According to New Credit System Syllabus

DBATU

Second Year (S.Y.) B. TECH Semester - IV
Course in COMPUTER ENGINEERING

DIGITAL LOGIC DESIGN AND MICROPROCESSORS

Dr. NARENDRA S. JADHAV
Dr. (Mrs.) ALPANA P. ADSUL

www.pragationline.com
niralipune@pragationline.com
www.facebook.com/niralibooks
@nirali.prakashan

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

A TEXT BOOK OF

DIGITAL LOGIC DESIGN AND MICROPROCESSOR

FOR

Semester – IV

SECOND YEAR (S.Y.) B. TECH COURSE IN COMPUTER ENGINEERING

Strictly According to New Revised Credit System Syllabus of Dr. Babasaheb Ambedkar Technological University (DBATU), Lonere, Dist. Raigad, Maharashtra.

(w.e.f 2021 - 2022)

Dr. NARENDRA S. JADHAV
B.Tech (E&TC), M.Tech., Ph.D.
Assistant Professor,
Electronics & Telecommunication Engg. Deptt.,
Dr. Babasaheb Ambedkar Technological University (BATU),
LONERE, Dist. RAIGAD

Dr. (Mrs.) ALPANA P. ADSUL
B.E. (Electronics), M.E. (CSE-IT), Ph.D.
Associate Professor & Head,
Information Technology Deptt.
Sinhgad Institute of Technology & Science
Narhe, PUNE

Price ₹ 200.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N6226

Digital Logic Design & Microprocessor (COMP., DBATU) **ISBN: 978-93-5451-724-2**

First Edition : **June 2022**

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312 Shivaji Nagar
Off J.M. Road, PUNE 411005
Tel : (020) 25512336/37/39
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune 411041

DISTRIBUTION CENTRES

PUNE

Nirali Prakashan
(For orders outside Pune)
S. No. 28/27, Dhayari Narhe Road, Near Asian College
Pune 411041, Maharashtra
Tel : (020) 24690204; Mobile : 9657703143
Email : bookorder@pragationline.com

Nirali Prakashan
(For orders within Pune)
119, Budhwar Peth, Jogeshwari Mandir Lane
Pune 411002, Maharashtra
Tel : (020) 2445 2044; Mobile : 9657703145
Email : niralilocal@pragationline.com

MUMBAI

Nirali Prakashan
Rasdhara Co-op. Hsg. Society Ltd., 'D' Wing Ground Floor, 385 S.V.P. Road
Girgaum, Mumbai 400004, Maharashtra
Mobile : 7045821020, Tel : (022) 2385 6339 / 2386 9976
Email : niralimumbai@pragationline.com

DISTRIBUTION BRANCHES

DELHI

Nirali Prakashan
Room No. 2 Ground Floor
4575/15 Omkar Tower, Agarwal Road
Darya Ganj, New Delhi 110002
Mobile : 9555778814/9818561840
Email : delhi@niralibooks.com

BENGALURU

Nirali Prakashan
Maitri Ground Floor, Jaya
Apartments, No. 99, 6th Cross, 6th
Main, Malleswaram, Bengaluru
560003 Karnataka; Mob : 9686821074
Email : bengaluru@niralibooks.com

NAGPUR

Nirali Prakashan
Above Maratha Mandir, Shop No. 3,
First Floor, Rani Jhanshi Square,
Sitabuldi Nagpur 440012 (MAH)
Tel : (0712) 254 7129
Email : nagpur@niralibooks.com

KOLHAPUR

Nirali Prakashan
438/2, Bhosale Plaza, Ground Floor,
Khasbag, Opp. Balgopal Talim,
Kolhapur 416 012 Maharashtra
Mob : 9850046155
Email : kolhapur@niralibooks.com

JALGAON

Nirali Prakashan
34, V. V. Golani Market, Navi Peth,
Jalgaon 425001, Maharashtra
Tel : (0257) 222 0395
Mob : 94234 91860
Email : jalgaon@niralibooks.com

SOLAPUR

Nirali Prakashan
R-158/2, Avanti Nagar, Near Golden
Gate, Pune Naka Chowk
Solapur 413001, Maharashtra
Mobile 9890918687
Email : solapur@niralibooks.com

marketing@pragationline.com | www.pragationline.com

Also find us on www.facebook.com/niralibooks

PREFACE

It gives us great pleasure to present the book **'Digital Logic Design & Microprocessor'** for the students of **Semester IV Second Year (S.Y.) B.Tech. Course in Computer Engineering of Dr. Babasaheb Ambedkar Technological University (DBATU), Lonere, Dist. Raigad (Maharashtra).** This book is strictly as per the new revised syllabus, 2020 Pattern, effective from the Academic Year 2021 - 2022.

The Theory Course will have 4 Credits.

In New Revised Syllabus, there will be Internal Assessment (20 Marks), Mid Sem. Exam. (20 Marks) and End Sem. Exam. (60 Marks). End Sem. Exam. will be based on all Five Units.

The basic objective of this book is to bridge the gap between the vast contents of the reference books, written by the renowned International Authors and the concise requirements of Undergraduate Students. This book has been written in a comprehensive manner using Simple and Lucid language, keeping in mind student's requirements. The main emphasis has been given on exploring the basic concepts rather than merely the Information. Solved Examples, Multiple Choice Questions (MCQ's) and Exercises have been provided throughout the book at the end of the Unit. Also we have given **Model Question Papers** for practice at the end of the book.

Our special thanks to our family members, students and all those who directly or indirectly supported us in this project.

We also take this opportunity to express our sincere thanks to Shri. Dineshbhai Furia, Shri. Jignesh Furia, Mrs. Nirali Verma and entire team of Nirali Prakashan, namely Mrs. Deepali Lachake (Co-ordinator), who really have taken keen interest and untiring efforts in publishing this text.

We wish to make a special mention of the valuable contribution made by Late Shri M.P. Munde during the last two decades in reaching out to students, parents and teachers which eventually made **Nirali Prakashan, a brand of trust and quality in technical books.** Without his perseverance and zeal, successive editions would not have been possible.

The advice and suggestions of our esteemed readers to improve the text are most welcome and will be highly appreciated.

Pune

Authors

SYLLABUS

Unit I: Introduction [7 Hours]

Digital signals, digital circuits, AND, OR, NOT, NAND, NOR and Exclusive-OR operations, Boolean algebra, examples of IC gates, Number Systems: binary, signed binary, octal hexadecimal number, binary arithmetic, one's and two's complements arithmetic, codes, error detecting and correcting codes.

Unit II: Combinational Digital Circuits [7 Hours]

Standard representation for logic functions, K-map representation, simplification of logic functions using K-map, minimization of logical functions, Don't care conditions, Multiplexer, De-Multiplexer / Decoders, Adders, Subtractors, BCD arithmetic, carry look ahead adder, serial adder, ALU, elementary ALU design, parity checker / generator.

Unit III: Sequential Circuits and Systems [7 Hours]

1-bit memory, the circuit properties of Bistable latch, the clocked SR flip flop, J-K-T and D-types flip flops, applications of flip flops, shift registers, applications of shift registers, serial to parallel converter, parallel to serial converter, ring counter, sequence generator, ripple(Asynchronous) counters, synchronous counters, counters design using flip flops, special counter IC's, asynchronous sequential counters, applications of counters.

Unit IV: Fundamentals of Microprocessors [7 Hours]

Fundamentals of Microprocessor, Comparison of 8-bit, (8085) 16-bit (8086), and 32-bit microprocessors (80386), The 8086 Architecture: Internal Block Diagram, CPU, ALU, address, data and control bus, Working registers, SFRs, Clock and RESET circuits, Stack and Stack Pointer, Program Counter, I/O ports, Memory Structures, Data and Program Memory, Timing diagrams and Execution Cycles.

Unit V: 8086 Instruction Set and Programming [7 Hours]

Memory Interfacing, I/O Interfacing, Direct Memory Access (DMA), Interrupts in 8086, 8086 Instruction Set and Programming: Addressing modes: Introduction, Instruction syntax, Data types, Subroutines Immediate addressing, Register addressing, Direct addressing, Indirect addressing, Relative addressing, Indexed addressing, Bit inherent addressing, bit direct addressing, Instruction timings, Data transfer instructions, Arithmetic instructions, Logical instructions, Branch instructions, Subroutine instructions, Bit manipulation instruction, Assembly language programs, C language programs, Assemblers and compilers, Programming and debugging tools.

CONTENTS

✠ ✠ ✠

Unit I

INTRODUCTION

1.1 INTRODUCTION TO ANALOG AND DIGITAL SIGNAL

In our day-to-day lives, we constantly deal with the numerical values of various physical quantities. These values are represented in two basic ways. First way to represent is referred as analog; in which the quantity of numerical value is expressed in continuous range of values. For example, voltage across a certain component in an electronic circuit may be measured as 6.5V or 6.49V or 6.487V or 6.4869V. In this type of representation, variation in the numerical value of the quantity is continuous and could have any of the infinite theoretical possible values between the two extremes.

The second way to represent is referred as digital, which represents the numerical value of the quantity in steps of discrete values. The numerical values are represented using binary numbers ('0' and '1').

An analog representation gives a continuous output; a digital representation produces a discrete output. Analog systems contain devices that process or work on various physical quantities represented in analog form. Digital systems contain devices that process the physical quantities represented in digital form. Fig. 1.1 shows the representation of both signals.

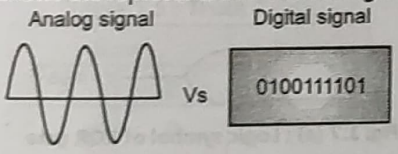
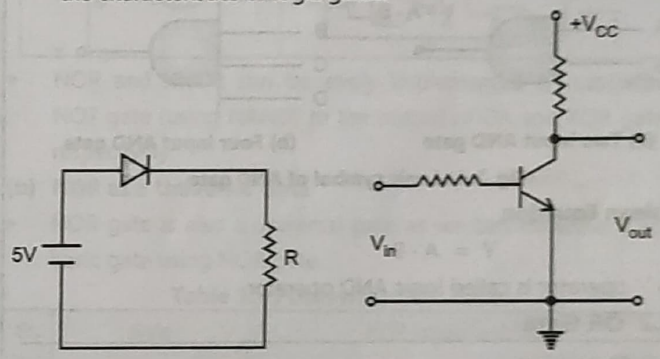


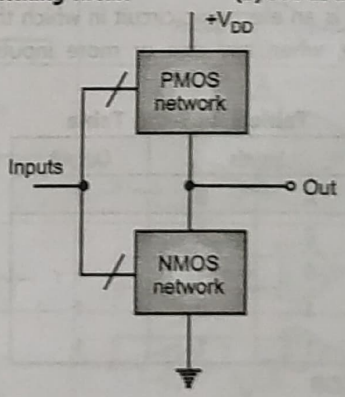
Fig. 1.1 : Analog and digital signal representation

1.2 BASIC LOGIC GATES

- The term **Logic** refers to something which can be reasoned out. In many situations, the problem statements can be expressed in true or false and yes or no formats.
- Since digital circuits also have two states or binary forms, these kind of problem statements can be formulated using logic states or logic functions.
- Because the voltage levels in a digital circuit are assumed to be switched from one value to another, the digital circuits are called logic circuits or switching circuits.
- Logic gates are the logic circuits which obey a certain set of logic rules. The manner in which a logic circuit responds to an input is referred to as the circuit's logic.
- The name logic gate is derived from the ability of such device to make decisions i.e. produce different outputs in response to different combinations of inputs.
- As mentioned earlier, logic gate is an electronic switching circuit.
- Semiconductor devices such as diode, BJT or MOSFET can be used to build logic gates.
- The inherent characteristics of these devices such as junction capacitance, uni- or bi-polar, diffusion capacitance decides the characteristics of logic gates.



(a) Diode switching circuit **(b) BJT as a switch**



(c) Generalised CMOS switch

Fig. 1.2 : Semiconductor devices as switches

The Logic Gates can be Classified as Below :

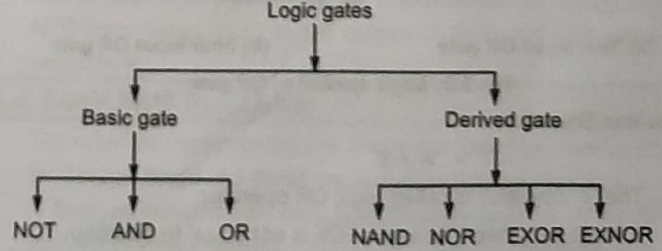


Fig. 1.3

1.2.1 AND Gate

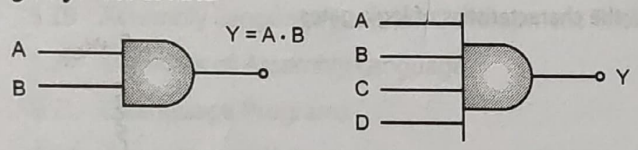
Logic Statement

- The AND gate is an electronic logic circuit in which the output is in logic '1' state only when all the inputs are in logic '1' state.

Table 1.1 : Truth Table

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

Logic Symbol of AND



(a) Two input AND gate (b) Four input AND gate

Fig. 1.4 : Logic symbol of AND gate

Boolean Equation

$$Y = A \cdot B$$

The '·' operator is called logic AND operator.

1.2.2 OR Gate

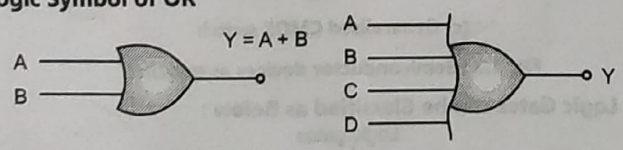
Logic Statement

- The OR gate is an electronic circuit in which the output is in logic 1 state, when any one or more inputs are in logic 1 states.

Table 1.2 : Truth Table

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

Logic Symbol of OR



(a) Two input OR gate (b) Four input OR gate

Fig. 1.5 : Logic symbol of OR gate

Boolean Equation

$$Y = A + B$$

- The '+' operator is called logic OR operator.
- It must be noted that logical OR is not equal to addition.
  When A = B = 1 the logical OR gate gives '1' output whereas addition will give output as '0' and carry as '1'.

1.2.3 NOT Gate

Logic Statement

- The output of a NOT circuit takes on the logic '1' state, if and only if the input does not take on the '1' state.

Table 1.3 : Truth Table

A	Y
0	1
1	0

Logic Symbol of NOT

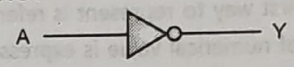


Fig. 1.6 : Logic symbol of NOT gate

- It is also called inverter.
- The bubble appearing at the output indicates inversion or negation.

Boolean Equation

$$Y = \bar{A}$$

- The bar above variable A represents the logical inversion operation.
- Some authors use ' ' ' symbol which is more appropriate.
  i.e. $Y = A'$

1.2.4 Exclusive OR Gate

Logic Statement

- The XOR gate is an electronic logic circuit having two or more number of inputs and only one output which recognizes only inputs that have an odd number of logic 1 state.

Logic Symbol

Fig. 1.7 (a) : Logic symbol of XOR gate

Boolean Equation

The Boolean equation for XOR gate is written as,

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

or $$Y = A \oplus B$$

Timing Diagram (Pulsed Operation)

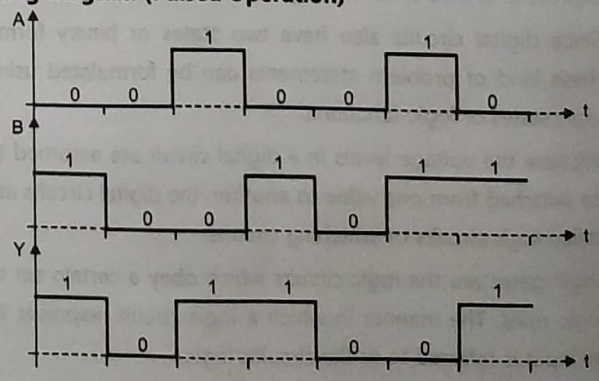


Fig. 1.7 (b) : Timing diagram for XOR gate

1.2.5 NAND Gate

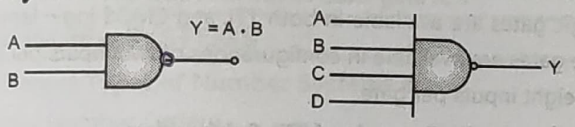
Logic Statement

- The NAND gate is an electronic logic circuit in which the output is in logic 1 state, when both or any one of the inputs are in logic 0 state.

Table 1.4 : Truth Table

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

Logic Symbol of NAND



(a) Two input NAND gate (b) Four input NAND gate

Fig. 1.8 : Logic symbol of NAND gate

Boolean Equation

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

1.2.6 NOR Gate

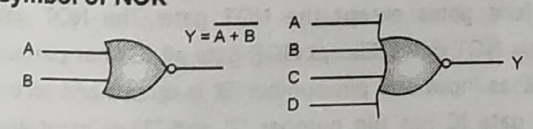
Logic Statement

- The NOR gate is an electronic logic circuit in which the output is in logic '1' state only when both or more inputs are in logic '0' states.

Table 1.5 : Truth Table

Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Logic Symbol of NOR



(a) Two input Nor gate (b) Four input NOR gate

Fig. 1.9 : Logic symbol of NOR gate

Boolean Equation

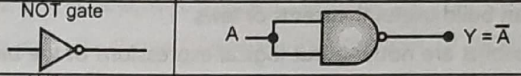
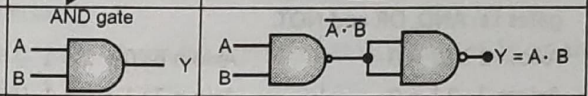
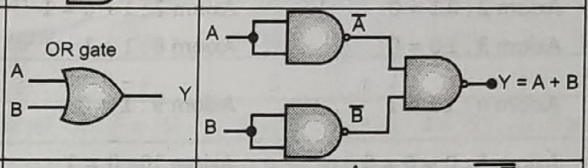
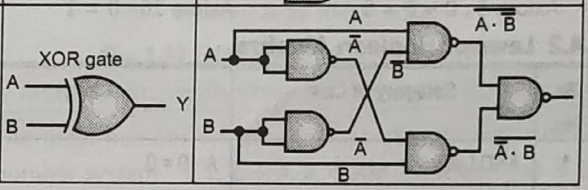
$$Y = \overline{(A + B)}$$

1.3 UNIVERSAL GATES – NAND AND NOR

(a) NAND as a Universal Gate

- The NAND gate is a universal gate because all basic gates can be implemented using NAND gates.

Table 1.6 : Universal NAND Gate

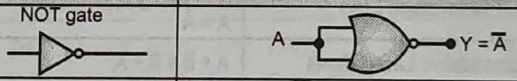
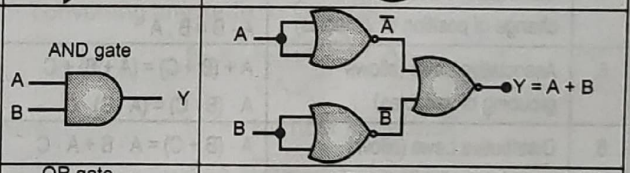
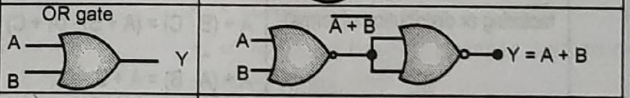
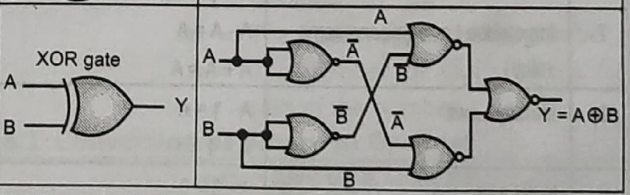
Sr. No.	Gate	NAND Gate Implementation
1.	NOT gate	
2.	AND gate	
3.	OR gate	
4.	XOR gate	

- NOR and XNOR can be easily implemented by cascading NOT gate (using NAND) to the output of OR and XOR gates respectively.

(b) NOR as a Universal Gate

- NOR gate is also a universal gate as we can implement any basic gate using NOR gate.

Table 1.7 : Universal NOR Gate

Sr. No.	Gate	NOR Implementation
1.	NOT gate	
2.	AND gate	
3.	OR gate	
4.	XOR gate	

- NAND and XNOR gates can be implemented by cascading NOT gate (using NOR) to the output of AND and XOR gates respectively.

1.4 BOOLEAN ALGEBRA

- The rules for manipulating the binary numbers are developed by George Boole and these are known as Boolean Algebra.
- A Boolean variable can only take binary values i.e. '1' or '0' just like ordinary algebra.
- Boolean algebra has also its own certain operators like AND (·), OR (+), NOT (–) and XOR (⊕).

1.4.1 Axioms of Boolean Algebra

- Axioms and postulates of Boolean algebra are a set of logical expressions that we accept without proof and upon which we can build useful theorems or laws.
- Axioms are nothing but logical expressions of the basic three gates i.e. AND, OR and NOT.

Axiom 1 : 0.0 = 0
Axiom 2 : 0.1 = 0
Axiom 3 : 1.0 = 0
Axiom 4 : 1.1 = 1
Axiom 5 : 0 + 0 = 0
Axiom 6 : 0 + 1 = 1
Axiom 7 : 1 + 0 = 1
Axiom 8 : 1 + 1 = 1
Axiom 9 : $\bar{1} = 0$
Axiom 10 : $\bar{0} = 1$

1.4.2 Laws of Boolean Algebra

Sr. No.	Category of Law	Laws
1.	AND Laws	$A \cdot 0 = 0$ $A \cdot 1 = A$ $A \cdot A = A$ $A \cdot \bar{A} = 0$
2.	OR Laws	$A + 0 = A$ $A + 1 = 1$ $A + A = A$ $A + \bar{A} = 1$
3.	NOT or Inversion Laws	$\bar{\bar{A}} = A$
4.	Commutative Laws (allows change of position of variables)	$A + B = B + A$ $A \cdot B = B \cdot A$
5.	Associative Laws (allows grouping of variables)	$A + (B + C) = (A + B) + C$ $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
6.	Distributive Laws (allows factoring or distribution of terms)	$A \cdot (B + C) = A \cdot B + A \cdot C$ $A + (B \cdot C) = (A + B) \cdot (A + C)$ $A + (\bar{A} \cdot B) = A + B$
7.	Impotence Laws (means same value)	$A \cdot A = A$ $A + A = A$
8.	Identity Laws	$A \cdot 1 = A$ $A + 1 = 1$
9.	Null Laws	$A \cdot 0 = 0$ $A + 0 = A$
10.	Absorption Laws	$A + (A \cdot B) = A$ $A (A + B) = A$

1.4.3 Rules of Boolean Algebra

The Rules that are Followed in Boolean Algebra are given Below :

- Capital letters are used for representing variables and functions of variables.
- It will be assumed that the positive logic is used unless and until the problem statement specifically mentions negative logic.
- The complement of a variable is represented by a "bar" (¯) over the variable letter.
- The logic AND function is shown by a "dot" (·) between the two variables, e.g. A · B. Many times, this dot is not written i.e. AB.
- Boolean addition is logical OR operation. It is different than mathematical addition. For example,

  1 + 1 = 1 in Boolean algebra

  1 + 1 = 0 with carry 1 in mathematics

1.4.4 Examples of IC Gates

All logic gates are available in both TTL and CMOS logic families. All the gates are available in configurations of two inputs per gate up to eight inputs per gate.

Following are the Example of TTL Subfamilies:

- 74LXX--low-power TTL (1/10 the speed, 1/10 the power of "regular" TTL)
- 74HXX--high-speed TTL (twice as fast, twice as much power)
- 74SXX—Schottky TTL (for high-frequency uses)
- 74LSXX--combination of low-power & Schottky, same speed as regular TTL, but at 1/5 the power consumption

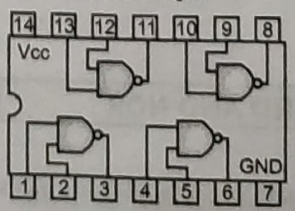
Table 1.8 shows the list of basic gate ICs

Table 1.8 : List of Basic Gate IC's

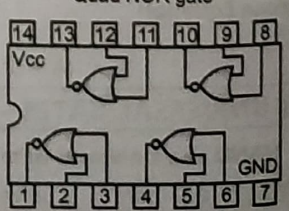
Sr. No.	IC No.	Gate
1.	7400	NAND GATE
2.	7402	NOR Gate
3.	7404	NOT GATE
4.	7408	AND GATE
5.	7432	OR GATE
6.	7486	EX-OR GATE

The Fig. 1.10 shows the pinout of the basic gate IC's. All ICs contains four gates except the NOT gate. The NOT gate IC contains six NOT gates. Except NOR gate all IC's has pin number '1 ' and '2' as input and pin number '3' is output and so on. But the NOR gate IC has pin number '2' and '3' as input and pin number '1' as output.

5400/7400
Quad NAND gate



5402/7402
Quad NOR gate



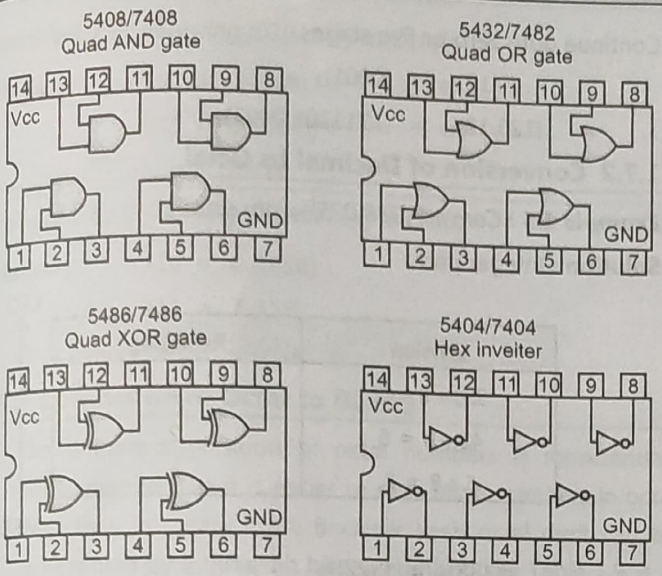


Fig. 1.10 : Pin out of basic gate IC's

1.5 NUMBER SYSTEMS

The Different Types of Number Systems are:

- Binary Number System
- Decimal Number System
- Hexadecimal Number System
- Octal Number System

1.5.1 Binary Number System

- Binary number system is used in digital electronics. It has the following characteristics.

  Two digits : 0, 1

  Base : 2

  Weights : Powers of Base 2 (2^0, 2^1, 2^2, 2^3...) or (1, 2, 4, 8).

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	0	0	1	1	0	0	0

MSB LSB

Fig. 1.11 : Binary number system

- In the binary number system, 1's and 0's are arranged into columns.
- Each column is weighted. The first column on the right has a binary weight of 2^0. This equivalent to decimal 1 and is referred to as the Least Significant Bit (LSB).
- The number in the far left hand column is called Most Significant Bit (MSB).

1.5.2 Decimal Number System

- In decimal number system, we can express any decimal number in units, ten, hundreds, thousands and so on.

  e.g. 6597.8 this number can be represented as

  $6000 + 500 + 90 + 7 + 0.8 = 6597.8 = 6597.8_{10}$

  radix or base of decimal number system is 10.

In power of 10	10^3	10^2	10^1	10^0		10^{-1}
	6	5	9	7	•	8
	6×10^3	5×10^2	9×10^1	7×10^0	•	8×10^{-1}

Fig. 1.12 : Decimal number system

1.5.3 Hexadecimal Number System

- In the hexadecimal number system, a base of 16 has 16 characters. The 0 to 9 is same as decimal number system and A to F is represented as 10 to 15 respectively.

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

- It is easy to convert hexadecimal number to binary and vice versa.

  e.g. 3FD. 48 can be represented in power of 16 as shown in Fig. 1.13.

16^2	16^1	16^0		16^{-1}	16^{-2}
3	F	D	•	4	8
3×16^2	$F \times 16^1$	$D \times 16^0$	•	4×16^{-1}	8×16^{-2}

Where F is 15 and D is 13

Fig. 1.13 : Hexadecimal number system

1.5.4 Octal Number System

- The octal number system consists of eight digits of decimal number system : 0, 1, 2, 3, 4, 5, 6 and 7. So it's base is 8.

  e.g. The octal number 8531.74 can be represented in power of 8 as shown in Fig. 1.14.

8^3	8^2	8^1	8^0		8^{-1}	8^{-2}
8	5	3	1	•	7	4
8×8^3	5×8^2	3×8^1	1×8^0	•	7×8^{-1}	4×8^{-2}

Fig. 1.14 : Octal number system

1.6 CONVERSION OF ANY RADIX NUMBERS TO DECIMAL NUMBER

- To convert from any Radix to Decimal. The formula for converting any Radix to Decimal is

$$N = A_{n-1} r^{n-1} + A_{n-2} r^{n-2} + \ldots + A_1 r^1 + A_0 r^0 + A_{-i} r^{-i} + \ldots + C_m r^{-m} \quad \ldots(1)$$

where N = Number in decimal

A = Digit

r = Radix or base of number system number of digits

m = The fractional part of the number

n = The number of digits in the fractional part of the number

1.6.1 Conversion of Binary to Decimal

SOLVED EXAMPLES

Example 1.1 : *Convert binary number 1011.01 into its decimal equivalent.*

Solution : Given binary number is 1011.01

Use formula given in equation 1.

$$(1011.01) = (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2})$$

$$= 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 + 0 \times \frac{1}{2} + 1 \times \frac{1}{4}$$

$$= 8 + 0 + 2 + 1 + \frac{1}{4} = (11.25)_{10}$$

1.6.2 Conversion of Octal to Decimal

Example 1.2 : *$(321)_8$ = ( )$_{10}$*

Solution : Given number is $(321)_8$

Use formula given in equation 1.

$$(321)_8 = 3 \times 8^2 + 2 \times 8^1 + 1 \times 8^0$$
$$= 3 \times 64 + 2 \times 8 + 1 \times 1$$
$$= 192 + 16 + 1$$
$$\therefore \quad (321)_8 = (209)_{10}$$

1.6.3 Conversion of Hexadecimal Number to Decimal Number

Example 1.3 : *Convert hexadecimal number $(4D7.2)_{16}$ into its equivalent decimal number.*

Solution : Given number is $(4D7.2)_{16}$

$$(4D7.2) = 4 \times 16^2 + 13 \times 16^1 + 7 \times 16^0 + 2 \times 16^{-1}$$
$$= 4 \times 256 + 13 \times 16 + 7 \times 1 + \frac{2}{16}$$
$$= 1024 + 208 + 7 + 0.125$$
$$\therefore \quad (4D7.2)_{16} = (1239.125)_{10}$$

1.7 CONVERSION FROM DECIMAL TO OTHER NUMBER SYSTEM

- In this conversion method, we need to first consider the given number having decimal point which separates out integer part and fractional parts.
- Then convert each part by different method and then combine both answers in the end.

1.7.1 Conversion of Decimal to Binary

Example 1.4 : *Convert 125.12 decimal into binary number.*

Solution : Integer part :

Division	Remainder
125 ÷ 2 = 62	1
62 ÷ 2 = 31	0
31 ÷ 2 = 15	1
15 ÷ 2 = 7	1
7 ÷ 2 = 3	1
3 ÷ 2 = 1	1
1 ÷ 2 = 0	1

LSD ↑ MSD

$$(125)_{10} = (1\,1\,1\,1\,1\,0\,1)_2$$

Fractional part :

0.12	0.24	0.48	0.96
× 2	× 2	× 2	× 2
0.24	0.48	0.96	1.92

Continue upto zero or five stages

$$(0.12)_{10} = (0001)_2$$
$$\therefore \quad (125.12)_{10} = (1111101.0001)_2$$

1.7.2 Conversion of Decimal to Octal

Example 1.5 : *Convert $(338.025)_{10}$ into octal.*

Solution : Integer part :

Division	Remainder
338 ÷ 8 = 42	2
42 ÷ 8 = 5	2
5 ÷ 8 = 0	5

LSD ↑ MSD

$$(338)_8 = (522)_{10}$$

Fractional part :

0.025	0.200	0.600	0.800
× 8	× 8	× 8	× 8
0.200	1.600	4.800	6.400
↓	↓	↓	↓
0	1	4	6

$$(0.025)_8 = (0146)_{10}$$
$$\therefore \quad (338.0.25)_8 = (522.0146)_{10}$$

1.7.3 Conversion of Hexadecimal to Decimal

Example 1.6 : *Convert $(438)_{16}$ into decimal.*

Solution : Integer part :

Division	Remainder
438 ÷ 16 = 26	0
26 ÷ 16 = 1	1
10 ÷ 16 = 0	10 = A

LSD ↑ MSD

$$(438)_{16} = (A10)_{16}$$
$$(438)_{16} = (A10)_{16}$$

1.8 REMAINING RADIX CONVERSION

1.8.1 Conversion of Binary to Octal

- Converting binary to octal is also a simple process. Break the binary digits into groups of three starting from the binary point and convert each group into its appropriate octal digit.
- For whole numbers, it may be necessary to add a zero as the MSB in order to complete a grouping of three bits. Note that this does not change the value of the binary number. Similarly, when representing fractions, it may be necessary to add a trailing zero in the LSB in order to form a complete grouping of three.

Example 1.7 : *Converting $(010111)_2$ to Octal*

Solution : 111 = 7 (LSB)

010 = 2 (MSB)

Thus, $(010111)_2 = (27)_8$

Example 1.8 : *Converting $(0.110111)_2$ to Octal*

Solution : 110 = 6 (MSB)

111 = 7 (LSB)

Thus, $(0.110101)_2 = (0.67)_8$

1.8.2 Conversion of Octal to Binary

- The primary application of octal numbers is representing binary numbers, as it is easier to read large numbers in octal form than in binary form. Because each octal digit can be represented by a three-bit binary number (see Table 1.9, it is very easy to convert from octal to binary.
- Simply replace each octal digit with the appropriate three-bit binary number as indicated in the examples below.

Table 1.9 : Octal and Binary Numbers

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

$13_8 = (001011)_2$

$\therefore \quad (37.12)_8 = (011111.001010)_2$

1.8.3 Conversion of Hexadecimal to Binary

- Because each hexadecimal digit can be represented by a four-bit binary number it is very easy to convert from hexadecimal to binary. Simply replace each hexadecimal digit with the appropriate four-bit binary number as indicated in the examples below.

Example 1.9 : $A3_{16} = (10100011)_2$

Solution : $(37.12)_{16} = (00110111\ .\ 00010010)_2$

1.8.4 Conversion of Binary to Hexadecimal

- Converting binary to hexadecimal is another simple process. Break the binary digits into groups of four starting from the binary point and convert each group into its appropriate hexadecimal digit.
- For whole numbers, it may be necessary to add a zero as the MSB in order to complete a grouping of four bits. Note that this addition does not change the value of the binary number.
- Similarly, when representing fractions, it may be necessary to add a trailing zero in the LSB in order to form a complete grouping of four.

Example 1.10 :

(a) Converting $(1010111)_2$ to hexadecimal.

(b) Converting $(0.00111111)_2$ to hexadecimal.

Solution :

(a) 0111 = 7 (LSB)

0101 = 5 (MSB)

Thus, $(1010111)_2 = (57)_{16}$

(b) 0011 = 3 (MSB)

1111 = F (LSB)

Thus, $(0.00111111)_2 = (0.3F)_{16}$

1.8.5 Conversion of Octal to Hexadecimal

- To convert an octal number to hexadecimal follow the steps given below.

Step 1 : First convert octal to its binary.

Step 2 : Then convert binary number into hexadecimal.

Example 1.11 : *Convert $(777)_8$ into hex.*

Solution :

Step 1 : Convert $(777)_8$ into binary.

$(777)_8 = (111111111)$

Step 2 : For equivalent hexadecimal group the binary number into group of 4.

$(777)_8 = (\underline{0001}\ \underline{1111}\ \underline{1111})$

$\downarrow \quad \downarrow \quad \downarrow$

$1 \quad F \quad F$

$\therefore \quad (777)_8 = (1FF)_{16}$

1.8.6 Conversion of Hexadecimal to Octal

- To convert hexadecimal numbers into octal, follow the steps below

Step 1 : First write down given hexadecimal numbers into 4 group of binary numbers.

Step 2 : Then remove the group of four binary.

Step 3 : Make group of 3 binary bits starting from the LSB side.

Example 1.12 : *Convert hexadecimal numbers 5DB into octal.*

Step 1 : Given number is 5DB.

Binary from of 5DB is = 010111011011

Step 2 : Rewrite number with removing group space.

$= (\underline{010}\ \underline{111}\ \underline{011}\ \underline{011})$

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

$2 \quad 7 \quad 3 \quad 3$

$(5DB)_{16} = (2733)_8$

1.9 SIGNED BINARY NUMBER REPRESENTATION

There are different ways to represent binary signed numbers, those are as follows :

- Signed Magnitude
- One's Complement
- Two's Complement
- 9's Complement
- 10's Complement

1.9.1 Signed Magnitude

- The simplest way to indicate negation is signed magnitude. In signed magnitude, the left-most bit is not actually part of the number, but is just the equivalent of a + × / – sign.
- "0" indicates that the number is positive, "1" indicates negative. In 8 bits, 00001100 would be 12 (break this down into (1 × 2 ^3) + (1 × 2 ^2)). To indicate - 12, we would simply put a "1" rather than a "0" as the first bit : 10001100.
- The +ve or –ve signs are also represented in the binary form i.e. by using 0 or 1 so a 0 is used to represent the (+ve) sign and 1 is used to represent (–ve) sign.
- The Most Significant Bit (MSB) of binary number is used to represent the sign and the remaining bits are used for representing the magnitude.

  e.g. 8 bit signed binary number show in Fig. 1.15.

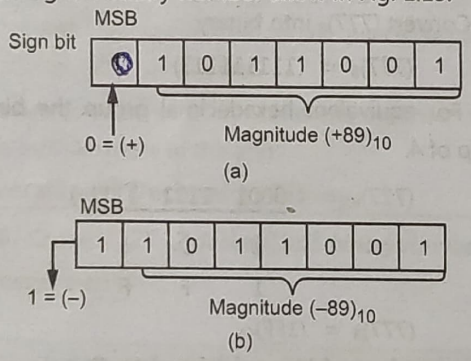


Fig. 1.15 : 8-Bit signed binary numbers

- This type of number is called ????? number or signed magnitude number.
- For an 8 bit sign magnitude number the largest negative number is (–127) to largest positive, number is (127) i.e. from 0 to 255.

Advantages :

- We can easily find out the magnitude by deleting the sign bit.
- The simplicity of sign magnitude.

Disadvantage :

- Signed number require complicated circuits.

1.9.2 One's Complement

- In one's complement, positive numbers are represented as usual in regular binary. However, negative numbers are represented differently. To negate a number, replace all zeros with ones, and ones with zeros-flip the bits. Thus, 12 would be 00001100, and –12 would be 11110011.
- As in signed magnitude, the leftmost bit indicates the sign (1 is negative, 0 is positive). To compute the value of a negative number, flip the bits and translate as before.

1.9.3 Two's Complement

- Begin with the number in one's complement. Add 1 if the number is negative. Twelve would be represented as 00001100, and – 12 as 11110100. to verify this, let's subtract 1 from 11110100, to get 11110011. If we flip the bits, we get 00001100, or 12 in decimal.

Example 1.13 : *Represent the decimal numbers 25 and – 25 in the 8 bit signed magnitude 1's complement and 2's complement forms.*

Solution :

Step 1 : Representation of 25 signed magnitude

$$25 = 0\ \underbrace{0011001}_{\text{magnitude}}$$

$$\downarrow \quad \downarrow$$

$$\text{sign}$$

Step 2 : Representation of –25 in signed magnitude form.

$$25 = 1\ \underbrace{0011001}_{\text{magnitude}}$$

$$\downarrow \quad \downarrow$$

$$\text{sign}$$

Step 3 : Representation of – 25 in 1's complement form

$$25 = 00011001 \ldots \text{sign magnitude}$$

$$\downarrow$$

invert all bits

$$-25 = 11100110 \ldots 1\text{'s complement form}$$

Step 4 : Representation of – 25 in 2's complement form

$$25 = 00011001 \ldots \text{sign magnitude}$$

invert all bits + add 1

$$-25 = 11100111 \ldots 2\text{'s complement form}$$

1.9.4 9's Complement

- The 9's complement of a number can be obtained by subtracting every digit of a number by 9.
- Consider some numbers such as 6, 27, 234, 672 and the 9's complement of these numbers can be obtained as :-

9-6 = 3, 99-27 = 72, 999-234 = 765, 999-672 = 327, thus 3, 72, 765, 327 are the 9's complement of the numbers described above.

1. BCD Subtraction Using 9's Complement :

The steps for 9's complement method :

- Find the 9's complement of negative number.
- Add two numbers using BCD addition.
- If carry is generated add carry to the result otherwise find 9's complement of the result.

Example 1.14 : *Subtract $(2)_{10}$ from $(7)_{10}$ in BCD.*

Solution :

Step 1 : Obtain 9's complement of $(2)_{10}$

```
  9
- 2
---
  7
```

Step 2 : Add 7 into 9's complement of 3.

	7 + 7	...9's complement of 2
end around...	1 4 + 1	...sum ...add carry
	5	...final result

Example 1.15 : *Subtract $(485)_{10}$ from $(228)_{10}$ in BCD.*

Solution :

Step 1 : Obtain 9's complement of $(485)_{10}$

```
  999
- 485
-----
  514   (9's complement of 485)
```

Step 2 : Add 228 to 9's complement of 485

```
  228
+ 514 (9's complement of 485)
-----
  742 (No carry indicates –ve value)
- 257 (9's complement of result)
```

1.9.5 10's Complement

The 10's complement of the any decimal number can be obtained by adding 1 to the 9's complement of the same number.

Steps to Find 10's Complement Method :

- Find the 9's complement of the number by subtracting every digit of the number from 9.
- The number thus obtained by subtraction will be added with 1.
- Thus, 10's complement of the number is obtained.
- Therefore, 10's complement is 9's complement + 1.

Consider some decimal numbers 8, 35, 567, 3457, now let's find 9's complement of each of these.

9-8 = 1, 99-35 = 64, 999-567 = 432, 9999-3457 = 6542

Thus, 1, 64, 432 & 6542 are the 9's complement of the above considered numbers. Now, in order to find 10's complement of these numbers let's add 1 to each of these numbers.

1 + 1 = 2, 64 + 1 = 65, 432 + 1 = 433, 6542 + 1 = 6543

Thus, 2, 65, 433 & 6543 are the 10's complement of the numbers 8, 35, 567, 3457 respectively.

In 10's Complement Two Cases will Arrive which are as Follows :

(i) When a smaller number is to be subtracted with the larger number, in this case one carry will be generated. Ignore this carry and the rest of the digits of the addition will be the answer.

(ii) When a larger number is to be subtracted from the smaller number, in this case the answer will be negative. There will be no carry generation after addition of the 10's complement of subtrahend with minuend. This indicates that the resultant answer is negative. The final answer can be evaluated by taking the 10's complement of the number which is obtained after addition of 10's complement of subtrahend with minuend.

Example 1.16 : *Subtract $(485)_{10}$ from $(228)_{10}$ in BCD.*

Solution :

Step 1 : Obtain 10's complement of $(485)_{10}$

```
  999
- 485
-----
  514   (9's complement of 485)
+   1
-----
  515 (10's complement of 485)
```

Step 2 : Add 228 to 10's complement of 485

```
  228
+ 515
-----
  743 (No carry so answer is in negative.
       Take 10's Complement answer)
```

Step 3 : Take 10's complement of 743.

```
  999
- 743
-----
  256
+   1
-----
- 257
```

1.10 CODES

1.10.1 Numeric Codes

Alphanumeric codes are also called as character codes. This code writes alphanumeric data, including data, letters of the alphabet, numbers, mathematical symbols and punctuation marks in the form which can be easily understandable and processed by the computers. With this code the input output devices such as keyboard, monitors, mouse can be interfaced with computer. The most common alphanumeric codes used these days are ASCII code, EBCDIC code and Unicode.

- **ASCII Code (American Standard Code for Information Interchange) :** It is a seven bit code based on the English alphabet. This code represents 128 characters.
- **EBCDIC Code (Extended Binary Coded Decimal Interchange Code) :** IBM invented this code to extend the Binary Coded Decimal. It is an 8 bit code and therefore can accommodate 256 characters.
- **Unicode :** Also known as universal code. It is 16-bit code so it can represent 65536 different characters.

1.10.2 Weighted and Non-Weighted Code

1. Weighted Codes

- The code which obeys the positional weighting principles, each position of the number represents a specific weight is called as weighted binary code. In these codes each decimal digit is represented by a group of four bits. The main characteristic of weighted code is that each binary digit is assigned a specific weight.
- In weighted codes, each digit is assigned a specific weight according to its position. For example, in 8421/BCD code, 1001 the weights of 1, 1, 0, 1 (from left to right) are 8, 4, 2 and 1 respectively.
- **Weighted Codes are Used in :**
  - Data manipulation during arithmetic operation.
  - For input/output operations in digital circuits.
  - To represent the decimal digits in calculators, volt meters etc.

Examples : 8421, 2421, HEX, Octal are all weighted codes.

For weighted code specific weight is associated with each bit.

Bit Position	Associated Weight
0	b^0=1
1	b^1
2	b^2
3	b^3

For base b, for example hexadecimal has b=16, to convert a hexadecimal number into decimal you just need to multiply each bit with its positional weight.

Ex. 1A21h = 1 × 16^3 + A × 16^2 + 2 × 16^1 + 1 × 16^0 = 6689

2. Non-Weighted Codes :

- Non weighted codes are those codes in which the digit value does not depend upon their position i.e., each digit position within the number is not assigned fixed value
- **Non-Weighted Codes are Used for :**
  - To perform certain arithmetic operations.
  - Shift position encodes.
  - Error detecting purpose

Example : Gray code, Excess-3 code.

- Gray Codes is minimum error code since its two nearer codes are differ by 1 bit only.
- Gray of 2 is 0011 and 3 is 0010 and so on.
- Excess 3 of 2 i.e 0010 is 5 i.e. 0101. (0010 + 0011 = 0101). To get Excess 3 code add 0011 (3) to all the 8421 coded numbers.

1.10.3 Sequential Code

These are those codes in which each succeeding code is one binary number greater than its preceding code.

This property is used for mathematical manipulation of data.

Example : BCD And Excess - 3 Code.

1.10.4 Self Complimenting Codes

- A code is said to be self-complementary if the code for 9's complement of N i.e. 9 – N can be obtained by interchanging all 0's and 1's.
- For a code to be self complementing, the sum of all its weights must be 9. The digit 8421 and 5421 codes are not self complementing codes whereas 5211,2421,3321,4221 are self complementing code.
- In general, a code is self-complementary if we produce a code by taking the first complement of the digit which is same as 9's complement of the number.

1.10.5 Cyclic Codes

Cyclic codes are those in which each successive code word differs from the preceding one in only one bit position. They are also called unit distance codes.

Example : Gray code.

1.10.6 Reflective Codes

- A code is said to be reflective code, if the code word of the 9's complement of N i.e. 9- N can be obtained from code word of N by interchanging all 1's as 0's and 0's as 1's i.e code word of 9 is the complement for the code 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4.
- In this code simply change in one bit, this code is reflected about the center entries with one bit changed. The 9's complement of a reflected BCD code word is formed simply by changing only one of its bit.

  Examples of reflective codes are 2421 code, 5211 code and Ex-3 code.

Example : In 642-3 code the decimal 7 is represented as = 1101

9 – N = 9 – 7 = 2 decimal 2 is represented as = 0010

i.e 9's complement of 7 can be obtained just by complementing each bit in '7'.

1.10.7 BCD Code

- Each digit of decimal number is represented by four bits. For example digit '5' is represented as '0101'. The BCD code is also called as 8-4-2-1 code where 8,4,2 and 1 represent weights of binary symbol in the respective positions. The examples of BCD codes are given below ;

Decimal	4	2	8	6	3
BCD	0100	0010	1000	0110	0011

- BCD code for 0 to 9 digits are given as;

Decimal Digit	BCD Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

The remaining 4 digit binary representations i.e. 1010, 1011, 1100, 1101, 1110 and 1111 are invalid BCD codes.

1. **BCD Arithmetic :** The arithmetic of BCD code is complex. It can be used to perform addition and subtraction.

Rules :

- If four bits sum is equal to or less than 9, no correction is needed. The sum is in proper BCD form.
- If the four bit sum is greater than of 9 or if a carry is generated from the four bit sum, the sum is invalid.
- To correct the invalid sum, add 6 $(0110)_2$ to the four bit sum. If carry results from this addition, add it to the next higher order BCD digit.

2. **BCD Addition**

- In BCD addition, each digit of decimal number is first represented using it's four - bit BCD equivalent numbers.
- The addition of two BCD numbers is carried out using simple binary addition. After the binary addition the result may be invalid or valid BCD.
- If the result is invalid BCD, then it is converted into the valid BCD by adding $(0110)_2$ or $(06)_{10}$. If carry is generated after the addition of $(0110)_2$ then it is added to next bit.

Example 1.17 : *$(569)_{10}$ + $(687)_{10}$*

Solution :

Step 1 :

$(569)_{10}$ = 0101 0110 1001

+ $(687)_{10}$ = 0110 1000 0111

0111 1111 0000

invalid BCD | invalid BCD | valid BCD with carry 1

Step 2 : Add $(6)_{10}$ to each one

0111 1111 0000

+ 0110 0110 0110

10010 0101 0110

↓

1 2 5 6

$(569)_{10}$ + $(687)_{10}$ = $(1256)_{10}$

3. **BCD Subtraction**

- Subtraction is nothing but addition of a signed number i.e. A – B = A + (–B). The negative BCD number can be expressed by taking the 9's or 10's complement of the BCD number which is to be subtracted.

(i) BCD Subtraction Using 9's Complement

- The 9's complement is obtained by subtracting the given number from 9. Thus 9's complement of 3 is 6 (9 – 3 = 6).

Steps to Perform BCD Subtraction Using 9's Complement.

- Find the 9's complement of subtractor.
- Perform the BCD addition of the first number and 9's complement of second number.
- If carry is generated, then result is positive. Add the carry into the result to get the correct result.
- If carry is not generated, then result is negative and it is in 9's complement form.

Example 1.18 : *Subtract 4 from 8 in BCD (8 – 4 – 2 – 1) using 9's complement of subtraction.*

Solution : 8 – 4 = 8 + 5 where 5 is 9's complement of 4

Step 1 :

1000 ← BCD code of 8

+

0101 ← BCD code of 5

1101 ← Invalid BCD answer

0110 ← add $(0110)_2$

11 ← Carry

10011

Because carry is generated, the answer is positive

Step 2 : Add carry into answer

0011

+

1

1

0100 ← valid BCD answer

(ii) BCD Subtraction Using 10's Complement

The 10's complement of a number is obtained by adding '1' into 9's complement.

Step to Perform BCD Subtraction Using 10's Complement.

- Find the 10's complement of the subtraction.
- Perform the BCD addition.
- If carry is not generated, then the result is negative and it is in 10's complement form.
- If carry is generated, then the result is positive. Discard the carry.

Example 1.19 : *Perform 8 – 3 using 10's complement.*

Solution : 8 – 3 = 8 + 7

where 7 is 10's complement of 3 i.e. [(9 – 3) + 1 = 7]

1000 ← BCD code of 8

+ 0111 ← BCD code of 7

1111 ← Invalid BCD code

0110 ← add $(0110)_2$

11 ← Carry

10101

The carry is generated.

∴ The result is positive. Discarded the carry

∴ The answer is (5).

Example 1.20 : *Represent $(7)_{10}$ using all the weighted 4-bit BCD codes*

Solution :

1. 3321 code $(7)_{10}$ = 3 3 2 1
   ↓ ↓ ↓ ↓
   1 1 0 1
   = $(1101)_{3221\ BCD}$
2. 4221 code $(7)_{10}$ = 4 2 2 1
   ↓ ↓ ↓ ↓
   1 0 1 1
   = $(1011)_{4221\ BCD}$
3. 5211 code $(7)_{10}$ = 5 2 2 1
   ↓ ↓ ↓ ↓
   1 1 0 0
   = $(1100)_{5211\ BCD}$
4. 5311 code $(7)_{10}$ = 5 3 1 1
   ↓ ↓ ↓ ↓
   1 0 1 1
   = $(1011)_{5311\ BCD}$
5. 5421 code $(7)_{10}$ = 5 4 2 1
   ↓ ↓ ↓ ↓
   1 0 1 0
   = $(1011)_{5421\ BCD}$
6. 6311 code $(7)_{10}$ = 6 3 1 1
   ↓ ↓ ↓ ↓
   1 0 0 1
7. 7421 code $(7)_{10}$ = 7 4 2 1
   1 0 0 0
   = $(1000)_{7421\ BCD}$
8. $74\bar{2}\bar{1}$ code $(7)_{10}$ = $7 + 4 - \bar{2} - \bar{1}$
   ↓ ↓ ↓ ↓
   1 0 0 0
   = $(1000)_{74\bar{2}\bar{1}\ BCD}$

1.10.8 Excess – 3 Code

- Excess – 3 wide is derived from the natural BCD code by adding 3 to each coded number.
- It is non-weighted code.
- Table 1.10 shows excess-3 codes to represent single digit.
- The excess – 3 codes are obtained as follows

Decimal number ⟶ 8421 BCD number $\xrightarrow[0011]{\text{add}}$ Excess-3 code.

Table 1.10

Decimal Number	BCD 8 4 2 1	Excess – 3
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

The excess-3 is also called as sequential code, because each succeeding code is one binary number than its preceding code.

The excess-3 is also known as self complementary because we get the 9's complement of number just complete of number just complement of each bit by means replacing ?

Example 1.21 : *What is Excess-3 code of binary number ?*

0010 B, 0110 B and 0111 B

Solution :

(a) 0010 B

add (0011) to the number

0010

\+ 0011

0101

(b) 0010 B

add (0011) to each number

0110

\+ 0011

1001

(c) 0111 B

add (0011) to the given number

0111

\+ 0011

1010

1.10.9 Gray Code

- Gray is a non-weighted code.
- The feature of gray code is that the only one bit will change each time the decimal number is incremented this is shown in Table 1.11.

Table 1.11

Decimal Code	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

- Gray code also exhibits the reflective property.
- The two least significant bits for decimal 4 to 7 are the mirror images of those for decimal 3 to 0.
- The three least significant bits for decimal 8 through 15 are mirror images of those for decimal 7 through 0.

(i) Application of Gray Codes

- Gray code is mostly used in the shaft position encoders.
- A shaft position encoder produces a code coordinate which represents the angular position of the shaft. The shaft position encoder consists of a light source, an optical disc and a light detector as shown in Fig. 1.16 (a).

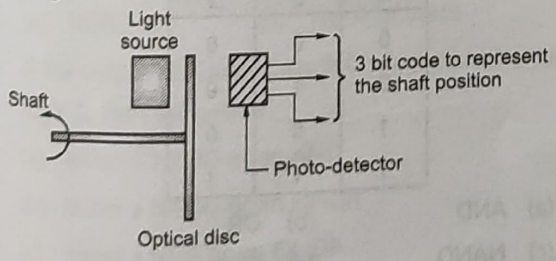
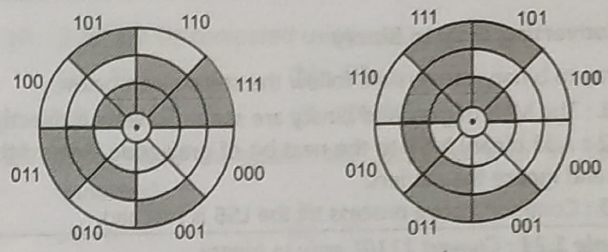


Fig. 1.16 (a) : Shaft position encoder

- Patterns of opaque and transparent segments is etched out on the optical disc. So, corresponding to the black portion, the photo detector produces a "1" and corresponding to the transparent portion, a "0" is produced.
- The patterns on the disc are according to the code required to be produced at the detector output.
- Fig. 1.16 (b) shows the patterns for binary code and Fig. 1.16 (c) shows the pattern for producing gray code.



(b) Pattern for Binary Code **(c) Pattern for Gray Code**

Fig. 1.16

(ii) Advantages of Gray Codes

- The advantage of Gray code over straight binary code is that gray code changes by only 1 bit as it sequences from one number to the next.
- The 3-bit gray code representations for number 0 through 7 are listed below. Design a decoder to convert a 3-bit binary number into a gray code number. Your decoder should have three inputs and three outputs and it should convert a binary number, say 011 (decimal 3), to a Gray code number, 010 for all the eight decimal numbers 0 to 7.

Table 1.12

Decimal	Binary b_2, b_1, b_0	Gray g_2, g_1, g_0
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

(iii) Converting Binary Code to Gray Code

A binary number can be converted in gray by following steps :

Step 1 : First take MSB as it is.

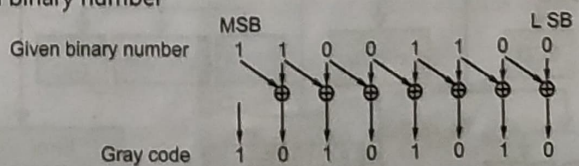
Step 2 : Add this MSB bit to the next position bit, recording the sum and neglecting any carry.

Step 3 : Take successive sums until complete.

Example 1.22 : *Convert binary 11001100*

Solution :

Given binary number



$\therefore \quad (11001100)_2 = (10101010)_{gray}$

In general, the conversion of binary to gray takes place as follows

$$G_3 (MSB) = B_3 (MSB) \qquad G_2 = B_3 \oplus B_2$$

$$G_1 = B_2 \oplus B_1 \qquad (LSB)\ G_0 = B_1 \oplus B_0$$

(iv) Converting Gray to Binary

For gray to binary conversion, follow the steps given below.

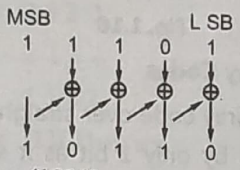
Step 1 : The MSB of gray and binary are same. So write it directly.

Step 2 : Add binary MSB to the next bit of gray code. Record the result and ignore the carriers.

Step 3 : Continue above process till the LSB is reached.

Example 1.23 : *Convert 11101 gray to binary.*

Solution :



$\therefore \quad (11101)_{gray} = (1011)_2$

In general, we can say that the conversion below 4-bit gray number $G_3G_2G_1G_0$ into a 4-bit binary number $B_3B_2B_1B_0$ it takes places as below

B_3 (MSB) $= G_3$ (MSB) $B_2 = B_3 \oplus G_2$

$B_1 = B_2$ $\oplus G_1$ $B_0 = B_1 \oplus G_0$

1.10.10 Error Detecting Codes and Correcting Codes

- When the digital information is transmitted from one system to another, the possibility of a bit being lost (due to transient or any other reason) in transmission becomes the point of interest.
- The transmission errors may occur because of electrical noise in the transmission channel. Due to transmission error a signal transmitted as '0' may be received as '1' or vice-versa. In complex digital systems, millions of bits per second are manipulated and it is desired to have data integrity, or at least a violation of data integrity must be detected.
- To maintain data integrity between transmitter and receiver, extra bits(s) are added to the data. These extra bits allow the detection and sometimes correction of errors in the data.
- The data along with extra bit(s) form the code. The codes which allow only error detection are called "Error Detecting Codes" and codes which allow error detection and correction are called 'error detecting and correcting codes'. Example for error detecting code is parity; in which and error detection and correction or error correcting code is Hamming code.

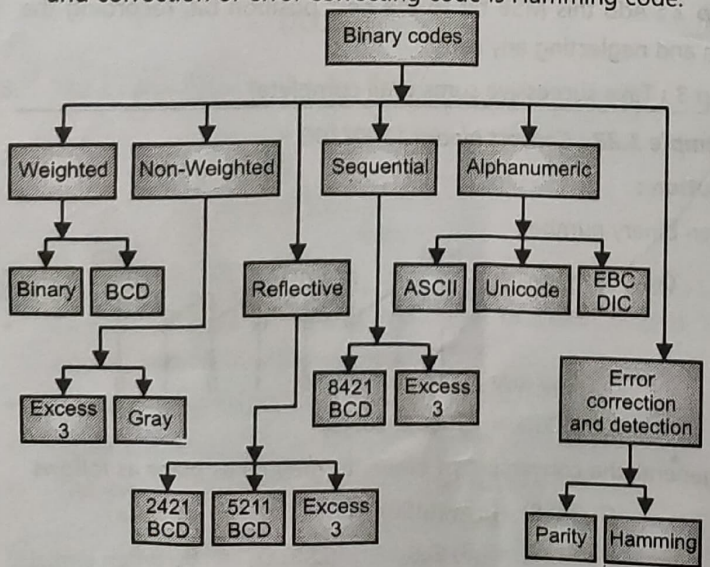


Fig. 1.17

MULTIPLE CHOICE QUESTIONS (MCQ's)

1. Which of the following gates is used for multiplication
   (a) NOR (b) OR
   (c) AND (d) EX-OR
2. The output of OR gate is logic zero when ____.
   (a) two input are logic 1
   (b) anyone input is logic 1
   (c) all inputs are logic 0
   (d) none of these
3. NOT symbol is used for ____ .
   (a) inversion (b) logical inversion
   (c) logic conversion (d) both (b) and (c)
4. Which of the following operators is used for addition?
   (a) NOT (b) NOR
   (c) OR (d) AND
5. For which gate the output is 0 when both inputs are 1?
   (a) NOR (b) NAND
   (c) AND (d) OR
6. The following symbol represents '0' as its output, th inputs must be ____.

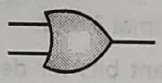


Fig. 1.18

   (a) One input "on" the other off
   (b) both inputs "on"
   (c) both inputs "off"
   (d) none of above
7. The following truth table is for ____ gate.

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

   (a) AND (b) OR
   (c) NAND (d) NOR
8. The logical ____ is represented by the + operator.
   (a) AND (b) OR
   (c) NAND (d) NOR
9. For three input NAND gate the output is '0' only when ____.
   (a) all inputs are logic 1
   (b) all inputs are logic 0
   (c) any one input is logic 1
   (d) none of these

10. XOR is ____
    (a) odd function (b) even function
    (c) both (a) and (b) (d) none of these
11. The implication relation is ____ .
    (a) reflective (b) transitive
    (c) antisymmetric (d) all of these
12. A positive logic AND operation will have a negative logic ____ .
    (a) AND-NOT (b) OR
    (c) NOR (d) OR-NOT
13. In the following Venn diagram representation, X'Y' is represented in which region?

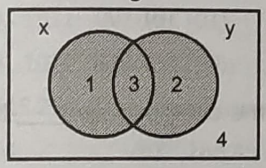


Fig. 1.19

    (a) 1 (b) 2
    (c) 3 (d) 4
14. Total number of min/max terms with 'n' variable is ____.
    (a) n^2 (b) n^n
    (c) 2^n (d) none of these
15. The total number of Boolean functions, which can be realized with four variables is ____.
    (a) 4 (b) 256
    (c) 16 (d) 65536
16. Which of the following does not follow the commutative law?
    (a) XNOR (b) NAND
    (c) NOR (d) None of these
17. If the output of a logic gate is 0 when all its inputs are at logic 1, then the gate is ____ .
    (a) either a NAND or an AND
    (b) either a NAND or an EX-OR
    (c) either a NOR or an EX-OR
    (d) either a NAND or an EX-NOR
18. Match the following:
    (a) (A) EX-OR (1) Commutative
    (b) (B) NAND (2) Odd
    (c) (C) NOR (3) Associative
    (d) (D) OR (4) None
    (e) (a) A2 B1 C1 D3 (b) A1 B3 C2 D4
    (f) (c) A3 B2 C4 D1 (d) A1 B4 C3 D2
19. 2-bit can be compared using ____.
    (a) AND (b) OR
    (c) NAND (d) EX-OR
20. How many functions are possible with n binary variables?
    (a) n (b) n^2
    (c) 2^n (d) 2^{2^n}
21. How many functions are possible with two Boolean variables?
    (a) 2 (b) 4
    (c) 16 (d) none of these
22. Minimum number of two input NAND gates used to perform the function of two input OR gates is ____ .
    (a) 1 (b) 2
    (c) 3 (d) 4
23. The symbol of EX-OR operator is ____

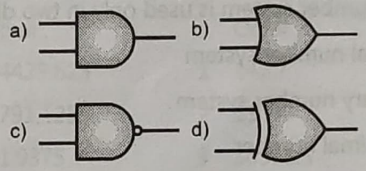


Fig. 1.20

24. If A and B are to be added in logic circuit, it is represented by ____ .
    (a) A + B (b) A · B
    (c) A ⊕ B (d) None of these
25. ____ are the devices used as basic building blocks of all the digital circuit.
    (a) Encoder (b) Logic gates
    (c) MUX (d) All of these
26. The ____ consists of all the possible combinations of the inputs and the corresponding state of output of a logic gate.
    (a) excitation table (b) flow table
    (c) truth table (d) none of these
27. A logic gate has one or more than one inputs and ____ output.
    (a) one (b) many
    (c) both (a) and (b) (d) none of these
28. ____ gate is disabled when its inhibit input is at logic 1.
    (a) OR (b) NAND
    (c) AND (d) All of the above
29. The output of a logic gate is 1 when all it inputs are at logic 1. The gate is either ____
    (a) a NAND or a NOR
    (b) an AND or an OR
    (c) an EX-NOR and an EX-OR
    (d) an AND or a NOR

30. If all the inputs are at logic 0 then output of a logic gate is 1. The gate is either ____
(a) an EX-OR or an EX-NOR
(b) an AND or an EX-NOR
(c) a NAND or a NOR
(d) an OR or an AND

31. ____ system is used in digital computers because all electrical and electronic circuits can be made to respond to the states concept.
(a) Binary number (b) Octal number
(c) Decimal number (d) Hexadecimal number

32. ____ number system has a base of 16.
(a) Binary (b) Octal
(c) Hexadecimal (d) Decimal

33. Which number system is used only in two digits, 0 and 1.
(a) Octal number system
(b) Binary number system
(c) Decimal number
(d) Hexadecimal number system

34. The binary number system is also called a ____.
(a) base system (b) base ten system
(c) base two system (d) base one system

35. In ____ numeral every position has a value 2 times the value of the position to its right.
(a) hexadecimal (b) octal
(c) binary (d) decimal

36. The number of bits in a nibble is ____.
(a) 16 (b) 5
(c) 4 (d) 8

37. Longform of MSD is ____.
(a) Most Significant Digit
(b) Many Significant Digit
(c) More Significant Digit
(d) Both (b) and (c)

38. A group of 8-bit binary numbers is called a ____.
(a) byte (b) nibble
(c) bits (d) word

39. Longform of LSD is ____
(a) Least Significant Digit
(b) Loss Significant Digit
(c) Low Significant Digit
(d) Less Significant Digit

40. How many number of digits are there in a hexadecimal system?
(a) 15 (b) 17
(c) 16 (d) 8

41. A byte corresponds to ____.
(a) 4 nibble (b) 2 nibble
(c) 1 nibble (d) 3 nibble

42. ____ system has a base or radix of 10.
(a) Binary digit (b) Octal digit
(c) Hexadecimal digit (d) Decimal digit

43. The number of digits in octal system is ____.
(a) 10 (b) 7
(c) 8 (d) 16

44. To represent the octal number ____ bit binary numbers are required.
(a) 3 (b) 4
(c) 8 (d) 10

45. ____ system is used in digital computer because all electrical and electronics circuits can be made to respond to the states concepts.
(a) Octal number (b) Binary number
(c) Decimal number (d) Hexadecimal number

46. ____ number is formed from a binary number by grouping bits in groups of 4 bits each.
(a) Binary (b) Octal
(c) Decimal (d) Hexadecimal

47. In a hexadecimal system, the digit F is equivalent to ____ in decimal system.
(a) 16 (b) 15
(c) 8 (d) 17

48. The digit E in hexadecimal number system is equivalent to ____ in binary number.
(a) 1101 (b) 1001
(c) 1110 (d) 1111

49. In hexadecimal system the number FF is equivalent to ____ in decimal system.
(a) 256 (b) 255
(c) 239 (d) 240

50. Which of the following octal numbers is equal to hexadecimal F?
(a) 18 (b) 17
(c) 16 (d) 15

51. Which is the largest decimal number that can be represented using an 8-bit binary number?
(a) 1024 (b) 255
(c) 4096 (d) 256

52. Which of the following hexadecimal number is equivalent to binary 1011010?
(a) 5C (b) 5F
(c) 5B (d) 5A

53. Convert hexadecimal number (2B2D) into equivalent binary number.
(a) $(0011\ 1010\ 0011\ 1101)_2$
(b) $(0010\ 0011\ 1001\ 1101)_2$
(c) $(0010\ 1011\ 0010\ 1101)_2$
(d) $(0010\ 1011\ 1100\ 1100)_2$

54. Convert $(615.25)_8$ to its hexadecimal equivalent.
(a) 18D.54 (b) 18B.45
(c) 18D.45 (d) 19D.54

55. The octal equivalent of $(564)_{10}$ is ____.
(a) $(1064)_8$ (b) $(1066)_8$
(c) $(1065)_8$ (d) $(1062)_8$

56. Convert $(1101001011)_2$ into decimal number.
(a) $(6451)_{10}$ (b) $(843)_{10}$
(c) $(1513)_{10}$ (d) $(243)_{10}$

57. Convert $(12.75)_{10}$ into binary number.
(a) $(1011.10)_2$ (b) $(1100.11)_2$
(c) $(1010.11)_2$ (d) $(1010.10)_2$

58. Convert $(2768)_{10}$ in to base 16.
(a) $(11310)_6$ (b) $(10131)_{16}$
(c) $(AD0)_{16}$ (d) $(0DA)_{16}$

59. Convert binary number 00001100 into base 16.
(a) CH (b) DH
(c) A1H (d) 4CH

60. Convert A23E into binary number.
(a) 1010 0010 0111 1010
(b) 1010 0010 0011 1111
(c) 1010 0010 0011 1110
(d) 1111 0011 0010 1000

61. Octal equivalent of decimal 324.987 is ____.
(a) 640.781 (b) 815.234
(c) 70.771 (d) 504.771

62. When 2n-bit binary numbers are added the sum will contain at the most ____
(a) n bits (b) n + 1 bits
(c) n + 2 bits (d) n + 3 bits

63. AB7A + 12E = ____.
(a) ACA8 (b) 12AB
(c) CACA (d) DEAC

64. The number $(123456)_8$ = ____.
(a) $(A72E)_{16}$ and $(22131122)_4$
(b) $(A72E)_{16}$ and $(22130232)_4$
(c) $(A73E)_{16}$ and $(22130232)_4$
(d) None of these

65. If $(33)_{10} = (201)_x$, X is ____.
(a) 2 (b) 3
(c) 4 (d) 5

66. 10^{th} element of base 3 number system is ____.
(a) 22 (b) 12
(c) 100 (d) none of these

67. Match the following:

Decimal		Octal
(a) 4429.625	1	142.7
(b) 791.125	2	13.74
(c) 1.9375	3	10515.5

68. Which of the following is not an octal number?
(a) 723 (b) 123
(c) 101 (d) 13 A

69. 1's complement of $(01110001)_2$ = ____.
(a) $(10001111)_2$ (b) $(10000111)_2$
(c) $(10001110)_2$ (d) $(10001101)_2$

70. 2's complement of −8 is ____.
(a) 0111 (b) 1000
(c) 01000 (d) 10100

71. The sign bit '0' represents a ____.
(a) positive number (b) negative number
(c) unsigned number (d) none of above

72. A negative number is represented by a ____.
(a) 0 in the sign bit (b) 1 in the sign bit
(c) both (a) and (b) (d) none of the above

73. 2's complement of $(10110010)_2$ is ____.
(a) $(01001110)_2$ (b) $(10110010)_2$
(c) $(11001111)_2$ (d) $(10001110)_2$

74. What is the maximum positive number of 1's complement number system?
(a) $(2^{n-1} - 1)$ (b) $-(2^{n-1} - 1)$
(c) (2^{n-1}) (d) None of these

75. 1's complement of $(11010101)_2$ is ____.
(a) (00101010) (b) 00101011
(c) (11011111) (d) 00111011

76. Solve 88-99 using 2's complement method.
(a) $(11100101)_2$ (b) $(11101010)_2$
(c) $(11110101)_2$ (d) $(11101011)_2$

77. In digital computers binary subtraction is performed ____.
(a) using 2's complement
(b) using 10's complement
(c) using 16's complement
(d) in the same way like subtraction decimal number system

78. BCD coded numbers express each digit as ____.
(a) 1 bit (b) nibble
(c) 1 byte (d) none of these

79. Match the following:
(A) Self complementing code (1) Internet code
(B) Minimum error code (2) Excess 3
(C) Unicode code (3) Gray code
(a) A1 B2 C3 (b) A2 B3 C1
(c) A3 B2 C1 (d) none of these

80. Match the following
(A) Gray code (1) Floating point multiplication
(B) Sign extension (2) Artificial intelligence
(C) 2's representation for 0 (3) Sign magnitude
(4) None
(a) A2 B1 C3 (b) A2 B2 C4
(c) A4 B1 C3 (d) A1 B2 C4

81. ASCII code is ____ .
(a) 4 bit code (b) 14 bit code
(c) 8 bit code (d) 7 bit code

82. An n-bit gray code can be obtained by reflecting an ____ bit code?
(a) n (b) n + 1
(c) n – 1 (d) None of these

83. BCD addition of A(0101) and B(0110) gives ____?
(a) 1001 (b) 1011
(c) 1000 (d) None of these

84. The decimal number 241 is represented in Binary Coded Decimal system as ____.
(a) 100100100 (b) 1000100010
(c) 100100001 (d) 001001000001

85. Which of the following is an unweighted code?
(a) 8421 code (b) 5211 code
(c) Excess 3 code (d) 2421 code

86. Match the following:

Excess 3-code	Decimal
(A) 1000	(1) 0
(B) 0011	(2) 2
(C) 0100	(3) 4
(D) 0101	(4) 1
(E) 0111	(5) 5

(a) A4 B3 C2 D1 E5 (b) A5 B1 C4 D2 E3
(c) A3 B1 C2 D4 E5 (d) none of these

87. Which code is used to represent numbers and characters?
(a) ASCII (b) Hollerith
(c) EBCDIC (d) All of these

88. In ____ code each digit position of the number represents a specific weight.
(a) alphanumeric (b) non-weighted
(c) weighted (d) error detecting

89. Which is 7-bit alphanumeric code?
(a) ASCII (b) Hollerith
(c) BCD (d) EBCDIC

90. Which of the following is a weighted code?
(a) Gray (b) BCD
(c) Reflective (d) Excess 3

91. In ____ code each successive code differs from its preceding code by a single bit only.
(a) gray (b) binary
(c) BCD (d) excess-3

92. Gray code is ____.
(a) reflective code (b) non-weighted code
(c) both (a) and (b) (d) none of these

93. Which is not a non-weighted code?
(a) Excess-3 (b) Gray
(c) Five-bit BCD code (d) Binary

94. The Excess-3 subtraction of 8-5 is ____.
(a) 0110 (b) 1100
(c) 0111 (d) 1001

95. Which code can be derived from the natural BCD code by adding 3 to each coded number?
(a) Gray (b) Excess-3
(c) Binary (d) Gray-3

96. Find out gray code for the 11001100 binary number.
(a) 10101010 (b) 11001010
(c) 11011101 (d) 10111011

ANSWERS

1. (c)	2. (c)	3. (d)	4. (c)	5. (d)
6. (c)	7. (a)	8. (b)	9. (a)	10. (a)
11. (d)	12. (b)	13. (d)	14. (c)	15. (d)
16. (a)	17. (b)	18. (a)	19. (d)	20. (d)
21. (c)	22. (c)	23. (b)	24. (a)	25. (b)
26. (c)	27. (a)	28. (a)	29. (b)	30. (c)
31. (a)	32. (c)	33. (b)	34. (c)	35. (c)
36. (c)	37. (a)	38. (a)	39. (a)	40. (c)
41. (b)	42. (d)	43. (c)	44. (a)	45. (b)
46. (d)	47. (b)	48. (c)	49. (b)	50. (b)
51. (b)	52. (d)	53. (c)	54. (a)	55. (a)
56. (b)	57. (b)	58. (c)	59. (a)	60. (c)
61. (d)	62. (b)	63. (a)	64. (a)	65. (c)
66. (c)	67. (b)	68. (d)	69. (c)	70. (b)
71. (a)	72. (b)	73. (a)	74. (a)	75. (a)
76. (c)	77. (a)	78. (b)	79. (b)	80. (a)
81. (d)	82. (c)	83. (b)	84. (d)	85. (c)
86. (b)	87. (d)	88. (c)	89. (a)	90. (b)
91. (a)	92. (c)	93. (d)	94. (a)	95. (b)
96. (a)				

EXERCISE

1. Which gates are known as universal gates ? Justify using examples.
2. Convert following number into decimal. $(10110.0101)_2$
3. Perform the following operation $(1011.101)_2$
4. Perform the following operation $(1001.10)_2 = (\ )_{10}$
5. Convert the following octal number into its equivalent decimal.
   (a) (555) octal (b) (777) octal
6. For a maximum 3-digit octal number obtain a equivalent decimal number.
7. Convert the following into its equivalent decimal numbers. $(327.4051)_8$
8. Express following number in decimal. Show your step by step calculation. $(16.5)_{16}$
9. Convert the following :
   1. $(BF8)_{16} = (\ )_{10}$ 2. $(1\ FFF)_{16} = (\ )_{10}$
   3. $(5A.FF)_{16} = (\ )_{10}$ 4. $(3\ FFF)_{16} = (\ )_{10}$
10. Express the following numbers in binary
    (i) $(1010.11)_{Decimal}$ (ii) $(428.10)_{Decimal}$
11. Express the following numbers in binary
    (i) $(2948)_{10}$ (ii) $(1101.11)_{10}$
12. Convert the following number into octal form $(3287.51)_{10}$
13. Convert the following number into decimal number.
    (i) $(1FFF)_{16}$ (ii) $(BF8)_{16}$
14. Convert the following number into decimal number.
    (i) $(5A.FF)_{16}$ (ii) $(3FFF)_{16}$
15. Convert the following octal number into octal.
    (i) $(11111001.1001100101)_2$
16. Convert the following octal number into binary.
    (a) $(123456)_8$ (b) $(5726.34)_8$
    (c) $(337)_8$ (d) $(0.53652)_8$
17. Convert the following hexadecimal to Binary numbers.
    (1) A72E (2) BD6.7 (3) 0.AF54 (4) DF (5) FF
18. Convert $(11011011101)_2$ into hex. number.
19. Convert the following octal numbers into its equivalent hex.
    (i) $(555)_{octal}$ (ii) $(777)_{octal}$
20. Convert $(36)_8$ into hexadecimal.
21. Convert $(377)_8$ into hexadecimal.
22. Convert hexadecimal numbers into octal numbers.
    (1) A72E (2) BD6.7 (3) 0.AF54 (4) DF (5) FF
23. What do you mean by signed magnitude representation of a number ?
24. What are different ways of representing signed binary numbers ? Explain with examples ?
25. Write short note on BCD code ?
26. Prepare a table 4 bit gray code along with relationship with binary code.
27. What will be the gray code 4-bit binary number ?
28. What is Gray code covert binary number.
    1100B, 0111B and 1101 B into gray number.

29. What are the advantages of gray code over pure binary code ?
30. State the applications of gray code or How gray codes are useful in digital system ?
31. Find gray codes for the following binary number.
    (i) 11001100 (ii) 01011110
32. Explain rule for any sequence binary to gray code conversion.
33. Explain rule for any sequence gray to binary code conversion.
34. Represent 37 into BCD.
35. With the help of suitable example, explain the meaning of self complementing code.
36. State and prove any two theorem of Boolean algebra.
37. Explain different types of number systems ?

✠ ✠ ✠

Unit II

COMBINATIONAL DIGITAL CIRCUITS

2.1 INTRODUCTION

- In this unit we discuss about standard representations for logic functions, K – map representation of logic functions (SOP and POS forms), minimization of logical functions for min-terms and max-terms (upto 4 variables), don't care conditions. This unit also covers Quine – McClusky reduction technique.
- This unit also covers design of CLC using SSI chips, code converter, half and full adder, half and full subtractor.
- We will also see MSI chips: Multiplexers (IC 74153, 74151), Decoder / Demultiplexer (IC 74138), Encoder, (IC 74147), ALU, Comparator, Parity generators/checkers.

2.2 LOGIC CIRCUITS

- Digital circuits or logical circuits are classified into two types : combinational circuits and sequential circuits.
- Sequential circuits are further divided into two groups : synchronous and asynchronous circuits.
- The following diagram shows the classification of logic circuits.

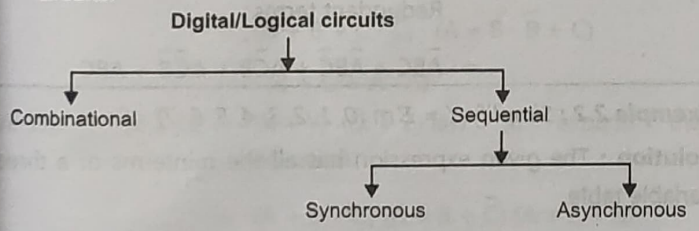


Fig. 2.1 : Classification of logic circuits

2.2.1 Combinational Circuit

- Combinational circuit uses logic gates to implement or satisfy a given Boolean expression.
- The output of combinational circuit is a logical function of the input variables.
- This can be shown as in Fig. 2.2.

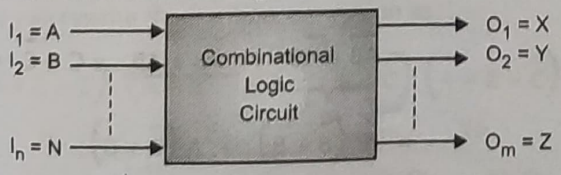


Fig. 2.2 : Block diagram of combinational logic circuit

- The combinational logic circuit has only logic gates and does not require memory. It's operation depends upon the present state of inputs and not on the previous (history) state of the inputs.

2.2.2 Sequential Logic Circuit

- Sequential circuit is designed using combinational logic circuit and memory elements. There are requirements in digital applications where the output variables depend on the sequence in which the input variables are received.
- For example, counter, a counter circuit counts input pulses and hence it has to remember number of input pulses it has received so far.
- This dependency on previous input conditions needs to be stored in memory.
- Combinational logic circuit cannot store data and therefore, we require memory elements along with combinational circuit to design sequential logic circuit.
- This is shown in Fig. 2.3.
- The output of a sequential logic circuit depends upon the present inputs as well as the last state of inputs and outputs.

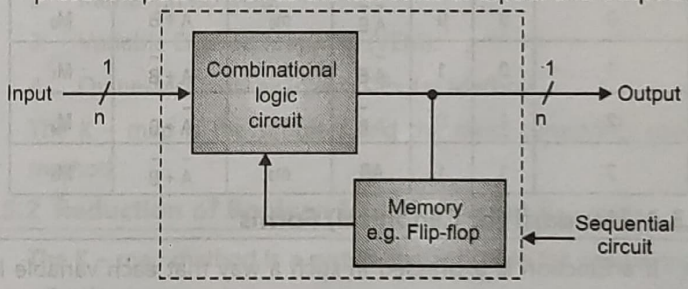


Fig. 2.3 : Block diagram of sequential circuit

- The present state stored in the memory and external inputs are logically evaluated in the combinational block to determine the outputs.

2.3 SUM OF PRODUCTS (SOP) FORM

2.3.1 Standard Terms and Standard (or Canonical) Forms

- The object of a Boolean algebra is to describe the behaviour and logic structure.
- The behaviour of the logic circuit can be expressed in standard forms using standard terms.

2.3.2 Sum Term or Maximum Term (M)

- The output of OR gate is called **sum** term.
- In OR gate, the output is logic '1' for maximum number of combinations of inputs.
- So, the output of OR gate is also called Maximum term or Maxterm (M).

- A sum term of any 'n' variable functions containing all the 'n' literals is called a maxterm. The 'n' variables functions have 2^n maxterms.
- These are denoted as $M_0, M_1, M_2, \ldots M_n$.
- Each variable taking value '0' appears in un-complemented form in maxterm and each variable taking '1' value appears in complemented form.

2.3.3 Product or Minimum Term (m)

- The output of AND gate is called product term.
- In AND gate, the output is logic '1' for minimum number of combinations of inputs. So, the output of AND gate is also called 'Minimum term or minterm (m).
- A product term of any 'n' variable functions containing all literals is called a minterm. The 'n' variable functions have 2^n minterms. These are denoted as $m_0, m_1, m_2, \ldots, m_n$.
- In minterms, each variables taking value '1' appears in uncomplemented form.

Table 2.1 : Maxterms and Minterms of Two-Variables

Decimal Equivalent	Variables		Minterms		Maxterms	
	A	B	m_i	Notation	M_i	Notation
0	0	0	$\bar{A}\bar{B}$	m_0	$A + B$	M_0
1	0	1	$\bar{A}B$	m_1	$A + \bar{B}$	M_1
2	1	0	$A\bar{B}$	m_2	$\bar{A} + B$	M_2
3	1	1	AB	m_3	$\bar{A} + \bar{B}$	M_3

2.3.4 Standard (or Canonical) Forms

- If a function is expressed in such a way that each variable is present in each term.

2.3.5 Sum-of-Products (SOP) Form

- The output of AND gate is called product term.
- The output of OR gate is called sum term.
- The output of AND-OR gate circuit is called Sum-Of-Products (SOP) form.
- Consider the equation,

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

- Each term in the equation is called the fundamental minterm. From table mentioned earlier, the output Y can be written as,

$$Y = m_1 + m_3 = \Sigma m_1, m_3$$
$$= \Sigma m_i$$

where, $i = 1, 3$

$$= \Sigma 1, 3$$

- The SOP form can be converted to standard SOP form by ANDing the terms in the expression with terms formed by ORing.
- The variable and its complement which are not present in that term.
- Following steps are followed to convert a given SOP form to standard SOP form :
  - ➢ Write down all the terms.
  - ➢ If one or more variables are missing in any term, expand that term by multiplying it with the sum of each one of the missing variable and its complement.

For example, $Y = AB + A\bar{B}C$

- The variable C is missing in first term. So, multiply the first term by $(C + \bar{C})$.

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

- Drop out the redundant terms.

SOLVED EXAMPLES

Example 2.1 : *Convert, $Y = \bar{A}B + A\bar{C} + B\bar{C}$ into standard SOP form.*

Solution :

$$Y = \bar{A}B + A\bar{C} + BC$$
$$= \bar{A}B(C + \bar{C}) + A\bar{C}(B + \bar{B}) + BC(A + \bar{A})$$
$$= \bar{A}BC + \bar{A}B\bar{C} + A\bar{C}B + A\bar{C}\bar{B} + BCA + BC\bar{A}$$
$$= \underbrace{\bar{A}BC} + \bar{A}B\bar{C} + A\bar{C}B + A\bar{C}\bar{B} + ABC + \underbrace{\bar{A}BC}$$

Redundant terms

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

Example 2.2 : *Simplify $Y = \Sigma m\,(0, 1, 2, 3, 4, 5, 6, 7)$*

Solution : The given expression has all the minterms of a three variable table.

$$\therefore \quad Y = \underline{\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C} + \underbrace{\bar{A}B\bar{C} + \bar{A}BC}$$
$$+ \underbrace{A\bar{B}\bar{C} + A\bar{B}C} + \underline{AB\bar{C} + ABC}$$
$$= \bar{A}\bar{B}(C + \bar{C}) + \bar{A}B(\bar{C} + C)$$
$$+ A\bar{B}(\bar{C} + C) + AB(\bar{C} + C)$$
$$= \underbrace{\bar{A}\bar{B} + \bar{A}B} + \underline{A\bar{B} + AB} \quad \therefore C + \bar{C} = 1$$
$$= \bar{A}(\bar{B} + B) + A(\bar{B} + B)$$
$$= \bar{A} + A$$
$$Y = 1$$

$\therefore$ The answer is always 'true' (1) because the given expression contains all possible minterms.

2.4 PRODUCT OF SUMS (POS) FORM

2.4.1 Product-Of-Sums (POS) Form

- The output of OR-AND gate circuit is called Product-Of-Sums (POS) form.

Consider the equation,

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

$$Y = M_0, M_2 = \pi (0, 2)$$

where, π stands for the product of maxterms.

- The POS form can be converted to standard POS form by ORing the terms in the expression with terms formed by ANDing the variable and its complement which are not present in that term.
- Following steps are followed to convert a POS form to standard POS form.
  - ➢ Write down all the terms.
  - ➢ If one or more variables are missing in any sum terms, expand that term by adding the products of each of the missing term and its complement.
  - ➢ Drop out the redundant terms.

Example 2.3 : *Convert $Y = (A + B) \cdot (A + C) \cdot (B + \bar{C})$ into standard POS form.*

Solution :

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

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

We use $X + YZ = (X + Y) \cdot (X + Z)$ law to expand the equation.

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

$$= (A + B + C)\,(A + B + \bar{C})\,(A + \bar{B} + C)\,(\bar{A} + B + \bar{C}) \quad (\because \text{Redundant terms})$$

Example 2.4 : *Simplify the following three variable expression.*

$$Y = \pi M\,(1, 3, 5, 7)$$

Solution : The given Boolean expression is in POS from. From the table, we can rewrite the Boolean expression as

$$Y = \underset{N_1}{(A + B + \bar{C})}\ \underset{N_3}{(A + \bar{B} + \bar{C})}\ \underset{N_5}{(\bar{A} + B + \bar{C})}\ \underset{N_7}{(\bar{A} + \bar{B} + \bar{C})}$$

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

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

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

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

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

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

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

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

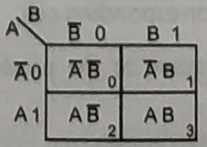
2.5 REDUCTION TECHNIQUES

2.5.1 Boolean Algebra Simplification Technique

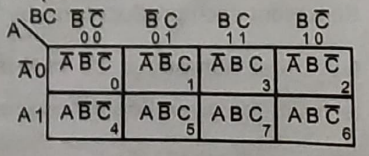
- A good digital circuit must have minimum number of logic gates.
- Less number of gates means minimum propagation delay, skew, power dissipation.
- The number of logic gates can be reduced only if the number of terms in the Boolean expression can be reduced.
- There are four methods that are used to simplify or reduce the Boolean equations.
  1. Algebraic (Boolean Laws, DeMorgan's Theorems).
  2. Karnaugh (K) Map.
  3. Variable Entered Mapping (VEM).
  4. Quine-McClauskey (Q-M) Tabular Method.
- The K – map is the simplest and the most commonly used method.

2.5.2 Reduction of Boolean Equation using K – map

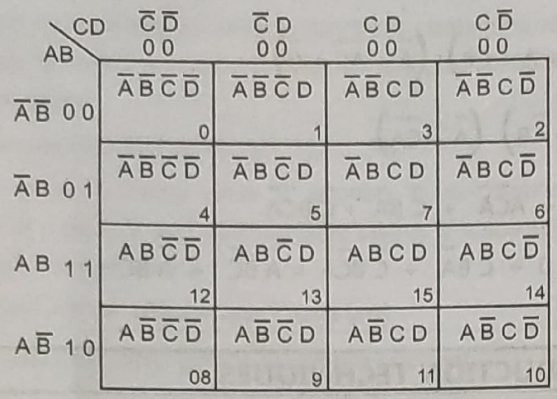
- The K – map method is a systematic approach for simplifying a Boolean expression.
- This method was proposed by Veitch and then modified by Karnaugh, hence it is called Karnaugh map.
- The basis of K – map method is graphical representation of minterms or maxterms in a chart called Karnaugh map (K – map). K – map contains cells.
- Each cell represents one of the 2^n possible product cells that can be formed from n variables.
- Thus, 2-variable K – map has 4 cells, 3-variable K – map has 8 cells and 4-variable K – map has 16 cells.
- Product terms are assigned to the cells of a K – map by labeling each row and each column of the map with a variable, with its complements or with a combination of variables and complements. Fig. 2.4 depicts the 2-variable, 3-variable and 4-variable maps.



(a) 2-variable K – map



(b) 3-variable K – map



(c) 4 - variable K-map

Fig. 2.4

- It is important to note that only one variable changes, when we move from one cell to another along any row or any column.
- Therefore, the third column and the third row in a two-variable K – map have '11' binary representation instead of '10'.
- This peculiar arrangement of K – map has special significance as mentioned below.
- When two inputs change simultaneously then digital circuit output goes in a metastable state.
- Output can swing to either logic '1' or logic '0' state in metastable state.
- This state is to be avoided by prohibiting two inputs from switching simultaneously.
- We know that any logic function can be represented in SOP or POS form. The given Boolean expression can be used to fill entries in the truth table and truth table can be represented on K – map.
- With little practice, it is also possible to fill entries in k – map directly.

2.5.3 Representing SOP Equation on K – map

Example 2.5 : *Plot Boolean expression.*

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

Solution :

- The Boolean expression has four variables, so we use 4-variable K - map.
- Represent each product term by '1' in corresponding cell.
- Note that number of 1's in K-map is equal to the product terms in the given Boolean expression.
- Fill 0's in all other cells.

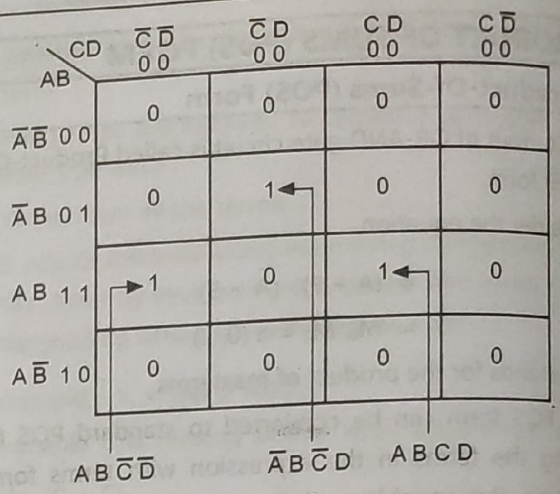


Fig. 2.5

2.5.4 Representing POS Equation on K - map

Example 2.6 : *Plot Boolean expression.*

$$Z = (X + \bar{Y})(\bar{X} + \bar{Y})$$

Solution :

- The Boolean expression has two variables, so we use variable K – map.

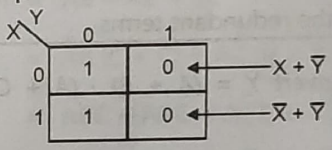


Fig. 2.6

- Represent each sum term by '0' in the corresponding cell.
- Note that number of '0's in K – map is equal to the terms in the given Boolean expression.
- Fill '1's in all other cells.

2.5.5 K – map Reduction Techniques

- In K – map minterms are represented by 1's and maxter are represented by 0's.
- The objective of K – map reduction or simplificat technique is to reduce the number of logic gates.
- Once the logic or Boolean expression is plotted on K – m we use grouping technique to simplify the given Boole expression as follows :

(a) Grouping Two Adjacent Ones (or Pair) :

- Consider a Boolean expression Y = ABC + AB$\bar{C}$.

  It can be seen from the given Boolean expression th we will require two three-input AND gates and one tw input OR gate to implement the logic equation.
- Now, if we plot the equation in a 3-variable K – map.

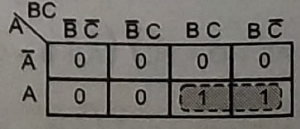
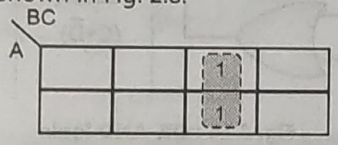


Fig. 2.7 : Grouping on two adjacent ones

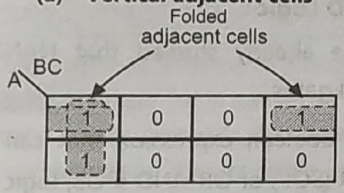
- It can be noticed that when the two adjacent 1's are grouped then only one variable appears in its complemented and un-complemented form i.e. C and $\bar{C}$.

$$Y = ABC + AB\bar{C}$$
$$= AB(C + \bar{C})$$
$$= AB \qquad (\because C + \bar{C} = 1)$$

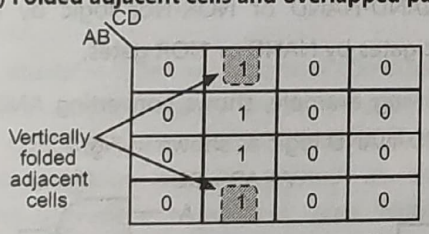
- So, these two terms can be combined together to eliminate the variable C.
- Once this third variable is eliminated then it is possible to use two-input AND gate instead of three-input AND. These adjacent 1's can be also in vertical or any other form as shown in Fig. 2.8.



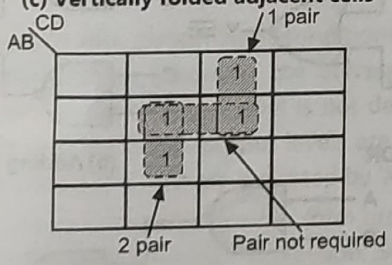
(a) Vertical adjacent cells



(b) Folded adjacent cells and overlapped pairs



(c) Vertically folded adjacent cells

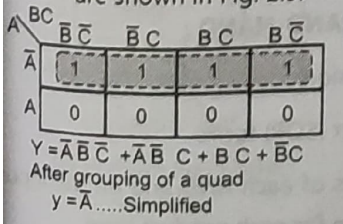


(d) Combining of pair

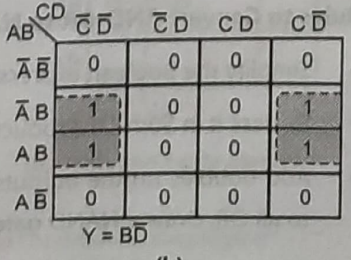
Fig. 2.8 : Various combinations of 1 pairs

) Grouping of Four Adjacent Ones (Quad) :

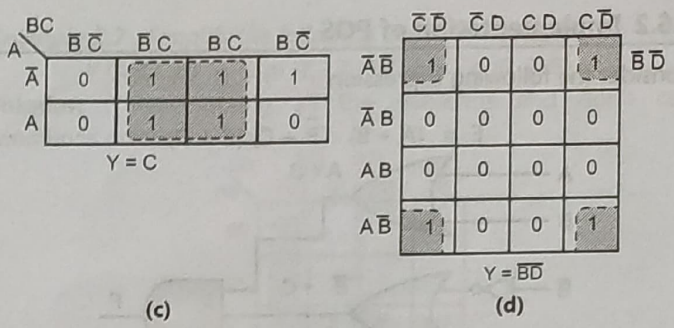
- We can group four adjacent ones to eliminate two variables out of four variables.
- The several ways to form four adjacent ones or quads are shown in Fig. 2.9.



(a)



(b)

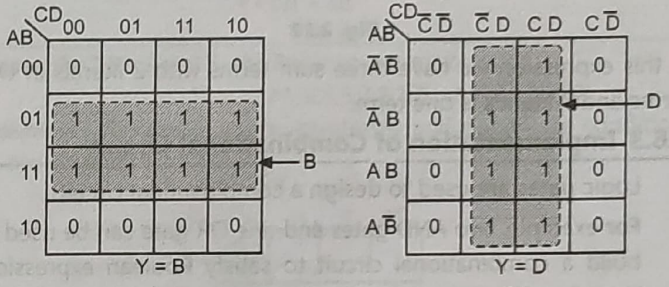


(c) **(d)**

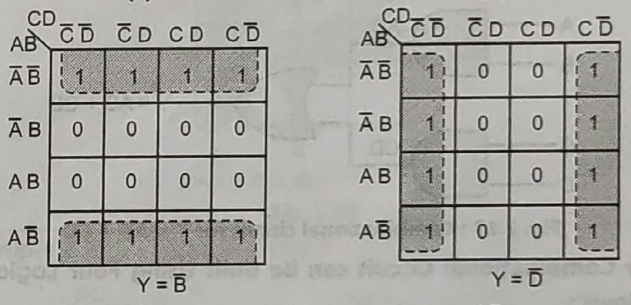
Fig. 2.9

(c) Grouping of Eight Adjacent Ones (Octet)

- We can group four adjacent ones to eliminate three variables out of four variables.



(a) **(b)**



(c) **(d)**

Fig. 2.10

2.6 IMPLEMENTATION OF BOOLEAN FUNCTION USING LOGIC GATE

- The Boolean algebra is used to express the output of any combinational network, such a network can be implemented using logic gates.

2.6.1 Implementation of SOP

Consider the following expression.

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

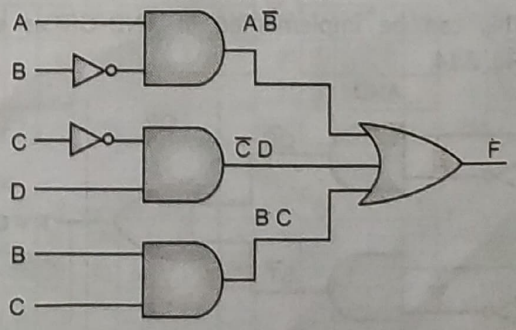


Fig. 2.11

In this expression, we have three product terms with 2 literals in each product term.

2.6.2 Implementation of POS

Consider the following expression.

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

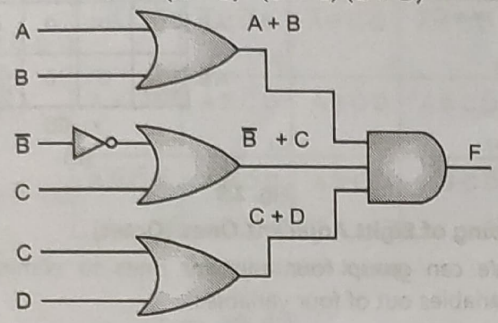


Fig. 2.12

In this expression we have three sum terms with 2 literals in two terms and 3 literals is one term.

2.6.3 Implementation of Combinational Circuit

- Logic gates are used to design a combinational circuit.
- For example, two AND gates and one OR gate can be used to build a combinational circuit to satisfy Boolean expression. Y = AB + CD as shown in Fig. 2.13.

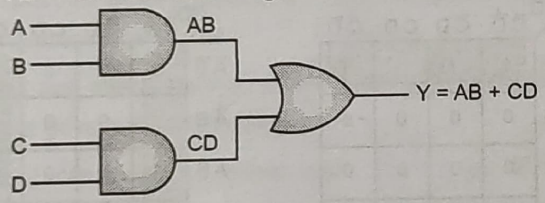


Fig. 2.13 : Combinational circuit for Y = AB + CD

Any Combinational Circuit can be Built Using Four Logics as Follows :

(i) AND-OR (Use of inverters allowed).
(ii) OR-AND (Use of inverters allowed).
(iii) NAND-NAND (Inverters implemented using NAND).
(iv) NOR-NOR (Inverters implemented using NOR).

(i) AND-OR Logic :

- When the given Boolean expression is represented by Sum of Product (SOP) terms then AND-OR logic is used.
- For example, Product terms

$$P = Q\overline{R} + S\overline{T}$$

- This can be implemented in AND-OR as shown in Fig. 2.14.

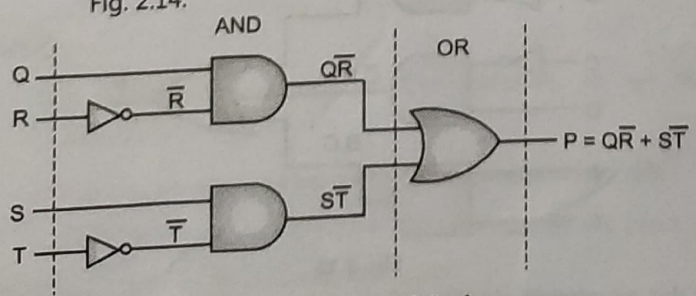


Fig. 2.14 : AND-OR logic

(ii) OR-AND Logic :

- When the Boolean expression is represented by product of sum (POS) terms then OR-AND logic is used.
- For example, Sum terms

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

- This can be implemented in OR-AND as shown in Fig. 2.15.

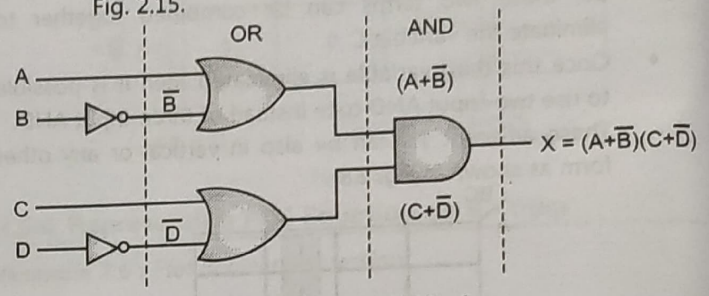


Fig. 2.15 : OR-AND logic

(iii) NAND-NAND Logic :

- We have already studied that NAND and NOR are universal gates.
- So, any Boolean expression that can be expressed in AND-OR (SOP) or OR-AND (POS) logic can be converted to NAND-NAND or NOR-Nor logic by replacing the basic gates by NAND or NOR gates.
- Following example, shows converting AND-OR logic to NAND-NAND logic as shown in Fig. 2.16.

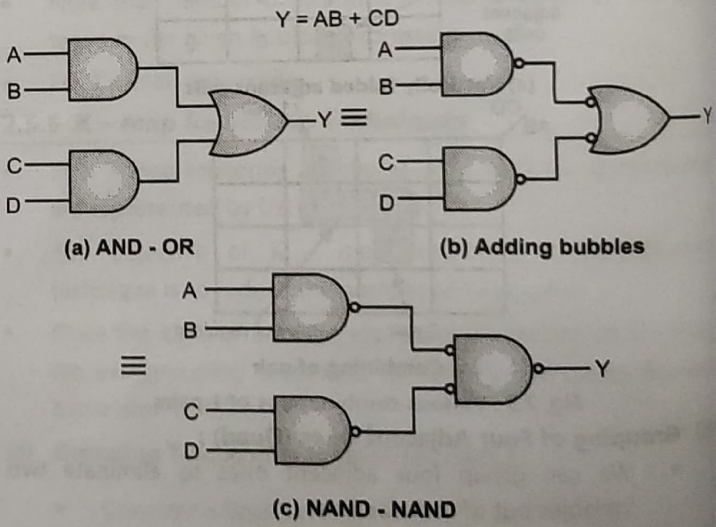


Fig. 2.16 : Converting AND-OR logic to NAND-NAND logic

Rules to Convert AND-OR to NAND-NAND :

- Simplify the Boolean expression.
- Express it in Sum-Of-Product (SOP) form.
- Add bubbles on the outputs of each AND and on the inputs to all OR. Draw a NAND gate for each product term.
- Draw a single NAND gate at the second level.

(iv) NOR-NOR Logic :

- Any OR-AND logic can be converted into NOR-NOR logic. For example, refer Fig. 2.17.

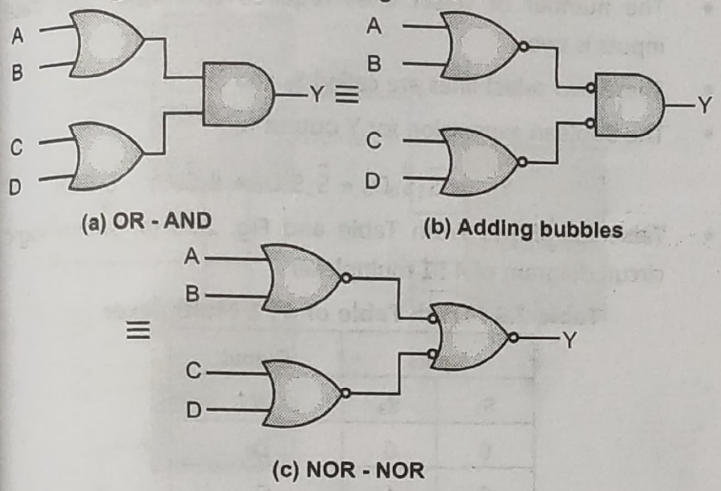
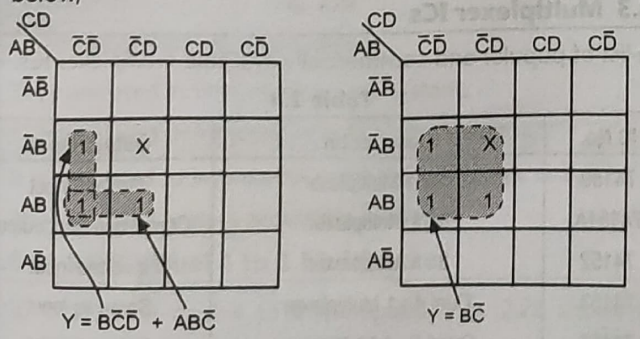


Fig. 2.17 : Converting OR-AND logic to NOR-NOR logic

2.7 DON'T CARE CONDITION

- An incompletely specified function is a Boolean function that only define output values for a subset of its inputs - i.e. a Boolean function whose output is don't care for at least one of its input combinations. Incompletely specified functions often make no guarantees as to the unspecified output whatsoever.
- In many situations when working with combinational circuits, some combinations of inputs should not occur under normal working conditions. For circuits with such combinations, those combinations can be treated as either 0 or 1 depending on whichever yields a more simplified Boolean expression.
- In some logic circuits, certain input conditions never occur or they are not possible. Therefore, the corresponding output never appears, and the output level is not defined. It can be either HIGH or LOW. These output levels are represented as 'Don't Care Conditions' and are indicated by 'X'.
- Don't care conditions can be used to form groups and hence help in simplifying the Boolean expression. See the example below;



(a) Without don't care condition **(b) With don't care condition**

Fig. 2.18 : Use of don't care condition in simplifying the Boolean expression

Example 2.7 : *Simplify the following Boolean expression*

$$Y(A,B,C,D) = \Sigma m(1,3,7,11,15) + d(0,2,8)$$

Solution : Representing all the minterms and don't care conditions on the K – map

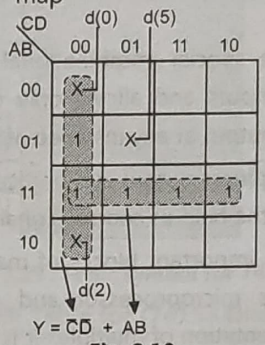


Fig. 2.19

Example 2.8 : *Simplify the Boolean expression*

$$Y = \pi M(4,5,6,7,8,12) \cdot d(0,13,15,14)$$

Solution : Represent all the maxterms and don't care conditions in the K – map

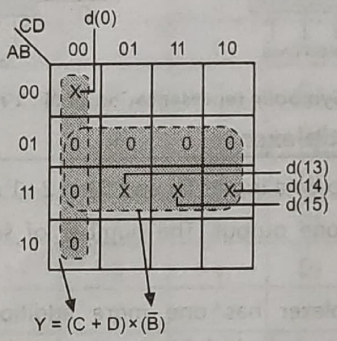


Fig. 2.20

Example 2.9 : *5 - variable example*

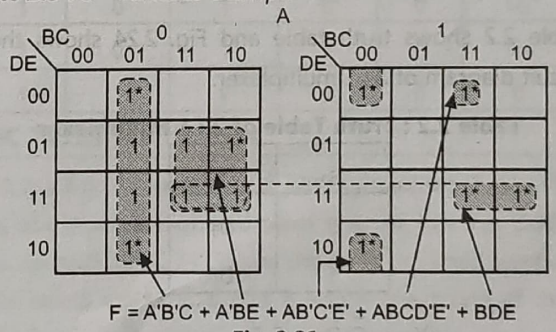


Fig. 2.21

Example 2.10 : *6- variable example*

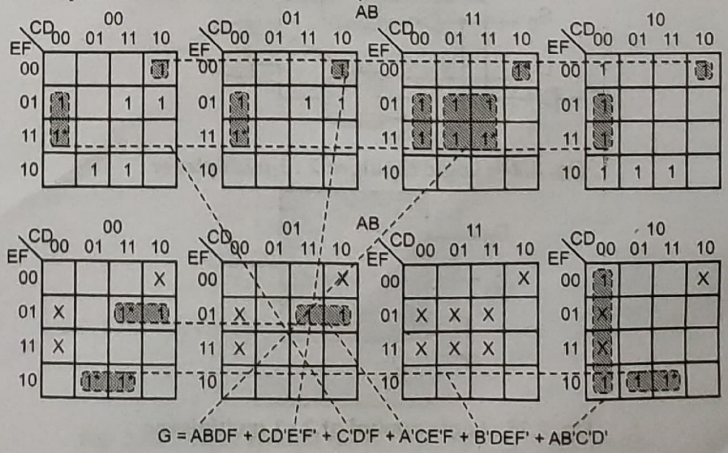


Fig. 2.22

2.8 MULTIPLEXERS

- The literal meaning of the word 'multiplex' is 'many into one'. A multiplexer circuit may have several inputs and only one output.
- Multiplexer is a special combinational logic circuit which accepts many inputs and allows only one of them to get through to the output at any instance of time.
- Therefore, multiplexer output is a particular data input which is selected with the help of control signal called 'Select'.
- Multiplexers are important blocks of many important digital circuits such as microprocessor and microcontroller. The symbolic representation of multiplexer is shown in Fig. 2.23.

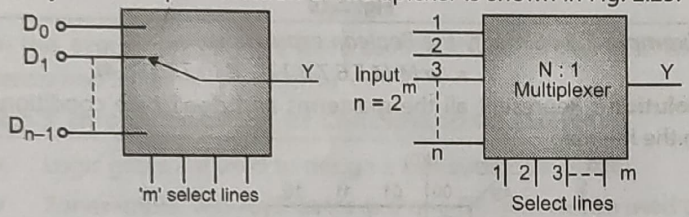


Fig. 2.23 : Symbolic representation of N : 1 multiplexer

2.8.1 2 : 1 Multiplexer

- There are two data inputs D_0 and D_1 in 2 : 1 multiplexer.
- It has only one output. The number of select lines (m) is equal to 1.
- Every multiplexer has one more additional input called 'Enable' or 'Strobe' which is an active low input. This input is always kept at ground potential or logic '0'.
- Table 2.2 shows truth table and Fig. 2.24 shows the logic circuit diagram of 2 : 1 multiplexer.

Table 2.2 : Truth Table of 2 : 1 Multiplexer

S_0	Y
0	D_0
1	D_1

$$Y = \bar{S}_0 D_0 + S_0 D_1$$

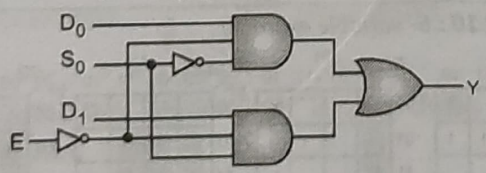


Fig. 2.24 : Logic circuit of 2 : 1 multiplexer

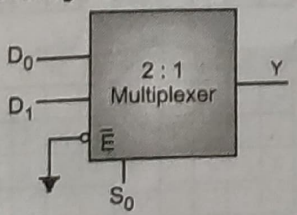


Fig. 2.25 : Logic symbol of 2 : 1 multiplexer

2.8.2 4 : 1 Multiplexer

- 4 : 1 Multiplexer circuit has four data inputs and one output.
- The number of select lines required to control four data inputs is two.
- These two select lines are called S_0 and S_1.
- The Boolean expression for Y output is,

$$Y_1 = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3$$

- Table 2.3 shows Truth Table and Fig. 2.26 (a) shows logic circuit diagram of 4 : 1 multiplexer.

Table 2.3 : Truth Table of 4 : 1 Multiplexer

Inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

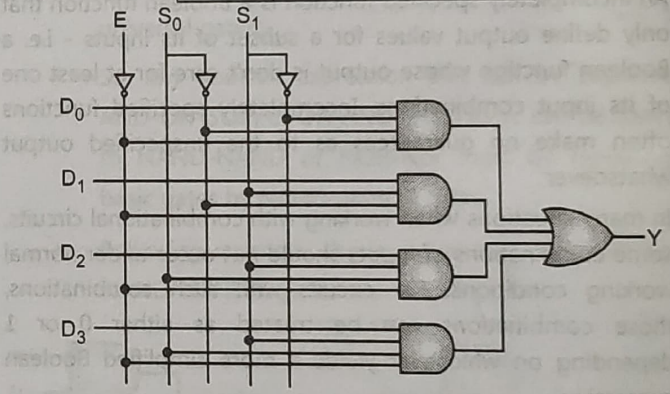


Fig. 2.26 (a) : Logic circuit diagram of 4 : 1 multiplexer

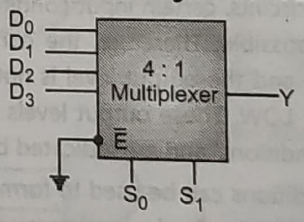


Fig. 2.26 (b) : Logic block diagram of 4 : 1 multiplexer

2.8.3 Multiplexer ICs

The list of popular and commercially available multiplexer ICs.

Table 2.4

IC No.	Description	Output/Input
74150	16 : 1 Multiplexer	Inverted input
74151A	8 : 1 Multiplexer	Complementary output
74152	8 : 1 Multiplexer	Inverted input
74153	Dual 4 : 1 Multiplexer	Same as input
74157	Quad 2 : 1 Multiplexer	Same as input
74158	Quad 2 : 1 Multiplexer	Inverted input
74352	Dual 4 : 1 Multiplexer	Inverted input

2.8.4 Multiplexer Tree

Fig. 2.27 shows design of 8 : 1 multiplexer using two 4 : 1 multiplexers and one 2 : 1 multiplexer.

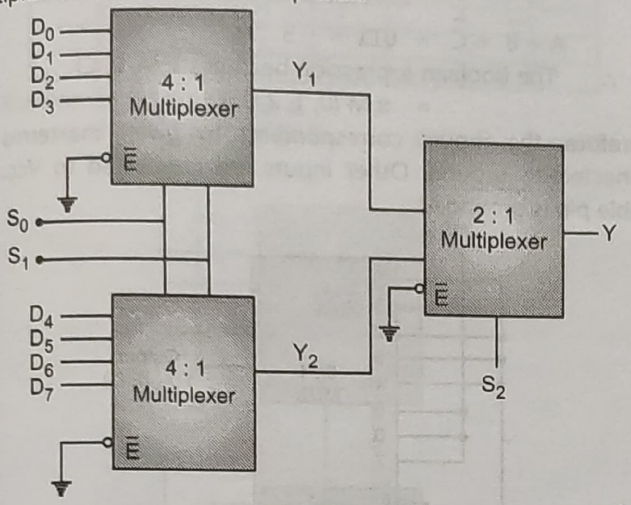


Fig. 2.27 : 8 : 1 Multiplexer

Example 2.11 : *Implement the following expression using a multiplexer*

$$Y = \Sigma m\ (0, 1, 2, 6, 7).$$

Solution : Given : Boolean expression is,

$$Y\ (A, B, C) = \Sigma\ m\ (0, 1, 2, 6, 7)$$

It is a three-variable Boolean function and hence a multiplexer will require three select inputs. The inputs of multiplexer corresponding to given minterms are connected to V_{CC}. Other inputs are grounded.

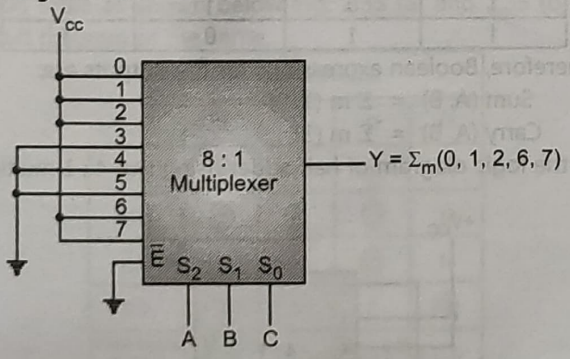


Fig. 2.28

2.8.5 Applications of Multiplexer

- They are used in time multiplexing system.
- They are used in frequency multiplexing systems.
- It is also used to implement combinational logic circuit.
- They are used in data acquisition systems.

2.8.6 IC 74153 Dual 4 to 1 Multiplexer

- IC 74153 is a dual 4 to 1 multiplexer Fig. 2.29 shows the symbol of IC 74153.
- It contains two identical and independent 4 to 1 multiplexers. Each multiplexer has separate enable inputs. The Table 2.5 shows the truth table for IC 74153.

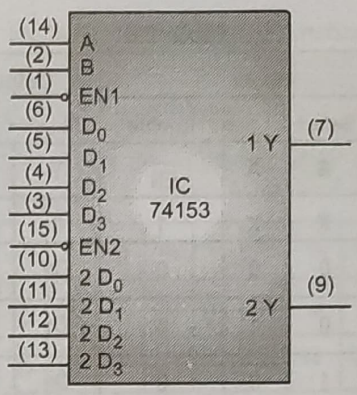


Fig. 2.29 : Logic symbol for 74153

Table 2.5 : Truth Table for (74153 Dual Multiplexer)

Inputs				Outputs	
EN1	EN2	B	A	Y	2Y
0	0	0	0	D_0	D_0
0	0	0	1	D_1	D_1
0	0	1	0	D_2	D_2
0	0	1	1	D_3	D_3
0	1	0	0	D_0	0
0	1	0	1	D_1	0
0	1	1	0	D_2	0
0	1	1	1	D_3	0
1	0	0	0	0	D_0
1	0	0	1	0	D_1
1	0	1	0	0	D_2
1	0	1	1	0	D_3
1	1	x	x	0	0

2.8.7 IC 74151 Multiplexer

IC 74151 is a 8 to 1 multiplexer. It has eight inputs. It provides two outputs, one is active high and other is active low. Fig. 2.30 shows the logic symbol for IC 74151. As shown in the logic symbol there are three select inputs C, B and A which select one of the eight inputs. IC 74151 is provided with active low enable input.

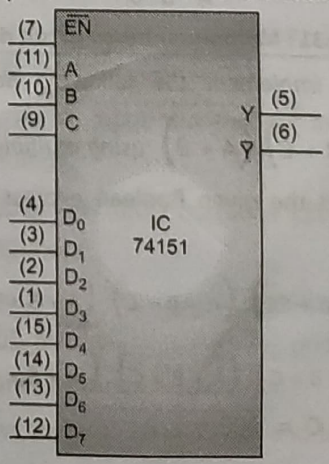


Fig. 2.30 : Logic symbol for IC 74151 8 to 1 multiplexer

Table 2.6

Inputs				Outputs	
Select			Enable		
C	B	A	$\overline{EN}$	Y	$\overline{Y}$
X	X	X	1	0	1
0	0	0	0	D_0	$\overline{D_0}$
0	0	1	0	D_1	$\overline{D_1}$
0	1	0	0	D_2	$\overline{D_2}$
0	1	1	0	D_3	$\overline{D_3}$
1	0	0	0	D_4	$\overline{D_4}$
1	0	1	0	D_5	$\overline{D_5}$
1	1	0	0	D_6	$\overline{D_6}$
1	1	1	0	D_7	$\overline{D_7}$

Example 2.12 : *Implement the following expression using a multiplexer*

$$Y(A, B, C) = \pi M(0, 1, 4, 5)$$

Solution : Given Boolean function is a three variable function expressed in terms of maxterms.

The inputs of multiplexer corresponding to given maxterms are connected to ground and other inputs are connected to Vcc. The enable input is grounded.

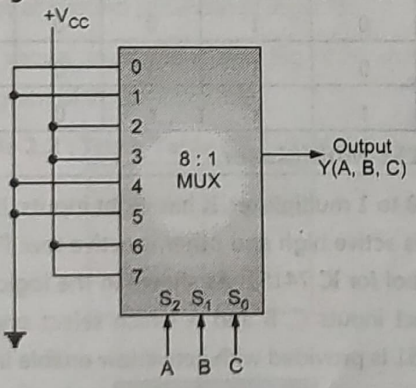


Fig. 2.31 : Multiplexer based circuit design

Example 2.13 : *Implement the following Boolean expression* $Y = (A + B)\left(\overline{A} + B + C\right)\left(A + \overline{B}\right)$ *using multiplexer*

Solution : Convert the given Boolean expression into standard POS form

$$\therefore \quad Y = \left(A + B + C\overline{C}\right)\left(\overline{A} + B + C\right)\left(A + \overline{B} + C\overline{C}\right)$$

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

But $A + B + C = 000 = 0$

$A + B + \overline{C} = 001 = 1$

$\overline{A} + B + C = 100 = 4$

$A + \overline{B} + C = 010 = 2$

$A + \overline{B} + \overline{C} = 011 = 3$

$\therefore$ The Boolean expression becomes Y (A, B, C)

$= \pi M (0, 1, 2, 3, 4)$

Therefore, the inputs corresponding to given maxterms are connected to ground. Other inputs are connected to Vcc. The enable pin is grounded.

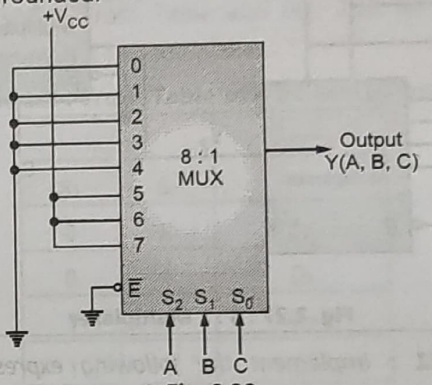


Fig. 2.32

Example 2.14 : *Implement following using multiplexer*

(a) Half-adder

(b) Half-subtractor

Solution :

(a) Half - Adder : The truth table for half adder is given below

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Therefore, Boolean expressions for the outputs are;

Sum (A, B) = Σ m (1, 2)

Carry (A, B) = Σ m (3)

Hence the logic diagram of half adder using two 4 : 1 multiplexer is;

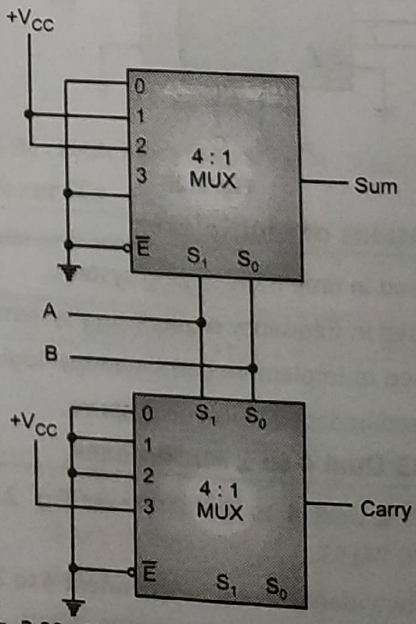


Fig. 2.33 : Half-adder using 4 : 1 multiplexer

(b) Half-Subtractor : The truth table for half subtractor is

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

The Boolean expressions are

$\therefore$ $D(A, B) = \Sigma m(1, 2)$ $C(A, B) = \Sigma m(1)$

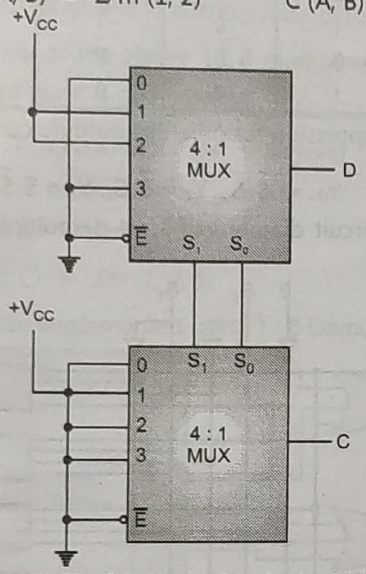


Fig. 2.34 : Half subtractor using 4 : 1 multiplexer

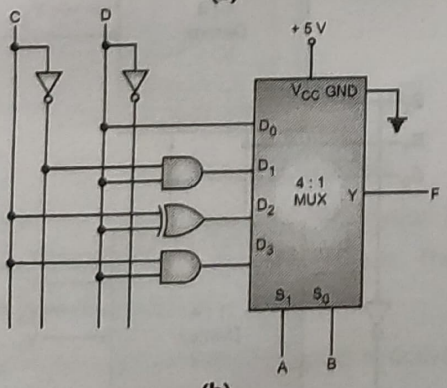
Example 2.15 : *Implement the following function using 4 :1 MUX and logic gates.*

$F(A,B,C,D) = \sum m(0,1,5,9,10,15)$

Solution : function to be implemented is a function of 4 variables and multiplexer to be used is 4 :1 MUX. So we need a multiplexer reduction table as shown below Fig. 2.35 (a) and 2.35 (b) shows the circuit diagram of the same.

	MUX Inputs			
	10	11	12	13
$\bar{C}\bar{D}$	0	4	8	12
$\bar{C}D$	1	6	9	13
$C\bar{D}$	2	0	10	14
CD	3	7	7	15
Final inputs to MUX	$\bar{C}\bar{D} + \bar{C}D = \bar{C}$	$\bar{C}D$	$\bar{C}D + C\bar{D} = CD$	CD

(a)



(b)

Fig. 2.35

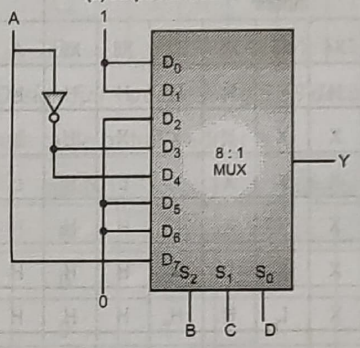
Example 2.16 : *Implement the following function using 8 :1 MUX and logic gates.*

$F(A,B,C,D) = \sum m(0,1,3,4,8,9,15)$

Solution : Fig. 2.36 shows the implementation of given Boolean function using 8 :1 MUX.

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_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

(a) Implementaiton table



(b) Implementation diagram

Fig. 2.36

2.8.8 Encoder (IC 74147)

- An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n (or fewer) input lines and n output lines.

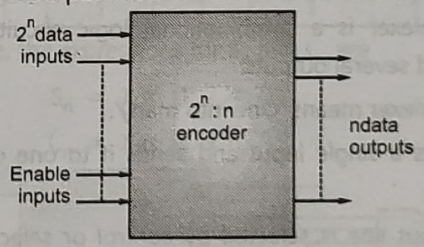


Fig. 2.37

- 74147 is a Decimal to BCD priority encoder. When all inputs are HIGH, all the outputs are HIGH (i.e., 1111) which is the inverse of 0000, the BCD code for 0.
- Since there is no X0 input, the encoder assumes this as the decimal 0 condition.
- When X1 is LOW, the ABCD output is 1110, which is inverse of 0001, the BCD of 1.
- Similarly, when X9 is LOW, the ABCD output is 0110, which is inverse of 1001, the BCD of 9. Here the priority is given to the input with higher subscript value.
- Fig. 2.38 shows the pin out of 74147 and table 2.7 shows the function table of it.

- Note that 1 : 4 demultiplexer has inverted output. Whenever S_2 is logic '0', the IC_1 is selected and IC_2 is disabled. Depending upon the status of S_1 and S_0 lines the data in will be fed to one of the outputs Y_0, Y_1, Y_2 or Y_3.
- To send the output on Y_4 or Y_5 or Y_6 or Y_7 line, the status of pin S_2 is to be made high. Based on the logic status of S_0 and S_1 the data in will be fed to corresponding output line.
- The truth table of the above 1 : 8 multiplexer circuit having initial inverted input is given below.

Table 2.10

Inputs			Selected IC	Outputs							
S_2	S_1	S_0		Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	IC_1	1	1	1	1	1	1	1	0
0	0	1	IC_1	1	1	1	1	1	1	0	1
0	1	0	IC_1	1	1	1	1	1	0	1	1
0	1	1	IC_1	1	1	1	1	0	1	1	1
1	0	0	IC_2	1	1	1	0	1	1	1	1
1	0	1	IC_2	1	1	0	1	1	1	1	1
1	1	0	IC_2	1	0	1	1	1	1	1	1
1	1	1	IC_2	0	1	1	1	1	1	1	1

Example 2.17 : *Implement the following function using demultiplexcer :*

$$f_1 (A, B, C) = \Sigma m (0, 3, 7)$$

$$f_2 (A, B, C) = \Sigma m (1, 2, 5)$$

Solution : It can be implemented using 1 : 8 Demux.

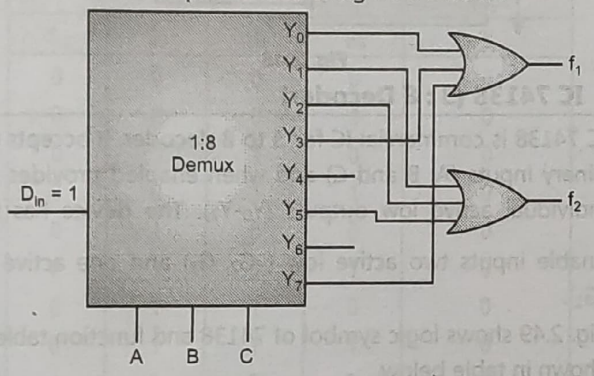


Fig. 2.42

2.9.2 Decoder

- A decoder is multiple input, multiple output logic circuit which converts coded inputs into coded outputs, where the input and output codes are different.
- Fig. 2.43 shows the structure of the decoder circuit.

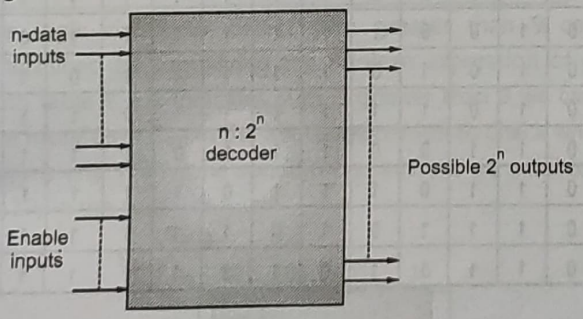


Fig. 2.43 : Structure of decoder

- As shown in Fig. 2.43 the encoded information is presented as n inputs producing 2^n possible outputs. The 2^n output values are from 0 through $2^n - 1$.
- A decoder is provided with enable inputs to activate decoded output based on data inputs. When any of the enable input is unasserted, all outputs of decoder are disabled.

Example 2.18 : *Design full - adder using 3 : 8 decoder*

Solution : The truth table of full - adder is

Inputs			Outputs	
A	B	C_{n-1}	S_n	C_n
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

Therefore, the expressions for S_n and C_n are

$$S_n = \Sigma m (1, 2, 4, 7)$$

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

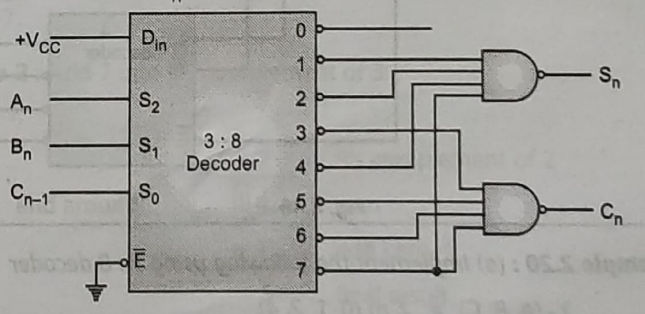


Fig. 2.44 : Full-adder using 3 : 8 decoder

Example 2.19 : *Design 1 : 16 demultiplexer using*

(a) 1 : 8 demultiplexer

(b) 1 : 4 demultiplexer

Solution :

(a) 1 : 8 demultiplexer is also known as 3 : 8 decoder. The MSB input S_3 of 4 - bit input i.e. $S_3 S_2 S_1 S_0$ is used to select one of these two 3 : 8 decoders. S_3 is connected to enable pin.

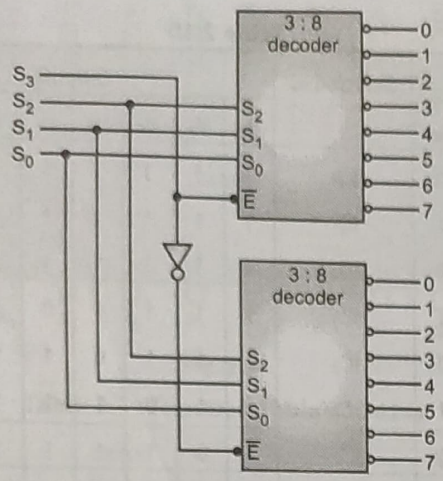


Fig. 2.45 : Demultiplexer using two 1 : 8 demultiplexers

(b)

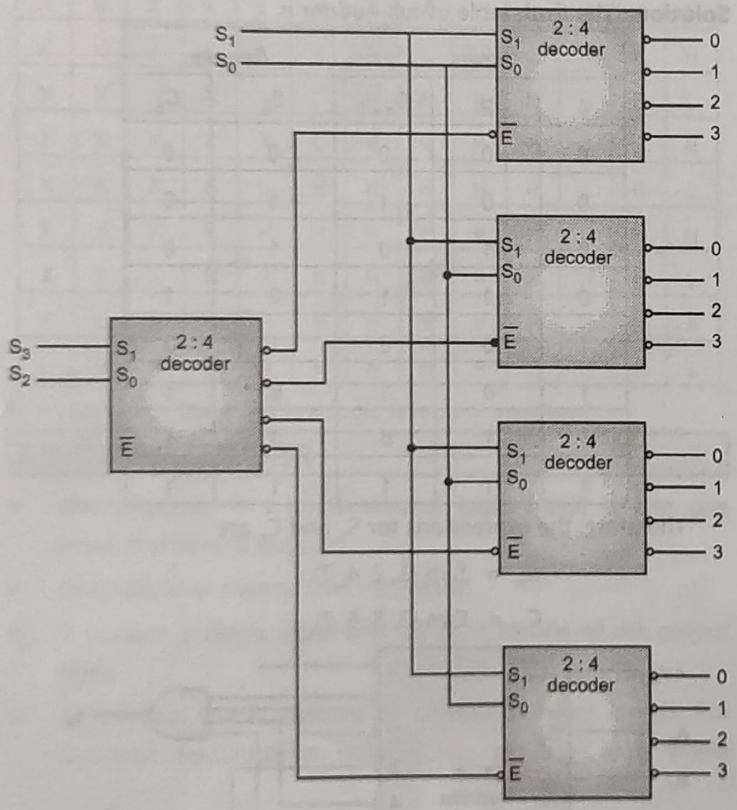


Fig. 2.46

Example 2.20 : *(a) Implement the following using 3 : 8 decoder*

$$Y_0(A, B, C) = \Sigma m(0, 1, 2, 4)$$

$$Y_1(A, B, C) = \Sigma m(1, 3, 5, 7)$$

$$Y_2(A, B, C) = \Sigma m(4, 5, 6, 7)$$

(b) Design some using NAND gates

Solution :

(a) Since the expressions for Y are in terms of minterms, we use OR gates. Because the outputs of decoder are active low, the inputs to OR gates are inverted.

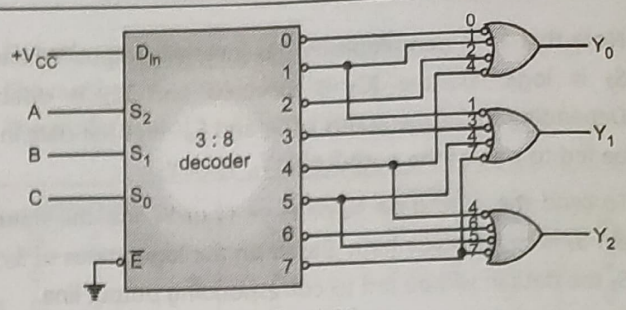


Fig. 2.47

(b) Inverted input OR gate is equivalent to NAND gate.

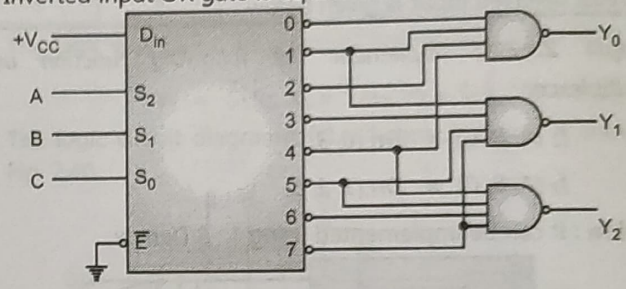


Fig. 2.48

2.9.3 IC 74138 (3 : 8 Decoder)

- IC 74138 is commercial IC for 3 to 8 decoder. It accepts three binary inputs (A, B and C) and when enabled provides eight individual active low outputs (Y_0-Y_7). The device has three enable inputs two active low ($\overline{G}_2$, $\overline{G}_3$) and one active high (G_1).
- Fig. 2.49 shows logic symbol of 74138 and function table also shown in table below.

Table 2.11 : Truth Table

Inputs						Outputs							
G_3	G_2	G_1	C	B	A	$\overline{Y}_7$	$\overline{Y}_6$	$\overline{Y}_5$	$\overline{Y}_4$	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
1	X	X	X	X	X	1	1	1	1	1	01	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	1	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1

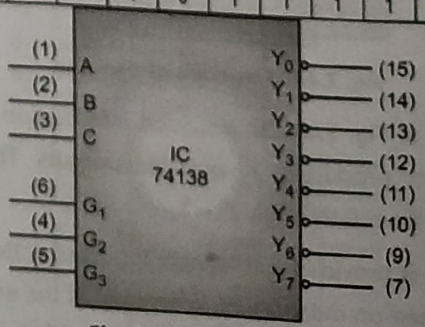


Fig. 2.49 : Logic symbol

2.10 BCD ARITHMETIC

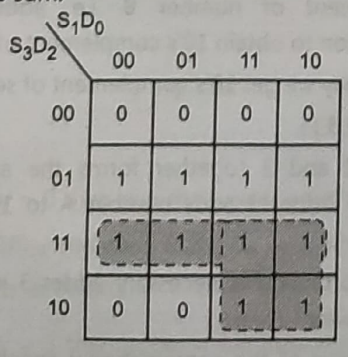
2.10.1 BCD Adder

- The digital systems handle the decimal numbers is in the form of binary coded decimal numbers (BCD) A BCD adder is a circuit that adds two BCD digits and produces a sum digit also in BCD.
- **To Implement BCD Adder we Require**
  - ➢ 4 bit binary adder for initial condition.
  - ➢ logic circuit to detect sum greater than 9 and one more 4-bit adder to add 0110_2 in the sum if sum is greater than 9 or carry is 1.
- BCD adders can be cascaded to add numbers several digits long by connecting the carry out of a stage to the carry in of the next stage.

Table 2.12

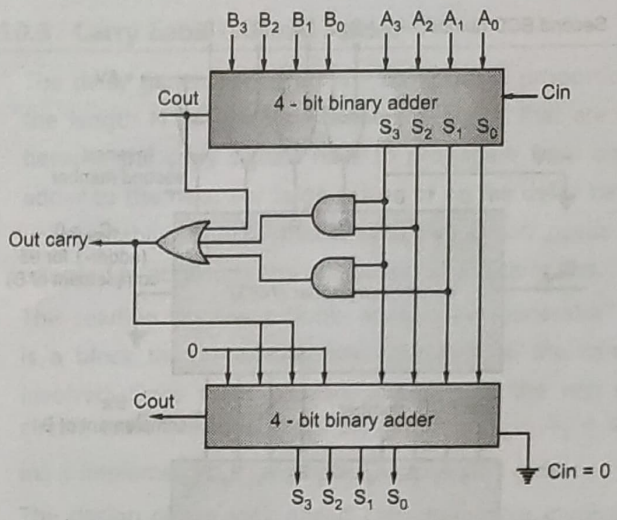
Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

- The logic circuit to detect sum greater than 9 can be determined by simplifying the Boolean expression of given truth table Y = 1 indicates sum is greater than 9 we can put one more term C_{out} in the above expression to check whether carry is one. If any one condition is satisfied, we need to add (0110) in the sum.



$Y = S_3S_2 + S_3S_1$

(a) K – map simplification



(b) Block diagram of BCD adder

Fig. 2.50

2.10.2 BCD Subtractor

- BCD subtraction can be performed using 9's and 10's complement.

BCD Subtraction using 9's Complement :

Then 9's complement is obtained by using following steps :

The Steps for 9's Complement Method :

1. Find the 9's complement of negative number.
2. Add two number using BCD addition.
3. If carry is generated add carry to the result otherwise find 9's complement of the result.

Example 2.21 : *Subtract $(2)_{10}$ from $(7)_{10}$ in BCD.*

Step 1 : Obtain 9's complement of $(2)_{10}$

$$\begin{array}{r} 9 \\ -\ 2 \\ \hline 7 \end{array}$$

Step 2 : Add 7 into 9's complement of 3.

$$\begin{array}{rll} & 7 & \\ & +\ 7 & \text{...9's complement of 2} \\ \text{end around...} & \boxed{1}4 & \text{...sum} \\ & +\ 1 & \text{...add carry} \\ & 5 & \text{...final result} \end{array}$$

BCD Subtractor using 9's Complement Method

Fig. 2.51 shows the circuit diagram of 4-bit BCD subtractor. It consists of four binary parallel adder (IC 7483).

- Adder 1 is to obtains the 9's complement of second number.
- Adder 2 and 3 are used for the normal 4-bit BCD adder with a facility to add 6 for correction.
- Adder 2 adds the first number with the 9's complement of second number B and adder 3 will correct the sum by adding six (0110) if necessary.

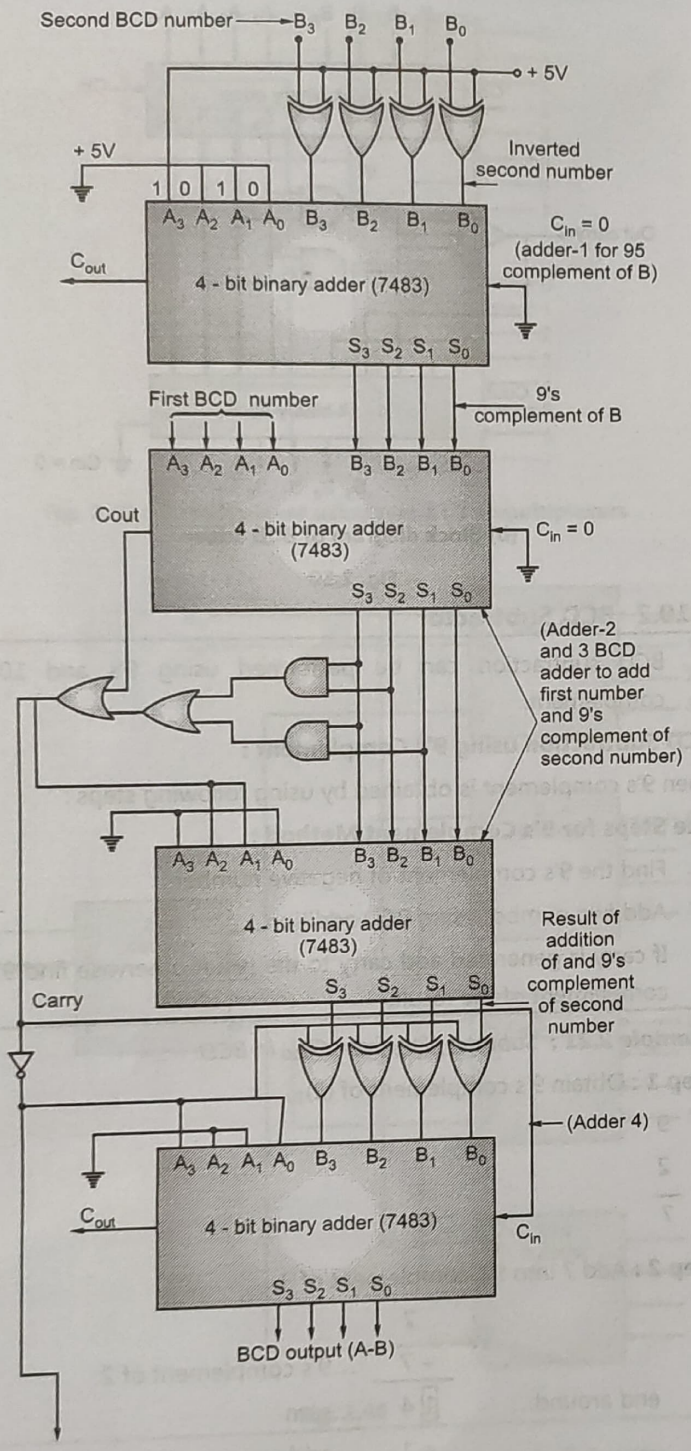


Fig. 2.51

- The output of combinational circuit is used further as a carry. At the output of adder – 3 we get the correct BCD sum of first number and as complement of second number.
- Adder – 4 is used to either add 1 to the output of adder – 3 or take the 9's complement of the output of adder – 3 depending on the status of carry.
- If carry = 1 ; then add 1 to the sum output of adders

If carry = 0; take as complement of sum output of adder 3.

BCD Subtraction using 10's Complement

The 10's complement is obtained by adding 1 to the 9's complement

Steps for 10's Complement BCD Subtraction given Below :

- Obtain the 10's complement of second number
- Add first number with 10's complement of second number.
- Discard the card carry. If carry is 1 then the answer is positive and is in its true form
- If carry is not produced the answer is negative so take 10's complement to get final answer.

Example 2.22 : *Perform the subtraction $(9)_{10} - (4)_{10}$ in BCD using the 10's complement.*

Solution : Step 1 : Obtain the 10's complement of $(4)_{10}$

$$\begin{array}{rl} 9 & \text{...9's complement} \\ -\,4 & \\ \hline 5 & \\ +\,1 & \\ \hline 6 & \text{...10's complement} \end{array}$$

Step 2 : Add $(9)_{10}$ and 10's complement of second number

$$\begin{array}{rll} (9)_{10} \rightarrow & 1\,0\,0\,1 & \\ (6)_{10} \rightarrow & 0\,1\,1\,0 & \\ \hline & 1\,1\,1\,1 & \leftarrow \text{invalid BCD number} \end{array}$$

Step 3 : Add $(6)_{10}$ to invalid BCD number

$$\begin{array}{rll} & 1\,1\,1\,1 & \\ & 0\,1\,1\,0 & \\ \hline \text{Discard final carry} \rightarrow & \boxed{1}\,0\,1\,0\,1 & \leftarrow \text{true BCD form} \end{array}$$

4-bit BCD Subtraction using 10's Complement Method

The circuit consist of 4 adders.

1. **Adder 1 :**
   - Performs the 10's complement of second number. The second numbers inverted using the EX-OR gates and then $C_{in} = 1$ is added to it to obtain 2's complement of second number.
   - $A_3\,A_2\,A_1\,A_0 = 1010$ i.e. $(10)_{10}$ adder 1 adds 1010 and 2's complement of number B. i.e. adder 1 performs subtraction to obtain 10's complement of B.
   - In that way we get 10's complement of second number.
2. **Adder 2 and 3 :**
   - Adder 2 and 3 together forms the addition of BCD number. Adder 2 adds number A to 10's complement of B.
   - If the correction is necessary adder 3 adds 6 (0110) to BCD answer.
   - The output of the combinational circuit is treated as carry and it passed to adder 4.

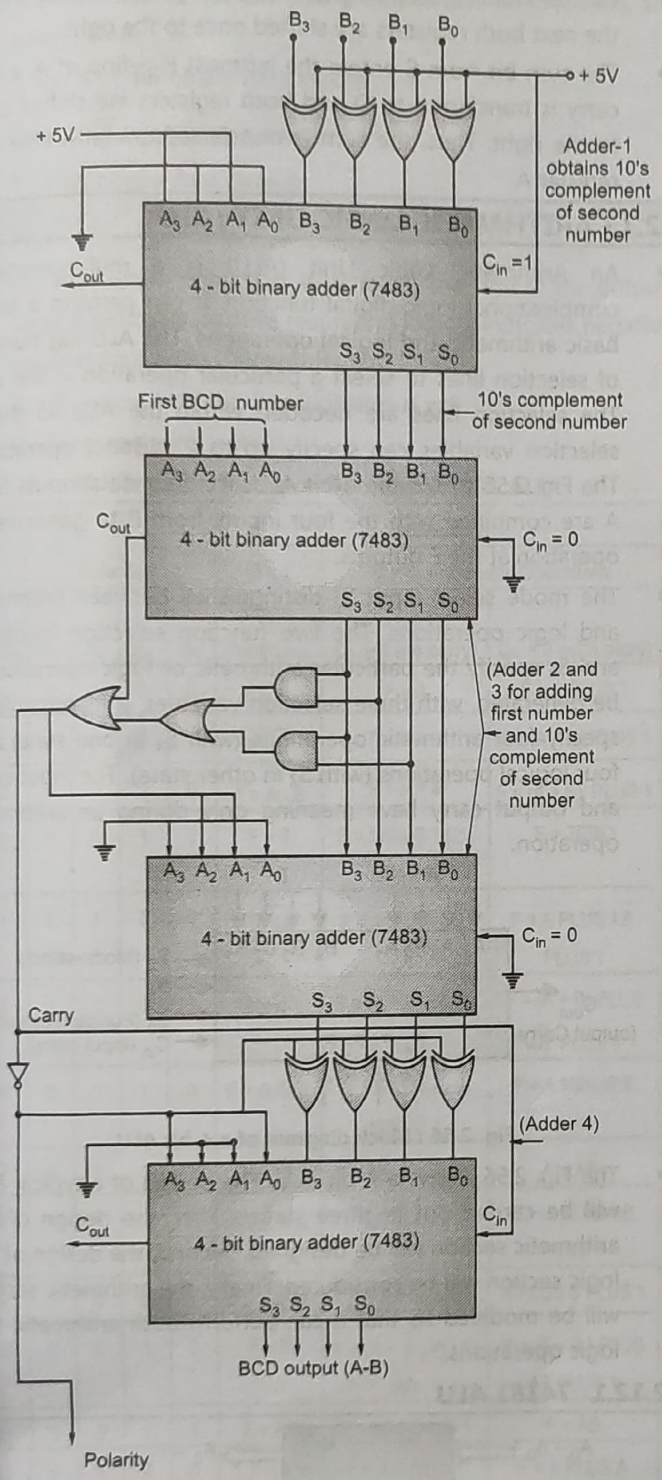


Fig. 2.52

3. **Adder 4 :**

- If carry = 0 then due to inverter used $A_3\ A_2\ A_1\ A_0 = 1010$ i.e. $C(10)_{10}$ and carry input $C_{in} = 1$. Also the EXOR gates will acts as inverter.
- If carry = 1 then adder 4 will pass the adder – 3 output unchanged.

2.10.3 Carry Look - Ahead Adder

- The delay generated by an N - bit adder is proportional to the length N of the two numbers X and Y that are added because the carry signals have to propagate from one full-adder to the next. For large values of N, the delay becomes unacceptably large so that a special solution needs to be adopted to accelerate the calculation of the carry bits.
- This solution involves a "look- ahead carry generator" which is a block that simultaneously calculates all the carry bits involved. Once these bits are available to the rest of the circuit, each individual three-bit addition ($X_1 + Y_1 +$ carry – in_i) is implemented by a simple 3 - input XOR gate.
- The design of the look ahead carry generator involves two Boolean functions named Generate and Propagate. For each input bits pair these functions are defined as :

$$G_i = X_i \cdot Y_i$$

$$P_i = X_i \oplus Y_i$$

- The carry bit $c\_out_{(i)}$ generated when adding two bits X_i and Y_i is '1' if the corresponding function G_i is '1' or if the $C\_out_{(i-1)}$ = '1' and the function P_i = '1' simultaneously. In the first case, the carry bit is activated by the local conditions (the values for X_i and Y_i).
- In the second, the carry bit is received from the less significant elementary addition and is propagated further to the more significant elementary addition. Therefore, the carry out bit corresponding to a pair of bits X_i and Y_i is calculated according to the equation :

$$carry\_out_{(i)} = G_i + P_i \cdot carry\_in_{(i-1)}$$

- For a four - bit adder the carry -outs are calculated as follows

$$carry\_out_0 = G_0 + P_0 \cdot carry\_in_0$$

$$carry\_out_1 = G_1 + P_1 \cdot carry\_out_0$$

$$= G_1 + P_1G_0 + P_1P_0 \cdot carry\_in_0$$

$$carry\_out_2 = G_2 + P_2P_1G_0 + P_2P_1P_0 \cdot carry\_in_0$$

$$carry\_out_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1 \cdot carry\_in_0$$

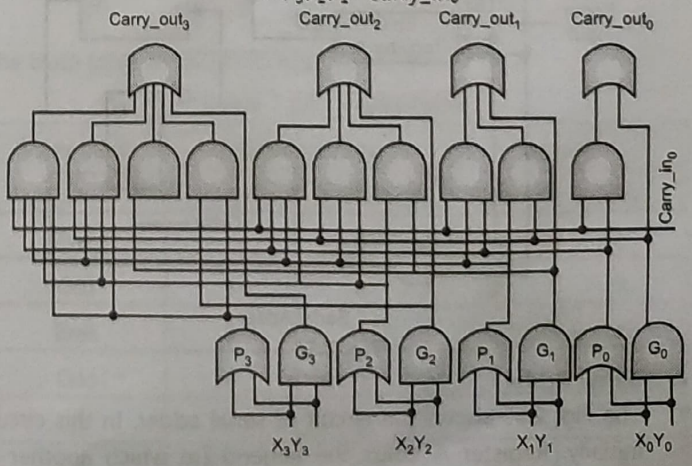


Fig. 2.53 : Carry look-ahead adder – carry output

- The set of equations above are implemented by the circuit below and a complete adder with a look - ahead carry generator is next. The input signals need to propagate through a maximum of 4 logic gate in such an adder as opposed to 8 and 12 logic gates.
- Sums can be calculated from the following equations, where carry_out is taken from the carry calculated in the above circuit.

$$sum\_out_0 = X_0 \oplus Y_0 \oplus carry\_out_0$$
$$sum\_out_1 = X_1 \oplus Y_1 \oplus carry\_out_1$$
$$sum\_out_2 = X_2 \oplus Y_2 \oplus carry\_out_2$$
$$sum\_out_3 = X_3 \oplus Y_3 \oplus carry\_out_3$$

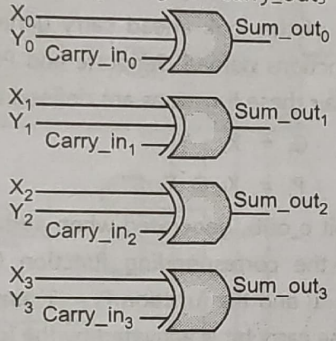


Fig. 2.54 : Carry look-ahead adder – sum output

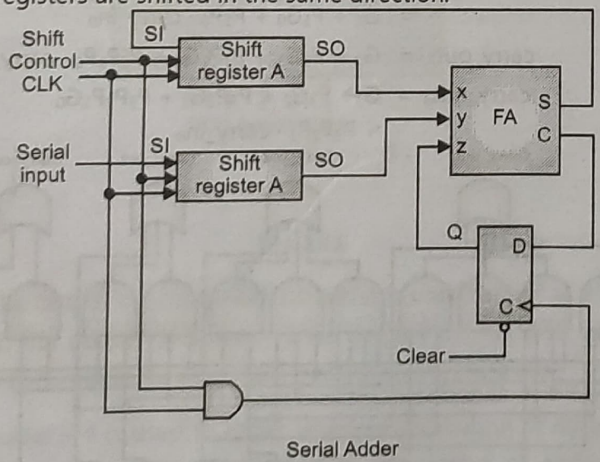
2.11 SERIAL ADDER

In the computer environment, there are 2 types of adders:

1. Parallel Adder
2. Serial Adder

1. Parallel Adder

- Parallel adder is an adder that performs addition concurrently for each bit involved. However, in the serial mode, the registers have a single serial input and a single serial output. The information us transferred one bit at a time while the registers are shifted in the same direction.



Serial Adder

Fig. 2.55

2. Serial Adder

- The Fig. 2.55 shows the circuit of serial adder. In this circuit initially, Register A holds the augend (to which another is added). Register B holds the addend (that is added). Shift control enables both Registers and carry Flip-Flop, so that at the next both registers are shifted once to the right.
- The sum bit from S enters the leftmost Flip-flop of A, a new carry is transferred to Q and both registers are shifted once to the right. Thus, the sum is transferred one at a time into register A.

2.12 ARITHMETIC LOGIC UNIT (ALU)

- An Arithmetic Logic Unit (ALU) is a multioperational combinational-logic digital function. It can perform a set of basic arithmetic and logical operations. The ALU has number of selection lines to select a particular operation in the unit. The selection lines are decoded within the ALU so that k selection variables can specify up to 2^k distinct operations. The Fig. 2.56 shows the 4-bit ALU. The four data inputs from A are combined with the four inputs from B to generate an operation at the F outputs.
- The mode select input S_2 distinguishes between arithmetic and logic operations. The two function selection inputs S_1 and S_0 specify the particular arithmetic or logic operation to be generated. with three selection variables, it its possible to specify four arithmetic operations (with S_2 in one state) and four logical operations (with S_2 in other state). The input carry and output carry have meaning only during an arithmetic operation.

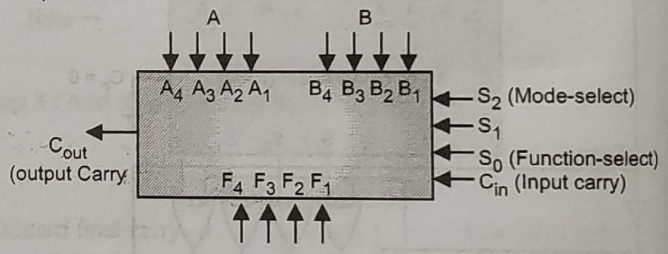


Fig. 2.56 : Block diagram of a 4-bit ALU

- The Fig. 2.56 shows a 4-bit ALU. The design of a typical ALU will be carried out in three stages. First, the design of the arithmetic section will be designed. Second, the design of the logic section will be considered. Finally, the arithmetic section will be modified so that it can perform both arithmetic and logic operations.

2.12.1 74181 ALU

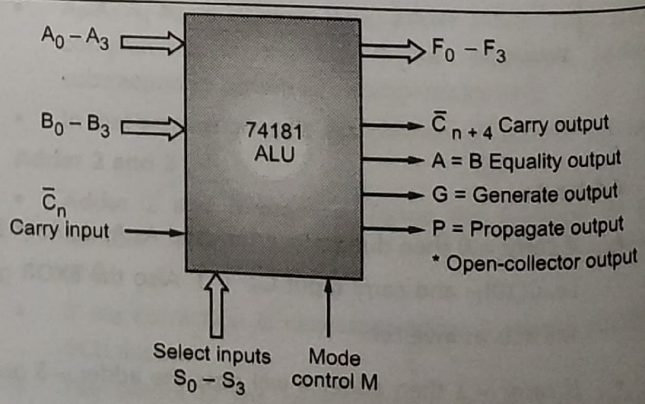


Fig. 2.57 : Block diagram of 74181 ALU

- The functions of various input, output and control lines are given below :

  A and B : 4-bit binary data inputs

  $\bar{C}_n$: Carry input (active-low)

  F : 4-bit binary data output

  $\bar{C}_{n+4}$: Carry output (active-low)

- For subtraction operation, it indicates the sign of the output. Logic 0 indicates positive result and logic 1 indicates negative result expressed in 2's complement form.

  A = B : Logic 1 on this line indicates A = B

  G : Carry generate output

Table 2.13 : Function Table of 74181 ALU

Line	Selection S_3	S_2	S_1	S_0	Active Data: M = 1 Logic Functions	M = 0; Arithmetic Operations: $\bar{C}_n$ = 1 (no carry)	$\bar{C}_n$ = 0 (with carry)
0	0	0	0	0	$F = \bar{A}$	F = A	F = A PLUS 1
1	0	0	0	1	$\overline{A+B}$	F = A + B	F = (A + B) PLUS 1
2	0	0	1	0	$F = \bar{A} \cdot B$	F = A + B	F = (A + $\bar{B}$) PLUS 1
3	0	0	1	1	F = 0	F = MINUS 1 (2's COMPL)	F = ZERO
4	0	1	0	0	$F = \overline{AB}$	F = A PLUS $A\bar{B}$	F = A PLUS $A\bar{B}$ PLUS 1
5	0	1	0	1	$F = \bar{B}$	F = (A + B) PLUS $A\bar{B}$	F = (A + B) PLUS $A\bar{B}$ PLUS 1
6	0	1	1	0	$F = A \oplus B$	F = A MINUS B MINUS 1	F = A MINUS B
7	0	1	1	1	$F = A\bar{B}$	F = $A\bar{B}$ MINUS 1	F = $A\bar{B}$
8	1	0	0	0	$F = \bar{A} + B$	F = A PLUS AB	F = A PLUS AB PLUS 1
9	1	0	0	1	$F = \overline{A \oplus B}$	F = A PLUS B	A PLUS B PLUS 1
10	1	0	1	0	F = B	F = (A + $\bar{B}$) PLUS AB	F = (A + $\bar{B}$) PLUS AB PLUS 1
11	1	0	1	1	F = AB	F = AB MINUS 1	F = AB
12	1	1	0	0	F = 1	F = A PLUS A*	F = A PLUS A PLUS 1
13	1	1	0	1	$F = A + \bar{B}$	F = (A + B) PLUS A	F = (A + B) PLUS A PLUS 1
14	1	1	1	0	F = A + B	F = (A + $\bar{B}$) PLUS A	F = (A + $\bar{B}$) PLUS A PLUS 1
15	1	1	1	1	F = A	F = A MINUS 1	F = A

- P : Carry propagate output
- G and P outputs are used when a number of 74181 circuits are used in cascade along with 74182 Look-ahead
- Carry-generator circuit to make the arithmetic operations faster.

  Select input (S) : Used to select any operation (Table 2.13).

  Mode Control (M) :

  M = 0 Arithmetic operations

  M = 1 Logic Operations

- The 74181 can be cascaded by connecting the carry-out of a stage to carry-in of the following stage.

2.13 PARITY GENERATOR AND CHECKER

- Parity bit is an extra bit included along with the binary information to detect the errors which might occur during the transmission of the binary information. The combinational logic circuits which generate the parity bit (either even or odd) is called as **'Parity Generator'**.
- The total number of ones in the binary information is either even (if even parity generator is used) or odd (if odd parity generator is used). Parity generator circuit is used at the transmission side of the communication channel.
- At the receiving end of the communication channel 'parity checker' circuit is used to check the parity of the received information. Parity checker circuit detects whether the received message is corrupted i.e. whether it has error.
- IC 74180 is a popular nine input parity generator/checker. It can be used as a parity generator or checker Its block representation is shown in Fig. 2.58 below :

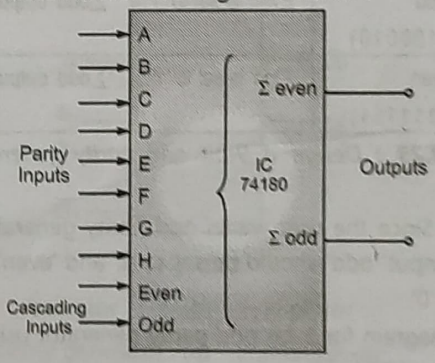


Fig. 2.58

The truth table for IC74180 is given below :

Table 2.14 : Truth Table

Parity Input	Cascading Inputs		Outputs	
	Even	Odd	Even	Odd
Even	1	0	1	0
Odd	1	0	0	1
Even	0	1	0	1
Odd	0	1	1	0
X	1	1	1	0
X	0	0	0	0

The working of IC 74180 can be explained with the help of following input conditions.

(a) User Wants Even Parity Coding in the Message :

- Therefore cascading input, "even" should be '1'
- If the number of 1's in the input message i.e. in parity inputs (A to H) is even then the IC 74180 will generate a '1' output at it's "Σ even" terminal.
- If the number of 1's in the input message (A - H) is odd then the IC 74180 will generate a '1' output at its "Σ even" terminal.

(b) User Wants Odd Parity Coding in the Message :

- Therefore cascading input, "odd" should be '1'.
- If the number of 1's in the input message i.e. in parity inputs (A–H) is even then the IC 74180 will generate a '1' at it's "Σ even" terminal output.
- If the number of 1's in the input message i.e. in parity inputs (A–H) is odd then the IC74180 will generate a '1' output at its "Σ even" terminal.
- A golden rule to understand the working of IC74180 is

Table 2.15

Number of 1's in Parity Inputs (A to H)	Cascading Input	Output
Even e.g. (01101100)	'Even' is set to '1'	'∑even' output will be set '1'
Odd e.g. (00010000)	'Odd' is set to '1'	'∑even' output will be set '1'
Odd e.g. (01100010)	'Even' is set to '1'	'∑odd' output will be set '1'
Even e.g. (11111111)	'Odd' is set to '1'	'∑odd' output will be set '1'

Example 2.23 : *Design a 9-bit odd parity generator using IC 74180.*

Solution : Since the user wants odd parity generator hence the cascading input 'odd' should be set to '1' and 'even' input should be reset to '0'.

The logic diagram for 9-bit odd parity generator using IC74180 is shown in Fig. 2.59 below :

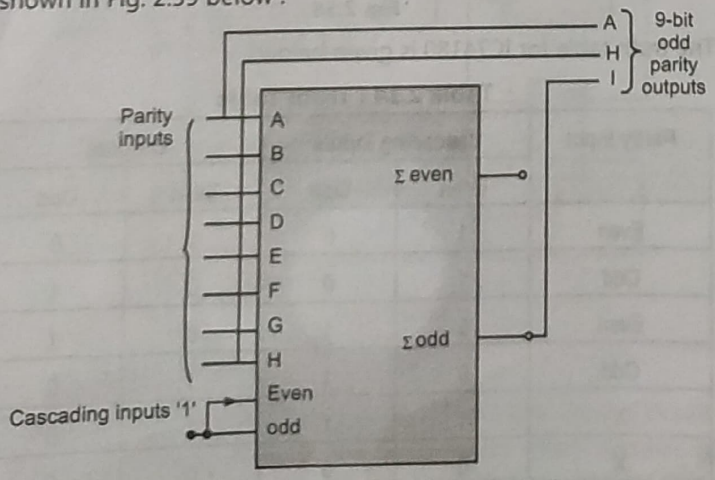


Fig. 2.59 : 9-Bit odd parity generator

If the number of 1's in parity inputs (A – H) is even then the output '∑odd' will be '1'. Thus, total number of 1's in the 9-bit outputs (A – I) will be odd.

If the number of 1's in parity inputs (A – H) is odd then the output '∑odd' will be '0'. Thus, again total number of 1's in the 9-bit outputs (A – I) will be odd.

MULTIPLE CHOICE QUESTIONS (MCQ's)

1. Which are the standard two forms of Boolean expression?
   (a) POS and SOP (b) POS and SOS
   (c) POP and SOP (d) SOS and POP
2. Each individual term in standard SOP form is called ____.
   (a) minterm (b) maxterm
   (c) literal (d) all of the above
3. Each individual term in standard POS form is called ____.
   (a) maxterm (b) minterm
   (c) literal (d) all of the above
4. ____means each term used in a switching equation must contain all the available input variables.
   (a) Standard form
   (b) Canonical form
   (c) Non-standard form
   (d) Minterm form
5. ____form means each term may contain one two or any number of literals.
   (a) Mixed (b) Standard
   (c) Canonical (d) Non-standard
6. The Boolean expression in the SOP form is most suitable for designing logic circuit using only.
   (a) EX-NOR gates (b) NOR gates
   (c) AND gates (d) NAND gates
7. Logical expression in the POS form is most suitable for designing logic circuit using only.
   (a) NAND gates (b) NOR gates
   (c) AND gates (d) OR gates
8. A multiplexer has ____input and ____output.
   (a) one, many (b) many, one
   (c) one, one (d) many, many
9. A multiplexer can be used as a ____.
   (a) logic element (b) counter
   (c) flip flop (d) 7-segment LED driver
10. A demultiplexer can be used to realize a ____.
    (a) counter (b) combinational circuit
    (c) display system (d) shift register

11. A 4-variable logic expression can be realized by using only one ____.
    (a) NAND gate (b) NOR gate
    (c) demultiplexer (d) 16 : 1 multiplexer
12. Which of the following combinational circuit is used to route data from one input to many outputs?
    (a) Demultiplexer (b) Multiplexer
    (c) Encoder (d) None of these
13. A demultiplexer performs reverse operation of ____.
    (a) DEMUX (b) multiplexer
    (c) counter (d) none of these
14. For an n : 1 multiplexer the number of select lines are ____.
    (a) $\log_2 n$ (b) 2^n
    (c) $\log_{10} n$ (d) $\log_2 2n$
15. If multiplexer has m input data, n controls inputs and y outputs, then ____.
    (a) $2^n = m$ (b) $2^m = n$
    (c) $m^n = 2$ (d) $m^2 = n$
16. A multiplex is also known as ____.
    (a) data selector (b) data recorder
    (c) data encoder (d) data decoder
17. A decoder means ____.
    (a) coded information into non-coded form
    (b) HIGHs to LOWs
    (c) LOWs to HIGHs
    (d) non-coded information into coded form
18. A encoder and decoder circuit does not have ____input.
    (a) control (b) data
    (c) select (d) none of these
19. Which of these come under the class of combinational of circuits?
    (1) Read only memories
    (2) D-latch
    (3) Multiplexers
    (4) Circuit as shown

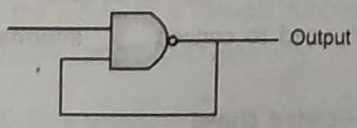


Fig. 2.60

    (a) 1 and 2 (b) 3 and 4
    (c) 1, 2 and 3 (d) 1, 2, 3 and 4
20. How many 3-line to 8-line decoders are required for a 1-of-32 decoder?
    (a) 1 (b) 2
    (c) 4 (d) 8
21. For the device shown here, let all D inputs be low, both S inputs be high, and the $\overline{EN}$ input be LOW. What is the status of the Y output?

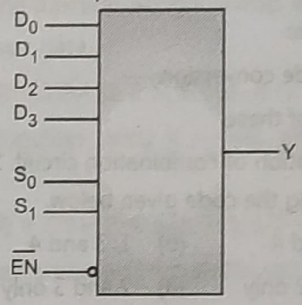


Fig. 2.61

    (a) Low (b) High
    (c) Don't care (d) Cannot be determined
22. How many 1-of-16 decoders are required for decoding a 7-bit binary number?
    (a) 6 (b) 7
    (c) 5 (d) 8
23. The device shown in Fig. 15 is most likely a ____.

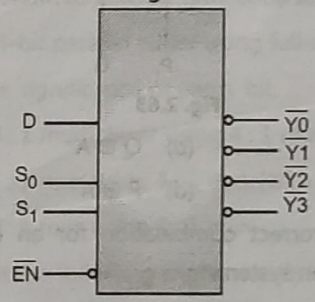


Fig. 2.62

    (a) comparator (b) multiplexer
    (c) parity generator (d) demultiplexer
24. Which of the following is used to convert non-coded information into coded form?
    (a) decoder (b) encoder
    (c) demux (d) MUX
25. The number of 4-line-to-16-line decoders required to make an 8-line to 256-line decoder is ____.
    (a) 64 (b) 17
    (c) 32 (d) 16
26. When two 16-input multiplexers drive a 2-input MUX, what is the result?
    (a) 2-input MUX (b) 32-input MUX
    (c) 16-input MUX (d) 4-input MUX

27. ____is the number of selector lines required in a single input n output demultiplexer?
(a) $\log_2 n$ (b) n
(c) 2^n (d) 2

28. A digital multiplexer can be used for which of the following?
(a) Many-to-one switch
(b) Decoder
(c) For code conversion
(d) None of these

29. Implementation of combination circuit. Select the correct answer using the code given below.
(a) 2, 3 and 4 (b) 1, 3 and 4
(c) 1 and 2 only (d) 2 and 3 only

30. The logic circuit realized by the circuit shown in the Fig. 16 will be ____.

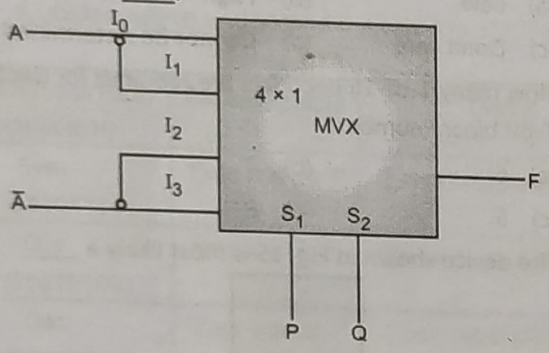


Fig. 2.63

(a) $Q \cdot A$ (b) $Q \oplus A$
(c) $P \cdot A$ (d) $P \oplus A$

31. ____is a correct combination for an odd parity data transmission system.
(a) data = 11011011, parity = 1
(b) data = 00010101, parity = 1
(c) data = 10101111, parity = 0
(d) data = 11011011, parity = 1

32. The 7447A is a BCD-to-7 segment decoder with ripple blanking input and output functions. The purpose of these lines is to ____.
(a) turn off the display for any zero
(b) turn off the display for any non-significant digit
(c) turn off the display for leading or trailing zeros
(d) test the display to assure all segments are proportional

33. In an odd parity system, the data that will produce a parity bit = 1 is ____.
(a) data = 1111000 (b) data = 1100000
(c) data = 1010011 (d) all of the above

34. IC7485 is a ____.
(a) Demultiplexer (b) decoder
(c) comparator (d) multiplexer

35. Two 4-bit comparators are cascaded to form an 8-bit comparator. The cascading inputs of the most significant 4-bit should be connected ____.
(a) to the output from the least significant 4-bit comparator
(b) ground
(c) A = B to logic HIGH, A < b and a > B to a logic LOW
(d) to the cascading inputs of the least significant 4-bit comparator

36. IC74153 is a ____.
(a) parity generator and checker
(b) demultiplexer
(c) comparator
(d) multiplexer

37. The IC 7483 is a ____.
(a) BCD adder (b) demultiplexer
(c) comparator (d) multiplexer

38. The IC 74180 is a ____.
(a) multiplexer (b) comparator
(c) demultiplexer (d) parity generator and checker

39. Parity checker circuit is at ____side.
(a) sender (b) transmitter
(c) receiver (d) both (a) and (c)

40. IC74151 is a ____.
(a) comparator (b) demultiplexer
(c) decoder (d) multiplexer

41. IC74153 is a ____.
(a) dual 4 : 1 mux (b) 8 : 1 mux
(c) 4 : 1 mux (d) 16 : 1 mux

42. The IC74151 is a ____.
(a) 32 : 1 MUX (b) 8 : 1 MUX
(c) 4 : 1 MUX (d) 16 : 1 MUX

43. Multiplexer is used to change ____ data into ____ data.
(a) serial, parallel (b) parallel, parallel
(c) parallel, serial (d) serial, serial

44. Encoder is used to convert ____ Information into ____ form.
(a) non – coded, coded
(b) non coded, non coded
(c) coded, non coded
(d) none of these

45. Which circuit is at transmitter side ?
    (a) Parity generator (b) Comparator
    (c) Multiplexer (d) Parity checker

46. A ___ can be used to realize a combinational circuit.
    (a) demultiplexer (b) multiplexer
    (c) decoder (d) flip-flop

ANSWERS

1. (a)	2. (a)	3. (a)	4. (b)	5. (d)
6. (d)	7. (b)	8. (a)	9. (a)	10. (b)
11. (d)	12. (a)	13. (b)	14. (a)	15. (a)
16. (a)	17. (a)	18. (b)	19. (c)	20. (c)
21.	22. (d)	23. (c)	24. (b)	25. (b)
26. (b)	27. (a)	28. (d)	29. (b)	30. (c)
31. (d)	32. (b)	33. (c)	34. (c)	35. (a)
36. (d)	37. (a)	38. (d)	39. (b)	40. (d)
41. (a)	42. (b)	43. (c)	44. (a)	45. (d)
46. (b)				

EXERCISE

1. Write a short note on combinational circuit.
2. What is the product of sum form?
3. List out the reduction techniques.
4. Simplify following logical expression using K – maps.

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

5. Solve the following using minimization technique.

   $z = f(A, B, C, D) = \Sigma (0, 2, 4, 7, 11, 13, 15)$

6. Simplify the following function

   $f_1 (A, B, C, D) = \Sigma m (0, 3, 5, 6, 9, 10, 12, 15)$

7. What is prime implicant and essential prime implicant ?
8. What is the limitation of K – map?
9. Show implementation of SOP and POS with the example.
10. Minimize $F (A, B, C, D) = \Sigma m (0, 2, 5, 6, 7, 13) + d (8, 10, 15)$ implement using NAND gates.
11. Simplify the following 4 variable function using K – map and represent using NAND gate only.

    $F (A, B, C, D) = (\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D}) + d (\bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + \bar{A}\bar{B}CD + A\bar{B}\bar{C}D + A\bar{B}CD + ABC\bar{D})$

12. Minimize the following equation using K – map and realize it using NAND gates only.

    $Y = \Sigma m (0, 12, 3, 5, 7, 8, 9, 11, 14)$

13. Minimize the function using K – map and implement using only NOR gates.

    $f(\omega, x, y, z) = TTM (1, 3, 5, 7, 13, 15)$

14. Minimize the function using K – map and implement using only one type gate.

    $f (A, B, C, D) = TTM (0, 3, 5, 6, 10) \cdot d (1, 2, 11, 12)$

15. Minimize function using K – map and implement using NOR gate.

    $f (A, B, C, D) = TTM (1, 4, 5, 6, 7, 8, 12) + d (3, 9, 11, 14)$

16. Design a half-adder and full-adder circuits using K – maps.
17. Design a half and full subtractor circuit using K – map.
18. Design 4-bit binary to gray code converter.
19. Design a gray to BCD code converter.
20. Design and implement a 8421 to gray code converter.
21. Design a logic circuit to convert the 8421 BCD to Excess-3 code.
22. Design a 4-bit binary to BCD converter.
23. Design a circuit to convert gray code to binary code.
24. Design a 4-bit parallel adder using full-adders.
25. Explain the significance of parity bit.
26. Design 16 : 1 multiplexer using 4 : 1 multiplexers.
27. Design 14 : 1 mux using 4 : 1 mux (with enable input)
28. Design 28 : 1 mux using 8 : 1 mux.
29. Implement the following expression using 8 : 1 multiplexer.

    $f (A, B, C, D) = \Sigma m (2, 4, 6, 7, 9, 10, 11, 12, 15)$

30. Explain decimal to BCD encoder with logic symbol and truth table.
31. What is encoder ?
32. Implement the following functions using demultiplexer.

    $f_1 (A, B, C) = \Sigma m (1, 5, 7)$, $f_2 (A, B, C) = \Sigma m (3, 6, 7)$

33. Implement two bit comparator using 1 : 16 demultiplexer (active low output). Draw the truth table of two bit comparator.
34. Design and Implement a full adder circuit using 3 : 8 decoder.
35. Implement the logic circuit for full subtractor using decoder.
36. Explain decoder (1 : 8) as a binary to gray code converter.
37. Write short note on combinational circuit.

38. Realize the each of the following functions of four variables using :

(a) 8 : 1 multiplexers (b) 16 : 1 multiplexers

1. $f_1 = \Sigma m (0, 3, 5, 6, 9, 10, 12, 15)$
2. $f_2 = \Sigma m (0, 1, 2, 3, 11, 12, 14, 15)$

38. Realise the logic function in SOP form using Quine-McCluskey method :

$F (A, B, C, D) = \pi M (2, 7, 8, 9, 10, 12, 15)$

39. A staircase light is controlled by two switches, one at top of the stairs and another at the bottom of stairs.

(a) Make the truth table of the system

(b) Write the logic equation in SOP form

(c) Realise the circuit using AND-OR gates

(d) Realise the circuit using NAND gates only.

40. Design a 3-bit GRAY? to its 2's complement converter.

✠ ✠ ✠

Unit III

SEQUENTIAL CIRCUITS AND SYSTEMS

3.1 INTRODUCTION

- In this unit we will study comparison between combinational circuit and sequential circuit. We will also study design of latch and flip-flop.
- Study of flip-flops with asynchronous and synchronous preset and clear, master slave configuration.
- We will study conversion from one type to another type of flip-flop, we will also study 7473, 7474, 7476 flip-flop ICs.
- We will cover study of flip-flop applications and synchronous as well as asynchronous counters.
- Study of ICs 7490, 74191 and their applications to implement mod counter.

3.2 SEQUENTIAL CIRCUITS

- In many applications, it is required to generate digital outputs in accordance with the sequence in which the input signals are applied.
- Thus, these applications require that the outputs to be generated are not only dependent on present input conditions. The outputs also depend upon the past history of these inputs.
- The past history is provided by storing it in memory elements and providing feedback from the output back to the input. such circuits are known as sequential circuits.
- Block diagram of a sequential circuit is shown in Fig. 3.1.

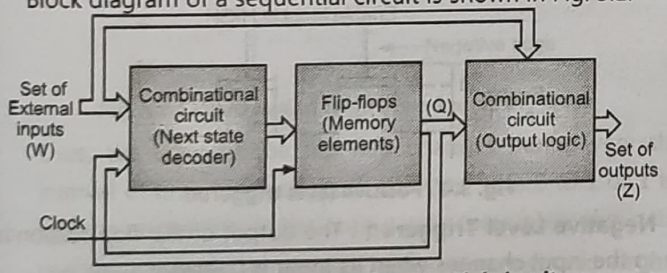


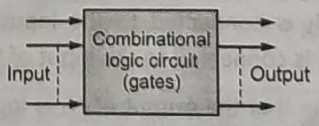
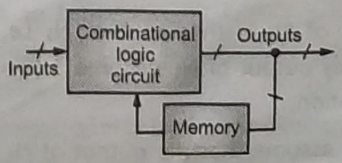
Fig. 3.1 : Block diagram of sequential circuit

- As shown in above Fig. 3.1, the circuit accepts a set of external inputs W and produces a set of outputs Z.
- The values of the outputs of the flip–flops are known as the state Q of the circuit.
- Upon application of clock pulse, the flip flop outputs change their state (circuit goes to next state).
- As shown in Fig. 3.1 the next state of the circuit is decided by the combinational circuit (next state decoder) that provides the inputs to the flip–flops.
- The combinational logic that provides the inputs to the flip–flops, derives its input from two sources :
  1. Set of external inputs (W) and
  2. The present state Q of the circuit (outputs of the flip–flops).
- Thus, changes in state depend upon the external inputs as well the present state of the circuit.
- As shown in Fig. 3.1, the outputs of the sequential circuit are generated by another combinational circuit (output logic). The outputs are generated from the present state Q of the circuit and the set of primary inputs W.
- Sequential circuits are also known as finite state machines (FSMs) as the functional behaviour of these circuits can be represented using finite number of states.

Sequential Circuits are Broadly Classified into Two Categories :

1. Moore type sequential circuits and
2. Mealy type sequential circuits.

3.3 COMPARISON BETWEEN COMBINATIONAL AND SEQUENTIAL CIRCUIT

Sr. No.	Combinational Circuit	Sequential Circuit
1.	The outputs depend on the combination of inputs.	The outputs depend on the past history of inputs as well as the present input states.
2.	Memory is not required.	Memory is required to store previous states of the inputs.
3.	The delay between outputs and inputs is less, so the combinational circuit is faster.	Sequential circuit is slower due to propagational delay of additional memory element.
4.	Combinational circuits are concurrent in nature.	Sequential circuit is not entirely concurrent.
5.	Easier to design.	Complex to design. Timings can be critical.
6.	**Block Diagram :**  **Fig. 3.2 (a) : Combinational circuit** **Block Diagram :**  **Fig. 3.2 (b) : Sequential circuit**	

3.3.1 Sequential Circuit Types

(A) Synchronous Circuit :

- The change in inputs can affect memory element upon the activation of clock signal.
- Memory elements are clocked flip-flops.
- The maximum operational speed of synchronous circuit is governed by the clock speed, which in turn, is decided by the propagation delays of the logic gates.

(B) Asynchronous Circuit :

- The change in inputs can occur at any instant of time.
- Memory elements are unclocked flip-flops or time delay elements.
- Asynchronous circuits can operate faster than synchronous circuits because the clock is absent.

3.3.2 One Bit Memory Cell (Basic Bistable Elements)

- One bit memory cell, as the name suggests can store 'one' bit (logic 0 or logic 1) information.
- It can be built using NAND or NOR gates.
- A one bit memory cell using NAND gates is as shown in Fig. 3.3.

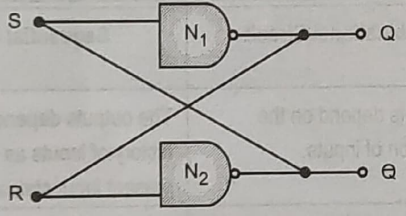


Fig. 3.3 : One bit memory cell

- The above circuit is also known as 'S-R' (Set-Reset) latch.
- This circuit has two stable states : '1 state' (Output Q = 1) and '0 state' (output Q = 0).
- The '1 state' is also called as 'set state' and the '0 state' is known as 'reset state'.
- The digital information gets locked or latched in this circuit. Therefore, it is known as S-R i.e. set-reset latch.

Operation of the Circuit :

- Two NAND gates (N_1 and N_2) are used as inverters. The output of N_1 is connected to the input of N_2 (R) and the output of N_2 is connected to the input of N_1 (S).
- Let us assume that the output of N_1 is logic 1 (Q = 1). This is the input of N_2 i.e. R=1. Therefore, the output of N_2 becomes logic 0 ($\bar{Q} = 0$).
- The output of N_2 is the input of N_1 i.e. S become 0 and consequently output of N_1 become 1 (Q=1), which confirms our assumption.
- Let us now assume that the output of N_1 is logic 0 (Q = 0). This is the input of N_2 i.e. R = 0. Therefore the output of N_2 becomes logic 1 ($\bar{Q} = 1$).
- The output of N_2 is the input of N_1 i.e. S becomes 1 and consequently output of N_1 becomes 0 (Q = 0) which confirms our assumption.

Drawbacks :

- In the above circuit there is no way to enter the desired digital information. When the power is turned on, the circuit switches to one of the stable states i.e. 1 state or 0 state and it is not possible to predict it.
- To overcome this drawback, a modified circuit with 2 input NAND gates and two additional inverters are used. The desired digital information can be entered in this circuit.

3.4 LATCH V/s FLIP-FLOP

- The main difference between latches and flip-flops is the method used for changing their state.
- Latches are controlled by enable signal, and they are level triggered, either positive level triggered or negative level triggered.
- Flip-flops are pulse or clock edge triggered instead of level triggered.

3.5 LEVEL TRIGGERED AND EDGE TRIGGERED

(A) Level Triggered : In level triggering the output state changes according to input (s) when active level (i.e. positive or negative) is maintained at the enable input.

Two Types of Level Triggered are :

1. **Positive Level Triggered :** The output of flip flop responds to the input changes when its enable input is 1 (high).

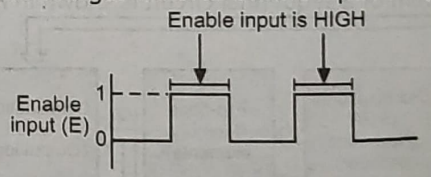


Fig. 3.4 : Positive level triggered

2. **Negative Level Triggered :** The output of flip-flop responds to the input changes when its input is 0 (low).

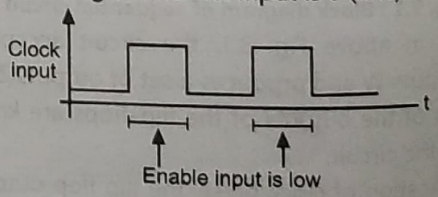


Fig. 3.5 : Negative level triggered

- S-R flip flop and JK flip flop are known as level triggered flip flops as their output changes according to applied inputs as long as clock is present.
- As these flip flops respond when CLK=1, they are further called as positive level triggered flip flops.
- We know that level triggered JK flip flop has the drawback of race around condition. And to overcome that drawback we use master slave JK flip flop which is called as pulse triggered flip flop.

- In a pulse triggered flip flop like MS JK flip flop, output changes according to applied inputs, when a pulse is applied at the clock input. The state of this flip flop changes at the negative transition of the clock.
- Thus, in MS JK flip flop the race around condition is eliminated as the fed back output is blocked at the master when the CLK = 0.
- But in certain systems there is a possibility that the inputs of flip flop may change during the presence of the clock pulse. This causes uncertainty in the output of flip flop. This uncertainty can be eliminated by using edge triggered flip flops.

(B) Edge Triggered : In edge triggered flip flops output changes according to applied inputs only at the positive or negative edge of the clock pulse.

- **Based on the Type of Edge There are Two Types of Edge Triggered Flip Flops :**
  1. Positive edge triggered and
  2. Negative edge triggered.
- In case of positive edge triggered flip flop output changes only when the clock pulse changes from 0 to 1. While in case of negative edge triggered flip flop output responds only when the clock pulse changes from 1 to 0. The positive and negative edge of the clock is shown in Fig. 3.6.

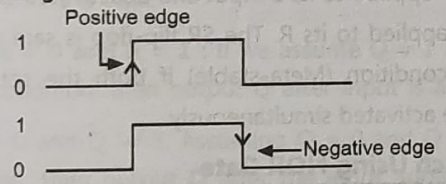
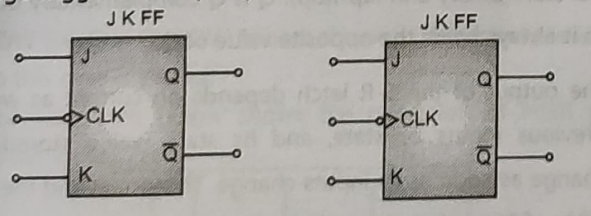


Fig. 3.6 : Positive and negative edge of the clock

- Thus, the state of the flip flop changes during very short interval of time in which clock changes from 0 to 1 or 1 to 0 and the uncertainty in the output gets completely eliminated.
- The logic symbol of positive edge triggered and negative edge triggered JK flip flop is shown in Fig. 3.7.



(i) Positive edge triggered **(ii) Negative edge triggered**

Fig. 3.7 : Edge triggered JK flip flop

- Note that the logic symbol of negative edge triggered JK flip flop is same as that of MS JK flip flop without preset and clear inputs.
- Also, in case of positive edge triggered JK flip flop bubble is absent.

3.6 LATCH

- This circuit is with 2 input NAND gates N_1 and N_2; two additional inverters N_3 and N_4 is as shown in Fig. 3.8.

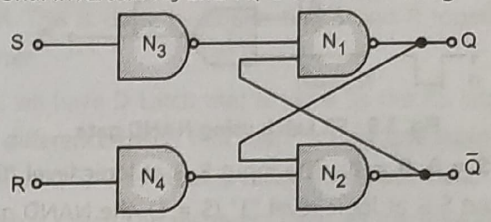


Fig. 3.8 : Memory cell with provision for entering data

- If S = R = 0, the circuit will behave exactly as the previous circuit shown in Fig. 3.7.
- If S = 1 and R = 0 then the output of N_3 will be 0 and the output of N_4 will be 1. As one of the inputs of N_1 is 0, its output will be certainly 1 (Q = 1).
- When Q become 1, both inputs of N_2 become 1 causing its output to go low ($\bar{Q}$ = 0). This is known as 1 state or set state of the circuit, which is achieved with the input pattern S = 1 and R = 0.
- If S = 0 and R = 1, then the output of N_4 will be 0 and the output of N_3 will be 1. As one of the inputs of N_2 becomes 0, its output will be certainly 1 ($\bar{Q}$ = 1).
- When $\bar{Q}$ becomes 1, both inputs of N_1 become 1 causing its output to go low (Q = 0). This is known as 0 state or reset state of the circuit which is achieved with the input pattern S = 0 and R = 1.
- In this way, user can enter desired information in the one bit memory cell.
- Until now we have seen that the outputs Q and $\bar{Q}$ are always complementary. If we apply the input S = 1 and R = 1, then the output of N_3 and N_4 become 0. This makes one input of both N_1 and N_2 as 0, which in turn cause both outputs Q and $\bar{Q}$ to become 1.
- Both Q and $\bar{Q}$ getting same state is not allowed and therefore the condition of inputs S = R = 1 is prohibited.

3.6.1 SR Latch Using NAND Gate

- The simplest way to make any basic single bit set-reset SR latch is to connect together a pair of cross-coupled 2-input NAND gates as shown, to form a set-reset bistable also known as an active low SR NAND gate latch, so that there is feedback from each output to one of the other NAND gate inputs.
- This device consists of two inputs, one called the Set, S and the other called the Reset, R with two corresponding outputs Q and its inverse or complement $\bar{Q}$ (not-Q) as shown below in Fig. 3.9.

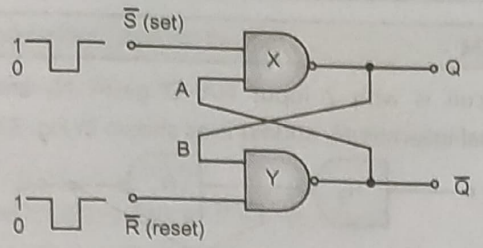


Fig. 3.9 : SR Latch using NAND gate

- **When S = 0, R = 0 :** The input R is at logic level "0" (R = 0) and input S is at logic level "1" (S = 1), the NAND gate Y has at least one of its inputs at logic "0" therefore, its output $\bar{Q}$ must be at a logic level "1" (NAND Gate principles). Output $\bar{Q}$ is also fed back to input "A" and so both inputs to NAND gate X are at logic level "1", and therefore its output Q must be at logic level "0".
- **When S = 1, R = 1 :** The NAND gate Y inputs are now R = "1" and B = "0". Since one of its inputs is still at logic level "0" the output at $\bar{Q}$ still remains HIGH at logic level "1" and there is no change of state. Therefore, the flip-flop circuit is said to be "Latched" or "Set" with $\bar{Q}$ = "1" and Q = "0".
- **When S = 0, R = 1 :** $\bar{Q}$ is at logic level "0", ($\bar{Q}$ = "0") its inverse output at Q is at logic level "1", ($\bar{Q}$ = "1"), and is given by R = "1" and S = "0". As gate X has one of its inputs at logic "0" its output $\bar{Q}$ must equal logic level "1" (again NAND gate principles). Output $\bar{Q}$ is fed back to input "B", so both inputs to NAND gate Y are at logic "1", therefore, $\bar{Q}$ = "0".
- **When S = 1, R = 1 :** If the set input, S now changes state to logic "1" with input R remaining at logic "1", output $\bar{Q}$ still remains LOW at logic level "0" and there is no change of state. Therefore, the flip-flop circuits "Reset" state has also been latched and we can define this "set/reset" action in the following truth Table 3.1.

Table 3.1 : Truth Table of Latch Using NAND Gate

State	S	R	Q	$\bar{Q}$	Description
Set	1	0	0	1	Set Q » 1
	1	1	0	1	no change
Reset	0	1	1	0	Reset Q » 0
	1	1	1	0	no change
Invalid	0	0	1	1	Invalid Condition

- It can be seen that when both inputs S = "1" and R = "1" the outputs $\bar{Q}$ and $\bar{Q}$ can be at either logic level "1" or "0", depending upon the state of the inputs S or R before this input condition existed. Therefore, the condition of S = R = "1" does not change the state of the outputs $\bar{Q}$ and $\bar{Q}$.
- The input state of S=0 and R=0 is an undesirable or invalid condition and must be avoided. The condition of S = R = "0" causes both outputs $\bar{Q}$ and $\bar{Q}$ to be high together at logic level "1" when we would normally want $\bar{Q}$ to be the inverse of $\bar{Q}$. The result is that the flip-flop loses control of $\bar{Q}$ and $\bar{Q}$, and if the two inputs are now switched "high" again after this condition to logic "1", the flip-flop becomes unstable and switches to an unknown data state based upon the unbalance as shown in the following switching diagram.

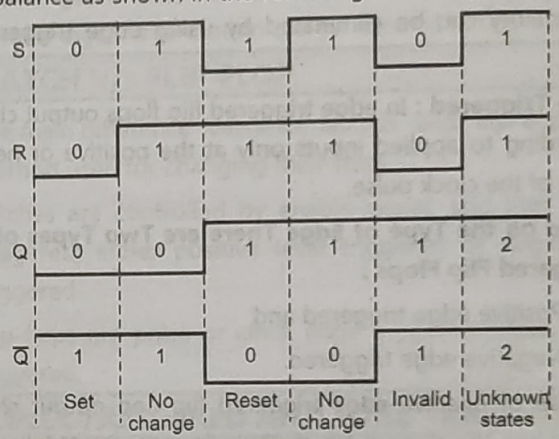


Fig. 3.10 : S-R latch using NAND gate switching diagram

- Then, a bistable SR flip-flop or SR latch is activated or set by a logic "1" applied to its S input and deactivated or reset by a logic "1" applied to its R. The SR flip-flop is said to be in an "invalid" condition (Meta-stable) if both the set and reset inputs are activated simultaneously.

3.6.2 SR Latch Using NOR Gate

- RS latch has two inputs, S and R. S is called set and R is called reset.
- The S input is used to produce HIGH on Q (i.e. store binary 1 in flip-flop). The R input is used to produce low on Q (i.e. store binary 0 in flip-flop). $\bar{Q}$ is Q complementary output, so it always holds the opposite value of Q.
- The output of the S-R latch depends on current as well as previous inputs or state, and its state (value stored) can change as soon as its inputs change. The circuit and the truth table of RS latch is shown below.

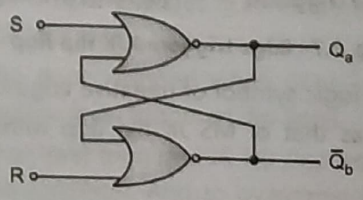


Fig. 3.11

Table 3.2 : Truth Table SR Latch Using NOR Gate

S	R	Q	Q+
0	0	0	0
0	0	1	1
0	1	X	0
1	0	X	1
1	1	X	0

- The operation has to be analyzed with the 4 inputs combinations together with the 2 possible previous states.
- **When S = 0 and R = 0 :** If we assume Q = 1 and $\bar{Q}$ = 0 as initial condition, then output Q after input is applied would be Q = 1 and $\bar{Q}$ = 0. Assuming Q = 0 and $\bar{Q}$ = 1 as initial condition, then output Q after the input applied would be Q = 0 and $\bar{Q}$ = 1. So it is clear that when both S and R inputs are low, the output is retained as before the application of inputs. (i.e. there is no state change).
- **When S = 1 and R = 0 :** If we assume Q = 1 and $\bar{Q}$ = 0 as initial condition, then output Q after input is applied would be Q = 1 and $\bar{Q}$ = 0. Assuming Q = 0 and Q = 1 as initial condition, then output Q after the input applied would be $\bar{Q}$ = 1 and $\bar{Q}$ = 0. So in simple words when S is HIGH and R is low, output Q is high.
- **When S = 0 and R = 1 :** If we assume Q = 1 and $\bar{Q}$ = 0 as initial condition, then output Q after input is applied would be Q = 0 and $\bar{Q}$ = 1. Assuming Q = 0 and $\bar{Q}$ = 1 as initial condition, then output Q after the input applied would be Q = 0 and $\bar{Q}$ = 1. So in simple words when S is LOW and R is HIGH, output Q is LOW.
- **When S = 1 and R =1 :** No matter what state Q and $\bar{Q}$ are in, application of 1 at input of NOR gate always results in 0 at output of NOR gate, which results in both Q and $\bar{Q}$ set to LOW (i.e. Q = $\bar{Q}$). LOW in both the outputs basically is wrong, so this case is invalid.

The waveform below shows the operation of NOR gates based RS Latch.

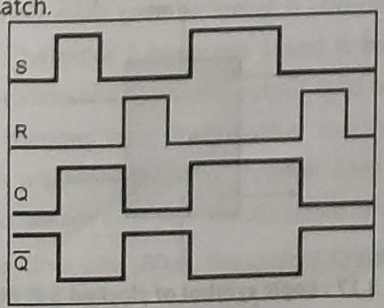


Fig. 3.12 : Waveform of SR latch using NOR gates

3.6.3 D Latch

- The SR latch seen earlier contains ambiguous state, to eliminate this condition we can ensure that S and R are never equal. This is done by connecting S and R together with an inverter.
- Thus, we have D Latch that is same as the RS latch, with the only difference being that there is only one input, instead of two (R and S). This input is called D or Data input.
- D latch is called D transparent latch for the reasons explained earlier. Delay flip-flop or delay latch is another name used. Below is the truth table and circuit of D latch.

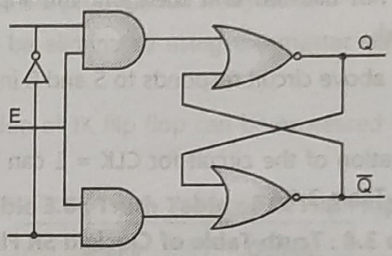


Fig. 3.13 (a)

Table 3.3 : Truth Table of D Latch

D	Q	Q+
1	X	1
0	X	0

- Below is the D latch waveform, which is similar to the RS latch one, but with R removed.

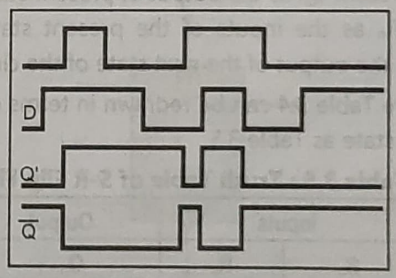


Fig. 3.13 (b) : D latch waveform

3.7 CLOCKED S-R FLIP-FLOP

- It is often required to enter the desired digital information in the memory cell, in synchronism with a train of pulses known as clock. The circuit of clocked S-R flip flop is as shown in Fig. 3.14.

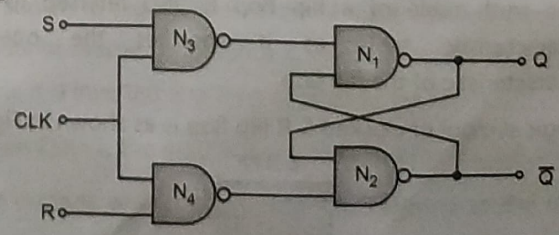


Fig. 3.14 : Clocked S-R flip flop

- In above circuit, when CLK = 0, output of both N_3 and N_4 is certainly 1. In this case, both S and R inputs have no effect on output Q.

- When CLK = 1, the operation of this circuit is exactly the same as that of SR latch.
- For S = R = 0, then output Q does not change i.e. if it is 0 it remains 0 and if it is 1, it remains 1. Thus, there is no change in the output for this input condition.
- For S = 1 and R = 0, the output Q becomes 1 in SR latch. This is known as the set state of the circuit.
- For S = 0 and R = 1 the output Q becomes 0 in SR latch. This is known as the reset state of the circuit.
- For S = R = 1, both the outputs Q and $\bar{Q}$ try to become 1 which is not allowed and therefore this input condition is prohibited.
- Thus, the above circuit responds to S and R inputs, only when CLK = 1.
- The operation of the circuit for CLK = 1 can be tabulated as shown in Table 3.4.

Table 3.4 : Truth Table of Clocked SR Flip-Flop

Inputs		Output
S	R	Q
0	0	No change
0	1	0 (Reset)
1	0	1 (Set)
1	1	Prohibited

- If we represent Q_n as the output of present state of the circuit and S_n, R_n as the inputs of the present state, then Q_{n+1} becomes the output of the next state of the circuit.
- The above Table 3.4 can be redrawn in terms of present state and next state as Table 3.5.

Table 3.5 : Truth Table of S-R Flip Flop

Inputs		Output
S_n	R_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Prohibited

- The Table 3.5 is the truth table of clocked S-R flip flop.
- The truth table of a flip flop is also referred to as the characteristic table as it specifies the operational characteristic of the flip flop.
- Logic symbol of clocked S-R flip flop is as shown in Fig. 3.15.

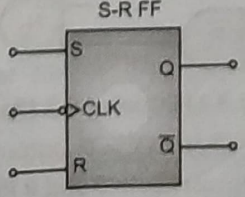


Fig. 3.15 : Logic symbol of clocked S-R flip flop

3.8 CLOCKED S-R FLIP FLOP WITH PRESET AND CLEAR INPUTS

- The circuit of clocked S-R flip flop shown in Fig. 3.15 switches to either set state or reset state when the power is turned on i.e. the state of the circuit is uncertain.
- In many applications it is required to define the initial state of the flip flop when the power is turned on.
- This is accomplished by using the preset and clear inputs.
- Preset and clear inputs are known as asynchronous inputs as they do not work in synchronism with the clock.
- Clocked S-R flip flop with preset and clear inputs can be obtained by using N_1 and N_2 NAND gates as 3 input gates as shown in Fig. 3.16.

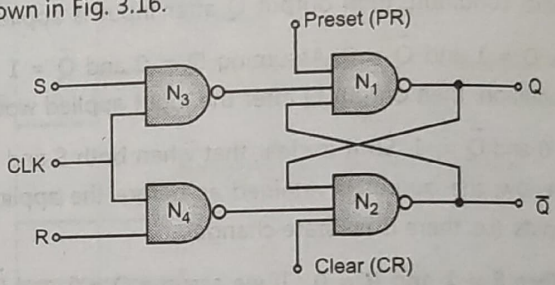


Fig. 3.16 : Clocked S-R flip flop with preset and clear inputs

- When PR = CR = 1, above circuit operates in accordance with the truth table of clocked S-R flip flop given in Table 3.4.
- When CR = 0 and PR = 1, one of the inputs of N_2 is 0, therefore its output is certainly high ($\bar{Q}$ = 1). Consequently all three inputs of N_1 are high which make Q = 0. Thus CR = 0 resets or clears the flip flop.
- Similarly, when CR = 1 and PR = 0, one of the inputs of N_1 is 0, therefore its output is certainly high (Q = 1). Consequently all three inputs of N_2 are high which make $\bar{Q}$ = 0. Thus PR = 0 sets the flip flop.
- Both preset and clear inputs are known as active low inputs as they perform the intended operation of setting or clearing the flip flop, when they are low.
- Once the desired initial state of the flip flop is achieved using preset and clear inputs, these inputs are connected to logic 1 while the normal operation of the flip flop takes place.
- The condition PR = CR = 0 must not be used, since this leads to an uncertain state.
- The logic symbol of this flip flop is as shown in Fig. 3.17. Preset and clear inputs are shown as bubbled inputs indicating that they are active low inputs.

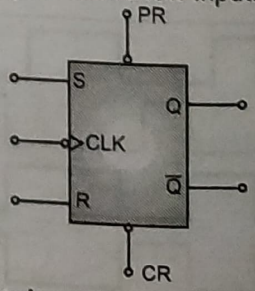


Fig. 3.17 : Logic symbol of clocked S-R flip flop with preset and clear inputs

3.9 JK FLIP FLOP

- We know that in case of clocked S-R flip flop, for the input condition S = R = 1 both the outputs Q and $\bar{Q}$ try to become 1, which is not allowed and therefore this input condition is prohibited.
- This drawback can be eliminated by converting S-R flip flop into a JK flip flop.
- The data input J is ANDed with $\bar{Q}$ to obtain S input and the data input K is ANDed with Q to obtain R input as shown in Fig. 3.18.

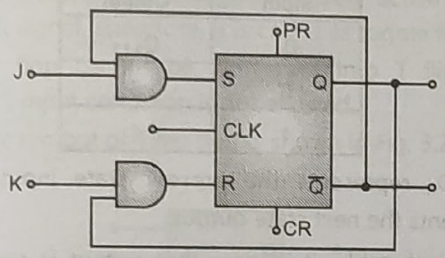


Fig. 3.18 : JK flip flop constructed using S-R flip flop

- When J = K = 0, output of both AND gates is 0. Therefore, S and R both become 0. So next state output Q_{n+1} remains same as that of present state output Q_n.
- When J = 0 and K = 1, the output of upper AND gate is 0, so S = 0. If the present state output Q_n = 0, the output of lower AND gate is also 0 and R becomes 0.
- For the input condition S = R = 0 the next state output remains unchanged. But if the present state output Q_n = 1, the output of lower AND gate becomes 1 i.e. R becomes 1.
- With S = 0 and R = 1 input combination the next state output Q_{n+1} is reset. Thus for J = 0 and K = 1 input condition, irrespective of the present state Q_n, the next state output Q_{n+1} be is 0 i.e. the flip flop is reset.
- Similarly, for J = 1 and K = 0 input condition, the next state output Q_{n+1} is certainly 1 i.e. the flip flop is set.

Race Around Condition

- The race around condition occurs for the input combination J = K = 1.
- Let us assume that initially the output Q is 0. With this the output of lower AND gate becomes 0 and upper AND gate becomes 1. Therefore S becomes 1 and R becomes 0. This input combination of S-R causes output Q to become 1. Thus the output changes from 0 to 1 after the time interval Δt equal to the propagation delay through AND gate and S-R flip flop. Now we have J = K = 1 and output Q = 1.
- After another time interval Δt, the output Q will change back to 0 and the cycle repeats till CLK=1.
- At the end of the clock pulse the output Q is uncertain and this situation is known as race around condition. It is shown in Fig. 3.19.

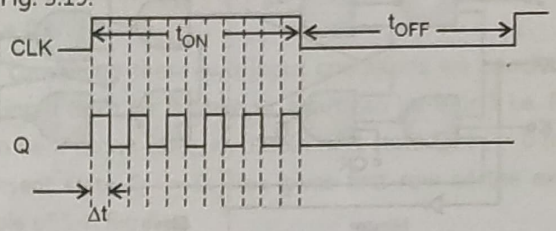


Fig. 3.19 : Timing diagram showing race around condition

- The race around condition can be eliminated if t_{ON} is made smaller than the propagation delay Δt.
- It can also be eliminated using the master slave JK (MS JK) flip flop.
- The operation of JK flip flop can be expressed with the truth Table 3.6.

Table 3.6 : Truth Table of JK Flip Flop

Inputs		Output
J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

- The logic symbol of JK flip flop is shown in Fig. 3.20.

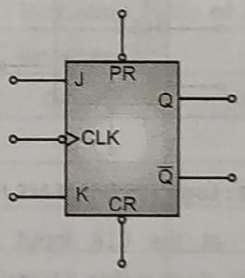


Fig. 3.20 : Logic symbol of JK flip flop

3.10 MASTER SLAVE JK (MS JK) FLIP FLOP

- Master slave JK flip flop is a cascade of two S-R flip flops as shown in Fig. 3.21.
- As shown in Fig. 3.21, outputs of slave are fed back to the inputs of master. Also, clock is directly applied to the master while it is inverted and then applied to the slave.
- When CLK=1, the master is enabled and the slave is disabled. The outputs of master Q_m and $\bar{Q}_m$ respond to the inputs J and K according to the Table 3.6. As long as CLK=1, Q and $\bar{Q}$ outputs do not change as the slave is disabled and therefore the fed back inputs of master also do not change.

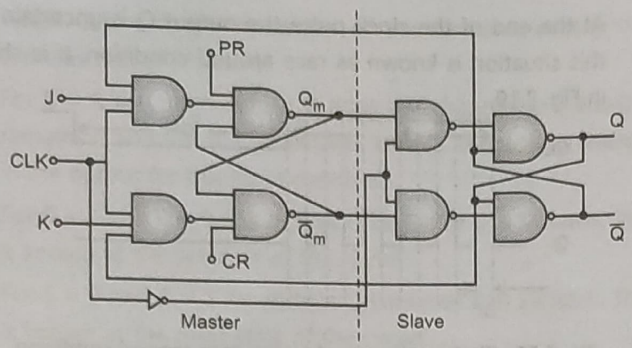


Fig. 3.21 : Master slave JK flip flop

- When CLK= 0, the slave is enabled and the master gets disabled. The outputs Q and $\bar{Q}$ change according to the outputs of the master Q_m and $\bar{Q}_m$. As long as CLK = 0, Q_m and $\bar{Q}_m$ outputs do not change as the master is disabled and therefore Q and $\bar{Q}$ outputs also retain their new values. Thus, the race around condition gets eliminated.
- The state of the master slave JK flip flop shown in Fig. 3.21, changes at the negative transition of the clock pulse.
- The logic symbol of master slave JK flip flop is shown in Fig. 3.22.

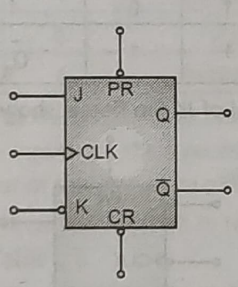


Fig. 3.22 : Logic symbol of MS JK flip flop

- The symbol '>' at the CLK input indicates that output changes when the clock makes a transition.
- The bubble indicates that the output changes when there is a negative transition of the clock (i.e. when the clock changes from 1 to 0).

3.11 D FLIP FLOP

- It has only one input called as data input (D).
- It is also known as data flip flop or delay flip flop.
- If we use only middle two rows of the truth table of S-R flip flop or JK flip flop, we obtain D flip flop.
- The middle two rows of both truth tables indicate that the two inputs S, R or J, K are always complements of each other.
- Thus, a D flip flop can be constructed from S-R flip flop or JK flip flop by connecting a NOT gate in between the two inputs as shown in Fig. 3.23.

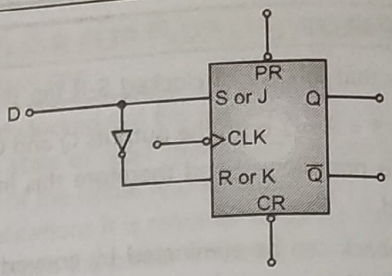


Fig. 3.23 : D flip flop using S-R flip flop or JK flip flop

- The truth table of D flip flop is as shown in Table 3.7.

Table 3.7 : Truth Table of D Flip Flop

Input	Output
D_n	Q_{n+1}
0	0
1	1

- Here D_n represents the present state input and Q_{n+1} represents the next state output.
- From truth table, it is clear that output is same as that of input therefore it is known as 'data' flip flop.
- The input data appears at the output at the end of the clock pulse. Thus, transfer of data from input to the output is delayed by clock pulse and hence it is also called as 'delay' flip flop.
- The logic symbol of D flip flop is shown in Fig. 3.24.

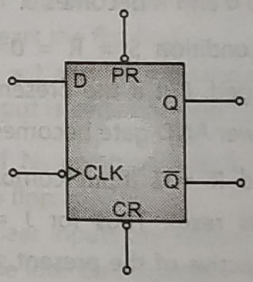


Fig. 3.24 : Logic symbol of D flip flop

3.12 T FLIP FLOP

- It has only one input called as toggle input (T). It is known as toggle flip flop.
- A T flip flop can be constructed from JK flip flop, just by connecting J and K input terminals together as shown in Fig. 3.25.

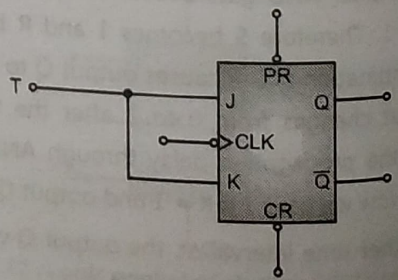


Fig. 3.25 : T flip flop using JK flip flop

- The truth table of T flip flop is as below.

Table 3.8 : Truth Table of T Flip Flop

Input	Output
T_n	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

- Here T_n represents the present state input, Q_n represents the present state output and Q_{n+1} represents the next state output.
- From truth table it is clear that, for T = 1, it acts as toggle switch. The output Q changes for every active transition of the clock signal. Therefore, it is called as toggle flip flop.
- S-R flip flop cannot be converted into T flip flop since S = R = 1 input condition is not allowed.
- The logic symbol of T flip flop is shown in Fig. 3.26.

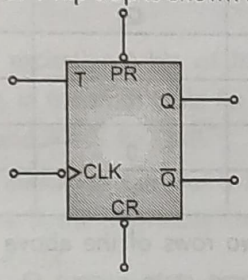


Fig. 3.26 : Logic symbol of T flip flop

3.13 DIFFERENT REPRESENTATION OF FLIP FLOP

- There are various ways in which a flip flop can be represented. Each represented type is used for a different application.
- **Different Types of Representation of Flip Flop are :**
  1. Characteristics equations.
  2. Flip flop as finite state machine.
  3. Excitation tables

3.14 EXCITATION TABLE OF FLIP FLOP

- In the design of sequential circuits, it is often required to find input conditions so that desired next state of the circuit is obtained from the present state of the circuit.
- These input conditions can be obtained using the excitation table of a flip flop.
- The truth table of a flip flop specifies its operational characteristic while the excitation table of a flip flop gives an idea regarding the present input conditions along with present state, to obtain the desired next state.
- Construction of excitation table is discussed below.

3.14.1 Excitation Table of S-R Flip-Flop

- Let the present state of the S-R flip flop be $Q_n = 0$ and the desired next state be $Q_{n+1} = 0$.
- As there is no change in the state of the flip flop (present state and next state is same), from the first row of the truth table of S-R flip flop (Pl. refer Table 3.8) we obtain the input condition as $S_n = 0$ and $R_n = 0$.
- Similarly, from the second row of the truth table of S-R flip flop, it is clear that whatever may be the present state, the next state of the flip flop is certainly 0 for the input condition $S_n = 0$ and $R_n = 1$.
- By combining these two input conditions we conclude that, S_n input must be 0 while R_n input can be 0 or 1 i.e. R_n input can be X (don't care), to obtain next state $Q_{n+1} = 0$ from the present state $Q_n = 0$. This gives first row of the excitation table of S-R flip flop.
- Similarly input conditions can be found for remaining three combinations of present state and next state. The excitation table is given in Table 3.9.

Table 3.9 : Excitation Table of S-R Flip Flop

Present State	Next State	Flip Flop Inputs	
Q_n	Q_{n+1}	S_n	R_n
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

3.14.2 Excitation Table of JK, D and T Flip Flop

- In the similar manner excitation table of JK, D and T flip flops can be prepared by using their truth tables. Table 3.10, Table 3.11 and Table 3.12 are the excitation table of JK, D and T flip flop respectively.

Table 3.10 : Excitation Table of JK Flip Flop

Present State	Next State	Flip Flop Input	
Q_n	Q_{n+1}	S_n	R_n
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table 3.11 : Excitation Table of D Flip Flop

Present State	Next State	Flip Flop Input
Q_n	Q_{n+1}	D_n
0	0	0
0	1	1
1	0	0
1	1	1

Table 3.12 : Excitation Table of T Flip Flop

Present State	Next State	Flip Flop Inputs
Q_n	Q_{n+1}	T_n
0	0	0
0	1	1
1	0	1
1	1	0

SOLVED EXAMPLES

Example 3.1 : *Prepare the truth table for the circuit shown in Fig. 3.27 and show that it acts as T type flip flop.*

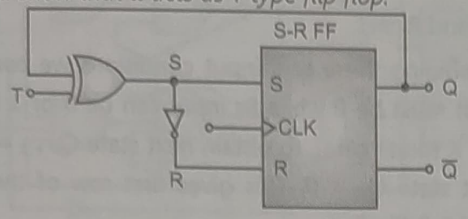


Fig. 3.27

Solution :

- Let us assume that initially Q=0 and T=0. Therefore, output of EX-OR is also 0 which leads S=0 and R=1. From the truth table of S-R flip flop, for this input condition, next state output is 0. This gives first row of the truth table for the circuit.
- Now let us assume that Q=1 and T=0.Therefore output of EX-OR is 1 which leads S = 1 and R = 0. From the truth table of S-R flip flop for this input condition, next state output is 1. This gives second row of the truth table for the circuit.
- Proceeding in a similar manner, we can obtain the remaining two rows of the truth table. The complete truth table is given in Table 3.13.

Table 3.13

Data Input	Present State	Next State
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

- From the first two rows of the Table 3.13, it is clear that when T=0, the next state output Q_{n+1} is same as that of present state output Q_n. From the last two rows of the Table 3.13, it is clear that, when T=1, the next state output Q_{n+1} is complement of the present state output Q_n. this can be represented in a tabular form as shown in Table 3.14.

Table 3.14

Data Input	Next State Output
T_n	Q_{n+1}
0	Q_n
1	$\overline{Q}_n$

- The Table 3.14 is the truth table of T type flip flop. Thus, the given circuit is same as that of T type flip flop.

Example 3.2 : *Analyze the circuit shown in Fig. 3.28 and prove that it is equivalent to T flip flop.*

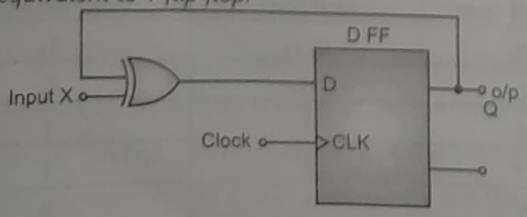


Fig. 3.28

Solution :

- Let us assume that initially X=0 and Q=0. Therefore, output of EX-OR gate is also 0 which leads D=0. From the truth table of D flip flop, for this input condition next state output is 0. This gives first row of the truth table for the circuit.
- Now let us assume that Q=1 and X=0. Therefore, output of EX-OR gate is 1 which leads D=1. From the truth table of D flip flop, for this input condition next state output is 1. This gives second row of the truth table for the circuit.
- Proceeding in a similar manner we can obtain the remaining two rows of the truth table. The complete truth table is given Table 3.15.

Table 3.15

Data Input	Present State	Next State
X	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

- From the first two rows of the above table, it is clear that when X=0 the next state output Q_{n+1} is same as that of present state output Q_n.
- From the last two rows of the above table it is clear that, when X=1, the next state output Q_{n+1} is complement of the present state output Q_n. This can be represented in tabular form as shown in Table 3.16.

Table 3.16

Data Input	Next State Output
X_n	Q_{n+1}
0	Q_n
1	$\overline{Q}_n$

- The Table 3.16 is the truth table of T-type flip flop. Thus the given circuit is equivalent to T type flip flop.

3.15 CONVERSION OF FLIP FLOPS

- Conversion of Flip flops is based upon the block diagram as shown in Fig. 3.29.

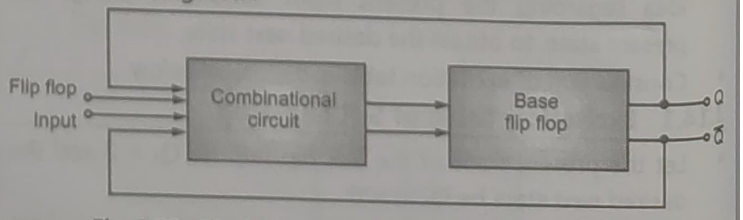


Fig. 3.29 : Block diagram used for flip flop conversion

- Base flip flop shown in the Fig. 3.29 is the flip flop to be converted. Here we need to design a combinational circuit for converting the base flip flop into desired one.

- For designing the combinational circuit, we have to use the excitation tables of both, base flip flop and desired flip flop. From that we construct a truth table with desired flip flop data inputs, present state Q, as inputs and base flip flop data inputs as outputs.
- Then we write separate K-maps for individual outputs and obtain the simplified expressions. Based on these expressions we get the combinational circuit required for conversion.

3.15.1 Convert S-R Flip Flop to JK Flip Flop

- Here the base flip flop is S-R and desired flip flop is JK.
- We first construct a truth table in which inputs are - desired flip flop data inputs i.e. J, K and present state Q. In the truth table outputs are - based flip flop data inputs i.e. S,R.
- Using the excitation table of both flip flops we construct the truth Table 3.17.
- The first row of excitation table of JK flip flop for Q=0 is JK=0X. Therefore, in the truth table for first two rows we get inputs as JK=00 and JK=01 for Q=0.
- The first row of excitation table of S-R flip flop for Q=0 is S-R=0X. Therefore, in the truth table for first two rows we get the outputs as S-R=0X.
- Proceeding in this manner we obtain the truth Table 3.17.
- In this table cell number for the K-map is also written so that it becomes easy while representing the truth table in the K-map.
- Fig. 3.30 in the right bottom corner of K-map cell indicates the cell numbers.

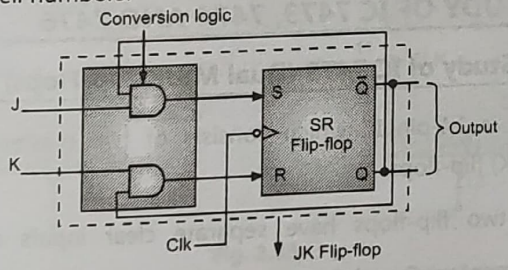


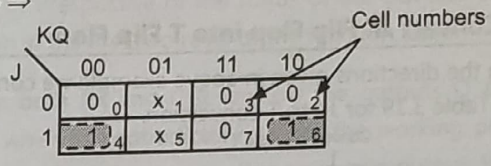
Fig. 3.30 : S-R to JK conversion

Table 3.17 : Truth Table for S-R to JK Conversion

Cell no.	Desired FF Data Inputs		Present State	Base FF Data Inputs	
	J	K	Q	S	R
0	0	0	0	0	x
2	0	1	0	0	x
4	1	0	0	1	0
6	1	1	0	1	0
3	0	1	1	0	1
7	1	1	1	0	1
1	0	0	1	x	0
5	1	0	1	x	0

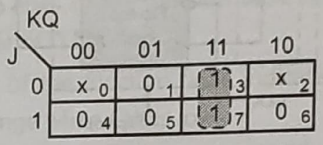
- Now we write separate K-maps for S and R outputs according to the cell numbers.

(1) For S ⇒



$\therefore \quad S = J\bar{Q}$

(2) For R ⇒



$\therefore \quad R = KQ$

- The resulting conversion diagram is as shown in Fig. 3.30.

3.15.2 Convert S-R Flip Flop to D Flip Flop

- Using the directions given for S-R to JK conversion we construct the truth Table 3.18 for S-R to D conversion.

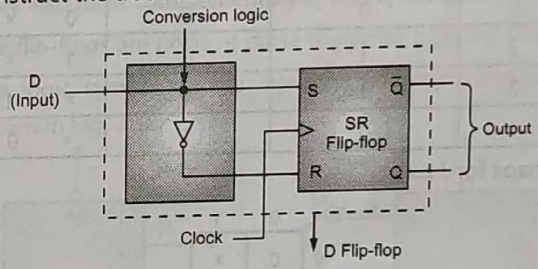
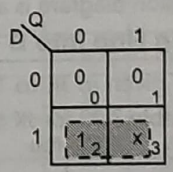


Fig. 3.31 : S-R to D conversion

Table 3.18 : Truth Table for S-R to D Conversion

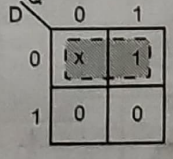
Cell No	Desired FF Data Input	Present State	Base FF Data Inputs	
	D	Q	S	R
0	0	0	0	x
2	1	0	1	0
1	0	1	0	1
3	1	1	x	0

K map for S ⇒



$\therefore \quad S = D$

K map for R ⇒



$\therefore \quad R = \bar{D}$

- The resulting conversion diagram is as shown in Fig. 3.31.

3.15.3 Convert S-R Flip Flop into T Flip Flop

- Converting SR flip flop to T flip flop is same as SR to D flip flop conversion.

3.15.4 Convert JK Flip Flop into T Flip Flop

- Using the directions given in above example we construct the truth Table 3.19 for JK to T conversion.

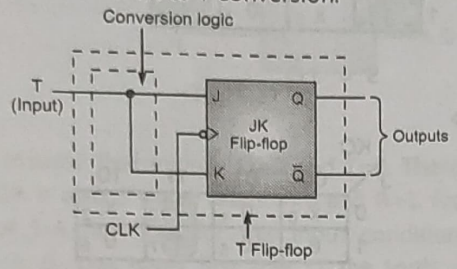
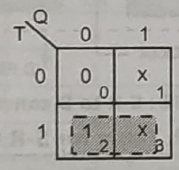


Fig. 3.32 : JK to T flip flop conversion

Table 3.19 : Truth Table for JK to T conversion

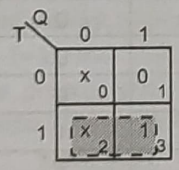
Cell No	Desired FF Data Input	Present State	Base FF Data Input	
	T	Q	J	K
0	0	0	0	×
2	1	0	1	×
3	1	1	×	1
1	0	1	×	0

- K maps for J ⇒



$\therefore \quad J = T$

- K maps for K ⇒



$\therefore \quad K = T$

- The resulting conversion diagram is as known in Fig. 3.32.

3.15.5 Convert JK Flip Flop into D Flip Flop

- Using the directions given in JK to T flip flop conversion we construct the truth Table 3.20 for JK to D conversion.

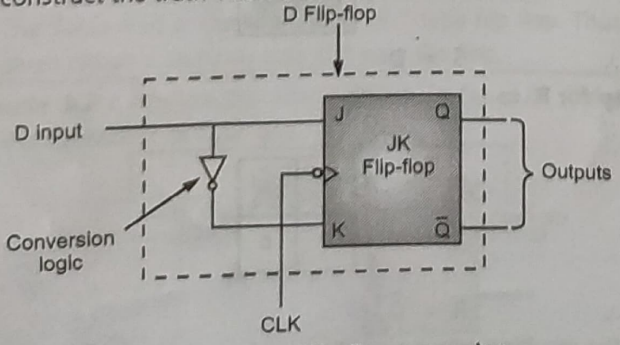
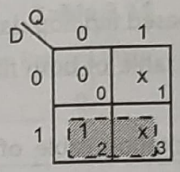


Fig. 3.33 : JK to D flip flop conversion

Table 3.20 : Truth Table for JK to D Conversion

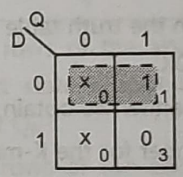
Cell No	Desired FF Data Input	Present State	Base FF Data Input	
	D	Q	J	K
0	0	0	0	x
2	1	0	1	x
1	0	1	x	1
3	1	1	x	0

K – map for J=>



$\therefore \quad J = D$

K – map for K=>



$\therefore \quad K = \bar{D}$

- The resulting conversion diagram is as shown in Fig. 3.33.

3.16 STUDY OF IC 7473, 7474 AND 7476

3.16.1 Study of IC 7473 (Dual MSJK Flip-Flops)

- 7473 is 14-pin IC which consists of two master slave JK (MSJK) flip-flops.
- The two flip-flops have separate clear inputs and have commentary Q and $\bar{Q}$ outputs.
- The flip-flops are positive pulse triggered type of flip-flops.
- The logic symbol of the two flip-flops with pin numbers is as shown in Fig. 3.34.

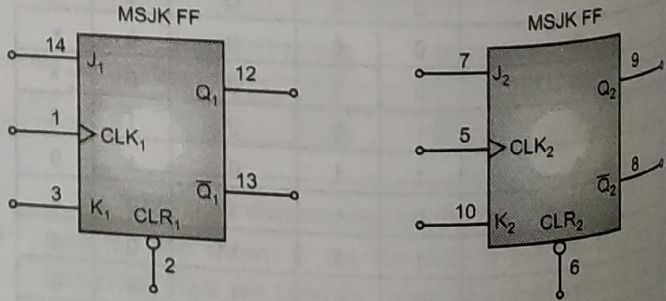


Fig. 3.34

- The function table of IC 7473 is as shown in Table 3.21.

Table 3.21

Input				Outputs	
CLR	CLK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	⎍	L	L	Same as present state	
H	⎍	L	H	L	H
H	⎍	H	L	H	L
H	⎍	H	H	Toggle	

- As shown in Table 3.21 when CLR = 0 the output of the flip-flop is certainly low (Q = 0) irrespective of the status of CLK, J and K inputs which is the first line of the function table.
- The remaining four lines of the function table are same as that of the truth table of MSJK flip-flop.

3.16.2 Study of IC 7474 (Dual D Flip-Flops)

- 7474 is a 14 - pin IC which consists of two D flip-flops.
- The two D flip-flops have separate preset and clear inputs.
- They also have complementary Q and $\bar{Q}$ outputs
- The flip-flops are positive edge triggered type flip-flops.
- The logic symbol of the two flip-flops with pin numbers is as shown in Fig. 3.35.

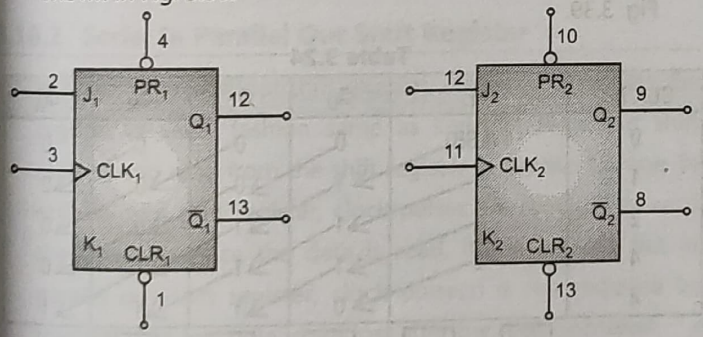


Fig. 3.35

- The function table of IC 7474 is as shown in Table 3.22.

Table 3.22

Inputs				Outputs	
PR	CLR	CLK	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H	H
				(Invalid output)	
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	same as present state	

When PR = 0 the output of the flip-flop GIs certainly high (Q = 1) and when CLR = 0, the output is certainly low (Q = 0) irrespective of the status of the CLK and D inputs as shown in the first two rows of the function table.

- When both PR and CLR are low the outputs Q and $\bar{Q}$ are high which is contradictory as per the working principle of flip-flop. Therefore, we say it is an invalid output. It is shown in third row of the function table.
- The next two lines of the function table are same as that of the truth table of D flip-flop.
- The last row of the function table indicates that the output does not change when CLK is low.

3.16.3 Study of IC 7476 (Dual MS JK Flip-Flop)

- 7476 is 16 - pin IC which consists of two master slave JK (MS JK) flip-flops.
- The two flip-flops have separate clear and preset inputs.
- They also have complementary Q and $\bar{Q}$ outputs.
- The flip-flops are positive pulse triggered type flip-flops.
- The logic symbol of two flip-flops with pin numbers is as shown in Fig. 3.36.

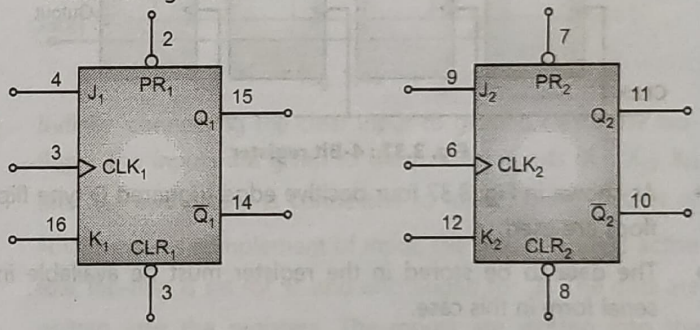


Fig. 3.36

- The function table of IC 7476 is as shown in Table 3.23.

Table 3.23

Inputs					Outputs	
PR	CLR	CLK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H	H
					(Invalid output)	
H	H	⎍	L	L	same as present state	
H	H	⎍	L	H	L	H
H	H	⎍	H	L	H	L
H	H	⎍	H	H	Toggle	

- When PR = 0 the output of the flip-flop is certainly high (Q = 1) and when CLR = 0, the output is certainly low (Q = 0) irrespective of the status of CLK, J and K inputs as shown in the first two rows of the function table.

- When both PR and CLR are low the outputs Q and $\overline{Q}$ are high which is contradictory as per the working principle of flip-flop. Therefore, we say it is an invalid output. It is shown in third row of the function table.
- The last four rows of the function table are same as that of the truth table of MS JK flip-flop.

3.17 APPLICATION OF FLIP FLOPS

3.17.1 Registers

- Registers are used for storing the digital information. A register that stores N-bit information is called as N-bit register.
- Flip-flops are used for the construction of registers. As flip-flop can store 1-bit information, a N-bit register consists of N flip-flops.
- D type flip-flops are most widely used in the registers. Also JK flip-flop and SR flip-flop when converted to D flip-flop can be used in the registers.
- A 4-bit register is shown in Fig. 3.37.

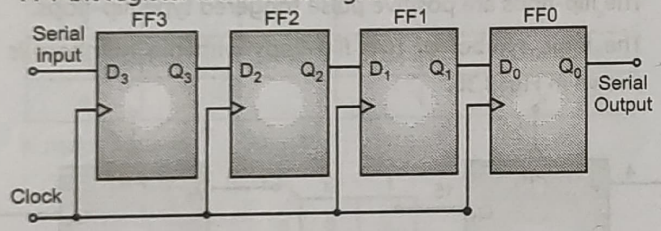


Fig. 3.37 : 4-Bit register

- As shown in Fig. 3.37 four positive edge triggered D type flip flops are used.
- The data to be stored in the register must be available in serial form in this case.
- The serial data is applied at the serial input bit, by bit starting from least significant bit along with the clock pulses.
- As Q_3 is connected to D_2, Q_2 is connected to D_1 and Q_1 is connected to D_0, the data gets shifted from one flip-flop to another. After four clock pulses the 4-bit information gets stored in the register.

3.18 SHIFT REGISTERS

- The data can be entered in serial or parallel form and can be retrieved in the serial or parallel form. The serial form means bit by bit (one bit at time) and parallel means all the bits are simultaneously retrieved. On the basis of data entered (write) and retrieved (read), the registers are classified as,
  - Serial In Serial Out
  - Serial In Parallel Out
  - Parallel In Serial Out
  - Parallel In Parallel Out
- Registers, in which data are entered or/and retrieved in serial form, are referred to as shift register.

3.18.1 Serial In Serial Out Shift Register

- In serial in serial out shift register, data is entered and retrieved in serial fashion with clock.
- The logic diagram of 4-bit serial in serial out shift register using J-K flip-flop is shown Fig. 3.38.

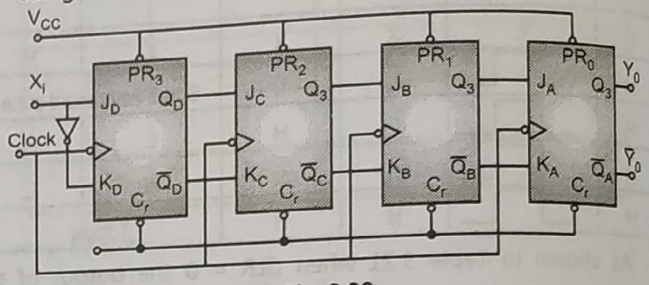


Fig. 3.38

- In Fig. 3.38, X_i is input and Y_O is output of serial in serial out shift register.
- The process of entering the digital data starts with the least significant bits. The data input is entered with falling edge of clock pulse, hence number of clock pulses required to enter the data is equal to the length of digital data or size of shift register. The data is read, bit by bit at output Y_O with clock pulse.
- Let us consider the data 0111 is applied to the input. How the data was entered in shift register is given in Table 3.24 and waveforms of shift register for serial input are shown in Fig. 3.39.

Table 3.24

CLK No.	X_i	Q_D	Q_C	Q_B	Q_A
0	1 (LSB)	0	0	0	0
1	1	1	0	0	0
2	1	1	1	0	0
4	0	1	1	1	0
4		0	1	1	1

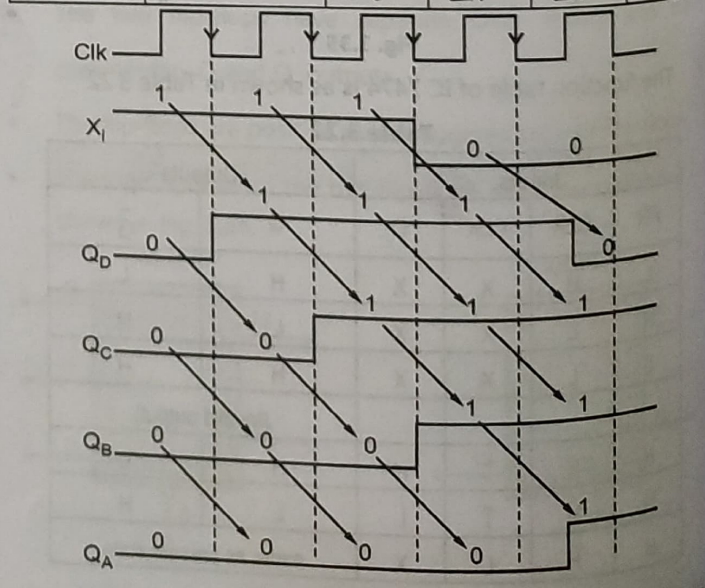


Fig. 3.39

Operation :

The input of flip-flop D is Xi, flip-flop of C is Q_D, flip-flop of B is Q_C and flip-flop of A is Q_B. All flip-flops operate as D flip-flop and input applied is 0111.

Initially shift register is cleared.

$Q_D\ Q_C\ Q_B\ Q_A$ = 0000 and input $X_i = 1$

- At negative edge of first clock pulse, the input data is entered into the flip-flop and at the end of first clock pulse,

  $Q_D\ Q_C\ Q_B\ Q_A$ = 1000 and input $X_i = 1$

- At negative edge of second clock pulse, the inputs are entered and at the end of second clock pulse,

  $Q_D\ Q_C\ Q_B\ Q_A$ = 1100 and input $X_i = 1$

- At negative edge of third clock pulse, the inputs are entered and at the end of third clock pulse,

  $Q_D\ Q_C\ Q_B\ Q_A$ = 1110 and input $X_i = 0$

- At negative edge of fourth clock pulse, the inputs are entered and at the end of fourth clock pulse,

  $Q_D\ Q_C\ Q_B\ Q_A$ = 0111

Disadvantages :

- n clock pulses are required to enter the n-bit data.
- n clock pulses are required to read the n-bit data.
- Once the data is read, it will be lost.

3.18.2 Serial In Parallel Out Shift Register

- In serial in parallel out shift register, data is entered into the register in serial fashion same as serial in serial out shift register and read from the shift register in parallel fashion. In serial output shift register, clock pulses are required to read the data and once the data is read, it will be lost, but in parallel out shift register, clock pulse(s) is not required to read the data and data is not lost after the read operation.
- The logic diagram of four-bits serial in parallel out shift register using D flip-flop is shown in Fig. 3.40.

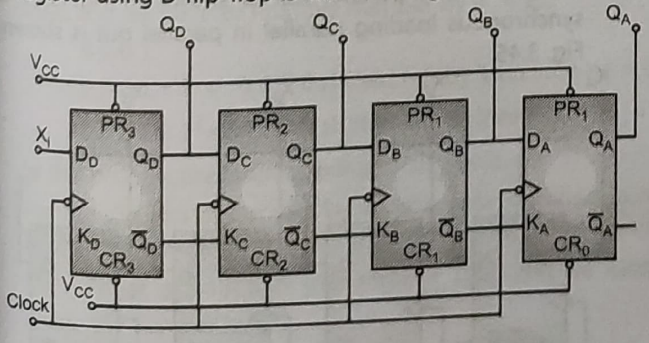


Fig. 3.40

where X_i is serial input for shift register and $Q_D\ Q_C\ Q_B\ Q_A$ are the parallel outputs of shift register.

3.18.3 Parallel In Serial Out Shift Register

- In parallel in serial out shift register, the data is entered in parallel fashion and data is read in serial fashion.

There Are Two Types of Parallel Loading :

1. Asynchronous Loading,
2. Synchronous Loading.

1. Asynchronous Loading :

- In asynchronous loading, the preset inputs are used to load the data simultaneously. The logic diagram of four bit parallel in serial out shift register with asynchronous loading is shown in Fig. 3.41.

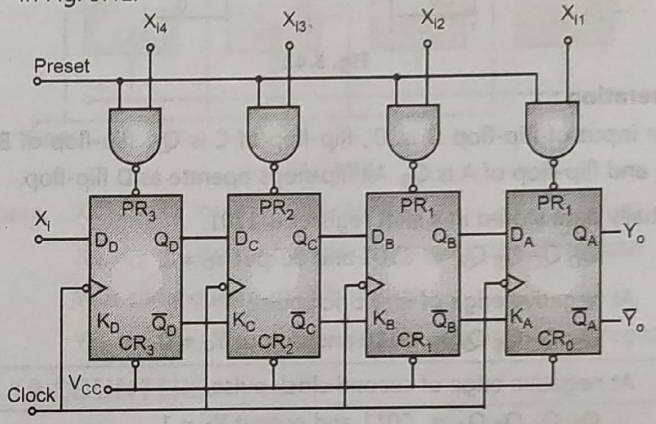


Fig. 3.41

- Initially, connecting the clear input to ground clears the flip-flops. The inputs are given to the parallel inputs (X_{i1}, X_{i2}, X_{i4} and X_{i4}) and preset is connected to logic '0', the output of NAND gate is complement of input, the preset input is active low, flip-flop is set for '0' and unchanged for '1'. The data are written into the registers. The inputs are written into the registers without clock pulse. Such parallel loading is known as asynchronous loading. The data is read from output lines Y_O bit by bit by applying the clock pulse. Once, data is read, it will be lost.
- For example, let us assume the data stored in a shift register is 0101 and it will be read from the output line Y_O. How the data is read from the shift register is given in Table 3.25 and waveforms of shift register for serial out are shown in Fig. 3.42.

Table 3.25

CLK No.	Q_D	Q_C	Q_B	Q_A	Y_O
0	1	1	0	1	1
1	0	1	1	0	0
2	0	0	1	1	1
4	0	0	0	1	1
4	0	0	0	0	0

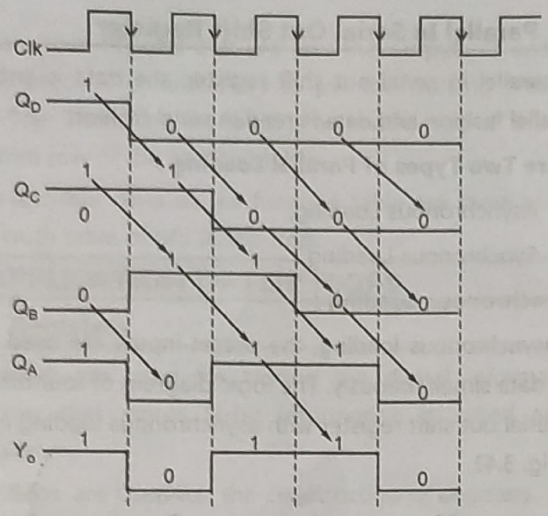


Fig. 3.42

Operation :

The input of flip-flop D is '0', flip-flop of C is Q_D, flip-flop of B is Q_C and flip-flop of A is Q_B. All flip-flops operate as D flip-flop.

Initially data stored in a shift register is 1101.

$Q_D\ Q_C\ Q_B\ Q_A$ = 1101 and output Y_O = 1

- At negative edge of first clock pulse,
  $Q_D\ Q_C\ Q_B\ Q_A$ = 0110 and output Y_O = 0
- At negative edge of second clock pulse,
  $Q_D\ Q_C\ Q_B\ Q_A$ = 0011 and output Y_O = 1
- At negative edge of third clock pulse,
  $Q_D\ Q_C\ Q_B\ Q_A$ = 0001 and output Y_O = 1
- At negative edge of fourth clock pulse,
  $Q_D\ Q_C\ Q_B\ Q_A$ = 0000 and output Y_O = 0

In n-bit parallel in parallel out shift register, n clock pulses are required to read the data and once the data is read, it will be lost.

2. Synchronous Loading :

- In synchronous loading, the input data is entered in parallel form with clock pulse. The logic diagram of four bit parallel in serial out shift register in synchronous mode is shown in Fig. 3.43.

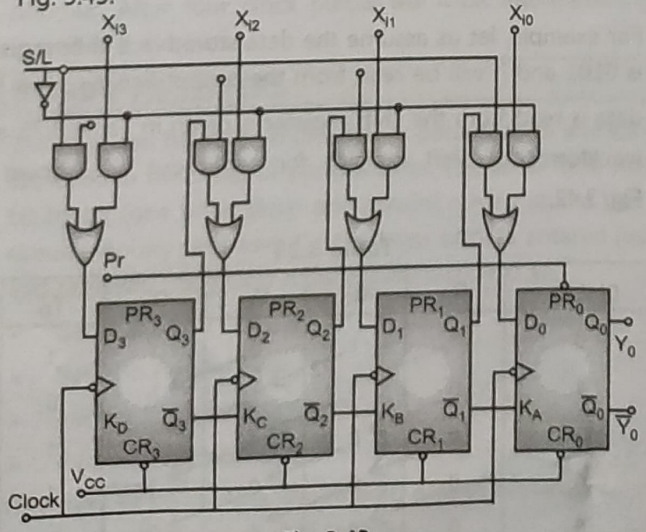


Fig. 3.43

- Shift/ $\overline{\text{Load}}$ control signal is used to control the operation of the shift register. When Shift/ $\overline{\text{Load}}$ is at logic '1', the data is read from Y_O bit-by-bit with clock pulse. When Shift/ $\overline{\text{Load}}$ is at logic '0', the data inputs X_{i1}, X_{i2}, X_{i4} and X_{i4} load simultaneously with clock pulse. Such type of loading is referred as synchronous loading.
- For example : Consider data inputs are 0101.
- When Shift/ $\overline{\text{Load}}$ signal is 0, the outputs of gates G_1, G_3, G_5 and G_7 are 0 and outputs of gates G_2, G_4, G_6 and G_8 are same to the inputs 0101. The outputs of OR gates are 0101, and these are inputs to D flip-flops. It is loaded into register with falling of the clock pulse.
- When Shift/ $\overline{\text{Load}}$ signal is 1. the outputs of gates G_2, G_4, G_6 and G_8 are 0 and the outputs of gates G_1, G_3, G_5 and G_7 are 0. $Q_4\ Q_2\ Q_1$. The outputs of OR gates are 0 $Q_4\ Q_2\ Q_1$, and these are the input to D flip-flops. It is loaded into the register with falling edge of clock pulse. It shows that data is shifted in register and we get output at Y_O.

3.18.4 Parallel In Parallel Out Shift Register

- In parallel in parallel out shift register, data is entered as well as read in parallel fashion.

There are Two Types of Parallel Loading :

1. Asynchronous loading
2. Synchronous loading.

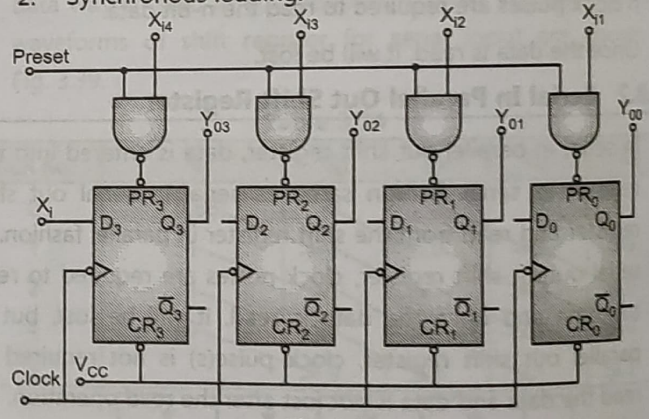


Fig. 3.44

- The logic diagram of four-bit asynchronous loading parallel in and parallel out is shown in Fig. 3.44 and synchronous loading parallel in parallel out is shown in Fig. 3.45.

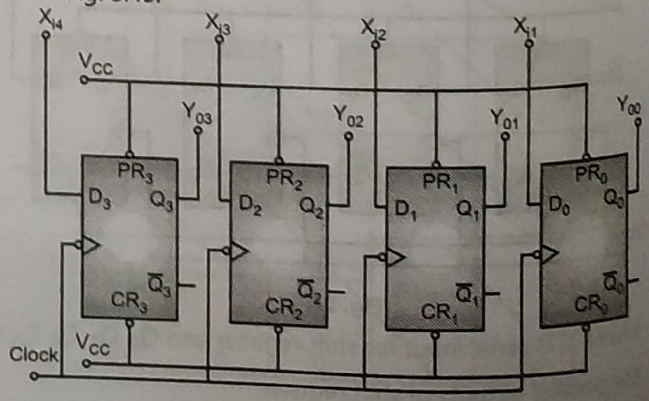


Fig. 3.45

3.18.5 Bi-Directional Shift Register

- In bi-directional shift register, the data is shifted to left as well as to right direction. The direction is controlled by the control input $R/\overline{L}$. The four-bit bi-directional shift register is shown in Fig. 3.46
- When $R/\overline{L}$ control signal is high, the gates G_1, G_3, G_5 and G_7 are enabled. The output of flip-flop A is input for flip-flop B, the output of flip-flop B is input for flip-flop C, the output of flip-flop C is input for flip-flop D and X_{iR} is input for flip-flop. Data is shifted right with clock pulse.
- When $R/\overline{L}$ control signal is low, the gates G_2, G_4, G_6 and G_8 are enabled. The output of the flip-flop D is input for flip-flop C, the output of the flip-flop C is input for flip-flop B, the output of the flip-flop B is input for flip-flop A and X_{iL} is input for flip-flop D. Data is shifted left with clock pulse.

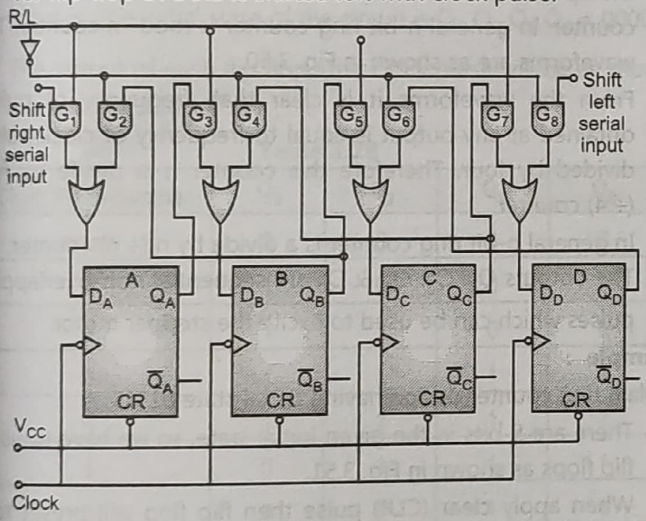


Fig. 3.46

Application of Bi-Directional Shift Register :

- The bi-directional shift register is used to multiply or divide the number by 2^n, provided that '1' is not shifted out of register. For the multiplication of number by 2^n, the data is shifted in left side by the amount of n bits with $X_{iL} = 0$.
- **For Example :** Consider the number is loaded in shift register 0001 and we have to multiply number by $4 = 2^2$.

  0 0 0 1

  $0\,0\,0\,1 \times 2 =$ 0 0 1 0 shifted left by 1 bit with $X_{iL} = 0$

  $0\,0\,0\,1 \times 2^2 =$ 0 1 0 0 shifted left by 2 bits with $X_{iL} = 0$
- In this process the most significant bit is lost.
- For the division of number by 2^n, the data is shifted in right by the amount of n bits with $X_{iR} = 0$. For example, consider the number is loaded in shift register 1000 and we have to divide the number by 2^2.

  1 0 0 0

  $1\,0\,0\,0 / 2 =$ 0 1 0 0 shifted right by 1 bit with $X_{iR} = 0$

  $1\,0\,0\,0 / 2^2 =$ 0 0 1 0 shifted right by 2 bits with $X_{iR} = 0$

  In this process the least significant bit is lost.

3.18.6 Universal Register

- The universal shift register operates in all possible four modes (SISO, SIPO, PISO, PIPO) and also as bi-directional shift registers. Logic diagram of four-bit shift register operates in all four modes as shown in Fig. 3.47.

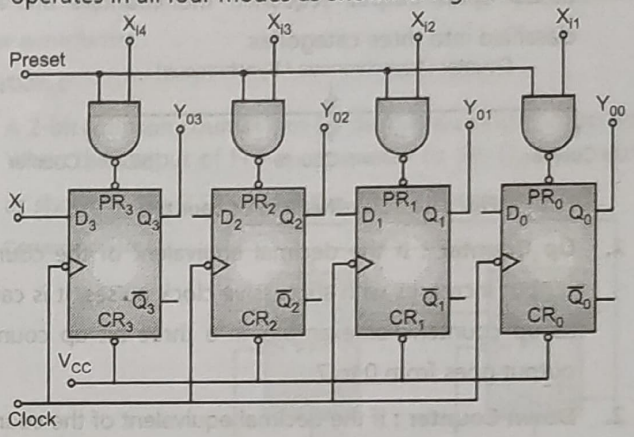


Fig. 3.47

where, X_i is serial input of shift register

X_{i4}, X_{i2}, X_{i1}, X_{i0} are four parallel inputs of shift register

Y_0 is serial output of shift register

Y_{04}, Y_{02}, Y_{01}, Y_{00} are four parallel outputs of shift register.

3.19 COUNTERS

- Counters are used for counting the number of events occurred.
- A circuit that counts electrical pulses applied as input to it is known as a counter.
- In practice these circuits are used as event counters i.e. to count number of events occurred. Electrical pulses are generated corresponding to the occurrence of an event and these pulses are given as input to the counters.
- Flip-flops are used for the construction of counters an N-bit counter consists of N flip-flops.
- A counter with n flip-flops has 2^n possible states. Therefore a 3 bit up counter can count from 0 to 7 while 4-bit down counter can count from 15 down to 0.
- Number of distinct states in the operation of counter is known as modulus of that counter and that counter is called mod 2^n counter.

  e.g. 3-bit counter, the number of states is $2^3 = 8$. Thus modulus of three bit counter is 8 and it is also called as modulo 2^3 i.e. mod 8 counter.

3.19.1 Classification of Counters

Basically Counters are Divided into Two Types as Follows :

1. **Synchronous Counters :** In synchronous counters all the flip-flops receive the external clock pulse simultaneously e.g. Ring and Johnson counter.

2. **Asynchronous Counters :** For asynchronous counters the external clock signal is applied to one flip-flop and then the output of preceding flip-flop is connected to the clock of next flip-flop.

- Based upon output sequence the counters are also classified into three categories.

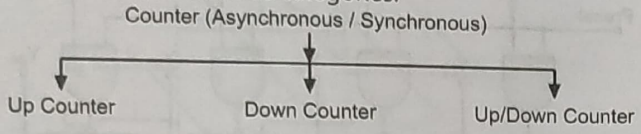


Fig. 3.48 : Classification of counters

1. **Up Counter :** If the decimal equivalent of the counter output increases with successive clock pulses, it is called as up counter. For example, in a three bit up counter, output goes from 0 to 7.
2. **Down Counter :** If the decimal equivalent of the counter output decreases with successive clock pulses, it is called as down counter. For example, in a four bit down counter, output goes from 15 down to 0.
3. **Up/Down Counter :** A counter which can count in any direction i.e. up or down, depending upon direction control input is called as up/down counter.

3.19.2 Ring Counter

- The shift register acts as ring counter, if the serial output is connected back to the serial input as shown in Fig. 3.49.

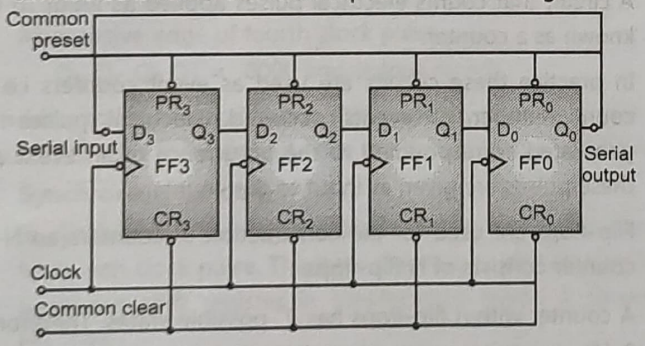


Fig. 3.49 : Ring counter

- Let us consider the initial state of the circuit as $Q_3\ Q_2\ Q_1\ Q_0$ = 1000.
- The output of each flip flop after every clock pulse will be as shown in the table 3.26.

Table 3.26

Clock Pulse Number	Q_3	Q_2	Q_1	Q_0
Initially	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0

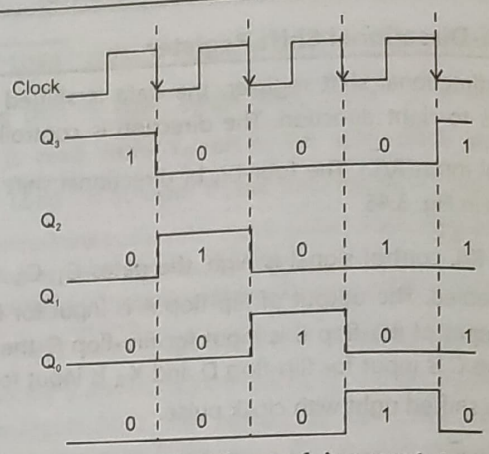


Fig. 3.50 : Waveforms of ring counter

- The n-bit ring counter counts n clock pulses, therefore the circuit shown in Fig. 3.50 counts four clock pulses.
- From table 3.26 it is clear that the number of distinct states in the operation of this counter is four. Therefore, it is a mod - 4 counter. In general n-bit ring counter is mod- n counter. The waveforms are as shown in Fig. 3.50.
- From the waveforms it is clear that, frequency of pulses obtained at any output is equal to frequency of clock pulses divided by four. Therefore this counter is a divide by four (÷ 4) counter.
- In general n-bit ring counter is a divide by n (÷ n) counter.
- The outputs Q_3, Q_2, Q_1 & Q_0 are sequential non-overlapping pulses which can be used to excite the stepper motor.

Example :

Explain ring counter design having initial state 01011.

- There are 5 bits in the given initial state, so we have to use 5 flip flops as shown in Fig. 3.51.
- When apply clear (CLR) pulse then flip flop will preset to 1 output and 4 and 2 are reset to 0 output.

$Q_4\ Q_3\ Q_2\ Q_1$ = 01011

Fig. 3.51 shows the arrangement of 5 bit ring counter.

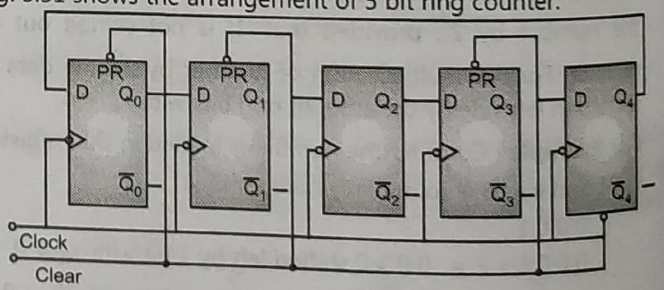


Fig. 3.51

Table 3.27 : 5-Bit Ring Counter Table

CLR	CLK	Q_0	Q_1	Q_2	Q_3	Q_4
0	1	1	1	0	1	0
1	1	0	1	1	0	1
1	1	1	0	1	1	0
1	1	0	1	0	1	1
1	1	1	0	1	0	1
1	1	1	1	0	1	0

3.19.3 Johnson Counter or Twisted Ring Counter

- It is also known as twisted ring counter or moebius counter.
- The shift register acts as a Johnson counter if the $\bar{Q}_0$ output is connected to the serial input as shown in Fig. 3.52.

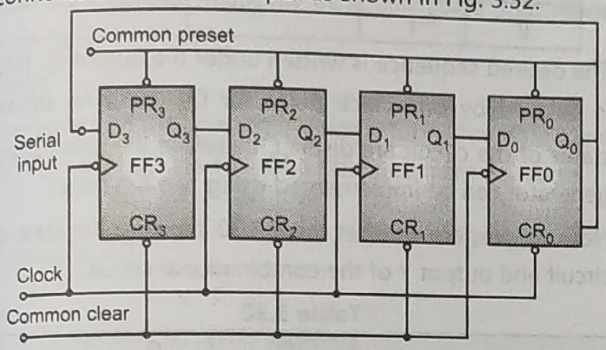


Fig. 3.52 : Johnson counter

- Initially all the flip flops are cleared using the common clear input ie. the initial state of the circuit is $Q_3\ Q_2\ Q_1\ Q_0 = 0000$.
- The output of each flip flop after clock pulse will be as shown in the table 3.28.

Table 3.28

Clock Pulse Number	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

- From table 3.28 it is clear that the number of distinct states in the operation of this counter is eight. Therefore it is a mod 8 counter. In general, every n-bit Johnson counter is a mod 2n counter. The waveforms are as shown in Fig. 3.53.

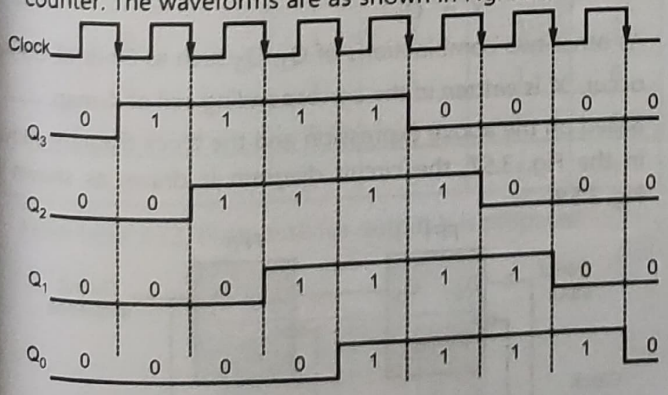


Fig. 3.53 : Waveforms of Johnson counter

- As shown in Fig. 3.53 square waveforms are obtained at each flip flop output.
- Also for completion of one cycle, each output requires eight clock pulses therefore it is a divide by 8 (÷ 8) counter.
- In general, n-bit Johnson counter is a divide by 2n (÷ 2n) counter.

Example 3.3 : *Design Johnson counter using 2-bit shift register. Draw waveforms.*

Solution :

- A 2-bit Johnson counter can be designed using two flip flops. When the output of FF0 is connected to the D_1 input of FF1 as shown in Fig. 3.54 we get the 2 bit Johnson counter.

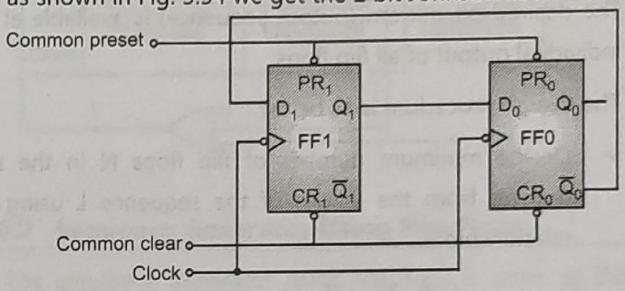


Fig. 3.54 : 2-bit Johnson counter

- The waveforms are shown in Fig. 3.55.

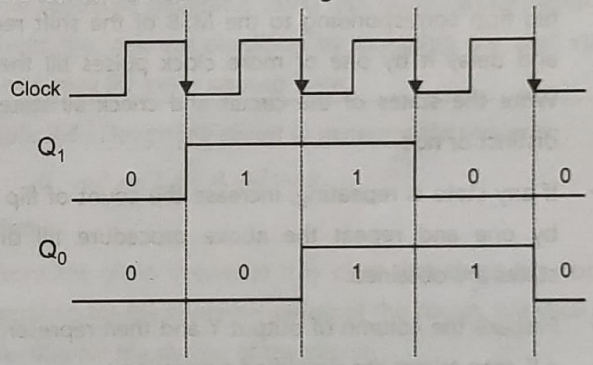


Fig. 3.55 : Waveforms of 2-bit Johnson counter

3.20 SEQUENCE GENERATOR

- Sequence generator circuits can be built with shift registers or flip flops. The two methods are discussed here.

3.20.1 Sequence Generator Using Shift Registers

- A circuit that generates prescribed sequence of bits upon application of clock pulses is known as sequence generator.
- For this type of sequence generator shift register is used as the basic building block. The block diagram of it is as shown in Fig. 3.56.

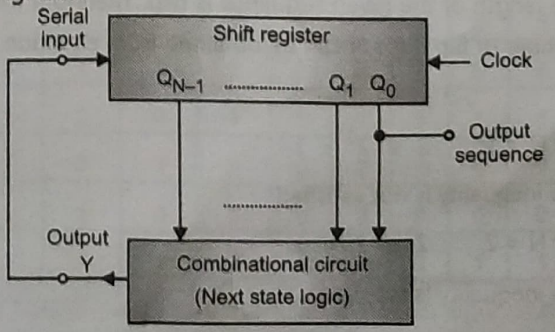


Fig. 3.56 : Block diagram for sequence generator

- As shown in Fig. 3.56, a Serial In Parallel Out (SIPO) shift register is used.
- Depending upon requirement outputs of one or more flip flops of the shift register are connected as input to the combinational circuit which is also known as next state logic. Further the output Y of the combinational circuit is connected as the serial input of the shift register.
- Here main task is to design the combinational circuit that decides the serial input of the flip flop.
- The desired bit pattern (output sequence) is available at the individual output of all flip flops.
- The design procedure is as below.
  - Decide minimum number of flip flops N in the shift register from the length of the sequence L using the relationship.

    $$L \leq 2^N - 1$$
  - Write down the desired sequence under the output of flip flop corresponding to the MSB of the shift register and delay it by one or more clock pulses till the LSB. Write the states of the circuit and check all states are distinct or not.
  - If any state is repeating, increase the count of flip flops by one and repeat the above procedure till distinct states are obtained.
  - Prepare the column of output Y and then represent it in a K-map to get the simplified expression.
  - Built the combinational circuit from the simplified expression and connect its output as serial input of shift register.

Example 3.4 : *Design and implement the following sequence generator using shift register.*

------ 1010 ------

Solution :

- The length of the given sequence is two. Therefore minimum number of flip flops N can be obtained from equation.

  $$L \leq 2^N - 1$$

  For N = 1; $2 \leq 2^1 - 1$

  The inequality is not satisfied

  For N = 2; $2 \leq 2^2 - 1$

  The inequality is satisfied.

  Therefore number of flip flops required = N= 2

- Now we prepare the state table 3.29.

Table 3.29

Q_1	Q_0	State of the Circuit
1	0	2
0	1	1

- The desired sequence is written under the output Q_1 (MSB) & is delayed by one clock pulse for Q_0 (LSB). As shown the states of the circuit are distinct, therefore the given sequence generator can be implemented using two flip flops.
- Now we prepare another table 3.30 showing the state of the circuit and output Y of the combinational circuit.

Table 3.30

State of Circuit	Q_1	Q_0	Y
2	1	0	0
1	0	1	1
2	1	0	0

- As shown in Fig. 3.57 the output Y of the combinational circuit is connected as the serial input of the shift register. In this example, output Y is to be connected to D1 input of FF1. The output Y applied at D1 will become Q_1 after application of the clock pulse. Therefore whatever state of Q_1 is required during the next clock cycle, the same must be generated by the combinational circuit during the present clock cycle.
- Based on this, the column for output Y is completed. In the second row of table 3.30, $Q_1 = 0$ therefore, in the first row of column Y, 0 is written as indicated with arrow. Similarly, in the second row of column Y, 1 is written.
- Now we prepare a K-map for Y in terms of Q_1 & Q_0.

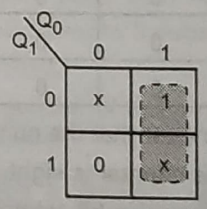


Fig. 3.57

$\therefore \quad Y = Q_0$

- As other two combinations of Q_1, Q_0 such as 00 & 11 do not occur, 'X' is written in the corresponding cell of K-map.
- Based on the above expression and the block diagram shown in the Fig. 3.57, the circuit diagram is drawn as shown in Fig. 3.58.

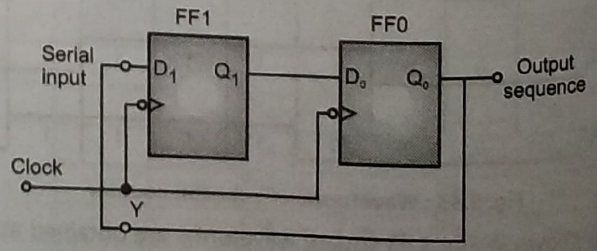


Fig. 3.58

- As shown in Fig. 3.58, as $Y = Q_0$, combinational circuit is a simple connection between output Q_0 and serial input D_1. The desired output sequence is available at both outputs Q_1 and Q_0 with the application of successive clock pulses.

Example 3.5 : *Design sequence generator using Shift register to generate sequence 1101.*

Solution :

- The length of the given sequence is four. Therefore minimum number of flip flops N can be obtained from equation.

$$L \leq 2^N - 1$$

- Putting N = 2,

$$4 \leq 2^2 - 1$$

The inequality is not satisfied.

Putting N = 3, $4 \leq 2^3 - 1$

The inequality is satisfied

Therefore, number of flip flops required = N = 3.

- Now we prepare the state table 3.31.

Table 3.31

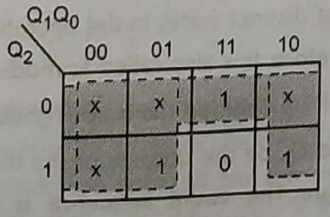
Q_2	Q_1	Q_0	State of the Circuit
1	1	0	6
1	1	1	7
0	1	1	3
1	0	1	5

- The desired sequence is written under the output Q_2 (MSB) and is delayed by one clock pulse upto Q_0 (LSB). As shown, the states of the circuit are distinct, therefore the given sequence generator can be implemented using three flip-flops.
- Now we prepare table 3.32 shows the state of the circuit and output Y of the combinational circuit.

Table 3.32

State of Circuit	Q_2	Q_1	Q_0	Y
6	1	1	0	1
7	1	1	1	0
3	0	1	1	1
5	1	0	1	1

- From table 3.32, the k-map for output Y is prepared.



$\therefore \quad Y = \bar{Q}_2 + \bar{Q}_1 + \bar{Q}_0$

- Based on the above expression and the block diagram shown in Fig. 3.58, the circuit diagram is drawn as shown in Fig. 3.59.
- As shown in Fig. 3.59, the combinational circuit consists of 3 input OR gate and its output Y is connected to the serial input D_2 of the shift register. The desired output sequence is available at any of the outputs Q_2, Q_1 and Q_0 with the application of successive clock pulses.

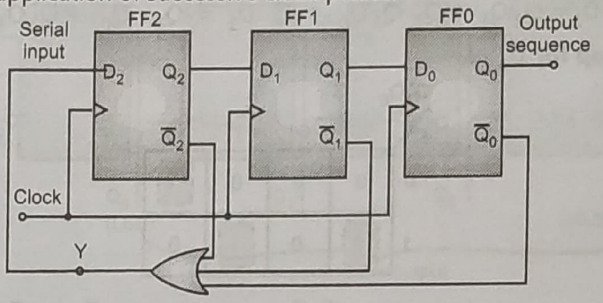


Fig. 3.59

3.20.2 Sequence Generator Using Flip-Flops

- The sequence generator using flip flops is same as that of design of a synchronous counter.
- The sequence generator using flips flops can be designed to avoid the lock out condition by assigning the next state as used state for every unused state.

Example 3.6 : *Design the circuit to generate the sequence :*

$$0 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 7 \rightarrow 3.$$

Solution :

- From the given sequence it is clear that three flip flops are required for the implementation of the circuit. We shall use D flip flop for the design of the circuit.
- First the table 3.33 consisting of present state of the circuit, next state of the circuit and flip flop input is prepared using the excitation table of D flip flop as shown

Table 3.33

Present States			Next States			Flip Flop Inputs		
Q_1	Q_2	Q_0	Q'_1	Q'_2	Q'_0	D_1	D_2	D_0
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	1
0	1	1	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1
1	0	1	1	0	0	1	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	1	1	0	1	1

- We now prepare the K-maps for individual input of flip flops and obtain the simplified expressions.

k - map for $J_3 \Rightarrow$

Q_2 \ Q_1Q_0	00	01	11	10
0	0	0	0	1
1	1	1	0	0

$\therefore D_1 = Q_2\bar{Q}_1 + \bar{Q}_2 Q_1 \bar{Q}_0$

k - map for $K_3 \Rightarrow$

Q_2 \ Q_1Q_0	00	01	11	10
0	1	0	0	0
1	1	0	1	0

$\therefore D_1 = \bar{Q}_1 \cdot \bar{Q}_0 + Q_2 Q_1 Q_0$

K-map for $D_0 \Rightarrow$

Q_2 \ Q_1Q_0	00	01	11	10
0	0	0	0	1
1	1	0	1	0

$\therefore D_0 = Q_2\bar{Q}_1\bar{Q}_0 + Q_2 Q_1 Q_0 + \bar{Q}_2 Q_1 \bar{Q}_0$

- Based on the above expressions the circuit diagram can be constructed as shown in Fig. 3.60.

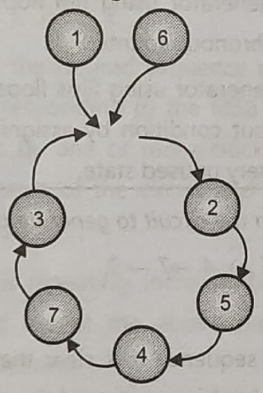


Fig. 3.60 (a) : Bush diagram

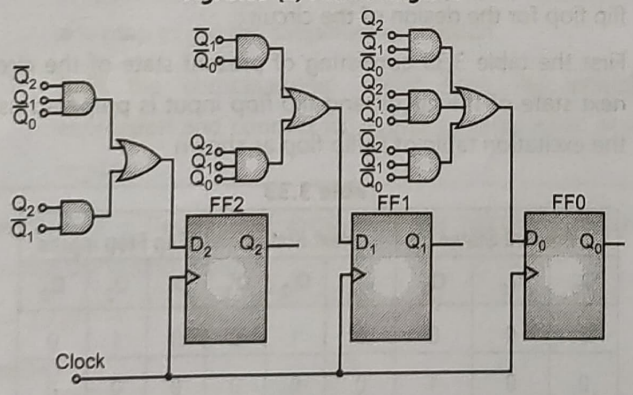


Fig. 3.60 (b)

3.21 ASYNCHRONOUS (RIPPLE) COUNTERS

- A two-bit asynchronous counter is shown in Fig. 3.61 (a). The external clock is connected to the clock input of the first flip-flop (FF_0) only. So, FF_0 changes state at the falling edge of each clock pulse, but FF_1 changes only when triggered by the falling edge of the Q output of FF_0.
- Because of the inherent propagation delay through a flip-flop, the transition of the input clock pulse and a transition of the Q output of FF0 can never occur at exactly the same time. Therefore, the flip-flops cannot be triggered simultaneously, producing an asynchronous operation.

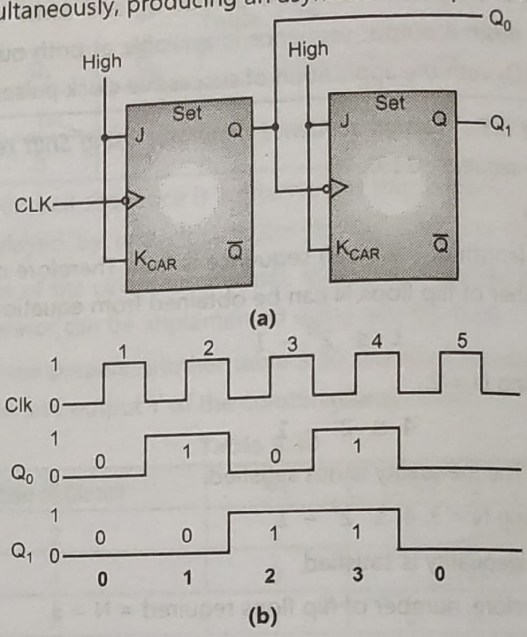


Fig. 3.61

- Note that for simplicity, the transitions of Q_0, Q_1 and CLK in the timing diagram above are shown as simultaneous even though this is an asynchronous counter. Actually, there is some small delay between the CLK, Q_0 and Q_1 transitions.
- Usually, all the CLEAR inputs are connected together, so that a single pulse can clear all the flip-flops before counting starts. The clock pulse fed into FF_0 is rippled through the other counters after propagation delays, like a ripple on water, hence the name Ripple Counter.
- The 2-bit ripple counter circuit above has four different states, each one corresponding to a count value. Similarly, a counter with n flip-flops can have 2 to the power n states. The number of states in a counter is known as its mod (modulo) number. Thus a 2-bit counter is a mod-4 counter.
- A mod-n counter may also described as a divide-by-n counter. This is because the most significant flip-flop (the furthest flip-flop from the original clock pulse) produces one pulse for every n pulses at the clock input of the least significant flip-flop (the one triggers by the clock pulse).

3.21.1 3-Bit Asynchronous Up (Ripple) Counter

- For the implementation of 3-bit counter, three flip flops are required.
- The number of distinct states in the operation of this counter is $2^3 = 8$. Therefore, it is also called as mod-8 counter.
- In case of 3-bit up counter, the output goes from 0 to 7.
- Let Q_2, Q_1 and Q_0 be the outputs of the three flip flops used for the design. The count sequence is as shown in the Table 3.34.

Table 3.34 : Count Sequence of 3-Bit up Counter

Q_2	Q_1	Q_0	State of the Counter
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

- From the Table 3.34 it is clear that the output Q_0 of the least significant flip flop changes for every clock pulse applied to it. So, it can be implemented using a T type flip flop with $T_0 = 1$.
- Also, the output Q_1 changes from 0 to 1 or 1 to 0, only when the corresponding states Q_0 changes from 0 to 1. So it can be implemented using a T-type flip flop with $T_1 = 1$ and Q_0 is connected as its clock input.
- Similarly, the output Q_2 changes only when Q_1 changes from 0 to 1. So, it can be implemented using a T-type flip flop with $T_2 = 1$ and Q_1 is connected as its clock input. This completes the design and the resulting circuit is as shown as Fig. 3.62 (a).

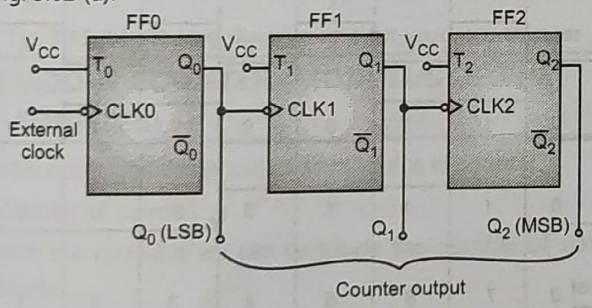


Fig. 3.62 (a) : 3-Bit ripple up counter

- The waveforms of the outputs are shown in Fig. 3.62 (b).

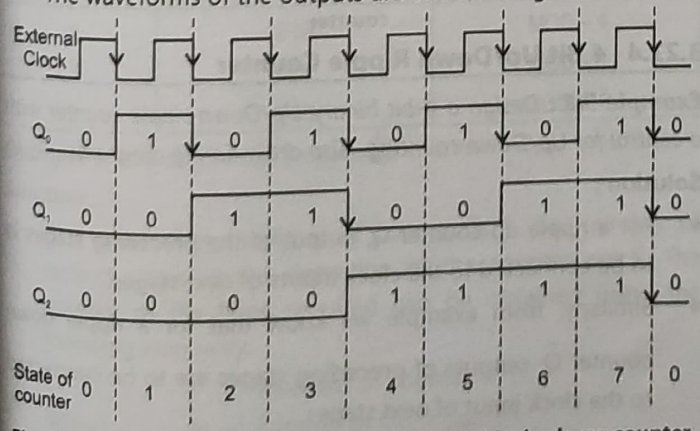


Fig. 3.62 (b) : Waveforms/Timing diagram of 3 bit ripple up counter

- From the waveform of external clock and Q_0, it is clear that two clock periods are required for the completion of one cycle of Q_0. Therefore, clock frequency is twice to that of Q_0 output. In other words, Q_0 is the divide by 2 (÷ 2) output with respect to clock frequency. Similarly, Q_1 is the divide by 4 (÷ 4) output and Q_2 is divide by 8 (÷ 8) output with respect to clock frequency.
- Therefore, this mod - 8 counter is also known as divide by 8 (÷ 8) counter.

3.21.2 4-Bit Asynchronous Up Counter

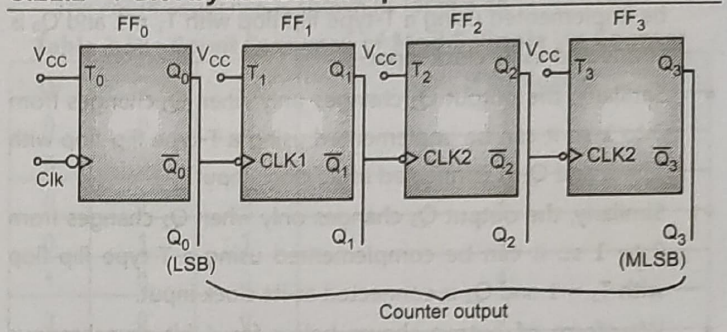


Fig. 3.63 (a) : 4-Bit (asynchronous ripple) counter

Fig. 3.63 shows the circuit diagram of 4-bit asynchronous counter using the T flip-flops.

- Since it is 4 bit asynchronous up counter, we need to use four flip-flops number of distinct states in the operation of this counter is $2^4 = 16$.

  Therefore, it is called as mod-16 counter.
- Let Q_0, Q_1, Q_2 and Q_3 be the outputs of the three flip-flops used for the design. The count sequence is as shown in the Table.
- Table 3.35 count sequence of 4-bit up counter.

Table 3.35

Q_3	Q_2	Q_1	Q_0	State of the Counter
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

- From the table it is clear that the output Q_0 of the least significant flip-flop changes for every clock pulse applied to it. So, it can be implemented using a T type flip-flop with $T_0 = 1$.
- Also, the output Q_1 changes from 0 to 1 or 1 to 0, only when in the corresponding states Q_0 changes from 0 to 1. So, it can be implemented using a T-type flip flop with $T_1 = 1$ and Q_0 is connected as its clock input.
- Similarly, the output Q_2 changes only when Q_1 changes from 0 to 1 so it can be implemented using a T-type flip-flop with $T_2 = 1$ and Q_1 is connected as its clock input.
- Similarly, the output Q_3 changes only when Q_2 changes from 0 to 1 so it can be complemented using a T type flip-flop with $T_3 = 1$ and Q_2 is connected as its clock input.
- Waveform of output shown below for 4 bit asynchronous (ripple) counter.

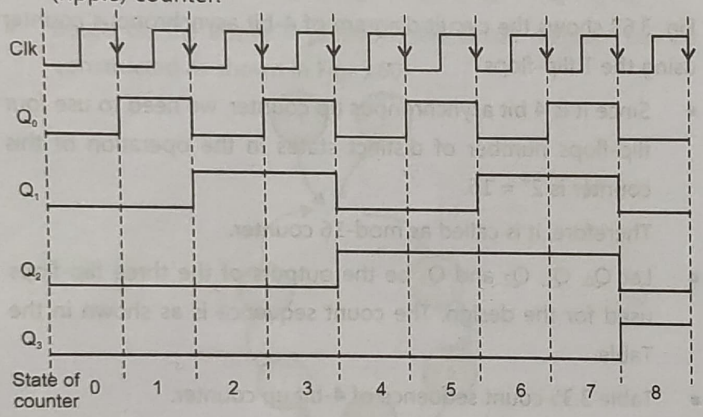


Fig. 3.63 (b)

3.21.3 3-Bit Asynchronous Down Counter

Example 3.7 : *Design 3 bit down ripple counter. Draw waveforms.*

Solution :

- 3 bit ripple down counter requires three flip flops. The counter output goes from 7 down to 0. The count sequence is as shown in Table 3.36.

Table 3.36 : Count Sequence of 3-bit Down Counter

Q_2	Q_1	Q_0	State of the Counter
1	1	1	7
1	1	0	6
1	0	1	5
1	0	0	4
0	1	1	3
0	1	0	2
0	0	1	1
0	0	0	0

- As like previous example, the least significant stage can be implemented using T flip flop with $T_0 = 1$.
- Output Q_1 changes whenever there is 0 to 1 transition of Q_0, in the corresponding states. So, we can realize it with a flip flop which is positive edge triggered. The Q_0 output needs to be connected to the clock input and $T_1 = 1$.
- When Q_0 makes transition from 0 to 1, $\bar{Q}_0$ changes from 1 to 0. So we can realize the second stage by using a negative edge triggered flip flop as shown in Fig. 3.64 (a) with $T_1 = 1$. $\bar{Q}_0$ output needs to be connected as clock input.
- Similarly, the most significant stage can be realized with a negative edge triggered T flip flop with $T_2 = 1$ and $\bar{Q}_1$ connected as its clock input. This completes the design and the resulting circuit is as shown in Fig. 3.64 (a). Also the waveforms of the outputs are shown in Fig. 3.64 (b).

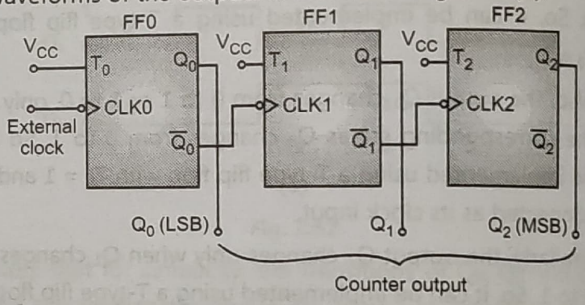


Fig. 3.64 (a) : 3 Bit ripple down counter

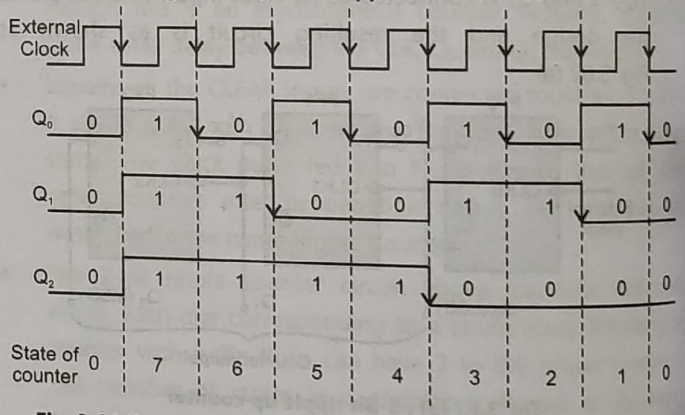


Fig. 3.64 (b) : Waveforms/Timing diagram of 3 bit ripple down counter

3.21.4 4 Bit Up/Down Ripple Counter

Example 3.8 : *Design a 4-bit binary Up/Down ripple counter with a control for Up/Down counting. Also draw timing diagram.*

Solution :

- For a ripple up counter Q, output of the preceding stages is to be connected to the clock inputs of next stages.
- Similarly, from example we know that for a ripple down counter $\bar{Q}$ outputs of preceding stages are to be connected to the clock input of next stages.
- Therefore, to design a Up/Down counter, AND-OR gates are used between flip flops as shown in Fig. 3.65 (a).
- The upper AND gates are enabled when UP/Down input is at logic 1 which connect Q outputs to the inputs. While the

lower AND gates are enabled when UP/Down input is 0 which connect $\bar{Q}$ outputs to the clock inputs. For 4-bit counter, 4 four bit flip flops are required.

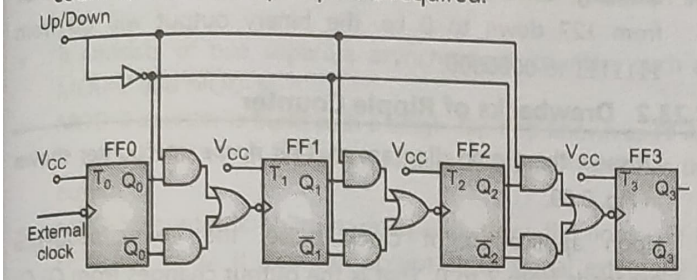


Fig. 3.65 (a) : 4-Bit ripple up/down counter

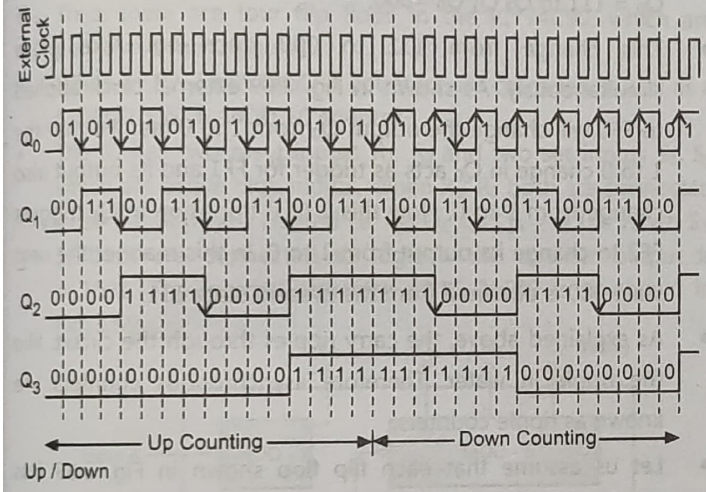


Fig. 3.65 (b) : Timing diagram of 4-bit up/down counter

3.22 MOD-N COUNTER (MODULUS OF THE COUNTER)

- N bit ripple counter is called as modulus N counter.
- Modulus of counter = 2^n
- From the modulus we can conclude the number of states of counter.

Table 3.37

Sr. No.	Counter Type	Modulus
1	2 bit	MOD – 4
2	3 bit	MOD – 8
3	4 bit	MOD - 16

Example 3.9 : *Design mod 5 ripple up counter.*

Solution :

- It is given that modulus of counter is 5. So, number of distinct states in the operation of the counter are 5. The number of flip flops required can be obtained using the following inequality.

$$2^N \geq \text{Modulus of counter}$$

Where N = number of flip flops.

For N = 1 and N = 2 the inequality is not satisfied.

Putting N = 3, we get,

$$2^3 \geq 5$$

$$\Rightarrow \quad 8 \geq 5$$

- The inequality is satisfied. Therefore, for the implementation of mod 5 counter, three flip flops are required.
- As ripple up counter is to be designed, we have to use three T flip flops with the T inputs connected to V_{CC}. Also, Q_0 and Q_1 outputs are to be connected as the clock input of the respective next stages.
- The count sequence is shown in Table 3.38.

Table 3.38 : Count Sequence of Mod 5 Ripple Up Counter

Q_2	Q_1	Q_0	State of the Counter
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
0	0	0	0

- The counter output goes from 0 to 4. Therefore, it is a truncated counter. So, we need to design 'reset logic'.
- From the table it is clear that after state 4 the counter should be reset i.e. state 5 should not occur.

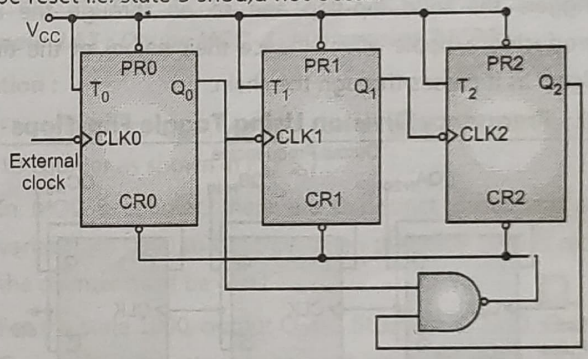


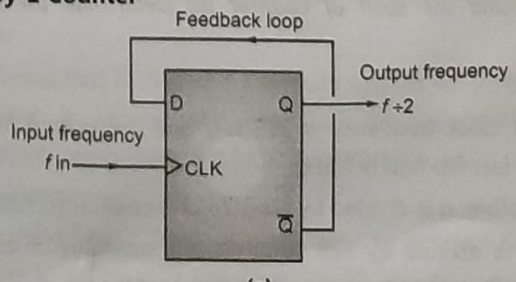
Fig. 3.66 : Mod-5 ripple up counter

- For state 5, Q_2 Q_1 Q_0 = 101. Therefore, whenever both Q_2 and Q_0 become 1, at that instant the counter should be reset. It can be achieved with a simple 2 input NAND gate. The output of the NAND gate must be connected to the clear input of all flip flops, so that the counter gets reset after the state 4. The resulting circuit diagram is as shown in Fig. 3.66.

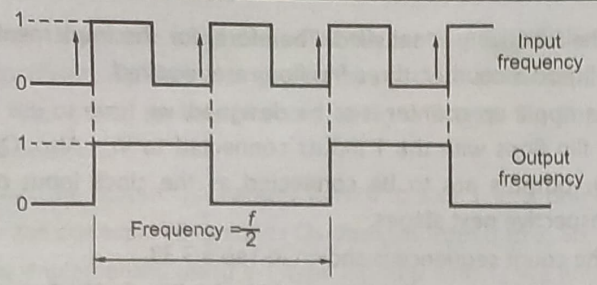
3.23 FREQUENCY DIVISION

- The feature of the D-type flip-flop is as a binary divider, for frequency division or as a "divide-by-2" counter. Here the inverted output terminal Q (NOT-Q) is connected directly back to the Data input terminal D giving the device "feedback" as shown below.

Divide-by-2 Counter



(a)



(b)

Fig. 3.67

- It can be seen from the frequency waveforms above, that by "feeding back" the output from Q to the input terminal D, the output pulses at Q have a frequency that are exactly one half (f ÷ 2) that of the input clock frequency. In other words, the circuit produces frequency division as it now divides the input frequency by a factor of two (an octave).
- This then produces a type of counter called a "ripple counter" and in ripple counters, the clock pulse triggers the first flip-flop whose output triggers the second flip-flop, which in turn triggers the third flip-flop and so on through the chain producing a ripple effect (hence their name) of the timing signal as it passes through the chain.

3.23.1 Frequency Division Using Toggle Flip-Flops

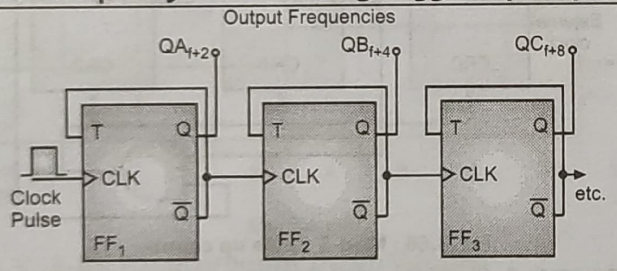


Fig. 3.68

- This type of counter circuit used for frequency division is commonly known as an Asynchronous 3-bit Binary Counter as the output on QA to QC, which is 3 bits wide, is a binary count from 0 to 7 for each clock pulse. In an asynchronous counter, the clock is applied only to the first stage with the output of one flip-flop stage providing the clocking signal for the next flip-flop stage and subsequent stages derive the clock from the previous stage with the clock pulse being halved by each stage.

Example 3.10 : *A certain counter is being pulsed by a 256 kHz clock signal. The output frequency from the last flip flop is 2 kHz : (i) Determine the mod of counter (ii) Determine the counting range.*

Solution :

- Input clock frequency is 256 kHz and the output frequency from last flip flop is 2 kHz.
- Therefore, it is divided by (256/2=128) counter (÷ 128).
- As it is divided by 128 counters, the modulus of counter is also 128. Therefore, it is a mod-128 counter.
- The counting range in case of up counter will be from 0 to 127 i.e. the binary output will go from 0000000 to 1111111.
- Similarly, the counting range in case of down counter will be from 127 down to 0 i.e. the binary output will go from 1111111 to 0000000.

3.23.2 Drawbacks of Ripple Counter

- Observe the timing diagram of 3 bit ripple up counter shown in Fig. 3.69.
- Upon application of clock pulse, the count sequence proceeds from 7 to 0. That is the output changes from $Q_2\ Q_1\ Q_0$ = 111 to $Q_2\ Q_1\ Q_0$ =000.
- This change from 111 to 000 does not take place simultaneously. As shown in Fig. 3.69 external clock applied to FF0 will change the output Q_0 from 1 to 0 first. Further the 1 to 0 change in Q_0 acts as trigger for FF1 and its output also change from 1 to 0. Now this 1 to 0 transition in Q_1 triggers FF2 to change its output from 1 to 0. In this manner the next state 000 is obtained from the present state 111.
- As explained above, the carry ripples through the circuit, like the ripple in water. Therefore asynchronous counters are known as ripple counters.
- Let us assume that each flip flop shown in Fig. 3.69 has propogation delay of 50 nS duration, the output Q_0 will change from 1 to 0. After another 50 nS duration Q_1 will change from 1 to 0, i.e. after 50 + 50 = 100 nS duration since the clock pulse is applied to FF0. Similarly FF2 will take separate 50 ns duration for the change from 1 to 0. So Q_2 will change after 50 + 50 + 50 = 150 nS duration since the clock pulse is applied to FF0.
- This time period will increase as the numbers of flip flops are increased. This limits the frequency of operation of ripple counters.
- In short, 3 bit ripple up counter requires 150 nS duration to change from state 111 to 000 while in case of synchronous counter shown in Fig. 3.69 as all the flip flops are clocked simultaneously, it takes 50 nS duration for the same change. Also this duration is constant for any number of flip flops.
- Thus asynchronous counters are slower than synchronous counters.
- Another drawback of asynchronous counters is that they can generate straight binary sequences in up or down direction while synchronous counters can be designed for any count sequence.
- Also we have to use JK or T flip flops only in the design of asynchronous counters.

3.24 STUDY OF IC 7490

- IC 74C90 is also known as decade counter IC or MOD-10 counter IC as number of distinct states in its operation are ten.
- It consists of two separate asynchronous counters such as MOD-2 and MOD-5.
- MOD-2 counter is build with a single flip flop known as FF A.
- It can accept clock signal externally at input A and has the output.
- MOD-5 counter is built internally with three flip flops - FF B, FF C and FF D. It also can accept clock signal externally at input B and has the outputs Q_B, Q_C, Q_D.
- Thus, there are four flip flops in the IC 74C90, which are negative edge triggered.
- When MOD-2 and MOD-5 counters are cascaded, it acts as a MOD (2 × 5) = MOD 10 counter.
- The IC has two reset inputs R_1, R_2 and two set inputs S_1, S_2 which are active high inputs. When R_1, R_2 both are connected to all flip flops are cleared i.e. Q_D Q_C Q_B Q_A = 0000. When S_1, S_2 both are connected to logic 1 the counter output is = 1001. The internal structure of IC 74C90 is as shown in Fig. 3.69.

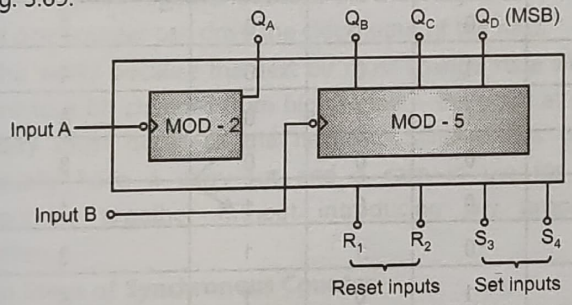


Fig. 3.69 : Internal structure of IC74C90

- As shown in Fig. 3.69 MOD - 2 and MOD - 5 counters can work separately with respect to clock signal applied at input A and input B respective.
- If we connect the output of MOD-2 counter as to input B i.e. as clock signal of MOD-5 counter, then it becomes a MOD-10 counter. The count sequence is given in Table 3.39.

Table 3.39

Flip-Flop Outputs				State of the Counter
Q_D	Q_C	Q_B	Q_A	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

- As the external clock signal is applied at input A, output of MOD 2 counter changes at the negative edge of every clock pulse.
- Further MOD-5 counter changes from 0 to 4 (Q_D Q_C Q_B = 000 to Q_D Q_C Q_B = 100) only when the Q_A output changes from 1 to 0 as shown in Table 3.39.
- MOD-10 counter using IC 74c90 is as shown in Fig. 3.70.

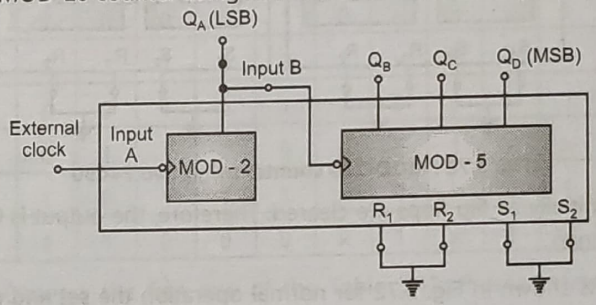


Fig. 3.70 : MOD-10 counter using 74C90

- As shown in Fig. 3.70, for normal operation the set and reset inputs are disabled by connected them to ground.

Example 3.11 : *Design MOD-8 counter using 74C90*

Solution :

- For MOD-8 counter, we first connect the IC 74C90 as MOD-10 counter as shown in Fig. 3.71.
- In MOD-8 counter, there are 8 distinct states and output varies from 0000 to 0111 i.e. when the state 1000 is required the counter must be reset.
- For the state 1000, output Q_D= 1 Study of IC 7490. Therefore, Q_D is to be connected to the reset inputs R_1 and R_2 and
- As soon as the circuit reaches to state 1000, Q_D becomes 1 and immediately the counter is reset to 0000. Fig. 3.71 shows the MOD-8 counter.

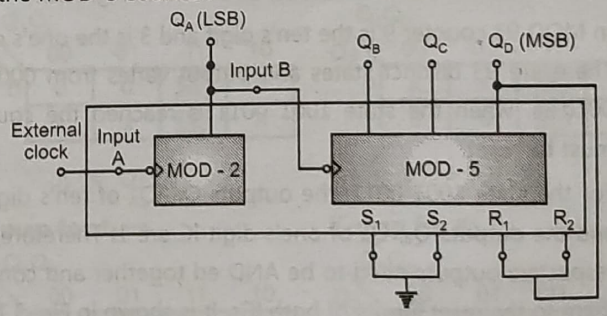


Fig. 3.71 : MOD - 8 counter using IC 74C90

Example 3.12 : *Design divide by 100 counter using ICs 74C90*

Solution :

- We know that IC 74C90 is a decade counter i.e. divide by ten counter, if we cascade two such decade counters, the resultant circuit is MOD-10X10 = MOD 100 counter i.e. divide by 100 counter.
- Cascading of two ICs means we need to connect the output of one IC as input of the other. In this example we carry out

the cascading by connecting Q_D out of first IC as clock input A of second IC. External clock pulses are needed to be applied at input A of the first IC as shown in Fig. 3.72.

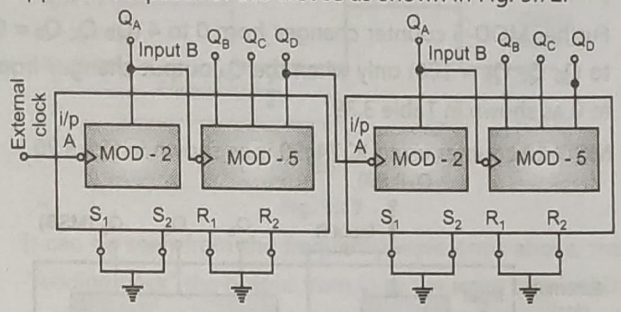


Fig. 3.72 : MOD-100 counter using two 74C90

- Initially all flip flops are cleared. Therefore, the output is 0000 0000.
- As shown in Fig. 3.72 for normal operation the set and reset inputs of both ICs are disable by connecting them to ground.
- An external clock pulses are applied to the first 74C90. Therefore, with the application of successive clock pulses its output goes from 00 to 09 (i.e. from 0000 to 1001).
- When the output of first IC changes from 1001 to 0000, the second IC gets triggered and its output changes from 0000 to 0001 i.e. the decimal output of the circuit becomes 10.
- With the application of clock pulses, it increases upto 19 and like wise upto 99. Then the cycle repeats.
- In the Fig. 3.72, the first IC represents one's digit and the second IC represent ten's digit.

Example 3.13 : *Design divide by 93 counter using ICs 74C90.*

Solution :

- For divide by 93 or MOD-93 counter we first correct the two ICs 74C90 as MOD -100 counter as shown in Fig. 3.73.
- In MOD 93 counter 9 is the ten's digit and 3 is the one's digit. There are 93 distinct states and output varies from 0000 to 0010 ie. when the state 1001 0011 is reached the counter must be reset.
- For the state 1001 0011, the outputs Q_D, Q_A of ten's digit IC and the outputs Q_B, Q_A of one's digit IC are 1. Therefore, the respective outputs need to be AND ed together and connect them to the reset inputs of both ICs. It is shown in Fig. 3.73.

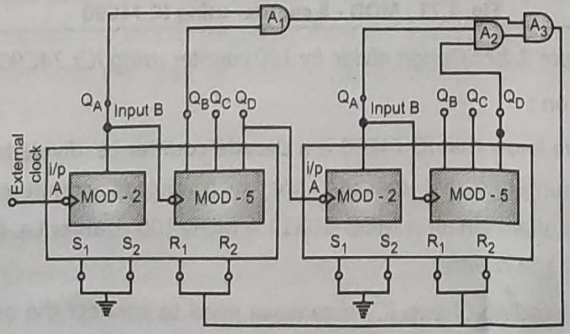


Fig. 3.73 : MOD 93 counter

- As shown in Fig. 3.73, when the circuit reaches to state 1001 00011, the outputs of all AND gates A_1, A_2 ,A_3 become 1 and as outputs of A_3 is connected to reset inputs of both ICs, immediately the counter is reset to 0000 0000.

Example 3.14 : *In 7490, if Q output is connected to input A and pulses are applied at input B, find counter sequence.*

Solution :

- The circuit is as shown in Fig. 3.74.
- When the clock pulses are applied at input B, the output of the MOD-5 counter changes from 0 to 4 (from Q Q Q =000 to Q Q Q =100) with the application of successive pulses.
- As Q output is connected to input A, the output QA will make transition when Q changes from 1 to 0. The count sequence is as shown in Table 3.40.

Table 3.40

Flip-Flop Outputs				State of the Counter
Q_D	Q_C	Q_B	Q_A	
0	0	0	0	0
0	0	1	0	2
0	1	0	0	4
0	1	1	0	6
1	0	0	0	8
0	0	0	1	1
0	0	1	1	3
0	1	0	1	5
0	1	1	1	7
1	0	0	1	9
0	0	0	0	0

- Though the states of counter are not in straight binary form, the number of distinct states is ten, therefore it is also a MOD-10 counter.

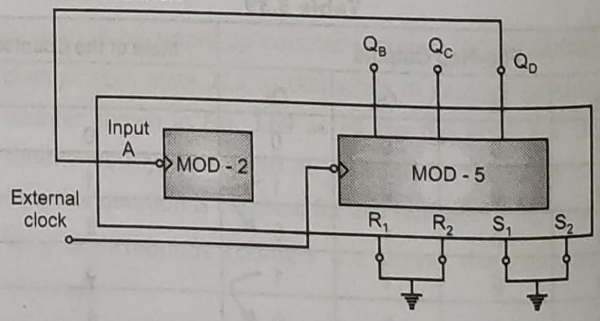


Fig. 3.74

3.25 SYNCHRONOUS COUNTER

- In the Synchronous Counter, the external clock signal is connected to the clock input of every individual flip-flop within the counter so that all the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship.

- In the synchronous counter the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.
- Synchronous Counters use edge-triggered flip-flops that change states on either the "positive-edge" (rising edge) or the "negative-edge" (falling edge) of the clock pulse on the control input resulting in one single count when the clock input changes state.
- Synchronous counters count on the rising-edge which is the low to high transition of the clock signal and asynchronous ripple counters count on the falling-edge which is the high to low transition of the clock signal.

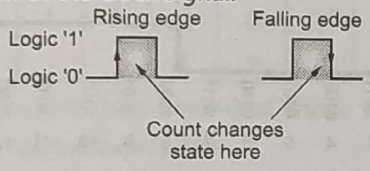


Fig. 3.75

- It may seem unusual that ripple counters use the falling-edge of the clock cycle to change state, but this makes it easier to link counters together because the most significant bit (MSB) of one counter can drive the clock input of the next.
- This works because the next bit must change state when the previous bit changes from high to low – the point at which a carry must occur to the next bit. Synchronous counters usually have a carry-out and a carry-in pin for linking counters together without introducing any propagation delays.

Design Steps of Synchronous Counter

1. In synchronous counter, all the flip flops are clocked simultaneously. Therefore, it is faster in operation.
2. It can be designed for any count sequence which need not be always straight binary.
3. For the design of synchronous counter, first find out the number of flip flops required.
4. Then prepare a table consisting of present state, next state and determine the flip flop inputs which must be present to obtain the next state using the excitation table of the flip flop.
5. Prepare k-map for each flip flop input and obtain the simplified expressions from which complete the circuit diagram.

3.25.1 3-Bit Synchronous Counter

Example 3.15 : *Design 3-bit synchronous counter using JK flip flop and explain.*

Solution :

- We know that for 3 bit counter three flip flops are required.
- The table consisting of present state, next state and the required inputs of flip flop is as shown in Table 3.41. Here Q_2, Q_1 and Q_0 represent the present state variables with Q'_2 Q'_1 and Q'_0 represent the next state variables.

Table 3.41

Present State			Next State			Flip Flip Inputs					
Q_2	Q_1	Q_0	Q'_2	Q'_1	Q'_0	J_2	K_2	J_1	K_1	J_0	K_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

- Observe the first row of the Table 3.41. Present states are Q_2 Q_1 Q_0 = 0 0 0 and next states are Q'_2 Q'_1 Q'_0 = 0 0 1. For flip flop 2, $Q_2 = Q'_2 = 0$. Therefore, from first row of excitation table of JK flip flop we get the J_2 K_2 input combination as 0 X.
- Similarly, for flip flop 1 as $Q_1 = Q'_1 = 0$, the J_1 K_1 input combination is 0 X.
- For flip flop 0, present state = Q_0 = 0 while the next state = Q'_0 =1. Therefore, from the second row of excitation table of JK flip flop we get J_0 K_0 input combination as 1 X.
- This completes the first row of the Table 3.41. Proceeding in a similar manner, the remaining rows of the table are completed.
- Now we represent each individual input in a k map and obtain the simplified expressions.

K – map for J_2 ⇒

Q_2 \ Q_1Q_0	00	01	11	10
0	0	0	1	0
1	x	x	x	x

∴ $J_2 = Q_1 Q_0$

K – map for K_2 ⇒

Q_2 \ Q_1Q_0	00	01	11	10
0	x	x	x	x
1	0	0	1	0

∴ $K_2 = Q_1 Q_0$

K-map for J_1 ⇒

Q_2 \ Q_1Q_0	00	01	11	10
0	0	1	x	x
1	0	1	x	x

∴ $J_1 = Q_0$

K-map for K_1 ⇒

Q_2 \ Q_1Q_0	00	01	11	10
0	x	x	1	0
1	x	x	1	0

∴ $K_1 = Q_0$

K-map for J_0 ⇒

Q_2 \ Q_1Q_0	00	01	11	10
0	1	x	x	1
1	1	x	x	x

∴ $J_0 = 1$

K-map for K_0 ⇒

Q_2 \ Q_1Q_0	00	01	11	10
0	x	1	1	x
1	x	1	1	x

∴ $K_0 = 1$

- Based on above expressions, the resulting circuit diagram is drawn in Fig. 3.76.

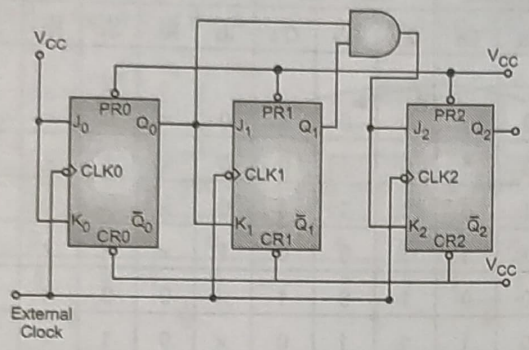


Fig. 3.76 : 3-Bit synchronous up counter

3.25.2 4-Bit Synchronous Counter

- It can be seen that the external clock pulses (pulses to be counted) are fed directly to each J-K flip-flop in the counter chain and that both the J and K inputs are all tied together in toggle mode, but only in the first flip-flop, flip-flop A (LSB) are they connected HIGH, logic "1" allowing the flip-flop to toggle on every clock pulse. Then the synchronous counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.
- The J and K inputs of flip-flop B are connected to the output "Q" of flip-flop A, but the J and K inputs of flip-flops C and D are driven from AND gates which are also supplied with signals from the input and output of the previous stage.

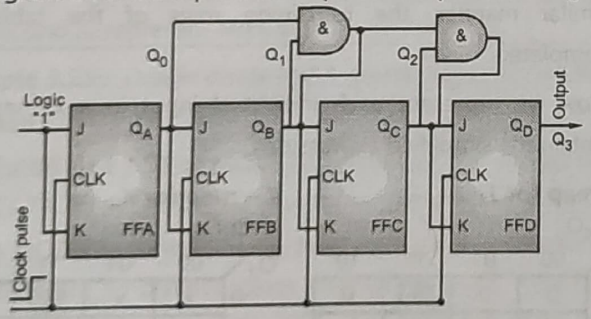


Fig. 3.77 (a)

- If we enable each J-K flip-flop to toggle based on whether or not all preceding flip-flop outputs (Q) are "HIGH" we can obtain the same counting sequence as with the asynchronous circuit but without the ripple effect, since each flip-flop in this circuit will be clocked at exactly the same time.
- Then as there is no inherent propagation delay in synchronous counters, because all the counter stages are triggered in parallel at the same time, the maximum operating frequency of this type of frequency counter is much higher than that for a similar asynchronous counter circuit.
- Because this 4-bit synchronous counter counts sequentially on every clock pulse the resulting outputs count upwards from 0 ("0000") to 15 ("1111"). Therefore, this type of counter is also known as a 4-bit Synchronous Up Counter.

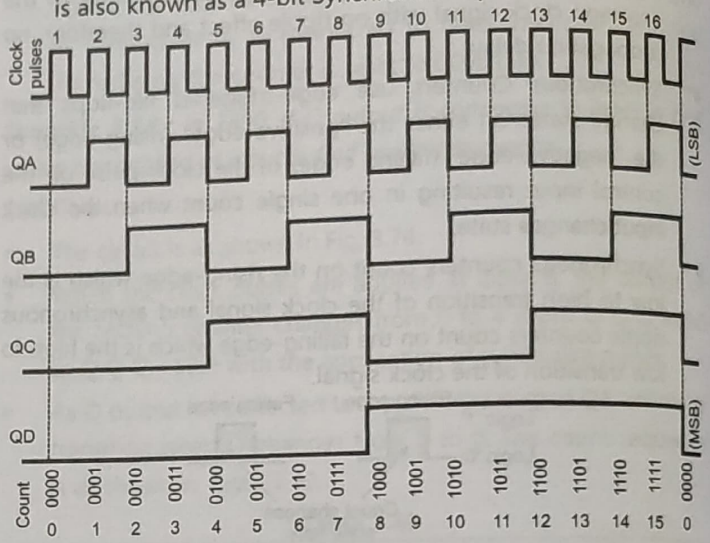


Fig. 3.77 (b) : 4-Bit synchronous counter waveform timing diagram

- As synchronous counters are formed by connecting flip-flops together and any number of flip-flops can be connected or "cascaded" together to form a "divide-by-n" binary counter, the modulo's or "MOD" number still applies as it does for asynchronous counters so a Decade counter or BCD counter with counts from 0 to 2n-1 can be built along with truncated sequences.

3.25.3 3-Bit Up/Down Synchronous Counter

- We know that for a 4-bit counter, four flip-flop are required.
- The table consisting of present state, next state and the required inputs of flip-flop is as shown in table. Here Q_2, Q_1 and Q_0 are present state variable and Q_2', Q_1' and Q_0' represents the next state variables.

Table 3.42 : For Up Counting

Mode Control M	Present States			Next States			Flip-Flop Inputs		
	Q_2	Q_1	Q_0	Q_2'	Q_1'	Q_0'	T_2	T_1	T_0
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1

Table 3.43 : For Down Counting

Mode Control M	Present States			Next States			Flip-Flop Inputs		
	Q_2	Q_1	Q_0	Q_2'	Q_1'	Q_0'	T_2	T_1	T_0
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	1	0	0	0	1

- Now we represent each individual input in a Kmap and obtain the simplified expression.

$$T_2 = \bar{M}Q_1Q_0 + M\bar{Q}_1\bar{Q}_0$$

$$T_1 = \bar{M}Q_0 + M\bar{Q}_0$$

$$T_1 = 1$$

- So logic diagram for 3-bit synchronous up down counter is given below :

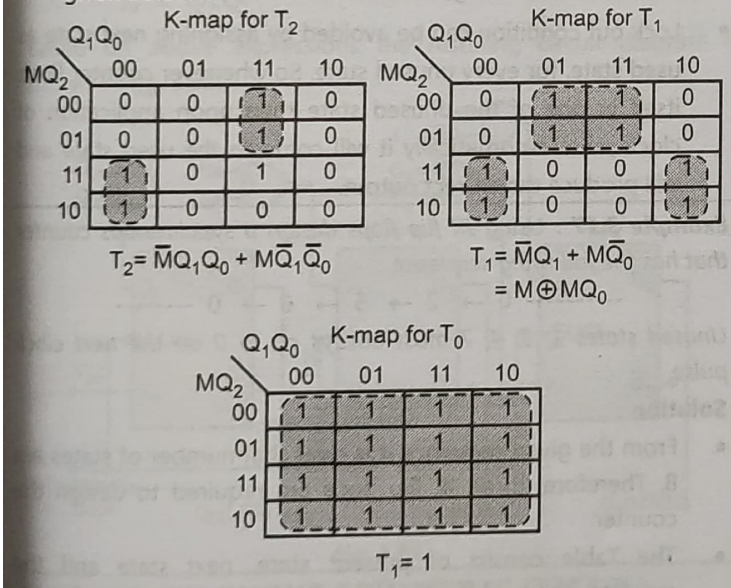


Fig. 3.78 : K-map for 3-bit synchronous up / down counter

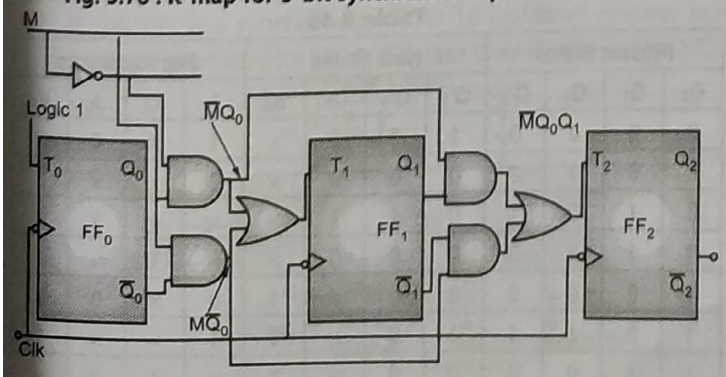


Fig. 3.79 : 3-bit synchronous up / down counter

3.25.4 Modulo N Synchronous Counter

Decade 4-bit Synchronous Counter

- A 4-bit decade synchronous counter can also be built using synchronous binary counters to produce a count sequence from 0 to 9.
- A standard binary counter can be converted to a decade (decimal 10) counter with the aid of some additional logic to implement the desired state sequence.
- After reaching the count of "1001", the counter recycles back to "0000". We now have a decade or Modulo-10 counter.

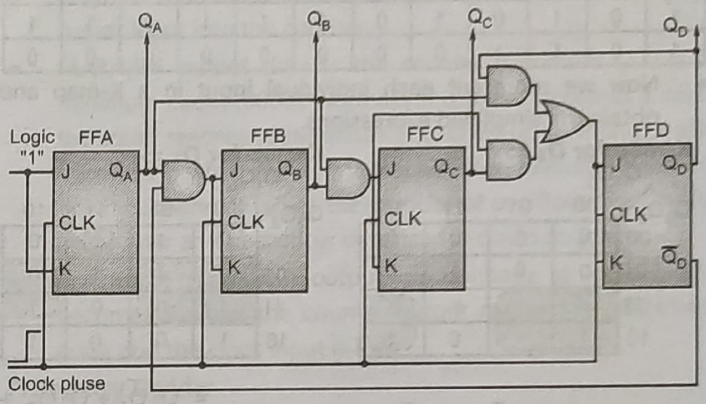


Fig. 3.80 : Decade 4-bit synchronous counter

- The additional AND gates detect when the counting sequence reaches "1001", (Binary 10) and causes flip-flop FF3 to toggle on the next clock pulse. Flip-flop FF0 toggles on every clock pulse. Thus, the count is reset and starts over again at "0000" producing a synchronous decade counter.
- We could quite easily re-arrange the additional AND gates in the above counter circuit to produce other count numbers such as a Mod-12 counter which counts 12 states from"0000" to "1011" (0 to 11) and then repeats making them suitable for clocks, etc.

Example 3.16 : *Design mod-12 synchronous counter using D flip flop.*

Solution :

- For the implementation of MOD-12 counter, four D flip flops are required.
- It is also a truncated counter. 0 to 11 are used states while 12 to 15 are unused states.
- Next state of unused state is unknown. Let it be don't care. Therefore, the corresponding flip flop input is also don't care.
- The Table 3.44 consist of present state, next state and the required input of flip flop.

Example 3.18 : *Design MOD-10 up counter using IC 74C191*

Solution :

- We know that MOD 10 up counter output goes from 0000 to 1001 i.e. as soon as the state 1010 is reaches the counter must be resettled.
- When the counter reaches to 1010 the outputs Q_D, Q_B both becomes 1. They can be connected as inputs of NAND gates. The output of NAND gate is to be connected to active low load input and all the preset inputs need be connected to ground.
- It is as shown in Fig. 3.84.

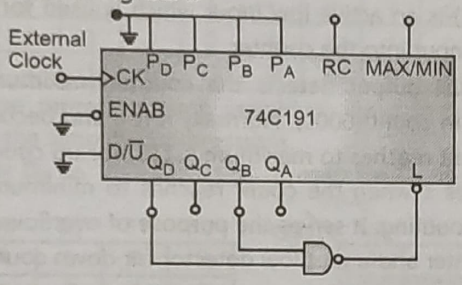


Fig. 3.84

- As shown in Fig. 3.84, the counter will count clock pulses in up direction starting from 0000 to 1001.
- With the next clock pulse counter changes to 1010 and both inputs of NAND gate become 1. Its 0 output at load input cause is the loading of 0000 at present inputs into the counter immediately.

Example 3.19 : *Design a counter to count from 1101 to 0011 using 74C191 IC.*

Solution :

- The counter output goes from 1101 to 0011, therefore it is a down counter and D/ $\bar{U}$ = 1.
- The starting state 1101 must be applied at preset inputs, therefore P_D P_C P_B P_A = 1101.
- After going through the state 0011, the next state 0010 should not occur, for 0010 output Q_B = 1, therefore Q_B needs to be inverted first and then connect it to the load input. It is shown in Fig. 3.85.

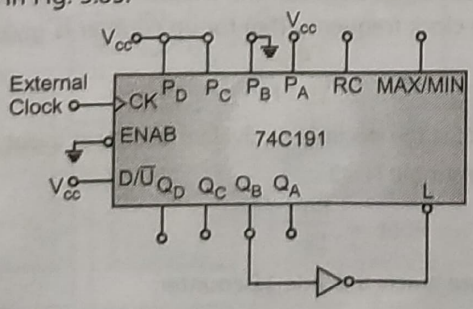


Fig. 3.85

- As shown in Fig. 3.85, the initial state 1101 can be loaded into counter using the active low load input.
- With the application of successive clock pulses, the counter output decrements by one till 0011.
- When in state 0011 and clock pulse is applied, counter enter into state 0010 momentarily, with this Q_B = 1. As Q_B is invested and connected to load input, the binary input 1101 gets loaded into counter immediately.
- Thus, the counter output varies from 1101 to 0011.

Example 3.20 : *Design synchronous counter using D flip flop for the sequence as shown in Fig. 3.86.*

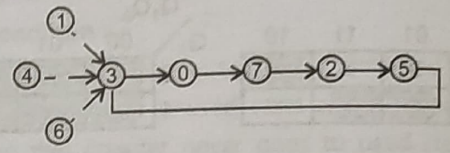


Fig. 3.86

Solution :

- From Fig. 3.86, it is clear that number of states is 8. Therefore, three D flip flops are required to design the counter.
- The Table 3.46 consist of present state, next state and the required input of flip flop.

Table 3.46

Present State			Next State			Flip Flip Input		
Q_2	Q_1	Q_0	Q'_2	Q'_1	Q'_0	D_2	D_1	D_0
0	0	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1
0	1	0	1	0	1	1	0	1
0	1	1	0	0	0	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	0	1	1	0	1	1
1	1	0	0	1	1	0	1	1
1	1	1	0	1	0	0	1	0

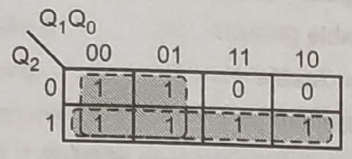
- As shown in Fig. 3.86, for present state 3 next state is 0, similarly for present state 0 next state is 7 etc. Based on this, the Table 3.46 is prepared.
- Also, in this counter, 3, 0, 7, 2 and 5 are called as used states while 1, 4, and 6 are called as 'unused states'.
- For every unused state the next state assigned is 9, used state. Therefore, we can say that this circuit is designed to avoid lock out condition.
- Now we represent each individual input in k-map and obtain the simplified expressions.

K-map for D_2 ⇒

Q_2 \ Q_1Q_0	00	01	11	10
0	1	0	0	1
1	0	0	0	0

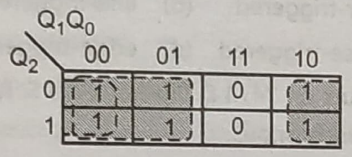
$D2 = \bar{Q}_2 \bar{Q}_0$

K-map for $D_1 \Rightarrow$



$$D1 = \bar{Q}_1 + \bar{Q}_2$$

K-map for $D_0 \Rightarrow$



$$D0 = \bar{Q}_1 + \bar{Q}_0$$

- Based on above expressions, the resulting circuit diagram is shown in Fig. 3.87.

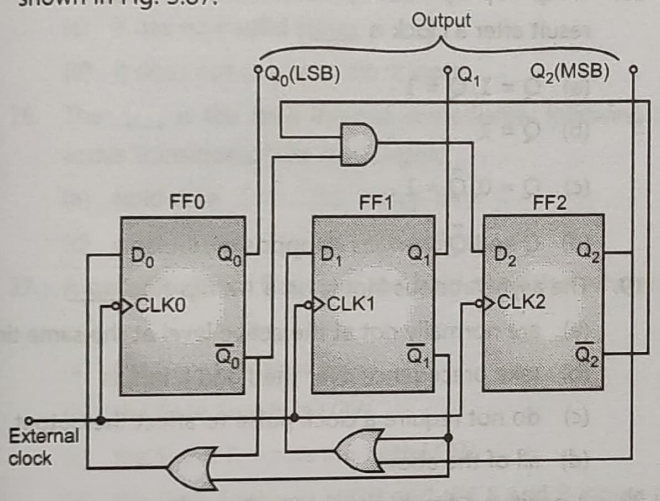


Fig. 3.87

3.27 DIFFERENCE BETWEEN SYNCHRONOUS AND ASYNCHRONOUS COUNTERS

Table 3.47 : Synchronous Vs Asynchronous Counters

Sr. No.	Asynchronous Counters	Synchronous Counters
1.	In this type of counter flip-flops are connected in such a way that output of first flip-flop drives the clock for the next flip-flop.	In this type there is no connection between output of first flip-flop and clock input of the next flip-flip.
2.	Main drawback of these counters is their low speed as the clock is propagated through number of flip-flips before it reaches last flip-flop.	As clock is simultaneously given to all flip-flop there is no problem of propagation delay. Hence, they are high speed counters and are preferred when number of flip-flop increase in the given design.
3.	Logic circuit is very simple even for more number of states.	Design involves complex logic circuit as number of state increases.
4.	As the flip-flops are not clocked simultaneously.	All the flip-flops are clocked simultaneously.

3.28 APPLICATIONS OF COUNTERS

Counters are used as Digital clocks, Frequency counters, Binary counters etc.

System Design Application :

A 3-digit decimal counter (000 – 999)

Fig. 3.88 shows the circuit diagram of BCD output counter. When the count changes from $(1001)_2$ to $(0000)_2$, the 2^3 output line goes from HIGH to LOW and trigger the next counter.

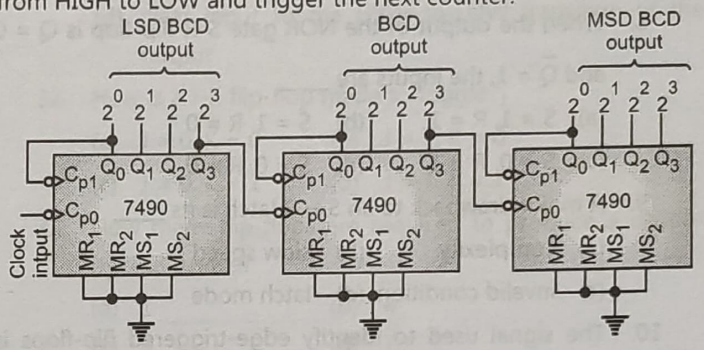


Fig. 3.88 : Cascading 7490s to form a 000-999 BCD output counter

MULTIPLE CHOICE QUESTIONS (MCQ's)

1. Flip flop is not an element of ____ circuit.
   (a) Sequential (b) Combinational
   (c) Both (a) and (b) (d) None of these.
2. Assume a J-K flip-flop has 1 on the J and K inputs. The next clock pulse will cause the output to ____
   (a) set (b) reset
   (c) latch (d) toggle
3. In synchronous systems, the exact times at which any output can change state are determined by a signal commonly called the ____
   (a) Traffic (b) D
   (c) flip-flop (d) clock
4. Assume an S – R latch, made from cross-coupled NAND gates, has a 0 on both inputs. The outputs will be ____
   (a) $Q = 0, \bar{Q} = 0$ (b) $Q = 0, \bar{Q} = 1$
   (c) $Q = 1, \bar{Q} = 0$ (d) $Q = 1, \bar{Q} = 1$
5. Setup time specifies ____
   (a) the minimum time for the control levels to be maintained on the inputs prior to the triggering edge of the clock in order for data to be reliably clocked into the FF
   (b) the maximum time interval required for the control levels to remain on the inputs before the triggering edge of the clock in order for the data to be reliably clocked out of the FF
   (c) how long the operator has in order to get the flip-flop running before the maximum power level is exceeded
   (d) how long it takes the output to change states after the clock has transitioned

39. With regard to a D latch, ____
    (a) the Q output follows the D input when EN is LOW
    (b) the Q output is opposite the D input when EN is LOW
    (c) the Q output follows the D input when EN is HIGH
    (d) the Q output is HIGH regardless of EN's input state

40. How can the cross-coupled NAND flip-flop be made to have active-HIGH S-R inputs?
    (a) It can't be done
    (b) Invert the Q outputs
    (c) Invert the S-R inputs
    (d) None of these

41. When is a flip-flop said to be transparent?
    (a) When the Q output is opposite the input
    (b) When the Q output follows the input
    (c) When you can see through the IC packaging
    (d) None of these

42. Master-slave J-K flip-flops are called pulse-triggered or level-triggered devices because input data is read during the entire time the clock pulse is at a LOW level.
    (a) True (b) False

43. Which of the following is correct for a D latch?
    (a) The output toggles if one of the inputs is held HIGH
    (b) Q output follows the input D when the enable is HIGH
    (c) Only one of the inputs can be HIGH at a time
    (d) The output complement follows the input when enabled

44. A J-K flip-flop is in a "no change" condition when ____
    (a) J = 1, K = 1 (b) J = 1, K = 0
    (c) J = 0, K = 1 (d) J = 0, K = 0

45. A correct output is achieved from a master-slave J-K flip-flop only if its inputs are stable while the ____
    (a) clock is low
    (b) slave is transferring
    (c) flip-flop is reset
    (d) clock is HIGH

46. Which of the following describes the operation of a positive edge-triggered D flip-flop?
    (a) If both inputs are high, the output will toggle
    (b) The output will follow the input on the leading edge of the clock
    (c) When both inputs are LOW, an invalid state exists
    (d) The input is toggled into the flip-flop on the leading edge of the clock and is passed to the output on the trailing edge of the clock

47. What does the triangle on the clock input of a J-K flip-flop mean?
    (a) Level enabled
    (b) Edge-triggered
    (c) Level – triggered
    (d) Edge - enabled

48. A J-K flip-flop with J = 1 and K = 1 has a 20 kHz clock input. The Q output is ____
    (a) constantly low
    (b) constantly high
    (c) a 20 kHz square wave
    (d) a 10 kHz square wave

49. The toggle condition in a master-slave J-K flip-flop means that Q and $\bar{Q}$ will switch to their ____ state(s) at the ____
    (a) opposite, active clock edge
    (b) inverted, positive clock edge
    (c) quiescent, negative clock edge
    (d) reset, synchronous clock edge

50. What is the hold condition of a flip-flop?
    (a) Both S and R inputs activated
    (b) No active S or R input
    (c) Only S is active
    (d) Only R is active

51. A D flip-flop utilizing a positive clock is in the CLEAR state. Which of the following input actions will cause it to change states?
    (a) CLK = negative, D = 0
    (b) CLK = positive, D = 0
    (c) CLOCK negative, D = 1
    (d) CLOCK positive, D = 1

52. Why are the S and R inputs of a gated flip-flop said to be synchronous?
    (a) They must occur with the gate
    (b) They occur independent of the gate
    (c) both (a) and (b)
    (d) None of these

53. Gated S-R flip-flops are called asynchronous because the output responds immediately to input changes.
    (a) True (b) False

54. Which of the following is not generally associated with flip-flops?
    (a) Hold time
    (b) Propagation delay time
    (c) Interval time
    (d) Set up time

55. Edge-triggered flip-flops must have ____
    (a) very fast response times
    (b) at least two inputs to handle rising and falling edges
    (c) positive edge-detection circuits
    (d) negative edge-detection circuits
56. What is one disadvantage of an S-R flip-flop?
    (a) It has no enable input
    (b) It has an invalid state.
    (c) It has no clock input
    (d) It has only a single output
57. To completely load and then unload an 8-bit register requires how many clock pulses?
    (a) 2 (b) 4
    (c) 8 (d) 16
58. What is one disadvantage of an S-R flip-flop?
    (a) It has no enable input
    (b) It has an invalid state
    (c) It has no clock input
    (d) It has only a single output
59. Which of the following best describes the action of pulse-triggered FF's?
    (a) The clock and the S-R inputs must be pulse shaped
    (b) The data is entered on the leading edge of the clock, and transferred out on the trailing edge of the clock
    (c) A pulse on the clock transfers data from input to output
    (d) The synchronous inputs must be pulsed
60. An invalid condition in the operation of an active-HIGH input S-R latch occurs when ____
    (a) HIGHs are applied simultaneously to both inputs S and R
    (b) LOWs are applied simultaneously to both inputs S and R
    (c) a LOW is applied to the S input while a HIGH is applied to the R input
    (d) a HIGH is applied to the S input while a LOW is applied to the R input
61. On a J-K flip-flop, when is the flip-flop in a hold condition?
    (a) J = 0, K = 0 (b) J = 1, K = 0
    (c) J = 0, K = 1 (d) J = 1, K = 1
62. Edge-triggered flip-flops must have ____
    (a) very fast response times
    (b) at least two inputs to handle rising and falling edges
    (c) a pulse transition detector
    (d) active-LOW inputs and complemented outputs
63. Asynchronous inputs will cause the flip-flop to respond immediately with regard to the clock input.
    (a) True (b) False
64. Two J-K flip-flops with their J-K inputs tied HIGH are cascaded to be used as counters. After four input clock pulses, the binary count is ____
    (a) 00 (b) 11
    (c) 01 (d) 10
65. Latches constructed with NOR and NAND gates tend to remain in the latched condition due to which configuration feature?
    (a) cross coupling
    (b) gate impedance
    (c) low input voltages
    (d) asynchronous operation
66. The output of a gated S-R flip-flop changes only if the ____
    (a) flip-flop is set
    (b) control input data has changed
    (c) flip-flop is reset
    (d) input data has no change
67. If both inputs of an S-R flip-flop are low, what will happen when the clock goes HIGH?
    (a) An invalid state will exist
    (b) No change will occur in the output
    (c) The output will toggle
    (d) The output will reset

ANSWERS

1. (b)	2. (d)	3. (d)	4. (d)	5. (a)
6. (b)	7. (a)	8. (c)	9. (c)	10. (d)
11. (d)	12. (d)	13. (d)	14. (b)	15. (c)
16. (d)	17. (d)	18. (d)	19. (d)	20. (d)
21. (a)	22. (b)	23. (b)	24. (c)	25. (c)
26. (a)	27. (d)	28. (b)	29. (b)	30. (a)
31. (a)	32. (d)	33. (d)	34. (b)	35. (b)
36. (b)	37. (c)	38. (a)	39. (c)	40. (c)
41. (b)	42. (b)	43. (b)	44. (d)	45. (d)
46. (b)	47. (b)	48. (d)	49. (a)	50. (b)
51. (d)	52. (a)	53. (b)	54. (c)	55. (c)
56. (b)	57. (d)	58. (b)	59. (b)	60. (a)
61. (a)	62. (c)	63. (b)	64. (a)	65. (a)
66. (b)	67. (b)			

EXERCISE

1. Explain sequential circuit?
2. Explain difference between combinational and sequential circuit.
3. Explain difference between Latch and Flip flop.
4. Explain different type of level triggered and edge triggered?
5. Explain what is mean by latch?
6. Explain SR latch using NAND gate?
7. Explain SR latch using NOR gate?
8. What is D latch?
9. Explain clocked SR flip flop?
10. Explain SR flip flop with preset and clear inputs?
11. What is race around condition? Explain with the help of timing diagram how it is removed in basic flip-flop circuit.
12. Compare race and race around condition. How will you avoid race around condition? Explain?
13. Draw neat diagram of JK flip-flop using SR flip-flop. Write the truth table and explain what happens if both the inputs are 1 (J = K = 1).
14. How the race around condition is avoided?
15. What is the advantage of MS JK flip-flop ? Also explain working of MS JK flip-flop.
16. What do you mean by master slave JK flip-flop? Explain the advantages of this flip-flop draw suitable circuit diagram and timing diagram.
17. Explain D flip flop with preset and clear input?
18. Convert SR flip-flop (SR FF) into D FF.
19. What is excitation table of flip flop?
20. Convert SR flip-flop (SR FF) into D FF.
21. Convert JK flip-flop into TFF. Show the truth table.
22. How will you convert JK flip-flop into T-flip-flop. Explain application of T flip-flop in sequential circuit.
23. Convert SR flip-flop into JK flip-flop.
24. Convert SR flip-flop into T-flip-flop.
25. Convert JK flip-flop into T flip-flop.
26. Convert JK flip-flop into D flip-flop.
27. Which IC is used for MSJK flip flops?
28. Which IC is used for D flip flops?
29. Draw and explain the circuit diagram of 4 bit shift register with serial left shift.
30. Explain with a neat diagram working of parallel in serial out 4-bit shift register. Draw necessary timing diagram.
31. How will you design parallel in serial out 4bit register having both shift right and shift left facility.
32. Draw and explain the circuit diagram of 4 bit shift register with the following facility parallel in serial out and reset.
33. How will you design parallel in serial out 4bit register having both shift right and shift left facility.
34. Draw and explain 4 bit bidirectional shift register.
35. Draw the circuit of 4-bit bidirectional shift register.
36. Draw and explain 4-bit shift register having shift and right. Explain any one application of such register.
37. Write short note on Ring counter.
38. Explain ring counter design having initial state 01011 from initial state explain all possible states in the ring.
39. Explain Ring counter design having initial state 10110.
40. Explain the Johnson's counter design for initial state 0110. From initial state explain and draw all possible state.
41. Draw and explain 3-bit asynchronous up-counter. Also draw the necessary timing diagram.
42. Draw 3-bit asynchronous counter. Explain with timing diagram.
43. Draw 4-bit asynchronous counter. Also explain timing diagram for the same.
44. Draw and explain 4-bit binary up counting with this concept. Also draw the necessary timing diagram. Is there any frequency division concept in it? Comment on frequency generated at the output of each flip-flop.
45. What is Mod counter ?
46. What is frequency division?
47. What are drawbacks of ripple counter?
48. Which IC is useful for Mod 10 counter?
49. What is synchronous counter?
50. Design and implement 3-bit up/down synchronous counter using MS-JK flip-flop with its truth table. Also draw timing diagram.
51. Explain with a neat diagram working of 3-bit up-down synchronous counter. Draw necessary timing diagram.
52. Design sequence generator to generate '11010' sequence using MSJK flip flops. How to avoid lockout condition in sequence generator?
53. Explain effect on synchronous design.
54. Explain the conversion logic of one flip-flop to another. Converter J-K flip flop to S-R flip flop.
55. Explain the procedure of conversion of one Flip-flop to another. Convert J-K flip-flop to S-R flip-flop.
56. Design a 3-bit synchronous Up-counter using D flip-flop.

Unit IV

FUNDAMENTALS OF MICROPROCESSORS

4.1 INTRODUCTION

- The microprocessor, also known as the Central Processing Unit (CPU), is the brain of all computers and many household and electronic devices. Multiple microprocessors, working together, are the "hearts" of datacenters, super-computers, communications products, and other digital devices.
- In this unit we are going to learn about the 8086 Architecture : Internal block diagram, ALU, address bus and control bus.
- This unit also covers the details of 8086 of Input /output ports, memory structure, data and program memory, timing diagrams and execution cycles.

4.2 FUNDAMENTALS OF MICROPROCESSOR

- A Microprocessor or processor is the brain of a computer and it performs all the computational tasks, calculations and data processing etc. inside the computer. In computers, the most popular type of the processor is the Intel Pentium chip by Intel Corporation.

Microprocessors can be Classified Based on the Following Features :

- **Instruction Set :** It is a set of the instructions that the Microprocessor can execute.
- **Bandwidth :** The number of bits processed by the processor in a single instruction.
- **Clock Speed :** Clock speed is measured in the MHz and it determines how many instructions a processor can process.
- The speed of a microprocessor is measured in MHz or GHz. The processor is also known as the CPU (Central Processing Unit). It contains the control unit and the arithmetic unit and both of them work together to process the commands. A CPU is used in every computer whether it is a workstation, server or a laptop. A CPU is a complete computational engine that is designed as a chip.
- A CPU is designed to perform the arithmetic and logical operations inside the computer. Common operations inside the computer include adding, subtracting, multiplying, comparing the values and fetching the different numbers to process them. The higher the CPU's clock speed the more efficient will be the performance of the computer.

A Microcomputer System :

- Fig. 4.1 shows a simple microcomputer system with the CPU as its heart, peripheral devices for inputting and outputting data (Like keyboards and monitors), and various memories for storage of data. All these units are interfaced to it through the address, data and control buses.

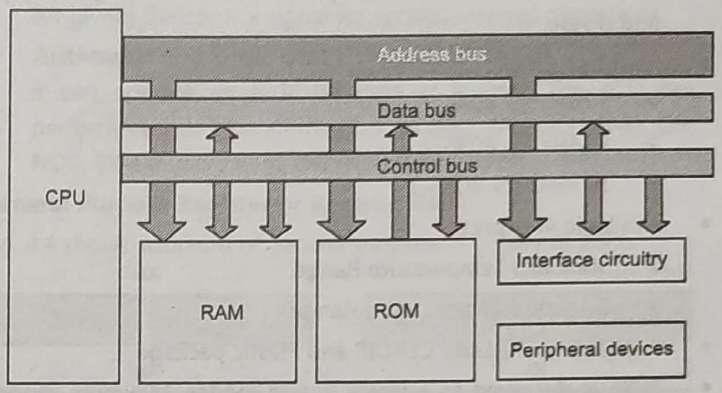


Fig. 4.1 : Simple microprocessor system

4.3 COMPARISON OF 8-BIT (8085), 16-BIT (8086), AND 32-BIT MICROPROCESSORS (80386)

Following table 4.1 shows the comparison of 8-bit (8085), 16-bit (8086) and 32-bit (80386) microprocessor.

Table 4.1 : Comparison of 8-bit, 16-bit and 32-bit Microprocessor

Sr. No.	Parameter	8085 (8-bit)	8086 (16-bit)	80386 (32-bit)
1.	Year of Introduction	1976	1978	1985
2.	Data bus	8-bit	16-bit	32-bit
3.	Address bus	16-bit	20-bit	32-bit
4.	physical memory	64KB	1MB	4GB
5.	Register size	8-bit	16-bit	32-bit
6.	Voltage required	5V	5V	5V
7.	Pipelining	no	yes	yes
8.	Operating modes	No other mode	1. Minimum Mode 2. Maximum mode	1. Real mode, 2. Protected mode, 3. Virtual Mode
9.	Number of flags	5	9	13

4.4 8086 MICROPROCESSOR FEATURES

- The 8086 has direct addressing capability 1M Byte of Memory.
- Its architecture is designed for a powerful assembly language and efficient high level languages.
- It has 14 Word, by 16-Bit Register Set with Symmetrical Operations.

- It has 24 Operand Addressing Modes.
- In 8086 a Bit, Byte, Word, and Block Operations can be implemented.
- The 8086 can perform 8 and 16-bit signed and unsigned arithmetic operations in binary or decimal including multiply and divide.
- Range of Clock Rates :
  - 5 MHz for 8086,
  - 8 MHz for 8086-2,
  - 10 MHz for 8086-1.
- Available in express
  - Standard Temperature Range.
  - Extended Temperature Range.
- Available in 40-Lead CERDIP and Plastic package.
- 8086 is designed to operate in two modes, Minimum mode and Maximum mode.
  - The minimum mode is selected by applying logic 1 to the MN/$\overline{MX}$ input pin. This is a single microprocessor configuration.
  - The maximum mode is selected by applying logic 0 to the MN/$\overline{MX}$ input pin. This is a multiprocessor configuration.

4.5 THE 8086 ARCHITECTURE : INTERNAL BLOCK DIAGRAM OF 8086

- The Intel 8086 is a 16-bit microprocessor that is intended to be used as the CPU in a microcomputer. The term 16-bit means that its arithmetic and logic unit, its internal registers and most of its instructions are designed to work with 16-bit binary words.

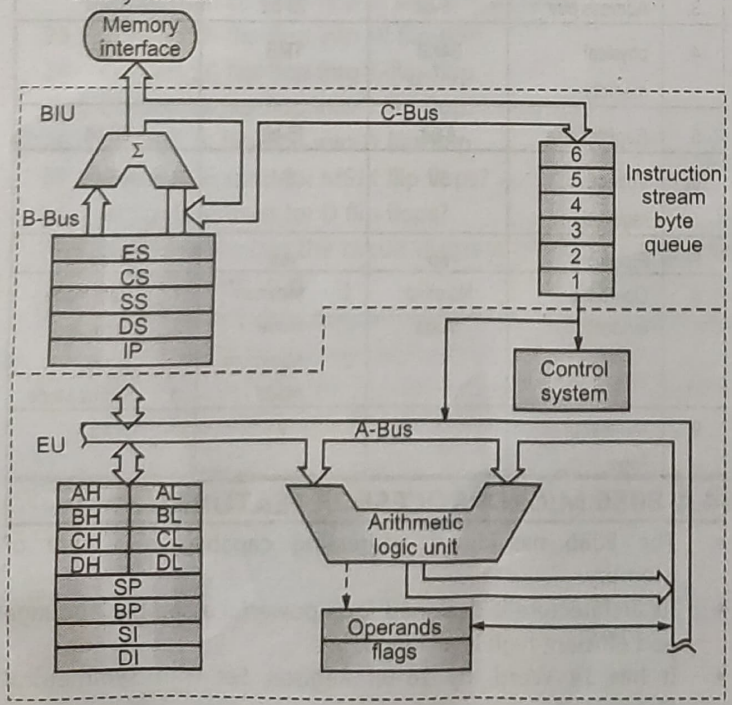


Fig. 4.2 : Block diagram of 8086

- It has 16-bit data bus, so it can read data from or write data to the memory and ports either 16 bits or 8 bits at a time. The 8086 has a 20-bit address bus, so it can address any one of 2^{20}, or 1,048,576 memory locations.
- The internal functions of the 8086 processor are partitioned logically into two processing units.
  1. The Bus Interface Unit (BIU)
  2. Execution Unit (EU)
- As shown in the block diagram of Fig. 4.2 above. These units can interact directly but for most of the part perform as separate units of processors and hence the speed of processing increases.

4.5.1 The Bus Interface Unit (BIU)

The BIU consists of segment registers, instruction pointer, address decoding logic, and 6-byte instruction queue.

The Functions of BIU are :

- The bus interface unit provides the functions related to instruction fetching and queuing, operand such as instruction fetch and store such as
  - Fetch the instruction or data from memory.
  - Write the data to memory.
  - Write the data to the port.
  - Read data from the port.
- It provides the address relocation. That is it takes care of all addresses and data to be placed on the buses.
- This unit also provides the basic bus control.
- The overlap of instruction pre-fetching provided by this unit serves to increase the processor performance through improved bus bandwidth utilization. Up to 6 bytes of the instruction stream can be queued while waiting for decoding and execution.
- As the instruction queuing mechanism is there in the BIU, the memory is utilized very efficiently. Whenever there is space for at least 2 bytes in the queue, the BIU will attempt a word fetch memory cycle. This greatly reduces the "Dead Time" on the memory bus.
- The queue acts as a First-In-First-Out (FIFO) buffer, from which the EU extracts instruction bytes as required.

Instruction Queue

- To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory.
- All six bytes are then held in first out 6 byte register called the instruction queue.
- Then all bytes have to be given to EU one-by-one.
- This pre fetching operation of BIU may be in parallel with the execution operation of EU, which improves the speed execution of the instruction.

4.5.2 Execution Unit (EU)

The EU consists of a register bank, ALU, Flag register and control system.

The Functions of an EU are :

- It receives pre-fetched instructions from the BIU queue and provides un-relocated operand addresses to the BIU.
- Memory operands are passed through the BIU for processing by the EU. Then the EU passes the results to the BIU for storage.

4.5.3 Segment Registers (SFR's) and Program Counter

Additional registers called segment registers generate a memory address when combined with others in the microprocessor. In 8086 microprocessor, the memory is divided into 4 segments as given below. The Fig. 4.3 shows segment registers in 8086 microprocessor

1. **Code Segment (CS) :** The CS register is used for addressing a memory location in the Code Segment of the memory, where the executable program is stored.
2. **Data Segment (DS) :** The DS contains most of the data used by the program. Data are accessed in the Data Segment by an offset address or the content of the other register that holds the offset address.
3. **Stack Segment (SS) :** SS is defined as the area of memory used for the stack.
4. **Extra Segment (ES) :** ES is the additional data segment that is used by some of the strings to hold the destination data.

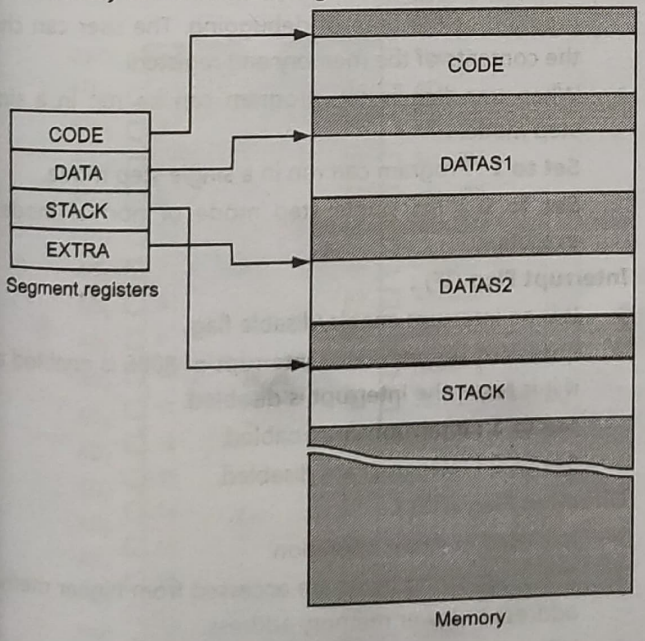


Fig. 4.3

- **Program Counter :** This register holds the address of next instruction to execute. In 8086 the task of program counter is handled by CS and IP.

4.5.4 Control Unit, Arithmetic and Logic Unit and Working Registers

The various units can be understood one-by-one as follows

- **Control Unit :** The control unit is a part of the EU. It is used for giving the control signal for various internal operations.
- **Arithmetic and Logic Unit :** The size of the ALU is 16-bit i.e. it can operate on a 16-bit data at a time. The ALU can perform various operations such as add, subtract, AND, OR, NOR, EX-OR, increment, decrement and so on.

General Purpose Register or Register Bank :

Fig. 4.4 shows structure of general purpose registers of 8086.

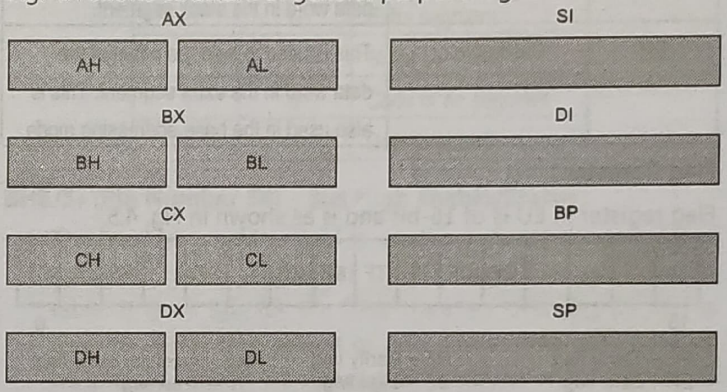


Fig. 4.4 : General purpose registers

- The execution unit consists of eight 8-bit general purpose registers. Each register can be used for the temporary storage of an 8-bit data. The 8-bit registers are AL, AH, BL, BH, CL, CH, DL and DH. Any of these registers can be used as an 8-bit operand also.
- It also consists of eight 16-bit registers which are meant for some special functions as mentioned later. These can also be used for storage of 16-bit data when not used for special functions. The 16-bit registers are AX (AH and AL), BX (BH and BL), CX (CH and CL), DX (DH and DL), SI (source index), DI (destination index), SP (stack pointer), BP (base pointer)

The special functions of all registers are as mentioned below in the table 4.2

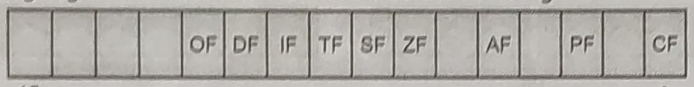
Table 4.2 : Register Functions

Register	Name	Function
AX	Accumulator	It is used for all input and output operations and some arithmetic operations. For example, multiply, divide, translate instruction assume the use of AX.
BX	Base register	This is used as an address register.
CX	Count register	It is used by the instructions which require count. It is used for controlling the number of times a loop is repeated and it is also used in bit-shift instructions for how many times the data is to be shifted.

Register	Name	Function
DX	Data register	It is used together with AX for multiply and divide instructions. It is also used to hold the port number for IN and OUT instructions.
SI	Source Index	This is used to hold the offset of the data word in the data segment.
DI	Destination index	This is used to hold the offset of the data word in the extra segment.
SP	Stack pointer	This is used to hold the offset of the data word in the stack segment.
BP	Base pointer	This is used to hold the offset of the data word in the extra segment. This is also used in the base addressing mode.

Flag Register :

Flag register in EU is of 16-bit and is as shown in Fig. 4.5.



FC – Carry flag PF – Parity flag AF – Auxiliary carry flag
ZF – Zero flag SF – Sign flag TF – Trap flag
If – Interrupt enable flag DF – Direction flag OF – overflow

Fig. 4.5 : Flag register of 8086

Flags Register determines the current state of the processor. They are modified automatically by the CPU after mathematical operations. This allows it to determine the type of the result, and to determine the conditions to transfer control to other parts of the program. 8086 has 9 flags and they are divided into two categories :

I. Conditional Flags.

II. Control Flags.

I. Conditional Flags

Conditional flags represent the result of the last arithmetic or logical instruction executed.

Conditional Flags are as Follows :

- **Carry Flag (CF) :** This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic. This flag is set whenever there is a carry or borrow into the higher order bit of the result i.e. from bit 7 or bit 15.

  Set to 1 : If there is a carry from the MSB.

  Set to 0 : If there is no carry from the MSB.

- **Auxiliary Flag (AF) :** If an operation performed in ALU generates a carry/borrow from the lower nibble (i.e. $D_0 - D_3$) to the upper nibble (i.e. $D_4 - D_7$), the AF flag is set i.e. carry given by D_3 bit to D_4 is AF flag. This is not a general-purpose flag, it is used internally by the processor to perform Binary to BCD conversion.

  Set to 1 : If there is a carry from bit 3 to bit 4.

  Set to 0 : If there is no carry from bit 3 to bit 4.

- **Parity Flag (PF) :** This flag is used to indicate the parity of the result. If the lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity Flag is reset.

  Set to 1 : If result contains even number of 1's.

  Set to 0 : If result contains odd number of 1's.

- **Zero Flag (ZF) :** It is set, if the result of the arithmetic or logical operation is zero else it is reset.

  Set to 1 : If result is zero.

  Set to 0 : If result is not zero.

- **Sign Flag (SF) :** In sign magnitude format the sign of number is indicated by the MSB bit. If the result of the operation is negative, the sign flag is set.

  Set to 1 : If the MSB is 1 (i.e. number is negative).

  Set to 0 : If the MSB is 0 (i.e. number is positive).

- **Overflow Flag (OF) :** It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of the destination location.

  Set to 1 : If overflow occurs.

  Set to 0 : No overflow.

II. Control Flags

Control flags are set or reset deliberately to control the operations of the execution unit.

Control Flags are as Follows :

- **Trap Flag (TP) :**
  - It is used for a single step control.
  - It allows the user to execute one instruction of the program at the time of debugging. The user can check the contents of the memory and registers.
  - When trap flag is set, program can be run in a single step mode.

  Set to 1 : Program can run in a single step mode.

  Set to 0 : No single step mode or normal mode of execution.

- **Interrupt Flag (IF) :**
  - It is an interrupt enable/disable flag.
  - If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.

  Set to 1 : Interrupts are enabled.

  Set to 0 : Interrupts are disabled.

- **Direction Flag (DF) :**
  - It is used in string operation.
  - If it is set, string bytes are accessed from higher memory address to lower memory address.
  - When it is reset, the string bytes are accessed from lower memory address to higher memory address.

  Set to 1 : Increment the pointer.

  Set to 0 : Decrement the pointer.

4.6 PIN DESCRIPTION OF 8086 : (ADDRESS, DATA AND CONTROL BUS)

- The microprocessor 8086 high performance 16-bit CPU is available in three clock rates : 5, 8 and 10 MHz. The CPU is implemented in N-Channel, depletion load, silicon gate technology (HMOS-III), and packaged in a 40-pin CERDIP or plastic package. The 8086 operates in both single processor and multiple processor configurations to achieve high performance levels.
- The pin configuration is as shown in the Fig. 4.6. Some of the pins perform a particular function in minimum mode (single processor mode) and others function in the maximum mode (multiprocessor mode).
- **The 8086 Signals can be Categorized in Three Groups :**
  1. The signals having common functions in minimum and maximum mode.
  2. The signals which have special functions for minimum mode.
  3. The signals which have special functions for maximum mode.

The following signal descriptions are common for both the minimum and maximum modes.

AD_{15}-AD_0 (Pin Number 2-16 and 39) :

- These are time multiplexed memory, I/O address and data lines. When ALE=1 these pins act as address lines and when ALE=0, these act as data lines

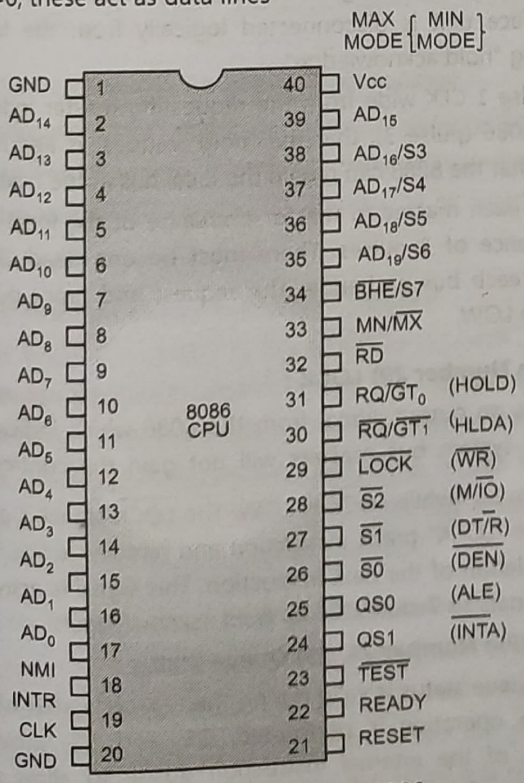


Fig. 4.6 : Pin Diagram for 8086

A_{19}/S_6, A_{18}/S_5, A_{17}/S_4 (Pin no 35-38) Address/Status :

- These are time multiplexed address and status lines. During T_1 these are the four most significant address lines for memory operations. During I/O operations these lines are LOW. During memory and I/O operations, status information is available on these lines during T_2, T_3, T_W, T_4.
- The status of the interrupt enables FLAG bit (S5) which is updated at the beginning of each CLK cycle.
- S_4 and S_3 are encoded as shown table 4.3.

Table 4.3

S_4	S_3	Function
0	0	Extra segment
0	1	Stack segment
1	0	Code or no segment
1	1	Data segment

$\overline{BHE}/S_7$ (Pin Number 34) - Bus High Enable/Status :

- The bus high enable signal is used to indicate the transfer of data over the higher order (D_{15}-D_8) data bus as shown in table. It goes low for the data transfers over D_{15}-D_8 and is used to derive chip selects of odd address memory bank or peripherals. Table 4.4 shows the combination of addresses with combination of BHE and A_0.
- $\overline{BHE}$ is low during T_1 for read, write and interrupt acknowledge cycles, when a byte is to be transferred on the higher byte of the data bus. The status information is available during T_2, T_3 and T_4. The signal is active low and is tri-stated during 'hold'.

Table 4.4 : BHE & A_0 Signal

$\overline{BHE}$	A_0	Indication
0	0	Whole word
0	1	Upper byte from or to odd address
1	0	Lower byte from or to even address
1	1	None

$\overline{RD}$ (Pin Number 32) Read :

- When read signal is low, it indicates to the peripherals that the processor is performing read operation. This signal is used to read devices which reside on the 8086 local bus. This signal floats to 3-state OFF i.e. tristated in "hold acknowledge".

READY (Pin Number 22) :

- This is the acknowledgement from the slow devices or memory that they have completed the data transfer operation. The READY signal from the memory/IO is synchronized by the 8284A Clock Generator to form READY input to the 8086. This signal is active HIGH.

INTR (Pin Number 18) Interrupt Request :

- This is a level triggered input. This is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. If there is any interrupt pending, then the processor enters into the interrupt acknowledge cycle. It can be internally masked by the software resetting the interrupt enable bit.

$\overline{TEST}$ (Pin Number 23) :

- This input is examined by the "Wait" instruction. If the TEST input is LOW then the execution continues, otherwise the processor waits in an "Idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.

NMI (Pin Number 17) Non-Maskable Interrupt :

- This is an edge triggered input which causes a type 2 interrupt. A subroutine is vectored via an interrupt vector lookup table located in the system memory. NMI is not maskable internally by software. A transition from LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.

RESET (Pin Number 21) RESET :

- This signal causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, when RESET returns LOW. RESET is internally synchronized.

CLK (Pin Number 19) CLOCK :

- The clock provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.

VCC (Pin 40) :

- A 5V power supply pin for the operation of the internal circuit.

GND (Pin 1) :

- This is ground for the internal circuit.

MN/$\overline{MX}$ (Pin Number 33) MINIMUM/ $\overline{MAXIMUM}$:

- Indicates what mode the processor is to operate in. The two modes are discussed in the following sections.
- The following pin function descriptions are for the 8086 system in maximum mode (i.e., MN/MX).

$\overline{S_2}$, $\overline{S_0}$ $\overline{S_0}$ (Pin Number 26-28) Status :

- These signal pins indicate the type of operation being performed by the processor. These are active during T_4, T_1, and T_2 and is returned to the passive state (1, 1, 1) during T_3 or during TW when READY is HIGH.
- This status is used by the 8288 Bus Controller to generate all memory and I/O access control signals. Any change by S_2, S_1, or S_0 during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle.

These signal lines are encoded as shown in the table 4.5.

Table 4.5

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Characteristics or Indication
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write memory
1	1	1	Passive

$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$ (Pin no 30, 31) Request/Grant :

- These pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT_0}$ having higher priority than $\overline{RQ}/\overline{GT_1}$. RQ/GT pins have internal pull-up resistors and may be left unconnected.
- A pulse of 1 CLK wide from another local bus master indicates a local bus request (i.e. hold) to the 8086 (pulse 1).
- During a T_4 or T_1 clock cycle, a pulse 1 CLK wide from the 8086 to the requesting master (pulse 2), indicates that the 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge".
- A pulse 1 CLK wide from the requesting master indicates to the 8086 (pulse 3), that the "hold" request is about to end and that the 8086 can regain the local bus at the next CLK.
- Thus each master to master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. The request and grant pulses are active LOW.

$\overline{LOCK}$ (Pin Number 29) LOCK :

- This is an output signal from the 8086 which indicates that other system bus masters will not gain the control of the system bus while $\overline{LOCK}$ is LOW. The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF in "hold acknowledge".

QS_1, QS_0 (Pin Number 24, 25) Queue Status :

- The queue status is valid during the CLK cycle after which the queue operation is performed. QS_1 and QS_0 provide the status of the internal instruction queue to allow external tracking of the queue

- The status of the queue is as given in the table 4.6.

Table 4.6

QS_1	QS_0	Characteristics
0	0	No Operation
0	1	First Byte of Op Code from Queue
1	0	Empty Queue
1	1	Subsequent Byte from Queue

- The following pin function descriptions are for the 8086 in the **Minimum Mode** (i.e., MN/$\overline{MX}$ equal to VCC). Only the pin functions which are unique to the minimum mode are described; all other pin functions are as described above.

M/$\overline{IO}$ (Pin Number 28) Status Line :

- This pin signal is logically equivalent to S_2 in the maximum mode. It is used to distinguish a memory access from an I/O access. When this signal is low, it indicates an I/O operation and when it high it indicates a memory operation. M/$\overline{IO}$ becomes valid in the T_4 preceding a bus cycle and remains valid until the final T_4 of the cycle. M/$\overline{IO}$ floats to 3-state OFF in the local bus "hold acknowledge".

$\overline{WR}$ (Pin Number 29) Write :

- This signal indicates that the processor is performing a memory write or I/O write cycle, depending on the state of the M/$\overline{IO}$ signal. WR is active for T_2, T_3 and T_W of any write cycle. It is active LOW, and floats to 3-state OFF in the local bus "hold acknowledge".

$\overline{INTA}$ (Pin Number 24) Interrupt Acknowledge :

- This signal is used as a read strobe for interrupt acknowledge cycles. That is, when it goes low, the processor has accepted the interrupt. It is active LOW during T_2, T_3 and T_W of each interrupt acknowledge cycle.

ALE (Pin Number 25) Address Latch Enable :

- This signal is provided by the processor to latch the address into the 8282/8283 address latch. This signal indicates the availability of the valid address on the address/data lines and it is connected to the enable input of the latches. It is a HIGH pulse active during T_1 of any bus cycle. ALE is never floated or tristated.

DT/$\overline{R}$ (Pin Number 27) Data Transmit/Receive :

- This signal is needed in the minimum system that desires to use data bus transceiver. It is used to control the direction of data flow through the transceiver. When the processor is sending out the data this is high and when it is receiving the data this signal is low.
- This signal floats to a 3-state OFF in the local bus "hold acknowledge".

$\overline{DEN}$ (Pin Number 26) Data Enable :

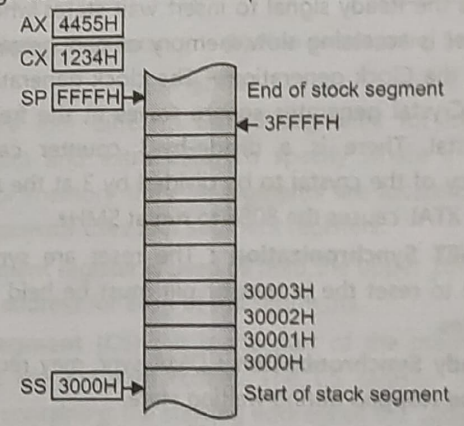
- This signal indicates the availability of the valid data on the address and data lines and is used to enable the transreceivers to separate the data from the multiplexed address and data bus DEN floats to a 3-state OFF in the local bus "hold acknowledge".

HOLD, $\overline{HLDA}$ (Pin Number 31, 30) I/O HOLD and HOLD Acknowledge :

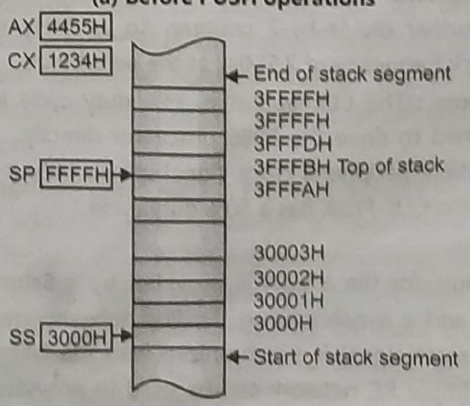
- When the HOLD pin goes low it indicates that another master is requesting a local bus "hold" i.e. local bus access. The processor after receiving the HOLD issues $\overline{HLDA}$ to the requesting master in the middle of the next clock cycle and after the execution of the current cycle. At the same time the processor floats or tristates all the buses and control lines. After HOLD is detected as being LOW, the processor will lower the HLDA.

4.7 STACK AND STACK POINTER

- A stack pointer is a small register that stores the address of the last program request in a stack. A stack is a specialized buffer which stores data from the top down. As new requests come in, they "push down" the older ones.
- The most recently entered request always resides at the top of the stack, and the program always takes requests from the top. Fig. 4.7 shows the stack operation of 8086.



(a) Before PUSH operations



(b) After PUSH AX and PUSH CX operation

Fig. 4.7 : Stack operation

4.8 CLOCK AND RESET CIRCUITS

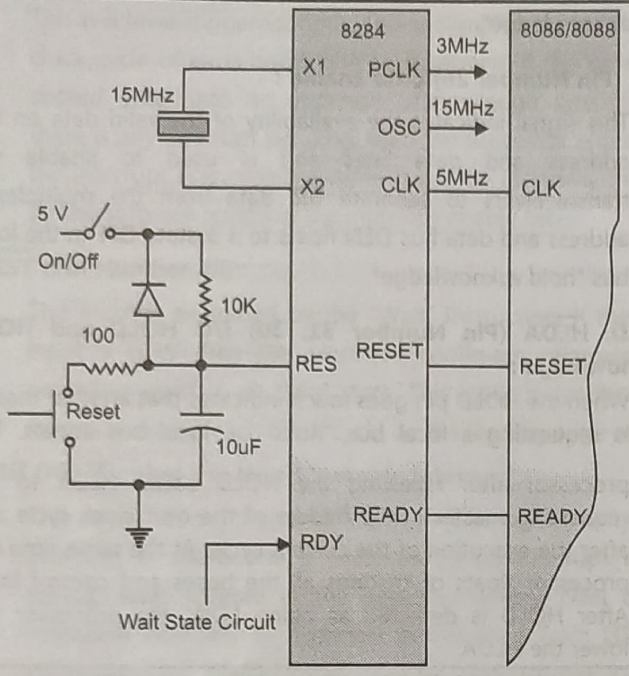


Fig. 4.8 : The clock generation circuit for 8086

The 8284 Chips Serve Three Purposes :

1. Generates the main clock (CLK) for the processor (fc/3 with 33% duty cycle) and the clock for the peripheral devices (fc/5).
2. Provides the Reset pulse according to the state of the RC circuit connected at the RES input.
3. Provides the Ready signal to insert wait states whenever the processor is accessing slow memory or peripheral I/O ports. In 8086 the Clock generation – The clock generation uses a crystal. Crystal generates square waves at the frequency of the crystal. There is a divide-by-3 counter causing the frequency of the crystal to be divided by 3 at the 8086. So a 15 MHz XTAL causes the 8086 to run at 5MHz.
   - **RESET Synchronization :** The reset are synchronized and to reset the processor pin must be held high for 4 cycles.
   - **Ready Synchronization :** DMA sync may require clock to be stopped during waiting state.
   - **Peripheral Clock Signal :** The peripheral frequency has a further divide-by-2 counter. So 15 MHz XTAL gives clock frequency of 2.5MHz at the peripherals.

Clock Outputs : The CLK output is 33% duty cycle MOS clock driver designed to drive the 8086 processor directly. It is a TTL level compatible peripheral clock signal whose output frequency is ½ that of the CLK. PCLK has a 50% duty cycle.

Reset Logic :

The reset logic for the 8284A is provided by a Schmitt trigger input (RES) and a synchronizing flip-flop to generate the reset timing. The reset signal is synchronized with the falling edge of the CLK. A simple RC network can be used to provide power-on reset by utilizing this function of the 8284A.

4.9 I/O PORTS

- The 8086 can be interfaced to 8 -bit and 16-bit I/O devices using either standard or memory-mapped I/O. The standard I/O uses the instructions IN and OUT and is capable of providing 64 K bytes of I/O ports. Using standard I/O, the 8086 can transfer 8- or 16-bit data to or from a peripheral device.
- The 64 K byte I/O locations can then be configured as 64 K 8-bit ports or 32 K 16-bit ports. All I/O transfer between the 8086 and the peripheral devices take place via. AL for 8-bit ports and AX for 16-bit ports.

Interfacing an 8-bit or 16–bit Input Port to the 8086 :

The processor can accept 8-bit or 16-bit data on input port. The interfacing of the 8-bit port and 16-bit port can be as shown in the Fig. 4.9.

- The selection of the device is done by enabling the OE of the buffer connected to the device as shown in Fig. 4.9.
- The output of the buffer is connected to the data bus of the 8086.
- The port here when enabled by the control signals can be read by the input instructions.
- For interfacing the ports the absolute decoding schemes can be used. In this the all remaining address lines can be used for decoding as shown in the Fig. 4.9.
- The control signals used are as shown in the Fig. 4.9.

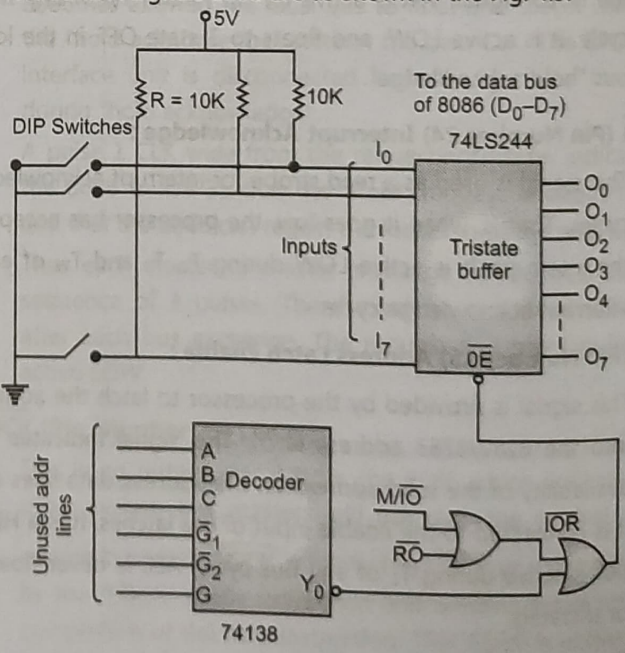


Fig. 4.9

Interfacing an 8-bit or 16–bit Output Port to the 8086 :

The processor can send 8-bit or 16-bit data on input port. The interfacing of the 8-bit port and 16-bit port can be as shown in the Fig. 4.10.

- The decoding logic will be done in such that the latch connected to the respective port will be enabled.

- The output of the buffer is connected to the data bus of the 8086.
- The port here when enabled by the control signals can be read by the input instructions.
- For interfacing the ports, the absolute decoding schemes can be used. In this, all remaining address lines can be used for decoding as shown in the Fig. 4.10.
- The control signals used are as shown in the Fig. 4.10.

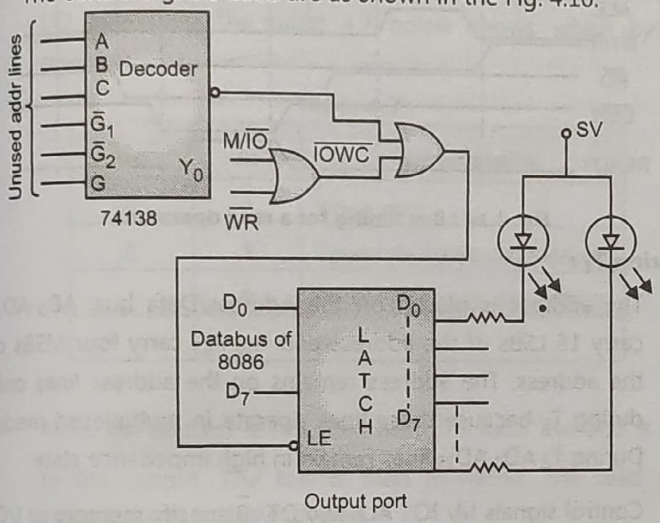


Fig. 4.10

- Depending on this the I/O devices can be interfaced with microprocessor system in two different ways.

4.10 MEMORY STRUCTURES : (PROGRAM MEMORY AND DATA MEMORY)

- Memory is a collection of bytes which are to be organized in such a manner so that they are used efficiently. Segmentation means dividing the processors memory into different memory chunks or partitions. Segmentation provides a powerful memory management mechanism.

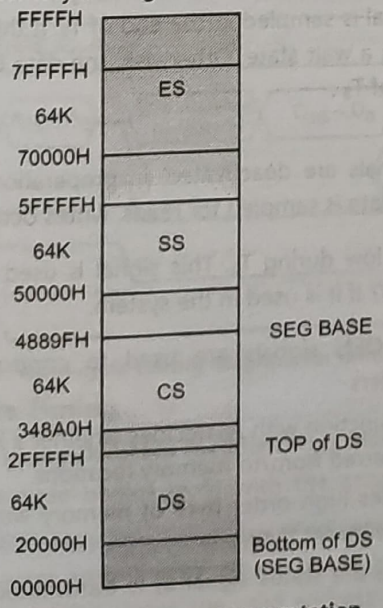
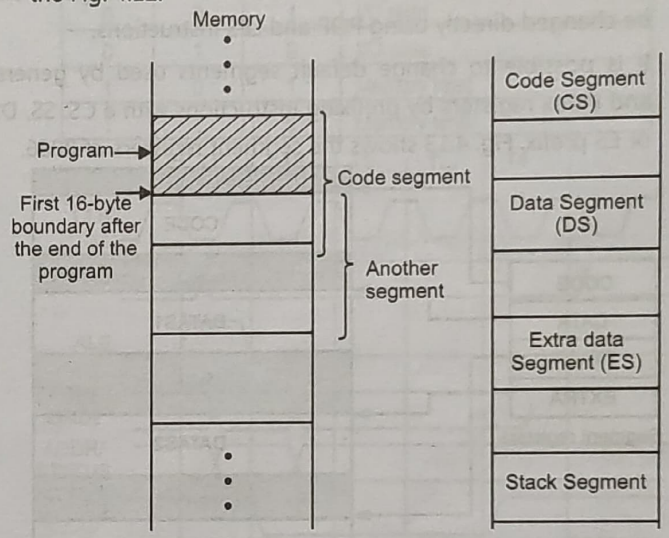


Fig. 4.11 : Memory segmentation

- It allows programmers to partition their programs into modules that operate independently of one another. Segments provide a way to easily implement object-oriented programs. It also allows two processes to easily share data. Fig. 4.11 shows the memory segmentation of 8086.
- In 8086, the memory of 1 M byte is divided into 16-segments each of 64K bytes. The addresses of the segments may be assigned as 0000H to F000H. The offset address value ranges from 0000h to FFFFH. In this case the segments are said to be non-overlapping.
- In some cases, the segments can be overlapping. Both overlapping and non-overlapping segmentation is shown in the Fig. 4.12.



(a) Overlapping Segments **(b) Non-overlapping Segments**

Fig. 4.12

- There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located, the 8086 microprocessor uses four segment registers.
- The segment register is used to hold the upper 16-bits of the starting address for each of the segments.

1. **Code Segment (CS) :** It is the part of the memory which contains the program (code). The CS register is a 16-bit register containing the starting address of the code segment. The processor uses CS segment for all accesses to instructions referenced by the instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.
2. **Stack Segment (SS) :** A stack segment is a memory set aside to store addresses and data while a subprogram executes. The SS register is a 16-bit register containing the starting address of the stack segment. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

3. **Data Segment (DS) :** It is the part of the memory which contains data. The DS register is a 16-bit register containing the starting address of the Data segment. By default, the processor assumes that all data referenced by the general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions.

4. **Extra Segment (ES) :** It is the part of the memory which contains data. The ES register is a 16-bit register containing the starting address of the extra segment. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions.

It is possible to change default segments used by general and index registers by prefixing instructions with a CS, SS, DS or ES prefix. Fig. 4.13 shows the segment registers of 8086.

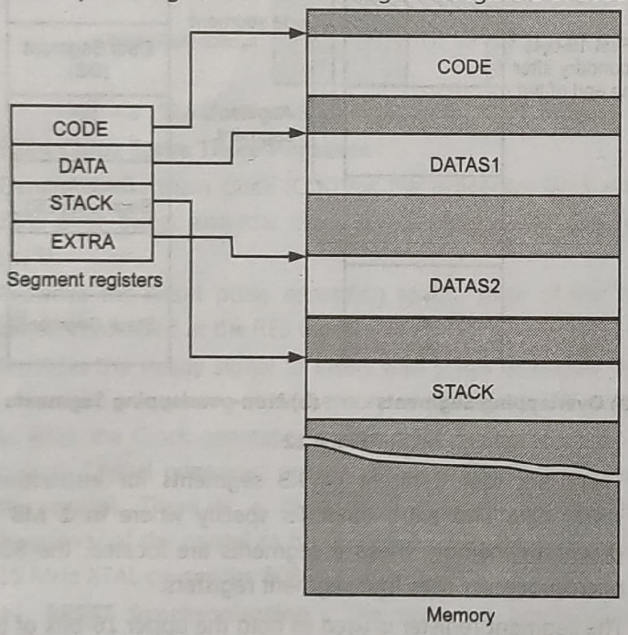


Fig. 4.13 : Segment registers of 8086

Instruction Pointer (IP) :

- The instruction pointer contains the address of the next instruction to be executed in the program.

4.11 TIMING DIAGRAMS AND EXECUTION CYCLES

- The EU executes instruction in certain clock periods which do not constitute any form of machine cycles. The BIU fetches instructions and operands from the memory. Any external access to M/IO device requires 4 clock periods called bus cycle. So, the 8086/8088 microprocessors use the memory and I/O in periods called bus cycles.
- Each bus cycle consists of 4 clock cycles. Thus for 8086 running at 5MHz it would take 800ns for a complete bus cycle. Each read or write operation takes 1 bus cycle as shown in Fig. 4.14.

4.11.1 Read Timing

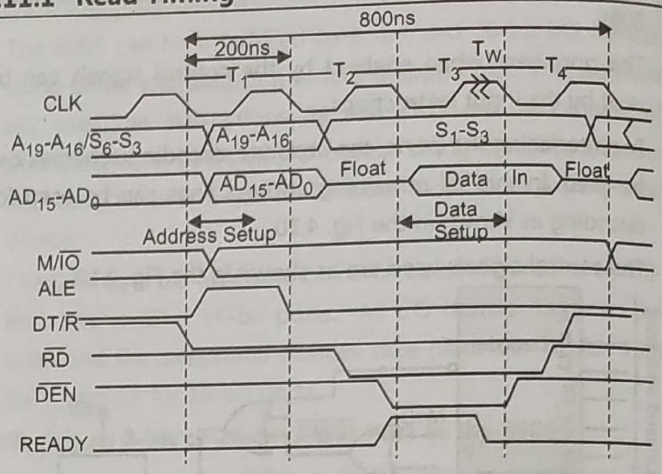


Fig. 4.14 : Bus timing for a read operation

During T_1 :

- The address is placed on the Address/Data bus. AD_0-AD_{15} carry 16 LSBs of the address and A_{16}-A_{19} carry four MSBs of the address. The address remains on the address lines only during T_1 because these lines operate in multiplexed mode. During T_2 AD_0-AD_{15} lines remain in high impedance state
- Control signals M/ IO , ALE and DT/ $\overline{R}$ specify memory or I/O, latch the address onto the address bus and set the direction of data transfer on data bus.

During T_2 :

- 8086 issues the RD or WR signal, DEN, and for a write, the data. The DEN signal enables the memory or I/O device to receive the data for writes and the 8086 to receive the data for reads. During T_2, T_3 and T_4 status signals S_3, S_4, S_5 and S_6 are carried by A_{16}-A_{19} lines.

During T_3 :

- This cycle is provided to allow memory to access data. The READY signal is sampled at the end of T_2. If this signal is low, T_3 becomes a wait state. Otherwise, the data bus is sampled at the end of T_3 .

During T_4 :

- All bus signals are deactivated, in preparation for next bus cycle. The Data is sampled for reads, writes occur for writes.
- DT/$\overline{R}$ goes low during T_1. This signal is used by transceiver 8286 or 8287 if it is used in the system.
- DT/$\overline{R}$ and DEN signals are used to control bidirectional latched buffers.
- BHE in conjunction with A_0 decides whether a byte or word is to be transferred from/to memory locations.
- BHE identifies high order byte of memory word whereas A_0 identifies low order byte.
- On BHE line the status signal S_7 is transmitted during T_2, T_3 and T_4.

1. READ Cycle Timing Diagram for Minimum Mode

- The read cycle begins in T_1 with the assertion of address latch enable (ALE) signal and also M/$\overline{IO}$ signal. During the negative going edge of this signal, the valid address is latched on the local bus.
- The $\overline{BHE}$ and A_0 signals can address the low, high or both bytes. From T_1 to T_4, the M/$\overline{IO}$ signal indicates a memory or I/O operation. The table 4.7 below shows which byte is available.

Table 4.7

$\overline{BHE}$	A_0	Characteristics
0	0	Whole word
0	1	Upper byte from/to odd address
1	0	Lower byte from/to even address
1	1	None

- At T_2, the address is removed from the local bus and is sent to the output. The bus is then tri-stated. The read ($\overline{RD}$) control signal is also activated in T_2.
- The read ($\overline{RD}$) signal causes the address device to enable its data bus drivers. After $\overline{RD}$ goes low, the valid data is available on the data bus.
- The addressed device will drive the READY line high. When the processor returns the read signal to high level, the addressed device will again tri-state its bus drivers.

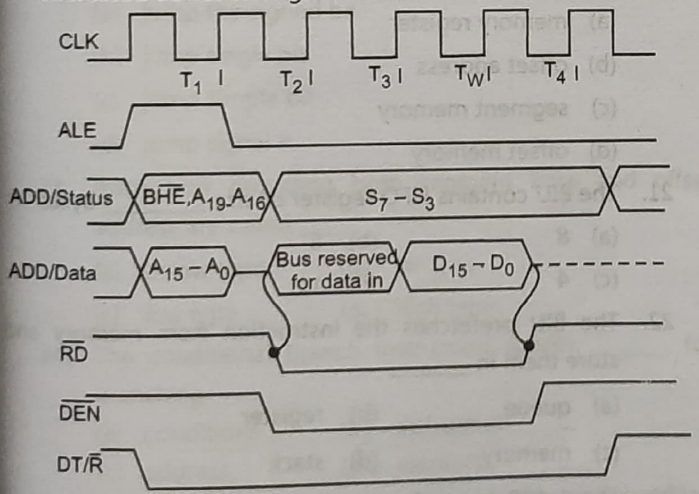


Fig. 4.15 : Read cycle timing diagram for minimum mode

4.11.2 Write Timing

Write Cycle Timing Diagram for Minimum Mode :

- The write cycle begins in T_1 with the assertion of address latch enable (ALE) signal and also M/$\overline{IO}$ signal indicating the memory write or I/O write operation. During the negative going edge of this signal, the valid address is latched on the local bus.
- In T_2, after sending the address in T_1, the processor sends the data to be written to the addressed location.
- The data remains on the bus until the middle of T_4 state. The $\overline{WR}$ becomes active at the beginning of T_2 (unlike $\overline{RD}$ is somewhat delayed in T_2 to provide time for floating).
- The $\overline{BHE}$ and A_0 signals are used to select the proper byte or bytes of memory or I/O word to read or write.
- The M/$\overline{IO}$, $\overline{RD}$ and $\overline{DEN}$ signals indicate the type of data transfer as specified in Table 4.8 as follows.

Table 4.8

M/$\overline{IO}$	$\overline{RD}$	$\overline{DEN}$	Transfer Type
0	0	1	I/O read
0	1	0	I/O write
1	0	1	Memory read
1	1	0	Memory write

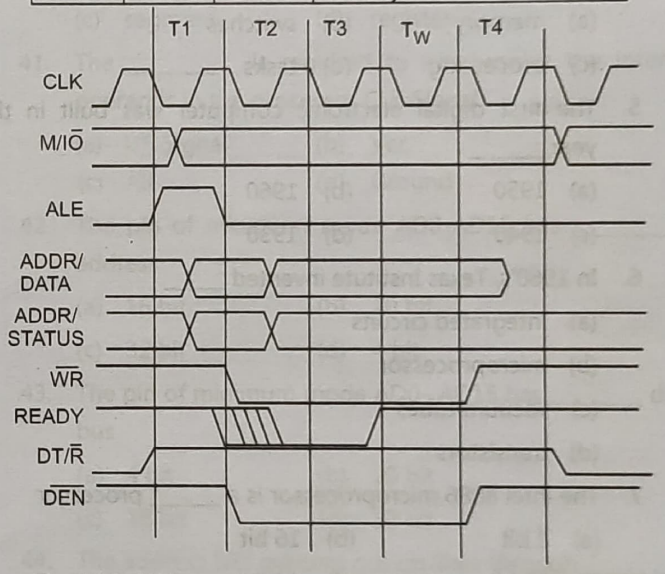


Fig. 4.16 : Write cycle timing diagram for minimum mode

4.11.3 HOLD Response

- The HOLD pin is checked at the leading edge of each clock pulse. If it is received active by the processor before T_4 of the previous cycle or during T_1 state of the current cycle, the CPU activates HLDA in the next clock cycle and for succeeding bus cycles, the bus will be given to another requesting master.
- The control of the bus is not regained by the processor until the requesting master does not drop the HOLD pin low. When the request is dropped by the requesting master, the HLDA is dropped by the processor at the trailing edge of the next clock. As shown in Fig. 4.17.

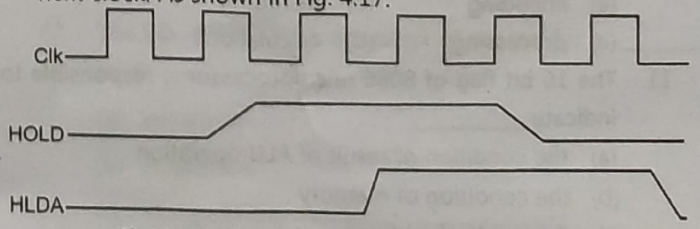


Fig. 4.17 : Bus request and bus grant timings in minimum mode system

MULTIPLE CHOICE QUESTIONS (MCQ's)

1. A microprocessor is a _____ chip integrating all the functions of a CPU of a computer.
   (a) multiple (b) single
   (c) double (d) triple
2. Microprocessor is a/an _____ circuit that functions as the CPU of the compute
   (a) electronic (b) mechanic
   (c) integrating (d) processing
3. Microprocessor is the _____ of the computer and it perform all the computational tasks
   (a) main (b) heart
   (c) important (d) simple
4. The purpose of the microprocessor is to control _____
   (a) memory (b) switches
   (c) processing (d) tasks
5. The first digital electronic computer was built in the year_____
   (a) 1950 (b) 1960
   (c) 1940 (d) 1930
6. In 1960's, Texas Institute invented _____
   (a) integrated circuits
   (b) microprocessor
   (c) vacuum tubes
   (d) transistors
7. The intel 8086 microprocessor is a _____ processor
   (a) 8 bit (b) 16 bit
   (c) 32 bit (d) 4 bit
8. The microprocessor can read/write 16 bit data from or to _____
   (a) memory (b) I /O device
   (c) processor (d) register
9. In 8086 microprocessor, the address bus is _____ bit wide
   (a) 12 bit (b) 10 bit
   (c) 16 bit (d) 20 bit
10. The work of EU is _____
    (a) encoding (b) decoding
    (c) processing (d) calculations
11. The 16 bit flag of 8086 microprocessor is responsible to indicate _____
    (a) the condition of result of ALU operation
    (b) the condition of memory
    (c) the result of addition
    (d) the result of subtraction
12. The CF is known as _____
    (a) carry flag (b) condition flag
    (c) common flag (d) single flag
13. The SF is called as _____
    (a) service flag (b) sign flag
    (c) single flag (d) condition flag
14. The OF is called as _____
    (a) overflow flag (b) overdue flag
    (c) one flag (d) over flag
15. The IF is called as _____
    (a) initial flag (b) indicate flag
    (c) interrupt flag (d) inter flag
16. The register AX is formed by grouping _____
    (a) AH & AL (b) BH & BL
    (c) CH & CL (d) DH & DL
17. The SP is indicated by _____
    (a) single pointer (b) stack pointer
    (c) source pointer (d) destination pointer
18. The BP is indicated by _____
    (a) base pointer (b) binary pointer
    (c) bit pointer (d) digital pointer
19. The SS is called as _____
    (a) single stack (b) stack segment
    (c) sequence stack (d) random stack
20. The index registers are used to hold _____
    (a) memory register
    (b) offset address
    (c) segment memory
    (d) offset memory
21. The BIU contains FIFO register of size _____ bytes
    (a) 8 (b) 6
    (c) 4 (d) 12
22. The BIU prefetches the instruction from memory and store them in _____
    (a) queue (b) register
    (c) memory (d) stack
23. The 1 MB byte of memory can be divided into ____ segment
    (a) 1 Kbyte (b) 64 Kbyte
    (c) 33 Kbyte (d) 34 Kbyte
24. The DS is called as _____
    (a) data segment (b) digital segment
    (c) divide segment (d) decode segment

25. The CS register stores instruction _____ in code segment
(a) stream (b) path
(c) codes (d) stream line

26. The IP is _____ bits in length
(a) 8 bits (b) 4 bits
(c) 16 bits (d) 32 bits

27. The push source copies a word from source to _____
(a) stack (b) memory
(c) register (d) destination

28. LDs copies to consecutive words from memory to register and _____
(a) ES (b) DS
(c) SS (d) CS

29. INC destination increments the content of destination by _____
(a) 1 (b) 2
(c) 30 (d) 41

30. IMUL source is a signed _____
(a) multiplication (b) addition
(c) subtraction (d) division

31. _____destination inverts each bit of destination
(a) NOT (b) NOR
(c) AND (d) OR

32. The JS is called as _____
(a) jump the signed bit
(b) jump single bit
(c) jump simple bit
(d) jump signal it

33. Instruction providing both segment base and offset address are called _____
(a) below type (b) far type
(c) low type (d) high type

34. The conditional branch instruction specify _____ for branching
(a) conditions (b) instruction
(c) address (d) memory

35. The microprocessor determines whether the specified condition exists or not by testing the _____
(a) carry flag (b) conditional flag
(c) common flag (d) sign flag

36. The LES copies to words from memory to register and _____
(a) DS (b) CS
(c) ES (d) DS

37. The _____ translates a byte from one code to another code
(a) XLAT (b) XCHNG
(c) POP (d) PUSH

38. The _____ contains an offset instead of actual address
(a) SP (b) IP
(c) ES (d) SS

39. The 8086 fetches instruction one after another from _____ of memory
(a) code segment (b) IP
(c) ES (d) SS

40. The BIU contains FIFO register of size 6 bytes called _____.
(a) queue (b) stack
(c) segment (d) register

41. The _____ is required to synchronize the internal operands in the processor CLK Signal
(a) UR Signal (b) Vcc
(c) AIE (d) Ground

42. The pin of minimum mode AD0-AD15 has _____ address
(a) 16 bit (b) 20 bit
(c) 32 bit (d) 4 bit

43. The pin of minimum mode AD0- AD15 has _____ data bus
(a) 4 bit (b) 20 bit
(c) 16 bit (d) 32 bit

44. The address bits are sent out on lines through _____
(a) A16-19 (b) A0-17
(c) D0-D17 (d) C0-C17

45. _____ is used to write into memory
(a) RD (b) WR
(c) RD / WR (d) CLK

46. The functions of Pins from 24 to 31 depend on the mode in which _____ is operating
(a) 8085 (b) 8086
(c) 80835 (d) 80845

47. The RD, WR, M/IO is the heart of control for a _____ mode
(a) minimum
(b) maximum
(c) compatibility mode
(d) control mode

48. In a minimum mode there is a _____ on the system bus
(a) single (b) double
(c) multiple (d) triple

49. If MN/MX is low the 8086 operates in _____ mode
(a) Minimum (b) Maximum
(c) both (a) and (b) (d) medium

50. In max mode, control bus signal So, S1 and S2 are sent out in _____ form
(a) decoded (b) encoded
(c) shared (d) unshared

51. The ___ bus controller device decodes the signals to produce the control bus signal
(a) internal (b) data
(c) external (d) address

52. A ____ Instruction at the end of interrupt service program takes the execution back to the interrupted program
(a) forward (b) return
(c) data (d) line

53. The main concerns of the _____ are to define a flexible set of commands
(a) memory interface
(b) peripheral interface
(c) both (a) and (b)
(d) control interface

54. Primary function of memory interfacing is that the _____ should be able to read from and write into register
(a) multiprocessor (b) microprocessor
(c) dual Processor (d) coprocessor

55. To perform any operations, the Mp should identify the _____
(a) register (b) memory
(c) interface (d) system

56. The Microprocessor places _____ address on the address bus
(a) 4 bit (b) 8 bit
(c) 16 bit (d) 32 bit

57. The Microprocessor places 16 bit address on the add lines from that address by ____ register should be selected
(a) address (b) one
(c) two (d) three

58. The _____of the memory chip will identify and select the register for the EPROM
(a) internal decoder (b) external decoder
(c) address decoder (d) data decoder

59. Microprocessor provides signal like ___ to indicate the read operation
(a) LOW (b) MCMW
(c) MCMR (d) MCMWR

60. To interface memory with the microprocessor, connect register the lines of the address bus must be added to address lines of the _____ chip.
(a) single (b) memory
(c) multiple (d) triple

61. The remaining address line of _____ bus is decoded to generate chip select signal
(a) data (b) address
(c) control bus (d) both (a) and (b)

62. _____ signal is generated by combining RD and WR signals with IO/M
(a) control (b) memory
(c) register (d) system

63. Memory is an integral part of a _____ system
(a) supercomputer (b) microcomputer
(c) mini computer (d) mainframe computer

64. ____ has certain signal requirements write into and read from its registers
(a) memory (b) register
(c) both (a) and (b) (d) control

65. An _____ is used to fetch one address
(a) internal decoder (b) external decoder
(c) encoder (d) register

66. The primary function of the _____ is to accept data from I/P devices
(a) multiprocessor (b) microprocessor
(c) peripherals (d) interfaces

67. _____ signal prevent the microprocessor from reading the same data more than one
(a) pipelining (b) handshaking
(c) controlling (d) signaling

68. Bits in IRR interrupt are _____
(a) reset (b) set
(c) stop (d) start

69. _____ generate interrupt signal to microprocessor and receive acknowledge
(a) priority resolver
(b) control logic
(c) interrupt request register
(d) interrupt register

70. The _____ pin is used to select direct command word
(a) A0 (b) D7-D6
(c) A12 (d) AD7-AD6

71. The _____ is used to connect more microprocessor
(a) peripheral device (b) cascade
(c) I/O devices (d) control unit

72. CS connect the output of _____
(a) encoder (b) decoder
(c) slave program (d) buffer

73. In which year, 8086 was introduced?
(a) 1978 (b) 1979
(c) 1977 (d) 1981

74. Expansion for HMOS technology_____
(a) high level mode oxygen semiconductor
(b) high level metal oxygen semiconductor
(c) high performance medium oxide semiconductor
(d) high performance metal oxide semiconductor

75. 8086 and 8088 contains _____ transistors
(a) 29000 (b) 24000
(c) 34000 (d) 54000

76. ALE stands for _____
(a) address latch enable
(b) address level enable
(c) address leak enable
(d) address leak extension

77. What is DEN?
(a) direct enable (b) data entered
(c) data enable (d) data encoding

78. In 8086, Example for Non maskable interrupts are _____.
(a) TRAP (b) RST6.5
(c) INTR (d) RST6.6

79. In 8086 the overflow flag is set when _____.
(a) the sum is more than 16 bits.
(b) signed numbers go out of their range after an arithmetic operation.
(c) carry and sign flags are set.
(d) subtraction

80. In 8086 microprocessor the following has the highest priority among all type interrupts?
(a) NMI (b) DIV 0
(c) TYPE 255 (d) OVER FLOW

81. In 8086 microprocessor one of the following statements is not true?
(a) coprocessor is interfaced in max mode.
(b) coprocessor is interfaced in min mode.
(c) I /O can be interfaced in max / min mode.
(d) supports pipelining

82. Address line for TRAP is?
(a) 0023H (b) 0024H
(c) 0033H (d) 0099H

83. Access time is faster for _____.
(a) ROM (b) SRAM
(c) DRAM (d) ERAM

84. The First Microprocessor was_____.
(a) Intel 4004 (b) 8080
(c) 8085 (d) 4008

85. Status register is also called as _____.
(a) accumulator (b) stack
(c) counter (d) flags

86. Which of the following is not a basic element within the microprocessor?
(a) Microcontroller
(b) Arithmetic logic unit (ALU)
(c) Register array
(d) Control unit

87. Which method bypasses the CPU for certain types of data transfer?
(a) Software interrupts
(b) Interrupt-driven I/O
(c) Polled I/O
(d) Direct memory access (DMA)

88. Which bus is bidirectional?
(a) Address bus
(b) Control bus
(c) Data bus
(d) None of the above

89. The first microprocessor had a(n)_____.
(a) 1 – bit data bus (b) 2 – bit data bus
(c) 4 – bit data bus (d) 8 – bit data bus

90. Which microprocessor has multiplexed data and address lines?
(a) 8086 (b) 80286
(c) 80386 (d) Pentium

91. Which is not an operand?
(a) Variable (b) Register
(c) Memory location (d) Assembler

92. Which is not part of the execution unit (EU)?
(a) Arithmetic logic unit (ALU)
(b) Clock
(c) General registers
(d) Flags

93. A 20-bit address bus can locate _____.
(a) 1,048,576 locations
(b) 2,097,152 locations
(c) 4,194,304 locations
(d) 8,388,608 locations

94. Which of the following is not an arithmetic instruction?
(a) INC (increment) (b) CMP (compare)
(c) DEC (decrement) (d) ROL (rotate left)

95. During a read operation the CPU fetches _____.
(a) a program instruction
(b) another address
(c) data itself
(d) all of the above

96. Which of the following is not an 8086/8088 general-purpose register?
(a) Code segment (CS)
(b) Data segment (DS)
(c) Stack segment (SS)
(d) Address segment (AS)

97. A 20-bit address bus allows access to a memory of capacity
(a) 1 MB (b) 2 MB
(c) 4 MB (d) 8 MB

98. Which microprocessor accepts the program written for 8086 without any changes?
(a) 8085 (b) 8086
(c) 8087 (d) 8088

99. Which group of instructions do not affect the flags?
(a) Arithmetic operations
(b) Logic operations
(c) Data transfer operations
(d) Branch operations

100. The result of MOV AL, 65 is to store
(a) store 0100 0010 in AL
(b) store 42H in AL
(c) store 40H in AL
(d) store 0100 0001 in AL

ANSWERS

1. (b)	2. (a)	3. (b)	4. (a)	5. (c)
6. (a)	7. (b)	8. (a)	9. (d)	10. (b)
11. (a)	12. (a)	13. (b)	14. (a)	15. (c)
16. (a)	17. (b)	18. (a)	19. (b)	20. (a)
21. (b)	22. (a)	23. (b)	24. (a)	25. (c)
26. (c)	27. (a)	28. (b)	29. (a)	30. (a)
31. (a)	32. (a)	33. (b)	34. (a)	35. (b)
36. (c)	37. (a)	38. (b)	39. (a)	40. (a)
41. (a)	42. (b)	43. (c)	44. (a)	45. (b)
46. (b)	47. (a)	48. (a)	49. (b)	50. (b)
51. (a)	52. (b)	53. (a)	54. (b)	55. (a)
56. (a)	57. (b)	58. (a)	59. (c)	60. (b)
61. (b)	62. (a)	63. (b)	64. (a)	65. (a)
66. (b)	67. (b)	68. (b)	69. (b)	70. (a)
71. (b)	72. (b)	73. (a)	74. (d)	75. (a)
76. (a)	77. (c)	78. (a)	79. (a)	80. (a)
81. (b)	82. (b)	83. (b)	84. (a)	85. (d)
86. (a)	87. (d)	88. (c)	89. (c)	90. (a)
91. (d)	92. (b)	93. (a)	94. (d)	95. (d)
96. (d)	97. (a)	98. (d)	99. (c)	100. (d)

EXERCISE

1. What is microprocessor?
2. Explain the features of microprocessor.
3. Compare between 8-bit, 16-bit and 32-bit microprocessor.
4. With a neat diagram explain the architecture of 8086.
5. Explain the flags in details.
6. Draw the pin diagram of 8086 and explain in brief.
7. Explain stack and stack pointer.
8. What is clock and reset circuits?
9. Explain I/O ports.
10. Explain the memory structure.
11. What are the 4 categories of segments explain with neat diagram?
12. Explain timing diagrams and execution cycles.

Unit V

8086 INSTRUCTION SET AND PROGRAMMING

5.1 MEMORY INTERFACING

- This unit covers about the memory and I/O interfacing of 8086.
- DMA transfers are essential for high performance embedded systems where large chunks of data need to be transferred from the input/output devices to or from the primary memory, this unit also discuss DMA in detail.
- At last this unit talks about the interrupts of 8086 microprocessor.

5.2 EVEN AND ODD MEMORY BANKS OF 8086

- We cannot think of a microcomputer system without memories and I/Os. But the question that arises in our mind is that how these memories and I/Os are connected (interfaced) to the microprocessor so that only one memory or I/O device will be selected at a time. For this the address decoder 74LS138 plays an important role. Let us now understand how this interfacing of memories and I/Os is done and the role of the 74LS138 in address decoding.
- The 8086 is a 16-bit microprocessor hence it can access 16-bits (one word) at a time from memory or I/O device. But the commercially available memories are of only byte size and store one byte at each location. Therefore to store 16-bit data two successive memory locations are used and the lower byte is stored in first memory location and the higher byte is stored at the second memory location. Thus if we think that all 16-bit data are stored in the memory locations then the entire memory will get divided into two halves as the data D_7-D_0 will be stored at all odd locations and D_{15}-D_8 will be stored at all even locations.
- In 8086 based system the 1 Mbytes of physical memory is organized as an odd bank and even bank, each of 512 Kbytes. The memory is addressed in parallel by the processor. The physical memory organization can be understood with the following points.
  - Byte data with even address is transferred on D_7-D_0 and byte data with odd address is transferred on D_{15}-D_8.
  - The processor provides two enable signals, $\overline{BHE}$ and A_0 for selecting of either even or odd or both the banks.
- The instructions are fetched from the memory as words and internally decoded by the decoder. The word fetched from the memory may consists of :
  - Both the bytes may be data operands.
  - Both the bytes may be opcodes.
  - One of the byte may be opcode and other may be data.
- While referring the 16-bit data i.e. word data the BIU requires one or two memory cycles depending on whether the starting byte is located at the odd address or even address. It is always better to store the word data at even address for efficient memory access.
- If the word is located at even address only one read or write cycle will be required. But if it is located at the odd address two memory read/write cycles will be required, the first read/write cycle will get lower byte and the second cycle will get the higher byte.

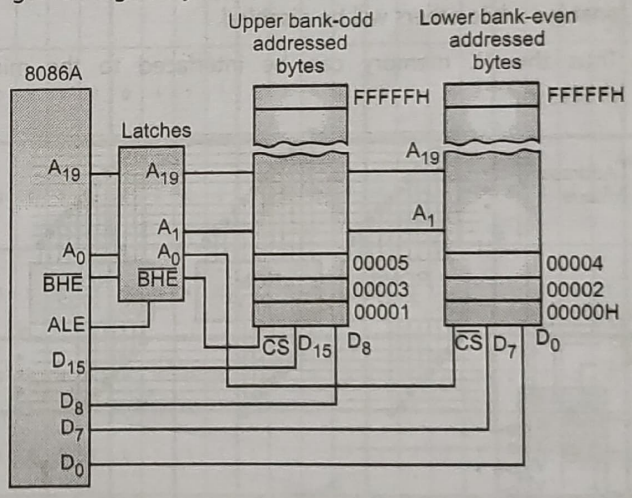


Fig. 5.1 : Block diagram of 8086 memory banks

Table 5.1 : Signals for Byte and Word Operations

Address	Data Type	$\overline{BHE}$	A_0	Bus Cycles	Data Lines Used
0000	Byte	1	0	One	D_0-D_7
0000	Word	0	0	One	D_0-D_{15}
0001	Byte	0	1	One	D_7-D_{15}
0001	Word	0	1	First	D_0-D_7
		1	0	Second	D_7-D_{15}

5.3 MEMORY ADDRESSING IN AN 8086 BASED MICROCOMPUTER SYSTEM

The addressing decoding scheme for every microcomputer system is different. Let us understand the concept of address decoding.

5.3.1 Interfacing of the ROM (Any Type of Read Only Memory) to the Microprocessor 8086

To start with let us understand how eight EPROMs can be connected in parallel to a common address and data bus. By looking at the above interfacing we can make following conclusions.

- As data bus is 8-bit wide the memory device can store 8-bit at one location.
- As each EPROM has 12 address lines therefore it can store 2^{12} i.e. 4096 bytes.
- The $\overline{CS}$ of each ROM device is connected to the output of the decoder (74LS138).
- The 74LS138 is enabled by making G2A and G2B inputs low and its GI input high.
- The output that is low is determined by the 3-bit address applied to the C. B and A select inputs. For example if CBA is 000 then the Y_0 output will be low and only ROM 0 will be enabled while others will be disabled.
- Thus the 4K memory can be interfaced to the micro-processor.

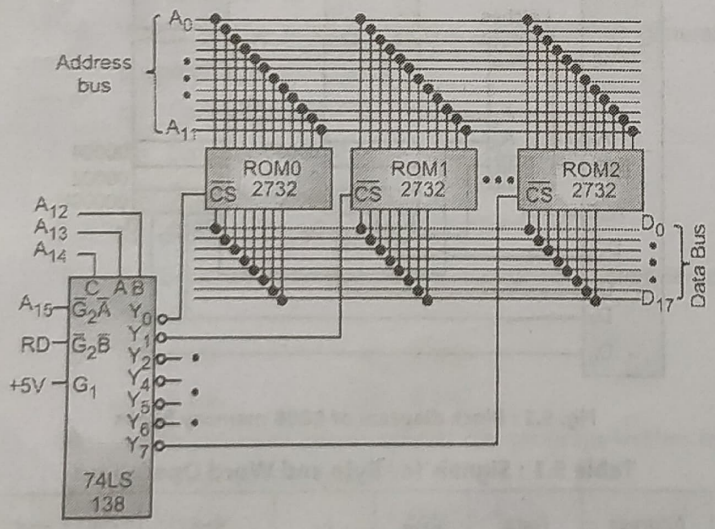


Fig. 5.2 : Parallel ROMs with decoder

To determine the addresses of the ROMs, RAMS and ports in a system one should use worksheet such as that in Fig. 5.2.

- To address the first location in any of the ERROM the A_0 to A_{11} address lines must all be low so put 0 under each of these address bits on the worksheet.
- To enable EPROM 0, the select inputs of the decoder must be all '0's therefore A_{14}, A_{13} and A_{12} are connected to these select inputs.
- As address line A_{15} is connected to the $\overline{G2A}$ enable input of the decoder, it must be asserted low in order for the decoder to work at all. Write 0 under the A_{15} on the worksheet
- The $\overline{RD}$ of the microprocessor is connected to the $\overline{G2B}$ enable input of the decoder. So that the decoder will be enabled only during the read operation. This is done to prevent any accidental writing of the memory contents in the ROM (Read Only Memory).
- The G_1 enable input of the decoder is permanently tied to the 5V.
- Thus the processor can now read directly from the ROM blocks with the given addresses. For example, the ROM 4 has starting address of 3000H i.e. the decoder input is 011 and (Y_3) will be low hence the Chip select of the ROM4 will be active and the processor will read the data from 3000h to 3FFFH.

Table 5.2

	Hex Digit				Hex Digit				Hex Digit				Hex Digit				Hex
Memory	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Equivalent Addresses
Block 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
START																	
END	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFFH
Block 2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1000H
START																	
END	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH
Block 3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H
START																	
END	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFFH
Block 4	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3000H
START																	
END	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH
Block 5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000H
START																	
END	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFFH
Block 6	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5000H
START																	
END	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFFH
Block 7	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000H
START																	
END	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	6FFFH
Block 8	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	7000H
START																	
END	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFFH

Decoder address Input

5.3.2 Interfacing of the RAM (Any Type of Random Access Memory) to the Microprocessor 8086

- Let us take an example of interfacing a RAM of 2K x 8 to the system and have the address of first RAM at 8000H i.e. just above the address of the ROM (7FFFH).
- To start with draw the worksheet shown similar to the above worksheet addressing one of the RAM of 2048 bytes (2^{11}) in each RAM requires 11 address lines, A_0 through A_{10}. Let us use the upper three address lines as input to decoder to select any one of the eight RAMS. We want the start address of the RAM as 8000H. For this connect the A_{15} line to 1 and A_{14} line to 0. Hence the complete work sheet would be as shown.

Table 5.3

	Hex Digit				Hex Digit				Hex Digit				Hex Digit				Hex Equivalent Addresses
Memory	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Block 1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H
START																	
END	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FFH
Block 2	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	8800H
START																	
END	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	8FFFH
Block 3	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	9000H
START																	
END	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	97FFH
Block 4	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	9800H
START																	
END	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	9FFFH
Block 5	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	A000H
START																	
END	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	A7FFH
Block 6	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	A800H
START																	
END	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	AFFFH
Block 7	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	B000H
START																	
END	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	0	B7FFH
Block 8	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	B800H
START																	
END	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFFH

Decoder address input

5.4 EXAMPLES OF DESIGN OF 8086 BASED MICROCOMPUTER SYSTEM

Example 5.1 : Design an 8086 based system with the following specifications :

(a) 8086 in minimum mode.
(b) 64 K byte EPROM.
(c) 64 K bytes of RAM.

Draw the complete schematic of the design indicating address map.

Solution : For 8086 the data bus is of 16-bit hence it can address the 16-bit data simultaneously. To interface the above memory modules can be interfaced as two 32 K byte of EPROM and two 32 K byte of the RAM.

For 32 K byte RAM and EPROM 15 address lines will be required hence A_0 through A_{14} lines will be used. The A_0 and $\overline{BHE}$ are used to select even and odd memory banks respectively. The figure below shows the memory map and the interfacing between the 8086 and given memory chips.

Table 5.4

	Hex Digit				Hex Digit				Hex Digit				Hex Digit				Hex Digit				Hex Equivalent Addresses
BHE	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000H(E)
1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	07FFEH(E)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	00001H(O)
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	07FFFH(O)
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80000H(E)
1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	87FFEH(E)
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	80001H(O)
0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	87FFFH(O)

The addresses of the memories are as given below.

Table 5.5

Memory	Odd/Even Bank	Starting Address	End Address
EEPROM	EVEN	00000H	07FFFH
	ODD	00001H	07FFFH
RAM	EVEN	80000H	87FFFH
	ODD	80001H	87FFFH

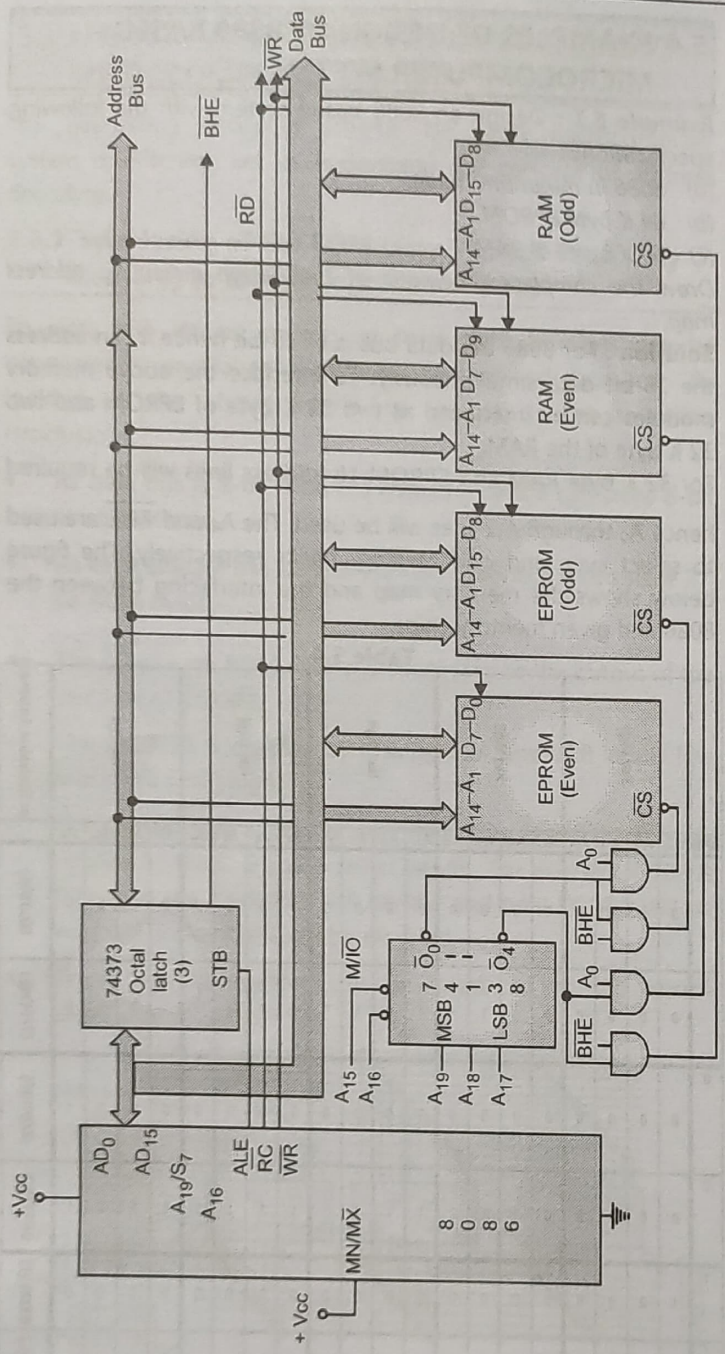


Fig. 5.3

5.5 I/O ADDRESSING IN A 8086 BASED MICROCOMPUTER SYSTEM

I/O Ports :

- The 8086 can be interfaced to 8- and 16-bit I/O devices using either standard or memory-mapped I/O. The standard I/O uses the instructions IN and OUT and is capable of providing 64 K bytes of I/O ports. Using standard I/O, the 8086 can transfer 8- or 16-bit data to or from a peripheral device.
- The 64 K byte I/O locations can then be configured as 64 K 8-bit ports or 32 K 16-bit ports. All I/O transfer between the 8086 and the peripheral devices take place via. AL for 8-bit ports and AX for 16-bit ports.
- The I/O port addressing can be done either directly or indirectly as follows :

DIRECT :

- IN AL, PORT A or IN AX, PORT B inputs 8-bit contents of port A into Al or 16-bit contents of port B into AX, respectively. Port A and Port B are assumed as 8- and 16-bit ports, respectively.
- OUT PORT A, AL or OUT PORT B, AX outputs 8-bit contents of AL into port A or 16-bit contents of AX into port B, respectively.

INDIRECT :

- IN AX, DX or IN AL, DX inputs 16-bit data addressed by DX into AX or 8-bit data addressed by DX into AL, respectively.
- OUT DX, AX or OUT DX, AL outputs 16-bit contents of AX into the port addressed by DX or 8-bit contents of AL into the port addressed by DX, respectively. In indirect addressing, register DX is used to hold the port address.

5.5.1 Interfacing an 8-bit or 16–bit Input Port to the 8086

The processor can accept 8-bit or 16-bit data on input port. The interfacing of the 8-bit port and 16-bit port can be as shown in the Fig. 5.4.

- The selection of the device is done by enabling the OE of the buffer connected to the device as shown in Fig. 5.4.
- The output of the buffer is connected to the data bus of the 8086.
- The port here when enabled by the control signals can be read by the input instructions.
- For interfacing the ports the absolute decoding schemes can be used. In this the all remaining address lines can be used for decoding as shown in the Fig. 5.4.
- The control signals used are as shown in the figure.

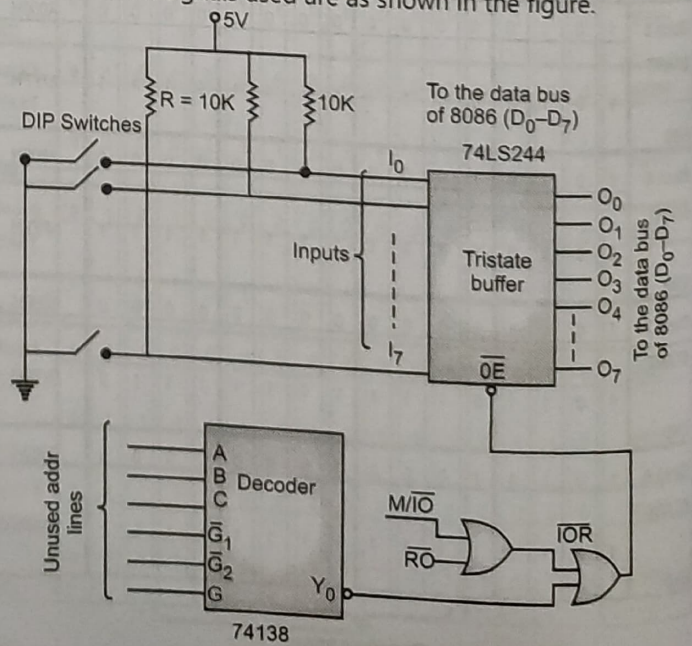


Fig. 5.4

5.5.2 Interfacing an 8-bit or 16–bit Output Port to the 8086

The processor can send 8-bit or 16-bit data on input port. The interfacing of the 8-bit port and 16-bit port can be as shown in the Fig. 5.5

- The decoding logic will be done in such away that the latch connected to the respective port will be enabled.
- The output of the buffer is connected to the data bus of the 8086.
- The port here when enabled by the control signals can be read by the input instructions.
- For interfacing the ports the absolute decoding schemes can be used. In this the all remaining address lines can be used for decoding as shown in the Fig. 5.5.
- The control signals used are as shown in the Fig. 5.5.

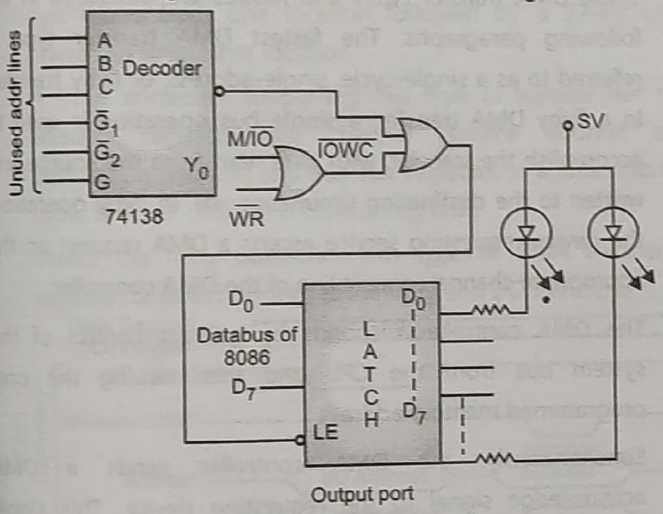


Fig. 5.5

Depending on this the I/O devices can be interfaced with microprocessor system in two different ways.

5.6 MEMORY MAPPED I/O AND I/O MAPPED I/O

5.6.1 Memory Mapped I/O

- Portions of a program's address space are assigned to I/O devices.
- Reads and writes to these addresses are interpreted as commands to the input and output device.
- These memory addresses are not directly accessible to user programs.
- All memory related instructions can be used to access the I/O ports.

Data transfer using the memory-mapped I/O is accomplished by using memory-oriented instructions such as MOV reg 8 or reg 16, [BX] and MOV [BX], reg 8 or reg 16 for inputting and outputting 8- or 16-bit data from or to an 8-bit register or a 16-bit register addressed by the 20-bit memory-mapped port location computed from DS and BX.

5.6.2 I/O Mapped I/O

- Special I/O instructions are used to transfer the data between the processor and I/O devices.
- I/O instructions can specify both the device number and the command word (or the location of the command word in memory).
- I/O instructions can only be executed by the operating system.

Data transfer using the I/O mapped I/O is accomplished by using instructions such as IN and OUT.

Table 5.6 : Difference between Memory Mapped I/O and I/O Mapped I/O

Sr. No.	Memory Mapped I/O	I/O Mapped I/O
1.	All memory related instructions can be used to access the I/O ports.	Special instructions such as IN, OUT, INS and OUTS are used to transfer and access the data to the I/O ports.
2.	The address of the I/O device is of 20-bits.	The address of the I/O device is 8-bit or 16-bit for direct and indirect addressing respectively.
3.	The control signal M/$\overline{IO}$ is High for accessing the I/O port.	The control signal M/$\overline{IO}$ is low for accessing the I/O port.
4.	Maximum no. of devices that can be interfaced are 1 MB.	The maximum number of I/O devices that can be interfaced are 256 for direct and 65536 (2^{16}) for indirect addressing.
5.	More hardware will be required as 20 address lines are to be decoded.	Less hardware will be required as only 8 or 16 address lines are to be decoded.
6.	Some of the memory addresses will be reserved for addressing I/O.	I/O addresses will be separate for the I/O.

5.7 DIRECT MEMORY ACCESS CONTROLLER

- Direct Memory Access (DMA) allows devices to transfer data without subjecting the processor a heavy overhead. Otherwise, the processor would have to copy each piece of data from the source to the destination.
- This is typically slower than copying normal blocks of memory since access to I/O devices over a peripheral bus is generally slower than normal system RAM. During this time the processor would be unavailable for any other tasks involving processor bus access.
- But it can continue to work on any work which does not require bus access.
- DMA transfers are essential for high performance embedded systems where large chunks of data need to be transferred from the input/output devices to or from the primary memory.

5.7.1 DMA Controller

- A DMA controller is a device, usually peripheral to a CPU that is programmed to perform a sequence of data transfers on behalf of the CPU. A DMA controller can directly access memory and is used to transfer data from one memory location to another, or from an I/O device to memory and vice versa. A DMA controller manages several DMA channels, each of which can be programmed to perform a sequence of these DMA transfers. Devices, usually I/O peripherals, that acquire data that must be read (or devices that must output data and be written to) signal the DMA controller to perform a DMA transfer by asserting a hardware DMA request (DRQ) signal. A DMA request signal for each channel is routed to the DMA controller.
- This signal is monitored and responded to in much the same way that a processor handles interrupts. When the DMA controller sees a DMA request, it responds by performing one or many data transfers from that I/O device into system memory or vice versa. Channels must be enabled by the processor for the DMA controller to respond to DMA requests. The number of transfers performed, transfer modes used, and memory locations accessed depends on how the DMA channel is programmed. A DMA controller typically shares the system memory and I/O bus with the CPU and has both bus master and slave capability.
- Fig. 5.6 (a) shows the DMA controller architecture and how the DMA controller interacts with the CPU. In bus master mode, the DMA controller acquires the system bus (address, data, and control lines) from the CPU to perform the DMA transfers. Because the CPU releases the system bus for the duration of the transfer, the process is sometimes referred to as cycle stealing.

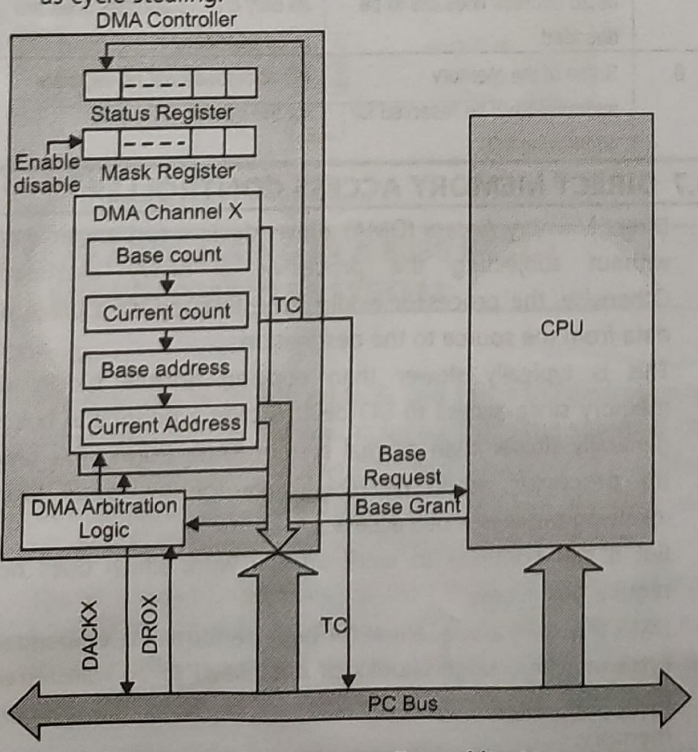


Fig. 5.6 (a) : DMA controller architecture

- In bus slave mode, the DMA controller is accessed by the CPU, which programs the DMA controller's internal registers to set up DMA transfers. The internal registers consist of source and destination address registers and transfer count registers for each DMA channel, as well as control and status registers for initiating, monitoring, and sustaining the operation of the DMA controller.

5.7.2 DMA Transfer Types and Modes

- DMA controllers vary as to the type of DMA transfers and the number of DMA channels they support.
- The two types of DMA transfers are flyby DMA transfers and fetch-and-deposit DMA transfers.
- The three common transfer modes are single, block and demand transfer modes.
- These DMA transfer types and modes are described in the following paragraphs. The fastest DMA transfer type is referred to as a single-cycle, single-address, or flyby transfer. In a flyby DMA transfer, a single bus operation is used to accomplish the transfer, with data read from the source and written to the destination simultaneously. In flyby operation, the device requesting service asserts a DMA request on the appropriate channel request line of the DMA controller.
- The DMA controller responds by gaining control of the system bus from the CPU and then issuing the pre-programmed memory address.
- Simultaneously, the DMA controller sends a DMA acknowledge signal to the requesting device. This signal alerts the requesting device to drive the data onto the system data bus or to latch the data from the system bus, depending on the direction of the transfer.
- In other words, a flyby DMA transfer looks like a memory read or write cycle with the DMA controller supplying the address and the I/O device reading or writing the data. Because flyby DMA transfers involve a single memory cycle per data transfer, these transfers are very efficient. Fig. 5. 6 (b) shows the flyby DMA transfer signal protocol.

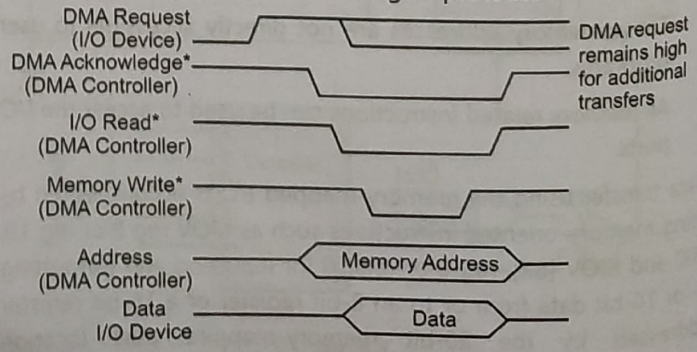


Fig. 5.6 (b) : Flyby DMA transfer signal protocol

- The second type of DMA transfer is referred to as a dual-cycle, dual-address, flow-through, or fetch-and-deposit DMA transfer.
- As these names imply, this type of transfer involves two memory or I/O cycles. The data being transferred is first read from the I/O device or memory into a temporary data register internal to the DMA controller.
- The data is then written to the memory or I/O device in the next cycle.
- Fig. 5.6 (b) shows the fetch-and-deposit DMA transfer signal protocol. Although inefficient because the DMA controller performs two cycles and thus retains the system bus longer, this type of transfer is useful for interfacing devices with different data bus sizes.
- For example, a DMA controller can perform two 16-bit read operations from one location followed by a 32-bit write operation to another location.
- A DMA controller supporting this type of transfer has two address registers per channel (source address and destination address) and bus-size registers, in addition to the usual transfer count and control registers.
- Unlike the flyby operation, this type of DMA transfer is suitable for both memory to memory and I/O transfers.

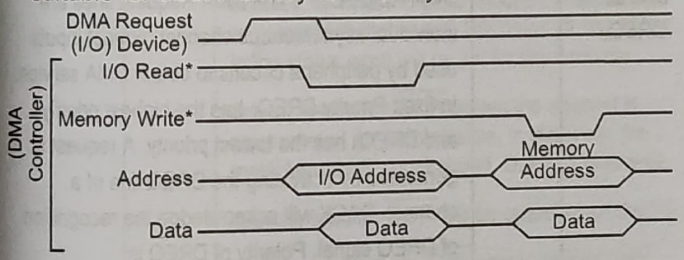


Fig. 5.7

- Single, block and demand are the most common transfer modes. Single transfer mode transfers one data value for each DMA request assertion.
- This mode is the slowest method of transfer because it requires the DMA controller to arbitrate for the system bus with each transfer.
- This arbitration is not a major problem on a lightly loaded bus, but it can lead to latency problems when multiple devices are using the bus.
- Block and demand transfer modes increase system throughput by allowing the DMA controller to perform multiple DMA transfers when the DMA controller has gained the bus.
- For block mode transfers, the DMA controller performs the entire DMA sequence as specified by the transfer count register at the fastest possible rate in response to a single DMA request from the I/O device. For demand mode transfers, the DMA controller performs DMA transfers at the fastest possible rate as long as the I/O device asserts its DMA request. When the I/O device unasserts this DMA request, transfers are held off.

5.7.3 Steps in a Typical DMA Cycle

Device wishing to perform DMA asserts the processors bus request signal.

- Processor completes the current bus cycle and then asserts the bus grant signal to the device.
- The device then asserts the bus grant ack signal.
- The processor senses in the change in the state of bus grant ack signal and starts listening to the data and address bus for DMA activity.
- The DMA device performs the transfer from the source to destination address.
- During these transfers, the processor monitors the addresses on the bus and checks if any location modified during DMA operations is cached in the processor. If the processor detects a cached address on the bus, it can take one of the two actions :
  1. Processor invalidates the internal cache entry for the address involved in DMA write operation.
  2. Processor updates the internal cache when a DMA write is detected.
- Once the DMA operations have been completed, the device releases the bus by asserting the bus release signal.
- Processor acknowledges the bus release and resumes its bus cycles from the point it left off.

5.8 8237 FEATURES

- Enable/Disable Control of Individual DMA Requests.
- Four Independent DMA Channels.
- Independent Auto initialization of all Channels.
- Memory-to-Memory Transfers.
- Memory Block Initialization.
- Address Increment or Decrement.
- High Performance : Transfers up to 1.6 M Bytes/Second with 5 MHz 8237A-5.
- Directly Expandable to any Number of Channels.
- End of Process Input for Terminating Transfers.
- Independent Polarity Control for DREQ and DACK Signals.
- Available in 40-Lead Cerdip and Plastic Packages.
- Supplies memory and I/O with control signals and addresses during DMA transfer.
- 4-channels (expandable)
  - 0 : DRAM refresh
  - 1 : Free
  - 2 : Floppy disk controller
  - 3 : Free

- 1.6 M Byte/sec transfer rate.
- 64 K Byte section of memory address capability with single programming "fly-by" controller (data does not pass through the DMA-only memory to I/O transfer capability)
- Initialization involves writing into each channel :
  - The address of the first byte of the block of data that must be transferred (called the base address).
  - The number of bytes to be transferred (called the word count).
- The 8237A Multimode Direct Memory Access (DMA) Controller is a peripheral interface circuit for microprocessor systems. It is designed to improve system performance by allowing external devices to directly transfer information from the system memory. Memory to memory transfer capability is also provided. The 8237A offers a wide variety of programmable control features to enhance data throughput and system optimization and to allow dynamic reconfiguration under program control.
- The 8237A is designed to be used in conjunction with an external 8-bit address latch. It contains four independent channels and may be expanded to any number of channels by cascading additional controller chips. The three basic transfer modes allow programmability of the types of DMA service by the user. Each channel can be individually programmed to Autoinitialize to its original condition following an End Of Process (EOP). Each channel has a full 64 K address and word count capability.

5.8.1 8237 Pins Description

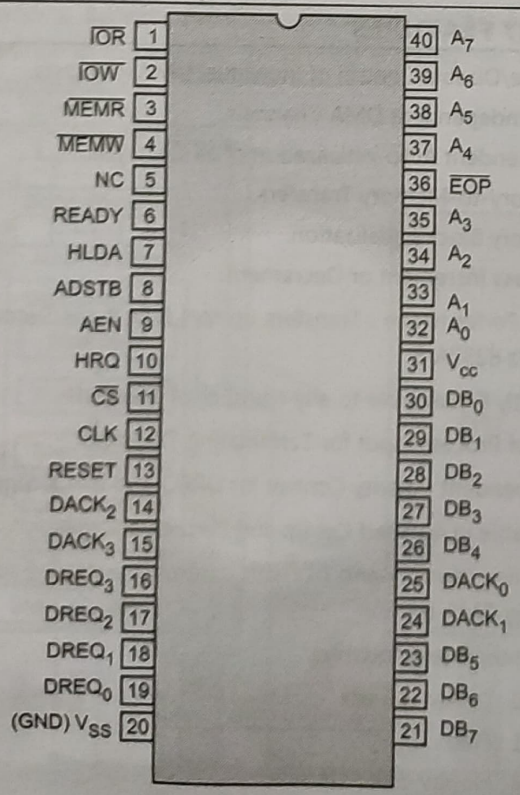


Fig. 5.8 : 8237 pin out

Table 5.7 : Pin Description

Symbol	Type	Name and Function
V_{CC}		**POWER :** + 5 V supply.
V_{SS}		**GROUND :** Ground.
CLK	I	**CLOCK INPUT :** Clock Input controls the internal operations of the 8237A and its rate of data transfers. The input may be driven at up to 5 MHz for the 8237A-5.
$\overline{CS}$	I	**CHIP SELECT :** Chip Select is an active low input used to select the 8237A as an I/O device during the Idle cycle. This allows CPU communication on the data bus.
RESET	I	**RESET :** Reset is an active high input which clears the Command Status Request and Temporary registers. It also clears the first/last flip/flop and sets the Mask register. Following a Reset the device is in the Idle cycle.
READY	I	**READY :** Ready is an input used to extend the memory read and write pulses from the 8237A to accommodate slow memories or I/O peripheral devices. Ready must not make transitions during its specified setup/hold time.
HLDA	I	**HOLD ACKNOWLEDGE :** The active high Hold Acknowledge from the CPU indicates that it has relinquished control of the system busses.
$DREQ_0$-$DREQ_3$	I	**DMA REQUEST :** The DMA Request lines are individual asynchronous channel request inputs used by peripheral circuits to obtain DMA service. In fixed Priority $DREQ_0$ has the highest priority and $DREQ_3$ has the lowest priority. A request is generated by activating the DREQ line of a channel. DACK will acknowledge the recognition of DREQ signal. Polarity of DREQ is programmable. Reset initializes these lines to active high. DREQ must be maintained until the corresponding DACK goes active.
DB_0-DB_7	I/O	**DATA BUS :** The Data Bus lines are bidirectional three-state signals connected to the system data bus. The outputs are enabled in the Program condition during the I/O Read to output the contents of an Address register, a Status register, the Temporary register or a Word Count register to the CPU. The outputs are disabled and the inputs are read during an I/O Write cycle when the CPU is programming the 8237A control registers. During DMA cycles the most significant 8 bits of the address are output onto the data bus to be strobed into an external latch by ADSTB. In memory-to-memory operations data from the memory comes into the 8237A on the data bus during the read from memory transfer. In the write-to-memory transfer, the data bus outputs place the data into the new memory location.

Symbol	Type	Name and Function
$\overline{IOR}$	I/O	**I/O READ :** I/O Read is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to read the control registers. In the Active cycle, it is an output control signal used by the 8237A to access data from a peripheral during a DMA Write transfer.
$\overline{IOW}$	I/O	**I/O WRITE :** I/O Write is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to load information into the 8237A. In the Active cycle it is an output control signal used by the 8237A to load data to the peripheral during a DMA Read transfer.
$\overline{EOP}$	I/O	**END OF PROCESS :** End of Process is an active low bidirectional signal. Information concerning the completion of DMA services is available at the bidirectional $\overline{EOP}$ pin. The 8237A allows an external signal to terminate an active DMA service. This is accomplished by pulling the $\overline{EOP}$ input low with an external $\overline{EOP}$ signal. The 8237A also generates a pulse when the terminal count (TC) for any channel is reached. This generates an $\overline{EOP}$ signal which is output through the $\overline{EOP}$ line. The reception of $\overline{EOP}$, either internal or external will cause the 8237A to terminate the service, reset the request and if Autoinitialize is enabled, to write the base registers to the current registers of that channel. The mask bit and TC bit in the status word will be set for the currently active channel by $\overline{EOP}$ unless the channel is programmed for Autoinitialize. In that case, the mask bit remains unchanged. During memory-to-memory transfer $\overline{EOP}$ will be output when the TC for channel 1 occurs $\overline{EOP}$ should be tied high with a pull-up resistor if it is not used to prevent erroneous end of process inputs.
A_0-A_3	I/O	**ADDRESS :** The four least significant address lines are bidirectional three-state signals. In the Idle cycle they are inputs and are used by the CPU to address the register to be loaded or read. In the Active cycle they are outputs and provide the lower 4 bits of the output address.
A_4-A_7	O	**ADDRESS :** The four most significant address lines are three-state outputs and provide 4 bits of address. These lines are enabled only during the DMA service.
HRQ	O	**HOLD REQUEST :** This is the Hold Request to the CPU and is used to request control of the system bus. If the corresponding mask bit is clear the presence of any valid DREQ causes 8237A to issue the HRQ.
$DACK_0$-$DACK_3$	O	**DMA ACKNOWLEDGE :** DMA Acknowledge is used to notify the individual peripherals when one has been granted a DMA cycle. The sense of these lines is programmable. Reset initializes them to active low.

Symbol	Type	Name and Function
AEN	O	**ADDRESS ENABLE :** Address Enable enables the 8-bit latch containing the upper 8 address bits onto the system address bus. AEN can also be used to disable other system bus drivers during DMA transfers. AEN is active HIGH.
ADSTB	O	**ADDRESS STROBE :** The active high Address Strobe is used to strobe the upper address byte into an external latch.
$\overline{MEMR}$	O	**MEMORY READ :** The Memory Read signal is an active low three-state output used to access data from the selected memory location during a DMA Read or a memory-to-memory transfer.
$\overline{MEMW}$	O	**MEMORY WRITE :** The Memory Write is an active low three-state output used to write data to the selected memory location during a DMA Write or a memory-to-memory transfer.
PIN_5	I	**PIN_5 :** This pin should always be at a logic HIGH level. An internal pull-up resistor will establish a logic high when the pin is left floating It is recommended however that PIN_5 be connected to V_{CC}.

5.8.2 Block Diagram of 8257 (DMA Controller)

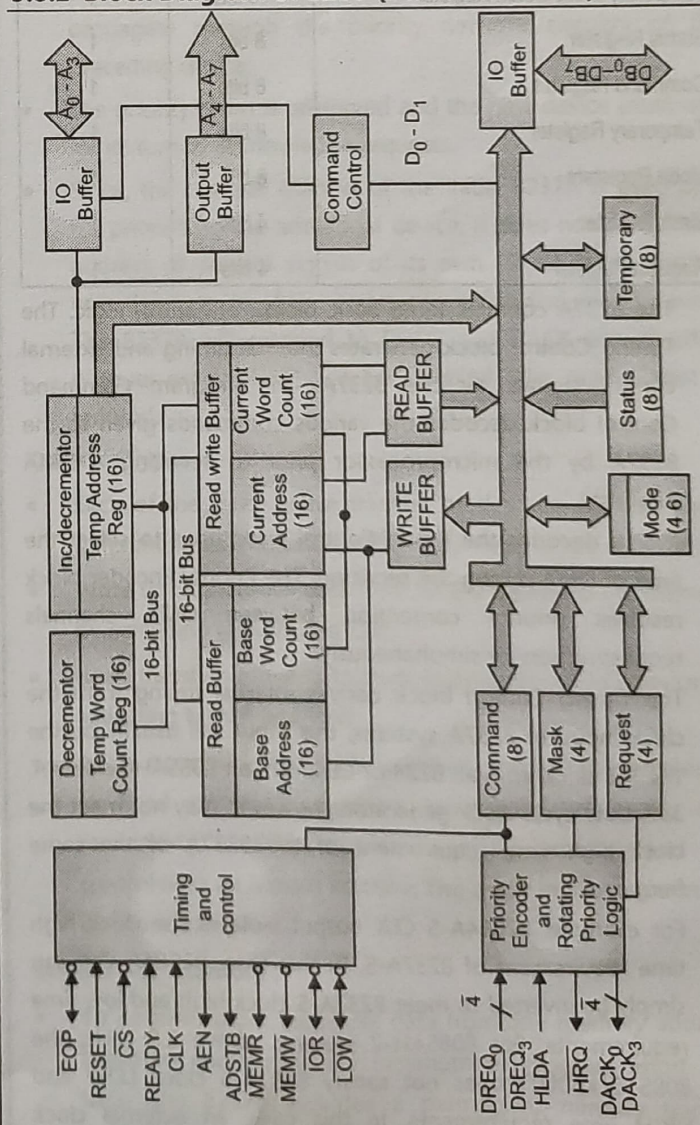


Fig. 5.9 : Block diagram

5.8.3 Functional Description

- The 8237A block diagram includes the major logic blocks and all of the internal registers. The data interconnection paths are also shown. Not shown are the various control signals between the blocks. The 8237A contains 344 bits of internal memory in the form of registers. Fig. 5.3 lists these registers by name and shows the size of each. A detailed description of the registers and their functions can be found under Register Description.

Table 5.8 : 8237A Internal Registers

Names	Size	Number
Base Address Registers	16 bits	4
Base Word Count Registers	16 bits	4
Current Address Registers	16 bits	4
Current Word Count Registers	16 bits	4
Temporary Address Register	16 bits	1
Temporary Word Count Register	16 bits	1
Status Register	8 bits	1
Command Register	8 bits	1
Temporary Register	8 bits	1
Mode Registers	8 bits	4
Mask Register	4 bits	1
Request Register	4 bits	1

- The 8237A contains three basic blocks of control logic. The Timing Control block generates internal timing and external control signals for the 8237A. The Program Command Control block decodes the various commands given to the 8237A by the microprocessor prior to servicing a DMA Request.
- It also decodes the Mode Control word used to select the type of DMA during the servicing. The Priority Encoder block resolves priority contention between DMA channels requesting service simultaneously.
- The Timing Control block derives internal timing from the clock input. In 8237A systems, this input will usually be the W2 TTL clock from an 8224 or CLK from an 8085AH or 8284A. 33% duty cycle clock generators, however, may not meet the clock high time requirement of the 8237A of the same frequency.
- For example, 82C84A-5 CLK output violates the clock high time requirement of 8237A-5. In this case, 82C84A CLK can simply be inverted to meet 8237A-5 clock high and low time requirements. For 8085AH-2 systems above 3.9 MHz, the 8085 CLK(OUT) does not satisfy 8237A-5 clock LOW and HIGH time requirements. In this case, an external clock should be used to drive the 8237A-5.

5.8.4 DMA Operation

- The 8237A is designed to operate in two major cycles. These are called Idle and Active cycles. Each device cycle is made up of a number of states. The 8237A can assume seven separate states, each composed of one full clock period.
- State I (SI) is the inactive state. It is entered when the 8237A has no valid DMA requests pending. While in SI, the DMA controller is inactive but may be in the Program Condition, being programmed by the processor. State S_0 (S_0) is the first state of a DMA service.
- The 8237A has requested a hold but the processor has not yet returned an acknowledge. The 8237A may still be programmed until it receives HLDA from the CPU. An acknowledge from the CPU will signal that DMA transfers may begin. S_1, S_2, S_3 and S_4 are the working states of the MA service.
- If more time is needed to complete a transfer than is available with normal timing, wait states (SW) can be inserted between S_2 or S_3 and S_4 by the use of the Ready line on the 8237A.
- Note that the data is transferred directly from the I/O device to memory (or vice versa) with IOR and MEMW (or MEMR and IOW) being active at the same time. The data is not read into or driven out of the 8237A in I/O to memory or memory to I/O DMA transfers.
- Memory-to-memory transfers require a read-from and a write-to-memory to complete each transfer. The states, which resemble the normal working states, use two digit numbers for identification. Eight states are required for a single transfer. The first four states (S_{11}, S_{12}, S_{13}, S_{14}) are used for the read from memory half and the last four states (S_{21}, S_{22}, S_{23}, S_{24}) for the write-to-memory half of the transfer.

Idle Cycle :

- When no channel is requesting service, the 8237A will enter the Idle cycle and perform "SI" states. In this cycle, the 8237A will sample the DREQ lines every clock cycle to determine if any channel is requesting a DMA service.
- The device will also sample CS, looking for an attempt by the microprocessor to write or read the internal registers of the 8237A.
- When CS is low and HLDA is low, the 8237A enters the Program condition.
- The CPU can now establish, change or inspect the internal definition of the part by reading from or writing to the internal registers.
- Address lines $A_0 \pm A_3$ are inputs to the device and select which registers will be read or written.
- The IOR and IOW lines are used to select and time reads or writes. Due to the number and size of the internal registers, an internal flip-flop is used to generate an additional bit of address. This bit is used to determine the upper or lower byte

of the 16-bit Address and Word Count registers. The flip-flop is reset by Master Clear or Reset.

- A separate software command can also reset this flip-flop. Special software commands can be executed by the 8237A in the Program condition.
- These commands are decoded as sets of addresses with the CS and IOW.
- The commands do not make use of the data bus. Instructions include Clear First/Last Flip-Flop and Master Clear Special software commands can be executed by the 8237A in the Program Condition. These commands are decoded as sets of addresses with the CS and IOW. The commands do not make use of the data bus. Instructions include Clear First/Last Flip-Flop and Master Clear.

Active Cycle :

When the 8237A is in the Idle cycle and a non-masked channel requests a DMA service, the device will output an HRQ to the microprocessor and enter the Active cycle. It is in this cycle that the DMA service will take place, in one of four modes :

1. **Single Transfer Mode :**

- In Single Transfer mode, the device is programmed to make one transfer only. The word count will be decremented and the address decremented or incremented following each transfer. When the word count "rolls over" from zero to FFFFH, a Terminal Count (TC) will cause an Autoinitialize if the channel has been programmed to do so. DREQ must be held active until DACK becomes active in order to be recognized. If DREQ is held active throughout the single transfer, HRQ will go inactive and release the bus to the system. It will again go active and, upon receipt of a new HLDA, another single transfer will be performed.
- In 8080A, 8085AH, 8088, or 8086 system, this will ensure one full machine cycle execution between DMA transfers. Details of timing between the 8237A and other bus control protocols will depend upon the characteristics of the microprocessor involved.

2. **Block Transfer Mode :**

- In Block Transfer mode, the device is activated by DREQ to continue making transfers during the service until a TC, caused by word count going to FFFFH, or an external End of Process (EOP) is encountered. DREQ need only be held active until DACK becomes active.
- Again, an Autoinitialization will occur at the end of the service if the channel has been programmed for it.

3. **Demand Transfer Mode :**

- In Demand Transfer mode, the device is programmed to continue making transfers until a TC or external EOP is encountered or until DREQ goes inactive. Thus transfers may continue until the I/O device has exhausted its data capacity.
- After the I/O device has had a chance to catch up, the DMA service is re-established by means of a DREQ. During the time between services when the microprocessor is allowed to operate, the intermediate values of address and word count are stored in the 8237A Current Address and Current Word Count registers.
- Only an EOP can cause an Autoinitialize at the end of the service. EOP is generated either by TC or by an external signal. DREQ has to be low before S_4 to prevent another Transfer.

4. **Cascade Mode :**

- This mode is used to cascade more than one 8237A together for simple system expansion. The HRQ and HLDA signals from the additional 8237A are connected to the DREQ and DACK signals of a channel of the initial 8237A.
- This allows the DMA requests of the additional device to propagate through the priority network circuitry of the preceding device.
- The priority chain is preserved and the new device must wait for its turn to acknowledge requests.
- Since, the cascade channel of the initial 8237A is used only for prioritizing the additional device, it does not output any address or control signals of its own. These could conflict with the outputs of the active channel in the added device. The 8237A will respond to DREQ and DACK but all other outputs except HRQ will be disabled. The ready input is ignored.

Transfer Types :

- Each of the three active transfer modes can perform three different types of transfers. These are Read, Write and Verify.
- Write transfers move data from an I/O device to the memory by activating MEMW and IOR.
- Read transfers move data from memory to an I/O device by activating MEMR and IOW.
- Verify transfers are pseudo transfers. The 8237A operates as in Read or Write transfers generating addresses, and responding to EOP, etc. However, the memory and I/O control lines all remain inactive. The ready input is ignored in verify mode.

Memory-to-Memory :

- To perform block moves of data from one memory address space to another with a minimum of program effort and time, the 8237A includes a memory-to-memory transfer feature.

- Programming a bit in the Command register selects channels 0 and 1 to operate as memory-to-memory transfer channels.
- The transfer is initiated by setting the software DREQ for channel 0. The 8237A requests a DMA service in the normal manner. After HLDA is true, the device, using four state transfers in Block Transfer mode, reads data from the memory.
- The channel 0 Current Address register is the source for the address used and is decremented or incremented in the normal manner. The data byte read from the memory is stored in the 8237A internal Temporary register.
- Channel 1 then performs a four-state transfer of the data from the Temporary register to memory using the address in its Current Address register and incrementing or decrementing it in the normal manner. The channel 1 current Word Count is decremented. When the word count of channel 1 goes to FFFFH, a TC is generated causing an EOP output terminating the service.
- Channel 0 may be programmed to retain the same address for all transfers. This allows a single word to be written to a block of memory.
- The 8237A will respond to external EOP signals during memory-to-memory transfers. Data comparators in block search schemes may use this input to terminate the service when a match is found. Memory-to-memory operations can be detected as an active AEN with no DACK outputs.

Autoinitialize :

- By programming a bit in the Mode register, a channel may be set up as an Autoinitialize channel. During Autoinitialize initialization, the original values of the Current Address and Current Word Count registers are automatically restored from the Base Address and Base Word count registers of that channel following EOP. The base registers are loaded simultaneously with the current registers by the microprocessor and remain unchanged throughout the DMA service. The mask bit is not altered when the channel is in Autoinitialize.
- Following Autoinitialize the channel is ready to perform another DMA service, without CPU intervention, as soon as a valid DREQ is detected. In order to Autoinitialize both channels in a memory-to-memory transfer, both word counts should be programmed identically. If interrupted externally, EOP pulses should be applied in both bus cycles.

Priority :

- The 8237A has two types of priority encoding available as software selectable options. The first is Fixed Priority which fixes the channels in priority order based upon the descending value of their number. The channel with the lowest priority is 3 followed by 2, 1 and the highest priority channel, 0.
- After the recognition of any one channel for service, the other channels are prevented from interfering with that service until it is completed.
- After completion of a service, HRQ will go inactive and the 8237A will wait for HLDA to go low before activating HRQ to service another channel. The second scheme is Rotating Priority.
- The last channel to get service becomes the lowest priority channel with the others rotating accordingly. With Rotating Priority in a single chip DMA system, any device requesting service is guaranteed to be recognized after no more than three higher priority services have occurred. This prevents any one channel from monopolizing the system.

Compressed Timing :

- In order to achieve even greater throughput where system characteristics permit, the 8237A can compress the transfer time to two clock cycles.
- From Fig. 5.9 it can be seen that state S_3 is used to extend the access time of the read pulse. By removing state S_3, the read pulse width is made equal to the write pulse width and a transfer consists only of state S_2 to change the address and state S_4 to perform the read/write. S_1 states will still occur when $A_8 \pm A_{15}$ need updating (see Address Generation). Timing for compressed transfers is found in Fig. 5.9.

Address Generation :

- In order to reduce pin count, the 8237A multiplexes the eight higher order address bits on the data lines. State S_1 is used to output the higher order address bits to an external latch from which they may be placed on the address bus.
- The falling edge of Address Strobe (ADSTB) is used to load these bits from the data lines to the latch. Address Enable (AEN) is used to enable the bits onto the address bus through a three-state enable. The lower order address bits are output by the 8237A directly. Lines $A_0 \pm A_7$ should be connected to the address bus. Fig. 5.9 shows the time relationships between CLK, AEN, ADSTB, $DB_0 \pm DB_7$ and $A_0 \pm A_7$.
- During Block and Demand Transfer mode services, which include multiple transfers, the addresses generated will be sequential. For many transfers the data held in the external address latch will remain the same. This data need only change when a carry or borrow from A_7 to A_8 takes place in the normal sequence of addresses. To save time and speed transfers, the 8237A executes S_1 states only when updating of $A_8 \pm A_{15}$ in the latch is necessary. This means for long services, S_1 states and Address Strobes may occur only once every 256 transfers, a savings of 255 clock cycles for each 256 transfers.

5.8.5 Register Description

Current Address Register :

- Each channel has a 16-bit Current Address register. This register holds the value of the address used during DMA transfers. The address is automatically incremented or decremented after each transfer and the intermediate values of the address are stored in the Current Address register during the transfer.
- This register is written or read by the microprocessor in successive 8-bit bytes. It may also be reinitialized by an Autoinitialize back to its original value. Autoinitialize takes place only after an EOP.

Current Word Register :

- Each channel has a 16-bit Current Word Count register. This register determines the number of transfers to be performed. The actual number of transfers will be one more than the number programmed in the Current Word Count register (i.e., programming a count of 100 will result in 101 transfers). The word count is decremented after each transfer. The intermediate value of the word count is stored in the register during the transfer.
- When the value in the register goes from zero to FFFFH, a TC will be generated. This register is loaded or read in successive 8-bit bytes by the microprocessor in the Program Condition. Following the end of a DMA service it may also be reinitialized by an autoinitialization back to its original value. Autoinitialize can occur only when an EOP occurs. If it is not autoinitialized, this register will have a count of FFFFH after TC.

Base Address and Base Word Count Registers :

- Each channel has a pair of Base Address and Base Word Count registers. These 16-bit registers store the original value of their associated current registers. During autoinitialize, these values are used to restore the current registers to their original values.
- The base registers are written simultaneously with their corresponding current register in 8-bit bytes in the Program Condition by the microprocessor. These registers cannot be read by the microprocessor.

Command Register :

- This 8-bit register controls the operation of the 8237A. It is programmed by the microprocessor in the Program Condition and is cleared by Reset or a Master Clear instruction.
- The following Fig. 5.10 lists the function of the command bits. See Fig. 5.10 for address coding.

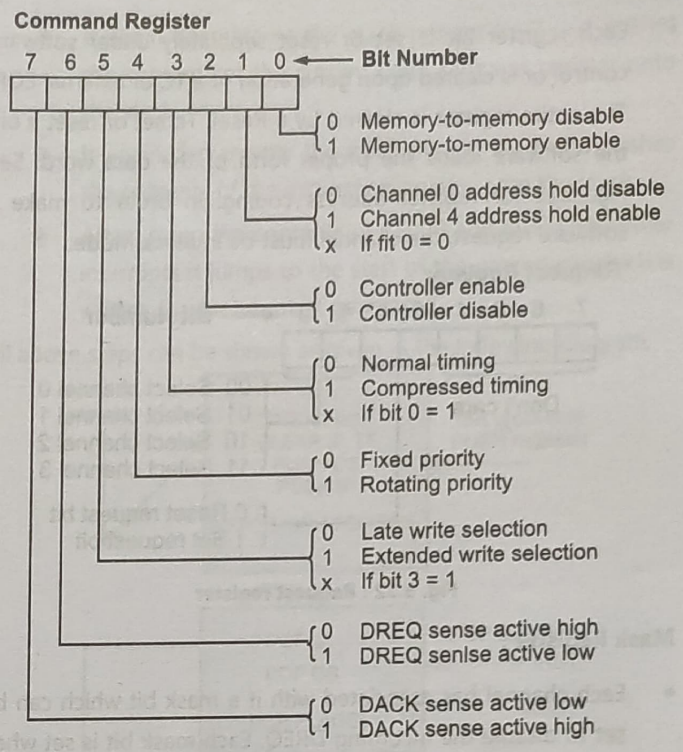


Fig. 5.10 : Command register

Mode Register :

- Each channel has a 6-bit Mode register associated with it. When the register is being written to by the microprocessor in the Program Condition, bits 0 and 1 determine which channel Mode register is to be written.

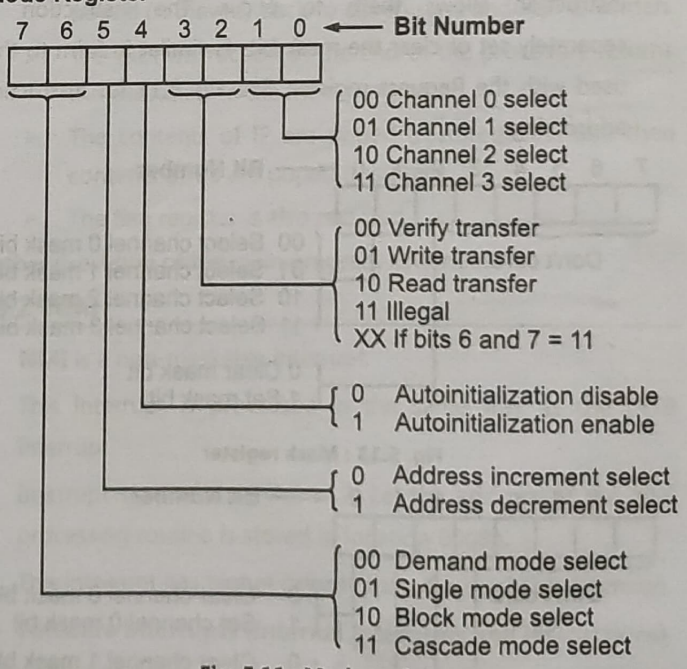


Fig. 5.11 : Mode register

Request Register :

- The 8237A can respond to requests for DMA service which are initiated by software as well as by a DREQ. Each channel has a request bit associated with it in the 4-bit Request register. These are non-maskable and subject to prioritization by the Priority Encoder network.

- Each register bit is set or reset separately under software control or is cleared upon generation of a TC or external EOP. The entire register is cleared by a Reset. To set or reset a bit, the software loads the proper form of the data word. See Fig. 5.12 for register address coding. In order to make a software request, the channel must be in Block Mode.

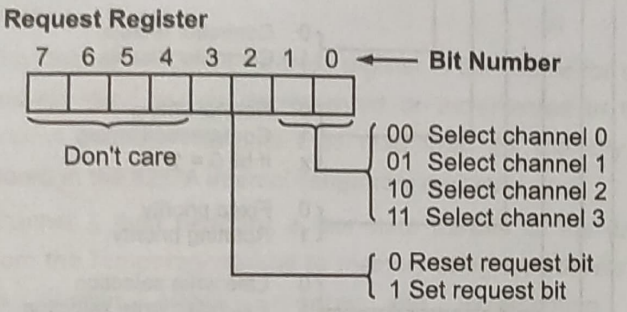


Fig. 5.12 : Request register

Mask Register :

- Each channel has associated with it a mask bit which can be set to disable the incoming DREQ. Each mask bit is set when its associated channel produces an EOP if the channel is not programmed for Autoinitialize. Each bit of the 4-bit Mask register may also be set or cleared separately under software control. The entire register is also set by a Reset.
- This disables all DMA requests until a clear Mask register instruction allows them to occur. The instruction to separately set or clear the mask bits is similar in form to that used with the Request register. See Fig. 5.13 for instruction addressing.

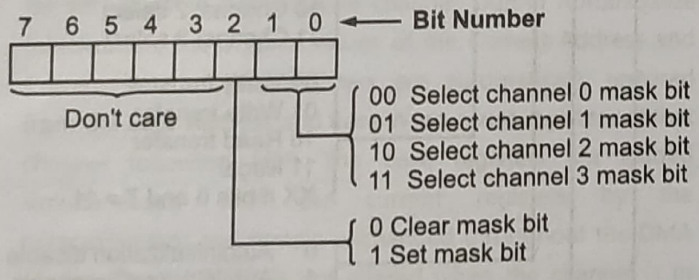


Fig. 5.13 : Mask register

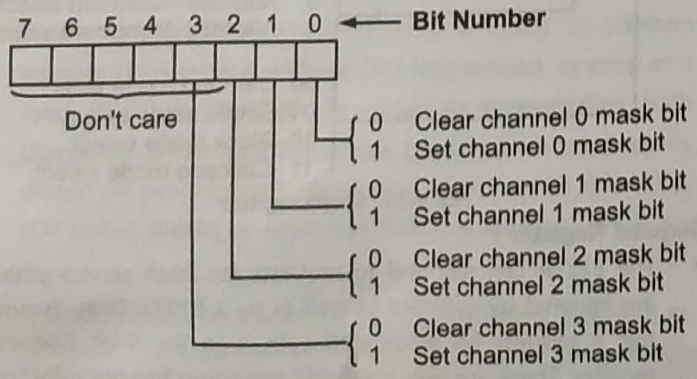


Fig. 5.14

Table 5.9 : Definition of Register Codes

Register	Operation	Signals						
		$\overline{CS}$	$\overline{IOR}$	$\overline{IOW}$	A_3	A_2	A_1	A_0
Command	Write	0	1	0	1	0	0	0
Mode	Write	0	1	0	1	0	1	1
Request	Write	0	1	0	1	0	0	1
Mask	Set/Reset	0	1	0	1	0	1	0
Mask	Write	0	1	0	1	1	1	1
Temporary	Read	0	0	1	1	1	0	1
Status	Read	0	0	1	1	0	0	0

Status Register :

- The Status register is available to be read out of the 8237A by the microprocessor. It contains information about the status of the devices at this point. This information includes which channels have reached a terminal count and which channels have pending DMA requests. Bits 0-3 are set every time a TC is reached by that channel or an external EOP is applied.
- These bits are cleared upon Reset and on each Status Read. Bits 4-7 are set whenever their corresponding channel is requesting service.

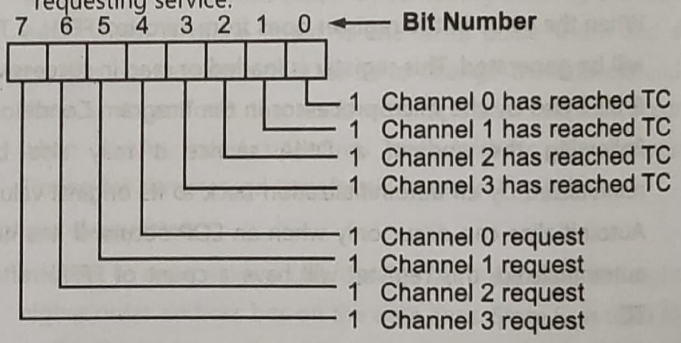


Fig. 5.15

Temporary Register :

- The Temporary register is used to hold data during memory-to-memory transfers. Following the completion of the transfers, the last word moved can be read by the microprocessor in the Program Condition.
- The Temporary register always contains the last byte transferred in the previous memory-to-memory operation, unless cleared by a Reset.

Software Commands :

- These are additional special software commands which can be executed in the Program Condition. They do not depend on any specific bit pattern on the data bus. The three software commands are :
  1. **Clear First/Last Flip-Flop :** This command must be executed prior to writing or reading new address or word count information to the 8237A. This initializes the flip-flop to a known state so that subsequent accesses to register contents by the microprocessor will address upper and lower bytes in the correct sequence.

2. **Master Clear :** This software instruction has the same effect as the hardware Reset. The Command, Status, Request, Temporary, and Internal First/Last Flip-Flop registers are cleared and the Mask register is set. The 8237A will enter the Idle cycle.
3. **Clear Mask Register :** This command clears the mask bits of all four channels, enabling them to accept DMA requests.

Table 5.10 : Software Command Codes

Signals						Operation
A_3	A_2	A_1	A_0	$\overline{IOR}$	$\overline{IOW}$	
1	0	0	0	0	1	Read Status Register
1	0	0	0	1	0	Write Command Register
1	0	0	1	0	1	Illegal
1	0	0	1	1	0	Write Request Register
1	0	1	0	0	1	Illegal
1	0	1	0	1	0	Write Single Mask Register Bit
1	0	1	1	0	1	Illegal
1	0	1	1	1	0	Write Mode Register
1	1	0	0	0	1	Illegal
1	1	0	0	1	0	Clear Byte Pointer Flip/Flop
1	1	0	1	0	1	Read Temporary Register
1	1	0	1	1	0	Master Clear
1	1	1	0	0	1	Illegal
1	1	1	0	1	0	Clear Mask Register
1	1	1	1	0	1	Illegal
1	1	1	1	1	0	Write All Mask Register Bits

5.9 THE 8086 INTERRUPTS

(A) Hardware Interrupts (External Interrupts) :

The Intel microprocessors **Support Hardware Interrupts** through :

- Two pins that allow interrupt requests, INTR (interrupt request) and NMI (Non-maskable interrupt).
- One pin that acknowledges, INTA, the interrupt requested on INTR.

5.9.1 INTR

- **INTR** is a maskable hardware interrupt. This interrupt can be enabled/disabled using STI/CLI (set interrupt/clear interrupt) instructions or using method of updating the FLAGS register with the help of the POPF instruction.
- When an interrupt occurs
  - The processor decrements the stack pointer by 2 and stores flag registers into stack.
  - The processor disables further interrupts by clearing interrupt Flag (IF) in the flag register. It also resets the Trap Flag (TF).
  - It again decrements the stack pointer by 2 and pushes the contents of the current code segment register onto the stack.
  - It again decrements the stack pointer by 2 and pushes the contents of the instruction pointer onto the stack.
  - After saving the contents of CS : IP and disabling further interrupts it jumps to the start of the procedure which is written for the interrupt generated.

All above steps can be shown as given in the following diagram.

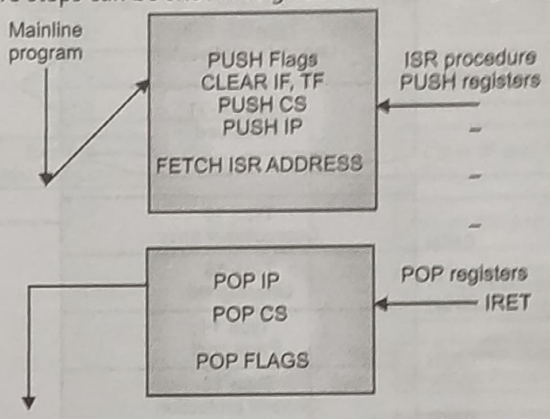


Fig. 5.16

- In the starting of the procedure contents of all register corresponding to the current program under execution are stored (pushed) onto the stack.
- At the end of the procedure the contents are again poped (retrieved) back to all registers and IRET is written.
- The IRET instruction at the end of the procedure returns the execution to the main program.
- The contents of IP are poped (restored) first and then contents of CS are poped (restored).
- The flag register is also restored.

Further execution of the main program continues.

5.9.2 NMI

- **NMI** is a non-maskable interrupt.
- This Interrupt is processed in the same way as the INTR interrupt.
- Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h.
- This interrupt has higher priority than the maskable interrupt.

(B) Software Interrupts (Internal Interrupts and Instructions)

Software Interrupts can be caused by the instruction INT. the INT instruction can be used to do one of the 256 interrupts (Type 0-255). The interrupt type is specified by the number as a part of the instruction.

For example, INT 0 instruction can be used to send the execution to a divide by zero interrupt service routine (ISR).

5.10 THE INTERRUPT VECTOR TABLE (IVT)

The Interrupt vector table is the link between an interrupt type code and the ISR address. In 8086, the first 1 K byte of memory is kept aside as a table for storing the starting addresses of interrupt service procedures. As the table saves the starting addresses of the ISR it is called Interrupt Vector Table(IVT). Fig. 5.17 shows how the 256 interrupt vectors are arranged in the table in the table in the memory.

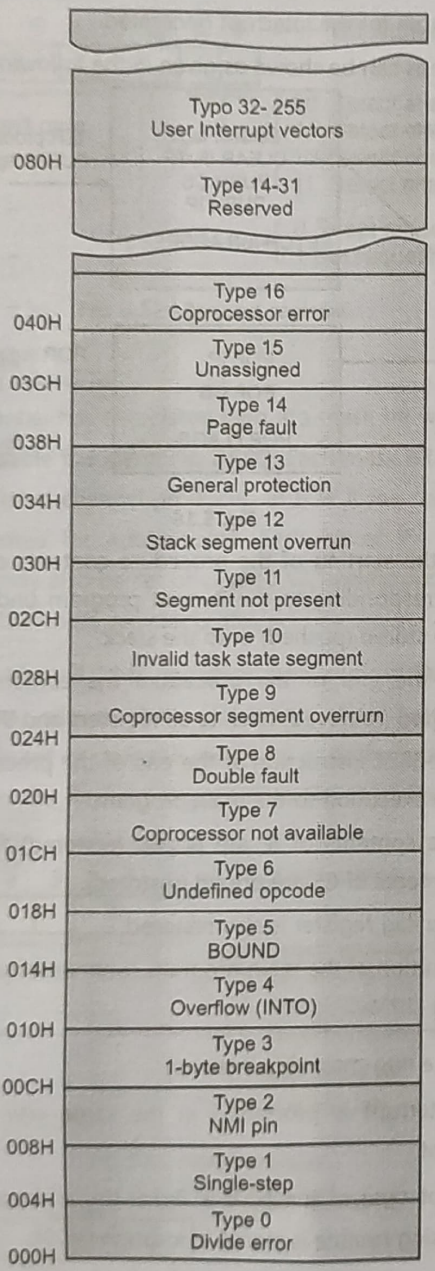


Fig. 5.17 : Interrupt vector table

TYPE -0 to TYPE -4 Interrupts :

These lowest five types of interrupts are dedicated to specific tasks as given below.

INT 00 (Divide error) - TYPE 0

- INT00 is invoked by the microprocessor whenever there is an attempt to divide a number by zero.
- ISR is responsible for displaying the message "Divide Error" on the screen

INT 01 (Single Step Interrupt) TYPE 1

- For single stepping the trap flag must be 1.
- After execution of each instruction, 8086 automatically jumps to 00004H to fetch 4 bytes for CS : IP of the ISR.
- The job of ISR is to dump the registers on to the screen

INT 02 (Non-Maskable Interrupt) TYPE 2

- Whenever NMI pin of the 8086 is activated by a high signal (5v), the CPU Jumps to physical memory location 00008 to fetch CS :IP of the ISR associated with NMI.
- This is the ISR for the Non-maskable Interrupt.

INT 03 (Break Point) TYPE 3

- A break point is used to examine the CPU registers and memory after the execution of a group of Instructions.
- The main use of the INT 03 instruction is to implement the breakpoint in the program.

INT 04 (Signed Number Overflow) TYPE 4

- If the signed result of an arithmetic operation is too large it will set overflow flag (OF).
- There are two ways to handle the overflow error in the program.
- One method is to put Jump if overflow instruction (JO) immediately after the arithmetic instruction. If overflow occurs the execution of program will be transferred to the address specified by the JO instruction. Here the error subroutine can be put to handle the overflow.
- The second method to handle the overflow is to put Interrupt on overflow instructo INTO immediately after the arithmetic instruction. This instruction will simply act as NOP. However if overflow occurs 8086 will execute INT 04 i.e. TYPE -4 interrupt.

INT 05 to INT 31 (TYPE 5 to TYPE 31) :

These interrupts are reserved by Intel for use in more complex microprocessors such as 80286, 80386, 80486.

INT 32 TO INT 255 (TYPE 32 TO TYPE 255) :

The upper 224 interrupt types form 32 to 255 are available for the user to use for hardware or software interrupts.

5.11 INTERRUPT SERVICE ROUTINES : (THE 8086 INTERRUPT)

At the end of each instruction cycle the 8086 checks for any interrupt request is there, if any interrupt occurs it will complete the current instruction. It will follow the following set of sequence at the end of each instruction.

- The processor will first complete the execution of the current instruction.
- It will check for any interrupt if it is not present then it will check for NMI, if NMI is also not there it will check for INTR.

- If INTR is not present it will check for the TF (Trap Flag) for single step.
- If TF is not there then it will switch to the execution of the next instruction.
- If you observe the flow chart all interrupts except INTR will come to the same point of pushing flags and follow the same sequence of operation.
- In case of INTR the processor will check for IF, if it is '0' no service will be given to INTR.
- If IF is '1' then it well send Interrupt Acknowledgement and read the code for the ISR and the following common sequence of operation will be followed for the execution of the ISR.
  - The processor decrements the stack pointer by 2 and stores flag registers into stack.
  - The processor disables further interrupts by clearing Interrupt Flag (IF) in the flag register. It also resets the Trap Flag (TF).
  - It again decrements the stack pointer by 2 and pushes the contents of the current code segment register onto the stack.
  - It again decrements the stack pointer by 2 and pushes the contents of the instruction pointer onto the stack.
  - The processor will CALL the ISR.
- With this CALL the new values of the CS :IP will be loaded from the IVT.
- Before transferring the control to the new ISR it will again check for any NMI if NMI it will jump back.

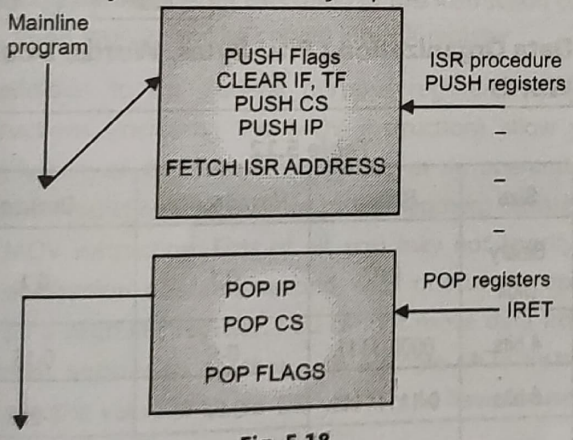


Fig. 5.18

- If no NMI it will again check the TEMP, it TF is '1' again the microprocessor will jump back.
  - IF No TF flag i.e. TF='0' then the processor will execute the user interrupt procedure (ISR written by the user)
  - At the end of ISR the user will write IRET instruction i.e. it will return form an interrupt.
  - In response to the IRET instruction the following steps will be performed
- The IRET instruction at the end of the procedure returns the execution to the main program.
- The contents of IP are poped (restored) first and then contents of CS are poped (restored).
- The flag register is also restored.
- Further execution of the main program continues.

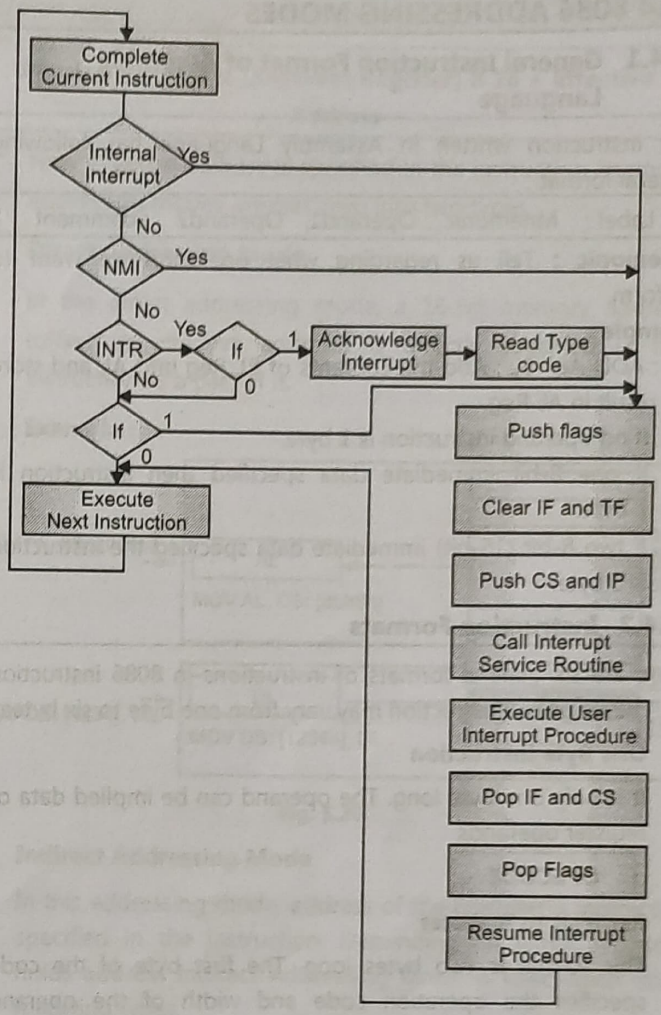


Fig. 5.19

5.12 INTERRUPT PRIORITIES

In 8086 the priorities of the software interrupts is higher which is followed by the NMI, INTR and single step

Table 5.11

Interrupt	Priority
Divide Error, INT N, INT 0	Highest
NMI	↓
INTR	↓
Single Step	Lowest

5.13 8086 INSTRUCTION SET AND PROGRAMMING

- The microprocessor is having specific format of instructions as well as there are different addressing modes available for it.
- This unit is concerned with the instruction set, instruction format and different addressing mode of 8086 micro-processor.
- This unit also focuses on the assembly language programs, C programs.
- In this unit we are also going to study Assemblers and Compilers, Programming and Debugging Tools.

5.14 8086 ADDRESSING MODES

5.14.1 General Instruction Format of Assembly Language

Any instruction written in Assembly Language has following general format.

Label :	Mnemonic	Operand1, Operand2	; comment

Mnemonic : Tell us regarding what operation we want to perform.

Example :

Up : ADD AL, BL ; Add the contents of BL Reg into AL and store the result in AL Reg.

- If no operand instruction is 1 byte.
- If one 8-bit immediate data specified then instruction is 2 byte.
- If two 8-bit (16-bit) immediate data specified the instruction is 3 byte.

5.14.2 Instruction Formats

There are six general formats of instructions in 8086 instruction set. The length of instruction may vary from one byte to six bytes.

1. One Byte Instruction

It is only one byte long. The operand can be implied data or register operands

OPECODE

2. Register to Register

This format is two bytes long. The first byte of the code specifies the operation code and width of the operand specified by w bit. The second byte of the code shows the register operands and R/M field as shown below.

D_7	D_1 D_0	D_7 D_6	D_5 D_4 D_3	D_2 D_1 D_0
OPCODE	W	1 1	REG	R/M

The register represented by the REG field is one of the operands. The R/M field specifies another register or memory location, i.e. the other operand.

3. Register to/from Memory with No Displacement

This format is also 2 bytes long and similar to register to register format expect for the MOD filed as shown.

D_7	D_1 D_0	D_7 D_6	D_5 D_4 D_3	D_2 D_1 D_0
OPCODE	W	MOD	REG	R/M

The MOD field shows the mode of addressing.

4. Register to/from Memory with Displacement

This type of instruction format contains one or two additional bytes for displacement along with 2-byte format of the register to/from memory without displacement. The format is as shown below :

D_7 D_0	D_7 D_6	D_5 D_4 D_3	D_2 D_1 D_0	D_7 D_0	D_7 D_0
OPCODE	MOD	REG	R/M	Lower Byte of Disp	Higher Byte of Disp

5. Immediate Operand to Register

In this format, the first byte as well as the 3-bits from the second byte is used for REG field in case of register to register format is used for OPCODE. It also contains one or two bytes of immediate data. The complete instruction format is as shown below

D_7 D_0	D_7 D_6	D_5 D_4 D_3	D_2 D_1 D_0	D_7 D_0	D_7 D_0
OPCOD E	1 1	OPCODE	R/M	Lower Byte of Disp	Higher Byte of Disp

6. Immediate Operand to Memory with 16-Bit Displacement

This type of instruction format requires 5 or 6 bytes for coding. The first 2 bytes contain the information regarding OPCODE, MOD and R/M fields. The remaining 4-bytes contain 2 bytes of displacement and 2 bytes of data as shown below.

D_7 D_0	D_7 D_6	D_5 D_4 D_3	D_2 D_1 D_0	D_7 D_0	D_7 D_0
OPCODE	MOD	REG	R/M	Lower Byte of Disp	Higher Byte of Disp

D_7 D_0	D_7 D_0
Lower Byte of data	Higher Byte of Data

5.14.3 Data Organization : Bits, Bytes, Words, Double-Words

Table 5.12

Name	Size	Binary	Hexadecimal	Decimal
Bit	Binary Digit	0,1	0,1	0,1
Nibble	4 bits	0000-1111	0-F	0-15
Byte	8 bits	0-11111111	0-FF	0-255
Word	16 bits	0-(16 '1's)	0-FFFF	0-65535
Double Word	32 bits	0-(32 '1's)	0-FFFFFFFF	0-4,294,967,295

5.15 ADDRESSING MODES

Addressing mode indicates a way of locating data or operands. The addressing modes describe the types of operands and the way they are accessed for executing an instruction.

The operand addressing modes of the 8086 :

- Immediate Addressing Mode.
- Register Addressing Mode.
- Direct Addressing Mode.

- Indirect Addressing Mode : Sub categories of indirect addressing mode are :
  - Based Addressing Mode
  - Index Addressing Mode } Register Indirect Addressing Mode
  - Base + Index Addressing Mode
  - Base + Displacement Addressing Mode
  - Index + Displacement Addressing Mode
  - Base + Index + Displacement Addressing Mode

Explanation of Each Addressing Mode with Example

1. Immediate Addressing Mode (Bit Direct Addressing)

- Immediate addressing mode is when one of the operands is "immediately" located after the opcode. It is more correct to say that the operand is part of the instruction.

  Example : MOV AX, 0054H

  Explanation : After execution of this instruction, the contents of AX register will be 0054H.

2. Register Addressing Mode

- In this addressing mode, the data is stored in a register and it is referred using the particular register.
- **Note :** All the registers except IP may be used in this mode.

  Example : MOV AX, BX

  Explanation : After execution of this instruction contents of BX register will be copied into AX register.

- In addition to the general purpose registers, many 8086 instructions (including the MOV instruction) allow you to specify one of the segment registers as an operand. There are two restrictions on the use of the segment registers with the MOV instruction. First of all, you may not specify CS as the destination operand; second, only one of the operands can be a segment register. You cannot move data from one segment register to another with a single MOV instruction. To copy the value of CS to DS, you would have to use some sequence like :

```
MOV   AX, CS
MOV   DS, AX
```

Implied Addressing Mode (Bit Inherent Addressing) :

- In this mode, the operands are implied and are hence not specified in the instruction.
- **Example :** STC

  This sets the carry flag.

8086 Memory Addressing Modes

- When numbers stored in memory location are accessed, 8086 generates 20-bit physical address.

We know that,

Physical Address = [Segment Register] X 16 + Effective Address

How effective address is specified in the instruction, memory addressing modes are classified into two types

1. Direct Addressing Mode

In the direct addressing mode, a 16-bit memory address (offset or effective address) is directly specified in the instruction as a part of it.

Example :

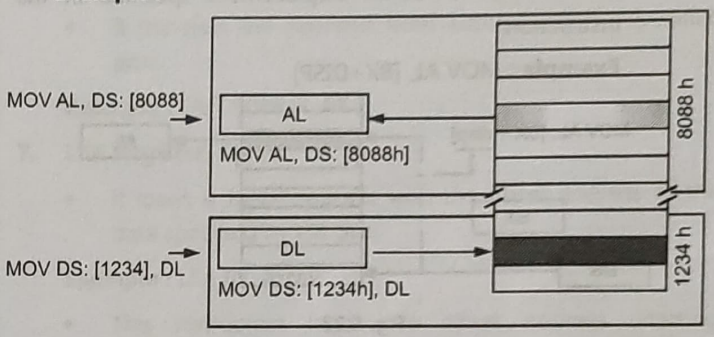


Fig. 5.20

2. Indirect Addressing Mode

In this addressing mode, address of the operand is indirectly specified in the instruction. Depending on which register holds address Indirect Addressing Mode is subdivided into following types.

(a) Based Addressing Mode

In this addressing mode, the offset address of data is in BX register. The default segment register is either DS or ES.

Example : MOV AL, [BX]

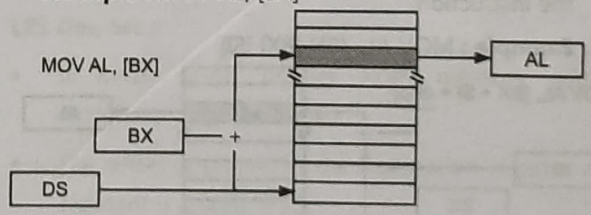


Fig. 5.21

(b) Indexed Addressing Mode

In this addressing mode, the offset address of data is in either SI or DI register. The default segment register is either DS or ES.

Example : MOV AL, [SI]

(c) Base + Index Addressing Mode

The effective address of data is formed, in this addressing mode, by adding contents of a base register to the contents of an index register. The default segment register is either DS or ES.

Example : MOV AL, [BX] [SI]

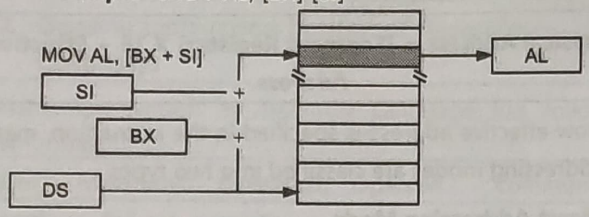


Fig. 5.22

(d) Base + Displacement Addressing Mode

The effective address of data is formed, in this addressing mode, by adding contents of a base register to the 8-bit or 16-bit displacement specified in the instruction.

Example : MOV AL, [BX+DISP]

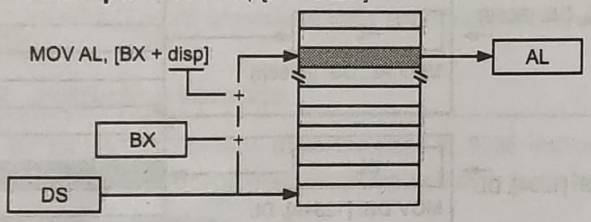


Fig. 5.23

(e) Index + Displacement Addressing Mode

The effective address of data is formed, in this addressing mode, by adding contents of a index register to the 8-bit or 16-bit displacement specified in the instruction.

Example : MOV AL, 30H [SI]

(f) Base + Index + Displacement Addressing Mode

The effective address of data is formed, in this addressing mode, by adding contents of base, index register to the 8-bit or 16-bit displacement specified in the instruction.

Example : MOV AL, 30H [BX] [SI]

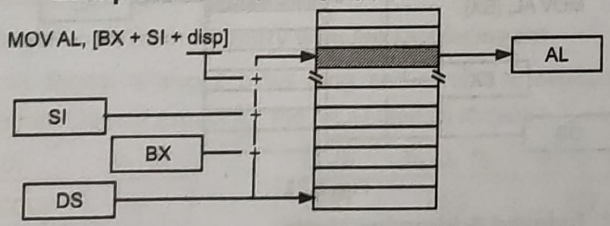


Fig. 5.24

5.16 INSTRUCTION TIMINGS

- Some instructions require additional clock cycles due to a "Next Instruction Component" identified by a "+m" in the instruction clock cycle listings. This is due to the prefetch queue being purge on a control transfers. Below is the general rule for calculating "m" :

88/86 not applicable

286 "m" is the number of bytes in the next instruction

386 "m" is the number of components in the next instruction (the instruction coding (each byte), plus the data and the displacement are all considered components)

Table 5.13

Description	Clock Cycles
Displacement	6
Base or Index (BX,BP,SI,DI)	5
Displacement+(Base or Index)	9
Base+Index (BP+DI,BX+SI)	7
Base+Index (BP+SI,BX+DI)	8
Base+Index+Displacement (BP+DI,BX+SI)	11
Base+Index+Displacement (BP+SI+disp,BX+DI+disp)	12

- Add 4 cycles for word operands at odd addresses
- Add 2 cycles for segment override
- 80188/80186 timings differ from those of the 8088 / 8086 / 80286
- All timings are for best case and do not take into account wait states, instruction alignment, the state of the prefetch queue, DMA refresh cycles, cache hits/misses or exception processing.
- To convert clocks to nanoseconds divide one microsecond by the processor speed in MegaHertz :

(1000MHz/(n MHz)) = X nanoseconds

5.17 CLASSIFICATION OF INSTRUCTION SET

1. Data Transfer Instructions

These instructions are used to transfer/copy data from source to destination. The operand can be a constant, memory location, register or I/O port address.

The source may be any one of the segment register or either general or special Function Register, Immediate Number, memory Location.

Note : This Category of Instructions do not Modify Flag Register

2. Arithmetic Logical Instructions

All the instructions performing arithmetic, increment in operand, decrement the operand belong to this category.

Note : This category of instructions do modify flag register

3. Logical Instructions/(Bit Manipulation Instructions)

These instructions are used at the bit level. These instructions can be used for :

- Testing a zero bit
- Set or reset a bit
- Shift bits across registers

Note : This category of instructions do modify flag register

4. **Program Execution Transfer Instructions**

These instructions transfer control of execution to the specified address. All the call, jump, interrupt and return instructions belong to this class.

5. **String Instructions**

These instructions involve various string manipulation operations like load, move, scan, compare, store etc. These instructions are only to be operated onto the strings.

6. **Processor Control Instructions**

These instructions control the machine status.

5.17.1 Data Transfer Instructions

1. **MOV Des, Src :**
   - **Src** operand can be register, memory location or immediate operand.
   - **Des** can be register or memory operand.
   - Both **Src** and **Des** cannot be memory location at the same time.
   - Both **Src** and **Des** operand Size should match.

Example :

MOV CX, 037A H => Immediate number 057AH is transferred to CX Reg.

MOV AL, BL => Contents of BL Reg are copied into AL Register

MOV BX, [0301H] => Assume DS = 1000H, then BL Register will be loaded with contents of memory location address [DS : Offset] i.e. [1000H : 0301H] and BH Register will be loaded with contents of memory location address [1000H : 0302H].

2. **PUSH Operand :**
   - It pushes the operand onto top of stack.
   - 16-bit data is transferred on to the stack.
   - When Data is pushed SP pointer is decremented first and data is stored.

Example : PUSH BX

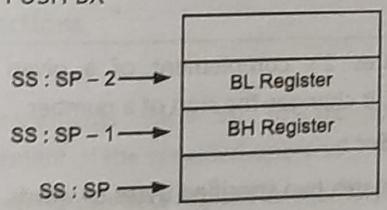


Fig. 5.25

3. **POP Des :**
   - It pops the operand from top of stack to **Des**.
   - **Des** can be a general purpose register, segment register (except CS) or memory location.
   - When data is popped, data is retrieved first and then SP is incremented.

Example : POP AX.

4. **XCHG Des, Src :**
   - This instruction exchanges **Src** with **Des**.
   - It cannot exchange two memory locations directly.

Example : XCHG DX, AX.

Exchange the contents of AX register with DX register.

5. **IN Accumulator, Port Address :**
   - It transfers the operand from specified port to accumulator register.

Example : IN AX, 0028 H.

6. **OUT Port Address, Accumulator :**
   - It transfers the operand from accumulator to specified port.

Example : OUT 0028 H, AX.

7. **LEA Register, Src :**
   - It loads a 16-bit register with the offset address of the data specified by the **Src**.

Example : LEA BX, **array**
   - This instruction loads the offset address (starting address) of variable **array** into the BX register.

8. **LDS Des, Src :**
   - It loads 32-bit pointer from memory source to destination register and DS.
   - The offset is placed in the destination register and the segment is placed in DS.
   - To use this instruction the word at the lower memory address must contain the offset and the word at the higher address must contain the segment.

Example : LDS BX, [0301 H]

9. **LES Des, Src :**
   - It loads 32-bit pointer from memory source to destination register and ES.
   - The offset is placed in the destination register and the segment is placed in ES.
   - This instruction is very similar to LDS except that it initializes ES instead of DS.

Example : LES BX, [0301 H]

10. **LAHF :**
   - It copies the lower byte of flag register to AH.
   - This instruction can be used to read the lower byte of flag register into AH registers.
   - Based on this, decision can be taken from the contents of AH register.

11. **SAHF :**
    - It copies the contents of AH to lower byte of flag register.
    - This instruction can be used to modify lower byte of flag register as per initial required settings stored in the AH register.
12. **PUSHF :** Pushes flag register onto top of stack.
13. **POPF :** Pops the stack top onto flag register.

5.17.2 Arithmetic Instructions

1. **ADD Des, Src :**
    - It adds a byte to byte or a word to word.
    - It effects AF, CF, OF, PF, SF, ZF flags.

    Example :
    - **ADD AL, 74H :** It adds contents of AL Register with number 74H and stores addition into AL Register.
    - **ADD AX, DX :** It adds contents of AX Register with DX register and stores addition into AX Register.
    - **ADD AL, [BX] :** It adds contents of AL Register with memory location contents pointed by [DS : BX] and stores addition into AL Register
2. **ADC Des, Src :**
    - It adds the two operands with CF.
    - It effects AF, CF, OF, PF, SF, ZF flags.
    - Similar to add instruction it adds two operands with previously generated carry.

    Example :

    ADC AL, 74H

    ADC DX, AX

    ADC AL, [BX]
3. **SUB Des, Src :**
    - It subtracts a byte from byte or a word from word.
    - It effects AF, CF, OF, PF, SF, ZF flags.
    - For subtraction, CF acts as borrow flag.

    Example :
    - **SUB AL, 74H :** It subtracts number 74H from contents of AL Register and stores subtraction into AL Register.
    - **SUB AX, DX :** It subtracts contents of DX register from AX and stores subtraction into AX Register.
    - **SUB AX, [BX] :** It subtracts contents of memory location pointed by [DS : BX] from AX and stores subtraction into AX Register
4. **SBB Des, Src :**
    - It subtracts the two operands and also the borrow from the result.
    - It effects AF, CF, OF, PF, SF, ZF flags.

    Example,

    SBB AL, 74H

    SBB AX, DX

    SBB AX, [BX]
5. **INC Src :**
    - It increments the byte or word by one. The operand can be a register or memory location.
    - It effects AF, OF, PF, SF, ZF flags. **CF is not effected.**

    Example : INC AX → [AX] = [AX] + 1
6. **DEC Src :**
    - It decrements the byte or word by one. The operand can be a register or memory location.
    - It effects AF, OF, PF, SF, ZF flags. **CF is not effected.**

    Example : DEC AX
7. **AAA (ASCII Adjust after Addition) :**
    - The data entered from the terminal is in ASCII format. In ASCII, 0 – 9 are represented by 30H – 39H.
    - This instruction allows us to add the ASCII codes. This instruction does not have any operand.
8. **Other ASCII Instructions :**
    - **AAS** (ASCII Adjust after Subtraction)
    - **AAM** (ASCII Adjust after Multiplication)
    - **AAD** (ASCII Adjust Before Division)
9. **DAA (Decimal Adjust after Addition)**
    - It is used to make sure that the result of adding two BCD numbers is adjusted to be a correct BCD number. It only works on AL register.
10. **DAS (Decimal Adjust after Subtraction)**
    - It is used to make sure that the result of subtracting two BCD numbers is adjusted to be a correct BCD number. It only works on AL register.
11. **NEG Src :**
    - It creates 2's complement of a given number. That means, it changes the sign of a number.
12. **CMP Des, Src :**
    - It compares two specified bytes or words.
    - The **Src** and **Des** can be a constant, register or memory location.
    - Both operands cannot be a memory location at the same time.
    - The comparison is done simply by internally subtracting the source from destination.
    - The value of source and destination does not change, but the flags are modified to indicate the result.

Table 5.14

Condition	CY	Z
Des > Src	0	0
Des < Src	1	0
Des = Src	0	1

13. MUL Src :

- It is an unsigned multiplication instruction. It multiplies two bytes to produce a word or two words to produce a double word.

  AX = AL * Src

  DX : AX = AX * Src

- This instruction assumes one of the operand in AL or AX. Src can be a register or memory location.

14. IMUL Src :

- It is a signed multiplication instruction.

15. DIV Src :

- It is an unsigned division instruction. It divides word by byte or double word by word.
- The operand is stored in AX, divisor is Src and the result is stored as : AH = remainder AL = quotient.

16. IDIV Src :

- It is a signed division instruction.

17. CBW (Convert Byte to Word) :

- This instruction converts byte in AL to word in AX.
- The conversion is done by extending the sign bit of AL throughout AH.

18. CWD (Convert Word to Double Word) :

- This instruction converts word in AX to double word in DX : AX.
- The conversion is done by extending the sign bit of AX throughout DX.

5.17.3 Logical Instructions/Bit Manipulation Instructions

1. NOT Src :

- It complements each bit of Src to produce 1's complement of the specified operand.
- The operand can be a register or memory location.

2. AND Des, Src :

- It performs AND operation of Des and Src bitwise.
- **Src** can be immediate number, register or memory location.
- **Des** can be register or memory location. Both operands cannot be memory locations at the same time.
- CF and OF become zero after the operation. PF, SF and ZF are updated.

3. OR Des, Src :

- It performs OR operation of Des and Src bitwise.
- **Src** can be immediate number, register or memory location.
- **Des** can be register or memory location.
- Both operands cannot be memory locations at the same time.
- CF and OF become zero after the operation. PF, SF and ZF are updated.

4. XOR Des, Src :

- It performs XOR operation of **Des** and **Src** bitwise.
- **Src** can be immediate number, register or memory location.
- **Des** can be register or memory location.
- Both operands cannot be memory locations at the same time.
- CF and OF become zero after the operation.
- PF, SF and ZF are updated.

5. SHL Des, Count :

- It shift bits of byte or word left, by count.
- It puts zero(s) in LSBs.
- MSB is shifted into carry flag.
- If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
- However, if the number of bits to be shifted is more than 1, then the count is put in CL register.

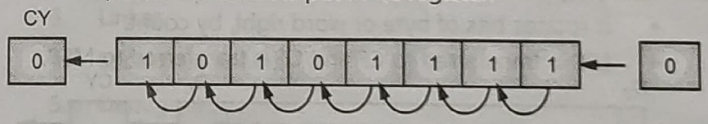


Fig. 5.26 (a)

- After Execution : of SHL AL, 01H

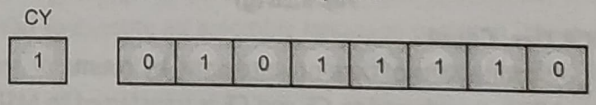


Fig. 5.26 (b)

6. SHR Des, Count :

- It shift bits of byte or word right, by count.
- It puts zero(s) in MSBs. LSB is shifted into carry flag.
- If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count. However, if the number of bits to be shifted is more than 1, then the count is put in CL register.

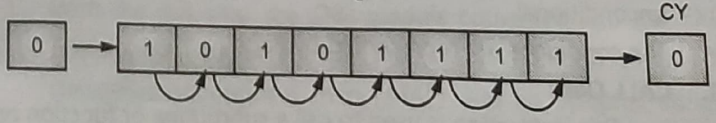


Fig. 5.26 (c)

- After Execution of Instruction

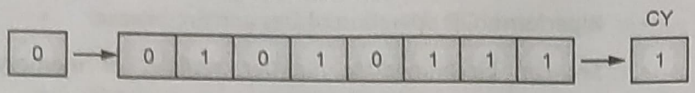


Fig. 5.26 (d)

7. **ROL Des, Count :**
   - It rotates bits of byte or word left, by count.
   - MSB is transferred to LSB and also to CF.
   - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
   - However, if the number of bits to be shifted is more than 1, then the count is put in CL register.

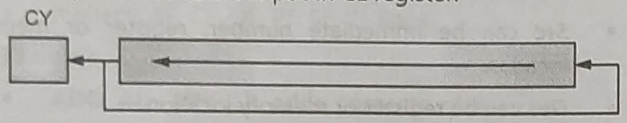


Fig. 5.26 (e)

8. **ROR Des, Count :**
   - It rotates bits of byte or word right, by count.
   - LSB is transferred to MSB and also to CF.
   - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count. However, if the number of bits to be shifted is more than 1, then the count is put in CL register.

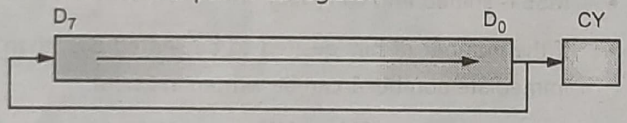


Fig. 5.26 (f)

9. **RCL Des, Count**
   - It rotates bits of byte or word right, by count.
   - LSB is transferred to CF and CF is transferred to MSB.

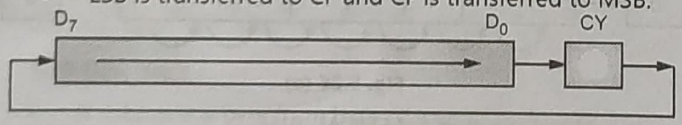


Fig. 5.26 (g)

10. **RCR Des, Count**
    - It rotates bits of byte or word right, by count.
    - LSB is transferred to CF and CF is transferred to MSB.

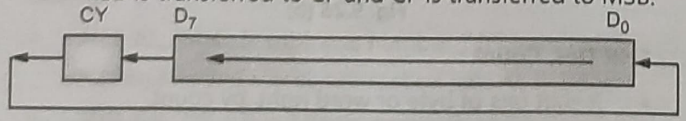


Fig. 5.26 (h)

5.17.4 Program Execution Transfer Instructions

- These instructions cause change in the sequence of the execution of instruction.
- This change can be through a condition or sometimes unconditional.
- The conditions are represented by flags.

1. **CALL Des :**
   - This instruction is used to call a subroutine or function or procedure.
   - The address of next instruction after CALL is saved onto stack.
2. **RET :**
   - It returns the control from procedure to calling program.
   - Every CALL instruction should have a RET.
3. **JMP Des :**
   - This instruction is used for unconditional jump from one place to another.
4. **JXX Des (Conditional Jump) :**
   - All the conditional jumps follow some conditional statements or any instruction that affects the flag.

Table 5.15 : Conditional Jump Instruction

Mnemonic	Meaning	Jump Condition
JA	Jump if Above	CF = 0 and ZF = 0
JAE	Jump if Above or Equal	CF = 0
JB	Jump if Below	CF = 1
JBE	Jump if Below or Equal	CF = 1 or ZF = 1
JC	Jump if Carry	CF = 1
JE	Jumps if Equal	ZF = 1
JNC	Jump if Not Carry	CF = 0
JNE	Jump if Not Equal	ZF = 0
JNZ	Jump if Not Zero	ZF = 0
JPE	Jump if Parity Even	PF = 1
JPO	Jump if Parity Odd	PF = 0
JZ	Jump if Zero	ZF = 1

Loop Des :
- This is a looping instruction.
- The number of times looping is required is placed in the CX register.
- With each iteration, the contents of CX are decremented.
- ZF is checked whether to loop again or not.

5.17.5 String Instructions

- String in assembly language is just a sequentially stored bytes or words.
- There are very strong set of string instructions in 8086. By using these string instructions, the size of the program is considerably reduced.

1. **CMPS Des, Src :**
   - It compares the string bytes or words.
2. **SCAS String :**
   - It scans a string.
   - It compares the String with byte in AL or with word in AX.
3. **MOVS / MOVSB / MOVSW :**
   - It causes moving of byte or word from one string to another.
   - In this instruction, the source string is in Data Segment and destination string is in Extra Segment. SI and DI store the offset values for source and destination index.

4. **REP (Repeat) :**
   - This is an instruction prefix.
   - It causes the repetition of the instruction until CX becomes zero.

   Example : REP MOVSB STR1, STR2
   - It copies byte by byte contents.
   - REP repeats the operation MOVSB until CX becomes zero.

5.17.6 Processor Control Instructions

- These instructions control the processor itself. 8086 allows to control certain control flags that causes the processing in a certain direction processor synchronization if more than one microprocessor attached.
  - **STC :** It sets the carry flag to 1.
  - **CLC :** It clears the carry flag to 0.
  - **CMC :** It complements the carry flag.
  - **STD :** It sets the direction flag to 1. If it is set, string bytes are accessed from higher memory address to lower memory address.
  - **CLD :** It clears the direction flag to 0. If it is reset, the string bytes are accessed from lower memory address to higher memory address.

5.18 ASSEMBLY LANGUAGE

- To run any program the microcomputer must have program stored in binary form in successive memory location. There were two main classes of programming languages that is high level and low level. The e.g. of high level language are C, BASIC, JAVA etc; which uses powerful commands each of which may generate many machine language instructions.
- On the other hand program written in low level assembly language is a code symbolic instruction, each of which generate only one machine instruction.
- The program with assembly language is written with strict set of rules; an editor is used for entering it into the computer as a file and then the assembler translator program to read the file and to convert it into machine code.

5.18.1 Advantages of Assembly Language

- It gives more control over handling particular hardware requirements.
- With this language it is possible to generate smaller, more compact executable modules.
- The code written in this language executes faster.

5.18.2 Standard Assembly Language Program Format

- Assembly language is having different features such as program comments, reserve words, identifiers, statements and directives. These features provide the basic rules and framework for the language.
- The Assembly language instructions are written in a special format such as :

Table 5.16 : Assembly Language Instructions Format

Labels	Mnemonics	Operand 1	Operand 2	Comments
Start :	MOV	AX,	@data	; Initialization of data segment
	MOV	DS,	AX	; data segment initialization

- Table 5.16 shows the different fields used for writing Assembly Language program instructions. The *label* column is the name used to represent an address referred to in a jump or call instruction. A label is followed by a colon (:).
- The **Mnemonics** column contains opcode mnemonics for the instruction. It is also called as operation fields which enlighten the type of operation it is such as MOV, ADD, SUB, MUL etc. The operand(s) column contains registers, memory locations or data used by instructions. A comments column utilized to describe the function of the instruction for future reference. It starts with semicolon (;).

5.19 ASSEMBLY LANGUAGE PROGRAM DEVELOPMENT TOOLS

To Develop an Assembly Language Program Following Programming Tools are Used :

1. Editor
2. Assembler
3. Linker
4. Locator or Loader
5. Debugger
6. Emulator

Before discussing the detailing of each tool lets first see the steps required to execute an assembly language program. The program written in assembly language is a just a text file that cannot execute;

Following Steps Required for Executing it :

1. An editor is used to create a file containing assembly language statement for your program.
2. Next the assembly step translates the source code into object code and generating an intermediate .OBJ (object) file or module. It also create an optional listing (.LST) and an optional file for use as a cross reference.
3. With the link step the .OBJ module converted into the .EXE (executable) machine code module. In addition to this it generates an optional map (.MAP) and an optional library (.LIB) file.

5.20 EXAMPLES OF ASSEMBLY LANGUAGE

1. Simple Program Showing use of Conditional Jump Instruction

```
ORG 100h
MOV AL, 25              ; set AL to 25.
MOV BL, 10              ; set BL to 10.
CMP AL, BL              ; compare AL - BL.
JNE not_equal           ; jump if AL <> BL (ZF = 0).
JMP equal
not_equal : MOV CL, 00H ; clear the CL register
JMP stop                ; jump to stop.
equal :   MOV CL, 0FFH; set all bits of CL register to 1
stop :   END
```

This program checks whether contents of AL and BL register are equal or not.

- If two numbers are not equal CL register is cleared and if two numbers are equal CL register is set to FFh.

2. Write a Program to Convert a Lowercase Character into Uppercase. Accept the Character from User.

```
TITLE "Program to convert lowercase letter to uppercase"
.MODEL SMALL ; this defines the memory model
.STACK 100 ; define a stack segment of 100 bytes
.DATA        ; this is the data segment
MSG1 DB 'Enter a lower case letter : $'
MSG2 DB 0DH, 0AH, 'The letter in uppercase is : $'
CHAR DB ?
.CODE        ; this is the code segment
MOV AX,@DATA     ; get the address of the data segment
MOV DS, AX       ; and store it in register DS
MOV AH, 9  ; display string function
LEA DX, MSG1   ; get memory location of first message
INT 21H      ; display the string
MOV AH, 01 ; single character keyboard input function
INT 21H      ; call the function, result will be stored in AL
SUB AL, 20H      ; convert to the ASCII code of upper case
LEA SI, CHAR     ; load the address of the storage location
MOV [SI], AL     ; store the ASCII code of the converted
letter to memory
MOV AH, 9  ; display string function
LEA SI, MSG2   ; get memory location of second message
MOV DX, [SI]     ; and store it in the DX register
INT 21H      ; display the string
MOV AX, 4C00H; Exit to DOS function
INT 21H
```

String output function is used in this program to print a string on screen. The effective address of string must first be loaded in the DX register and then the following two lines are executed

```
MOV AH, 09
INT 21H
```

3. Write a Program to load two 8-bit numbers in AL and BL register, add two numbers and store the result into CL register.

```
.MODEL SMALL
.CODE
```

Program :

```
MOV AL, 08H
MOV BL, 09H
ADD AL, BL
MOV CL, AL
END Program
```

5.21 C LANGUAGE PROGRAMS

The different assembly language interrupts are used in 'C' language. int86, int86x, intdos, intdosx are the functions which required for this.

"Union" & "Structure"

- Derived Data Type, both are used to group a no. of variables together.
- **Structure :** Enable us to treat a no. of variables stored at different place in memory.
- **Union :** Enable us to treat the same space in memory as no. of different variables. In simple, a union offers a way for a section of memory to be treated as a variable of one type on one occasion and as a different variable of different type on another occasion.

int 86() - ROM BIOS Function

- To make S/W interrupt occurrence and invoke ROM-Bios function.
- The function needs 3 arguments
  - Interrupt no.
  - Two union var.
- **First :** Passing values to ROM-BIOS routine
- **Second :** Returning values from ROM-BIOS routine.

  e.g. int 86(16, &inregs, &outregs);

  Where,

  - 16 is interrupt number.

  Inregs : Values being sent to ROM- BIOS function.

  Outregs : Values being returned by ROM- BIOS function.

  Note : int86() requires only addresses of union, not union values.

Using the Declarations in dos.h

- Actually, structures WORDREGS & BYTEREGS union REGS are already declared in dos.h file. Inclusion of this file solve the purpose of program.

intdos() - Dos Interrupt Functions

- To invoke DOS interrupt

int 86x() &intdosx() :

- Used to invoke ROM_BIOS & DOS services respectively, if service requires use of DS & ES register else int 86() &intdos() are used.

5.22 ASSEMBLERS AND COMPILERS

Compiler :

- It's a computer program(s) that transforms source code written in a programming language into machine language that is the target language which usually has a binary form known as object code.
- A compiler is more intelligent than an assembler. It checks all kinds of limits, ranges, errors etc. But its program run time is more and occupies a larger part of the memory.
- It has slow speed. Because a compiler goes through the entire program and then translates the entire program into machine codes. If a compiler runs on a computer and produces the machine codes for the same computer then it is known as a self compiler or resident compiler.
- On the other hand, if a compiler runs on a computer and produces the machine codes for other computer then it is known as a cross compiler.

Interpreter :

It translates high level instructions into an intermediate form, it translates the code into the intermediate form line by line an caries out specific actions.

Assembler : (refer section 5.6)

MULTIPLE CHOICE QUESTIONS (MCQ's)

1. Which interface allows the cardinal provision of communicating with one particular input-output device in addition to the programming capability for operating with specific device?
   (a) Parallel Peripheral Interface
   (b) Serial Communication Interface
   (c) Special Dedicated Interface
   (d) Direct Memory Access Interface
2. Which lines are supposed to control or handle the transfer operation between two devices in asynchronous mode by apprising the status of transfer using common bus ?
   (a) Control Lines (b) Data Lines
   (c) Transfer Lines (d) Handshake Lines
3. What registers are significantly incremented and decremented respectively for the transmission of each byte by Direct Memory Access (DMA) ?
   (a) Address REGISTER & Byte Count Register
   (b) Control Register & Byte Count Register
   (c) Transmitter Register & Byte Count Register
   (d) Status- REGISTER & Byte Count Register
4. The operation, $\overline{IOW}$ performs
   (a) write operation on input data
   (b) write operation on output data
   (c) read operation on input data
   (d) read operation on output data
5. If a typical static RAM cell require 6 transistors then corresponding dynamic RAM requires
   (a) transistor along with capacitance
   (b) transistors along with resistance
   (c) transistors along with diode
   (d) transistors along with capacitance
6. If the size of a memory chip is 512 x 1 bits how many chips are required to make up 1 K bytes of memory?
   (a) 2 (b) 8
   (c) 16 (d) 1024
7. If the memory chip size is 1024 x 4-bits how many chips are required to make 4K bytes of memory?
   (a) 4 (b) 8
   (c) 16 (d) 4096
8. The synchronization between microprocessor and memory is done by
   (a) ALE signal (b) HOLD signal
   (c) READY signal (d) None of these
9. For the most Static RAM the write pulse width should be at least
   (a) 10ns (b) 60ns
   (c) 300ns (d) 1μs
10. SDRAM refers to
    (a) Synchronous DRAM
    (b) Static DRAM (c) Semi DRAM
    (d) Second DRAM
11. Access time is faster for
    (a) ROM (b) SRAM
    (c) DRAM (d) EPROM
12. The no. of address lines required to address a memory of size 32 K is
    (a) 15 lines (b) 16 lines
    (c) 18 lines (d) 14 lines
13. The number of counters that are present in the programmable timer device 8253 is
    (a) 1 (b) 2
    (c) 3 (d) 4

14. The operation that can be performed on control word register in 8253 is
    (a) read operation (b) write operation
    (c) read and write operations
    (d) None
15. The mode of 8253 that is used to interrupt the processor by setting a suitable terminal count is
    (a) mode 0 (b) mode 1
    (c) mode 2 (d) mode 3
16. The generation of square wave is possible using 8253 in the mode
    (a) mode 1 (b) mode 2
    (c) mode 3 (d) mode 4
17. In control word register of 8253, if SC1=0 and SC0=1, then the counter selected is
    (a) counter 0 (b) counter 1
    (c) counter 2 (d) none
18. The control word register contents of 8253 are used for
    (a) initialising the operating modes
    (b) selection of counters
    (c) choosing binary/BCD counters
    (d) all of the above
19. 8253 chip is _____ chip.
    (a) timer counter
    (b) the interrupt controller.
    (c) the DMA controller
    (d) general-purpose parallel interface
20. 8253 counter pin OUT is to _____.
    (a) indicates the counting process ends
    (b) start counting process
    (c) control counting process
    (d) input clock signal
21. The 8253 contains _____ counters
    (a) 2-16 bit (b) 3-16 bit
    (c) 2-8 bit (d) 3-8 bit
22. In 8253 there are _____ pins
    (a) 20 (b) 24
    (c) 30 (d) 40
23. 8086 microprocessor is interfaced to 8253 a programmable interval timer. The maximum number by which the clock frequency on one of the timers is divided by
    (a) 2^{16} (b) 2^{8}
    (c) 2^{10} (d) 2^{20}
24. The status that cannot be operated by direct instructions is
    (a) Cy (b) Z
    (c) P (d) AC
25. Two positive numbers, A and B added will result in an overflow when
    (a) The result is a negative number
    (b) The result is zero
    (c) There is no carry into the sign bit
    (d) The result is a positive number
26. A programmer creates an identifier with DWORD data type. Which of the following register can be used to MOV the data?
    (a) CX (b) BH
    (c) DX (d) EAX
27. Choose the INCORRECT instruction in assembly language.
    (a) MOV VAL1,AX (b) MOV AL,BX
    (c) MOV AH,35H (d) MOV EAX,EBX
28. In 8086 microprocessor one of the following instructions is executed before an arithmetic operation
    (a) AAM (b) AAD
    (c) DAS (d) DAA
29. In a microprocessor, the service routine for a certain interrupt starts from a fixed location of memory which cannot be externally set, but the interrupt can be delayed or rejected such an interrupt is
    (a) non-maskable and non-vectored
    (b) maskable and non-vectored
    (c) non-maskable and vectored
    (d) maskable and vectored
30. In a 16-bit microprocessor, words are stored in two consecutive memory locations. The entire word can be read in one operation provided the first
    (a) word is even
    (b) word is odd
    (c) memory location is odd
    (d) memory address is even
31. The ESC instruction of 8086 may have two formats. In one of the formats, no memory operand is used Under this format, the number of external op-codes (for the co-processor) which can be specified is?
    (a) 64 (b) 128
    (c) 256 (d) 512

32. DB, DW and DD directives are used to place data in particular location or to simply allocate space without pre assigning anything to space. The DW and DD directories are used to generate
    (a) offsets
    (b) full address of variables
    (c) full address of labels
    (d) offsets of full address of labels and variables
33. When the RET instruction at the end of subroutine is executed,
    (a) the information where the stack is initialized is transferred to the stack pointer
    (b) the memory address of the RET instruction is transferred to the program counter
    (c) two data bytes stored in the top two locations of the stack are transferred to the program counter
    (d) two data bytes stored in the top two locations of the stack are transferred to the stack pointer
34. The load instruction is mostly used to designate a transfer from memory to a processor register known as___.
    (a) Accumulator
    (b) Instruction Register
    (c) Program counter
    (d) Memory address Register
35. A group of bits that tell the computer to perform a specific operation is known as___.
    (a) Instruction code (b) Micro-operation
    (c) Accumulator (d) Register
36. _____ register keeps tracks of the instructions stored in program stored in memory.
    (a) AR (Address Register)
    (b) XR (Index Register)
    (c) PC (Program Counter)
    (d) AC (Accumulator)
37. PSW is saved in stack when there is a ____.
    (a) interrupt recognized
    (b) execution of RST instruction
    (c) Execution of CALL instruction
    (d) All of these
38. An n-bit microprocessor has____.
    (a) n-bit program counter
    (b) n-bit address register
    (c) n-bit ALU
    (d) n-bit instruction register
39. When CPU is executing a Program that is part of the Operating System, it is said to be in ____.
    (a) Interrupt mode (b) System mode
    (c) Half mode (d) Simplex mode
40. A Stack-organised Computer uses instruction of ____.
    (a) Indirect addressing
    (b) Two-addressing
    (c) Zero addressing
    (d) Index addressing
41. In a program using subroutine call instruction, it is necessary_____.
    (a) initialize program counter
    (b) Clear the accumulator
    (c) Reset the microprocessor
    (d) Clear the instruction register
42. In a vectored interrupt.
    (a) the branch address is assigned to a fixed location in memory.
    (b) the interrupting source supplies the branch information to the processor through an interrupt vector.
    (c) the branch address is obtained from a register in the processor
    (d) none of the above
43. In 8086 the overflow flag is set when
    (a) The sum is more than 16 bits
    (b) Signed numbers go out of their range after an arithmetic operation
    (c) Carry and sign flags are set
    (d) During subtraction
44. Computers use addressing mode techniques for _____.
    (a) giving programming versatility to the user by providing facilities as pointers to memory counters for loop control
    (b) to reduce no. of bits in the field of instruction
    (c) specifying rules for modifying or interpreting address field of the instruction
    (d) All the above
45. A low memory can be connected to microprocessor by using
    (a) INTER (b) RESET IN
    (c) both (a) and (b) (d) READY
46. To put the microprocessor in the wait state
    (a) lower the HOLD input
    (b) lower the READY input
    (c) raise the HOLD input
    (d) None of these

47. A basic instruction that can be interpreted by a computer generally has
    (a) An operand and an address
    (b) A decoder and an accumulator
    (c) Sequence register and decoder
    (d) None of these

48. In a microprocessor system with memory mapped I/O
    (a) Devices have 8-bit addresses
    (b) Devices are accessed using IN and OUT instructions
    (c) There can be a maximum of 256 input devices and 256 output device
    (d) Arithmetic and logic operations can be directly performed with the I/O data

49. The ability to temporarily halt the CPU and use this time to send information on buses is called
    (a) Direct memory access
    (b) Vectoring the interrupt
    (c) System Interrupt
    (d) Cycle stealing

50. An instruction used to set the carry flag in a computer can be classified as
    (a) Data transfer (b) Process control
    (c) Logical (d) None of these

51. ..... instruction converts the addition result into valid BCD number
    (a) AAD (b) AAA
    (c) ADD (d) AND

52. Which type of JMP instruction assembles if the distance is 0020 h bytes
    (a) near. (c) far.
    (b) short. (d) none of the above.

53. Number of the times the instruction sequence below will loop before coming out of loop is

    MOV AL, 00h
    A1: INC AL
    JNZ A1

    (a) 00 (b) 01
    (c) 255 (d) 256

54. What will be the contents of register AL after the following has been executed

    MOV BL, 8C
    MOV AL, 7E
    ADD AL, BL

    (a) 0A and carry flag is set
    (b) 0A and carry flag is reset
    (c) 6A and carry flag is set
    (d) 6A and carry flag is reset

55. After adding following instruction which flag of 8086 is set.

    Mov Bl, 29
    Mov AL, 98
    Add Al,BL

    (a) DF (b) AC
    (c) TF (d) IF

56. LOCK prefix is used most often
    (a) during normal execution.
    (b) during DMA accesses
    (c) during interrupt servicing.
    (d) during memory accesses.

57. In 8086 microprocessor the following has the highest priority among all type interrupts.
    (a) NMI (b) DIV 0
    (c) TYPE 255 (d) OVER FLOW

58. What is the RST for the TRAP?
    (a) RST5.5 (b) RST4.5
    (c) RST4 (d) RST 6.5

59. In 8086, Example for Non maskable interrupts are
    (a) Trap (b) RST6.5
    (c) INTR

60. Address line for RST3 is?
    (a) 0020H (b) 0028H
    (c) 0018H (d) 0024H

61. BHE of 8086 microprocessor signal is used to interface the
    (a) Even bank memory
    (b) Odd bank memory
    (c) I/O
    (d) DMA

62. NMI input is
    (a) Edge triggered
    (b) Level sensitive
    (c) Both edge and level triggered
    (d) edge triggered and level sensitive

63. What do the symbols [ ] indicate?
    (a) Direct addressing
    (b) Register Addressing
    (c) Indirect addressing
    (d) None of the above

64. Direction flag is used with
(a) String instructions.
(b) Stack instructions.
(c) Arithmetic instructions.
(d) branch instructions.

65. In 8086 when there is carry from lower nibble to upper nibble which flag is set?
(a) CF (b) AC
(c) TF (d) IF

66. Which of the following statement is true?
(a) The group of machine cycle is called a state.
(b) A machine cycle consists of one or more instruction cycle.
(c) An instruction cycle is made up of machine cycles and a machine cycle is made up of number of states.
(d) None of the above

67. A bus cycle is equal to how many clocking periods
(a) Two (b) Three
(c) Four (d) Six

68. In 8086 PUSH and POP instructions uses which segment?
(a) Data Segment (b) code segment
(c) Stack Segment (d) Extra Segment

69. Following is the control instruction for 8086 microprocessor.
(a) ADD (b) CLC
(c) Call (d) AAD

70. Which instruction does not affect any flag?
(a) JA (b) ADD
(c) XOR (d) RCL

71. LOOP instruction decrements the content of which register?
(a) AX (b) BX
(c) CX (d) IP

72. In 8086 which segment : offset register pair is plays a role of a Program Counter
(a) DS:SI (b) CS:IP
(c) ES:DI SS:SP (d) none of these

73. In 8086 following instruction inserts delay in the program and does not affect any flag
(a) NOP (b) CALL
(c) JNZ (d) RET

74. CLD instruction is related to which flag
(a) Direction (b) Carry
(c) Auxiliary Carry (d) Interrupt

75. Following instruction does not check any conditional before change
(a) JNZ (b) JA
(c) JNE (d) JMP

76. Which of the following is an illegal instruction
(a) MOV AX, 30000 (b) INC AL, 1
(c) AND BX, BX (d) ADD AX, 30

77. Given that the subprogram putc displays the character in al, the effect of the following instructions: mov al, 'a'
add al, 2
call putc
is to
(a) display 2 (b) display c
(c) display a (d) display a blank

78. Which of the following is an illegal 8086 instruction
(a) MOV 20, BX (b) INC AL
(c) AND BX, BX (d) ADD AX, 30

79. Which of the following is an illegal 8086 instruction
(a) ret 2 (b) push al
(c) ADD bx, 25000 (d) AND AX, DX

80. Which is not a function of a loader
(a) allocation (b) translation
(c) relocation (d) loading

81. Assembler is a program that
(a) places programs into memory and prepares then for execution
(b) automates the translation of assemble language into machine language
(c) accepts a program written in a high level language and produces an object program
(d) appears to execute a resource as if it were machine language

82. Assembly language is
(a) Low-level programming language
(b) High-level programming language
(c) Also known as machine language
(d) Not considered closer to the computer

83. A program which translate high level language program into a machine language program is called
(a) Compiler (b) interpreter
(c) both (a) and (b) (d) None of these

84. C is a...
(a) completely high level language
(b) completely low level language
(c) high level language with low level features
(d) none of above

85. Which of the following related to machine language
    (a) difficult to learn
    (b) first generation language
    (c) machine dependent
    (d) all of above

86. A system program that set up an executable program in main memory ready for execution is
    (a) assembler (b) linker
    (c) loader (d) text-editor

87. The linker
    (a) is the same as loader
    (b) is required to create a load module
    (c) user source code as input
    (d) is always used before programs are executed

ANSWERS

1. (c)	2. (d)	3. (a)	4. (b)	5. (a)
6. (c)	7. (b)	8. (c)	9. (b)	10. (a)
11. (b)	12. (a)	13. (c)	14. (d)	15. (a)
16. (c)	17. (b)	18. (d)	19. (a)	20. (a)
21. (b)	22. (b)	23. (a)	24. (d)	25. (a)
26. (d)	27. (b)	28. (b)	29. (d)	30. (d)
31. (b)	32. (d)	33. (c)	34. (a)	35. (a)
36. (c)	37. (a)	38. (d)	39. (b)	40. (c)
41. (d)	42. (b)	43. (b)	44. (d)	45. (d)
46. (b)	47. (a)	48. (d)	49. (d)	50. (b)
51. (b)	52. (a)	53. (d)	54. (a)	55. (b)
56. (c)	57. (a)	58. (b)	59. (a)	60. (c)
61. (b)	62. (a)	63. (c)	64. (a)	65. (b)
66. (b)	67. (c)	68. (c)	69. (b)	70. (a)
71. (c)	72. (b)	73. (a)	74. (a)	75. (d)
76. (b)	77. (b)	78. (a)	79. (a)	80. (b)
81. (c)	82. (a)	83. (c)	84. (c)	85. (d)
86. (c)	87. (a)			

EXERCISE

1. Explain even and odd memory banks of 8086.
2. Explain memory addressing in 8086 microcontroller.
3. Explain I/O addressing in 8086.
4. Explain memory mapped I/O and I/O mapped I/O.
5. Explain DMA in brief.
6. Explain 8237 pin description.
7. Draw block diagram of 8257.
8. Explain in brief 8086 interrupts.
9. Explain 8086 addressing modes.
10. Explain instruction timings.
11. Explain classification of instruction set.
12. What is assembly language? Explain in short.
13. What are assembly language program development tools? Explain with steps.
14. Give an example of assembly language.

MODEL QUESTION PAPERS

Mid Sem. Examination

Time: 1 Hour **Max. Marks: 20**

Instructions to the candidates:

(1) Figures to the right side indicate full marks.

(2) Neat diagrams must be drawn wherever necessary.

(3) Assume suitable data, if necessary.

1. Attempt any two of the following

(a) Which gates are known as universal gates ? Justify using examples. **(5)**

(b) State and prove any two theorem of Boolean algebra. **(5)**

(c) What is the limitation of K – map? **(5)**

2. Attempt any two of the following

(a) Minimize the following equation using K – map and realize it using NAND gates only. **(5)**

$Y = \Sigma m (0, 12, 3, 5, 7, 8, 9, 11, 14)$

(b) What are different ways of representing signed binary numbers ? Explain with examples ? **(5)**

(c) Explain decimal to BCD encoder with logic symbol and truth table. **(5)**

End Sem. Examination

Time : 3 Hours **Max. Marks : 60**

Instruction to candidates :

(1) Each question carries 12 marks.

(2) Figures to the right indicate full marks.

(3) Neat diagrams should be drawn wherever necessary.

(4) Assume suitable data, if necessary.

1. Attempt any two of the following

(a) For a maximum 3-digit octal number obtain a equivalent decimal number. **(6)**

(b) Design a half-adder and full-adder circuits using K – maps. **(6)**

(c) What is race around condition? Explain with the help of timing diagram how it is removed in basic flip-flop circuit. **(6)**

2. Attempt any two of the following

(a) Convert $(377)_8$ into hexadecimal. **(6)**

(b) Implement the following expression using 8 : 1 multiplexer. **(6)**

$f(A, B, C, D) = \Sigma m (2, 4, 6, 7, 9, 10, 11, 12, 15)$

(c) Explain D flip flop with preset and clear input? **(6)**

3. Attempt any two of the following

(a) Draw and explain 4-bit binary up counting with this concept. Also draw the necessary timing diagram. Is there any frequency division concept in it? Comment on frequency generated at the output of each flip-flop. **(6)**

(b) Explain the features of microprocessor. **(6)**

(c) Explain memory mapped I/O and I/O mapped I/O. **(6)**

4. Attempt any two of the following

(a) With a neat diagram explain the architecture of 8086. **(6)**

(b) What are the 4 categories of segments explain with neat diagram? **(6)**

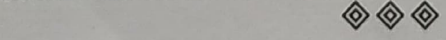
(c) Explain even and odd memory banks of 8086. **(6)**

5. Attempt any two of the following

(a) What are assembly language program development tools? Explain with steps. **(6)**

(b) Explain timing diagrams and execution cycles. **(6)**

(c) Explain instruction timings. **(6)**



About The Authors

Dr. NARENDRA S. JADHAV
B.Tech. (E&TC), M.Tech., Ph.D.,
Assistant Professor,
Electronics & Telecommunication Engg. Dept.,
Dr. Babasaheb Ambedkar Technological University (BATU),
LONERE, Dist. RAIGAD.

Dr. (Mrs.) ALPANA P. ADSUL
B.E. (Electronics), M.E. (CSE-IT), Ph.D.,
Associate Professor & Head,
Information Technology Dept.,
Sinhgad Institute of Technology & Science,
Narhe, PUNE.

About The Book

- Most Experienced, Eminent & Popular Authors.
- Content Explained in a Simple & Lucid Manner.
- Well Balanced Book to Achieve Maximum Results.
- Important Exam Oriented Hints inserted in the Exercises.
- Large Number of Solved Examples.
- Several Illustrations to Support the Content.
- Top Seller in the Last Three Decade.
- Affordable Price.

Note : Do not Xerox Or Copy the Book, It is Punishable Offence.

About All Books

Text Books For S.Y. B.TECH Course in Computer Engg. Sem. - IV

- Design & Analysis of Algorithms - Mrs. V. S. Tidke, Dr. V. S. Pawar A. S. Shitole
- Operating System - Mrs. P. P. Ahire
- Basic Human Rights - Dr. Y. Malshette, Dr. S. Pore, Dr. A. Shesh
- Probability & Statistics - Dr. P. G. Dixit
- Digital Logic Design & Microprocessors - Dr. N. S. Jadhav, Dr. A. P. Adsul

Books Available At Shop

ISBN 935451724-2
9 789354 517242
N6226

NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off. J. M. Road, Pune 411005,
Maharashtra, India.
(+91-020) 25512336/ 7/ 9
www.niralibooks.com