B, C, D and W, X, Y, Z denote BCD and its 9's complement respectively.

D.E	BCD Code				9's complement				D.E
	A	B	C	D	W	X	Y	Z	
0	0	0	0	0	1	0	0	1	9
1	0	0	0	1	1	0	0	0	8
2	0	0	1	0	0	1	1	1	7
3	0	0	1	1	0	1	1	0	6
4	0	1	0	0	0	1	0	1	5
5	0	1	0	1	0	1	0	0	4
6	0	1	1	0	0	0	1	1	3
7	0	1	1	1	0	0	1	0	2
8	1	0	0	0	0	0	0	1	1
9	1	0	0	1	0	0	0	0	0

K-map

For W,

AB \ CD	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

$\therefore \quad W = \bar{A}\bar{B}\bar{C}$

For X,

AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

$\therefore \quad X = B\bar{C} + \bar{B}C = B \oplus C$

For Y,

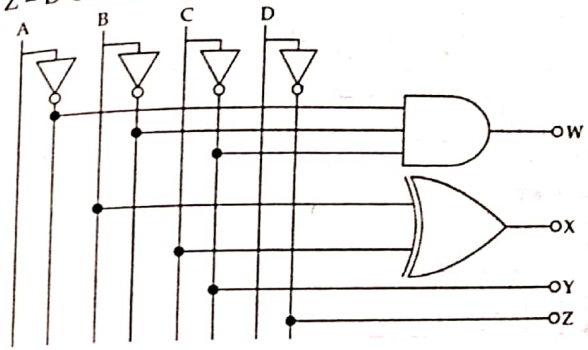
AB \ CD	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	X	X	X	X
10	0	0	X	X

$\therefore \quad Y = C$

For Z,

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$\therefore \quad Z = \bar{D}$ Circuit diagram;



10. **Design a combinational circuit that accepts a 3 bit number as input and generates the output binary equal to the 2's complement of input number.** [2014/S]

Solution: Let, A, B, C and X, Y, Z denotes input 3-bit number and its 2's complement respectively.

Truth Table:

Input			Output		
A	B	C	X	Y	Z
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	0	0	1

K-map

For X,

A \ BC	00	01	11	10
0	0	1	1	1
1	1	0	0	0

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

For Y,

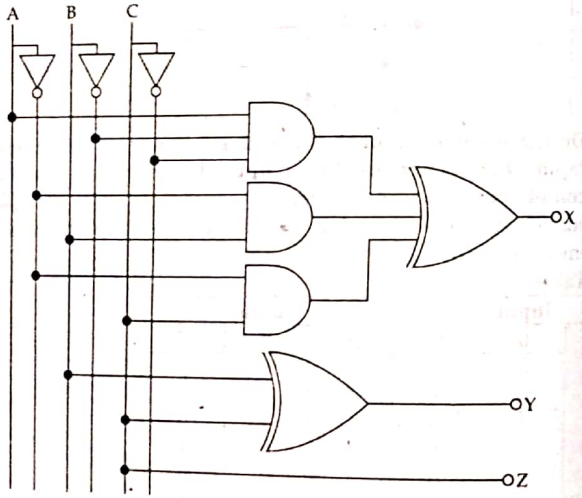
A \ BC	00	01	11	10
0	0	1	0	1
1	0	1	0	1

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

A \ BC	00	01	11	10
0	0	1	1	0
1	0	1	1	1

$\therefore \quad Z = C$

Circuit diagram;



11. Write short notes on parity checker. **[2014/F]**

Solution: See the 5.6.2.

12. Design a combinational circuit that converts a decimal digit from the 2421 code to 8 4-2-1 code to binary. **[2014/F, 2015/S]**

Solution: See the example number 3.

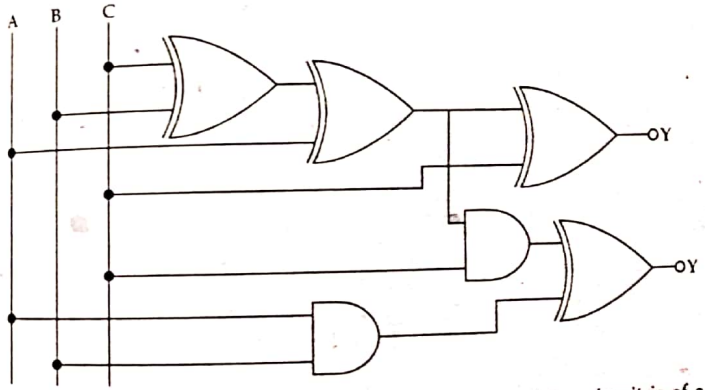
13. Design a single combinational logic circuit that performs the addition of two input bits (A and B) when third input bit C is set to 0 whereas the same circuit performs the subtraction of same two input bits when C is set to 1. **[2015/F]**

Solution:

A and B are two input bits and third input bit is C (act as switch).

Input			Output
A	B	C	
0	0	0	A + B
0	0	1	A – B
0	1	0	A + B
0	1	1	A – B
1	0	0	A + B
1	0	1	A – B
1	1	0	A + B
1	1	1	A – B

Circuit diagram;



From figure, except for XOR gate number 1, the remaining circuit is of a full adder with sum Y and carry Z.

When C = 0

At gate 1, $B \oplus C = B \oplus 0 = B$

So, the input for full adder are A and B with carry C = 0 and final out will be A + B

When C = 1

At gate 1, $B \oplus C = B \oplus 1 = \overline{B}$

So the input for full adder are A and $\overline{B}$ with carry set C = 1. So the final output of full adder will be $A + \overline{B} + 1$ which is subtraction (A – B) in 2's complement form.

14. Design a combination circuit that converts a decimal digit form the 2421 code to BCD. [2016/F]

Solution: See the example number 2

15. Design a circuit for 3-bit parity generation and 4-bit parity checker using even parity. [2016/S]

Solution:

For 3-bit even parity generator,

Let the three inputs A, B and C are applied to the circuits and output bit is the parity bit P. The total number of 1s must be even to generate even parity bit P.

Truth Table;

3 bit message			Even parity bit generator
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

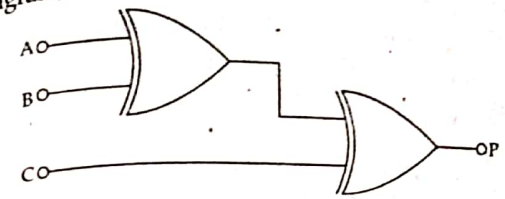
K-map

A \ B	00	01	11	10
00	0	(1)	0	(1)
01	(1)	0	(1)	0

From Table:

$$P = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$
$$= \overline{A}(\overline{B}C + B\overline{C}) + A(\overline{B}\,\overline{C} + BC)$$
$$= \overline{A}(B \oplus C) + A(\overline{B \oplus C})$$
$$\therefore \quad P = A \oplus B \oplus C$$

Circuit diagram;



For 4-Bit Even Parity Checker

Three input message along with the even parity bit is generated at the transmitting end. These 4 bits are applied as input to the parity checker circuit which checks the possibility of error on the data. Since the data is transmitted with even parity, four bits received at circuit must have an even number of 1s. If any error occurs, the received message consists of odd number of 1s.

Truth Table;

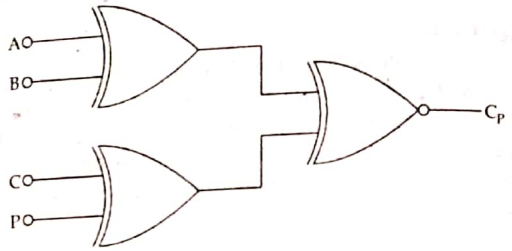
4-bit received message			Parity error check	
A	B	C	P	C_P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

K-map;

AB \ C_P	00	01	11	10
00	0	(1)	0	(1)
01	0	0	(1)	0
11	0	(1)	0	(1)
10	0	0	(1)	0

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

Circuit diagram;



16. Differentiate between combinational and sequential logic circuit. **[2011/F, 2015/F, 2012/S]**

Combinational circuits	Sequential circuits
It is type of digital circuit where the output is only a pure function of the present input.	It is a type of digital circuit whose output depends not only on the present value of its input signals but also on the sequence of past inputs.
There is no memory unit.	There is a memory unit to store immediate results.
There is no clock.	There is a clock.
Example; Half adder, full adder, decoder etc.	*Example;* Flip-flop, serial adder and registers.
Faster than sequential circuits.	Slower than combinational circuits.
Easy to design.	Difficult to design.
Number of gates required for the design is less.	Number of gates required for the design is more.
Input → Combinational logic circuit → Output *Figure: Combinational circuit block diagram*	Input → Combinational logic circuit → Output; Present state, Internal input, Next state, Memory, Clock signal *Figure: Sequential circuit block diagram*

17. Design a circuit of a 3-bit parity generator and the circuit of a 4-bit parity checker for odd parity. **[2017/F, 2018/F]**

Solution:

Let us consider that the 3-bit data is to be transmitted with an odd parity bit. The three inputs are A, B and C and P is the output parity bit. The total number of bits must be odd in order to generate the odd parity pit.

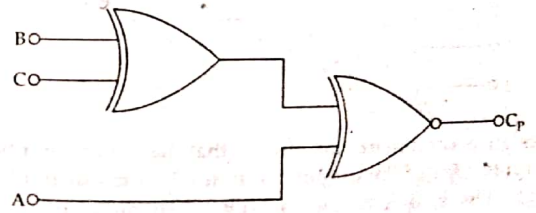
3 bit message			odd parity bit generator (P)
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

K-map;

A \ BC	00	01	11	10
00	(1)	0	(1)	0
01	0	(1)	0	(1)

$$\therefore \quad P = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C}$$
$$= \bar{A}(\bar{B}\bar{C} + BC) + A(\bar{B}C + B\bar{C})$$
$$= \bar{A}(\overline{B \oplus C}) + A(B \oplus C)$$
$$\therefore \quad P = \overline{A \oplus (B \oplus C)}$$

Circuit diagram;



4-bit odd parity checker;

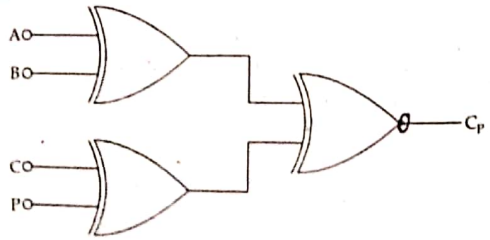
4 bit received message			Parity error check	
A	B	C	P	C_P
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

K-map,

AB \ C	00	01	11	10
00	(1)	0	(1)	0
01	0	(1)	0	(1)
11	(1)	0	(1)	0
10	0	(1)	0	(1)

$\therefore \quad C_P = \overline{(A \oplus B)} \oplus \overline{(C \oplus P)}$

Circuit diagram;



18. Design a combinational circuit that has four inputs and two outputs. One of the outputs is high when majority of inputs are high. The second output is high when majority of inputs are same type. **[2017/F, 2017/S, 2018/S, 2018/S]**

Solution:
Let A, B, C, D be four inputs. Let, X and Y be the outputs that is high when majority of inputs are high and when majority of inputs are high and when all inputs are of same type respectively. Majority of inputs are high means atleast 3 of 4 inputs are high.

Input				Output	
A	B	C	D	X	Y
0	0	0	0	0	1
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	1

K-map;
For X,

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	0

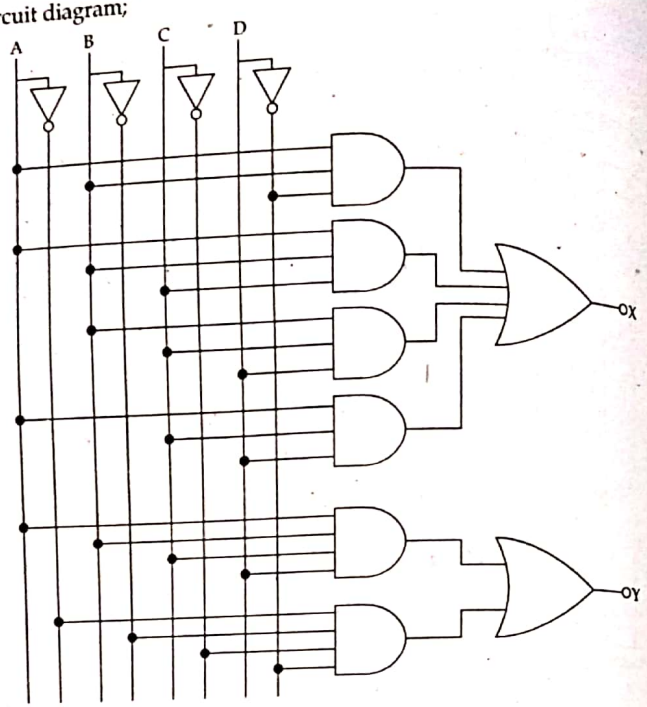
$\therefore \quad X = ABD + ABC + BCD + ACD$

For Y,

AB \ CD	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

$\therefore \quad Y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + ABCD$

Circuit diagram;



19. Write short notes on computation logic design procedure. [2019/S]

Solution: See the topic 5.1.

CHAPTER **6**

MSI AND LSI COMBINATIONAL LOGIC DESIGN

* * * * *

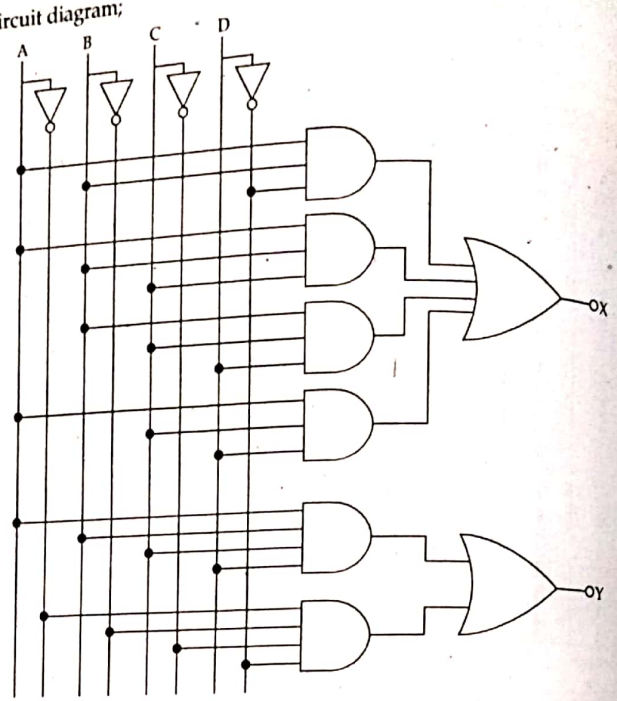
* * * * *

6.1 BINARY ADDER AND SUBTRACTOR

A. Four-Bit Binary Parallel Adder

In practical situations, it is required to add two data each containing more than one bit. Two binary numbers each of n bits can be added by means of a full adder circuit. Consider the example that two 4-bit binary numbers $B_4B_3B_2B_1$ and $A_4A_3A_2A_1$ are to be added with a carry input 1.

Circuit diagram;



19. Write short notes on computation logic design procedure. [2019/S]

Solution: See the topic 5.1.

CHAPTER **6**

MSI AND LSI COMBINATIONAL LOGIC DESIGN

* * * * *

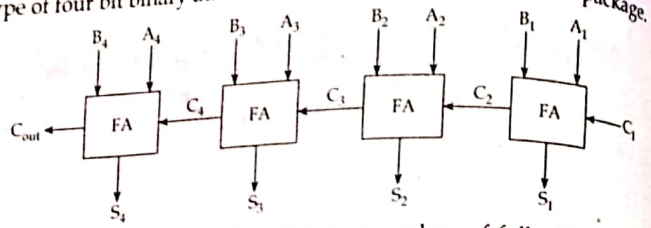
* * * * *

6.1 BINARY ADDER AND SUBTRACTOR

A. Four-Bit Binary Parallel Adder

In practical situations, it is required to add two data each containing more than one bit. Two binary numbers each of n bits can be added by means of a full adder circuit. Consider the example that two 4-bit binary numbers $B_4B_3B_2B_1$ and $A_4A_3A_2A_1$ are to be added with a carry input 1.

This can be done by cascading four full adder circuits. The least significant bits A_1, B_1 and C_1 are added to produce sum output S_1 and carry output C_2. Carry output C_2 is then added to the next significant bits A_2 and B_2 producing sum output S_2 and carry output C_3. C_3 is then added to A_3 and B_3 and so on. Thus finally producing the four output C_{out}. Such type of four bit binary adder is commercially available in an IC package.



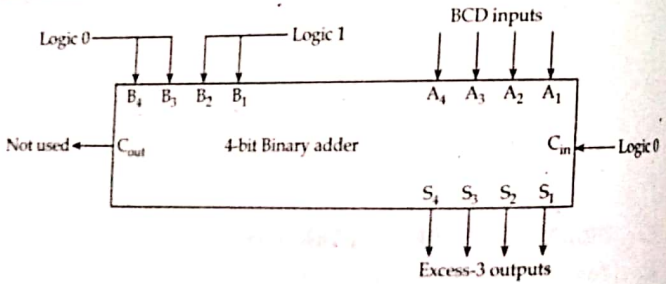
For the addition of two n bits of data, n numbers of full adders can be cascaded as demonstrated in above figure. It can be constructed with 4-bit, 2-bit and 1-bit full adder IC packages. The carry output of one package must be connected to the carry input of the next higher order bit IC package of higher order bits.

The addition technique adopted here is a parallel type as all the bit addition operations are performed in parallel. Therefore, this type of adder is called a parallel adder serial types of adders are also available where a signal full adder circuit can perform any n number of bit addition operations in association with shift registers and sequential logic networks. The 4-bit parallel binary adder IC package is useful to develop combinational circuits.

Example 1

Design a BCD to Excess-3 code converter.

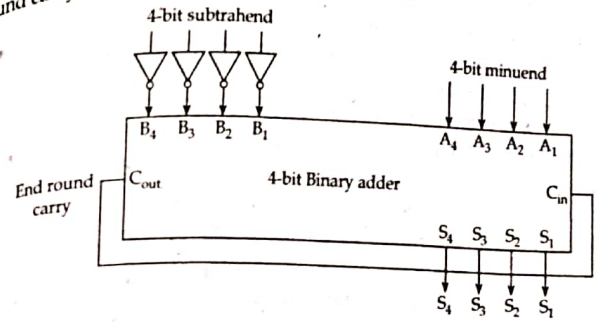
Solution:



Above figure requires only one MSI IC of 4-bit binary adder and interconnection have reduced drastically. So the combinational circuit is of low cost, less board, trouble free, less space consuming and less power dissipation.

8. Four bit Binary parallel Subtractor

Binary subtraction can be achieved by using 1's complement or 2's complement. By 1's complement method, the bits of subtrahend are complemented and added to the minuend. If the carry is generated it is added to the sum output. Figure below demonstrates the subtraction of $B_4B_3B_2B_1$ from $A_4A_3A_2A_1$. Each bit of $B_4B_3B_2B_1$ is first complemented by using Inverter gates and added to $A_4A_3A_2A_1$ by a 4-bit binary adder. End round carry is again using the C in pin of the IC.



6.2 DECIMAL ADDER

Calculators or computers that perform arithmetic operations directly in the decimal number system represent decimal numbers in binary coded form. An adder for such a computer must employ arithmetic circuits that accept coded decimal numbers and present results in the accepted code. For binary addition, it was sufficient to consider a pair of significant bits at a time, together with a previous carry. A decimal adder requires a minimum of nine inputs and five outputs, since four bits are required to code each decimal digit and the circuit must have an input carry and output carry. Of course, there is a wide variety of possible decimal adder circuits, dependent upon the code used to represent the decimal digits.

The design of a nine input, five output combinational circuit by the classical method requires a truth table with $2^9 = 512$ entries. Many of the input combinations are don't care conditions, since each binary code input has six combinations that are invalid. The simplified Boolean functions for the circuit may be obtained by a computer generated tabular method and the result would probably be a connection of gates forming an irregular pattern. An alternate procedure is to add the numbers with full adder circuits, taking into consideration the fact that six combinations in each 4-bit input are not used. The output must be modified so that only those binary combinations that are valid combinations of the decimal code are generated.

6.3 MAGNITUDE COMPARATOR

A magnitude comparator is one of the useful combinational logic networks and has wide applications. It compares two binary numbers and determines if one number is greater than, less than or equal to the other number. It is a multiple output combinational logic circuit. If two binary numbers are considered as A and B, the magnitude comparator gives three outputs for A > B, A < B and A = B.

For comparison of two n bit numbers, the classical method to achieve the Boolean expressions requires a truth table of 2^{2n} entries and becomes too lengthy and cumbersome. It is also desired to have a digital circuit possessing with a certain amount of regularity, so that similar circuits can be applied for the comparison of any number of bits. Digital function that allow an inherent well defined regularity can usually be developed by means of algorithm is a process that follows a finite set of steps to arrive at the solution to a problem. A method is illustrated here by deriving an algorithm to design a 4-bit magnitude comparator.

The algorithm is the direct application of the procedure to compare the relative magnitudes of two binary numbers. Let us consider the two binary numbers A and B are expanded in terms of bits in descending order as $A = A_4A_3A_2A_1$ $B = B_4B_3B_2B_1$

where, each subscripted letter represents one of the digits in the number. It is observed from the bit contents of the two numbers that A = B when $A_4 = B_4$, $A_3 = B_3$, $A_2 = B_2$ and $A_1 = B_1$. As the numbers are binary they possess the value of either 1 or 0, the equality relation of each pair can be expressed logically by the equivalent function as,

$$X_i = A_iB_i + \bar{A}_i\bar{B}_i \text{ for } i = 1, 2, 3, 4$$

or, $X_i = \overline{(A \oplus B)}$

or, $\bar{X}_i = A \oplus B$

or, $X_i = (A_i\bar{B}_i + \bar{A}_iB_i)'$

X_i is logic 1 when both A_i and B_i are equal, *i.e.*, either 1 or 0 at the same instant. To satisfy the equality condition of two numbers A and B, it is necessary that all X_i must be equal to logic 1. This distates the AND operation of all X_i variables. In other words, we can write the Boolean expression for two equal 4-bit numbers,

$$F(A = B) = X_4X_3X_2X_1$$

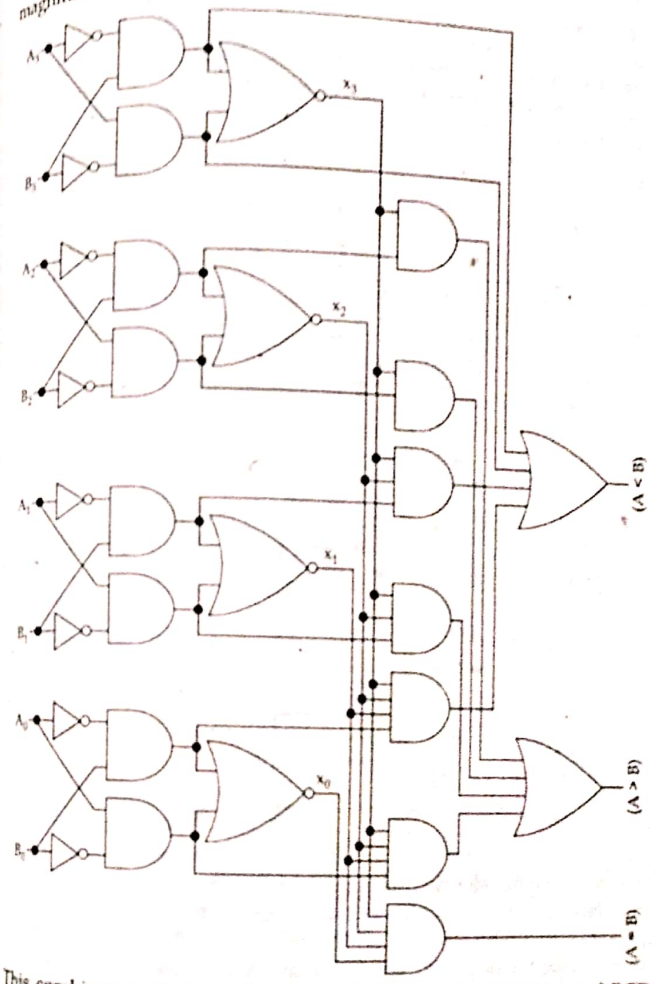
To determine the relative magnitude of two numbers A and B, the relative magnitudes of pairs of significant bits are inspected from the most significant positions are equal, the next significant pair of digits are compared. The comparison process is continued until a pair of unequal digits is found. It may be concluded that A > B, if the corresponding digit of A is 0 and B is 1.

Therefore, we can derive the logical expression of such sequential comparison by the following two Boolean functions.

$$F(A > B) = A_4\bar{B}_4 + X_4A_3\bar{B}_3 + X_4X_3A_2\bar{B}_2 + X_4X_3X_2A_1\bar{B}_1$$

and $F(A < B) = \bar{A}_4B_4 + X_4\bar{A}_3B_3 + X_4X_3\bar{A}_2B_2 + X_4X_3X_2\bar{A}_1B_1$

The logic gates implementation for the above expressions are not too complex as they contains many sub expressions of a repetitive nature and can be used at different places. The complete logic diagram of a 4-bit magnitude comparator is shown below;

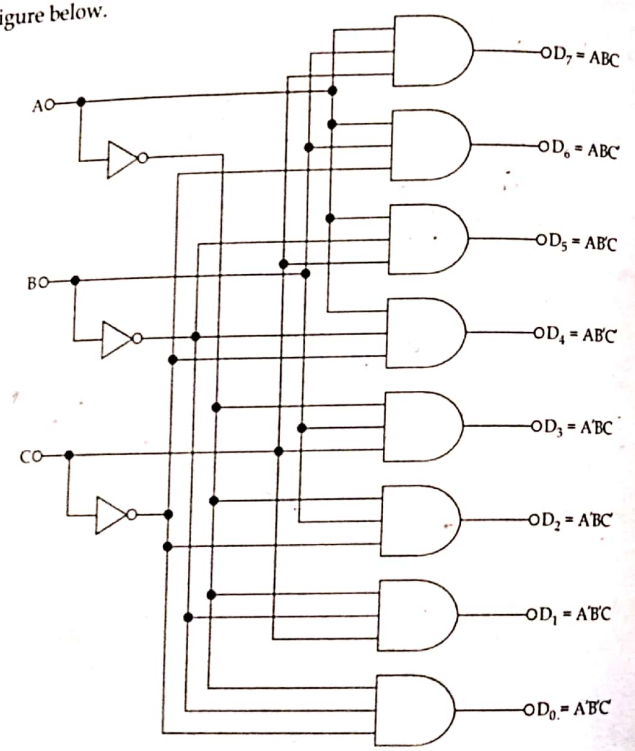


This combinational circuit is also applicable to the comparison of BCD numbers.

6.4 DECODER AND ENCODER

6.4.1 Decoder

In a digital system, discrete quantities of information are represented with binary codes. A binary code of n bits can represent up to 2^n distinct elements of the coded information. A decoder is a combinational circuit that converts n bits of binary information of input lines to a maximum of 2^n unique output lines. Usually decoders are designated as an n to m lines decoder, where n is the number of input lines and $m = 2^n$ is the number of output lines. Decoders have a wide variety of applications in digital systems such as data demultiplexing, digital display, digital to analog converting, memory addressing etc. A 3 to 8 line decoder is illustrated in figure below.



The 3 to 8 line decoder consists of three input variables and eight output lines. Note that each of the output lines represents one of the minterms generated from three variables. The internal combinational circuit is realized with the help of Inverter gates and AND gates. The operation of the decoder circuit may be further illustrated from the input output relationship as given in the table.

Input Variable			Output							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

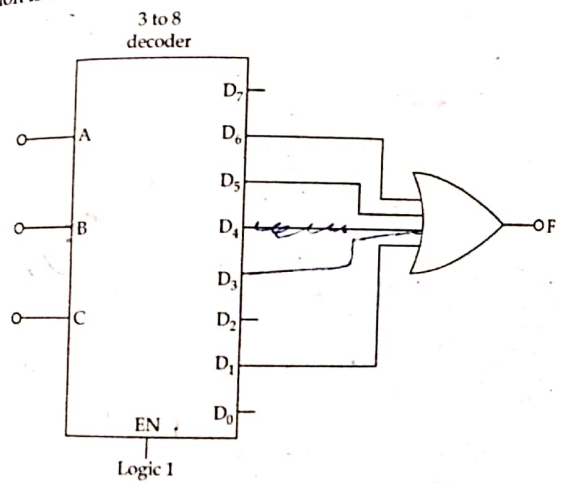
Note that the output variables are mutually exclusive to each other, as only one output is possible to be logic 1 at any one time. Higher order decoders like 4 to 16 lines, 5 to 32 lines etc are also available in MSI packages where the internal circuits are similar to the 3 to 8 line decoder.

Example 2

Implement the function $F(A, B, C) = \Sigma(1, 3, 5, 6)$

Solution:

Since the above function has three input variables, a 3 to 8 line decoder may be employed. It is in the sum of the products of the minterms m_1, m_3 m_5 and m_6 and so decoder output D_1, D_3, D_5 and D_6 may be OR-gated to achieve the desired function. The combinational circuit of the above function is shown below;



Example 3

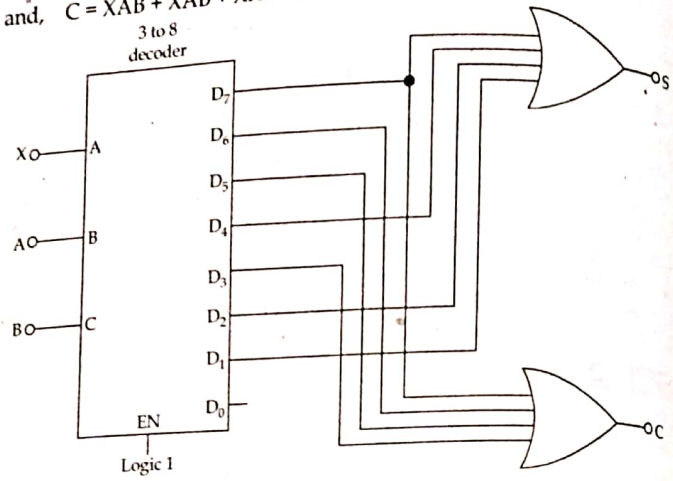
Design a full adder circuit with decoder IC

Solution:

In terms of minterms, the Boolean expression of sum output S and carry output C can be written as,

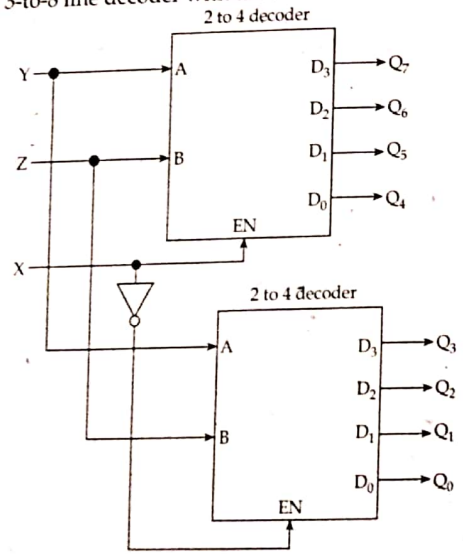
$S = \bar{X}\bar{A}B + \bar{X}A\bar{B} + X\bar{A} + XAB$

and, $C = \bar{X}AB + X\bar{A}B + XA\bar{B} + XAB$



Example 4

Construct a 3-to-8 line decoder with the use of 2-to-4 line decoder.

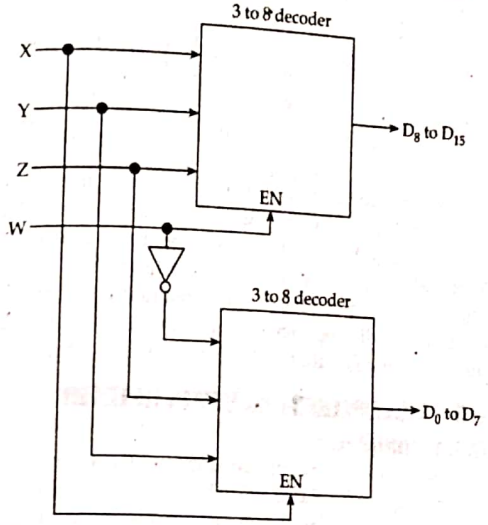


Solution:

Lower order decoders can be cascaded to build higher order decoders. Normally, ever commercially available decoder ICs have a special input other than normal working input variables called Enable. The use of this Enable input is that when activated the complete IC comes to the working

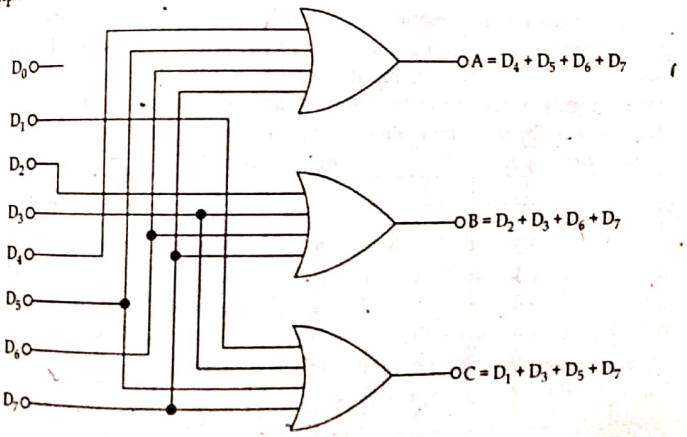
condition for its normal functioning. If Enable input is deactivated, the IC goes to sleep mode, the normal functioning is suspended and all the output become logic 0 irrespective of normal input variables conditions. This behaviour of Enable input makes good use of a cascade connection.

Here input variable are designated as X, Y, and Z and outputs are denoted as Q_0 to Q_7. X input is connected to the Enable input of one decoder and X is used as an Enable input of another decoder. When X is logic 0, a lower decoder is activated and gives output Q_0 to Q_3 and upper decoder is activated for X is logic 1, output Q_4 to Q_7 are available this time.



6.4.2 Encoders

An encoder is a combinational network that performs the reverse operation of the decoder. An encoder has 2^n or less numbers of inputs and n output lines. The output lines of an encoder generate the binary code for the 2^n input variables. Figure below illustrates an eight inputs/three outputs encoder.



The truth is given below;

Inputs								Output		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	1	0	0	1	1
0	0	0	0	1	0	0	1	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

It may be noted that for eight inputs there are possible $2^8 = 256$ combinations, but only eight combinations are useful and the rest are don't care conditions. It may also be noted that D_0 input is not connected to any of the gates. All the binary outputs A, B and C must be all 0s in this case. All 0s output may also be obtained if all input variables D_0 to D_7 are logic 0. This discrepancy can be eliminated by introducing another output indicating the fact that all the inputs are not logic 0.

6.5 MULTIPLEXER AND DEMULTIPLEXER

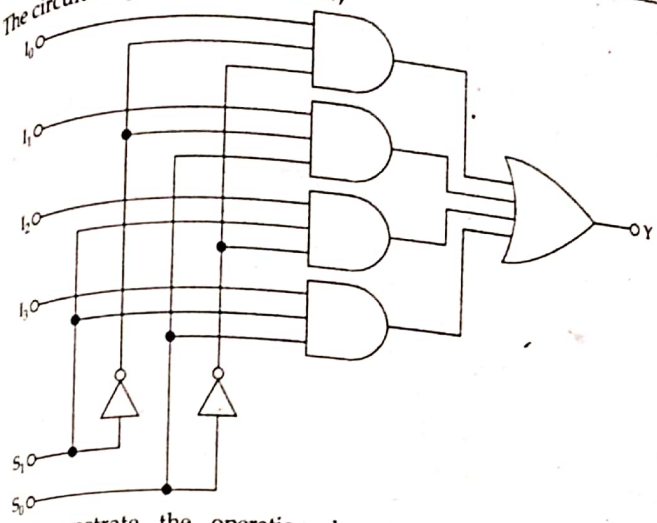
6.5.1 Multiplexer

A multiplexer is one of the important combinational circuits and has a wide range of applications. The term multiplex means "many into one". Multiplexers transmit large numbers of information channels to a smaller number of channels. A digital multiplexer is a combinational circuit that selects binary information from one of the many input channels and transmits to a single output line. That is why the multiplexers are also called the data selectors. The selection of the particular input channel is controlled by a set of select inputs. A digital multiplexer of 2^n input channels can be controlled by n numbers of select lines and an input line is selected according to the bit combinations of select lines.

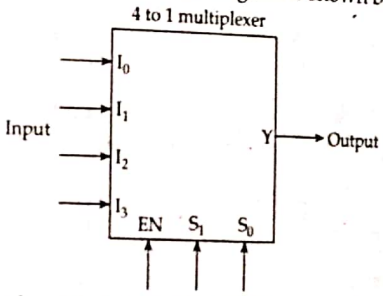
A 4 to 1 line multiplexer is defined as the multiplexer consisting of four input channels and information of one of the channels can be selected and transmitted to an output line according to the select inputs combinations. Selection of one of the four input channels is possible by two selection inputs. Figure below illustrates the truth table. Input channels I_0, I_1, I_2 and I_3 are selected by the combinations of select inputs S_1 and S_0.

Selection inputs		Input channels				Output
S_1	S_0	I_0	I_1	I_2	I_3	Y
0	0	0	X	X	X	0
0	0	1	X	X	X	1
0	1	X	0	X	X	0
0	1	X	1	X	X	1
1	0	X	X	0	X	0
1	0	X	X	1	X	1
1	1	X	X	X	0	0
1	1	X	X	X	1	1

The circuit diagram is shown below;

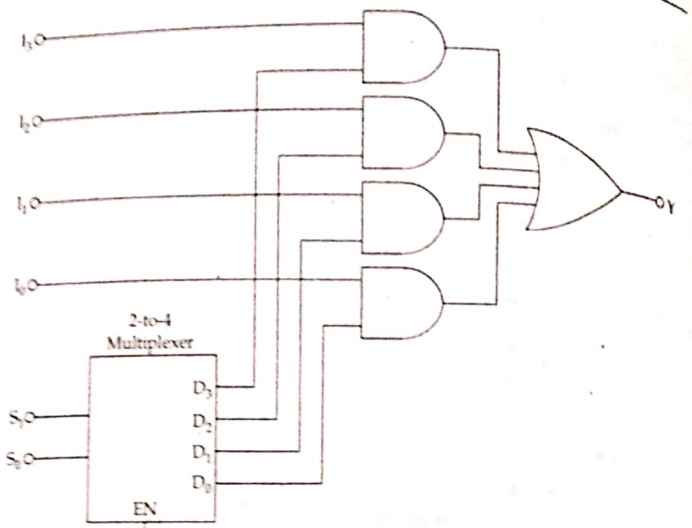


To demonstrate the operation, let us consider that select input combination S_1S_0 is 01. The AND gate associated with I_1 will have two of inputs equal to logic 1 and a third input is connected to I_1. Therefore, output of this AND gate is according to the information provided by channel I_1. The other three AND gates have logic 0 to at least one of their inputs which makes their outputs to logic 0. Hence, OR output (Y) is equal to the data provided by the channel I_1. Thus information from I_1 is available at Y. Normally a multiplexer has an Enable input to control its operation. The Enable input is useful to expand two or more multiplexer ICs to the digital multiplexer with a large number of inputs. A multiplexer is often abbreviated as MUX. Its block diagram is shown below;



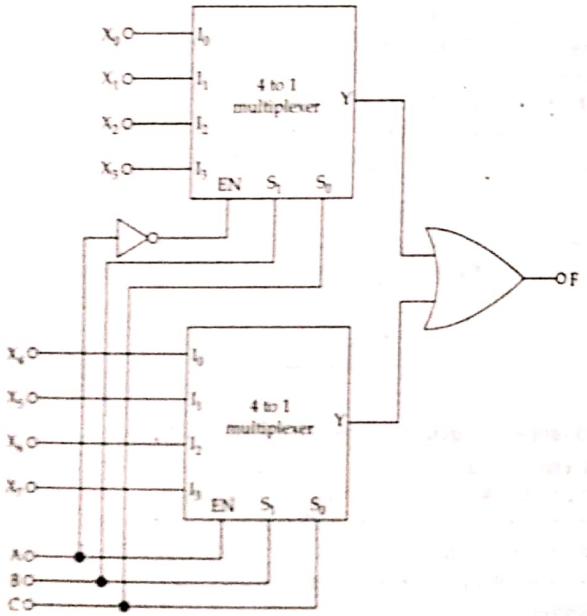
If the multiplexer circuit is inspected critically, it may be observed that the multiplexers circuit resembles the n select lines are decoded to 2^n lines which are ANDed with the channel inputs. In some cases two or more multiplexers are accomodated within one IC package. The selection and Enable inputs in multiple-unit ICs may be common to all multiplexers.

Figure below demonstrates how a decoder is employed to form a 4 to 1 multiplexer.



Cascading of Multiplexers

Multiplexers of a large numbers of inputs can be implemented by the multiplexers of a smaller number of input lines. Figure below shows an 8 to 1 line multiplexer that is realized by two 4 to 1 line multiplexers.



Here variables B and C are applied to select input S_1 and S_0 of both multiplexers whereas the Enable input of the upper multiplexer is connected to A. So for A = 0, the upper multiplexer is selected according to the selected inputs and data is transmitted to an output through the OR gate. When A = 1, the lower multiplexer is activated and input lines X_4 to X_7 are selected according to the selected inputs.

Boolean Function Implementation

Multiplexers can be constructed with decoders and OR gates. The selection of minterm outputs of the decoder can be controlled by the input lines. Hence the minterms included in the Boolean function may be chosen by making their corresponding input lines to logic 1. The minterms not needed for the function are disabled by making their input lines equal to logic 0. By this method, Boolean functions of n variables can be very easily implemented by using a 2^n to 1 multiplexers. However, a better approach may be adopted with the judicious use of the function variables.

If a Boolean function consists of n + 1 number of variables, n of these variables may be used as the select inputs of the multiplexer. The remaining single variable of the function is used as the input lines of the multiplexer. If X is the left-out variable, the input lines of the multiplexer may be chosen from four possible values-X, $\overline{X}$, logic 1 or logic 0. It is possible to implement any Boolean function with a multiplexer by intelligent assignment of the above values to input lines and other variables to selection lines. By this method a Boolean function of n + 1 variables can be implemented by a 2^n to 1 line multiplexer.

Example 5

Implement the 3-variable function F(A, B, C) = (0, 2, 4, 7) with a multiplexer.

Solution:

Here the function has three variables A, B and C and can be implemented by a 4 to 1 line multiplexer. Two variables B and are connected to the selection lines S_1 and S_0 respectively when both B and C are 0, I_0 is selected. At this time, the output required is logic 1 as both the minterms m_0 ($\overline{A}\overline{B}\overline{C}$) and m_4 ($A\overline{B}\overline{C}$) produce output logic 1 regardless of the input variable A. So I_0 should be connected to logic 1. When select inputs BC = 01, I_1 is selected and it should be connected to logic 0 as the corresponding minterms m_1 ($\overline{A}\overline{B}C$) and m_5 ($A\overline{B}C$) both produce output 0.

For select inputs BC = 10, I_2 is selected and connected to variable $\overline{A}$ as only one minterm m_2 ($\overline{A}\,\overline{B}C$) associated with $\overline{A}$ produce output logic 1, whereas the minterm m_6 ($AB\overline{C}$) associated with A produces output 0. And

finally I_3 is selected and connected to variable A, when select input BC = 11 because only the minterms m_7 (ABC) produce output 1 whereas output is 0 for the minterm m_3 ($\bar{A}$BC). Multiplexer must be in Enable mode to be at its working condition. Hence EN input is connected to logic 1.

The truth table for above Boolean function

Minterms	A	B	C	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

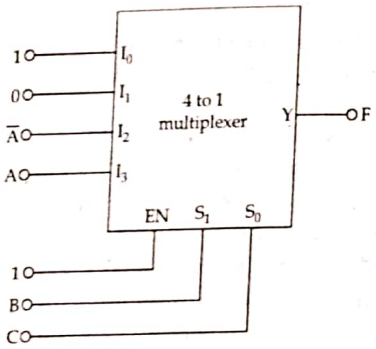


Figure: 4 to 1 line multiplexer circuit

Example 6

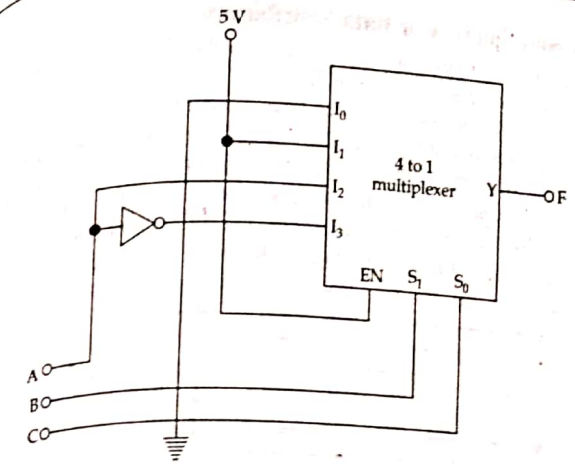
Implement the following function using a multiplexer

F(A, B, C) = (1, 3, 5, 6)

Solution:

The function contains three variables. The function can be realized by one 4 to 1 multiplexer. The implementation table is shown below;

	I_0	I_1	I_2	I_3
$\bar{A}$	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	$\bar{A}$



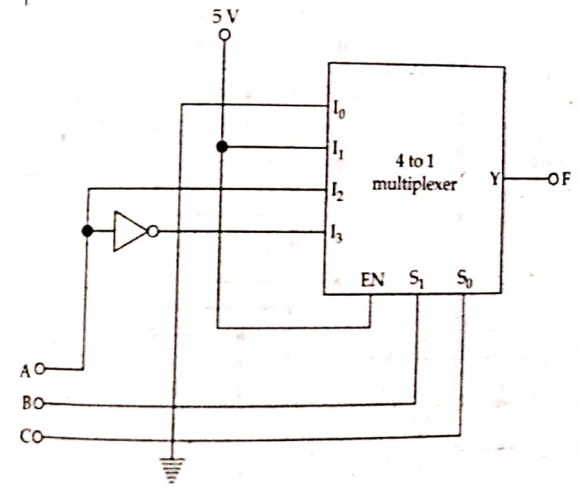
Example 7

Implement the following function with a multiplexer, F(A, B, C, D) = (0, 1, 3, 4, 8, 9, 15)

Solution:

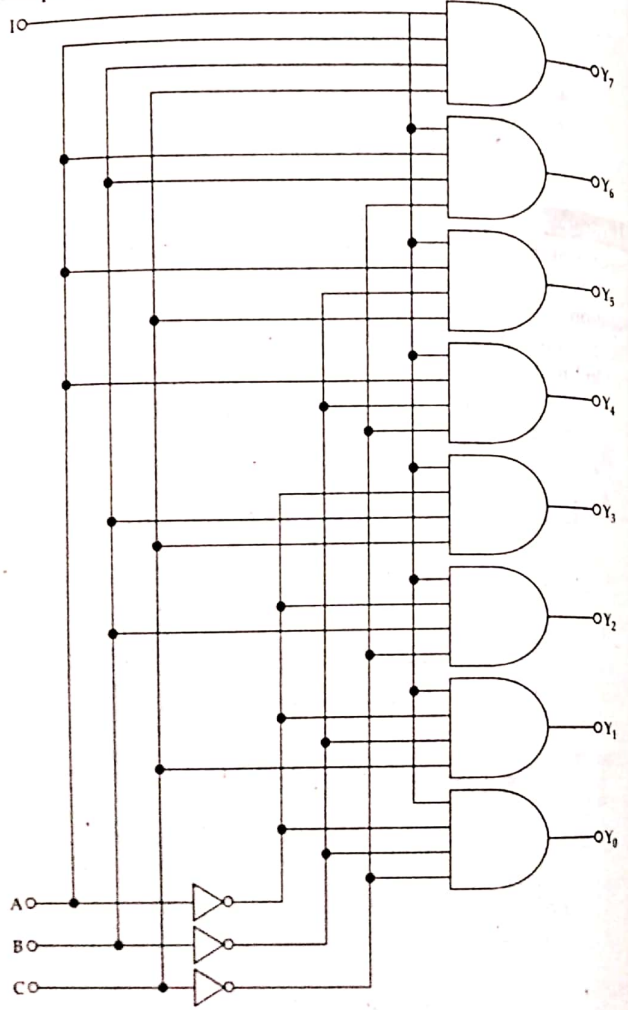
The given function contains four variables. The function can be realized by 8 to 1 multiplexer.

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	(0)	(1)	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A



6.5.2 Demultiplexers or Data Distributors

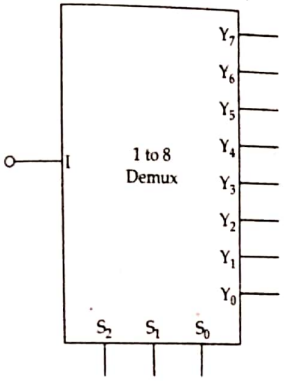
The term demultiplex means one into many. Demultiplexing is the process that receives information from one channel and distributes the data over several channels. It is the reverse operation of the multiplexer. A demultiplexer is the logic circuit that receives information through a single input line and transmits the same information over one of the possible 2^n output lines. The selection of a specific output line is controlled by the bit combination of the selection lines. A 1 to 8 demultiplexer circuit is demonstrated below;



The selection input lines A, B and C activate an AND gate according to its bit combination. The input line I is common to one of the inputs of all the AND gates. So information of I passed to the output line is activated by the particular AND gate.

Selection inputs			Outputs							
A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	I	0	0	0	0	0	0	0
0	0	1	0	I	0	0	0	0	0	0
0	1	0	0	0	I	0	0	0	0	0
0	1	1	0	0	0	I	0	0	0	0
1	0	0	0	0	0	0	I	0	0	0
1	0	1	0	0	0	0	0	I	0	0
1	1	0	0	0	0	0	0	0	I	0
1	1	1	0	0	0	0	0	0	0	I

The demultiplexer is symbolized in figure below;



Demultiplexer circuits may be derived from a decoder with the use of AND gates. Like decoders and multiplexer, demultiplexers can also be cascaded to form higher order demultiplexers.

Demultiplexer and Decoder Relations

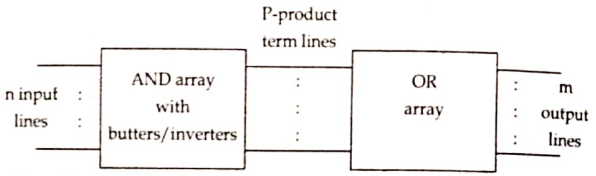
Decoder	Demultiplexer
Decoder is a many input to many outputs device.	Demultiplexer is a one input to many outputs device.
There are no selection lines.	The selection of specific output line is controlled by the value of selection lines.
Its main function is for decoding of the encoded input terminal.	It is used switching.
A decoder accepts control signals as input	A demultiplexer accepts the data as input.

They are practically used in many applications such as data demultiplexing, memory address decoding etc.	They are mainly used in communication systems and networking solutions for security purposes.
It is the inverse function of an encoder.	It is an inverse function of multiplexer.
A decoder has 'n' input lines and maximum of 2^n output lines.	A De-multiplexer has single input 'n' selection lines and maximum of 2^n output.

6.6 READ-ONLY MEMORY (ROM)

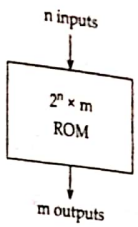
A ROM is essentially a memory device for storage purpose in which a fixed set of binary information is stored. An user must first specify the binary information to be stored and then it is embedded in the unit to form the required interconnection pattern. ROM contains special internal links that can be fused or broken. A ROM is a device that includes both the decoder and the OR gates within a single IC package. The connections between the outputs of the decoder and the inputs of the OR gates can be specified for each particular configuration. The ROM is used to implement complex combinational circuits within one IC package or as permanent storage for binary information. Once a pattern is established for a ROM, it remains fixed even if the power supply to the circuit is switched off and then switched on again.

A block diagram of ROM is shown below;



It consists of n input lines and M output lines. Each bit combination of input variables is called an address and each bit combination that is formed at output lines is called a word. Thus, an address is essentially a binary number that denotes one of the minterms of n variables and the number of bits per word is equal to the number of output lines m. It is possible to generate $P = 2^n$ number of distinct addresses from n number of input variables. Since there are 2^n distinct addresses in a ROM, there are 2^n distinct words which are said to stored in the device and an output word can be selected by a unique address. The address value applied to the input lines specifies the word at input lines at any given time. A ROM is characterized by the number of words 2^n and number of bits per word m and denoted as $2^n \times m$ ROM.

The ROM is a two level logic representation in the sum of products form. It may not be essentially on AND-OR realization but it can be realized by other two level minterm implementation. The second level is usually a wired logic connection to facilitate the blowing of the links.



ROM has many important applications in the design of digital computer systems. Realization of complex combinational circuits, code conversions, generating bit patterns, performing arithmetic functions like multipliers, forming look-up tables for arithmetic functions, and bit patterns for characters are some of its application. They are particularly useful for the realization of multiple output combinational circuits with the same set of inputs. As such, they are used to store fixed bit patterns that represent the sequence of control variables needed to enable the various operations in the system. They are also used in association with micro processors and microcontrollers.

Example 8

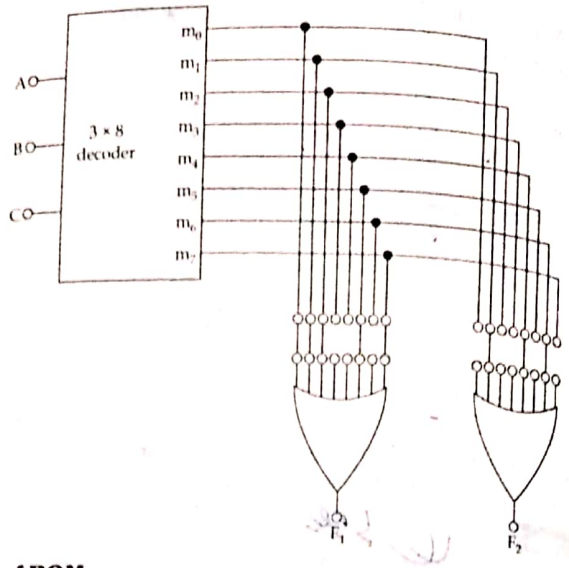
Consider that the following Boolean functions are to be developed using ROM $F_1(A, B, C) = (0, 1, 2, 5, 7)$ and $F_2(A, B, C) = (1, 4, 6)$

Solution:

When a combinational circuit is developed by means of a ROM, the functions must be expressed in sum of minterms or by a truth table. The truth table of the above functions is below;

Decimal equivalent	Input variable			Out puts	
	A	B	C	F_1	F_2
0	0	0	0	1	0
1	0	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	0	1
7	1	1	1	1	0

Since there are three input variables, a ROM centaining a 3 to 8 line decoder is needed. In addition, since there are two output functions, the OR array most contains at least two OR gates. That means, a $2^3 \times 2$ ROM or 8×2 ROM is to be employed to realize the above functions. The logic diagram of the ROM after blowing off the appropriate fuses is illustrated in figure below;



Types of ROM

a) Programmable Read only Memory (PROM)

It is more economic in cases requiring small quantities. In this method, the manufacturer provides the PROM with all 0s (or all 1s) in every bit of the stored words. The required links are broken by application of current pulses. This allows the user to program the device in his own laboratory to obtain the desired relationship between input addresses and stored words. Special equipments called PROM programmers are commercially available to facilitate this procedure. In any case, all procedures for programming ROMS are hardware procedures even though the word programming is used.

b) Erasable PROM (EPROM)

The hardware procedure for programming of ROMS or PROMS as described above is irreversible and once programmed, the configuration is fixed and cannot be altered. The device must be discarded if the bit pattern is required to be changed or modified. A third type of units is available to overcome this disadvantage which is called EPROM. This device can reconstruct the initial bit patterns of all 0s or all 1s though it is already programmed for some bit configuration. In other words, this device can be erased. This is achieved by placing the Erasable PROM under a special ultraviolet light for a given time. The short wave radiation discharges the internal gates that serve as links. After erasure, the device returns to its initial state and can be reprogrammed.

c) Electrically Erasable PROM (EEPROM)

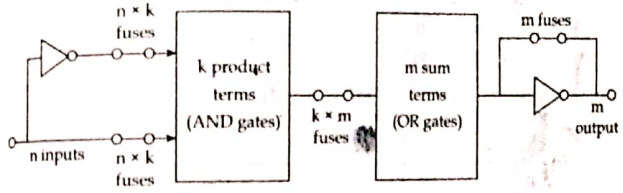
With the advancement of fabrication technology, further improvement of ROM has taken place, where ultraviolet light is not necessary to erase the programmed data. A new technique has been introduced to erase the bit pattern of ROM where bit patterns are reset to their original state of all 0s or all 1s by applying a special electrical signal. After by applying a special electrical signal. Afterwards, the device can be reprogrammed with an alternate bit pattern. The equipment called EPROM programmer serves the purpose of erasure as well as the programming the bit patterns.

6.7 PROGRAMMABLE LOGIC ARRAY (PLA)

A combinational network may occassionally contain don't care conditions. During the ROM implementation of this combinational circuit, this don't care condition also forms an address input that will never occur. The words at the don't care addresses need not be programmed and may be left in their original state of all 0s or all 1s. Since some of the bit patterns are not at all used, the address locations corresponding to don't care conditions are considered a waste of memory.

For the case where don't care conditions are excessive, it is more economical to use a second type of LSI device called programmable logic Array or PLA. PLA is similar to a ROM in concept. However, a PLA does not contain all AND gates to form the decoder or does not generate all the minterms like ROM. In the PLA, the decoder is replaced by a group of AND gates with buffers/inverters, each of which can be programmed to generate some product terms of input variable combinations that are essential to realize the output functions. The AND and OR gates inside the PLA are initially fabricated with the fusible links among them. The required Boolean functions are implemented in the sum of products form by opening the appropriate links and retaining the desired connections.

A block diagram of the PLA is shown below;



It consists of n inputs, m outputs, K product terms and m sum terms. The product terms constitute a group of K AND gates and the sum terms constitute a group of m or gates.

Fuses are inserted between all n inputs and their complement values to each of the AND gates. Another set of fuses in the output inverters allows the output function to be generated either in the AND-OR form or in the AND-OR-Invert form. With the inverter fuse in place, the inverter is by

passed, giving an AND-OR implementation. With the fuse blown, the inverter becomes part of the circuit and the function is implemented in the AND-OR-invert form.

The size of the PLA is specified by the number of inputs, the number of product terms and the number of outputs (the number of sum terms is equal to the number of outputs). A typical PLA has 16 inputs, 48 product terms and 8 outputs. The number of programmed fuses is $2n \times k + k \times m$ + m, whereas that of a ROM is $2^n \times m$.

Example 9

A combinational circuit is defined by the functions

$F_1(A, B, C) = \Sigma(3, 5, 6, 7)$

$F_2(A, B, C) = \Sigma(0, 2, 4, 7)$

Implement the circuit with a PLA having three inputs, four product terms and two outputs.

Solution:

The two functions are simplified in the maps. Both the true values and the complements of the functions are simplified.

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$			1	
A		1	1	1

$F_1 = AC + AB + BC$

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1			1
A	1		1	

$F_2 = \bar{B}\bar{C} + \bar{A}\bar{C} + ABC$

The combinations that give a minimum number of product terms are,

$\bar{F}_1 = \bar{B}\bar{C} + \bar{A}\bar{C} + \bar{A}B$

$\bar{F}_2 = \bar{B}\bar{C} + \bar{A}\bar{C} + AB\bar{C}$

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	0		0
A	0			

$\bar{F}_1 = \bar{B}\bar{C} + \bar{A}\bar{C} + \bar{A}B$

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$		0	0	
A		0		0

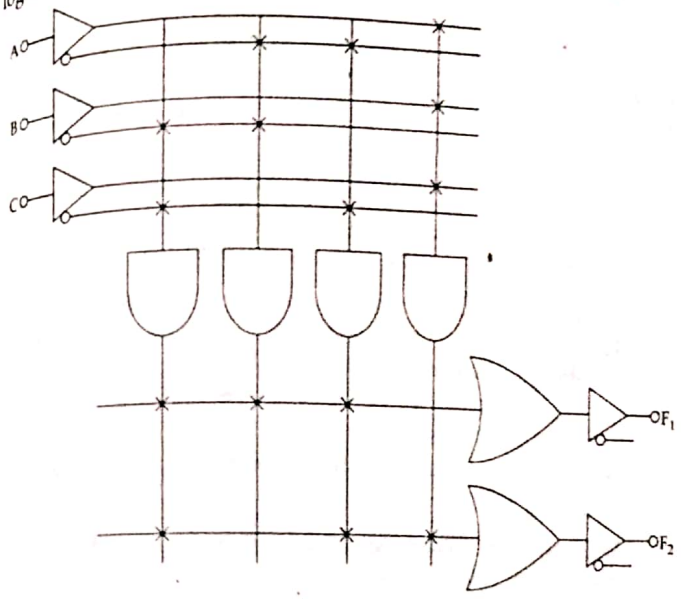
$\bar{F}_2 = \bar{B}\bar{C} + \bar{A}\bar{C} + AB\bar{C}$

PLA program Table

	Product term	Inputs			Outputs	
		A	B	C	F_1	F_2
$\bar{B}\bar{C}$	1	-	0	0	1	1
$\bar{A}\bar{C}$	2	0	-	0	1	1
$\bar{A}\bar{B}$	3	0	0	-	1	-
$AB\bar{C}$	4	1	1	1	-	1
					C	T

T/C

This gives only four distinct product terms $\bar{B}\bar{C}$, $\bar{A}\bar{C}$, $\bar{A}\bar{B}$ and $AB\bar{C}$. The logic diagram for the above combinational circuit is shown below;



EXAMINATION QUESTION SOLUTIONS

1. What is multiplexer? Describe 4:1 multiplexer with internal diagram. [2011/F]

Solution: See the topic 6.5.1.

2. Write short notes on PLA. [2011/S, 2012/F, 2014/S, 2016/F, 2016/S]

Solution: See the topic 6.7.

3. Write short notes on decoder and encoder. [2012/S, 2017/F]

Solution: See the topic 6.4.1 and 6.4.2.

4. Differentiate between PLA and ROM. [2013/S]

Solution:

PLA	ROM
PLA has the capability to take don't care terms.	RAM has not such capability.
PLA does not have all the combinations.	ROM has all the combinations of product terms.
IN PLA, both AND and OR arrays are configurable.	In ROM, only OR gate array are configurable.
It has one element per minterm.	It is fully populated.

5. Design a comparator circuit that compares two 4 bit numbers. The two numbers being A and B. It is required to obtain three possible outcomes *i.e.*, A > B, A < B and A = B. [2014/S, 2015/S, 2016/F, 2018/F]

OR

Write short notes on magnitude comparator. [2013/F, 2015/S,]

Solution: See the topic 6.3.

6. Define the term LSI and MSI. Design 3 : 8 decoder with its logic diagram and block diagram. [2016/S]

Solution:

Large scale Integration (LSI) is the process of integrating or embedding thousands of transistors on a single silicon semiconductor microchip. It contains 1,000 to 1,00,000 active devices per chip.

Medium scale Integration (MSI) is an integrated circuit that contains 100 to 1000 active devices per chip.

3 : 8 decoder

See the topic 6.4.1.

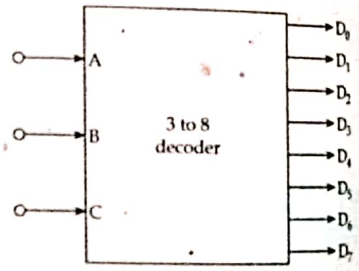


Figure: Block diagram

7. With the help of an example, show how you can construct a higher order MUX using two or more number of lower order MUXS. [2014/F]

Solution: See the topic 6.5.1 "Cascading of Multiplexers".

8. Implement the following function with a multiplexer.

$F(A, B, C, D) = \Sigma(0, 1, 3, 4, 8, 9, 15)$ [2011/S]

Solution:

This is a 4-variable function and therefore we need a multiplexer with three selection lines S_2, S_1 and S_0 and eight input lines. Variables B, C, D are connected to selection lines S_2, S_1 and S_0 respectively.

Implementation Table:

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	(0)	1	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A

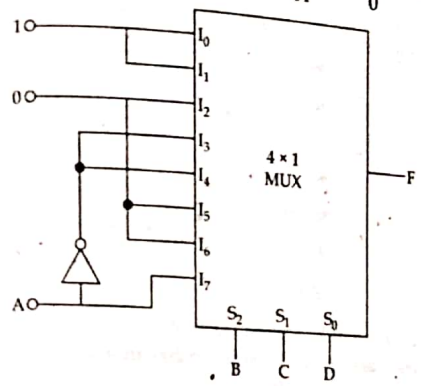


Figure: 4 × 1 MUX

9. A combinational circuit is defined by following three functions.

$F_1 = \bar{X}\bar{Y} + XY\bar{Z}$

$F_2 = \bar{X} + Y$

$F_3 = XY + \bar{X}\bar{Y}$

Design a circuit with decoder and external gates. [2011/F]

Solution:

$F_1 = \bar{X}\bar{Y} + XY\bar{Z}$

$F_2 = \bar{X} + Y$

$F_3 = XY + \bar{X}\bar{Y}$

Expressing each function as sum of minterms,

$$F_1 = \bar{X}\bar{Y}(Z + \bar{Z}) + XY\bar{Z}$$
$$= \bar{X}\bar{Y}Z + \bar{X}\bar{Y}\bar{Z} + XY\bar{Z}$$
$$= \Sigma(0, 1, 6)$$
$$F_2 = \bar{X}(Y + \bar{Y}) + Y(X + \bar{X})$$
$$= \bar{X}Y(Z + \bar{Z}) + \bar{X}\bar{Y}(Z + \bar{Z}) + XY(Z + \bar{Z}) + \bar{X}Y(Z + \bar{Z})$$
$$= \bar{X}YZ + \bar{X}Y\bar{Z} + \bar{X}\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z + XYZ + XY\bar{Z} + \bar{X}YZ + \bar{X}Y\bar{Z}$$
$$= \bar{X}\bar{Y}\bar{Z} + \bar{X}YZ + \bar{X}Y\bar{Z} + \bar{X}YZ + XY\bar{Z} + XYZ$$
$$= \Sigma(0, 1, 2, 3, 6, 7)$$
$$F_3 = XY(Z + \bar{Z}) + \bar{X}\bar{Y}(Z + \bar{Z})$$
$$= \bar{X}\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z + XY\bar{Z} + XYZ$$
$$= \Sigma(0, 1, 6, 7)$$

We need 3 × 8 decoder for 3 variable X, Y and Z and 3-OR gates each for each function.

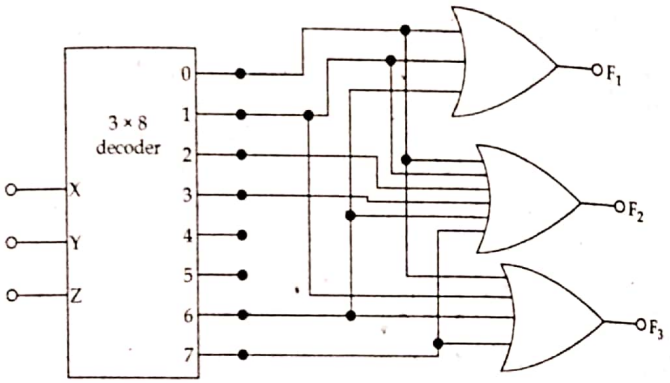


Figure: Circuit diagram

10. Design a circuit that compares two 3 bit numbers A and B to cheek if they are equal. The circuit has one output Y so that Y = 1 if A = B and Y = 0 if A ≠ B. **[2011/F]**

Solution:

Let, $A = a_2a_1a_0$

$B = b_2b_1b_0$

Let, $x_i = a_ib_i\,\bar{a}_i\bar{b}_i$, i = 0, 1, 2

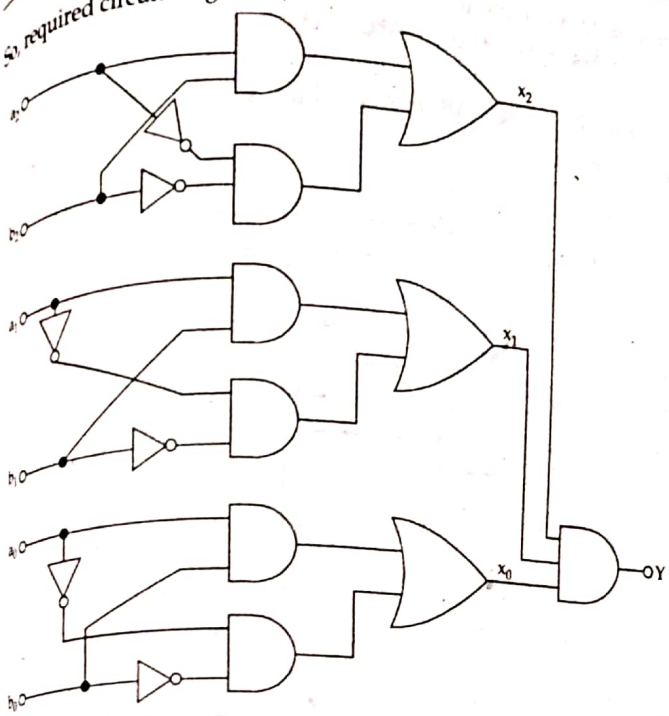
Here, $x_i = 1$ only when $a_i = b_i = 1$ or $a_i = b_i$

$= 0$ *i.e.*, when both digits are same. So the numbers A and B will be equal when, $x_2 \cdot x_1 \cdot x_0 = 1$

If Y be the output of circuit, then

$$Y = x_2 \cdot x_1 \cdot x_0$$
$$= (a_2b_2 + \bar{a}_2\bar{b}_2) \cdot (a_1b_1 + \bar{a}_1\bar{b}_1) \cdot (a_0b_0 + \bar{a}_0\bar{b}_0)$$

So, required circuit diagram is,



If Y = 1, then A = B

If Y = 0, then A ≠ B

11. Design a combinational circuit using a ROM. The circuit accepts 3 bit numbers and generates an output binary number equal to the sum of the input numbers. **[2011/F]**

Solution:

Let, A, B and C be three bit binary input to ROM and X, Y be output binary number where X is carry of sum and Y is the sum bit.

Truth Table;

Input			Output	
A	B	C	X	Y
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

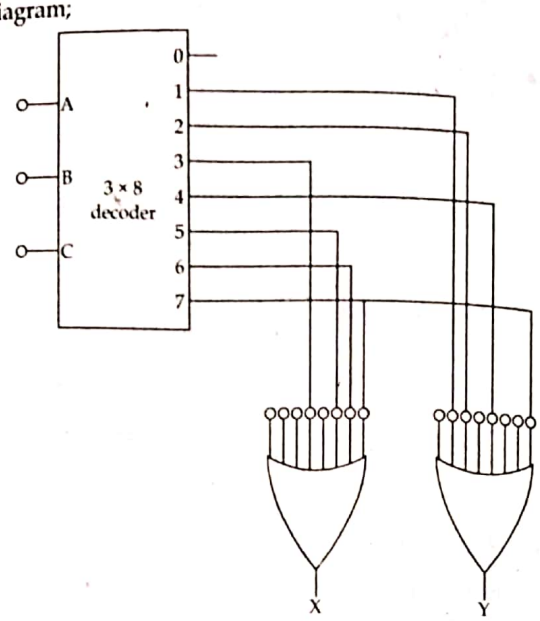
Expression for x in terms of minterms,

$X = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC = \Sigma(3, 5, 6, 7)$

For Y,

$Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC = \Sigma(1, 2, 4, 7)$

Circuit Diagram;



12. Derive a PLA program table for the combinational circuit that squares 3 bit number. [2012/S]

Solution:

The truth table and K-map for square of 3 bit number is already done in chapter 5-example number 1.

For inputs X, Y, Z and outputs A, B, C, D, E, F

$A = XY$

$B = X\bar{Y} + XZ$

$C = \bar{X}YZ + X\bar{Y}Z$

$D = Y\bar{Z}$

$E = 0$

$F = Z$

For $\bar{A}$, PLA Table:

X \ YZ	00	01	11	10
0	0	0	0	0
1	0	0		

$\therefore \quad \bar{A} = \bar{X} + \bar{Y}$

For $\bar{B}$,

X \ YZ	00	01	11	10
0	0	0	0	0
1				0

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

$\therefore$ For $\bar{C}$,

X \ YZ	00	01	11	10
0	0	0		0
1	0		0	0

$\bar{C} = \bar{X}\bar{Y} + XY + \bar{Z}$

$\therefore$ For $\bar{D}$,

X \ YZ	00	01	11	10
0	0	0	0	
1	0	0	0	

$\bar{D} = \bar{Y} + Z$

$\therefore$ For $\bar{E}$,

X \ YZ	00	01	11	10
0	0	0	0	0
1	0	0	0	0

$\therefore \quad \bar{E} = 1$

For $\bar{F}$,

X \ YZ	00	01	11	10
0	0			0
1	0			0

$\therefore \quad \bar{F} = \bar{Z}$

13. Implement the following Boolean function using the multiplexers.

i) $F(W, X, Y, Z) = \Sigma(0, 1, 3, 4, 5, 8, 9, 15)$

ii) $F(X, Y) = \Sigma(1, 2, 3)$. Also, draw the internal circuit of the resulting multiplexers. [2012/S]

Solution:

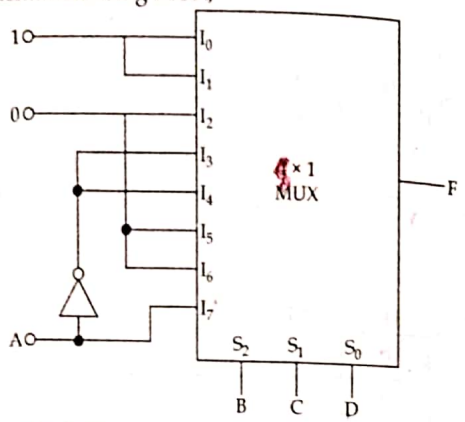
i) $F(W, X, Y, Z) = \Sigma(0, 1, 3, 4, 5, 8, 9, 15)$

Here, X, Y, Z are used as selection inputs. S_2, S_1 and S_0 for 8 × 1 MUX. For different inputs I_0, I_1, ..., I_7.

Implementation Table,

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	(0)	(1)	2	(3)	(4)	(5)	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	$\bar{A}$	$\bar{A}$	$\bar{A}$	0	A

Implementation using MUX,



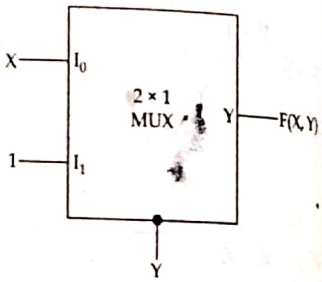
ii) $F(X, Y) = \Sigma(1, 2, 3)$

Solution:

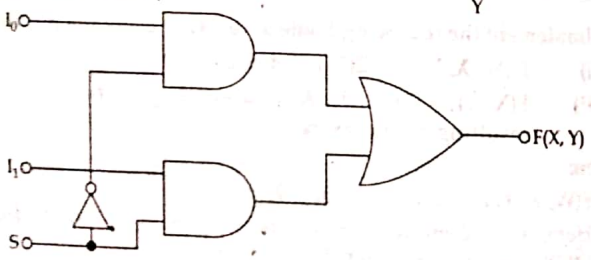
Implementation Table

	I_0	I_1
$\bar{X}$	0	(1)
X	(2)	(3)
	X	1

MUX implementation



Internal circuit of MUX,



14. **Implement the following function using 8 × 1 multiplexer. [2012/F]**

Solution:

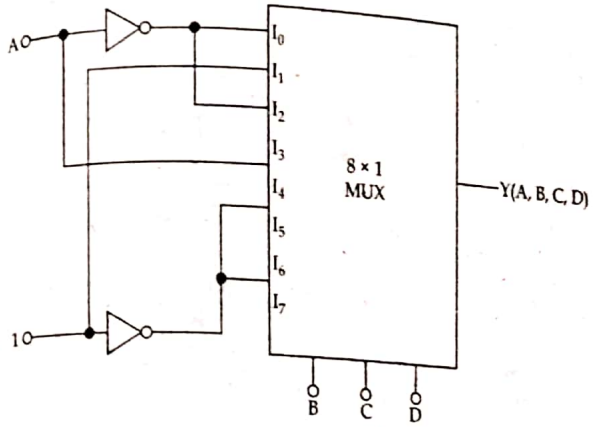
Given that;

$Y(A, B, C, D) = \Sigma(0, 1, 2, 5, 9, 11, 13, 15)$

Here, B, C and D are taken as selection lines S_2, S_1 and S_0 respectively and A as input for 8 × 1 MUX.

MUX Implementation Table;

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	(0)	(1)	(2)	3	4	(5)	6	7
A	8	(9)	10	(11)	12	(13)	14	(15)
	$\bar{A}$	1	$\bar{A}$	A	0	1	0	A



15. **Implement the following** **[2014/S, 2014/F, 2015/F, 2016/S]**

i) **$F(A, B, C) = \Sigma(1, 3, 5, 6)$ using MUX**

ii) **$F_1 = \Sigma(0, 2, 5)$, $F_2 = \Sigma(3, 4, 7)$, $F_3 = \Sigma(6, 7)$ using ROM**

Solution:

$F(A, B, C) = \Sigma(1, 3, 5, 6)$

We need 4 × 1 MUX and let B and C be connected to selection inputs S_1 and S_0.

MUX implementation Table;

	I_0	I_1	I_2	I_3
$\bar{A}$	0	(1)	2	(3)
A	4	(5)	(6)	7

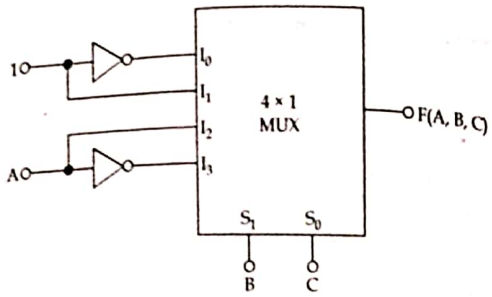


Figure: 4 to 1 line multiplexer circuit

ii) $F_1 = \Sigma(0, 2, 5)$, $F_2 = \Sigma(3, 4, 7)$, $F_3 = \Sigma(6, 7)$ since maximum value of minterm is 7, we take 3 input decoder in ROM. *i.e.*, size of four ROM is $2^3 \times 3$, as we have 3 functions. Let A, B and C be three inputs.

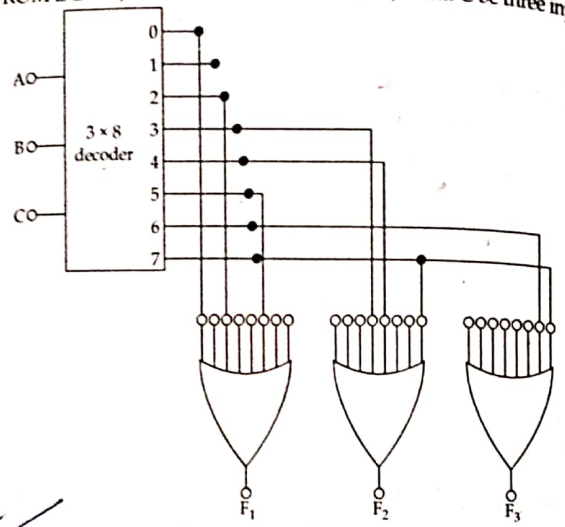


Figure: ROM implementation

16. Implement the following with appropriate MUX

i) F(A, B, C) = Σ (1, 3, 5, 6)

ii) F(A, B, C, D) = Σ(0, 1, 3, 4, 8, 9, 15) [2015/S, 2019/S(similar)]

Solution:

i) F(A, B, C) = Σ(1, 3, 5, 6)

See the Question number 15 (i).

ii) F(A, B, C, D) = Σ(0, 1, 3, 4, 8, 9, 15)

See the Question number 13 (i).

17. Implement the following three Boolean function with PLA. $F_1 = \Sigma(0, 1, 2, 4)$, $F_2 = \Sigma(0, 5, 6, 7)$ and $F_3 = \Sigma(0, 3, 5, 7)$. [2015/F]

Solution:

$F_1 = \Sigma(0, 1, 2, 4)$

$F_2 = \Sigma(0, 5, 6, 7)$

$F_3 = \Sigma(0, 3, 5, 7)$

Let X, Y, Z be three input variable of functions F_1, F_2 and F_3.

Using K-map to simplify each function and its complement:

For F_1

X \ YZ	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$\therefore F_1 = \bar{X}\bar{Y} + \bar{X}\bar{Z} + \bar{Y}\bar{Z}$

and, $\bar{F}_1 = XY + XZ + YZ$

For F_2

X \ YZ	00	01	11	10
0	1	0	0	0
1	0	1	1	1

$\therefore F_2 = \bar{X}\bar{Y}\bar{Z} + XZ + XY$

and, $\bar{F}_2 = X\bar{Y}\bar{Z} + \bar{X}Y + \bar{X}Z$

For F_3

X \ YZ	00	01	11	10
0	1	0	1	0
1	0	1	1	0

$\therefore F_3 = \bar{X}\bar{Y}\bar{Z} + XZ + YZ$

and, $\bar{F}_3 = \bar{X}\bar{Y}Z + X\bar{Z} + \bar{Y}Z$

From all possible combinations of F_1, F_2 and F_3 and their combinations, choosing $\bar{F}_1$, F_2 and F_3 give us minimum number of product terms *i.e.*, XY, XZ, YZ, $\bar{X}\bar{Y}\bar{Z}$.

Now, PLA program Table is;

	Product term	Inputs			Outputs			
		X	Y	Z	F_1	F_2	F_3	
1	XY	1	1	-	1	1	-	
2	XZ	-	-	1	1	1	1	
3	YZ	-	1	1	1	-	1	
4	$\bar{X}\bar{Y}\bar{Z}$	0	0	0	-	1	-	
					C	T	T	T/C

For each product term, the inputs are marked with 1, 0 or -. If a variable in product term appears in normal, complemented or is not present, we mark them with 1, 0 and - respectively. For output, the 1's represent that corresponding product term is present in that respective function and - represents that the product is not present in the function.

At the bottom, a T(true) output dictates that the fuse across output inverter remain intact, and a C (complement) specifies the corresponding fuse be blown.

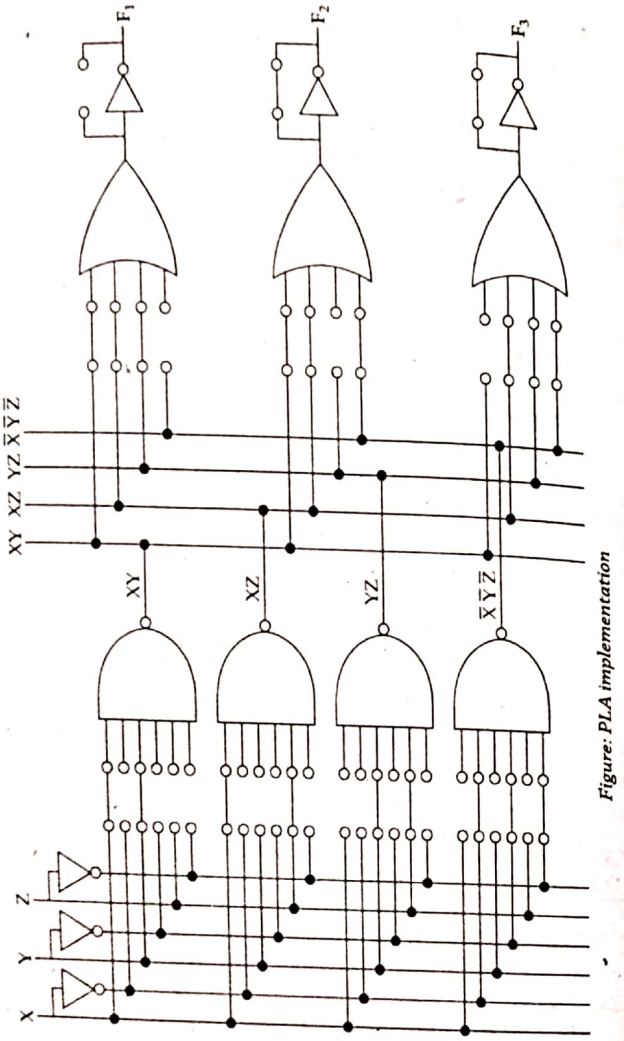


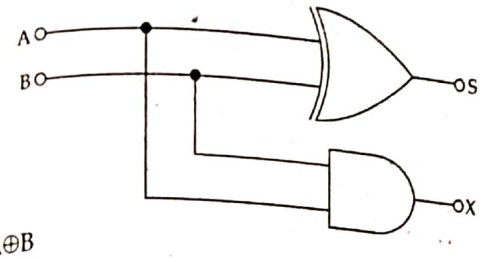
Figure: PLA implementation

18. Implement a full adder circuit with the help of two half adder circuit along with the truth table. **[2017/F]**

Solution:

A full adder can be implemented with two half adder and OR gate. Let A and B be two input variables and C be in carry and let S denote sum and X denote out carry.

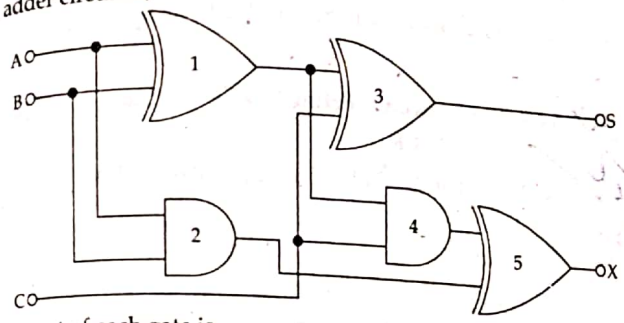
Half adder circuit is,



$S = A \oplus B$

$X = AB$

Full adder circuit is,



The output of each gate is,

$1 = A \oplus B$

$2 = A \cdot B$

$3 = (A \oplus B) \oplus C = A \oplus B \oplus C = S$

$4 = (A \oplus B) \cdot C = C(A\bar{B} + \bar{A}B) = A\bar{B}C + \bar{A}BC$

$5 = A\bar{B}C + \bar{A}BC + AB = X$

Let us verify the full adder circuit with truth table and derive expression for S and X for comparison.

Input			Output	
A	B	C	X	S
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

From truth table,

For S,

A \ BC	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

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

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

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

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

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

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

This shows that output from circuit and truth table are same.

i.e., $S = A \oplus B \oplus C$.

19. A combinational circuit is defined by the function. [2018/F]

$F_1 = (A, B, C) = \Sigma(3, 5, 6, 7)$

$F_2 = (A, B, C) = \Sigma(0, 2, 4, 7)$ Implement by using PLA.

Solution:

$F_1 = \Sigma(3, 5, 6, 7)$

$F_2 = \Sigma(0, 2, 4, 7)$

Simplifying F_1 and F_2 and its complement using k-map,

For, F_1

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$\therefore \quad F_1 = AB + AC + BC$

and, $\bar{F}_1 = \bar{B}\bar{C} + \bar{A}\bar{B} + \bar{A}\bar{C}$

For F_2

A \ BC	00	01	11	10
0	1	0	0	1
1	1	0	(1)	0

$\therefore \quad F_2 = \bar{B}\bar{C} + ABC + \bar{A}\bar{C}$

For minimum number of product terms, we take $\bar{F}_1$ and F_2. The PLA implement table is given below;

Product term	Input A	Input B	Input C	Output F_1	Output F_2	T/C
$\bar{B}\bar{C}$	-	0	0	1	1	
$\bar{A}\bar{C}$	0	-	0	1	1	
$\bar{A}\bar{B}$	0	0	-	1	-	
ABC	1	1	1	-	1	
				C	T	

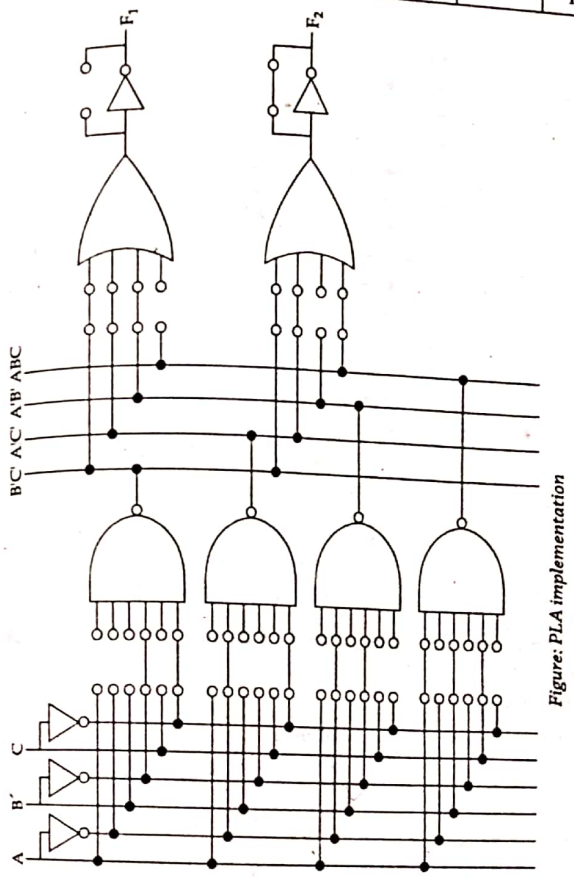


Figure: PLA implementation

20. Implement the following Boolean function using 16:1 multiplexer

$F(A, B, C, D, E) = \Sigma m(2, 4, 5, 7, 10, 14, 15, 16, 17, 25, 26, 30, 31)$ [2018/S]

Solution:

$F(A, B, C, D, E) = \Sigma m(2, 4, 5, 7, 10, 14, 15, 16, 17, 25, 26, 30, 31)$

Let, B, C, D, E be connected to 4 selection inputs S_3, S_2, S_1, S_0 of 16 × 1 MUX. For 16 input lines I_0, I_1, I_2, ..., I_{15} the MUX implementation table is;

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
$\bar{A}$	0	1	(2)	3	(4)	(5)	6	(7)	8	9	(10)	11	12	13	(14)	(15)
A	(16)	(17)	18	19	20	21	22	23	24	(25)	(26)	27	28	29	(30)	(31)
	A	A	$\bar{A}$	0	$\bar{A}$	$\bar{A}$	0	$\bar{A}$	0	A	1	0	0	0	1	1

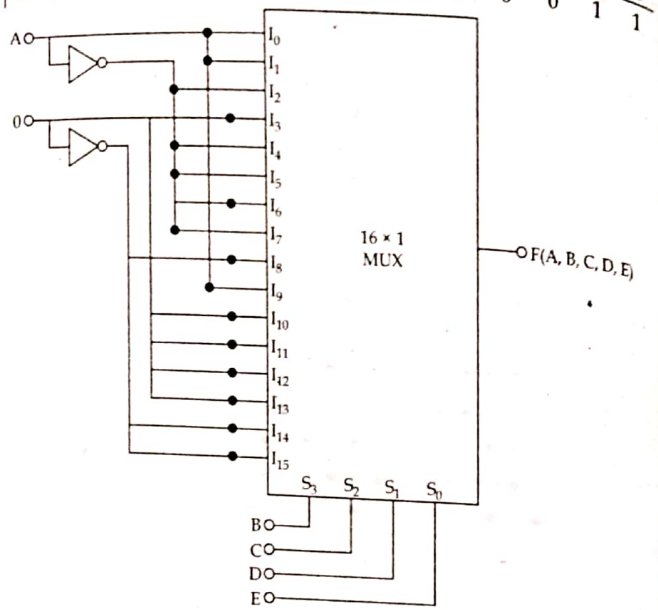


Figure: 16 × 1 MUX implementation

21. Implement a full adder using 3 × 8 decoder and OR gates. [2011/F]

Solution:

Full adder Truth Table:

Input variables			Outputs	
X	A	B	Sum (S)	Carry (C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = \bar{X}\bar{A}B + \bar{X}A\bar{B} + X\bar{A}\bar{B} + XAB = \Sigma(1, 2, 4, 7)$

and, $C = \Sigma(3, 5, 6, 7)$

Since there are three inputs and a total of eight minterms, we need a 3 to 8 line decoder. The implementation is shown in figure below. The decoder generates the eight minterms for X, A, B. The OR gate for output S forms the sum of minterms 1, 2, 4 and 7. The OR gate for output C forms the sum of minterms 3, 5, 6 and 7.

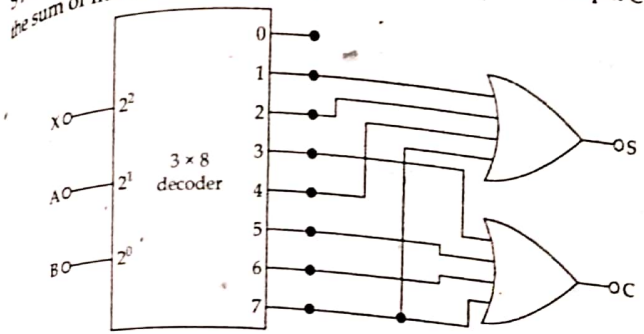


Figure: Implementation of full adder with a decoder

22. Design a full subtractor using two 4 × 1 MUX. [2012/F]

Solution:

Let, X and Y be two inputs-minuend and subtrahend and Z be previous borrow. let B and D denote outputs borrow and difference respectively.

Truth Table

X	Y	Z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Let, Y and Z be connected to two selection inputs S_1, S_0 of 4 × 1 MUX. The MUX implementation table for inputs of 4 × 1 MUX is given below.

For B,

	I_0	I_1	I_2	I_3
$\bar{X}$	0	(1)	(2)	(3)
X	4	5	6	(7)
	0	$\bar{X}$	$\bar{X}$	1

For D,

	I_0	I_1	I_2	I_3
$\overline{X}$	0	(1)	(2)	3
X	(4)	5	6	(7)
	X	$\overline{X}$	$\overline{X}$	X

MUX implement diagram:

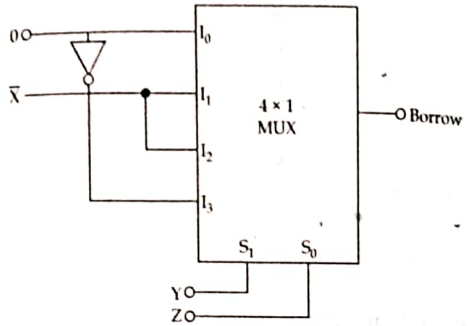


Figure: For Borrow

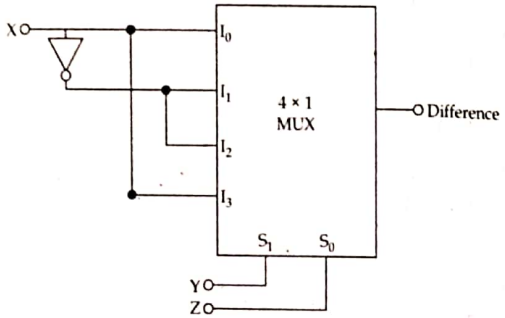


Figure: For difference

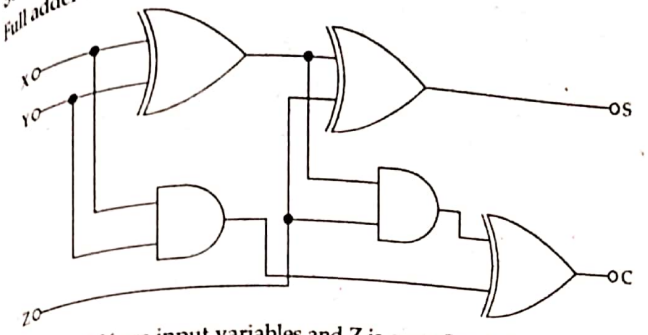
23. Define multiplexer and De-multiplexer construct 8 × 1 MUX using 4 × 1 MUX and explain with the truth table. [2019/F]

Solution: See the topic 6.5.1 and 6.5.2.

24. Show how a full adder can be converted to a full subtractor with the addition of one inverter circuit. [2014/S]

Solution:

Full adder circuit is shown below;



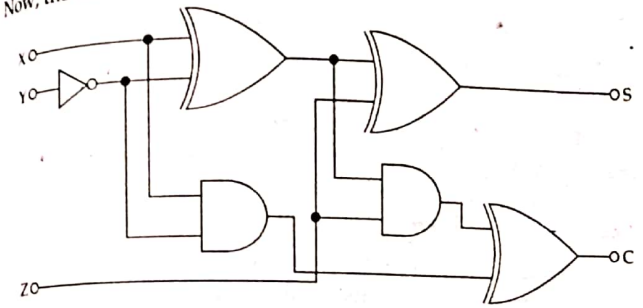
Here, X and Y are input variables and Z is carry. S and C are sum and out carry respectively.

When Z = 0, operation is X + Y

When Z = 1 operation is X + Y + 1

To convert this to full subtractor, add an inverter to Y,

Now, the circuit becomes,



Here S and C represent difference and borrow respectively.

When Z = 1, the operation is $X + \overline{Y} + 1$ which is a 2's complement subtraction between X and Y *i.e.*, $X - Y$.

When Z = 0 operation is $X + \overline{Y} = [(X + \overline{Y} + 1) - 1]$ which is also 2's complement subtraction with borrow.

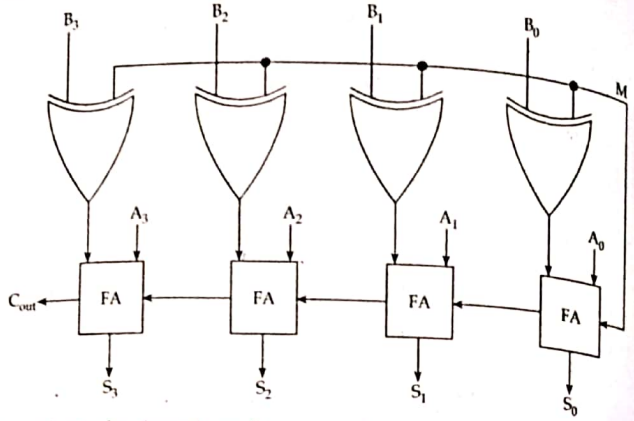
Hence, in this way a full adder can be converted to full subtractor using an inverter circuit.

25. Design a circuit for 4-bit full adder subtractor. [2015/S]

Solution:

The operation of both addition and subtraction can be performed by a one common binary adder. Such binary circuit can be designed by adding an EX-OR gate with each full adder as shown in figure below.

The mode input control line M is connected with carry input of the least significant bit of the full adder. This control line decides the type of operations whether addition or subtraction.



When M = 1, the circuit is a subtractor and when M = 0 the circuit becomes adder. The EX-OR gate consists of two inputs to which one is connected to the B and the other to input M. When M = 0, B⊕0 produces B. then full adders add the B with A with carry input zero and hence an addition operation is performed.

When M = 1, B⊕0 produces B complement and also carry input is 1. Hence the complemented B inputs are added to A and 1 is added through the input carry, nothing but a 2's complement operation. Hence subtraction operation is performed.

26. Design a 4-bit arithmetic unit that performs addition when mode control bit is 0 and subtraction when mode control bit is 1. [2019/F]

Solution: See the question number 25.

CHAPTER 7

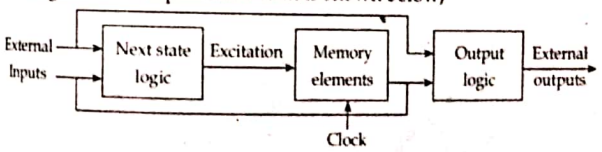
SEQUENTIAL LOGIC

*****

*****

INTRODUCTION

The logic circuits whose outputs at any instant of time depend on the present inputs as well as on the past outputs are called sequential circuits. In sequential circuits, the output signals are fed back to the input side. A block diagram of a sequential circuit is shown below;



It consists of combinational circuits, which accept digital signals from external inputs and from outputs of memory elements and generate signals for external outputs and for inputs to memory elements, referred to as excitation.

A memory element is a mediun in which one bit of information (0 or 1) can be stored or retained until necessary and there-after its contents can be replaced by a new value. The contents of memory elements can be changed by the outputs of combinational circuits that are connected to its input.

Combinational circuit are often faster than sequential circuit since the combinational circuits do not require memory elements whereas the sequential circuit needs memory elements to perform its operations in sequence. Sequential circuits are broadly classified into two main categories, known as synchronous or clocked and asynchronous or unlocked sequential circuits, depending on the timing of their signals.

A sequential circuit whose behaviour can be defined from the knowledge of its signal at discrete instants of time is referred to as a synchronous sequential circuit. In these systems, the memory elements are affected only at discrete instants of time. The synchronization is achieved by a timing device known as a system clock which generates a periodic train of clock pulses. The outputs are affected only with the application of clock pulses. The rate at which the master clock generates pulses must be slow enough to permit the slowest circuit to respond. This limits the speed of all circuits. Synchronous circuits have gained considerable domination and wide popularity.

A sequential circuit whose behaviour depends upon the sequence in which the input signals change is referred to as an asynchronous sequential circuit. The output will be affected whenever the input changes. The commonly used memory elements in these circuits are time delay devices. There is no need to wait for a clock pulse. Therefore in general, asynchronous circuits are faster than synchronous sequential circuits. However, in an asynchronous circuits, events are allowed to occur without any synchronization. And in such a case, the system becomes unstable. Since the designs of asynchronous circuits are more tedious and difficult, their uses are rather limited. The memory elements used in sequential circuits are flip flops which are capable of storing binary information.

7.1 FLIP FLOP AND THEIR TYPES

A flip flop circuit can maintain a binary state in definitely (as long as power is delivered to the circuit) until directed by an input signal to switch states. The basic 1-bit digital memory circuit is known as a flip flop. It can have only two states, either the 1 state or the 0 state. A flip-flop is also known as a bistable multi vibrator. Flip flops can be obtained by

using NAND or NOR gates. It has one or more inputs and two outputs. The two outputs are complementary to each other.

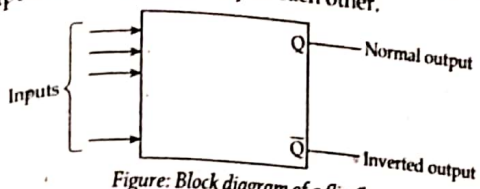


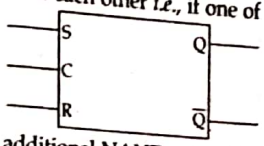
Figure: Block diagram of a flip flop

Types of Flip-Flops

There are different types of flip flops depending on how their inputs and clock pulses causes transition between two states.

7.1.1 S-R (Set-Reset) Flip-Flop

An S-R flip flop has two inputs named set (S) and Reset (R) and two outputs Q and $\overline{Q}$. The outputs are complement of each other *i.e.*, if one of the outputs is 0 then other should be 1. This can be implemented using NAND or NOR gates. The block diagram of an S-R flip flop is shown in figure.



It consists of a basic flip flop circuit and two additional NAND gates. The pulse input acts as an enable signal for the other two inputs. The outputs of NAND gates 3 and 4 stay at a logic 1 level as long s the clock pulse (CP) remains at 0. This is the quiescent condition for the basic flip-flop. When the pulse input goes to 1, information from the S or R input is allowed to reach the output. The set state is reached with S = 1, R = 0 and CP = 1. This causes the output of gate 3 to go to 0, the output of gate 4 to remain at 1, and the output of the flip flop at Q to go to 1. To change to the reset state, the input must be S = 0, R = 1 and CP = 1. In either case, when CP returns to 0, the circuit remains in its previous state when CP = 1 and both the S and R inputs are equal to 0, the state of the circuit does not change.

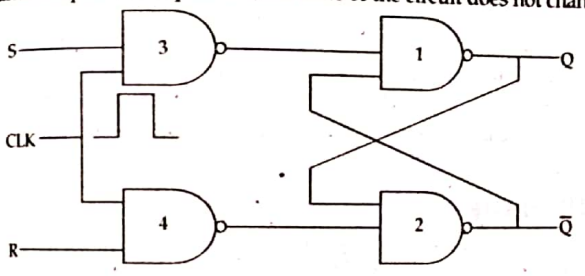


Figure: Logic diagram

An indeterminate condition occurs when CP = 1 and both S and R are equal to 1. This condition places 0's in the outputs of gates 3 and 4 and 1's in the both outputs Q and $\overline{Q}$. When the CP input goes back to 0 (while S and R are maintained at 1), it is not possible to determine the next state, as

it depends on whether the output of gate 3 or gate 4 goes to 1 first. This indeterminate condition makes the circuit difficult to manage. The characteristics table of the flip flop is shown below;

Q	S	R	Q(t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indetermined
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate

This table shows the operation of the flip flop in tabular form. Q is an abbreviation of Q(t) and stands for the binary states of the flip-flop before the application of a clock pulse, referred to as the present state. The S and R columns give the possible values of the inputs, and Q(t + 1) is the state of the flip flop after the application of a single pulse, referred to as the next state. Note that the CP input is not included in the characteristic table. The table must be interpreted as follows. Given the present state Q and the inputs S and R, the application of a single pulse in the CP input causes the flip flop to go to the next state, Q(t + 1)

The characteristic equation of the flip-flop is derived in the map.

	$\bar{S}\bar{R}$	$\bar{S}R$	SR	$S\bar{R}$
$\bar{Q}$			X	1
Q	1		X	1

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

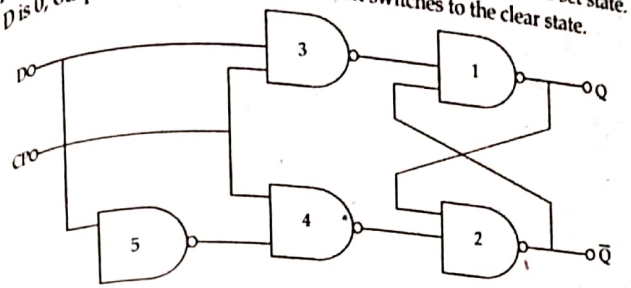
$SR = 0$

The equation specifies the value of the next state as a function of the present state and the inputs. The characteristics equation is an algebraic expression or the binary information of the characteristics input. The two indeterminate states are marked with don't care X's in the map, since they may result in either 1 or 0. However, the reaction SR = 0 must be included as part of the characteristics equation o specify that both S and R cannot equal to 1 simultaneously.

7.1.2 D Flip-Flop

One way to eliminate the undesirable condition of the indeterminate state in the RS flip flop is to ensure that input S and R are never equal to 1 at the same time. This is done in the D lip flop. The D flip- flop has only two inputs: D and CP. The D input goes directly to the S input and its complement is applied to the R input. As long as the pulse input is at 0, the outputs of gates 3 and 4 are at the 1 level and the circuit cannot change state regardless of the value of D. The D input is sampled when

CP = 1. If D = 1, the Q output goes to 1, placing the circuit in the set state. If D is 0, output Q goes to 0 and the circuit switches to the clear state.



The D flip flop receives the designation from its ability to hold data into its internal storage. This type of flip flop is sometimes called a gated D-latch. The CP input is often given the designation G (for gate) to indicate that this input enables the gated latch to make possible data entry into the circuit. The binary information present at the data input of the D flip flop is transferred to the Q output when the CP input is enabled. The output follow the data input as long as the pulse remains in its 1 state. When the pulse goes to 0, the binary information that was present at the data input at the time the pulse transition occurred is retained at the Q output until the pulse input is enabled again.

The characteristics table for the D flip flop is shown below;

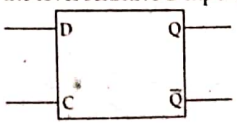
Q	D	Q(t + 1)
0	0	0
0	1	1
1	0	0
1	1	1

It shows that the next state of the flip flop is dependent of the present state since Q(t + 1) is equal to input D whether Q is equal to 0 or 1. This means that an input pulse will transfer the value of input D into the output of the flip flop independent of the value of the output before the pulse was applied. The characteristics equation shows clearly that Q(t + 1) is equal to D.

	$\bar{D}$	D
$\bar{Q}$		1
Q		1

$Q(t + 1) = D$

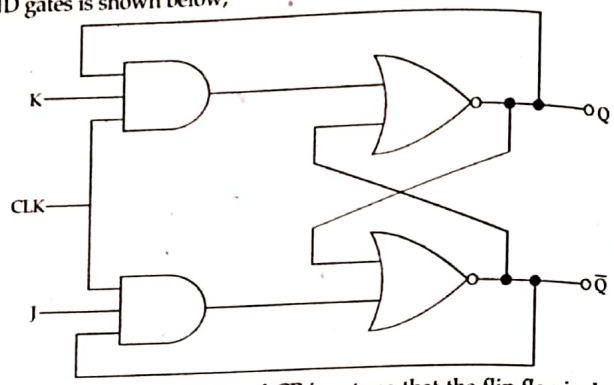
The graphic symbol for the level sensitive D flip flop is shown below.



7.1.3 JK Flip Flop

A JK flip flop is a refinement of the RS flip flop in that the indeterminate state of the RS type is defined in the JK type. Inputs J and K behave like inputs S and R to set and clear the flip flop respectively. The input marked J is for set and the input marked K is for reset. When both inputs J and K are equal to 1, the flip flop switches to its complement state, that is, if Q = 1, it switches to Q = 0 and vice versa.

A JK flip flop constructed with two cross coupled NOR gates and two AND gates is shown below;



Output Q is ANDed with K and CP inputs so that the flip flop is cleared during a clock pulse only if Q was previously 1. Similarly, output $\overline{Q}$ is ANDed with J and CP inputs so that the flip flop is set with a clock pulse only when $\overline{Q}$ was previously 1. When both J and K are 1, the input pulse is transmitted through one AND gate only the one whose input is connected to the flip flop output that is presently equal to 1. Thus, if Q = 1, the output of the upper AND gate becomes 1 upon application of the clock pulse and the flip flop is cleared. If $\overline{Q}$ = 1, the output of the lower AND gate becomes 1 and the flip flop is set. In either case, the output state of the flip flop is complemented. The behaviour of the JK flip flop is demonstrated in the characteristic table.

Q	J	K	Q(t +1)
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	1
1	1	1	0

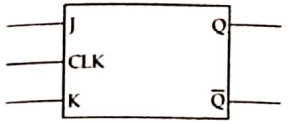
It is very important to realize that because of the feedback connection in the JK flip flop, a CP pulse that remains in the 1 state while both J and K

are equal to 1 will cause the output to complement again and repeat complementing until the pulse goes back to 0. To avoid this undesirable operation, the clock pulse must have a time duration that is shorter than the propagation delay time of the flip flop. This is a restrictive requirement, since the operation of the circuit depends on the width of pulse. The characteristic equation is shown below;

Q	$\overline{J}\overline{K}$	$\overline{J}K$	JK	$J\overline{K}$
$\overline{Q}$			1	1
Q	1			1

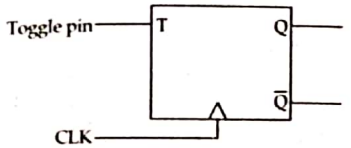
$Q(t + 1) = J\overline{Q} + \overline{K}Q$

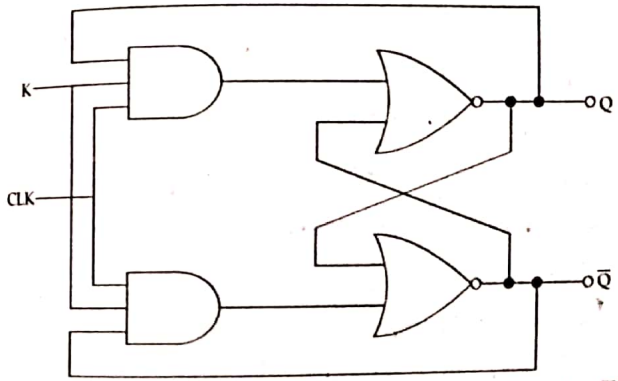
The graphic symbol for the JK flip flops is,



7.1.4 T Flip Flop

The T flip flop is a single input version of the JK flip flop. The T flip flop is obtained from the JK flip flop when both inputs are tied together. The designation T comes from the ability of the flip flop to toggle or complement, its state. Regardless of the present state, the flip flop complements its output when the clock pulse occurs while input T is 1.





The characteristic table and characteristic equation shows that when T = 0, Q (t +1) = Q, that is the next state is the same as the present state and no change occurs.

The characteristics table for the D flip flop is shown below;

Q	D	Q(t + 1)
0	0	0
0	1	1
1	0	1
1	1	0

Figure: Characteristic table

	$\bar{T}$	T
$\bar{Q}$		1
Q	1	

$Q(t + 1) = T\bar{Q} + \bar{T}Q$

Figure: Characteristic equation

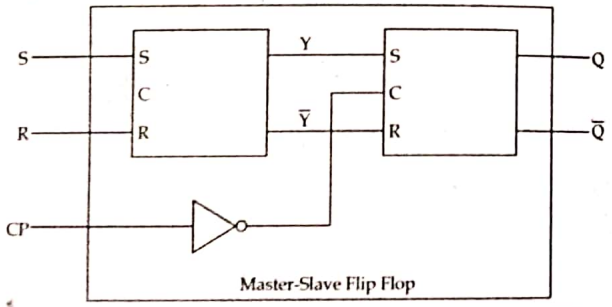
When T = 1, then $Q(t + 1) = \bar{Q}$ and the state of the flip flop is complemented.

Applications of Flips Flops

i) It can be used as a memory element.
ii) It can be used to eliminate key debounce.
iii) It can be used as a delay element.
iv) It is used as a basic building block in sequential circuits such as counters and registers.

MASTER-SLAVE FLIP FLOP

A master-slave flip flop is constructed from two separate flip flops. One circuit serves as a master and the other as a slave and the overall circuit is referred to as a master-slave flip flop. The logic diagram of an RS master-slave flip flop is shown below;

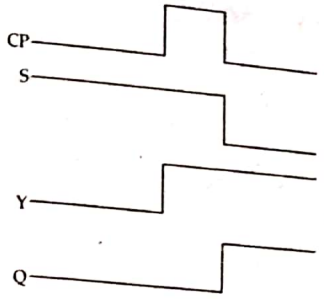


Master-Slave Flip Flop

It consists of master flip flop, a slave flip flop and an inverter. When clock pulse CP is 0, the output of the inverter is 1. Since the clock input of the slave is 1, the flip flop is enabled and output Q is equal to Y, while $\bar{Q}$ is equal to $\bar{Y}$. The master flip flop is disable because CP = 0. When the pulse becomes 1, the information then at the external R and S input is

transmitted to the master flip flop. The slave flip flop, however, is isolated as long as the pulse is at its 1 level, because the output of the inverter is 0. When the pulse returns to 0, the master flip flop is isolated, which prevents the external inputs from affecting it. The slave flip flop then goes to the same state as the master flip flop.

The timing relationship shown in figure illustrate the sequence of events that occur in a master slave flip flop.



Assume that the flip flop is in the clear state prior to the occurrence of a pulse, so that Y = 0 and Q = 0. The input conditions are S = 1, R = 0 and the next pulse should change the flip to the set state with Q = 1. During the pulse transition from 0 to 1, the master flip flop is set and changes Y to 1. The slave flip is not affected because its CP input is 0. Since the master flip flop is an internal circuit, its change of state is not noticeable in the outputs Q and $\bar{Q}$. When the pulse returns to 0, the information from the master is allowed to pass through to the slave, making the external output Q = 1. Note that the external S input can be changed at the same time that the pulse goes through its negative edge transition. This is because once the CP reaches 0, the master flip flop is disabled and its R and S inputs have no influence until the next clock pulse occurs. Thus, in a master slave flip flop it is possible to switch the output of the flip flop and its input information with the same clock pulse. It must be realized that the S input could come from the output of another master-slave flip flop that was switched with the same clock pulse.

The behavior of the master slave flip flop just described dictates that the state changes in all flip flops coincide with the negative edge transition of the pulse. However, some IC master slave flip flop changes output states in the positive edge transition of clock pulses. This happens in flip flops that have an additional inverter between the CP terminal and the input of the master. Such flip flops are triggered with negative pulses, so that the negative edge of the pulse affects the master and the positive edge effects the slave and the output terminals.

The master-slave flip flop combination can be constructed for any type of flip flop by adding a compact clocked RS flip flop with an inverted clock to form the slave.

Applications of Shift Registers

Shift registers are widely used in almost every sphere of a digital logic system. Shift registers are basically used to count number of pulses entering into a system as follows; A(t + 1) = B(t + 1) = ... switched tail counter. As ring counter it can generate

various control signals in a sequential manner. Shift register can also generate a prescribed sequence repetitively or detect a particular sequence from data input. It can also help in reduction of hardware by converting parallel data feed to serial one.

7.3 COUNTERS

A counter is a register capable of counting the number of clock pulses arriving at its clock input. Count represents the number of clock pulses arrived. On arrival of each clock pulse, the counter is incremented by one. In case of down counter, it is decremented by one.

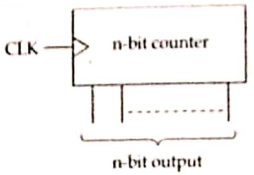


Figure: Positive edge triggered n-bit counter

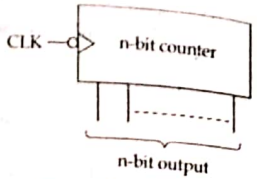


Figure: Negative edge triggered n-bit counter

Figure: Logic symbol of counter

External clock is applied to the clock input of the counter. The counter can be positive edge triggered or negative edge triggered. The n-bit binary counters has n flip-flops and it has 2^n distinct states of outputs. *For example*; 2-bit counter has 2 flip flops and it has 4 (2^2) distinct states 00, 01, 10 and 11. The maximum count that the binary counter can count is 2^{n-1}. *For example*; in 2-bit binary counter, the maximum count is $2^2 - 1 = 3$ (11 in binary). After reaching the maximum count the counter resets to zero (0) on arrival of the next clock pulse and it starts counting again.

7.3 ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS

The behaviour of a sequential circuit is determined from the inputs, the outputs and the state of its flip flops. The outputs and the next state are both a function of the inputs and the present state. The analysis of a sequential circuit consists of obtaining a table or a diagram for the time sequence of inputs, outputs and internal states. It is also possible to write Boolean expressions that describe the behaviour of the sequential circuit. However, these expressions must include the necessary time sequence, either directly or indirectly.

A logic diagram is recognized as a clocked sequential circuit if it includes flip flops. The flops may be any type and the logic diagram may or may not include combinational circuit gates.

A. State Table

The time sequence of inputs, outputs and flip flops states can be enumerated in a state table. The table consists four sections labelled present state, input, next state and output. The present state section

the states of flip flops A and B at any given time t. The input section gives a value of x for each possible present state. The next state section shows the states of the flip flops one clock period later at time t + 1. The output section gives the value of y for each present state. The derivation of a state table consists of first listing all possible binary combinations of present state and inputs.

In other configuration, the state table has only three sections: Present state, next state and output. The input conditions are enumerated under the next-state and output section.

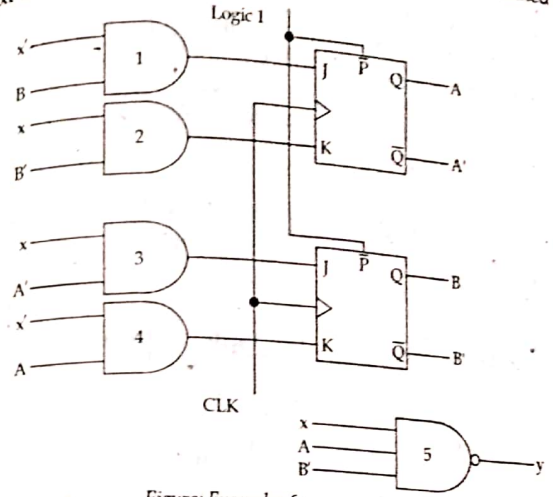


Figure: Example of a sequential circuit

The circuit consists of two D flip flops A and B, an input x and an output y. Since the D inputs determine the flip flop's next state, it is possible to write a set of next state equations for the circuit:

$$A(t + 1) = A(t)\,x(t) + B(t)\,x(t)$$

$$B(t + 1) = \bar{A}(t)\,x(t)$$

A state equation is an algebraic expression that specifies the condition for a flip flop state transition. The left side of the equation denotes the next state of the flip flop and the right side of the equation is a Boolean expression that specifies the present state and input conditions that make the next state equal to 1. Since all the variables in the Boolean expression are a function of the present state, we can omit the designation (t) after each variable for convenience. The previous equations can be expressed in more compact form as follows;

$$A(t + 1) = Ax + Bx$$

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

The present state value of the output can be expressed algebracially as follows;

$$y(t) = [A(t) + B(t)]\,\bar{x}(t)$$

Removing the symbol (t) for the present state, we obtain Boolean function:

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

The derivation of a state table consists of first listing all possible binary combinations of present state and inputs. In this case, we have eight binary combinations from 000 to 111. The next state values are then determined from the logic diagram or from the state equations. The next state of flip flop A must satisfy the state equation

$$A(t + 1) = Ax + Bx$$

The next section in the state table under column A has three 1's where the present state and input value satisfy the conditions that the present state of A and input x are both equal to 1 or the present state of B and input x are both equal to 1. Similarly the next state of flip flop B is derived from the state equation

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

It is equal to 1 when the present state of A is 0 and input x is equal to 1. The output column is derived from the output equation

$$y = A\bar{x} + Bx$$

State table:

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Another configuration of state table:

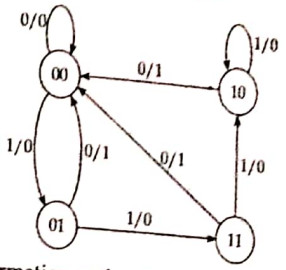
Present state		Next state				Output	
		x = 0		x = 1		x = 0	x = 1
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

The state table of any sequential circuit with D-type flip flops is obtained by the same procedure obtained in above example. In general, a sequential circuit with m flip flops and n inputs needs 2^{m+n} rows in the state table. The binary numbers from 0 through $2^{m+n} - 1$ are listed under the present state and input columns. The next state section has m

columns, one for each flip flop. The binary values for the next state are derived directly from the state equations. The output section has as many columns as there are output variables. Its binary value is derived from the circuit or from the Boolean function in the same manner as in a truth table. In general, a sequential circuit may have two or more inputs or outputs.

State Diagram

The information available in a state table can be represented graphically in a state diagram. In this type of diagram, a state is represented by a circle and the transition between states is indicated by directed lines connecting the circle. The state diagram of the sequential circuit of above figure is shown in figure.



The state diagram provides the same information as the state table and is obtained directly from table. The binary number inside each circle identifies the state of the flip flops. The directed lines are labeled with two binary numbers separated by a slash. The input value during the present state is labeled first and the number after the slash gives the output during the present state. *For example*, the direct line from state 00 to 01 is labeled 1/0 meaning that when the sequential circuit is in the present state 00 and input is 1, the outputs 0. After the clock transition, the circuit goes to the next state 01. The same clock transistion may change the input value. If the input changes to 0, then the output becomes 1, but if the input remains at 1, the output stays at 0. This information is obtained from the state diagram along the two directed lines emanating from the circle representing state 01. A directed line connecting a circle with itself indicates that no change of state occurs. There is no difference between a state table and a state diagram except in the manner of representation. The state table is easier to derive from a given logic diagram and the state diagram follows directly from the state table. The state diagram gives a pictorial view of state transitions and is the form suitable for human interpretation of the circuit operation.

STATE REDUCTION AND ASSIGNMENT

The analysis of sequential circuits starts from a circuit diagram and culminates in a state table or diagram. The design of a sequential circuit starts from a set of specification and culminates in a logic diagram.

1 State Reduction

Any design process must consider the problem of minimizing the cost of the final circuit. The two most obvious cost reductions are reductions in the number of flip flops and the number of gates. Because these two items seem the most obvious, they have been extensively studied and investigated. The reduction of the number of flip flops in a sequential circuit is referred

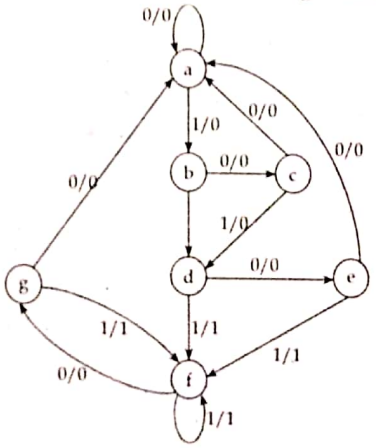
to as the state reduction problem. State reduction algorithms are concerned with procedures for reducing the number of states in a state table while keeping the external input-output requirement unchanged. Since m flip flops produce 2^m states, a reduction in the number of states may (or may not) result in a reduction in the number of flip flops. An unpredictable effect in reducing the number of flip flops is that sometimes the equivalent circuit (with less flip flops) may require more combinational gates.

B. State Assignment

The cost of the combinational circuit part of sequential circuit can be reduced by using the known simplification methods for combinational circuits. However, there is another factor, known as the state assignment problem that comes into play in minimizing the combinational gates. State assignment procedures are concerned with methods for assigning binary values to states in such a way as to reduce the cost of the combinational circuit that drives the flip-flops. This is particularly helpful when a sequential circuit is viewed from its external input-output terminals. Such a circuit may follow a sequence of internal gates, but the binary values of the individual states may be of no consequence as long as the circuit produces the required sequence of outputs for any given sequence in inputs. This does not apply to circuits whose external output are taken directly from flip flops with binary sequences fully specified. Various procedures have been suggested that lead to a particular binary assignment from the many available. The most common criterion is that the chosen assignment should result in a simple combinational circuit for the flip flop inputs. However to date, there are no state assignment procedures that guarantee a minimal cost combinational circuit. State assignment is one of the challenging problems of switching theory.

State Reduction and Assignment Example

Consider the input sequence 01010110100 starting from the initial state a



Complete the sequence to get following:

State	a	a	b	c	d	e	f	f	g	f	g	a
Input	0	1	0	1	0	1	1	0	1	0	0	
Output	0	0	0	0	0	1	1	0	1	0	0	

state e and g both go to states a, and f and have outputs of 0 and 1 for x = 0 and x = 1 respectively. States g and e are equivalent and one of these states can be removed. States f and d are also equivalent so state f can be removed and replaced by d.

State	a	a	b	c	d	e	d	d	e	d	e	a
Input	0	1	0	1	0	1	1	0	1	0	0	
Output	0	0	0	0	0	1	1	0	1	0	0	

Original state table:

Present state	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

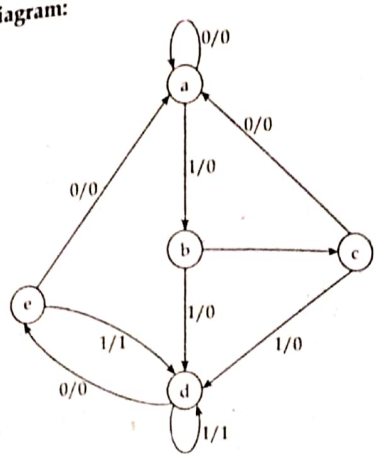
Reducing the state table:

Present State	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

Reduced State Table

Present State	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Reduced state diagram:



Reduced table with Binary assignment 1

Present state		Next state		Output	
		x = 0	x = 1	x = 0	x = 1
a	000	000	001	0	0
b	001	010	011	0	0
c	010	000	011	0	0
d	011	100	011	0	1
e	100	000	011	0	1

7.5 TRIGGERING OF FLIP FLOPS

The state of a flip flop is switched by a momentary change in the input signal. This momentary change is called a trigger and the transistion it causes is said to trigger the flip flop. As seen from the block diagram of sequential logic, a sequential circuit has a feedback path between the combinational circuit and the memory elements.

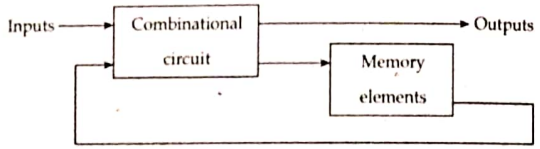


Figure: Block diagram of a sequential circuit

This path can produce instability if the outputs of memory elements (flip flops) are changing while the outputs of the combinational circuit that go to flip flop inputs are being sampled by the clock changing until the pulse input has returned to 0. To ensure such an operation, a flip flop must have a signal propagation delay from input to output in excess of the pulse duration. This delay is usually very difficult to control if the designer

depends entirely on the propagation delay of logic gates. One way of ensuring the proper delay is to include within the flip-flop circuit a physical delay unit having a delay equal to or greater than the pulse duration. A better way to solve the feedback timing problem is to make the flip flop sensitive to the pulse transistion rather than the pulse duration. A clock pulse may be either positive or negative. A positive clock source remains at 0 during the interval between pulses and goes to 1 during the occurrence of pulse. The pulse goes through two signal transistions. From 0 to 1 and the return from 1 to 0.

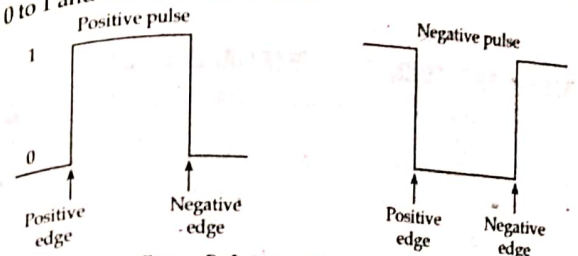


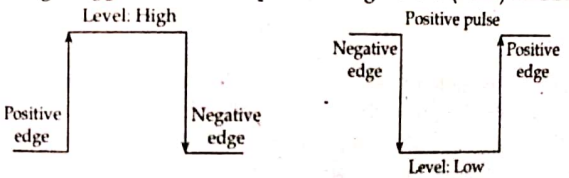
Figure: Definition of clock pulse transistion

As shown in figure, the positive transistion is defined as the positive edge and the negative transistion as the negative edge. This definition applies also to negative pulses.

One way to make the flip flop respond only to a pulse transition is to use capacitive coupling. In this configuration, an RC (resistor-capacitor) circuit is inserted in the clock input of the flip flop. This circuit generates a spike in response to a momentary change of input signal. A positive edge emerges from such a circuit with a positive spike and a negative edge emerges with a negative spike. Edge triggering is achieved by designing the flip flop to neglect one spike and trigger on the occurrence of the other spike. Another way to achieve edge triggering is to use a mater-slave or edge triggered flip-flop.

7.6 EDGE TRIGGERED DEVICE

A circuit whose outputs change on the transition of a signal is called an edge triggered device. Master slave circuits can be used for this purpose. Positive edge triggered device respond to a low to high (0 to 1) transition and negative edge-triggered device respond to a high-to-low (1 to 0) transistion.



Different type of flip flops are designed to respond either to positive edge or to a negative edge of a clock pulse. These flip flops are known as edge

triggered clocked flip flops. Positive edge triggering is schematically indicated by a small angle bracket on the clock input to the flip flop. On the other hand, the negative edge triggering is indicated schematically by a small circle and angle bracket on the clock input. Flip flops that are not edge triggered are called level triggered flip flops. They respond to their inputs while the clock signal is at high level and retain their output values after the level changes. If there were an inversion circle at the clock input, the device would respond during a low level instead. Level triggered operation is described as asynchronous since the output is not synchronized by a distinct trigger.

7.7 DESIGN PROCEDURE OF SEQUENTIAL CIRCUITS

The design of a sequential circuit follow certain steps. The steps may be listed as follows;

a) The word description of a circuit may be given accompanied with a state diagram or timing diagram or other pertinent information.
b) Then from the given state diagram the state table has to be prepared.
c) If the state reduction mechanism is possible, then the numbers of states may be reduced.
d) After state reduction, assign binary values to the states if the states contain letter symbols.
e) Then the number of flip flops required is to be determined. Each flip flop is assigned a letter symbol.
f) Then the choice has to be made regarding the type of flip flop to be used.
g) With the help of state table and the flip flop excitation and the output tables have to be determined.
h) Then using some simplification technique, *e.g.*, Karnaugh map or some other method, the circuit output functions and the flip flop input functions have to be determined.
i) Then the logic diagram has to be drawn.

Although certain steps have been specified for designing the sequential circuit, the procedure can be shortened with experience. A sequential circuit is made up of flip flops and combinational logic. One of the most important parts is the choice of flip flop. From the excitation table of different flip flops, we see that the J-K flip flop excitation table contains the maximum number of don't care conditions. Hence for designing any sequential circuit, it will be most simplified if the circuit is designed with J-K flip flop.

Any design process must consider the problem of minimizing the cost of the final circuit. The most obvious cost reductions are reductions in the number of flip flops and the number of gates. The reduction of the number of flip flops in a sequential circuit is referred to as the state reduction. Since m flip flops produce 2m states, a reduction in the number of [illegible] may or may not result in a reduction of the numbers of flip flops.

state reduction algorithms are concerned with procedures for reducing the number of states in a state table while keeping the external input-output requirements unchanged.

7.8 EXCITATION TABLE OF A FLIP-FLOP

The truth table of a flip flop is also referred to as the characteristic table of a flip-flop, since this table refers to the operational characteristics of the flip flop. But in designing sequential circuits, we often face situations where the present state and the next state of the flip-flop is specified and we have to find out the input conditions that must prevail for the desired output condition. By present and next states we mean to say the conditions before and after the clock pulse respectively. A table below gives the excitation table for S-R, D, J-K and T flip-flops. These conditions are derived from the corresponding characteristic tables of the flip-flops.

Present state	Next state	S-R FF		DFF	J-K FF		T-FF
Q_n	$Q_n + 1$	S_n	R_n	D_n	J_n	K_n	T_n
0	0	0	X	0	0	X	0
0	1	1	0	1	1	X	1
1	0	0	1	0	X	1	1
1	1	X	0	1	X	0	0

7.9 INTER CONVERSION OF FLIP-FLOPS

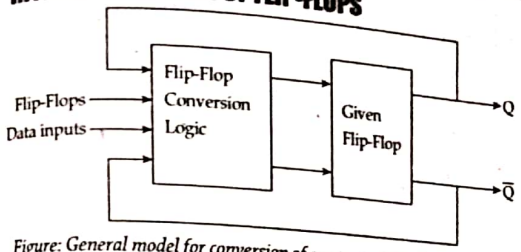


Figure: General model for conversion of one type of flip-flop to another type

From the model, we see that it is required to design the conversion logic for converting new input definitions into input codes that will cause the given flip flop to work like the desired flip-flop. To design the conversion logic, we need to combine the excitation table for both flips flops and make a truth table with data input (s) and Q as the inputs and the input (S) of the given flip flops as the output (s).

A. Conversion of an S-R flip-flop to a D flip-flop

The excitation table of S-R flip flops are given in the table.

FF data inputs	Output	S-R FF inputs	
D	Q	S	R
0	0	0	X
1	0	1	0
0	1	0	1
1	1	X	0

K-map for inputs S and R

D \ Q	0	1
0	0	0
1	1	X

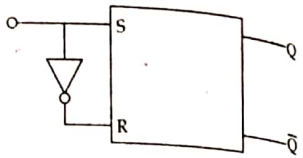
$\therefore$ $S = D$

and,

D \ Q	0	1
0	X	1
1	0	0

$\therefore$ $R = \overline{D}$

Hence, the circuit may be designed as



B. Conversion of an S-R flip flop to a J-K flip flop

The excitation tables of S-R and J-K flip-flop are given in the table below;

FF data inputs		Output	S-R FF inputs	
J	K	Q	S	R
0	0	0	0	X
0	1	0	0	X
1	0	0	1	0
1	1	0	1	0
0	1	1	0	1
1	1	1	0	1
0	0	1	X	0
1	0	1	X	0

From the truth table, the K-map is prepared as,

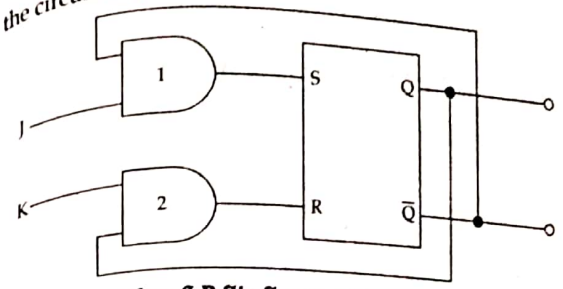
J \ KQ	00	01	11	10
0	0	X	0	0
1	1	X	0	1

$\therefore$ $S = J\overline{Q}$

J \ KQ	00	01	11	10
0	X	0	1	X
1	0	0	1	0

$\therefore$ $R = KQ$

Hence the circuit may be realized as



C. Conversion of an S-R flip flop to a T flip-flop

Excitation table:

FF data inputs	Output	S-R FF inputs	
T	Q	S	R
0	0	0	X
1	0	1	0
1	1	0	1
0	1	X	0

K-map;

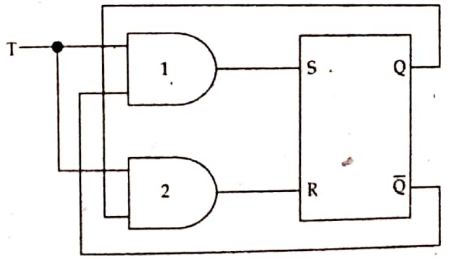
T \ Q	0	1
0	0	X
1	(1)	0

$\therefore$ $S = T\overline{Q}$

T \ Q	0	1
0	X	0
1	0	(1)

$\therefore$ $R = TQ$

Hence the circuit may be designed as,



D. Conversion of a D flip flop to an S-R flip-flop

Excitation Table:

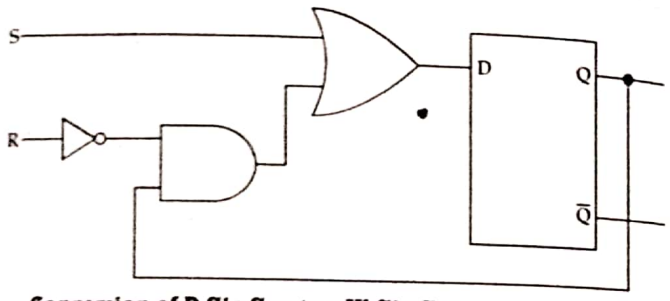
FF data inputs	Output	S-R FF inputs	
S	R	Q	D
0	0	0	0
0	1	0	0
1	0	0	1
0	1	1	0
0	0	1	1
1	0	1	1

K-map;

S \ RQ	00	01	11	10
0	0	1	1	0
1	1	1	X	X

$\therefore \quad D = S + \overline{R}Q$

Hence the circuit may be realized as,



E. Conversion of D flip flop to a JK flip-flop

Excitation Table:

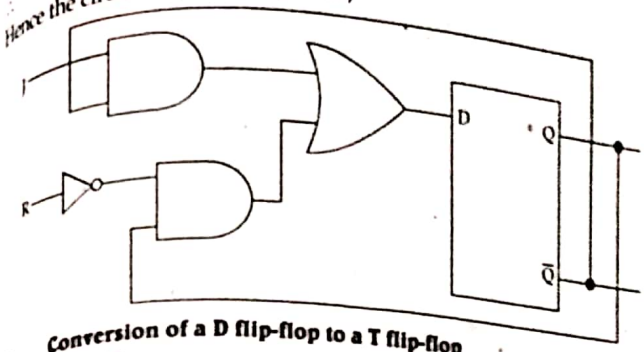
FF data inputs	Output	D FF inputs	
J	K	Q	D
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1
1	0	1	1

K-map;

J \ KQ	00	01	11	10
0	1	1	0	0
1	1	1	0	1

$D = J\overline{Q} + \overline{K}Q$

Hence the circuit may be designed as,



F. Conversion of a D flip-flop to a T flip-flop

Excitation table:

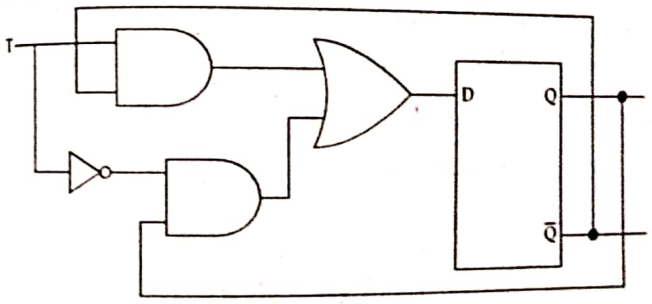
FF data inputs	Output	D FF inputs
T	Q	D
0	0	0
1	0	1
1	1	0
0	1	1

K-map;

T \ Q	0	1
0	0	(1)
1	(1)	0

$\therefore \quad D = T\overline{Q} + \overline{T}Q$

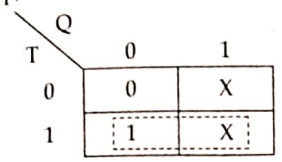
Hence the circuit may be designed as,



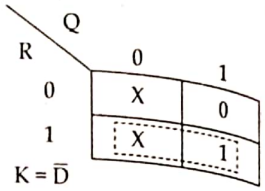
G. Conversion of a J-K flip flop to a D flip-flop excitation Table

FF data inputs	Output	J-K FF inputs	
D	Q	J	K
0	0	0	X
1	0	1	X
0	1	X	1
1	1	X	0

K-map;

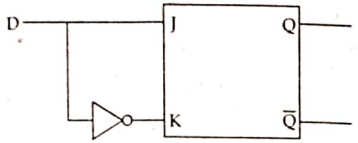


$\therefore$ J = D



$\therefore$ K = $\overline{D}$

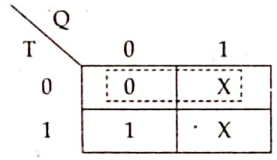
Hence the circuit may be designed as,



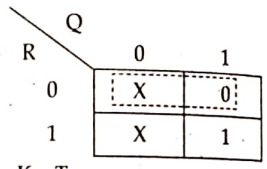
H. Conversion of a J-K flip-flop to a T flip-flop excitation table

FF data inputs	Output	J-K FF inputs	
T	Q	J	K
0	0	0	X
1	0	1	X
1	1	X	1
o	1	X	0

K-map;

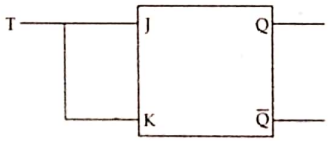


$\therefore$ J = T



$\therefore$ K = T

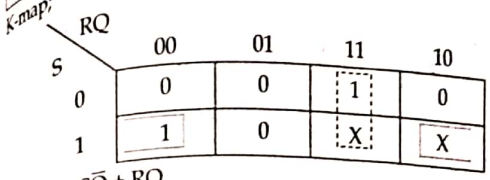
The circuit may be realized as,



I. Conversion of T flip-flop to a S-R flip-flop excitation table:

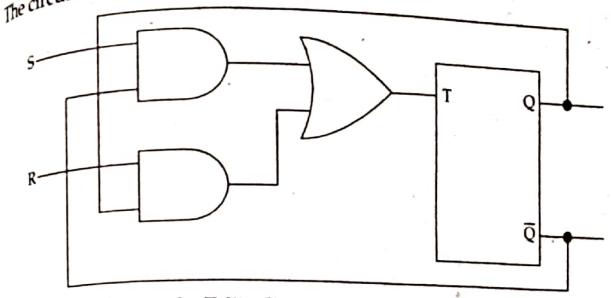
FF data inputs	Output	D FF inputs	
S	R	Q	T
0	0	0	0
0	1	0	0
0	0	0	1
1	1	1	1
0	0	1	0
0	0	1	0
1			

K-map;



$T = S\overline{Q} + RQ$

The circuit may be designed as,



J. Conversion of a T flip-flop to a J-K flip flop

Excitation table:

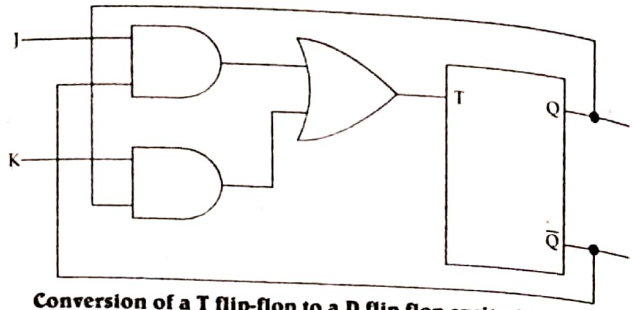
FF data inputs	Output	D FF inputs	
J	K	Q	T
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	1	1	1
1	1	1	1
0	0	1	0
1	0	1	0

K-map;

J \ KQ	00	01	11	10
0	0	0	1	0
1	1	0	1	1

$\therefore \quad T = J\overline{Q} + KQ$

Hence the circuit may be designed as,



K. Conversion of a T flip-flop to a D flip flop excitation table

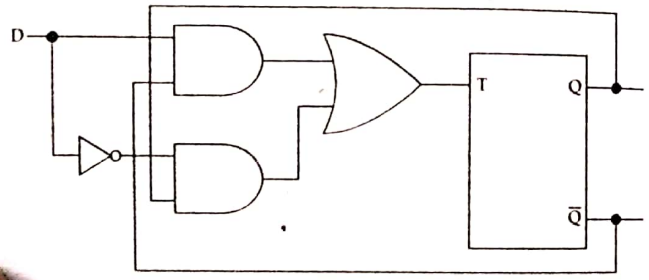
FF data inputs	Output	T FF inputs
D	Q	T
0	0	0
1	0	1
0	1	1
1	1	0

K-map

D \ Q	0	1
0	0	(1)
1	(1)	0

$\therefore \quad T = \overline{D}Q + D\overline{Q}$

The circuit may be designed as,



EXAMINATION QUESTION SOLUTIONS

1. **Explain about the JK Flip-Flop with respect to truth table, K-map, logic diagram, characteristic equation.** [2011/F, 2011/S, 2012/F, 2013/F, 2013/F, 2014/S, 2015/F, 2016/S, 2017/S, 2018/F, 2019/S]

Solution: See topic 7.1.3.

2. **Explain the operation of D Flip-Flop with the help of the table, logic diagram, K-map and characteristic equation.** [2011/S]

Solution: See the topic 7.1.2.

3. **Explain the operation of T Flip-flop with the help of truth table, diagram, K-map and characteristic equation.** [2012/S]

Solution: See the topic 7.1.4.

4. **Write short notes on state table state diagram.** [2012/F]

Solution: See the topic 7.3.A and 7.3.B.

5. **Write short notes on state reduction and assignment.** [2012/S, 2013/S, 2018/S]

Solution: See the topic 7.4.

6. **Write short notes on Master slave flip-flop.** [2013/S, 2013/F, 2014/S, 2017/S, 2018/F, 2019/F]

Solution: See the topic 7.2.

7. **How J-K flip flop can be converted into T flip-flop.** [2013/F, 2019/F]

Solution: See the topic 7.9 H.

8. **Write short notes on D flip-flop.** [2016/F]

Solution: See the topic 7.1.2.

9. **Explain the operation of clocked R-S flip flop with the help of its logic diagram characteristic table and characteristic equation.** [2017/F, 2017/S]

Solution: See the topic 7.1.1.

10. **How R-S flip flop can be converted into T flip-flop?** [2017/S]

Solution: See the topic 7.9 (C).

11. **Difference between latches and flip flops.** [2019/F]

Solution:

Latches	Flip-flops
Latches are building blocks of sequential circuit and these can be built from logic gates.	Flip flops are also building blocks of sequential circuits but these can be built from the latches.
Latch continuously checks its inputs and changes its output correspondingly.	It continuously check its inputs and changes its output correspondingly only at time determined by clocking signal.

It is based on the enable function input.	It works on the basis of clock pulse.
It is a level triggered.	It is an edge triggered.
It has no clock signal.	It has a clock signal.
It works only with binary inputs.	It works with binary inputs as well as clock signal.
It cannot be used as register.	It can be used register.
Power requirement is less.	Power requirement is more.
It is asynchronous device.	It is a synchronous device.

12. Difference between synchronous logic and asynchronous logic. **[2011/S, 2012/F, 2017/F]**

Solution:

Synchronous Logic	Asynchronous Logic
They have faster speed.	They have slower speed.
Components required is more.	Components required is less.
Higher cost.	Lesser cost.
Clock is different for all flip-flops.	Clock is same for all flip flops.
They have high operating frequency.	They have low operating frequency.
Memory elements are clocked flip flops.	Memory element are either unlocked flip-flop or time delay elements.
The change in input signals can affect memory elements upon activation of clock signal.	The change in input signals can affect memory element at any instant of time.

13. What are the disadvantages of R-S flip flop? **[2012/F, 2017/S]**

Solution:

The one major disadvantage of S-R flip flop is that in the condition when the clock is triggered the inputs become high which is an undesirable condition because it causes invalid input, the condition in which we can't predict the output.

14. Write the advantages of J-K flip flop over S-R flip-flop. **[2013/S]**

Solution:

Advantages of J-K flip flop over S-R flip flop are,

i) The indeterminate state is eliminated.

ii) There is a single input that can be used for setting or resetting.

15. Write short notes on edge triggered flip-flop. **[2017/S]**

Solution:

In a level triggered flip-flop, the output responds to the data present at the inputs during the time clock pulse level is high or low. That is, any changes at the input during the time the clock is active (High or Low) are reflected at the output as per its function table.

In an edge triggered flip-flop, the output responds to the data at the inputs only on low to high or high to low transition of the clock signal. The flip-flop in the two cases is referred to as positive edge triggered and negative edge triggered respectively. Any changes in the input during the time the clock pulse is high or low do not have any effect on the output. In the case of an edge triggered flip-flop, an edge detector circuit transforms the clock input into a very narrow pulse that is a few nano seconds wide. This narrow pulse coincides with either Low-to-High or High-to-Low transitions of the clock input, depending upon whether it is a positive edge triggered flip-flop or a negative edge-triggered flip-flop. This pulse is so narrow that the operation of the flip-flop can be considered have occurred on the edge itself.

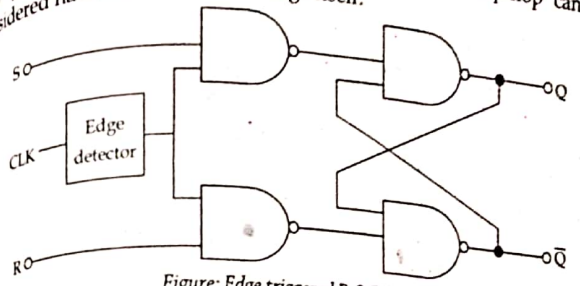


Figure: Edge triggered R-S flip flops

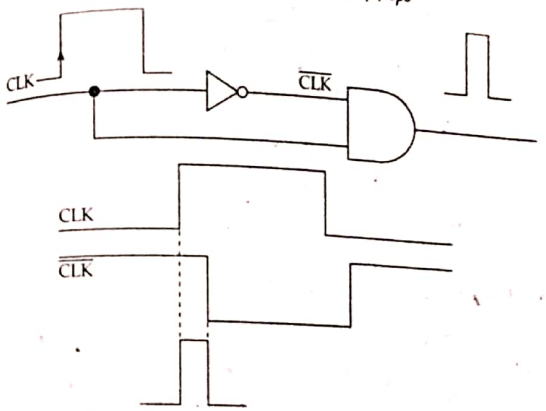


Figure: Positive edge triggered edge detector circuits

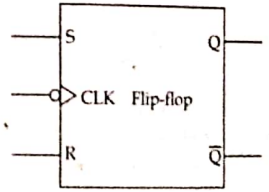


Figure: Positive edge R-S flip-flops circuit symbol

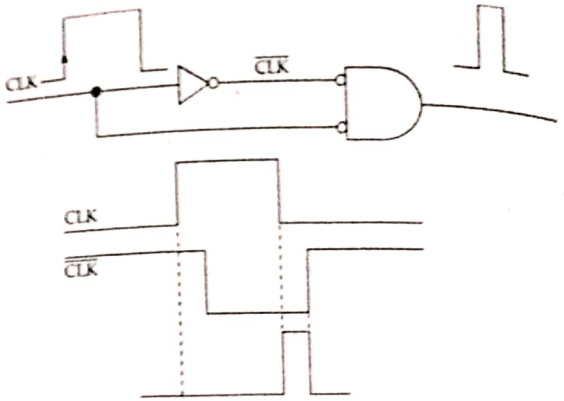


Figure: Negative edge triggered edge detector circuits

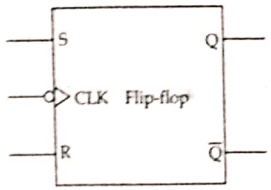
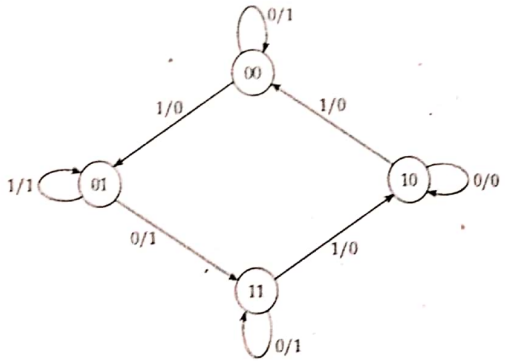


Figure: Circuit symbol of negative edge trigger R-S flip flop

16. Realize the following state diagram into a circuit using J-K flip-flop. **[2014/S]**



Solution:

Let A and B denote four different states, X and Y denote input and output respectively. The state excitation table for above state diagram using J-K flip flop is given below;

Present state A	Present state B	Input X	Next state A	Next state B	Output Y	Flip-flops inputs J_A	K_A	J_B	K_B
0	0	0	0	0	1	0	X	0	X
0	0	1	0	1	0	0	X	1	X
0	1	0	1	1	1	1	X	X	0
0	1	1	0	1	1	0	X	X	0
0	0	0	1	0	0	X	0	0	X
1	0	1	0	0	0	X	1	0	X
1	1	0	1	1	1	X	0	X	0
1	1	1	1	0	0	X	0	X	1

K-map:

$J_A = B\overline{X}$

A \ BX	00	01	11	10
0	0	0	0	1
1	X	X	X	X

$K_A = \overline{B}X$

A \ BX	00	01	11	10
0	X	X	X	X
1	0	1	0	0

$J_B = \overline{A}X$

A \ BX	00	01	11	10
0	0	1	X	X
1	0	0	X	X

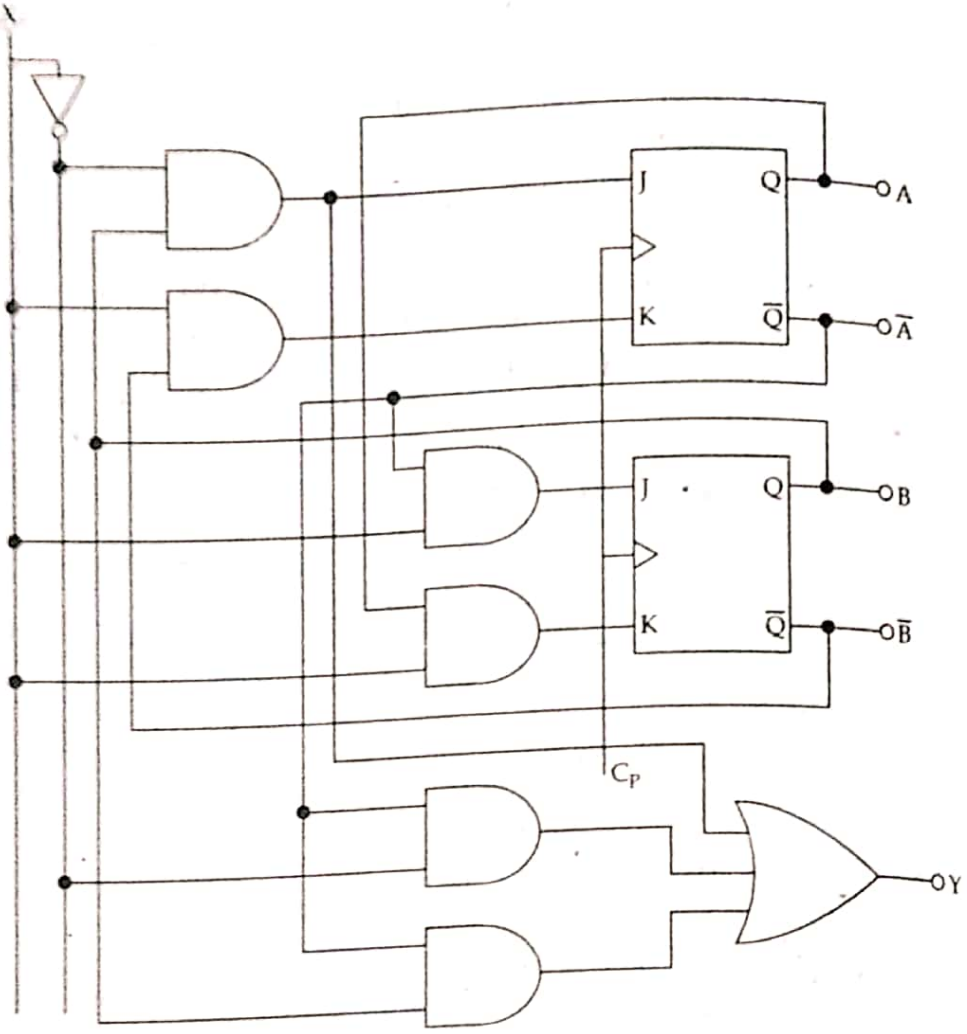
$K_B = AX$

A \ BX	00	01	11	10
0	X	X	0	0
1	X	X	1	0

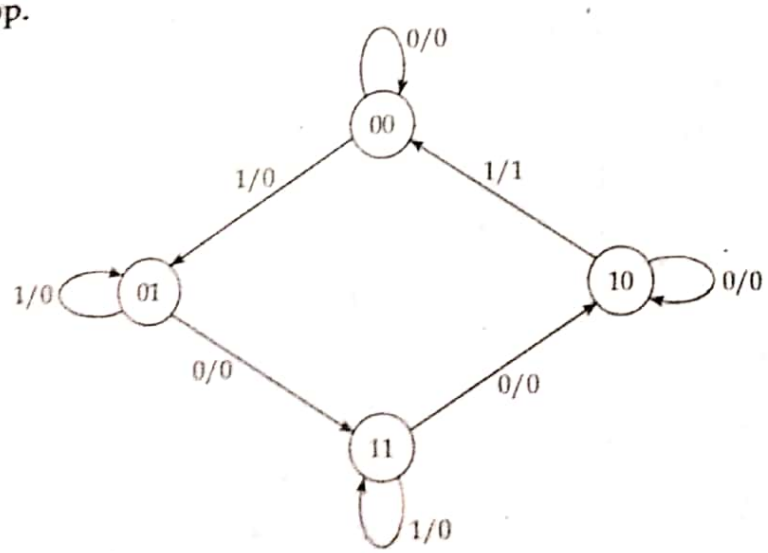
A \ BX	00	01	11	10
0	1	0	1	1
1			0	1

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

Circuit diagram is:



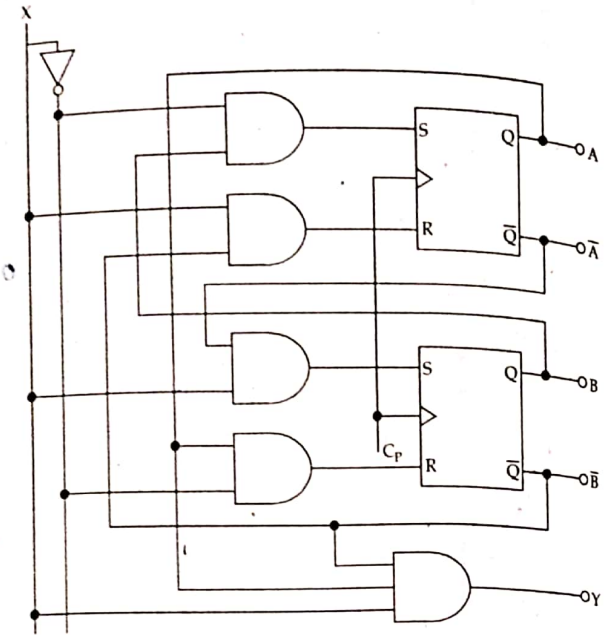
17. Realize the following state diagram into a circuit using S-R flip flop. [2014/F]



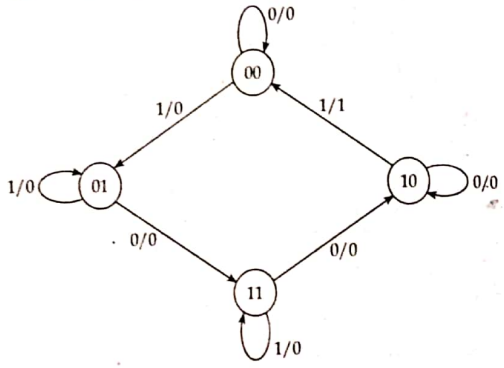
Solution:

State excitation table;

Circuit diagram is:



18. **Design a sequential circuit corresponding to given state diagram using J-K flip-flop.** [2015/5]



Solution:

Let AB denotes the state variables, X and Y denotes input and output respectively. The state excitation table above state diagram using J-K flip-flops is given below;

Present state		Input	Next state		Output	Flip-flops inputs			
A	B	X	A	B	Y	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	0	X	1	X
0	1	0	1	1	0	1	X	X	0
0	1	1	0	1	0	0	X	X	0
0	0	0	1	0	0	X	0	0	X
0	0	1	0	0	1	X	1	0	X
1	0	0	1	0	0	X	0	X	1
1	1	1	1	1	0	X	0	X	0

K-maps;

A \ BX	00	01	11	10
0	0	0	0	1
1	X	X	X	X

$J_A = B\bar{X}$

A \ BX	00	01	11	10
0	X	X	X	X
1	0	1	0	0

$K_A = \bar{B}X$

A \ BX	00	01	11	10
0	1	1	X	X
1	0	0	X	X

$J_B = \bar{A}X$

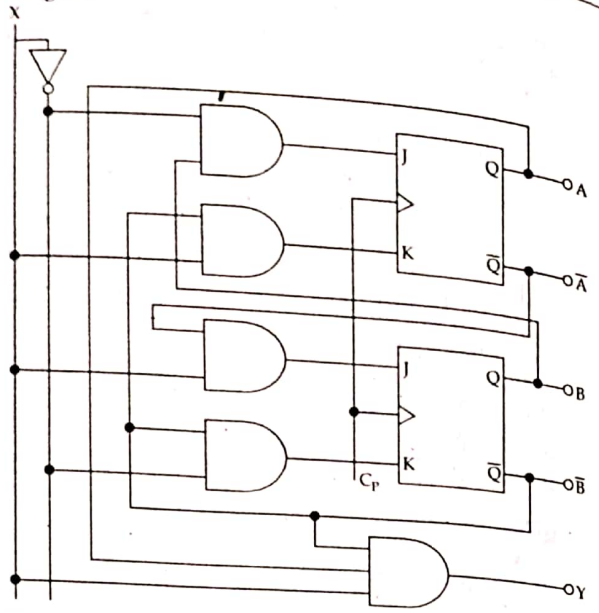
A \ BX	00	01	11	10
0	X	X	0	0
1	X	X	0	1

$K_B = A\bar{X}$

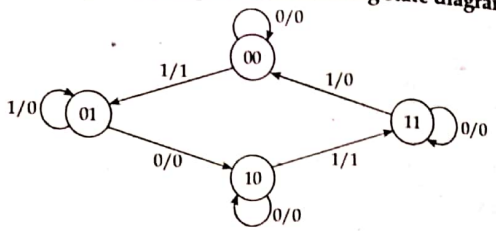
A \ BX	00	01	11	10
0	0	0	0	0
1	0	1	0	0

$Y = A\bar{B}X$

Circuit diagram is:



19. Design a sequential circuit corresponding to the given state diagram using D flip flop for the following state diagram. [2015/F]



Solution:

State excitation table using D flip-flop;

Present state		Input	Next state		Output	Flip-flops inputs	
A	B	X	A	B	Y	D_A	D_B
0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1
0	1	0	1	0	0	1	0
0	1	1	0	1	0	0	1
1	0	0	1	0	0	1	0
1	0	1	1	1	1	1	1
1	1	0	1	1	0	1	1
1	1	1	0	0	0	0	0

K-map:

A \ BX	00	01	11	10
0	0	0	0	1
1	1	1	0	1

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

A \ BX	00	01	11	10
0	0	1	1	0
1	0	1	0	1

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

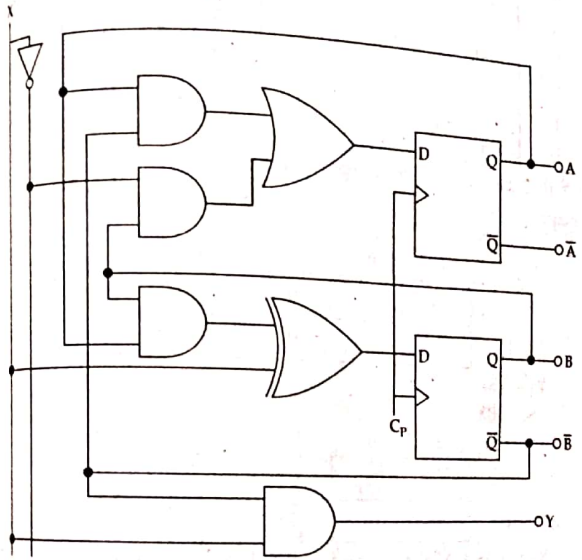
$= X(\bar{A}\bar{B}) + AB\bar{X}$

$= (AB) \oplus X$

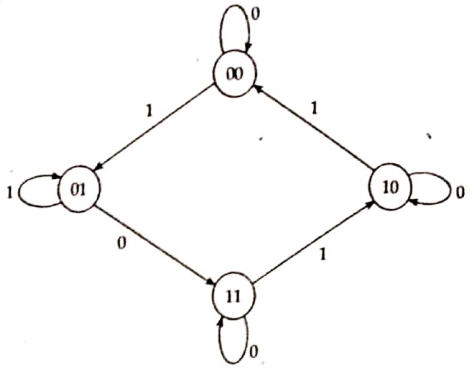
A \ BX	00	01	11	10
0	0	1	0	0
1	0	1	0	0

$Y = \bar{B}X$

Circuit diagram:



20. Realize the following state diagram into a circuit using T flip-flop. [2016/F]



Solution:

State excitation table using T flip-flop;

Present state		Input	Next state		Flip-flops inputs	
A	B	X	A	B	T_A	T_B
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	1	0	0
1	0	0	1	0	0	0
1	0	1	0	0	1	0
1	1	0	1	1	0	0
1	1	1	1	0	0	1

From the table, we can derive expression for T_A and T_B,

K-map;

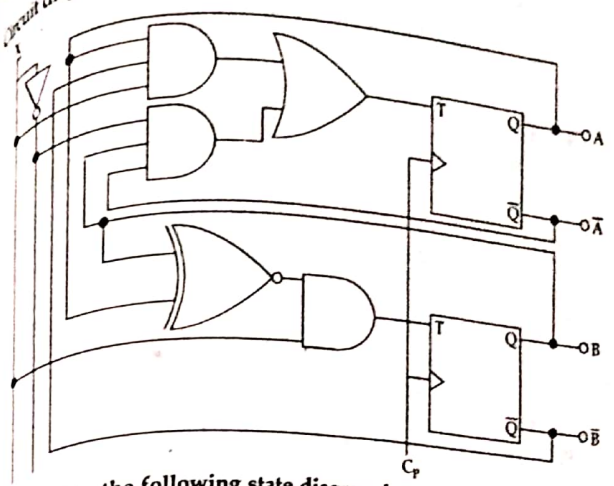
A \ BX	00	01	11	10
0	0	0	0	(1)
1	0	(1)	0	0

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

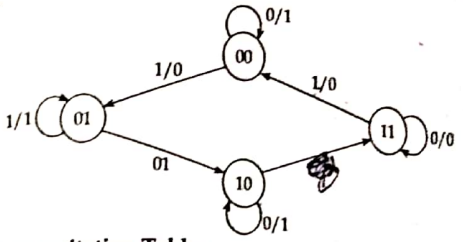
A \ BX	00	01	11	10
0	0	(1)	0	0
1	0	0	(1)	0

$\therefore T_B = \bar{A}\bar{B}X + ABX = X(AB + \bar{A}\bar{B}) = X(A \odot B)$

Circuit diagram;



21. Realize the following state diagram into a circuit using S-R flip-flop. [2016/S]



Solution: State excitation Table;

Present state		State	Next state		Output	Flip-flops inputs			
A	B	X	A	B	Y	S_A	R_A	S_B	R_B
0	0	0	0	0	1	0	X	0	X
0	0	1	0	1	0	0	X	1	0
0	1	0	1	1	1	1	0	X	0
0	1	1	0	1	1	0	X	X	0
1	0	0	1	0	0	X	0	0	X
1	0	1	0	0	0	0	1	0	X
1	1	0	1	1	1	X	0	X	0
1	1	1	1	0	0	X	0	0	1

K-map;

A \ BX	00	01	11	10
0	0	0	0	[1]
1	X	0	X	[X]

$\therefore S_A = B\bar{X}$

A \ BX	00	01	11	10
0	X	X	X	0
1	0	1	0	0

$\therefore R_A = \overline{B}X$

A \ BX	00	01	11	10
0	0	1	X	X
1	0	0	0	X

$\therefore S_B = \overline{A}X$

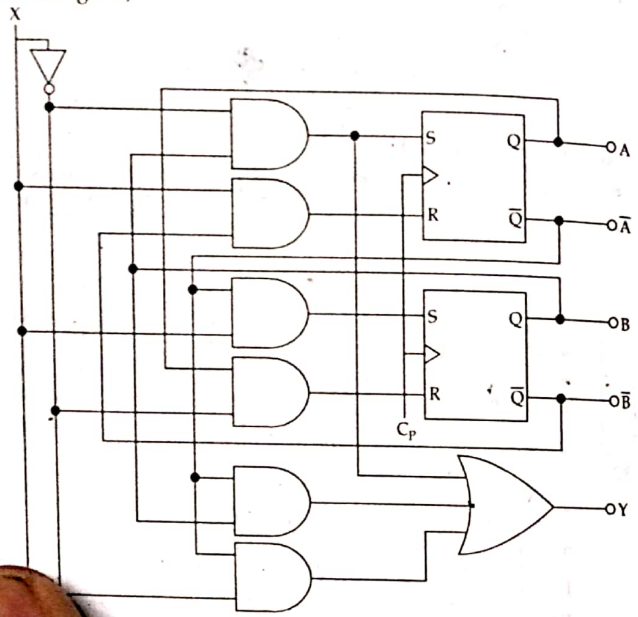
A \ BX	00	01	11	10
0	X	0	0	0
1	X	X	1	0

$\therefore R_B = AX$

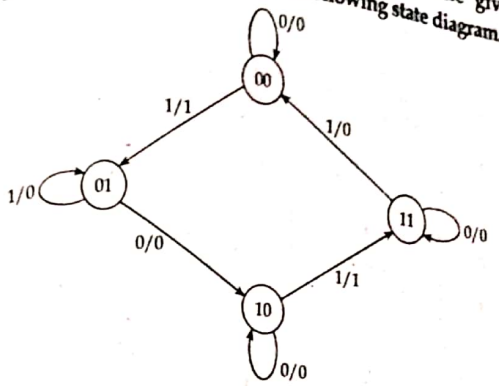
A \ BX	00	01	11	10
0	1	0	1	1
1	0	0	0	1

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

Circuit diagram;



Design a sequential circuit corresponding to the given state diagram using S-R flip-flop for the following state diagram. [2018/F]



Solution:

State excitation Table;

Present state		Input	Next state		Output	Flip-flops inputs			
A	B	X	A	B	Y	S_A	R_A	S_B	R_B
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	1	0	X	1	0
0	1	0	1	0	0	1	0	0	1
0	1	1	0	1	0	0	X	X	0
1	0	0	1	0	0	X	0	0	X
1	0	1	1	1	1	X	0	1	0
1	1	0	1	1	0	X	0	X	0
1	1	1	0	0	0	0	1	0	1

K-map;

A \ BX	00	01	11	10
0	0	0	0	1
1	X	X	0	X

$\therefore S_A = B\overline{X}$

A \ BX	00	01	11	10
0	X	X	X	0
1	0	0	1	X

$\therefore R_A = BX$

A \ BX	00	01	11	10
0	0	1	X	0
1	0	1	0	X

$\therefore \quad S_B = \overline{B}X$

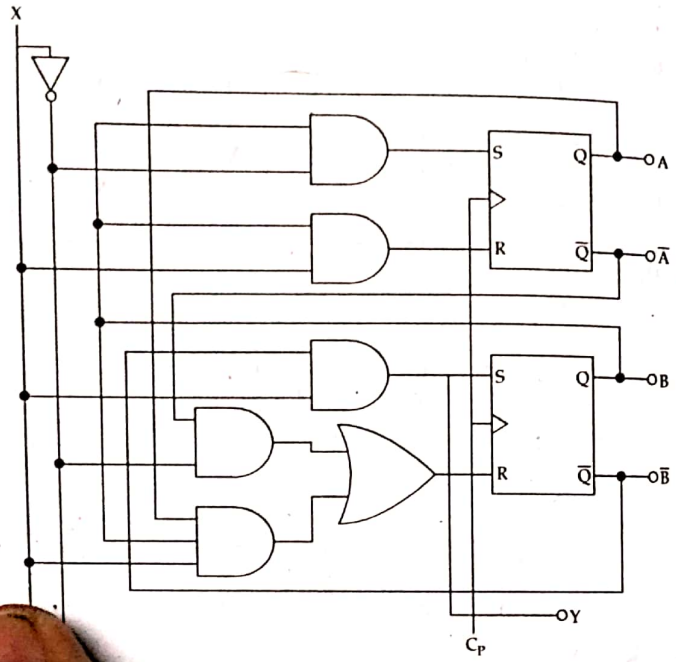
A \ BX	00	01	11	10
0	X	0	0	1
1	X	0	(1)	0

$\therefore \quad R_B = \overline{A}\,\overline{X} + ABX$

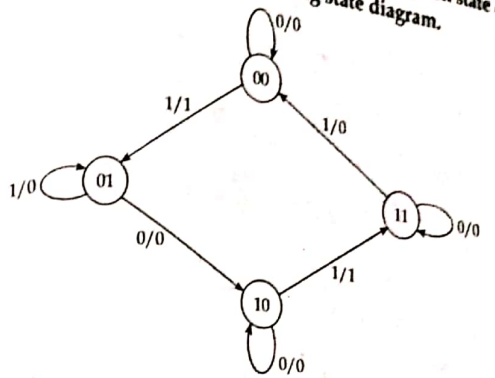
A \ BX	00	01	11	10
0	0	1	0	0
1	0	1	0	0

$\therefore \quad Y = \overline{B}X$

Circuit diagram;



Q3. Design a sequential circuit corresponding to the given state diagram using J-K flip-flop for the following state diagram. [2019/F]



Solution:

State excitation Table;

Present state		Input	Next state		Output	Flip-flops inputs			
A	B	X	A	B	Y	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	1	0	X	1	X
0	1	0	1	0	0	1	X	X	1
0	1	1	0	1	0	0	X	X	0
1	0	0	1	0	0	X	0	0	X
1	0	1	1	1	1	X	0	1	X
1	1	0	1	1	0	X	0	X	0
1	1	1	0	0	0	X	1	X	1

K-map;

A \ BC	00	01	11	10
0	0	0	0	1
1	X	X	X	X

$\therefore \quad J_A = B\overline{X}$

A \ BX	00	01	11	10
0	X	X	X	X
1	0	0	1	0

$\therefore \quad K_A = BX$

A \ BX	00	01	11	10
0	0	1	X	X
1	0	1	X	X

$\therefore \quad J_B = X$

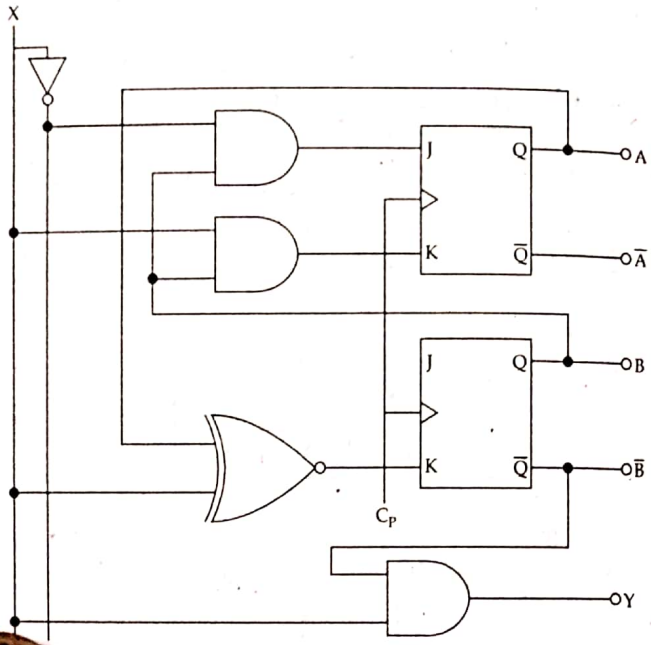
A \ BX	00	01	11	10
0	X	X	0	1
1	X	X	1	0

$\therefore \quad K_B = \overline{A}\,\overline{X} + AX = A \odot X$

A \ BX	00	01	11	10
0	0	1	0	0
1	0	1	0	0

$\therefore \quad Y = \overline{B}X$

Circuit diagram;

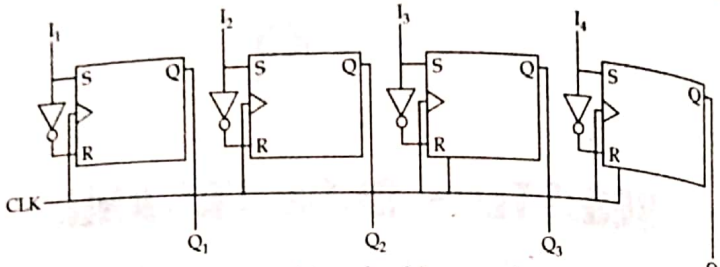


CHAPTER 8

REGISTERS, COUNTERS AND MEMORY UNIT

8.1 REGISTERS

A register is a group of binary storage cells capable of holding binary information. A group of flip flops constitutes a registers, since each flip-flop can work as a binary cell. An n-bit register, has n flip-flops and is capable of holding n-bits of information. In addition to flip flops, a register can have a combinational part that performs data processing tasks. Various types of registers are available in MSI circuits. The simplest possible register is one that contains no external gates and is constructed of only flip-flops. Figure below shows such type of register constructed of four S-R flip flops with a common clock pulse input. The clock pulse enable all the flip flops at the same instant so that the information available at the four inputs can be transferred into 4-bit register.



All the flip-flops in a register should respond to the clock pulse transistion. Hence they should be either of the edge triggered type or the master slave type. A group of flip flops sensitive to the pulse duration is commonly called a gated latch. Latches are suitable to temporarily store binary information that is to be transferred to an external destination. They should not be used in the design of sequential circuit that have feedback connections.

8.2 SHIFT REGISTERS

A register capable of shifting its binary contents either to the left or the right is called a shift register. The shift register permits the stored data to more from a particular location to some other location within the register. Registers can be designed using discrete flip-flops (S-R, J-K and D-type).

The data in shift register can be shifted in two possible ways: (a) serial shifting and (b) parallel shifting. The serial shifting method shifts one bit at a time for each clock pulse in a serial manner, beginning with either LSB or MSB. On the other hand, in parallel shifting operation, all the data (input or output) gets shifted simultaneously during a single clock pulse. Hence we may say that parallel shifting operation is much faster than serial shifting operation.

There are two ways to shift data into a register (serial or parallel) and similarly two ways to shift the data out of the register. This leads to the construction of four basic types of registers. All of the four configurations are commercially available as TTL MSI/LSI circuits. They are;

a) Serial In/Serial out (SISO)
b) Serial In/Parallel out (SIPO)
c) Parallel In/Serial out (PISO)
d) Parallel In/Parallel out (PIPO)

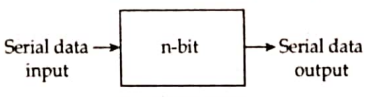


Figure: (a) SISO

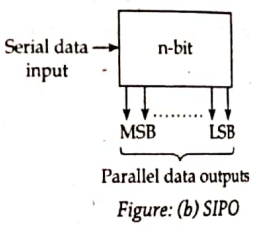


Figure: (b) SIPO

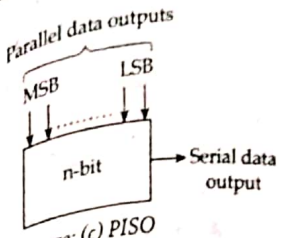


Figure: (c) PISO

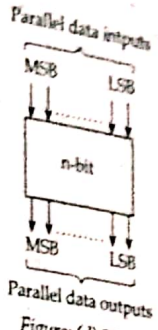


Figure: (d) PIPO

Figure: Four types of shift registers

Serial-In-Serial Out Shift Register (SISO)

This type of register accepts data serially, *i.e.*, one bit at a time at the single input line. The output is also obtained on a single output line in a serial fashion. The data within the register may be shifted from left to right using shift left register or may be shifted from right to left using shift right register.

Shift-right Register

A shift register can be constructed with either J-K or D flip-flops as shown in figure. A J-K flip-flop, based shift register required connection of both J and K inputs. Input data are connected to the J and K Inputs of the left most flip-flop. To input a 0, one should apply a 0 bit the J input, *i.e.*, J = 0 and K = 0 and vice-versa. With the application of a clock pulse, the data will be shifted by one bit to the right.

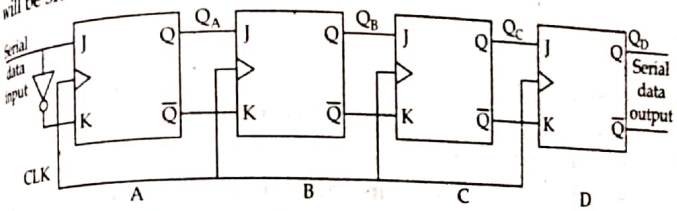


Figure: Shift-right register using J-K flip-flops.

In the shift register using D flip flop, D input of the left most flip-flop is used as a serial input line. To input 0, one should apply 0 at the D input and vice-versa.

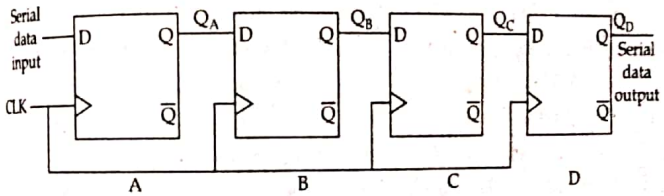


Figure: Shift-right register using D flip-flops

The clock pulse is applied to all the flip-flops simultaneously. When the clock pulse is applied, each flip-flop is either set or reset according to the data available at that point of time at the respective inputs of the individual flip flops. Hence the input data at the serial input line is entered into flip flop A by the first clock pulse. At the same time, the data of stage A is shifted into stage B and so on to the following stages. For each clock pulse, data stored in the register is shifted to the right by one stage. New data is entered into stage A whereas the data present in stage D are shifted out to the right.

Table: Operation of the shift-right register

Timing pulse	Q_A	Q_B	Q_C	Q_D	Serial output at Q_D
Initial value	0	0	0	0	0
After 1st clock pulse	1	0	0	0	0
After 2nd clock pulse	1	1	0	0	0
After 3rd clock pulse	0	1	1	0	0
After 4th clock pulse	1	0	1	1	1

For example; consider that all the stages are reset and a logical input 1011 is applied at the serial input line connected to stage A. The data after four clock pulses is shown in table above.

Let us now illustrate the entry of the 4-bit number 1011 into the register, beginning with the right most bit. A 1 is applied at the serial input line, making D = 1. As the first clock pulse is applied, flip flop A is SET, thus storing the 1. Next, a 1 is applied to the serial input making D = 1 for the flip flop A and D = 1 for the flip-flop B also, because the input of flip flop B is connected to the Q_A input.

When the second clock pulse occurs, the 1 on the data input is shifted to the flip flop A and the 1 in the flip flop A is shifted to flip flop B. The 0 the binary number is now applied at the serial input line and the third clock pulse is now applied. This 0 is entered in the flip flop A and the 1 stored in flip flop A is now shifted to flip flop B and the 1 stored in flip flop B is now shifted to flip flop C. The last bit in the binary number that is the 1 is now applied at the serial input line and the fourth clock pulse is now applied. This 1 now enters the flip-flop A and the 0 stored in the flip-flop A is now shifted to flip-flop B and the 1 stored in flip-flop B is now shifted to the flip flop C and the 1 stored in flip-flop C is new shifted to flip-flop D. Thus the entry of the 4-bit binary number in the shift-right register are completed.

For the third column of table, we can get the serial output of the data that is being entered in the register. We find that after the first, second and the third clock pulses the output at the serial output line *i.e.*, Q_D is 0. After the fourth clock pulse the output at the serial output line is 1. If we want to get the total data that we have entered in the register in a serial manner from Q_D, then we have to apply another three clock pulses. After the fifth clock pulse we will get another at Q_D will be 0 and after the seventh clock pulse the output at Q_D will be 1. In this process of the fifth, sixth and the seventh clock pulses if no data is being supplied at the serial input line then the A, B, and C flip-flops will again be RESET with output 0.

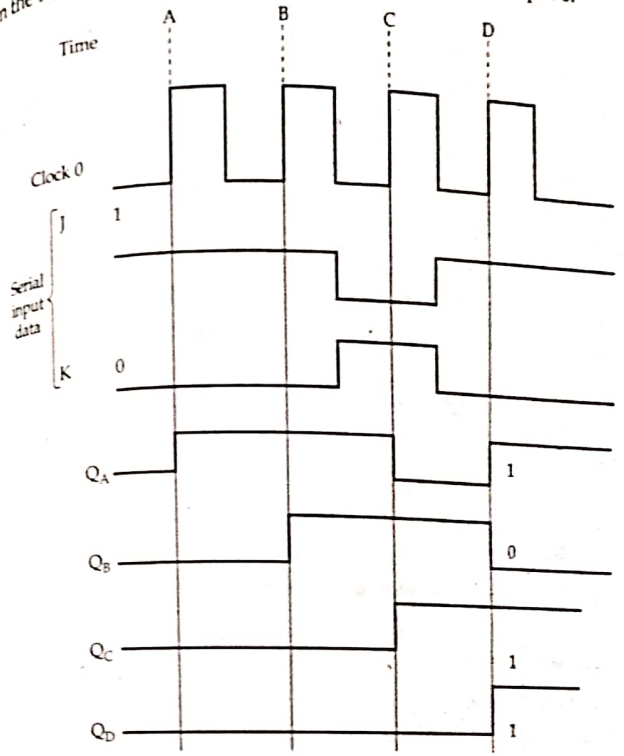


Figure: Waveforms of 4-bit serial input shift-right register

The waveforms shown in figure above illustrate the entry of a 4-bit number 1011. For a J-K flip-flop, the data bit to be shifted into the flip-flop must be present at the J and K inputs when the clock transistions from low to high occur. Since the data bit is either 1 or 0, there can be two different cases.

i) To shift a 1 into flip-flop, J = 1 and K = 0.

ii) To shift a 0 into the flip flop, J = 0 and K = 1.

At time A

All the flip-flops are reset. At the serial data input has a 1 is given and with the first clock pulse this 1 is shifted at Q_A making Q_A = 1. At the same time, the 0 in Q_A is shifted to Q_B and the 0 in Q_B is shifted to Q_C and the 0 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time A are;

$$Q_AQ_BQ_CQ_D = 1000$$

At time B

The flip-flop A contains 1 and all other flip-flop contains 0. Now, again 1 is given at the serial data input line. With the second clock pulse this 1 is shifted to Q_A. The 1 in Q_A is shifted to Q_B and the 0 in Q_B is shifted to Q_C and the 0 in Q_C is shifted to Q_D. Hence the flip flop outputs just after time B are;

$$Q_AQ_BQ_CQ_D = 1100$$

At time C

The flip-flop A and flip-flop B contain 1 and all other flip-flops contain 0. Now an 0 is given at the serial data input line. With the third clock pulse this 0 is shifted to Q_A. The 1 in Q_A is shifted to Q_B and the 1 in Q_B is shifted to Q_C and 0 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time C are

$$Q_AQ_BQ_CQ_D = 0110$$

At time D

The flip B and flip flop C contain 1 and all other flip-flops contain 0. Now another 1 is given at the serial data input line. With the fourth clock pulse this 1 is shifted to Q_A. The 0 in Q_A is shifted to Q_B and the 1 in Q_B is shifted to Q_C and the 1 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time C are;

$$Q_AQ_BQ_C\dot{Q}_D = 1011$$

II. Shift-Left Register

A shift-left register can also be constructed with either J-K or D flip-flops.

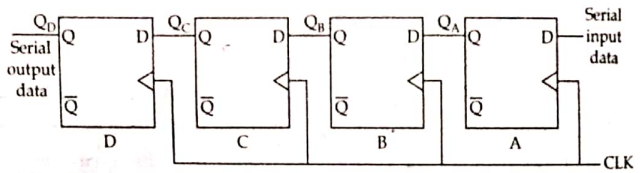


Figure: Shift-left register using D flip-flops

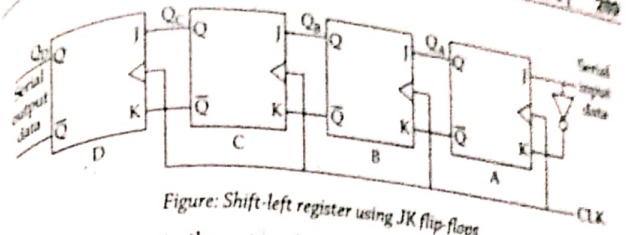


Figure: Shift-left register using JK flip-flops

Let us now illustrate the entry of the 4-bit number 1110 into the register, beginning with the right-most bit. A 0 is applied at the serial input line, making D = 0. As the first clock pulse is applied, flip-flop A is RESET, thus storing the 0. Next a 1 is applied to the serial input, making D = 1 for flip-flop A and D = 0 for the flip-flop B because the input of flip-flop B is connected to the Q_A output.

When the second clock pulse occurs, the 1 on the data input is shifted to the flip flop A and the 0 in the flip-flop A is shifted to flip flop B. The 1 in the binary number is now applied at the serial input line and the third clock pulse is now applied. This 1 is entered in flip-flop A and the 1 stored in flip flop A is now shifted to flip-flop B and the 0 stored in flip-flop B is now shifted to flip-flop C. The last bit in the binary number that is the 1 is now applied at the serial input line and the fourth clock pulse is now applied. This 1 now enters the flip flop A and the 1 stored in flip-flop A is now shifted to flip-flop B and the 1 stored in flip-flop B is now shifted to flip-flop C and the 0 stored in flip-flop C is now shifted to flip-flop D. Thus the entry of the 4-bit binary number in the shift-right register is now completed.

Table: Operation of Shift-Left register

Timing pulse	Q_D	Q_C	Q_B	Q_A	Serial output at Q_D
Initial value	0	0	0	0	0
After 1st clock pulse	0	0	0	0	0
After 2nd clock pulse	0	0	0	1	0
After 3rd clock pulse	0	0	1	1	0
After 4th clock pulse	0	1	1	1	0

B. Serial In-Parallel Out Register (SIPO)

In this type of register, the data is shifted in serially, but shifted out in parallel. To obtain the output in parallel, it is required that all the output bits are available at the same time. This can be accomplished by connecting the output of each flip-flop to an output pin.

IC 74164 is an example of 8-bit serial in parallel out shift register. There are eight S-R flip which are all sensitive to negative clock transitions.

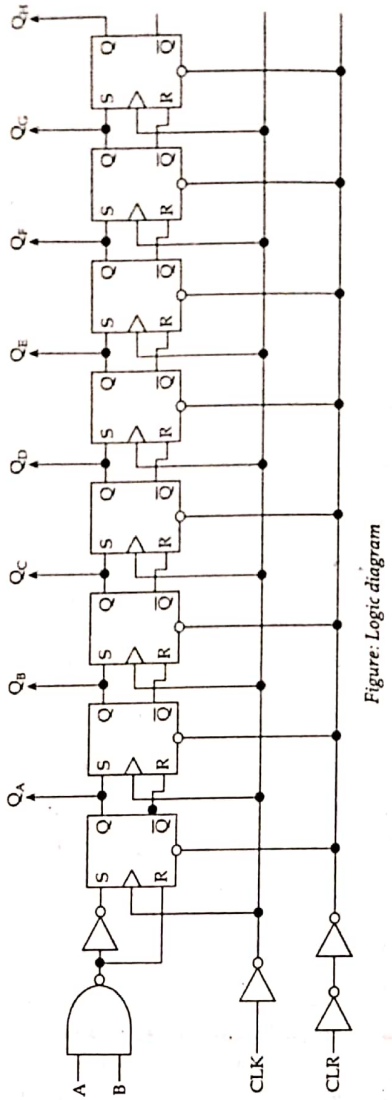


Figure: Logic diagram

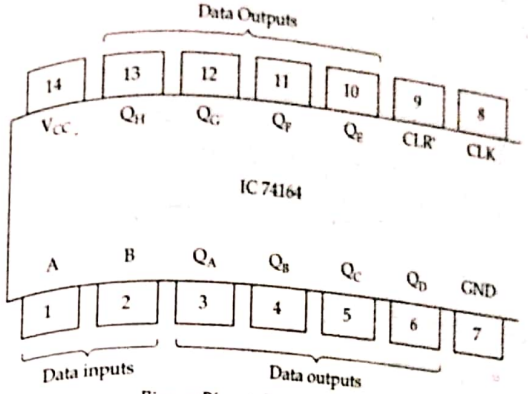


Figure: Pinout diagram of IC 74164

A is held high

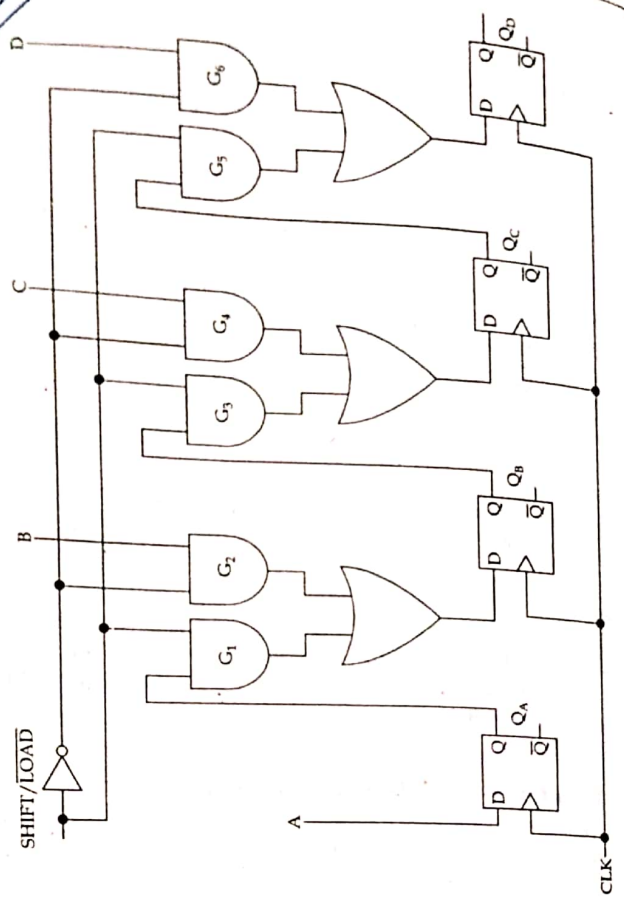
The NAND gate is enabled and the serial data passes through the NAND gate inverted. The input data is shifted serially into the register.

A is held low

The NAND gate output is forced high, the input data steam is inhibited, and the next clock pulse will shift 0 into the first flip flop. Each succeeding positive clock pulse will shift another 0 into the register. After eight clock pulse, the register will be full of zeros.

c. Parallel-In-Serial-Out Register (PISO)

In the preceding two cases, the data was shifted in to the register in a serial manner. We now can develop an idea for the parallel entry of data into the register. Here the data bits are entered into the flip-flops simultaneously rather than a bit-by bit basis. A 4-bit parallel-in-serial-out register is illustrated below;



A, B, C and D are the four parallel data input lines and SHIFT/$\overline{\text{LOAD}}$ (SH/$\overline{\text{LD}}$) is a control input that allows the four bits of data at A, B, C and D inputs to enter into the register in parallel or shift the data in serial. When SHIFT/$\overline{\text{LOAD}}$ is high, AND gates G_1, G_3 and G_5 are enabled, allowing the data bits to shift right from one stage the next. When SHIFT/$\overline{\text{LOAD}}$ is low, AND gates G_2, G_4 and G_6 are enabled, allowing the data bits at the parallel inputs. When a clock pulse is applied, the flip-flops with D = 1 will be set and the flip-flops with D = 0 will be reset thereby storing all the four bits simultaneously. The OR gates allow either the normal shifting operation or the parallel data entry the normal shifting ...ation or the parallel data entry operation depending on which of the ...gates are enabled by the level on the SHIFT/$\overline{\text{LOAD}}$ input.

Parallel-In-Parallel-out Register

In this type of register, there is no interconnection between the flip-flop since no serial shifting is required. Hence, the moment the parallel entry of the data is accomplished, the data will be available at the parallel output of the register. A simple parallel-in-parallel-out shift register is shown below;

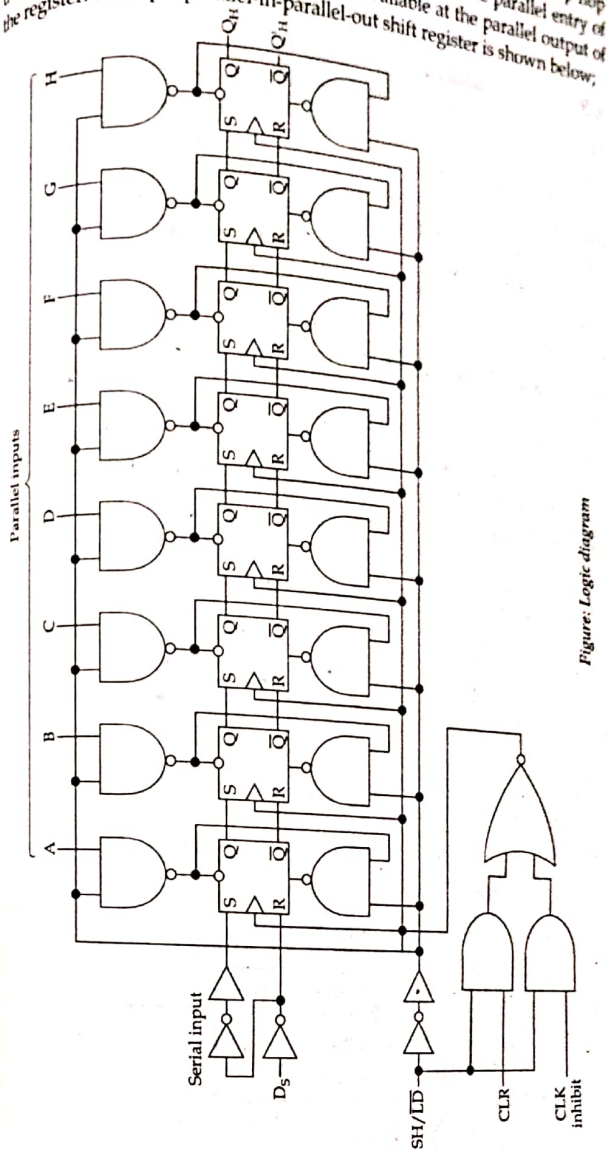


Figure: Logic diagram

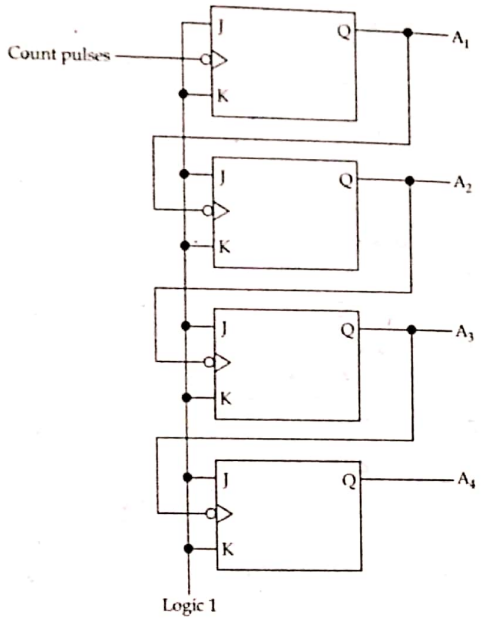
Here the parallel inputs to be applied at A, B, C and D inputs are directly connected to D inputs of the respective flip-flops. On applying the clock transitions, these inputs are entered into the register and are immediately available at the outputs Q_1, Q_2, Q_3 and Q_4.

8.3 RIPPLE COUNTERS

MSI counters come in two categories: ripple counters and synchronous counters. In a ripple counter, the flip-flop output transition serves as a source for triggering other flip-flop. In other words, the CP inputs of all flip-flops (except the first) are triggered not by the incoming pulses but rather by the transition that occurs in other flip-flops. In a synchronous counter, the input pulses are applied to all CP inputs of all flip-flops. The change of state of a particular flip-flop is dependent on the present state of other flip-flops.

A. Binary Ripple Counter

A binary ripple counter consists of a series connection of complementing the flip-flops (T or JK type), with the output of each flip-flop connected to the CP input of the next higher order flip-flop. The flip flop holding the least significant bit receives the incoming count pulses. The diagram of a 4-bit binary ripple counter is shown below;



Logic 1

All J and K inputs are equal to 1. The small circle in the CP input indicates that the flip-flop complements during a negative going transition or when the output to which it is connected goes from 1 to 0. Ripple counters are sometimes called asynchronous counters.

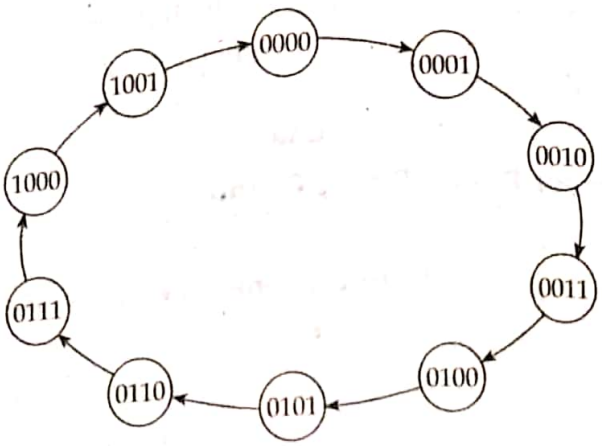
Count Sequence for Binary Ripple Counter

Count Sequence				Conditions for complementing Flip-Flops	
	A_3	A_2	A_1		
0	0	0	0	Complement A_1	
0	0	0	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2
0	0	1	0	Complement A_1	
0	0	1	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2. A_2 will go from 1 to 0 and complement A_3.
0	1	0	0	Complement A_1	
0	1	0	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2.
0	1	1	0	Complement A_1	
0	1	1	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2; A_2 will go from 1 to 0 and complement A_3; A_3 will go from 1 to 0 and complement A_4
1	0	0	0	and so on	

A binary counter with a reverse count is called a binary down counter. In a down counter, the binary count is decremented by 1 with every input count pulse. The count of a 4-bit down counter stats from binary 15 and continues to binary counts 14, 13, 12, ..., 0 and then back to 15. The circuit of binary ripple counter will function as a binary down counter if the outputs are taken from the complement terminals $\overline{Q}$ of all flip-flops.

B. BCD Ripple counter

A decimal counter follows a sequence of ten states and returns to 0 after the count 9. Such a counter must have atleast four flip flops to represent each decimal digit, since a decimal digit is represented by a binary code with atleast four bits. The sequence of states in a decimal counter is dictated by the binary code used to represent a decimal digit. If BCD is used, the sequence of states is as shown in the state diagram.



The design of a decimal ripple counter or of any ripple counter not following the binary sequence is not a straight forward procedure. The formal tools of logic design can serve only as a guide. A satisfactory end product requires the ingenuity and imagination of the designer. The logic diagram of a BCD ripple counter is shown below;

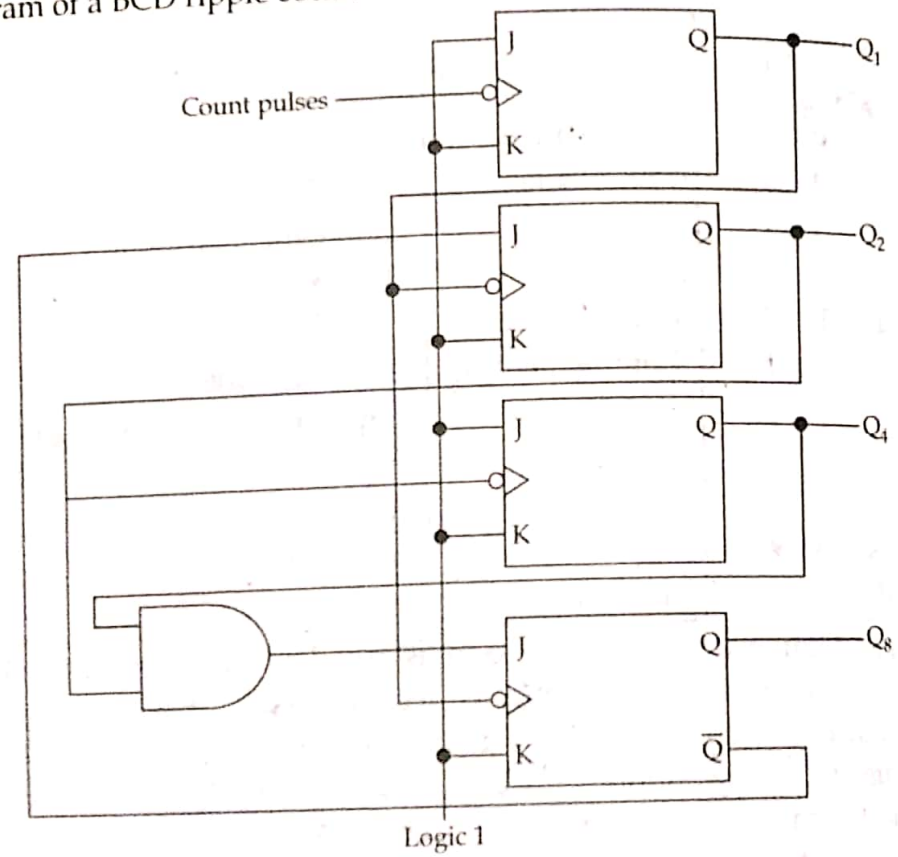


Figure: BCD Ripple Counter

The four outputs are designated by the letter symbol Q with a numeric subscript equal to the binary weight of the corresponding bit in the BCD code. The flip-flops trigger on the negative edge *i.e.*, when the CP signal goes from 1 to 0. Note that the output Q_1 is applied to the CP inputs of both Q_2 and Q_8 and the output of Q_2 is applied to the CP input of Q_4. The J and K inputs are connected either to a permanent 1 signal or to outputs of flip-flops.

succession as in a ripple counter. The decision whether a flip-flop is to be complemented or not is determined from the values of the J and K inputs at the time of the pulse. If J = K = 0, the flip-flop remains unchanged. If J = K = 1, the flip-flop complements.

A. Binary Counter

In a synchronous binary counter, the flip-flop in the lowest order position is complemented with every pulse. This means that its J and K inputs must be maintained at logic 1. A flip-flop in any other position is complemented with a pulse provided all the bits in the lower order positions are equal to 1, because the lower order bits (when all 1's) will change to 0's on the next count pulse. The binary count dictates that the next higher order bit be complemented. Synchronous binary counters have a regular pattern and can easily be constructed with complementing flip-flops and gates.

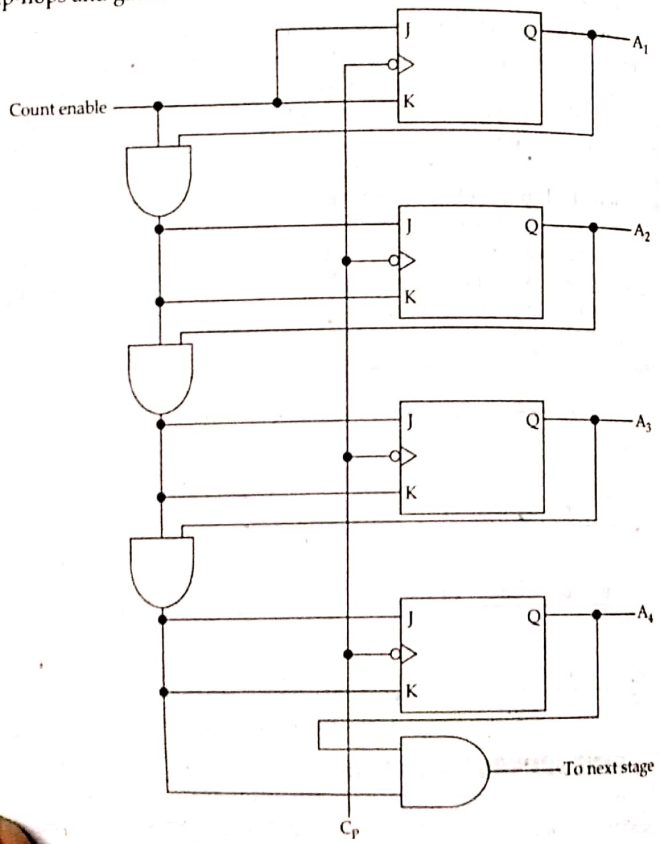


Figure: 4-bit synchronous binary counter

The CP terminals of all flip-flop are connected to a common clock-pulse source. The first stage A_1 has its J and K equal to 1 if the counter is enabled. The other J and K inputs are equal to 1 if all previous low order bits are equal to 1 and the count is enabled. The chain of AND gets generates the required logic for the J and K inputs in each stage. The counter can be extended to any number of stages, with each stage having an additional flip-flop and an AND gate that gives an output of 1 if all previous flip-flop outputs are 1's. Note that the flip-flops trigger on the negative edge of the pulse. This is not essential here as it was with the ripple counter. The counter could also be triggered on the positive edge of the pulse.

B. Binary UP-Down Counter

In a synchronous count-down binary counter, the flip-flop in the lowest order position is complemented with every pulse. A flip-flop in any other position is complemented with a pulse provided all the lower order bits are equal to 0. A count-down binary counter can be constructed as a binary counter, except that the inputs to the AND gates must come from the complement outputs $\overline{Q}$ and not from the normal outputs Q of the previous flip-flops. The two operations can be combined in one circuit.

C. BCD Counter

A BCD counter counts in binary coded from 0000 to 1001 and back to 0000. Because of the return to 0 after a count of 9, a BCD counter does not have a regular pattern as in a straight binary count. The excitation table of BCD counter is given below;

Present state				Next state				Output	Flip-Flop inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	Y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

The excitation for the T flip-flops is obtained from the present and next state conditions. An output Y is also shown in the table. This output is

equal to 1 when the counter present state is 1001. In this way, Y can enable the count of the next-higher order decade while the same pulse switches the present decade from 1001 to 0000. The flip-flop input functions from the excitation table can be simplified by means of maps. The unused states for minterms 10 to 15 are taken as don't care terms. The simplified functions are;

$$TQ_1 = 1$$
$$TQ_2 = \bar{Q}_8 Q_1$$
$$TQ_4 = Q_2 Q_1$$
$$TQ_8 = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$Y = Q_8 Q_1$$

The circuit can be easily drawn with four T flip-flops, five AND gates and one OR gate.

8.5 JOHNSON COUNTER

A Johnson or moebius counter is a switch-tail ring counter with 2K decoding gates to provide outputs for 2K timing signals. A K-bit ring counter circulates a single bit among the flip-flops to provide K distinguishable states. The number of states can be doubled if the shift register is connected as a switch-tail ring counter. A switch tail ring counter is a circular shift register with the complement output of the last flip-flop connected to the input of the first flip-flop. Figure below shows such a type of shift register.

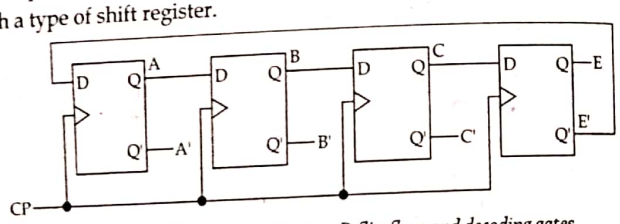


Figure: A 4-bit Johnson counter using D flip-flops and decoding gates

The circuit connection is made from the complement of the right most flip-flop to the input of the left most flip-flop. The register shifts its contents once to the right with every clock pulse and at the same time, the complement value of the E flip-flop is transferred into the A flip-flop. Starting from a cleared state, the switch-tail ring counter goes through a sequence of eight states. In general a K-bit switch-tail counter will go through 2K states. Starting with all 0s each shift operation inserts 1s from the left until the register is filled with all 1s. In the following sequences. 0s are inserted from the left until the register is again filled with all 0s.

Table: Count sequence of a 4-bit Johnson counter

Sequence number	Flip-Flop output			
	A	B	C	D
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0
5	1	1	1	1
6	0	1	1	1
7	0	0	1	1
8	0	0	0	1

The decoding of a K-bit switch tail ring counter to obtain 2K timing sequence follows a regular pattern. The all 0s state is decoded by taking the complement of the two extreme flip-flops. The all 1's state is decoded by taking the normal outputs of the two extreme flip-flops. All other states are decoded from an adjacent 1, 0 or 0,1 pattern in the sequence. One disadvantage of 4-bit Johnson counter using D flip-flops is that, if it finds itself in an unused state, it will persist in moving from one invalid state to another and never find its way to a valid state. The difficulty can be corrected by modifying the circuit to avoid this undesirable condition. One correcting procedure is to disconnect the output from flip-flop B that goes to the D input of flip-flop C, and instead enable the input of flip-flop C by the function;

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

where, DC is the flip-flop input function for the D input of the flip-flop C.

8.6 RANDOM ACCESS MEMORY (RAM)

A memory unit is a collection of storage cells together with associated circuits needed to transfer information in and out of the device. Memory cells can be accessed for information transfer to or from any desired random location and hence the name random access memory, abbreviated as RAM.

A memory unit stores binary information in groups of bits called words. A word in memory is an entity of bits that move in and out of storage as a unit. A memory word is a group of 1's and 0's and may represent a number, an instruction, one or more alphanumeric characters, or any other binary coded information. A group of eight bits is called a byte. Most computer memories use words that are multiples of 8 bits in length. Thus, a 16-bit word contains two bytes, and a 32-bit word is made up of four bytes. The capacity of a memory unit is usually stated as the total number of bytes that it can store. RAMs are basically sequential circuits (flip-flops). When the power is switched off, the data stored in a RAM is lost. Hence, RAMs are also called volatile memories. RAMs are read-write-erasable memories.

The basic, single unit of RAM consists of a J-K flip-flop and associated AND gates. Initially, we clear Q to zero. When a data is to be written, an AND gates 1 and 2 are enabled by applying positive pulse to their respective enable inputs. Now, a 1 is applied to the address line A and data is entered through the write -input line W.

If W = 0, J = 0 and K = 1, then Q = 0

If W = 1, J = 1 and K = 0, then Q = 1

Thus whatever be the data on the line, the J-K flip flops will store it. To read, we again address the cell by applying a pulse at A. AND 3 is now enabled to connect Q to the output.

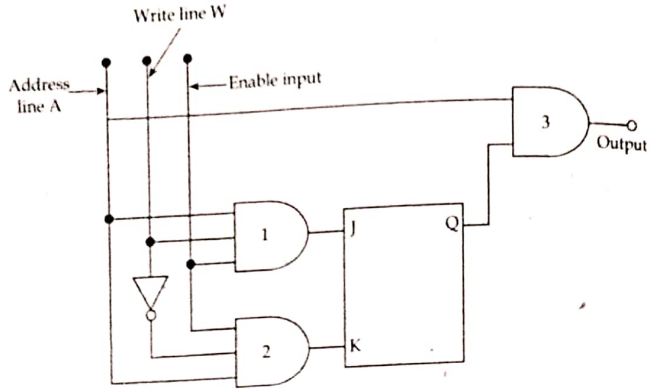


Figure: Basic RAM cell

The memory unit is specified by the number of words it contains and the number of bits in each word. The address lines select one particular word. Each word in memory is assigned an identification number, called an address, starting from 0 and continuing with 1, 2, 3 up to $2^K - 1$, where K is the number of address lines. The selection of a specific word inside the memory is done by applying the K-bit binary address to the address lines. A decoder inside the memory accepts this address and opens the paths needed to select the word specified. Computer memories may range from 1024 words requiring an address of 10 bits to 2^{32} words, requiring 32 address bits.

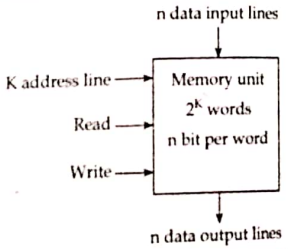


Figure: Block diagram of a memory unit

Write and Read Operations

The two operations that a RAM can perform are the write and read operations. The write signal specifies a transfer-in operation and the read

signal specifies a transfer-out operation. On accepting one of these control signals, the internal circuits inside the memory provide the desired operations. The step that must be taken for the purpose of transferring a new word to be stored into memory are as follows;

1. Transfer the binary address of the desired word to the address lines.
2. Transfer the data bits that must be stored in memory to the data input lines.
3. Activate the write input.

The memory unit will then take the bits from the input data lines and store them in the word specified by the address lines. The steps that must be taken for the purpose of transferring a stored word out of memory are as follows;

1. Transfer the binary address of the desired word to the address line
2. Active the read input

The memory unit will then take the bits from the word that has been selected by the address and apply them to the output data lines. The content of the selected word does not change after reading.

Integrated-circuit RAM units are available in two possible operating modes, static and dynamic. The static RAM consists essentially of internal flip-flops that store the binary information. The stored information remains valid as long as power is applied to the unit. The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors. The capacitors are provided inside the chip by MOS transistors. The stored charge on the capacitors tends to discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory. Refreshing is done by cycling through the words every few milliseconds to restore the decaying charge. Dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip, but static RAM is easier to use and has shorter read and write cycles.

ERROR CORRECTION CODES

The complexity level of a memory array may cause occasional errors in storing and retrieving the binary information. The reliability of a memory unit may be improved by employing error-detecting and correcting code. The most common error-detection scheme is the parity bit. A parity bit is generated and stored along with the data word in memory. The parity of the word is checked after reading it from memory. The data word is accepted if the parity sense is correct. If the parity checked results in an inversion, an error is detected, but it cannot be corrected.

An error correcting code generates multiple checks bits that are stored with the data word in memory. Each check bit is a parity over group of bits in the data word. When the word is read from memory, the associated parity bits are also read data. If the check bits compare, it

signifies that no error has occurred. If the check bits do not compare with the stored parity, they generate a unique pattern, called a syndrome that can be used to identify the bit in error. A single error occurs when a bit changes in value from 1 to 0 or from 0 to 1 during the write or read operation. If the specific bit in error is identified, them the error can be corrected by complementing the erroneous bit.

HAMMING CODE

One of the most common error-correcting codes used in random-access memories was devised by R.W. Hamming. In the Hamming code, K parity bits are added to an n-bit data word, forming a new word of n + K bits. The bit positions are numbered in sequence from 1 to n + K. Those positions numbered as a power of 2 are reserved for the parity bits. The remaining bits are the data bits. The code can be used with words of any length.

The Hamming code can be used for data words of any length. In general, the Hamming code consists of K check bits and n data bits for a total of n + K bits. The syndrome value C consists of k bits and has a range of 2^k values between 0 and $2^k - 1$. One of these values, usually zero, is used to indicate that no error was detected, leaving $2^k - 1$ values to indicate which of the n + K bits was in error. Each of these $2^k - 1$ values can be used to uniquely describe a bit in error. Therefore, the range of K must be equal to or greater than n + K, giving the relationship.

$$2^k - 1 \geq n + K$$

Solving for n in terms of K, we obtain,

$$2^k - K \geq n$$

This relationship gives a formula for evaluating the number of data bits that can be used in conjunction with K check bits. For example, when K = 3, the number of data bits that can be used is $n \leq (2^3 - 1 - 3) = 4$. For K = 4, we have $2^4 - 1 - 4 = 11$, giving $n \leq 11$. The data word may be less than 11 bits, but must have at least 5 bits, otherwise, only 3 check bits will be needed. Range of n for various bits will be needed. Range of n for various values of K are listed in table below.

Range of Data bits for K check bits

Number of check bits, K	Range of data bits n
3	2 - 4
4	5 - 11
5	12 -26
6	27 - 57
7	58 - 120

The Hamming code can detect and correct only a single error. Multiple errors are not detected. By adding another parity bit to the coded word

the Hamming code can be used to correct a single error and detect double errors. Consider, for example, the 8-bit data word 11000100, we include four parity bits with the 8-bit word and arrange the 12 bits as follows, Complete the sequence to get following;

Bit position:	1	2	3	4	5	6	7	8	9	10	11	12
	P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0

The four parity bits, P_1, P_2, P_4 and P_8 are in positions 1, 2, 4 and 8 respectively. The eight bits of the data word are in the remaining positions. Each parity bit is calculated as follows;

P_1 = XOR of bits (3, 5, 7, 9, 11) = $1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$

P_2 = XOR of bits (3, 6, 7, 10, 11) = $1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$

P_4 = XOR of bits (5, 6, 7, 12) = $1 \oplus 0 \oplus 0 \oplus 0 = 1$

P_8 = XOR of bits (9, 10, 11, 12) = $0 \oplus 1 \oplus 0 \oplus 0 = 1$

Remember that the exclusive-OR operation performs the odd function. It is equal to 1 for an odd number of 1's in the variables and to 0 for an even number of 1's. Thus, each parity bit is set, so that the total number of 1's in the checked positions, including the parity bit is always even. The 8-bit data word is stored in memory together with the 4 parity bits as a 12-bit composite word. Substituting the four P bits in their proper positions, we obtain the 12-bit composite word stored in memory.

Bit position:	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	1	1	1	0	0	1	0	1	0	0

When the 12 bits are read from memory, they are checked again for possible errors. The parity is checked over the same combination of bits including the parity bit. Four check bits are evaluated as follows:

C(1 = XOR of bits (1, 3, 5, 7, 9, 11)

C(2 = XOR of bits (2, 3, 6, 7, 10, 11)

C(3 = XOR of bits (4, 5, 6, 7, 12)

C(8 = XOR of bits (8, 9, 10, 11, 12)

A 0 check bit designates an even parity over the checked bits and a 1 designates an odd parity. Since the bits were stored with even parity, the result, $C = C_8C_4C_2C_1 = 0000$, indicates that no error has occurred. However, if $C \neq 0$, then the 4-bit binary number formed by the check bits give the position of the erroneous bit. *For example*; consider the following three cases:

Bit position:	1	2	3	4	5	6	7	8	9	10	11	12	
	0	0	1	1	1	0	0	1	0	1	0	0	No error
	1	0	1	1	1	0	0	1	0	1	0	0	Error in bit 1
	0	0	1	1	0	0	0	1	0	1	0	0	Error in bit 5

In the first case, there is no error in the 12-bit word. In the second case, there is an error in bit position number 1 because it changed from 0 to 1. The third case shows an error in bit position 5 with a change from 1 to 0

Evaluating the XOR of the corresponding bits, we determine the four check bits to be as follows;

	C_8	C_4	C_2	C_1
For no error :	0	0	0	0
With error in bit 1 :	0	0	0	1
With error in bit 5 :	0	1	0	1

Thus, for no error, we have C = 0000; with an error in bit 1, we obtain C = 0001; and with an error in bit 5, we get C = 0101. The binary number of C, when it is not equal to 0000, gives the position of the bit in error. The error can be corrected by complementing the corresponding bit. Note that an error can occur in the data word or in one of the parity bits.

If we include additional parity bit, then the previous 13-bit coded word becomes 0011100010100 P_{13}, where P_{13} is evaluated from the exclusive -OR of the other 12 bits. This produces the 12-bit word 0011100101001 (even parity). When the 13-bit word is read from memory, the check bits are evaluated and also the parity P over the entire 13 bits. If P = 0, the parity is correct (even parity), but if P = 1, then the parity over the 13 bits is incorrect (odd parity). The following four cases can occur;

If C = 0 and P = 0; No error occurred.

If C ≠ 0 and P = 1; A single error occurred, which can be corrected.

If C ≠ 0 and P = 0; A double error occurred, which is detected but cannot be corrected.

If C = 0 and P = 1; An error occurred in the P_{13} bit.

Note that this scheme cannot detect more than two errors.

8.8 HAZARDS

A Hazard is defined as the unwanted switching transient that can occur in a digital logic circuit. This transient is produced as a result of the varying propagation delays in different paths from the input to the output of the circuit. It makes the output of the circuit momentarily to switch on between its complemented and uncomplemented values. Hazards can occur in both combinational as well as sequential logic circuits. In general, hazards in combinational logic circuits are not as serious as those in the sequential logic circuits.

There are three types of hazards; static hazards, dynamic hazards and essential hazards. In general, these hazards may be caused due to single-input variable changes or multiple-input variable changes. The single-input variable changes produces logic hazards and can occur due to the varying propagation delays in the networks used to implement a given logic function. However the multiple-input variations do not require a network-propagation delay to produce hazard. Instead, the multiple input variations themselves produce hazard. Such hazards are called function hazards.

A. Static logic Hazard

Static hazards, in general can be sub-classified into static-1 hazard and static-0 hazard. Static-1 occur in SOP circuits whereas static-0 hazards occur in POS circuits.

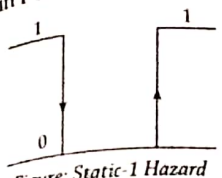


Figure: Static-1 Hazard

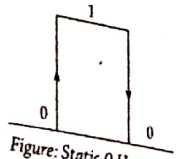


Figure: Static-0 Hazard

In static-1 hazards, the regular output of the network is a logical 1. However, during a hazardous condition, the output changes from 1 to 0 and then back to 1. The term static-1 indicates that the steady state output of the circuit is logical 1.

B. Dynamic Logic Hazard

This occurs during a transition from 0 to 1 (or 1 to 0). In the process of the transition, the output may produce a 0-1-0-1 (or 1-0-1-0) transient. The occurrence of a dynamic hazard can be expected in the complemented and uncomplemented forms.

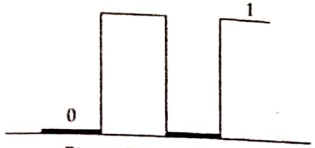


Figure: Dynamic logic Hazard

Here, also the reason for the hazard is the differing propagation delays of various gates used to generate the given function.

C. Essential Hazards

The third variety of hazards is called the essential hazard. While static and dynamic hazards can occur in combinational as well as sequential circuits, essential hazards occur in sequential circuits only. A critical race between an input signal change and a feedback signal change-may cause an incorrect state transition. Incorrect behaviour depends upon specific delays in gates interconnection. "If from any stable state, the final state reached after one change in an input is different to that reached after three changes in that input, then an essential hazard exists". The only way to prevent essential hazards is to introduce the same amount of delays in all the paths from input to output. For this, extra gates and delay circuits a can be used.

EXAMINATION QUESTION SOLUTIONS

1. **Write short notes on output Hazard Race.** [2011/F, 2013/F, 2013/S, 2014/F, 2015/S]

Solution: See the topic 8.8.

2. **Write short notes on Random Access memory.** [2015/F, 2018/S]

Solution: See the topic 8.6.

3. **Define shift register. Explain operation of parallel in parallel out shift register.** [2011/F, 2011/S, 2012/S, 2018/F]

Solution: See the topic 8.2 and 8.2. D.

4. **What is a shift register? With the help of timing diagram explain the operation of serial-in-serial-out shift register.** [2012/S, 2013/S, 2016/S]

Solution: See the topic 8.2. A.

5. **Discuss the purpose of shift register. Explain serial in parallel out shift register.** [2012/F, 2018/F, 2019/S]

Solution:

A shift register is a digital memory circuit found in calculators, computers and data-processing systems. They are commonly used in converters that translate parallel data to serial data or vice-versa. They can also function as delay circuits and digital pulse extenders.

See the topic 8.2. B.

6. **What is counter? Differentiate between serial in serial out register and parallel in serial out register with associated diagrams.** [2015/F, 2016/F]

Solution: See the topic 8.2. A and 8.2. C.

A counter is a device which stores the number of times a particular event or process has occurred, often in relationship to a clock signal. They are used in digital electronics for counting purpose.

7. **Write short notes on shift register.** [2016/F]

Solution: See the topic 8.2.

8. **Write short notes on counters.** [2018/F, 2014/F]

Solution: See the topic 8.3.

9. **Define shift register. Draw diagram for parallel in serial out shift register and discuss its operation with necessary explanation.** [2018/S, 2019/S]

Solution: See the topic 8.2 C.

10. **Describe Read and write operation in RAM with diagram.** [2014/S, 2013/S]

Solution: See the topic 8.6.

11. **Draw a circuit for 6 bit SIPO shift register.** [2014/S]

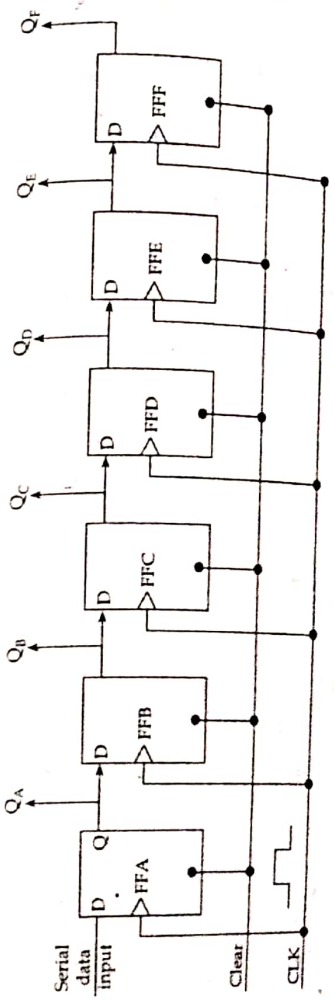
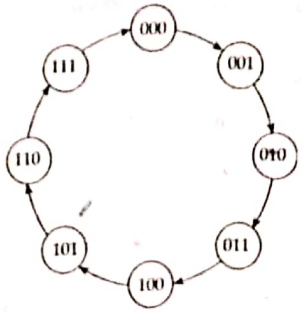


Figure: 6 fit SIPO shift register

12. **Design a 3-bit synchronous binary counter using T flip-flop.** [2011/F]
OR, **Design 3-bit up counter using T flip-flop.** [2019/S]

Solution:

Let $A_2A_1A_0$ denote the 3-bit number. The state diagram for 3 bit synchronous binary counter is shown below;



The excitation table is given below:

Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2	A_1	A_0	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

K-map;

A_2 \ A_1A_0	00	01	11	10
0	0	0	1	0
1	0	0	1	0

$\therefore$ $TA_2 = A_1 A_0$

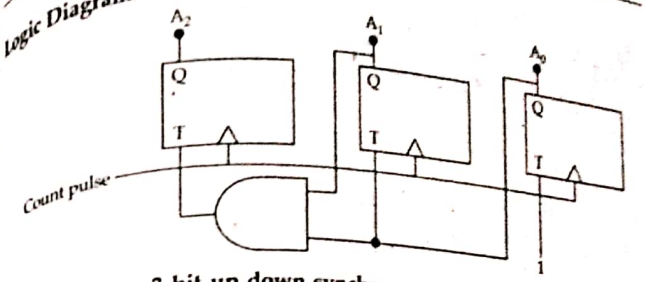
A_2 \ A_1A_0	00	01	11	10
0	0	1	1	0
1	0	1	1	0

$\therefore$ $TA_1 = A_0$

A_2 \ A_1A_0	00	01	11	10
0	1	1	1	1
1	1	1	1	1

$\therefore$ $TA_0 = 1$

Logic Diagram:



13. **Design a 3 bit up down synchronous counter using JK flip flop which is capable of counting the both up and down sequence as desired by user.** [2011/S]

Solution:

Let ABC denote 3 bit number and M denote mode select such that when M = 0 up count is done and when M = 1 down count is done. Excitation table is given below;

Present state			Input	Next state			Flip Flop Inputs					
A	B	C	M	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	0	1	0	X	0	X	1	X
0	0	0	1	1	1	1	1	X	1	X	1	X
0	0	1	0	0	1	0	0	X	1	X	X	1
0	0	1	1	0	0	0	0	X	0	X	X	1
0	1	0	0	0	1	1	0	X	X	0	1	X
0	1	0	1	0	0	1	0	X	X	1	1	X
0	1	1	0	1	0	0	1	X	X	1	X	1
0	1	1	1	0	1	0	0	X	X	0	X	1
1	0	0	0	1	0	1	X	0	0	X	1	X
1	0	0	1	0	1	1	X	1	1	X	1	X
1	0	1	0	1	1	0	X	0	1	X	X	1
1	0	1	1	1	0	0	X	0	0	X	X	1
1	1	0	0	1	1	1	X	0	X	0	1	X
1	1	0	1	1	0	1	X	0	X	1	1	X
1	1	1	0	0	0	0	X	1	X	1	X	1
1	1	1	1	1	1	0	X	0	X	0	X	1

K-map;

AB \ CM	00	01	11	10
00	0	1	0	0
01	0	0	0	1
11	X	X	X	X
10	X	X	X	X

$\therefore$ $J_A = BC\overline{M} + \overline{B}\,\overline{C}M = BC \oplus M$

AB \ CM	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	0	1
10	0	1	0	0

$\therefore K_A = BC\overline{M} + \overline{B}\,\overline{C}M = BC \oplus M$

AB \ CM	00	01	11	10
00	0	1	0	1
01	X	X	X	X
11	X	X	X	X
10	0	1	0	1

$\therefore J_B = \overline{C}M + C\overline{M} = C \oplus M$

AB \ CM	00	01	11	10
00	X	X	X	X
01	0	1	0	1
11	0	1	0	1
10	X	X	X	X

$\therefore K_B = \overline{C}M + C\overline{M} = C \oplus M$

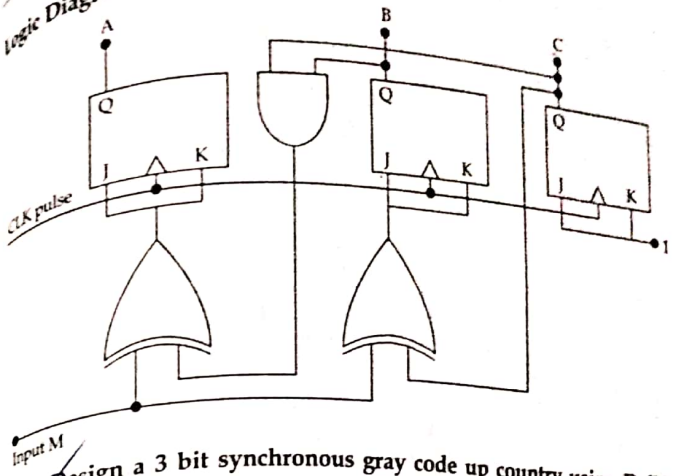
AB \ CM	00	01	11	10
00	1	1	X	X
01	1	1	X	X
11	1	1	X	X
10	1	1	X	X

$\therefore J_C = 1$

AB \ CM	00	01	11	10
00	X	X	1	1
01	X	X	1	1
11	X	X	1	1
10	X	X	1	1

$\therefore K_C = 1$

Logic Diagram:



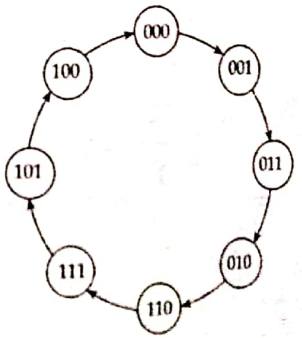
14. Design a 3 bit synchronous gray code up country using D flip-flops. [2012/S]

Solution:

Let, A, B, C denote 3 bit gray code. The excitation table of 3 bit gray code up counter using D flip-flop is given below;

Present state			Next state			Flip Flop input		
A	B	C	A	B	C	D_A	D_B	D_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	0
0	1	0	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	0	1
1	0	1	1	0	1	1	0	1
1	0	0	0	0	0	0	0	0

State diagram:



K-map:

A \ BC	00	01	11	10
0	0	0	0	1
1	0	1	1	1

$\therefore D_A = AC + B\overline{C}$

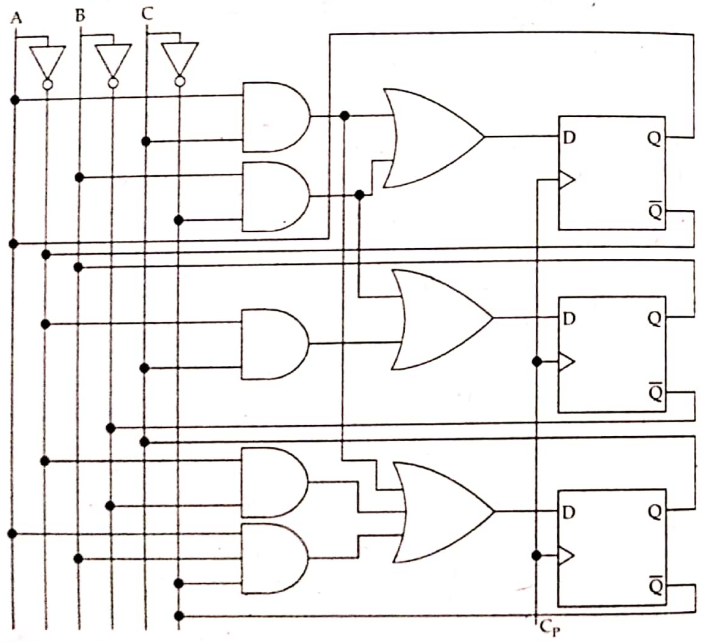
A \ BC	00	01	11	10
0	0	1	1	1
1	0	0	0	1

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

A \ BC	00	01	11	10
0	1	1	0	0
1	0	1	1	1

$\therefore D_C = \overline{A}\,\overline{B} + AB + AC$

Logic Diagram:

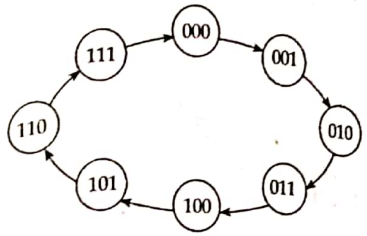


Design 3 bit synchronous binary up down counter using JK flip-flop. [2013/S]

Solution:

Let ABC denote 3 bit number

State diagram:



Excitation Table:

Present state			Next state			Flip Flop inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
0	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

K-map:

A \ BC	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J_A = BC$

A \ BC	00	01	11	10
0	X	X	X	X
1	0	0	1	0

$K_A = BC$

A \ BC	00	01	11	10
0	0	1	X	X
1	0	1	X	X

$J_B = C$

A \ BC	00	01	11	10
0	X	X	1	0
1	X	X	1	0

$\therefore K_B = C$

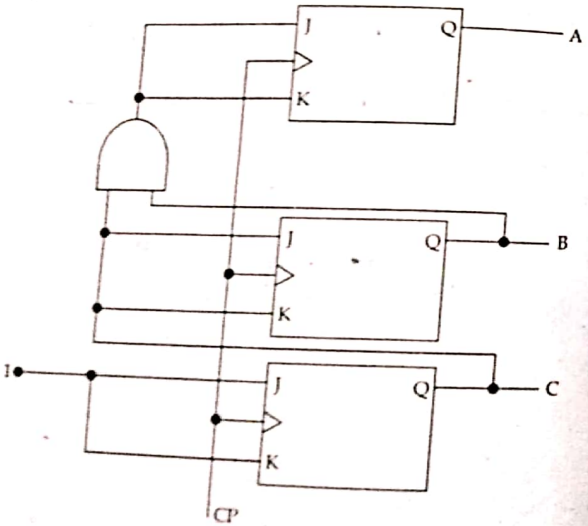
A \ BC	00	01	11	10
0	1	X	X	1
1	1	X	X	1

$\therefore J_C = 1$

A \ BC	00	01	11	10
0	X	1	1	X
1	X	1	1	X

$\therefore K_C = 1$

Logic Diagram:

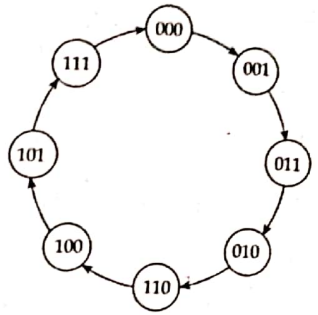


16. **Design a counter with the following binary sequence 0, 1, 3, 2, 6, 4, 5, 7 and repeat. Use T flip-flop.** **[2013/F, 2015/S]**

Solution:

The binary sequence 0, 1, 3, 2, 6, 4, 5, 7 can be represented by 3 bits. Let A, B, C represent 3 bit number.

State diagram:



Excitation Table

Present state			Next state			Flip Flop excitation		
A	B	C	A	B	C	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	1	0	0	0	1	0
1	0	0	1	0	1	0	0	1
1	0	1	1	1	1	0	1	0
1	1	1	0	0	0	1	1	1

K-map;

A \ BC	00	01	11	10
0	0	0	1	0
1	0	0	1	0

$\therefore T_A = BC$

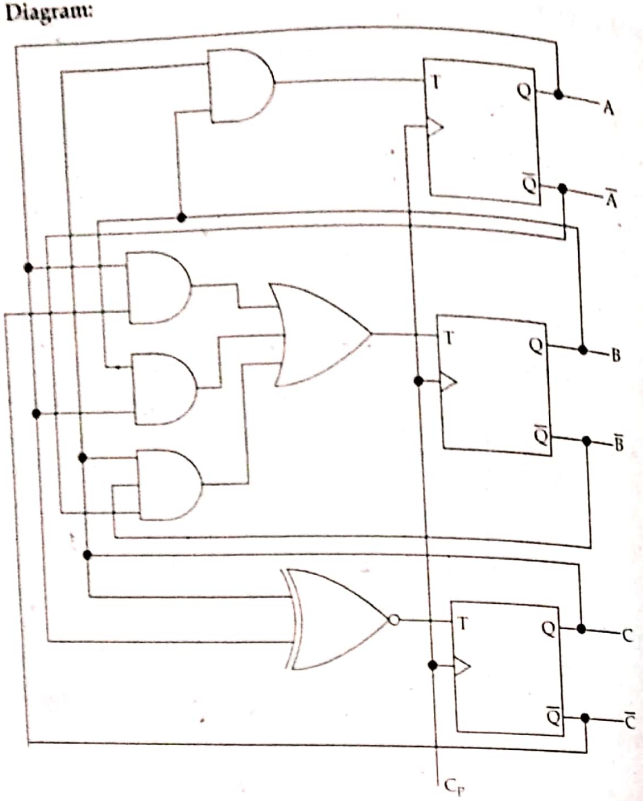
A \ BC	00	01	11	10
0	0	1	0	0
1	1	0	1	1

$\therefore T_B = A\overline{C} + AB + \overline{A}\,\overline{B}C$

A \ BC	00	01	11	10
0	1	0	0	1
1	0	1	1	0

$\therefore T_C = \overline{A}\,\overline{C} + AC = A \odot C$

Logic Diagram:



17. Write short notes on Johnson counter. **[2019/F, 2019/S]**

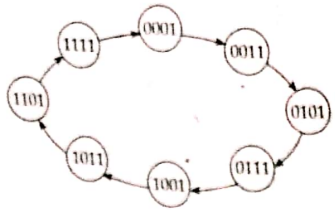
Solution: See the topic 8.5.

18. Design a synchronous 4 bit binary up counter using T flip-flop which counts all possible odd numbers. **[2017/F]**

Solution:

Let A, B, C, D denote four binary numbers. The 4 binary up counter that counts all possible odd numbers has the counting sequence as follows.

State diagram:



Excitation Table:

Present state				Next state				Flip Flop inputs			
A	B	C	D	A	B	C	D	T_A	T_B	T_C	T_D
0	0	0	1	0	0	1	1	0	0	1	0
0	0	1	1	0	1	0	1	0	1	1	0
0	1	0	1	0	1	1	1	0	0	1	0
0	1	1	1	1	0	0	1	1	1	1	0
1	0	0	1	1	0	1	1	0	0	1	0
1	0	1	1	1	1	0	1	0	1	1	0
1	1	0	1	1	1	1	1	0	0	1	0
1	1	1	1	0	0	0	1	1	1	1	0

K-map;

AB \ CD	00	01	11	10
00	X	0	0	X
01	X	0	1	X
11	X	0	1	X
10	X	0	0	X

$\therefore T_A = BC$

AB \ CD	00	01	11	10
00	X	0	1	X
01	X	0	1	1
11	X	0	1	X
10	X	0	1	X

$\therefore T_B = C$

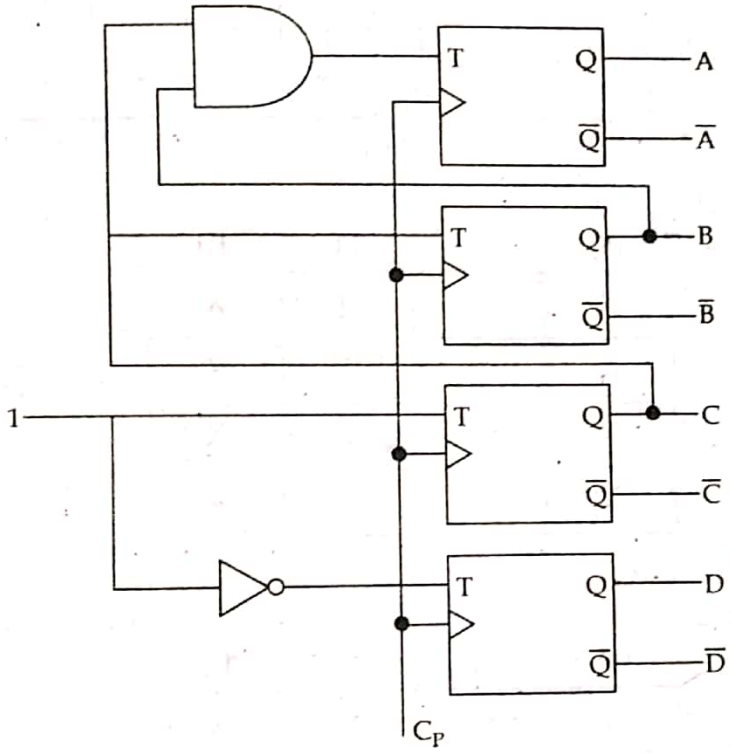
AB \ CD	00	01	11	10
00	X	1	1	X
01	X	1	1	X
11	X	1	1	X
10	X	1	1	X

$\therefore T_C = 1$

AB \ CD	00	01	11	10
00	X	0	0	X
01	X	0	0	X
11	X	0	0	X
10	X	0	0	X

$\therefore \quad T_D = 0$

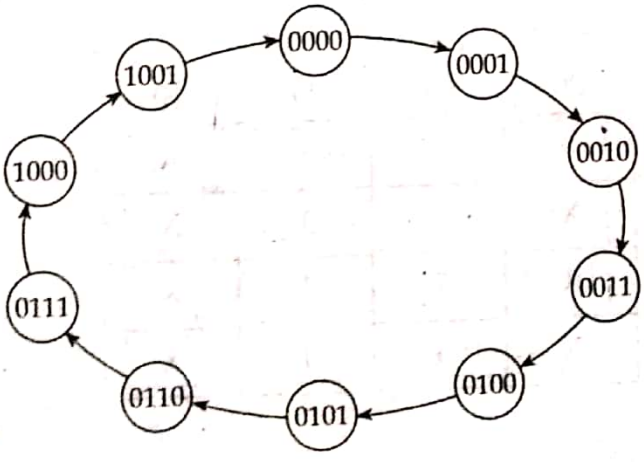
Logic Diagram:



19. Design a BCD counter that counts the binary sequence from 0000 to 1001 and returns to 0000 to repeat the sequence using T flip-flops. [2017/F]

Solution:

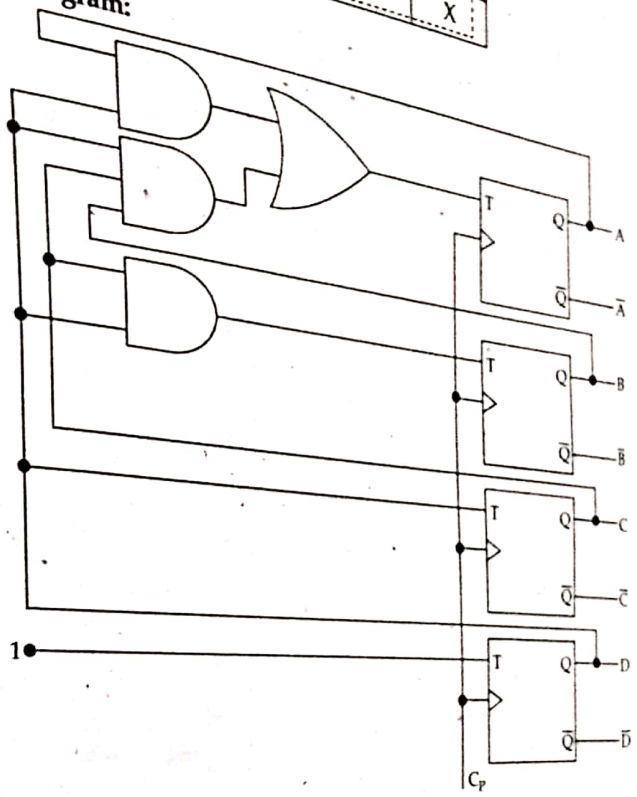
State diagram:



AB \ CD	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

$\therefore T_D = 1$

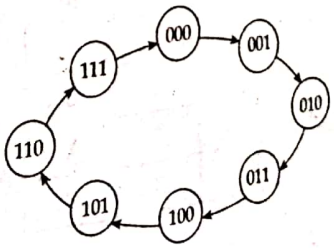
Logic Diagram:



20. Design a synchronous binary 3 bit up counter using R-S flip flop. [2017/S]

Solution:

Let ABC represents 3 bit number. The state diagram of 3 bit up counter is given below;



State Excitation table:

Present state			Next state			Flip Flop inputs					
A	B	C	A	B	C	S_A	R_A	S_B	R_B	S_C	R_C
0	0	0	0	0	1	0	X	0	X	1	0
0	0	1	0	1	0	0	X	1	0	0	1
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	1	0	0	1	0	0	1	0	1
1	0	0	1	0	1	X	0	0	1	0	1
1	0	1	1	1	0	X	0	0	X	1	0
1	1	0	1	1	1	X	0	1	0	0	1
1	1	1	0	0	0	0	1	0	1	0	1

K-map;

A \ BC	00	01	11	10
0	0	0	(1)	0
1	X	X	0	X

$\therefore S_A = \overline{A}BC$

A \ BC	00	01	11	10
0	X	X	0	X
1	0	0	(1)	0

$\therefore R_A = ABC$

A \ BC	00	01	11	10
0	0	1	0	X
1	0	1	0	X

$\therefore S_B = \overline{B}C$

A \ BC	00	01	11	10
0	X	0	1	0
1	X	0	1	0

$\therefore R_B = BC$

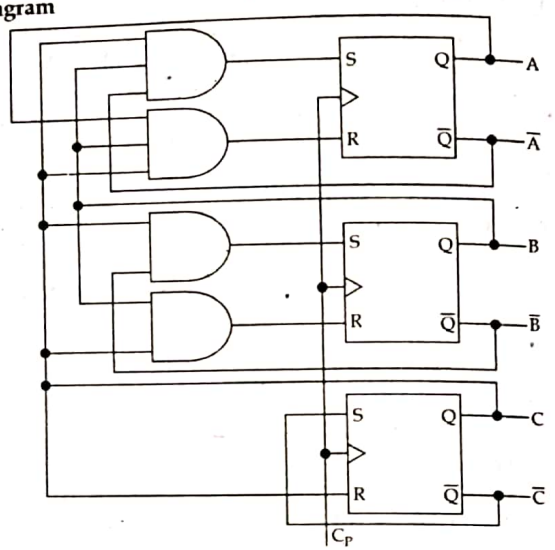
A \ BC	00	01	11	10
0	1	0	0	1
1	1	0	0	1

$\therefore S_C = \overline{C}$

A \ BC	00	01	11	10
0	0	1	1	0
1	0	1	1	0

$\therefore R_C = C$

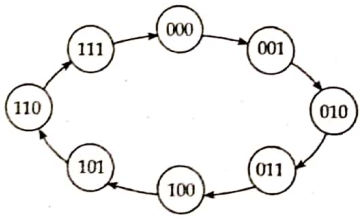
Logic Diagram



21. Design a 3-bit synchronous down counter using T-flip-flop. [2018/S]

Solution:

Let, A, B, C denote the 3 bit number. The state diagram for 3-bit synchronous down counter is given below;



State excitation table:

Present state			Next state			Flip Flop input		
A	B	C	A	B	C	T_A	T_B	T_C
0	0	0	1	1	1	1	1	1
1	1	1	1	1	0	0	0	1
1	1	0	1	0	1	0	1	1
1	0	1	1	0	0	0	0	1
1	0	0	0	1	1	1	1	1
0	1	1	0	1	0	0	0	1
0	1	0	0	0	1	0	1	1
0	0	1	0	0	0	0	0	1

K-map;

A \ BC	00	01	11	10
0	1	0	0	0
1	1	0	0	0

$\therefore T_A = \overline{B}\,\overline{C}$

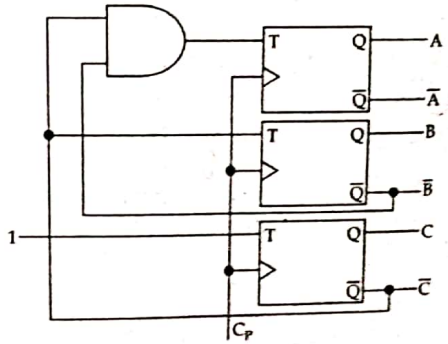
A \ BC	00	01	11	10
0	1	0	0	0
1	1	0	0	1

$\therefore T_B = \overline{C}$

A \ BC	00	01	11	10
0	1	1	1	1
1	1	1	1	1

$\therefore T_C = 1$

Logic diagram:



22. Design a synchronous MOD-6 counter using clocked D flip-flop. [2018/S]

Solution:

The counter with n flip flops has maximum MOD number 2^n. *For example;* 3 bit binary counter is a MOD 8 counter. This basic counter can be modified to produce MOD numbers less than 2^n by allowing the counter to skip states those are normally part of counting sequence.

Flip-flops required are = $2^n \geq N$

Here, $N = 6 \therefore n = 3$

i.e., three flip-flops are required

Transition Table:

Present state			Next state		
Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	X	X	X
1	1	1	X	X	X

K-map;

Q_A \ Q_BQ_C	00	01	11	10
0	0	0	1	0
1	1	0	X	X

$\therefore D_A = Q_A\overline{Q}_C + Q_BQ_C$

Q_A \ Q_BQ_C	00	01	11	10
0	0	1	0	1
1	0	0	X	X

$\therefore D_B = \overline{Q}_A\overline{Q}_BQ_C + Q_B\overline{Q}_C$

Q_A \ Q_BQ_C	00	01	11	10
0	1	0	0	1
1	1	X	X	X

$\therefore D_C = \overline{Q}_C$

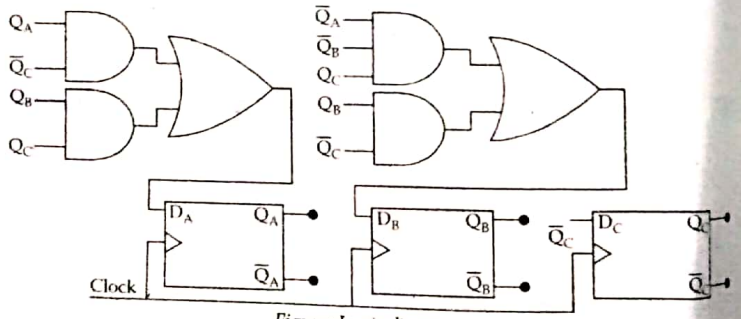


Figure: Logic diagram

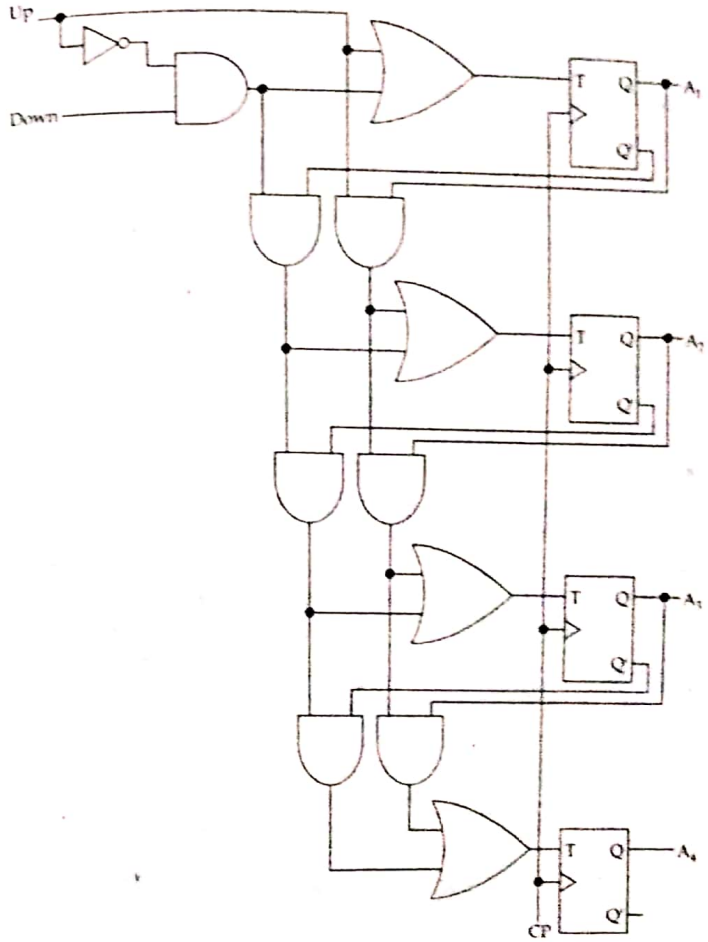
23. Design a BCD synchronous up counter using T-flip flop. [2015/F]

Solution: See the Question number 21.

24. Design 4 bit up down counter using T flip-flop. [2012/F, 2016/F]

Solution:

A binary counter capable of counting either up or down is shown in figure below:



The T flip flops employed in this circuit may be considered as JK flip flops with the J and K terminals tied together. When the up input control is 1 and the up input is 0, the circuit counts down, since the complemented the T inputs. When the up and down inputs are both 0, the circuit does not change state state but remains in the same count. When the up and down inputs are both 1, the circuit counts up. This ensures that only one operation is performed at any given time.

25. Explain in detail about synchronous up/down counter.

Solution: See the Question number 24.

26. Design MOD-6 Counter using Clocked T Flip Flop.

Solution:

Flip Flops required are: $2^n \geq N$

Here, $N = 6$ $\therefore$ $n = 3$

i.e., three flip flops are required.

Excitation table for T flip-flop

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Transition table:

Present state			Next state			Flip flop inputs		
Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	0	0	0	1	0	1
1	1	0	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X

K-map simplification for flip-clop inputs.

For T_A

Q_A \ Q_BQ_C	00	01	11	10
0	0	0	1	0
1	1	1	X	X

$\therefore$ $T_A = Q_AQ_C + Q_B\overline{Q}_C$

For T_B

Q_A \ Q_BQ_C	00	01	11	10
0	0	1	1	0
1	0	0	X	X

$\therefore$ $T_B = Q_A\overline{Q}_C$

For T_C

Q_A \ Q_BQ_C	00	01	11	10
0	1	1	1	1
1	1	1	X	X

$\therefore$ $T_C = 1$

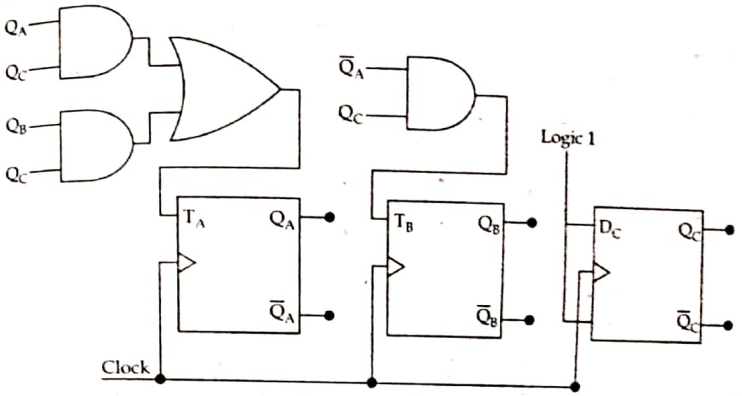


Figure: Logic diagram

27. Design Mod-6 Counter using Clocked SR Flip Flop.

Flip flips required are: $2^n \geq N$

Here, $N = 6$ $\therefore$ $n = 3$

i.e., There flip flops are required.

Excitation table:

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Transition table

Present state			Next state			Flip flop inputs					
Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	S_A	R_A	S_B	R_B	S_C	R_C
0	0	0	0	0	1	0	X	0	X	1	0
0	0	1	0	1	0	0	X	1	0	0	1
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	1	0	0	1	0	0	1	0	1
1	0	0	1	0	1	X	0	0	X	1	0
1	0	1	0	0	0	0	1	0	X	0	1
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

K-map simplification for flip-flop inputs.

For S_A

Q_A \ Q_BQ_C	00	01	11	10
0	0	0	1	0
1	X	0	X	X

$\therefore \quad S_A = Q_BQ_C$

For S_B

Q_A \ Q_BQ_C	00	01	11	10
0	0	1	X	0
1	0	0	X	X

$\therefore \quad S_B = \overline{Q}_AQ_C$

For S_C

Q_A \ Q_BQ_C	00	01	11	10
0	1	0	0	1
1	1	0	X	X

$\therefore \quad S_C = \overline{Q}_C$

For R_A

Q_A \ Q_BQ_C	01	01	11	10
0	X	X	0	X
1	0	1	X	X

$\therefore \quad R_A = \overline{Q}_BQ_C$

For R_B

Q_A \ Q_BQ_C	00	01	11	10
0	X	0	1	0
1	X	X	X	X

$\therefore \quad R_B = Q_BQ_C$

For R_C

Q_A \ Q_BQ_C	00	01	11	10
0	0	1	1	0
1	0	1	X	X

$\therefore \quad R_C = Q_C$

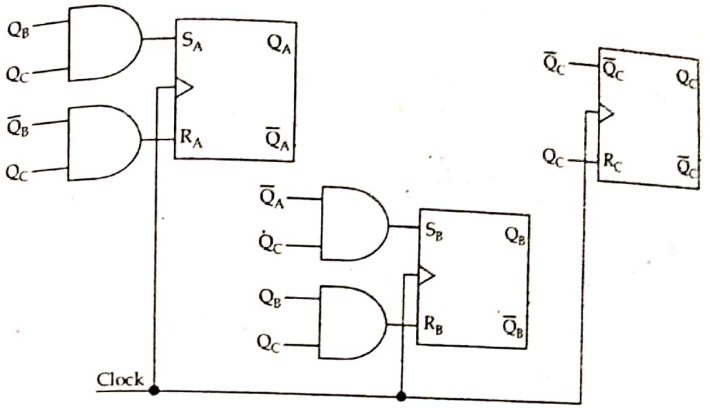


Figure: Logic diagram

28. Design a MOD-4 Dowm Counter using JK Flip Flop

Solution:

Excitation table:

Present state		Next state		Flip flop inputs			
A	B	A+	B+	J_A	K_A	J_B	K_B
0	0	1	1	1	X	1	X
0	1	0	0	0	X	X	1
1	0	0	1	X	1	1	X
1	1	1	0	X	0	X	1

K-map simplification,

For J_A

A \ B	0	1
0	1	0
1	X	X

$\therefore \quad J_A = \overline{B}$

For J_B

A \ B	0	1
0	X	X
1	1	0

$\therefore \quad K_A = \overline{B}$

For J_B

A \ B	0	1
0	1	X
1	1	X

$\therefore \quad J_B = 1$

For K_B

A \ B	0	1
0	X	1
1	X	1

$\therefore \quad K_B = 1$

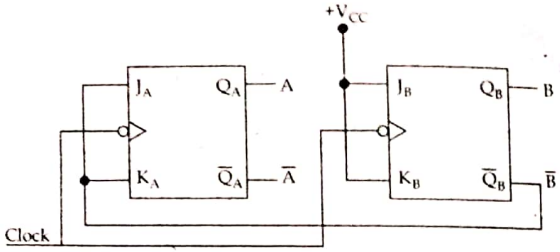


Figure: Logic diagram

CHAPTER 9

ARITHMETIC LOGIC UNITS

* * * * *

1. Write short notes on Nibble Adder. [2013/S, 2013/F, 2014/S, 2018/F]

Solution:

A nibble is a four bit aggregation or half an octet. A nibble contains 4 bits.

See the chapter 5 topic 'Full adder".

2. What is processor unit? Draw its block diagram. [2012/F]

Solution:

A processor unit is that part of a digital system or a digital computer that implements the operation in the system. It consists of a number of registers and the digital functions that implement arithmetic, logic, shift and transfer micro-operations. The processor unit on combining with a control unit is used to supervise the sequence of micro-operation and that is also termed as a central processing unit (CPU).

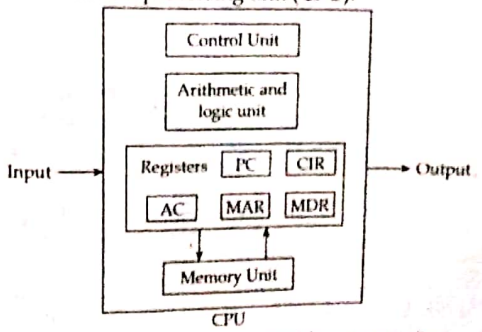


Figure: Control processing unit with processor unit

Registers are high speed storage areas in the CPU. All data must be stored in a register before it can be processed.

MAR = Memory address register – It holds the memory location of data that needs to be accessed.

MDR = Memory data register – Holds data that is being transferred to or from memory.

AC = Accumulator – It contains the address of the next instruction to be executed.

PC = Program counter – It contains the address of the next instruction to be executed.

CIR = Current instruction register – contains the current instruction during processing.

3. Write short notes on Accumulator. **[2012/S, 2017/F]**

An accumulator is a register for short term, intermediate storage of arithmetic and logic data in a computer's CPU. Intermediate results of an operation are progressively written to the accumulator overwriting the previous value.

The process of adding many numbers is carried out by initially storing these numbers in other processor register or in the memory unit of computer and accumulator is cleared to 0. The numbers are then added to the accumulator one at a time in consecutive order. The first number is added to 0 and the sum is transferred to the accumulator. The second number is added to the accumulator content and the newly formed sum replaces its previous value. This process is continued until all numbers are added and total sum is formed. Hence, the register accumulates the sum in a step by step manner by performing sequential addition between a new number and the previously accumulated sum.

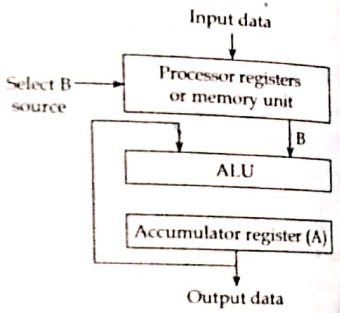


Figure: Processor unit with an accumulator register

In process unit, the accumulator register acts as a multipurpose register capable of performing not only the micro-operation addition but many other micro-operations as well. The gates associated with an accumulator register provide all digital functions found in an ALU.

The block diagram of processor unit that employs an accumulator register is shown in figure. The input B supplies one external source information which might come from other processor registers or directly from the computer with main memory. The register A supplies the other source information to the ALU at input. An operation result is transferred back to the register A and replaces its previous content. The output from the register A may got to an external destination or into the input terminals of other processor register or memory unit.

In the processor register, in order to form the sum of two numbers, it becomes necessary to add them in register A using the sequence of micro-operation mentioned below;

$T_1 : A \leftarrow 0$ Clear A

$T_2 : A \leftarrow A_1 + R_1$ Transfer R_1 to A

$T_3 : A \leftarrow A + R_2$ Add R_2 to A

First register A is cleared and the first number in R_1 is transferred into the register A by adding it to the present zero content of A. The second number in R_2 is then added to the present value of A. The sum formed in A may be used for other computations or may be transferred to a desired destination based on necessary requirement.

4. Design a 4-bit combinational logic shifter and explain it. **[2011/F]**

Solution:

A combinational logic shifter constructed with multiplexer is shown below;

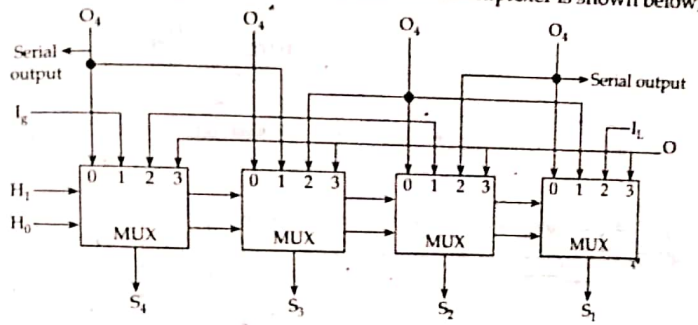


Figure: A 4-bit combinational logic shifter

Here, two selection variables H_1 and H_0 are applied to all four multiplexers to select the type of operation in shifter. With $H_1 H_0 = 00$, no shift is executed and the signal from 0 go directly to the S lines. The next two selection variable cause a shift-right operation and shift left operation. When $H_1H_0 = 11$, the multiplexer select the inputs attached to 0 and as a consequence the S output also becomes equal to 0, blocking the information transfer from ALU to the output bus.

Shifter Summarized operation is as:

$H_1 H_0$	Operation	Function
00	S ← 0	transfer 0 to S (no shift)
01	S ← shr 0	Shift right 0 into S
10	S ← shl 0	Shift left 0 into S
11	S ← 0	Transfer 0 into s

The transfer from source register to a destination register can be accomplished with one clock pulse if the shifter is implemented with combinational circuit. In a combinational logic shifter, the signals from

ALU to output bus propagate via gates without the need for a clock pulse. Hence, the only clock pulse needed in the processor system is for loading the data from the output bus into the destination register.

5. Explain briefly about status register. [2011/S, 2012/F, 2013/S, 2019/F]

Solution:

A status register, flag register or condition code register is a collection of status flag bits for a processor. It is a hardware register that contains information about the state of the processor.

C-carry bit: Set if the output carry of ALU is 1.

S-sign bit: Set if the highest order bit is 1.

Z-zero bit: Set if the ALU's output contains all zeros.

V-Overflow bit: Set if there is any overflow. For a 8-bit ALU, V is set if the result is greater than 127 and less than –128.

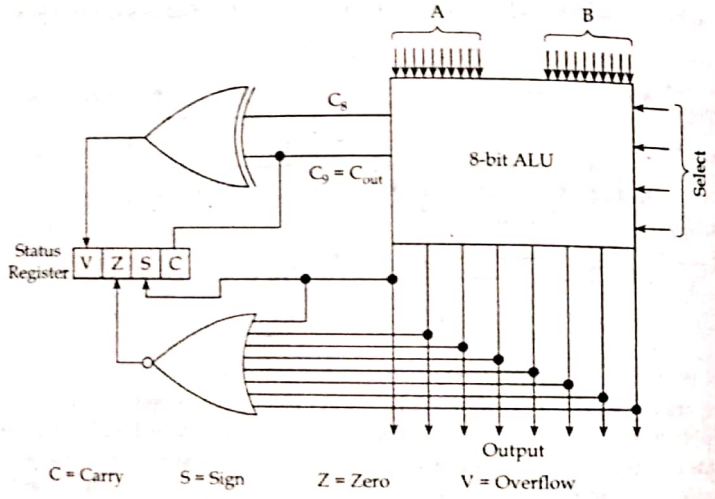


Figure: Setting bits in a status register

To determine two number relative magnitude, one should have to subtract one number from the other and then one must require to check certain bit conditions in the resultant difference. If two numbers are unsigned, the bit conditions of interest are the output carries and a possible zero results. If two numbers include a sign bit in the higher order position, the bit condition of interest are the sign of result, a zero indication and an overflow condition. Sometimes, it becomes convenient to supplement the ALU with status register where these bit conditions are stored for further analysis.

An 8-bit ALU with a 4-bit status register is shown in figure where in the four status bits are symbolized by C, S, Z and V. As a result of operation performed in the ALU, the bits are set or cleared.

i) Bit C is set if the ALU output carry becomes 1 and cleared if the output carry becomes 0.

ii) Bit S is set if the highest order bit of results in ALU output (the sign bit) becomes 1 and cleared if the highest order bit is 0.

iii) Bit Z is set if the ALU output contains all 0's and cleared otherwise Z = 1 if the result is zero and Z = 0 if the result is non-zero.

iv) Bit V is set if the carries C_8 and C_9 EX–OR becomes 1 and cleared otherwise. These conditions for overflow when the numbers are in sign 2's complement representation.

The status bits can be checked after an ALU operation to determine certain relationships that exists between A and B. If the bit V is set after the addition of two unsigned numbers, then it indicates an overflow condition. If Z is set after the two signed number addition, then it indicates that A = B. This is so because $X \oplus X = 0$, and the EX–OR of operands gives an all 0's result which set the Z-bit. A single bit in A can be checked to determine if it is 0 or 1 by making all bits except the bit in quotient and then checking the Z status bit.

6. Design a 4-bit arithmetic circuit which performs different arithmetic operations. [2011/F, 2013/S, 2014/F, 2014/S, 2015/S, 2016/S, 2017/S, 2018/F]

Solution:

A 4-bit arithmetic circuit which performs eight different operations consists of four full adder (FA) circuits. The carry into the first stage is the input carry. The fourth stage carry out is the output carry. All other carries are connected internally from one stage to the next. The selection variable S_0 and S_1 control all of B inputs to full adder circuit. The input A go directly to the other inputs of the full adder.

Operations Implemented:

Function select			Y equals	Output equals 0	Function
S_1	S_0	C_i			
0	0	0	0	A	Transfer A
0	0	1	0	A + 1	Increment of A
0	1	0	B	A + B	Add B to A
0	1	1	B	A + B + 1	Add B to A plus 1
1	0	0	$\overline{B}$	A + $\overline{B}$	Add 1's complement of B to A
1	0	1	$\overline{B}$	A + $\overline{B}$ + 1	Add 2's complement of B to A
1	1	0	All 1's	A – 1	Decrement of A
1	1	1	All 1's	A	Transfer A

The arithmetic circuit needs a combinational circuit in each stage specified by Boolean function.

$X_i = A_i$

$Y_i = B_i S_0 + \overline{B}_i S_1$

where, i = 1, 2, 3, 4, ..., n

In each stage, i must have same common selection variables S_1 and S_0. The combination circuit will be different if the circuit generates different arithmetic operations.

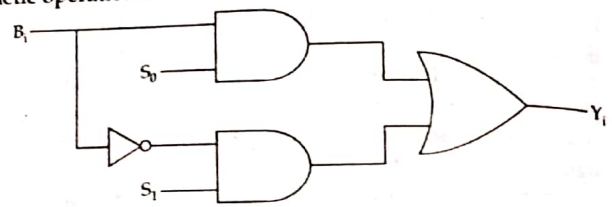
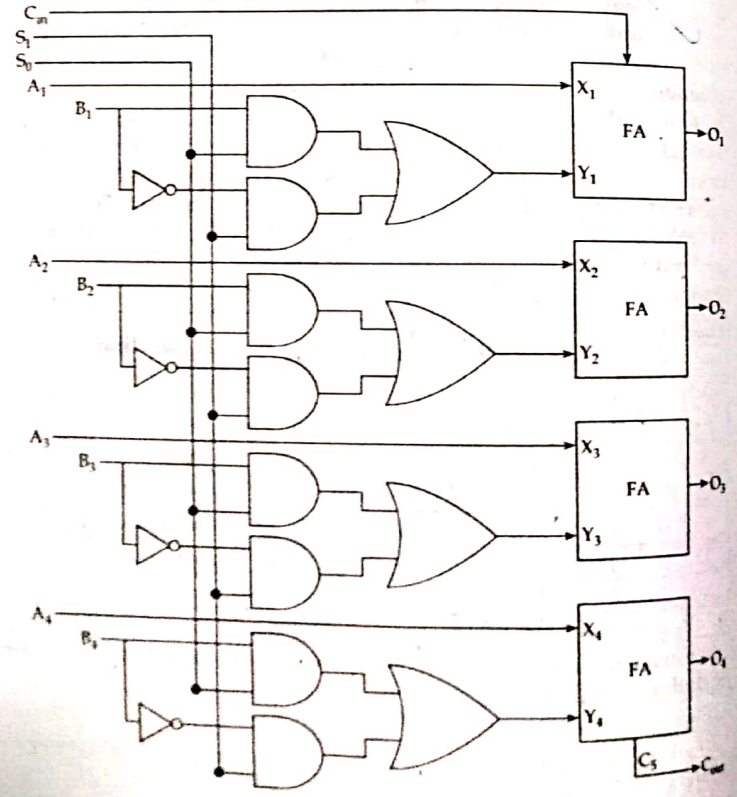


Figure: True/complement one/zero circuit

S_i	S_0	Y_i
0	0	0
0	1	B_i
1	0	$\overline{B_i}$
1	1	1

Airthmetic circuit logic design:

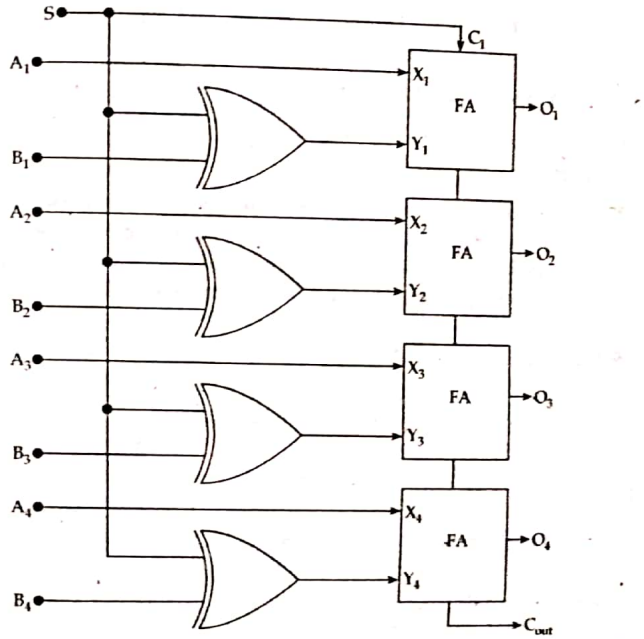


7. **Design an adder/subtractor unit with one selection variable S and two inputs A and B. When S = 0 the circuit performs A + B and when S = 1 the circuit performs A – B by taking the 2's complement of B.** **[2013/F, 2015/S, 2018/S]**

OR, **Design a 4-bit arithmetic unit that performs addition when mode control bit is 0 and subtraction when mode control bit is 1. [2019/F]**

Solution:

The 4-bit adder/subtractor circuit is shown below where in each input Bi requires an Ex-OR gate.



The selection variable S goes to each input of each gate and also to the input carry of the parallel adder. Thus, when S = 0, the circuit performs A + B and when S = 1, the circuit performs A – B by taking the 2's complement of B.

8. **Explain the process how does binary value of 4 flags in status register change with necessary diagram.** **[2015/F, 2018/S, 2019/S]**

Solution: See the Question number 5.

9. **What do you mean by ALU?** **[2017/F]**

Solution:

An arithmetic logic unit is a combinational digital electronic circuit that performs arithmetic and bitwise operations on integer binary numbers. It is a major component of the central processing unit of a computer system.

10. Design an arithmetic circuit to implement the following function table. A and B are 4 bit binary numbers. [2017/F]

S_1	S_0	C_{in}	F
0	0	0	A
0	0	1	A + 1
0	1	0	A + B
0	1	1	A + B + 1
1	0	0	A + $\overline{B}$
1	0	1	A – B
1	1	0	A – 1
1	1	1	A

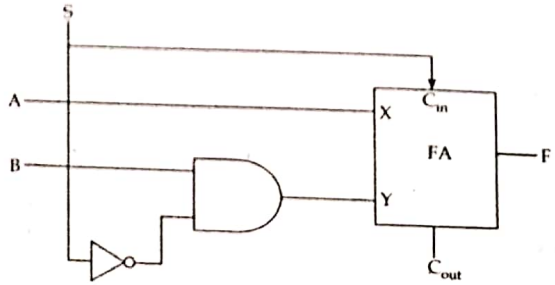
Solution: See the Question number 6.

11. Design a 4 bit arithmetic circuit. [2011/S]

Solution: See the Question number 6.

12. Design an arithmetic circuit with one selection variable and two data inputs A and B. When S = 0, the circuit performs the addition operation F = A + B and when S = 1, the circuit performs the increment operation F = A + 1. (Only show the block diagram). [2014/S]

Solution:



Here; S = Selection variable

A and B = two data inputs

Variable B is connected to input of full adder (FA) by ANDing with $\overline{S}$. S is complemented to input carry. F and C are output and out carry respectively. When, S = 0, the X input of FA receives data A. Since S = 0, $\overline{S}$ = 1. So the input Y receives data B. Hence the output will be

F = A + B + S = A + B = 0 = A + B.

When S = 1, the input X receives A, but input Y receives 0 and $\overline{S}$ = 0. So output of AND gate is 0. Also C_{in} is 1, then output is

F = A + B + S = A + 0 + 1 = A + 1

which is increment operation.

13. Design an arithmetic circuit with two selection variables S_1 and S_0 that generates the following arithmetic operations. [2012/S]

S_1	S_0	$C_{in} = 0$	$C_{in} = 1$
0	0	F = A	F = A + 1
0	1	F = A – B – 1	F = A – B
1	0	F = B – A – 1	F = B – A
1	1	F = A + B	F = A + B + 1

Solution:

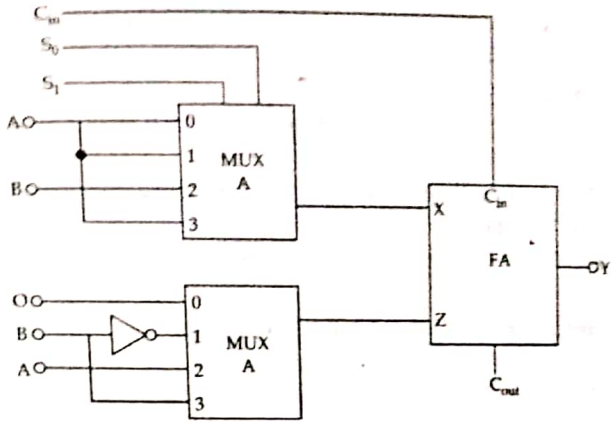


Figure: Logic diagram

Output, Y = X + Z + C_{in}

a) **$S_1S_0C_{in}$ = 000**

MUX A selects A and MUX B selects 0 and C_{in} = 0. So output will be Y = A + 0 + 0 = A. This transfers value of A to output.

b) **$S_1S_0C_{in}$ = 001**

MUX A selects A and MUX B selects 0, C_{in} 1. So, output will be Y = A + 0 + 1 = A + 1. This increments the value of A by 1.

c) **$S_1S_0C_{in}$ = 010**

MUX A selects A, MUX B selects $\overline{B}$ and C_{in} = 0. So,

Output will be Y = A + $\overline{B}$ + 0 = (A + $\overline{B}$ + 1) – 1 = A – B – 1

(Since A + $\overline{B}$ + 1 is 2's complement subtraction and is equal to A – B)

This subtracts B from A with initial borrow using 2's complement method.

d) **$S_1S_0C_{in}$ = 011**

MUX A selects A, MUX B selects $\overline{B}$ and C_{in} = 1. So output will be

Y = A + $\overline{B}$ + 1 = A – B

This performs subtraction A – B without initial borrow complement method.

e) $S_1S_0C_{in} = 100$

MUX A selects B, MUX B selects $\overline{A}$ and $C_{in}' = 0$, so output will be

$$Y = B + \overline{A} + 0 = (B + \overline{A} + 1) - 1 = B - A - 1$$

This performs subtraction B – A with initial borrow using 2's complement method.

f) $S_1S_0C_{in} = 101$

MUX A selects B, MUX B selects $\overline{A}$ and $C_{in} = 1$, so output will be

$$Y = B + \overline{A} + 1 = B - A$$

This performs subtraction B – A without initial borrow using 2's complement method.

g) $S_1S_0C_{in} = 110$

MUX A selects A, MUX B selects B and $C_{in} = 0$, so output will be

$$Y = A + B + 0 = A + B$$

This performs the addition A + B without initial carry.

h) $S_1S_0C_{in} = 111$

MUX A selects A, MUX B selects B and $C_{in} = 1$, so output will be

$$Y = A + B + 1$$

This performs addition A + B with initial carry.

CHAPTER 10

DIGITAL INTEGRATED CIRCUITS

* * * * *

* * * * *

10.1 BIPOLAR TRANSISTOR CHARACTERISTICS

A Bipolar junction transistor (BJT) is the familiar NPN or PNP junction transistor and they are constructed either with silicon or germanium semiconductor material. IC transistors however, are made with silicon. The operation of a bipolar transistor depends upon the flow of two types carriers-electrons and holes. In contrast to that, the field effect transistor (FET) is said to be unipolar, as its operation depends on the flow of only one type of majority carrier which may be electrons (N-channel) or holes (P-channel). The first five logic families - RTL, DTL, TTL, ECL and I^2L use the bipolar transistors. The last two logic families-MOS and CMOS

employ a type of unipolar transistor called a metal-oxide semiconductor field effect transistor, abbreviated MOSFET or MOS in short.

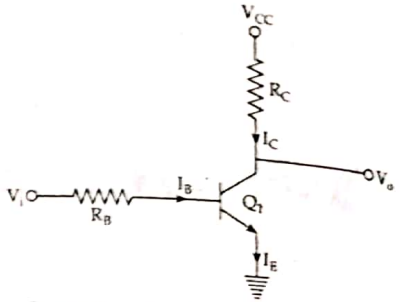
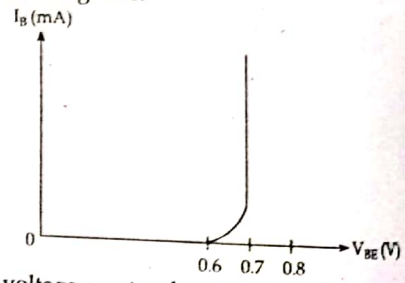


Figure: Common emitter NPN transistor

The current marked I_C flows through resistor R_C and the collector of the transistor. Current I_B flows through resistor R_B and the base of the transistor. The emitter is connected to the ground and its current $I_E = I_C + I_B$. The supply voltage is between V_{CC} and the ground. The input is between V_i and the ground and the output is between V_0 and the ground. The normal direction of currents of an NPN transistor is indicated by the arrow marks in the figure. Collector current I_C and base current I_B are assumed to be positive when they flow into the transistor and the emitter current I_E is positive when it flows out of the transistor. The symbol V_{CE} stands for the voltage difference from collector to emitter and is always positive. This junction is termed to be forward biased when V_{BE} is positive and termed as negative biased when V_{BE} is positive and termed as negative biased when V_{BE} is negative.

The base emitter characteristics of the NPN transistor is figure.



This is a plot of base current I_B with respect to V_{BE}. For a silicon-type transistor, if the base-emitter voltage is less than 0.6 V, the transistor is said to be cut-off and no base current flows. When the base-emitter junction is forward biased with a voltage greater than 0.6 V, the transistor conducts and its base current I_B starts rising very fast with a very small rise of V_{BE}. The voltage V_{BE} across the base to the emitter of a conducting transistor seldom exceeds 0.8 V.

When V_{BE} is less 0.6 V, the transistor is at cut-off and no base current flows ($I_B = 0$) and a negligible current flows in the collector. The collector to emitter circuit then behaves like an open circuit. In an active region, the collector to emitter voltage V_{CE} may be anywhere between 0.8 V to V_{CC}. Approximate collector current I_C in this region can be obtained by the relation $I_C = h_{FE} I_B$, where, h_{FE} is a transistor parameter called DC current gain. It should be noted that the maximum collector current does not depend on the I_B but on the external circuit components connected to the

collector. This is because V_{CE} is always positive and its lowest possible value is 0 V(practically minimum V_{CE} value 0.2 for silicon transistor).

The collector current-to-base current relationship $I_C = h_{FE} \cdot I_B$ is valid only when the transistor is at active region. The parameter h_{FE} varies widely over the operating range of the transistor characteristics. In a typical operating range, h_{FE} may be observed that the base current may be increased to any desirable value, but the collector current is limited by the external circuit component, as in this case it is limited to $\frac{V_{CC}}{R_C}$. As a consequence, a situation can be reached when $h_{FE} \cdot I_B$ is greater than I_C. This is the condition when the transistor is said to be in saturation region. Thus the condition for saturation is determined by the relation.

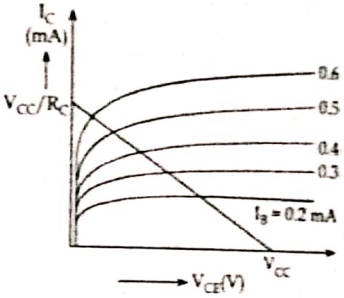


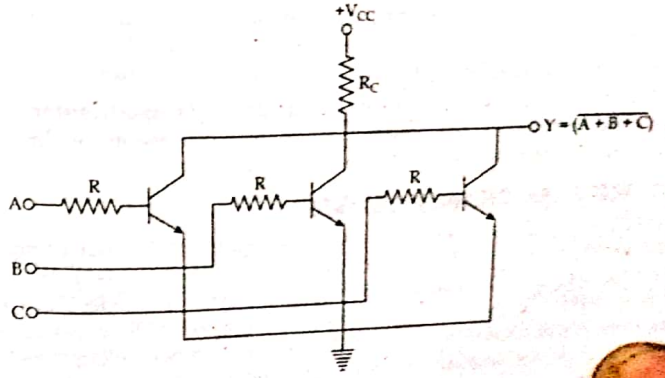
Figure: Graph of collector-emitter transistor characteristics

$$h_{FE} \cdot I_B \geq I_{CS} \text{ or } I_B \geq \frac{I_{CS}}{h_{FE}}$$

where, I_{CS} is the maximum collector current flowing during saturation and V_{CE} attains its minimum value at saturation to 0.2 V.

10.2 RESISTOR-TRANSISTOR LOGIC (RTL)

Resistor-Transistor logic or RTL is the first-generation digital logic circuit. The basic circuit of the RTL digital logic family is the NOR gate as shown in figure. Each input is associated with one resistor and a transistor. The collectors of the transistors are tied together with a common resistor to the V_{CC} supply. The output is taken from the collectors Joint. The voltage levels of the circuit are 0.2 V for low level and 1 to 3.6 V for high level.



If any of the inputs is at high level, the corresponding transistor is at saturation. This causes the output at low irrespective of the conditions of other transistors as all the transistors are connected in parallel. If all the inputs are at low level at 0.2 V, all the transistors are at cut-off condition because base to emitter voltage of all the transistor $V_{BE} < 0.6$ V, causing the output of the circuit at high level approaching the value of the supply voltage V_{CC}. Thus confirms the conditions of a NOR logic. Note that the noise margin for low signal input is 0.6 – 0.2 = 0.4 V

The RTL logic circuit has many drawbacks. The base current is practically independent of the emitter junction characteristics. The resistor increases the input resistance and reduce the switching speed of the circuit. This degrades the rise and fall times of any input pulse. Reduction in base resistor reduces any input pulse. Reduction in base resistor reduces the input resistance, increase power consumption and decrease the fan in. An approach used in practice to increase the speed of RTL circuit is connect speed-up capacitors parallel to the base resistors. In an RTL digital circuit the transistors are driven heavily to saturation resulting in long turn-off delays. The output high-voltage level reduces with the increase of load or the number of gates connected at output. Also, the collector reverse saturation current of a driver transistor at high temperature may become large enough to lower the already low output voltage.

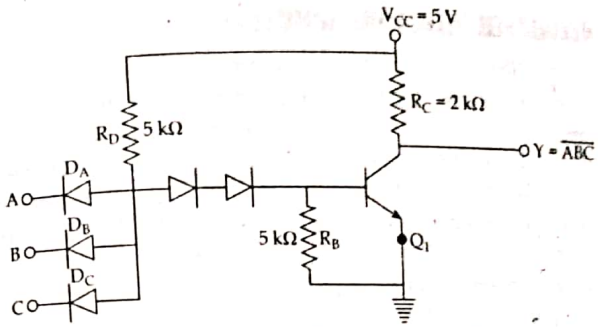
The fan out of the RTL gate is limited by the value of the output voltage when high. As the output is loaded with inputs of other gates more current is consumed by the load. This current must flow through R_C (typical value of R_C is 640 Ω). Assuming h_{FE} drops to the value of 20, with each of the base resistor value R = 450 Ω and V_{CC} = 3.6 V, the output voltage drops to 1 V when the fan out is 5. Any voltage below 1 V at the output may not drive the next transistor into saturation as required.

The following are the characteristics of the RTL family;

i) Speed of operation is low *i.e.*, the propagation delay is high up to the order of 500 ns. It cannot operate at more than 4 MHZ.
ii) Fan out is 4 or 5 with a switching delay of 50 ns and fan in is 4.
iii) Poor noise immunity.
iv) Sensitive to temperature.
v) High average power dissipation elimination of base resistors in RTL will reduce the power dissipation which results in direct-coupled transistor logic (DCTL).

10.3 DIODE TRANSISTOR LOGIC (DTL)

The DTL family eliminates the problem of decreasing the output voltage with the increase of load. The basic circuit of DTL NAND gate is shown in figure. Each input is associated with one diode. The input diodes D_A, D_B, D_C and resistor R_D form an AND gate. The transistor Q_1 serves as a current amplifier as well as a digital INVERTER. The two voltage levels are 0.2 V for the low level and 4 V for the high level.



If any input of the gate is low to 0.2 V, the corresponding diode is forward biased and conducts current through V_{CC} and R_D. The voltage at point P is equal to the input voltage 0.2 V plus one diode drop of 0.7 V for a total of 0.9 V. This is not sufficient to drive the transient Q_1 into conducting. In order for the transistor to conduct the voltage at P, it must overcome a potential of one V_{BE} (0.6 V) drop in Q_1 and two diode voltage drop (0.6 V each) across D_1 and D_2 or 0.6 + 2 × 0.6 = 1.8 V. Hence the transistor Q_1 remains at cut-off condition and its collector to emitter behaves like an open circuit. Therefore, the output voltage at collector of the transistor Y is high at 5 V (power supply V_{CC} is 5 V).

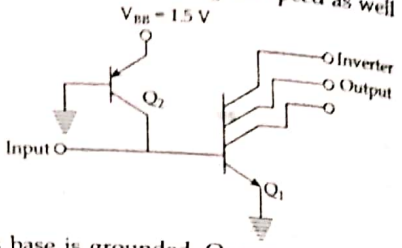
If all the inputs of the gate are high to 5 V, all the diodes are reverse biased and the current will flow through R_D, D_1, D_2 and the base of the transistor. The transistor is now driven to saturation region. The voltage at P is equal to one V_{BE} drop plus two diode drops across D_1 and D_2 or 3 × 0.7 V = 2.1 V (V_{BE} drops is 0.7 V while conducting and forward biased diode drop becomes 0.7 V). This confirms that all the input diodes are reverse biased and off. With the transistor at saturated condition, the output drops to $V_{CE(sat)}$ = 0.2 V which is low level for the gate. Thus the DTL circuit operation as above conforms NAND logic gate behaviour. The working principle is similar to any number of inputs.

The DTL family has better noise margin, higher fan out capability and faster response than the RTL family. The DTL family has the following characteristics;

i) The propagation delay of a DTL circuit is in the order of 30 ns. The turn-off delay is considerably larger than the turn on delay by a factor of 2 or 3.
ii) Fan in is at the order of 8.
iii) Fan out is as high as 8 because the input impedance is high for DTL gates due to reverse biased input diodes at logic 1 states.
iv) Noise margin is high due to two diodes D_1 and D_2 connected in series. Typically noise margin of a DTL NAND gate circuit is 0.8 V when the output is low and 3.4 V when the output is high.

10.4 INTEGRATED-INJECTION LOGIC (I²L)

Integrated Injection logic or IIL or I^2L is latest generation LSI technique also called merged Transistor logic (MTL), that uses both NPN and PNP bipolar junction Transistors to form a large number of logic gates on a single chip. It reduces the number of metal connections. This allows more circuits to be placed in a chip to form complex digital functions. It also eliminates all the resistors in the circuit thus increasing the speed as well as reducing power dissipation.



The schematic diagram of a basic I^2L inverter gate is shown in figure alongside.

It has a multiple collector transistor Q_1 and a NPN transistor Q_2 at the base of Q_1. The emitter of Q_2 is connected to the supply voltage V_{BB} and its base is grounded. Q_2 acts as a current source and active pull-up and the multiple collector NPN transistor Q_1 operates as an inverter when one or more collectors are connected with other gates. Most of the current leaving from Q_2 is injected directly to the base of Q_1 and hence the emitter of Q_2 is known as the injector and the integrated structure is called the integrated injection logic.

The operation of the basic gate can be best analyzed when it is connected to other gates.

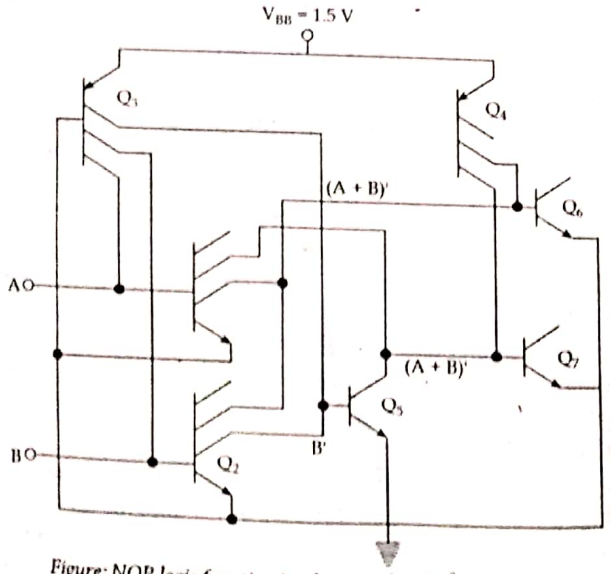
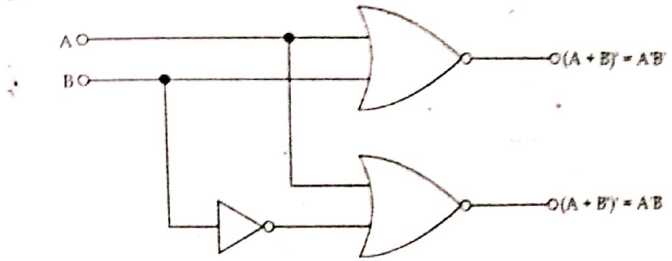


Figure: NOR logic function implemented with I^2L basic gates

Here, transistor Q_1 and Q_2 are multiple collector transistor acting as inverters. Their base currents are injected through the multiple collector NPN transistor Q_3, which is also acting as an active put up at the bases of Q_1 and Q_2. It is also acting as the active pull-up of the collector of transistor Q_2. Collectors of Q_1 and Q_2 are connected with active pull-up PNP transistor Q_4 producing a NOR function $(A + B)'$. Input signal B is complemented by the transistor Q_2 which is connected to the base of the transistor Q_5. One of the collectors of Q_5 is connected with one of the collectors of Q_1 and also to the active pull-up Q_6 producing another NOR function $(A + B)'$. OR functions may be available by the transistors Q_6 and Q_7, if they are connected to pull-up circuit or other gates. Thus we can see that two NOR functions are realized just with a few number of transistors or basic gates. The graphic symbol of the functions as produced by the figure is shown below.



The typical values of parameters of I^2L devices are as follows:

Packing density = 1500 gates/sq. mm
Gate delay: 25 to 250 ns
Supply voltage: 1 to 15 V
Logic voltage swing: 0.6 V
Power dissipation per gate: 5 mW to 75 mW

Because of high speed, low power consumption and high package density, the I^2L devices find their application mostly in LSI functions and large computers. It is not available in SSI packages containing individual gates. Its range of application include microprocessor and micro controller chips, memory devices, video games, watches, television tuning and control etc.

10.5 TRANSISTOR-TRANSISTOR LOGIC (TTL)

TTL is the most popular of all the logic families. The original basic TTL gate was a slight improvement over the DTL gate. As the TTL technology progressed, more and more additional improvements were carried out to make this logic family the most widely used type in the design of digital systems. Gates of this family possesses the highest switching speed when

compared to other logic families that utilizes the saturated transistors. TTL family or the commercially available 74/54 series, evolved into five major divisions.

i) Standard TTL (74/54 series)
ii) High-speed TTL (74 H/54 H series)
iii) Low power TTL (74 L/54 L series)
iv) Schottky diode clamped TTL (74 S/54 S series)
v) Low Power Schottky TTL (74 LS/54 LS series)

Although the high speed and low power TTL devices are designed for specific applications, all the groups of the family have several common features and are compatible and capable of interfacing directly with one another. They have the following typical characteristics in common.

i) Supply voltage is 5 V.
ii) Logic 0 output voltage level is 0 V to 0.4 V.
iii) Logic 1 output voltage level is 2.4 to 5 V.
iv) Logic 0 output voltage level is 0 V to 0.8 V.
v) Logic 1 output voltage level is 2 V to 5 V.
vi) Noise immunity is 0.4 V.

But the five different TTL series as mentioned above differ from one another in terms of propagation delay and power dissipation values. Speed power product is an important parameter for comparing the basic gates. This is a product of the propagation delay and the power dissipation measured in pico-joules. A low value of this parameter is desirable in designing the digital circuit because it indicates a low propagation delay without excessive power consumption or vice-versa.

The base TTL gate is the NAND gate shown in figure. In this basic gatem the diodes D_1 and D_2 of the DTL NAND gate is replaced with a multiple-emitter transistor T_1. In this transistor, there is only one collector and one base, as is the common practice for any BJT. However, for the multiple-emitter transistor, there will be more than one emitter. A and B represent two separate base-emitter junctions of transistor T_1 which act as two independent diodes. These are, respectively the equivalents of the diodes D_1 and D_2 of the DTL gate. The collector-base junction of T_1 forms diode D_3 of the DTL gate. The operation of this TTL gate is exactly similar to that of the DTL gate.

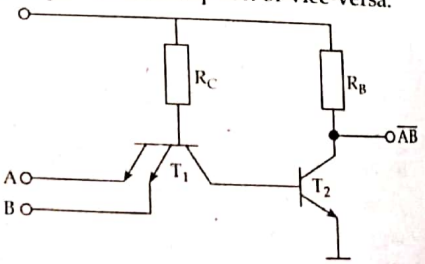


Figure: TTL NAND gate

The circuit of the TTL NAND gate available in the form of the IC is shown below.

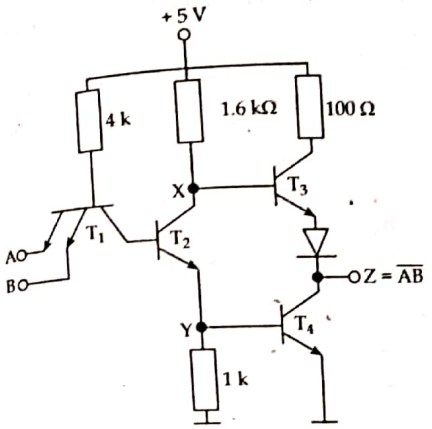


Figure: Basic TTL NAND gate

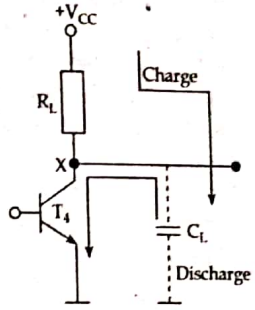


Figure: Effect of parasitic capacitance across T_4

This circuit incorporates a totem-pole output driver. This totem-pole output driver produces enough current to driven ten logic gates connected to this gate.

Working Principle

We know, T_1 is a multiple-emitter transistor. T_2 is a phase-splitter driving output transistors T_3 and T_4. The output section consisting of transistors T_3 and T_4, diode D and the 100 ohm resistor is known as totem-pole configuration since it looks like totem-pole with its ups and downs. A 'totem-pole' is a stick or wand held by kings, emperors and holy men in their hands to show their authority. In another type of NAND gate, called the open collector NAND gate, transistor T_3 and diode D are removed.

Consider a situation when both the inputs A and B are at logic 0 (0 V) state *i.e.*, A = B = 0. The emitter-base diodes of the multiple-emitter transistor T_1 are forward biased in this condition and current flows from V_{CC} through the base and emitters of T_1 to ground. This makes the base-collector diode of T_1 reverse biased and prevents current from flowing into the base of transistor T_2. Since there is h_0 current flow through its base, T_2 remain off making its collector current $I_{C2} = 0$ and collector voltage $V_X = V_{CC}$. We also find that since the collector current of T_2 is zero, is emitter voltage $V_Y = 0$.

Let us now investigate the states of the totem-pole transistors T_3 and T_4. Since the collector voltage of T_2, $V_X = V_{CC}$, the transistor T_3 turns on and it acts as a short circuit between V_{CC} and the output Z. Also, since the emitter voltage of T_2, $V_Y = 0$, the transistor T_4 remains off. Thus, since T_3 is ON and T_4 is OFF, we find that the output Z of the TTL NAND, for the input condition A = B = 0 is V_{CC} or logic 1.

Consider now the situations in which A = 1, B = 0 and A = 0, B = 1. In these cases also, one of the diodes is conducting and the input is

grounded. So the same arguments we have developed for the situation A = B = 0 prevails in these cases also.

Now let us discuss the situation in which A = B = 1. In this case, both the diodes A and B remain reverse biased. So current flows from V_{CC} through the base-collector diode of T_1 into the base of T_2 turning it ON. Collector current of T_2 starts flowing now, developing enough voltage at its emitter Y to turn T_4 ON. Also since T_2 is ON its collector emitter voltage drops to the saturation values V_{CES} = 0.2 V and its collector voltage V_X = 0.8 + 0.2 = 1 V. With diode D represent, T_3 remains OFF. However, since T_4 is ON, the collector emitter voltage of T_4, V_{CES} = 0.2 V, and the output Z is at logic 0. Thus for the situation A = B = 1, we have Z = 0. Summing up the working of the circuit, we have the conditions.

A = 0,	B = 0,	Z = 1
A = 0,	B = 1,	Z = 1
A = 1,	B = 0,	Z = 1
A = 1,	B = 1,	Z = 0

These relations shows that the circuit acts as a NAND gate.

Use of Diode D

Consider the situation when there is no diode and T_4 is in saturation (ON). At this time, we want T_3 to be OFF. To investigate this, consider the following figure which shows the output section of the TTL NAND gate.

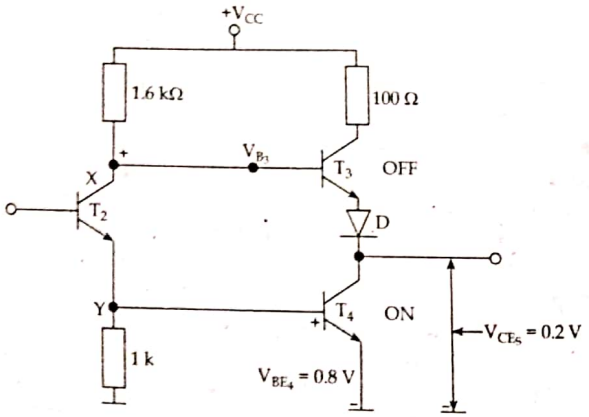


Figure: Totem-pole output Configuration

Consider T_2 be ON. Then heavy collector current flows through T_2 to develop enough potential at point Y, which turns T_4 ON. Since T_4 ON requires its base emitter voltage $V_{BE_4} = V_{BE_S}$, where V_{BE_S} is the saturation base-emitter voltage (= 0.8 V), we find that Y is clamped at 0.8 V. Hence the voltage at X is,

$$V_X = V_{BE_S} \text{ of } T_4 = V_{CE_S} \text{ of } T_2 = 0.8 + 0.2 = 1.0 \text{ V} = V_{B_3}$$

where, V_{BB} is the base voltage of T_3 with respect to ground. Now the collector of T_4 (and hence the emitter of T_3) is at V_{CE_S} (= 0.2 V) with respect to ground. Therefore, V_{BE_S} (base-emitter voltage of T_3) = 1 – 0.2 = 0.8 V. A voltage of 0.8 V between the base and emitter of T_3 means that it is in the saturation whereas our assumption was that T_3 was OFF. Hence to ensure that T_3 is indeed in the OFF state, diode D is introduced in between the emitter of T_3 and collector of T_4. The drop across the diode is usually 0.6 V so that T_3 is indeed in the cut-off state with its base having not enough voltage to turn it ON.

Open-Collector Gate

If transistor T_3 is removed from the totem-pole configuration, we can get an open collector TTL gate. The open collector TTL NAND gate is shown in figure below. This can be converted into a NAND gate by connecting a pull-up resistor across terminals P and Q.

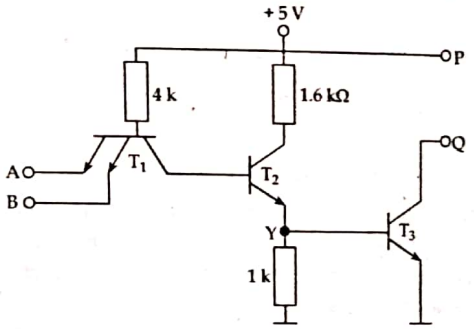


Figure: Open-collector TTL NAND gate

This gate is useful in what is known as the wired-AND connection. Several NAND gates can be AND-ed together by using open-collector TTL gates.

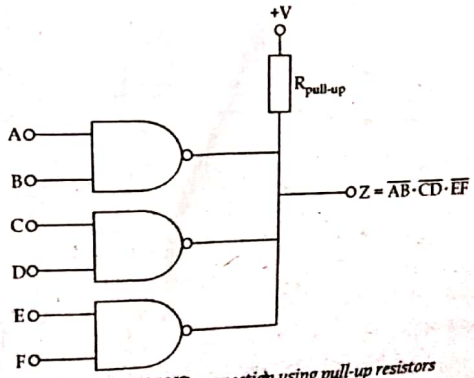


Figure: Wired AND connection using pull-up resistors

Figure below shows the actual circuit details of the wired-OR connections. T_{41}, T_{42} and T_{43} represent respectively the output transistors of gates G_1, G_2 and G_3. The output Z in wired-AND connection can be written as,

$$Z = \overline{AB} + \overline{CD} + \overline{EF} = \overline{AB} \cdot \overline{CD} \cdot \overline{EF}$$

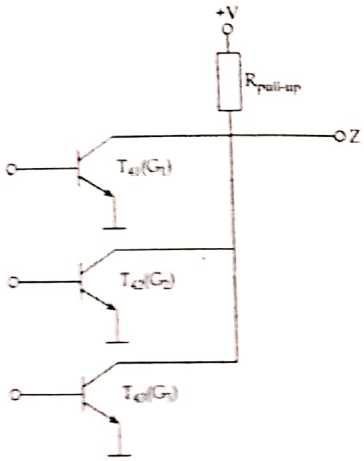


Figure: Connection details of wired-OR output section

Thus the output Z gives the AND-ing of several NAND gates which is also equivalent to NOR-ing of several AND gates. The value of the pull-up resistor $R_{pull-up}$ is determined by the maximum value of the collector current permissible through each of the output transistors.

10.6 EMITTER-COUPLED LOGIC (ECL)

The circuit diagram of an OR-NOR gate using emitter-coupled logic is shown below;

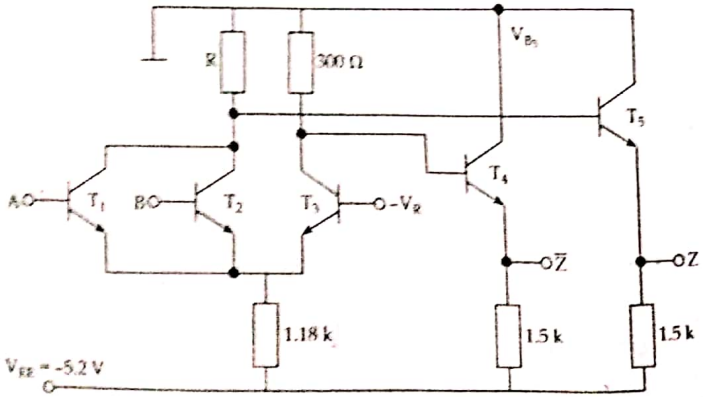


Figure: Emitter-coupled logic (ECL) gate

The fundamental gate in the ECL family is the OR-NOR gate. It is basically a set of differential amplifiers consisting of transistors T_1, T_2 and T_3. These differential amplifiers drives two emitter-follower stages comprising transistors T_4 and T_5 to deliver complementary outputs Z and $\overline{Z}$. The load resistance of T_3 is 300 ohms. For the ideal Op-AMP configuration the load of T_2 must also be of 300 ohms. However, in practice, it will be less than this. One of the main features of the ECL gate is that the collector terminals of its transistors are grounded and a DC supply voltage of -5.2 V is applied to the emitters. This negative supply voltage makes the calculations slightly difficult, since mostly we are familiar with positive supply voltages.

Working principle of the ECL OR-NOR gate

The ECL gate as state above is basically a differential amplifier working in its active region. Since it is operating in its active region, the voltage swing of the ECL gate is very low, being typically about + 200 mV. Also, it dissipates large amounts of power as the circuit draws heavy collector current at all times of operation. There is no OFF-state for the ECL gates to reduce power loss. The main advantage of the ECL gate is that it is the fastest logic circuit how in operation, typical speed being a few gigahertz.

Let us now analyze the voltage levels at various nodes of the circuit. Through the analysis, we shall also get an idea about the operation of the circuit. Consider the figure below, which shows the details of the circuit connections and node voltages required for the analysis. Let the input A be at logic 0 level. This makes T_2 OFF and T_3 ON.

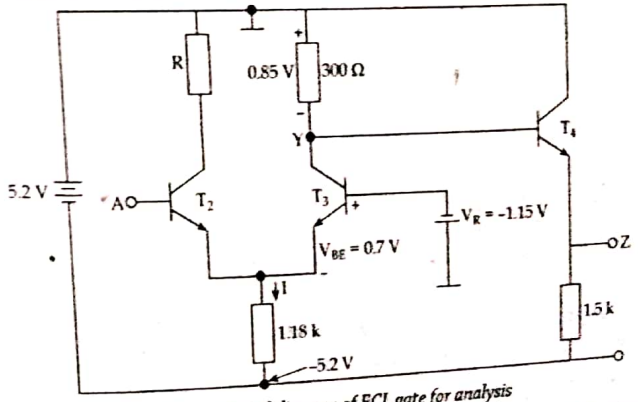


Figure: Detail diagram of ECL gate for analysis

The voltage at node X can be calculated by assuming V_{BE_3} = 0.7 V, where, V_{BE_3} is the base-to-emitter voltage of transistor T_3, assuming it to operate in the active region. We find that $V_X = -V_R - V_{BE_3} = -1.15 - 0.7 = -1.85$ V

The supply voltage of ECL gate is -5.2 V. Therefore, the voltage drop across the 1.18 K emitter resistor of the differential pair is,

$V_{1.18k} = -V_X - (-V_{EE}) = -1.85 - (-5.2) = 3.35$ V

Therefore, the current through the 1.18 K resistor I is given by;

$$I = \frac{3.35}{1.18} = 2.84 \text{ mA}$$

This current flowing through the collector of T_3 will produce a drop of 0.85 V across the 300 Ω collector resistor of T_3. Thus, the node Y is at – 0.85 V with respect to ground under this condition. Assuming T_4 to be ON this time, the voltage output t node Z (emitter of T_4) is:

$V_Z = -0.85 - 0.7 = -1.5$ V

This is the logic 0 level of the ECL gate. Notice that in the logic 0 level, all the transistors operate in their active region rather than in cut-off and saturation. Then, that the transistor T_2 or T_1 is indeed in logic 0 may now be ascertained by considering the the fact that with voltage at A,

$V_A = -1.5$ V and $V_X = -1.85$ V

$V_{BE_1} = V_{BE_2} = -1.5 - (-1.85) = 0.3$ V

where, V_{BE_1} and V_{BE_2} are the base- emitter voltages of T_1 and T_2 respectively. It can be seen that the minimum base-emitter voltage for a transistor to conduct is 0.4 to 0.5 V and therefore with $V_{BE} = 0.3$ V, both T_1 and T_2 are indeed in the OFF state.

Now, let us consider the logic 1 condition being applied to input A. This makes T_1 or T_2 ON, and the collector current switches to R from the 300 ohms resistor. With T_3 OFF, its collector voltage V_Y is at ground potential as no current is now flowing through the 300 ohms resistor. Again, with T_4 is an emitter-follower, the output voltage $V_Z = V_{BE_4} = 0.7$ V. Thus, the output voltage for logic 1 at A is given by;

$V_Z = -0.7$ V

We now conclude that for the ECL OR-NOR gate, the logic levels are given by:

Logic 0 level = –1.5 V

Logic 1 level = –0.7 V

These figures show that the output swing of an ECL gate is only

$V_{out\text{-}swing} = -0.7 (-1.5)$ V = 0.8 V

ECL gates have very low noise immunity, *i.e.*, they are easily vulnerable even to very low-level noise voltages.

10.7 METAL-OXIDE SEMICONDUCTOR (MOS)

The field effect transistor (FET) is a unipolar transistor, since its operation depends on the flow of only one type of carrier-either holes or electrons. There are two types of field effect transistors-junction field effect transistor (JFET) and metal oxide semiconductor field effect transistors (MOSFET).

Construction of a MOSFET

MOS technology derives its name from the basic structure of a metal electrode on an oxide insulator over a semiconductor substrate. The basic structure of the MOS transistor is shown below. The P-channel MOS consists of a lightly doped substrate of N-type silicon material. Two regions are heavily doped by diffusion with P-type impurities to form source and drain. The region between the two heavily doped areas of P-sections serves as the channel.

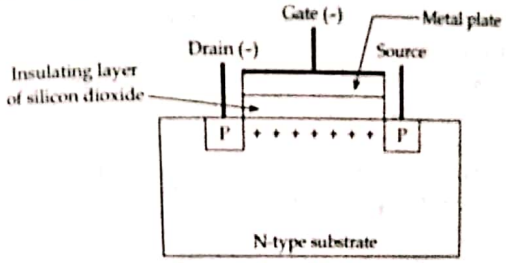


Figure: P-channel MOS

A metal plate is placed on the channel area with a separation layer consisting of insulating dielectric of silicon dioxide. The metal plate serves as a gate. When a negative voltage with respect to the substrate is applied to the gate, it causes an induced electric field in the channel, which attracts P-type carriers field in the channel, which attracts P-type carriers from the substrate. As the magnitude of the negative voltage on the gate increases, the region below the gate accumulate more P-type carriers, conductivity in the channel region increases and current can flow from source to drain provided a voltage difference is maintained between these two terminals.

There are four basic types of MOS structure. The channel can be either P-type or N-type. If the majority of carriers is positive type or holes, it is called a P-type MOS, and it is called an N-type MOS when the majority of carriers are negative type or electrons. The basic structure of an N-channel MOS is shown below;

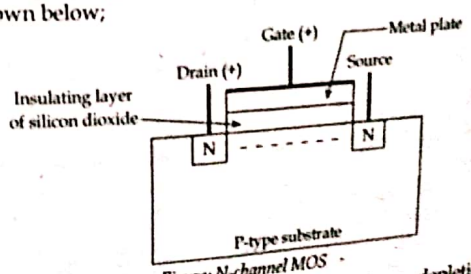


Figure: N-channel MOS

The mode of operation can be enhancement type or depletion type, depending on the state of the channel region at zero voltage. If the

channel is initially doped lightly with impurities (diffused channel), a conducting channel exists at zero gate voltage and the device is called a depletion type. In this mode, current flows unless the channel is depleted by an applied gate field. If the channel region is initially uncharged, the gate field induces a conducting channel before current can flow. Thus the current is enhanced with the application of gate voltage and such a device is called an enhancement type.

The terminal through which the majority of carriers enter the semiconductor bar is called the source and through drain the majority carries of leave the bar. For a P-channel MOS, the source terminal is connected to the substrate and a negative voltage is applied to the drain terminal. When the gate is above threshold voltage of about -2 V, no current will flow in the channel, and the drain to the source path is like an open circuit. When the gate voltage is sufficiently negative below threshold voltage, a channel is established and P-type carriers flow from source to drain. The P-type carriers are positive and hence the corresponding positive current flow from source to drain. The P-channel MOS are normally referred to as PMOS.

In the N-channel MOS, the source is connected to the substrate and positive voltage is applied to the drain terminal with respect to the source or the substrate. When the gate voltage is below the threshold voltage V_L no current flows through the channel. If sufficiently large positive gate voltage is applied above the threshold voltage, N-type carriers will flow from drain to source. The threshold voltage may vary from 1 V to 4 V, depending on the particular process of fabrication used. The N-channel MOS are generally referred as NMOS.

The graphic symbols of MOS transistors are shown below G, D and S represent the gate, drain and source respectively. Enhancement types of MOSFETS are symbolized by the broken lines between source and drain. The arrow indicates the direction of the flow of carriers.

Figure: P-channel enhancement type

Figure: P-channel depletion type

Figure: N-channel enhancement type

Figure: N-channel depletion type

Because of the symmetrical construction of the source and drain in the MOS transistors, they can be operated as bilateral devices. Although in normal operation, carriers are allowed to flow from source to drain in certain circumstances it is convenient to allow the carriers to flow drain to source. One advantage of the MOS device is that it can be used not only as transistor, but also a resistor. A resistor may be constructed with technology by permanently biasing the gate terminal for conduction. The value of resistance is determined by the ratio of the source-drain voltage to the channel current.

Three logic circuits using MOS devices are shown in figure below. For an N-channel MOS, supply voltage V_{DD} is positive (about 5 V) to allow positive current flow from drain to source. The two voltage levels are a function of the threshold voltage V_T. The low level is anywhere from zero to V_T and the high level ranges from V_T to V_{DD}. The N-channel gates usually employ positive logic. The P-channel MOS circuits use a negative voltage for V_{DD} to allow positive current flow from source to drain. The two voltage levels are both negative above and below the negative threshold voltage V_T. P-channel gates usually employ negative logic.

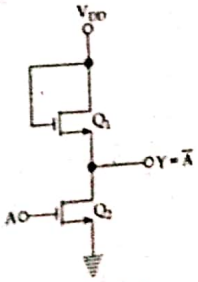


Figure: Inverter

This inverter circuit uses two MOS devices. Q_1 acts as the load resistor and Q_2 acts as the active device. The load resistor MOS has its gate connected to V_{DD}, thus maintaining it always in the conduction state. When the input voltage is low (below V_T), Q_2 turns off. Since Q_1 is always ON, the output voltage is high (above V_T). Q_2 turns ON. Current flows from V_{DD} through the load resistor Q_1 and into Q_2. The geometry of the two MOS device must be such that the resistance of Q_2 when conducting

is much less than the resistance of Q_1 to maintain the output Y at a voltage below V_T.

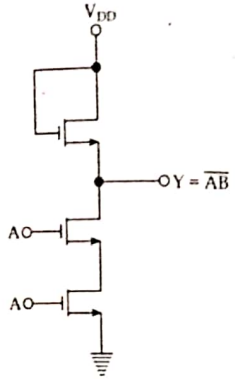


Figure: NAND gate

The NAND gate uses two transistors in series. Inputs A and B must both be high for all transistors to conduct and cause the output to go low. If either input is low, the corresponding transistor is turned OFF and the output is high. Again, the series resistance formed by the two active MOS devices must be much less than the resistance of the load resistor MOS.

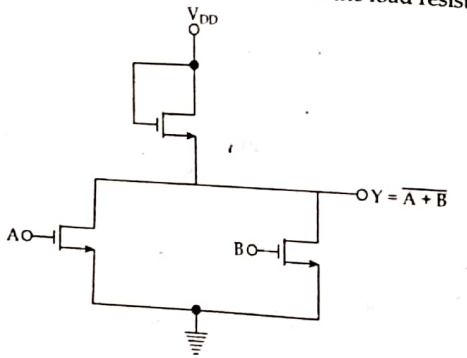


Figure: NOR gate

The NOR gate uses transistors in parallel. If either input is high, the corresponding transistor conducts and the output is low. If all inputs are low, all active transistors are off and the output is high.

OSFET logic gates consume much less power and possess higher noise rgin. It has high fan out capabilities because of the extremely high t resistance at each of the inputs. It is also very simple to fabricate as es not require resistors or diodes etc. that leads to high package y. Its low power dissipation makes it ideally suited for LSI packages where the MOS logic has made its greatest impact in the digital owever, MOS gates are slower than the TTL gates.

10.8 COMPLEMENTARY MOS (CMOS) LOGIC

Complementary MOS circuits take advantage of the fact that both N-channel and P-channel devices can be fabricated on the same substrate. CMOS circuits consist of both types of MOS devices interconnected to form logic functions. The basic circuit is the inverter which consists of one P-channel transistor and one N-channel transistor as shown in figure.

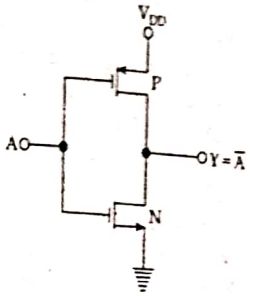


Figure: Inverter

The source terminal of the P-channel device is at V_{DD} and the source terminal of the N-channel device is at ground. The value of V_{DD} may be anywhere from +3 to +18 V. The two voltage levels are 0 V for the low level and V_{DD} for the high level. Two other CMOS basic gates are shown in figure below;

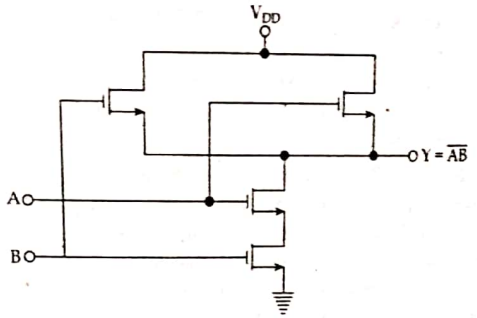


Figure: NAND gate

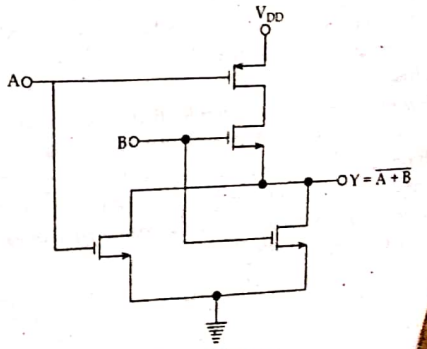


Figure: NOR gate

A two-input NAND gate consists of two P-type units in parallel and two N-type units in series. If all inputs are high, both P-channel transistors turn off and both N-channel transistors turn on. The output has a low impedance to ground and produces a low state. If any input is low, the associated N-channel transistor is turned OFF and the associated P-channel transistor is turned ON. The output is coupled to V_{DD} and goes to the high state. Multiple input NAND gates may be formed by placing equal numbers of P-type and N-type transistors in parallel and series respectively.

A two input NOR gate consists of two N-type units in parallel and two P-type units in series. When all inputs are low, both P-channel units are ON and both N-channel units are OFF. The output is coupled to V_{DD} and goes to the high state. If any input is high, the associated P-channel transistor is turned OFF and the associated N-channel transistor turns ON. This connects the output to ground, causing a low level output.

CMOS Characteristics

When a CMOS logic circuit is in a static state, its power dissipation is very low. This is because there is always an OFF transistor in the current path when the state of the circuit is not charging. As a result, a typical CMOS gate has a static power dissipation on the order of 0.01 mV. However, when the circuit is changing state at a rate of 1 MHz, the power dissipation increases to about 1 mW.

CMOS logic is usually specified for a single power-super operation over the voltage range between 3 and 18 V with a typical V_{DD} value of 5 V. Operating CMOS at a larger value of supply voltage reduces the propagation delay time and improves the noise margin, but the power and improves the noise margin, but the power dissipation is increased. The propagation delay time with V_{DD} = 5 V ranges from 8 to 50 ns depending on the type of CMOS used. The noise margin is usually about 40 percent of the V_{DD} supply voltage. The fan-out CMOS gates is 50 when operated at a frequency of less than 1 MHz. The fan-out decreases with increase in frequency of operation.

The CMOS fabrication process is simpler than TTL and provides a greater packing density. This means that more circuits can be placed on a given area of silicon at a reduced cost per function. This property of CMOS, together with its low power dissipation, excellent noise immunity and reasonable propagation delay makes it a strong contender for a popular standard as a digital logic family.

REFERENCES

1. Morris Mano, "*Digital Logic and Computer Design*", Prentice Halt of India.
2. M. Rafiquzzaman, "*Fundamentals of Digital logic and Microcomputer Design*", JOHN WILEY & SONS, I, Publication
3. A. SAHA and N. MANNA, "*Digital Principles and Logic Design*", Anil K. Maini, "*Digital Electronics; Principles, Devices and Applications*", JOHN WILEY & SONS, Ltd.
4. Subir Kumar Sarkar, Asish Kumar De, Souvik Sarkar, "*Foundation of Digital Electronics and Logic Design*", CRC Press, Taylor & Francis Group.
5. U.A Bakshi and A.P Godse, "*Analog and Digital Electronics*", Technical Publication Pune.
6. Frederic J. Mowle, "*A Systematic Approach to Digital Logic Design Malvino: Digital Computer Electronics*".
7. A. P Malvino, Jerald A. Brown "*Digital computer Electronics*", 1995
8. V.K Puri, "*Digital Electronics*", TMH
9. Brain Holdsworth, "*Digital Logic Design*", Elsevier Science
10. M. Morris Mano and Charles Kime, "*Logic and Computer Design Fundamentals*", Pearson New International.