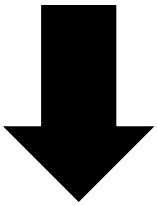
# DIGITAL ELECTRONICS & LOGIC DESIGN

N. G. PALAN



TECHNOVA
PUBLICATIO

# This is book is download from



http://MumbaiBscitStudy.Com

## Follow us @

**Facebook:** kamaltuniverse
**Twitter:** kamaltuniverse
**Instagram:** kamaltuniverse
**Google+:** kamalTUniverse

# WARNING

**This E-book is only for Education Purpose.**
**Don't Re-Use this E-Books, This makes violating Copyright Terms and services.**

# Contents


**Chapter 1 : Number Systems** ..... 1-1 to 1-86

1.1 Analog System ..... 1-3
1.1.1 Measuring Analog Signal ..... 1-3
1.1.2 Analysis of the Analog Signal ..... 1-3
1.1.3 Noise ..... 1-4
1.2 Digital System ..... 1-5
1.3 Numbering System ..... 1-7
1.3.1 Positional Number ..... 1-8
1.4 Binary Number System ..... 1-8
1.4.1 Decimal to Binary Conversion ..... 1-9
1.4.2 Binary to Decimal Conversion ..... 1-13
1.4.3 Nibble, Byte and Word in Binary ..... 1-15
1.5 Octal Number System ..... 1-15
1.5.1 Decimal to Octal Conversion ..... 1-16
1.5.2 Octal to Decimal Conversion ..... 1-18
1.6 Hexadecimal Number System ..... 1-19
1.6.1 Decimal to Hexadecimal (hex.) Conversion ..... 1-20
1.6.2 Hexadecimal to Decimal Conversion ..... 1-22


Scanned by CamScanner

1.12.3 ISCII Code ..... 1-48
1.12.4 Hollerith Code ..... 1-49
1.12.5 Morse Code (Telegraph Code) ..... 1-50
1.12.6 Teletypewriter (TTY) ..... 1-51
1.13 Error Detecting and Correcting Code ..... 1-52
1.13.1 Parity Bit ..... 1-52
1.13.2 Double Parity ..... 1-52
1.13.3 Checksum ..... 1-53
1.13.4 Five Bit Codes ..... 1-54
1.13.5 Code with More than 5 Bits or Equal to 5 Bits ..... 1-54
1.13.6 Error Correcting Codes ..... 1-55
1.14 Universal Product Code ..... 1-55
1.15 Code Conversion ..... 1-57
1.15.1 Decimal to BCD (Binary Coded Decimal) ..... 1-58
1.15.2 Binary to BCD ..... 1-58
1.15.3 BCD to Binary ..... 1-59
1.15.4 BCD to XS - 3 (Excess - 3) ..... 1-59
1.15.5 XS-3 (Excess - 3) to BCD ..... 1-60
1.15.6 Binary to Gray ..... 1-60
1.15.7 Gray to Binary ..... 1-60
1.16 Examples from University Papers : ..... 1-63

**Chapter 2 : Binary Arithmetic** ..... 2-1 to 2-82

2.1 Introduction ..... 2-2
2.2 Binary Addition ..... 2-2
2.3 Binary Subtraction ..... 2-5
2.4 Negative Number Representation in Binary System ..... 2-8
2.4.1 Signed Magnitude Code (SM Representation) ..... 2-8
2.4.2 One's Complement Notation (1's complement notation) ..... 2-10
2.4.3 2's Complement Notation (2C Notation) ..... 2-12
2.5 Binary Subtraction using 1's and 2's Complement Method ..... 2-15
2.5.1 Subtraction using 1's Complement Method ..... 2-15
2.5.2 Binary Subtraction using 2's Complement Method ..... 2-17
2.6 Binary Multiplication ..... 2-19
2.7 Binary Division ..... 2-22
2.8 Arithmetic in Octal Number System ..... 2-25
2.8.1 Addition ..... 2-2
2.8.2 Subtraction ..... 2-2
2.8.3 1's Complement to Represent Negative Octal Number ..... 2-2


Scanned by CamScanner

169128
169129
169




2.8.4 2's Complement to Represent Negative Digital Number ... 2-31
2.8.5 Multiplication ... 2-33
2.8.6 Division ... 2-35
2.9 Arithmetic in Hexadecimal Number System ... 2-39
2.9.1 Addition ... 2-41
2.9.2 Subtraction ... 2-44
2.9.3 One's Complement Method for Subtraction in Hex Numbers ... 2-46
2.9.4 Two's Complement Method for Subtraction in Hex Number ... 2-48
2.9.5 Multiplication ... 2-62
2.9.6 Division ... 2-62
2.10 BCD Arithmetic ... 2-65
2.10.1 Addition ... 2-68
2.10.2 BCD Subtraction ... 2-69
2.11 XS-3 Arithmetic ... 2-72
2.12 XS-3 Subtraction ... 2-73
2.13 Examples from University Papers ... 3-1 to 3-70

**Chapter 3 : Boolean Algebra and Logic Gates**

3.1 Introduction ... 3-3
3.2 Logic ... 3-3
3.2.1 AND Logic ... 3-4
3.2.2 OR Logic ... 3-8
3.2.3 NOT Operation (an inverter) (Complementation) ... 3-12
3.3 Boolean Theorems ... 3-14
3.4 Boolean Laws ... 3-16
3.5 De Morgan's Theorem ... 3-18
3.6 Perfect Induction ... 3-23
3.7 Reduction of Logic Expression using Boolean Algebra ... 3-24
3.8 Deriving Boolean Expression from Given Circuit ... 3-36
3.9 Exclusive or Gate-Exclusive Nor Gate ... 3-43
3.10 Universal Logic Gates ... 3-48
3.11 Implementation of other Gates using Universal Logic Gates ... 3-50
3.11.1 Use NAND to Implement all Gates ... 3-50
3.11.2 Use NOR to Implement all the Gates ... 3-51
3.12 Implementation Using NAND Gate ... 3-52
3.13 Implementation Using NOR Gate ... 3-54
3.14 Input Bubbled Logic ... 3-56
3.15 Assertion Level, Positive and Negative Logic ... 3-58
3.16 Examples from University Papers


Scanned by CamScanner

**Chapter 4 : Minterm, Maxterm and Karnaugh Map** 4-1 to 4-103

4.1 Introduction ..... 4-2

4.2 Minterms and Sum of Minterm Form ..... 4-2

4.2.1 Algebraic Construction of Minterms using Boolean Expressions ..... 4-6

4.3 Maxterm and Product of Maxterm Form ..... 4-9

4.4 Algebraic Construction of Maxterm using Boolean Expressions ..... 4-11

4.5 Reduction Technique using Karnaugh Map ..... 4-14

4.5.1 Standard 2/3/4 Variable K-maps ..... 4-15

4.5.2 Transferring Truth Table to K-map ..... 4-18

4.6 Grouping of Variables in K-map ..... 4-19

4.6.1 Grouping of Variables in Two Variable K-Map ..... 4-20

4.6.2 Grouping of Variables in Three Variable K-Map ..... 4-21

4.6.3 Grouping of Variables in Four Variable K-Map ..... 4-27

4.6.4 Don't Care Condition ..... 4-38

4.7 K-Map for POS Form ..... 4-42

4.8 Minimise Boolean Expression using K-Map or Obtaining K-Map for given Boolean Expression ..... 4-54

4.9 Quine Mc Cluskey Method or Tabular Method ..... 4-61

4.10 Examples from University papers. ..... 4-74

**Chapter 5 : Combinational Logic Circuits** 5-1 to 5-57

5.1 Introduction ..... 5-2

5.2 Multi-input Multi-output Combinational Circuit ..... 5-21

5.3 Code Conversion ..... 5-34

5.4 Examples from University papers. ..... 5-52

**Chapter 6 : Arithmetic Circuit** 6-1 to 6-44

6.1 Introduction ..... 6-2

6.2 Adder ..... 6-2

6.2.1 Half Adder ..... 6-2

6.2.2 Full Adder ..... 6-3

6.2.3 Fast Adders or Carry Look ahead Generator ..... 6-5

6.3 BCD Adder ..... 6-10

6.3.1 Packed BCD Addition ..... 6-12

6.4 Excess - 3 Adder ..... 6-13

6.5 Binary Subtractor ..... 6-15

6.5.1 Half Subtractor ..... 6-15

6.5.2 Full Subtractor ..... 6-16

Scanned by CamScanner

6.5.3 Binary Parallel Subtractor using 1's Complement Method ..... 6-17
6.5.4 Binary Parallel Subtractor using 2's Complement Method ..... 6-19
6.5.5 Controlled Addition / Subtraction ..... 6-20
6.6 BCD Subtractor ..... 6-22
6.6.1 Working of Circuit ..... 6-23
6.7 XS - 3 Subtractor ..... 6-26
6.8 Multiplier ..... 6-31
6.8.1 Fast Multiplier ..... 6-32
6.9 Comparator ..... 6-34
6.9.1 Single Bit Magnitude Comparator ..... 6-34
6.9.2 IC 7485 - 4 bit Magnitude Comparator ..... 6-36
6.10 Examples from University papers ..... 6-42

**Chapter 7 : Multiplexer, Demultiplexer, ALU, Encoder, Decoder 7-1 to 7-47**

7.1 Introduction ..... 7-2
7.2 Multiplexer (Data Selectors) ..... 7-2
7.3 Demultiplexer (Data Distributor) ..... 7-23
7.4 Decoder ..... 7-27
7.5 ALU (Arithmetic Logic Unit) ..... 7-31
7.6 Encoders ..... 7-36
7.6.1 Priority Encoders ..... 7-37
7.7 Examples from University papers ..... 7-39

**Chapter 8 : Flip-Flop 8-1 to 8-41**

8.1 Introduction ..... 8-2
8.2 Terminologies used in Sequential System ..... 8-2
8.2.1 Clock (CLK) - A Timing Signal ..... 8-2
8.2.2 State ..... 8-2
8.2.3 State Table ..... 8-3
8.2.4 State Diagrams ..... 8-4
8.2.5 State Equation ..... 8-5
8.3 NOR Gate Latch : [S-R Flip-Flop ] ..... 8-6
8.3.1 Nand Latch ..... 8-7
8.3.2 Level Clocking ..... 8-9
8.3.3 Clocked Nand Latch ..... 8-10
8.3.4 Edge Triggered Nand Latch ..... 8-10
8.3.5 Preset and Clear ..... 8-11
4 D FF (Delay Flip-Flop) ..... 8-13
5 Flip-Flop Timing Consideration ..... 8-14
6 JK Flip-Flop ..... 8-14
8.6.1 Race Condition in JK FF (Race Around Condition) .....


Scanned by CamScanner

8.7 Master Slave JK FF(M/SJ K FF) ..... 8-22
8.8 TFF (Toggle FF) ..... 8-24
8.9 Conversion from one Type of FF to Another ..... 8-25
8.10 Applications of Flip-Flops ..... 8-32
8.10.1 Elimination of Keyboard Debounce ..... 8-32
8.11 Examples from University papers. ..... 8-34

**Chapter 9 : Counters** 9-1 to 9-87

9.1 Introduction ..... 9-2
9.2 Asynchronous Counter ..... 9-2
9.2.1 3 Bit Ripple Counter ..... 9-3
9.3 Terms Related to Counter ..... 9-6
9.3.1 State Diagram ..... 9-6
9.3.2 Module N Counter ..... 9-6
9.3.3 Problems Involved In Ripple Counter ..... 9-12
9.3.4 Power On Reset Circuit ..... 9-14
9.4 IC 7493 (4 bit binary counter) ..... 9-15
9.5 Synchronous Counter ..... 9-18
9.5.1 Type T Design ..... 9-19
9.6 Bushing ..... 9-25
9.7 Type D Design ..... 9-28
9.8 Type JK Design ..... 9-33
9.9 Controlled Counter ..... 9-43
9.10 Presettable Counter ..... 9-45
9.11 IC 7490 ..... 9-46
9.12 IC 7492 ..... 9-51
9.13 Synchronous Counter ICs ..... 9-54
9.14 Analysis of Counter Circuit ..... 9-63
9.15 Examples from University Papers ..... 9-74

**Chapter 10 : Shift Register** 10-1 to 10-33

10.1 Register [Parallel and Shift Register] ..... 10-2
10.1.1 Buffer Register ..... 10-2
10.2 Serial Shifting ..... 10-3
10.3 Serial in - Serial out (SISO) ..... 10-5
10.4 Serial in-Parallel out ..... 10-6
10.5 Parallel in - Parallel out ..... 10-7
Ring Counter ..... 10-7
Johnson Counter (Twisted Ring Counter) ..... 10-10
10.6 Applications of Shift Register ..... 10-11
10.9 Pseudo Random Binary Sequence (PRBS) Generator ..... 1

Scanned by CamScanner

10.10 IC 7495 ..... 10-16
10.11 Analysis of Shift Counter ..... 10-21
10.12 Examples from University Papers ..... 10-25

**Chapter 11 : Analysis and Synthesis of Sequential Circuits 11-1 to 11-65**

11.1 Introduction ..... 11-2
11.2 Sequential circuit structure ..... 11-2
11.3 State diagram : Moore model ..... 11-6
11.4 General Procedure for Analysing Synchronous Flip-flop Circuits ..... 11-6
11.5 State Reduction and Assignment ..... 11-20
11.6 Sequential Circuit Synthesis ..... 11-23
11.6.1 Design Process ..... 11-23
11.7 Design of Counters ..... 11-48
11.7(A) Design with State Equation (States) ..... 11-48
11.8 Multi-input Synchronous Sequential Circuit Design ..... 11-50
11.9 Examples from University Papers ..... 11-57

**Chapter 12 : PAL and PLA 12-1 to 12-15**

12.1 Introduction ..... 12-2
12.2 PLA (Programmable Logic Array) ..... 12-2
12.3 PAL (Programmable Array Logic ) ..... 12-8
12.3.1 PAL Programming ..... 12-12

**Chapter 13 : Logic Families 13-1 to 13-59**

13.1 Introduction ..... 13-2
13.2 Digital IC Technology ..... 13-2
13.3 Terms Related to Digital IC ..... 13-3
13.3.1 General Definition ..... 13-3
13.3.2 Voltage Parameter ..... 13-3
13.3.3 Current Parameters ..... 13-4
13.3.4 Noise ..... 13-4
13.3.5 Fan Out ..... 13-6
13.3.6 Fan In ..... 13-7
13.3.7 Speed of Operation or Propagation Delay ..... 13-7
13.3.8 Current Source and Current Sink ..... 13-7
13.3.9 Power Dissipation ..... 13-8
13.3.10 Figure of Merit or Speed Power Product or Power Delay Product ..... 13-8
13.3.11 Operating Temperature ..... 13-8
13.3.12 Package Density ..... 13-8
13.3.13 Physical Packaging of Logic Circuits ..... 13-9

Scanned by CamScanner

Table of Contents




13.4 Bipolar Families ..... 13-10
13.4.1 Direct Coupled Transistor Logic (DCTL) ..... 13-10
13.4.2 Resistor Transistor Logic (RTL) ..... 13-10
13.4.3 Resistor Capacitor Transistor Logic (RCTL) ..... 13-11
13.4.4 Diode Transistor Logic (DTL) ..... 13-12
13.4.5 High Threshold Logic Gate (HTL) ..... 13-12
13.5 Transistor - Transistor Logic (TTL) ($T^2$ L Logic Family) ..... 13-15
13.5.1 Totem Pole Output Stage or Active Pull-up ..... 13-16
13.5.2 Transfer Characteristic of TTL Gate ..... 13-18
13.5.3 Current Sink and Source in TTL ..... 13-21
13.5.4 Clamping Diode ..... 13-22
13.5.5 Maximum Characteristic for Standard 54/74 TTL at $T_A = 25^\circ C$ ..... 13-23
13.6 TTL Families ..... 13-25
13.6.1 TTL Logic (DM 54/DM 74) ..... 13-25
13.6.2 Low Power TTL(54 L) ..... 13-25
13.6.3 Highspeed TTL ..... 13-26
13.7 Schottky TTL ..... 13-26
13.7.1 Schottky-Clamped BJT ..... 13-26
13.7.2 Low Power STTL(LSTTL) ..... 13-26
13.7.3 Advanced LSTTL (ALSTTL) ..... 13-27
13.7.4 Fairchild Advanced Schottky TTL (Fast) ..... 13-27
13.7.5 Advanced Schottky TTL (ASTTL) ..... 13-28
13.8 Comparison of TTL Families ..... 13-28
13.9 Important Points while Handling TTL ..... 13-28
13.10 Chip Number and Meaning ..... 13-30
13.11 Open Collector ..... 13-32
13.12 Wire ANDing ..... 13-32
13.13 Tri-state TTL Device ..... 13-34
13.13.1 Tri-State Buffer ..... 13-37
13.14 Emitter Coupled Logic (ECL) ..... 13-38
13.15 MOS Logic ..... 13-40
13.16 N-MOS Inverter ..... 13-44
13.17 N-MOS NAND Gate ..... 13-44
13.18 N-MOS NOR Gate ..... 13-45
13.19 Complementary MOS Logic (CMOS) ..... 13-45
13.19.1 CMOS Inverter ..... 13-46
13.19.2 CMOS NAND Gate ..... 13-4
13.19.3 CMOS NOR Gate ..... 13-4


Scanned by CamScanner

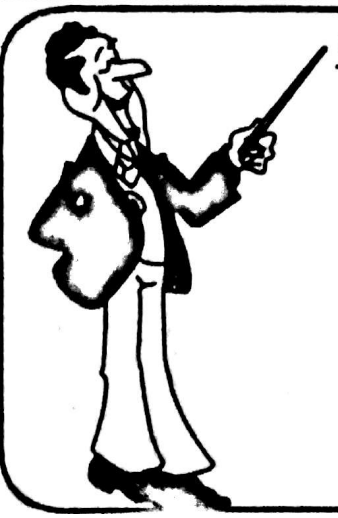
13.20 CMOS Characteristics ..... 13-48
13.20.1 Transfer Characteristics ..... 13-48
13.20.2 Noise Immunity ..... 13-49
13.20.3 Unused Inputs OR Floating Inputs ..... 13-49
13.20.4 Power Supply Filtering ..... 13-49
13.20.5 Electrostatic Discharge Prevention ..... 13-49
13.20.6 Protection Circuits ..... 13-49
13.20.7 Sourcing and Sinking ..... 13-50
13.21 Handling Guide for CMOS Devices ..... 13-50
13.22 CMOS Sub-Families ..... 13-51
13.22.1 4000/14000 CMOS ..... 13-51
13.22.2 74C Series ..... 13-51
13.22.3 74HC / HCT (High-Speed CMOS) ..... 13-51
13.22.4 74AC/ACT (Advanced CMOS) ..... 13-51
13.22.5 74AHC (Advanced High Speed CMOS) ..... 13-51
13.23 Logic Voltage Levels of CMOS and TTL ..... 13-52
13.24 Comparing TTL AND CMOS Series ..... 13-52
13.25 Interface ..... 13-52
13.25.1 CMOS Interfaced To PMOS ..... 13-53
13.25.2 CMOS Interfaced to NMOS ..... 13-54
13.25.3 TTL to CMOS Interface ..... 13-55
13.25.4 CMOS to TTL Interface ..... 13-57
13.26 BICMOS Digital Circuit ..... 13-57
• **Solved Question Paper** ..... **Q-1 to Q-17**

Scanned by CamScanner

**Syllabus :**

Decimal, binary, octal and hexadecimal number system and conversion, Binary weighted codes,

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Analog system | 1.1 | ✓ | – |
| • Digital system | 1.2 | ✓ | – |
| • Numbering system | 1.3,1.4,1.5 | ✓ | ✓ |
| • Hexadecimal number system | 1.6 | ✓ | ✓ |
| • Conversions | 1.7 | – | ✓ |
| • Floating point number systems | 1.8 | ✓ | ✓ |
| • Codes | 1.9,1.10,1.11, 1.12,1.13,1.14 | ✓ | ✓ |
| • Code conversion | 1.15 | ✓ | ✓ |



**Comments**

- Base chapter for the Digital subject.
- Concentrate more on numbering system & their conversions.
- Also clear the concepts about codes & code conversions.

Scanned by CamScanner

## 1.1 Analog System :

Before, I start explaining you digital electronics, let's first refresh some concepts about ***analog***. From that we will switch over to digital.

**Signal :**

A ***signal*** is defined as any physical quantity that varies with time, space, or any other independent variable or variables. Mathematically signal can be function of one or more independent variables, for example,

$$S_1(t) = 10\,t$$
$$S_2(t) = 10\,t^2$$

Most of the signals encountered in science and engineering are analog in nature i.e., the signals are functions of a continuous variable, such as time or space, and usually take on values in continuous range. The most common example of analog signal is ***sinusoidal waveform*** as shown in Fig. 1.1.

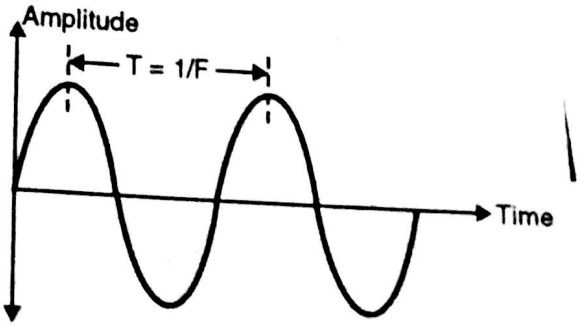



**Fig. 1.1 : Sine wave**

The expression for the signal can be written as,

$$S(t) = A \sin(\omega t + \theta)$$

where, A = Amplitude
$\omega$ = Radian/sec.
$\theta$ = Phase
T = Time duration of one cycle.
F = Frequency $= \frac{1}{T}$

### 1.1.1 Measuring Analog Signal :

To measure analog signal, in olden days analog instruments were developed by electronic Engg.'s. The most common example is ***analog multimeter***, you must have seen in your lab. The instrument is shown in Fig. 1.2.

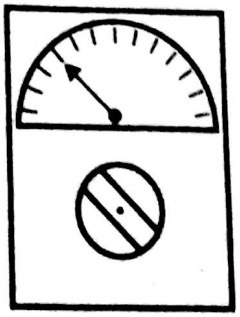


**Fig. 1.2**

**Problem Involved**

The problem involved with analog multimeter (Instrument) is the Accuracy. Reading taken by each person will be different, because reading changes as viewing angle changes. Accuracy is also limited by internally used mechanical structure.

## 1.2 Analysis of the Analog Signal :

In electronics you develop or design some circuits. To find how your circuit behaves, the system is excited with input signal. After exciting the system one should check output of the circuit. Depending on the nature of the output signal, and input signal applied to circuit, one can easily find behaviour of system (circuit). The block schematic of the same is given in Fig. 1.3.

Scanned by CamScanner

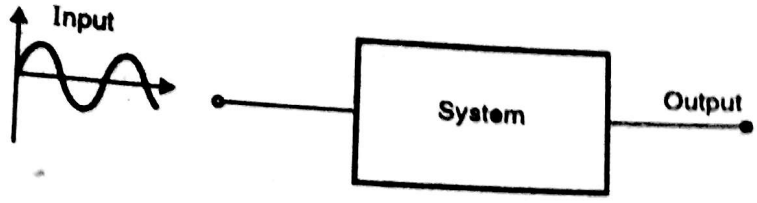



Fig. 1.3 : Block schematic

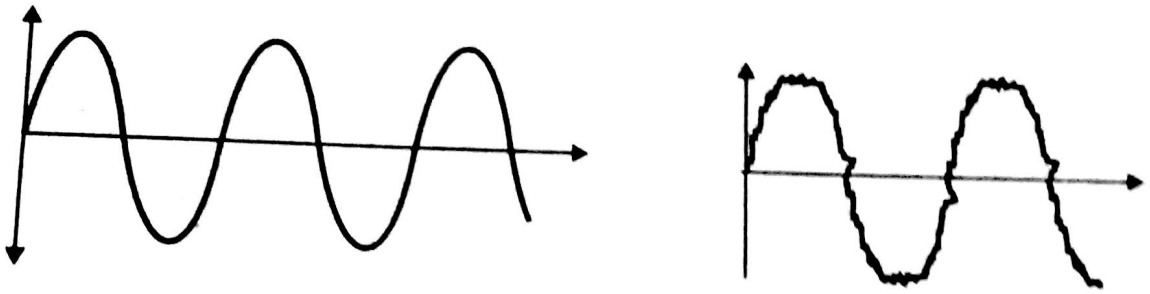
**Problem Involved**

Problem involved in this particular task is, as analog signal is continuously varying with respect to time, the person has to be expert in ***Time domain analysis***, to conclude perfect results.

### 1.1.3 Noise :

Noise is unwanted signal, which will get mixed up with wanted signal. Fig. 1.4 shows graphical representation of ***clear*** signal and signal with *noise*.

Noise gets Introduced in the circuit because of E M I (electromagnetic Interference) and RFI (Radio frequency Interference).



(a) Clear signal

(b) Noisy signal

Fig. 1.4

Fig. 1.5 shows EMI Interference caused because of Motor.

As shown, actual waveform may get changed because of noise pulses. To get ***good*** signal quality, one has to design a circuit called ***Filter***, to filter out noise. After filtering, you will get good signal. But to design filter circuit, built it and test it, requires more time and if by chance characteristic is not as per requirement, redesign the same. The circuit design becomes more complex as noise frequency and amplitude changes randomly.

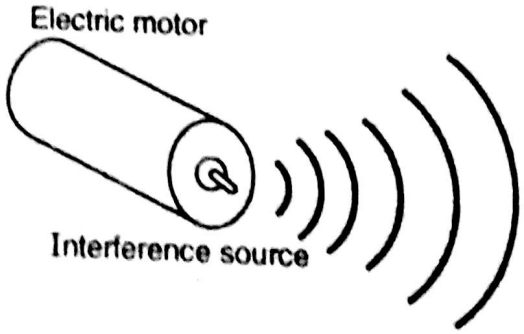



Fig. 1.5 : EMI interference

**Problem Involved**

To filter out noise from analog signal is bit complex and difficult job. Secondly to design analog circuit for noise filtering the person should have good mathematical background.

Scanned by CamScanner

**Conclusion :**

From the above discussion we conclude that to deal with analog system and signals.

*(1) The person should have mastery in Time domain analysis.*

*(2) The person should have good mathematical background.*

*(3) While measuring any parameter, accuracy suffers.*

*(4) Analog systems are bit slower.*

**Requirement :**

Here we require one such type of system or approach.

*(1) Which provides easy way of design.*

*(2) Easy to implement.*

*(3) Which should not involve too much of mathematical analysis.*

*(4) Which allows easy removal of noise signals.*

## 1.2 Digital System :

Digital system came into picture as a answer to problems involved in analog system. Digital system requires digital information. Digital Information is represented by fixed number of non continuous or discrete symbols called ***Digits***. You are already aware with decimal digits (10 digit system) i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, used to construct numbers. We prefer this number system possibly just because of 10 fingers. But binary system used in *digital* employ just two digit types, '0' and '1'.

### Binary System :

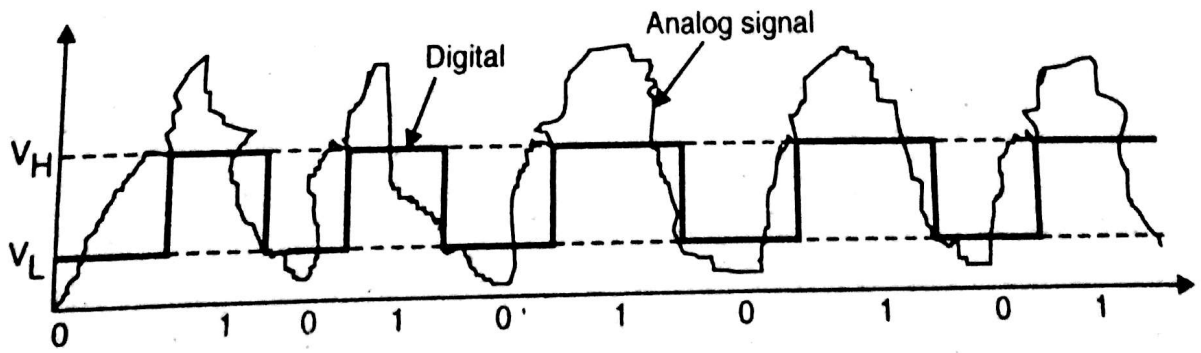
In digital we use binary, which restricts digital signal to have only two distinct values or states.

Here you may ask question why binary system ?

The reasons or advantages of binary system are as follows :

1. Most information processing system are constructed from switches, which are binary devices.
2. Binary signals are more reliable.
3. The basic–decision making processes required of digital systems are binary.

### Bits :




(a) (b) (c)

**Fig. 1.6**

Scanned by CamScanner

As binary quantities are encountered in many different physical forms, it is convenient to have a common or abstract way of representing binary states. As told, the approach is to use digit symbols '0' and '1' to represent two possible values of binary quantity at any point in time. These symbols are referred to as ***'bits'***, a contraction of term **Binary digits.** The Fig. 1.6 will show analog signal with digital.

We use '1' to denote 'HIGH' and '0' to denote 'LOW' level of the signal. Normally this convention is used. Binary voltage values $V_H$ and $V_L$ are represented as '1' and '0' respectively.

We have seen advantages of using binary system, let's elaborate that particular points.

1. **Switches :**

In digital system the basic building block is ON – OFF switches. Basically switches inherently have binary characteristic means they have two ***natural*** states, either ON or OFF i.e. CLOSE or OPEN respectively. Fig. 1.7 shows switch structure.

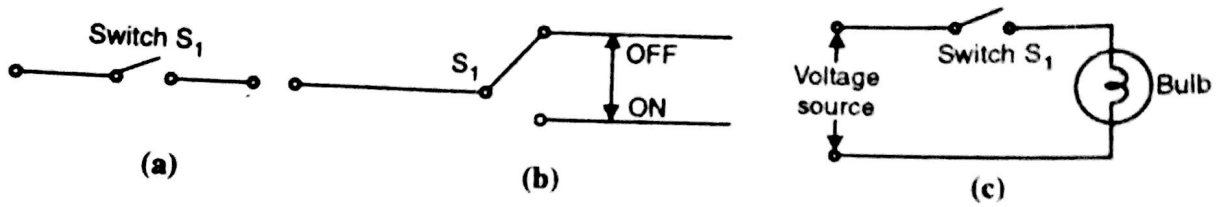



**Fig. 1.7**

Case I : When switch $S_1$ is closed, voltage source will lit the bulb.

Case II : When switch $S_1$ is open, circuit is cut off and bulb is OFF.

The output (bulb) is also following binary i.e. dark (OFF) and bright (ON).

One can structure these switches to get different selective control. Refer Fig. 1.8.

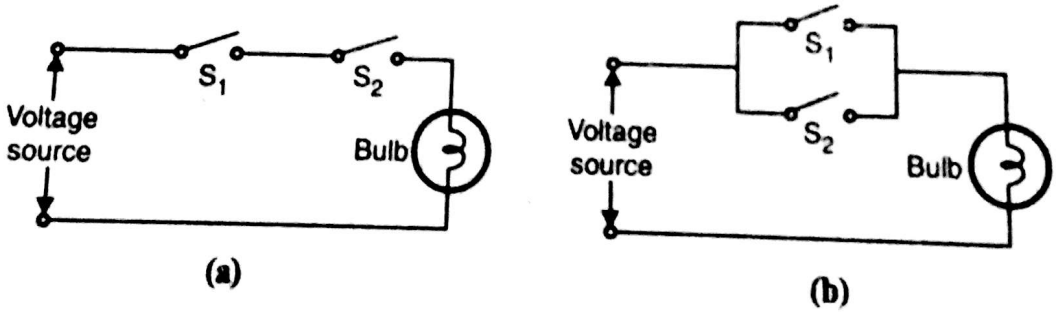



**Fig. 1.8**

In Fig. 1.8 (a) when both the switches are closed, bulb will lit. Either of the switch is open, bulb will not lit.

In Fig. 1.8 (b) when either of the switch is closed, bulb will lit. When both switches are open, bulb will be OFF.

Thus one can still go for more complicated switching network and get the final controlled output. This way, switch is the fundamental operating mode of electronic digital system.

2. **Reliability :**

I have mentioned that binary system have certain inherent reliability advantages over other system. In analog signal we saw that it is continuous and has one or more intermediate values between $V_H$ and $V_L$ of binary system. Because of more values, the distance seperating adjacent signal values is less.

Scanned by CamScanner

Therefore minor signal fluctuation i.e. nothing but noise, will cause signal value to spill over into an adjacent value and be misinterpreted. Such type of errors are least likely in binary system, if $V_H - V_L$ gap is large enough.

3. **Decision Making :**

Normally digital systems are used to control processes. To achieve, this it has to perform tests or make decision. Means one have to ask question.

Is X is true ?

Is Y, J is false ? etc.

The outcome of the question is either true or false i.e. binary system and therefore decision will be also binary.

i.e. if X is true, switch ON $S_1$.

if Y is false, switch OFF $S_2$ and so on.

Thus input variable signals will decide the final controlling output.

### Convention :

In binary system two states 1 and 0 are present. Here you would like to know, what will be the ***actual voltage*** for these levels ? These voltages are redefined or user defined ? Answer for the first question is, the voltage levels for High and Low levels will vary from family to family of the logic circuit. ***Family*** means TTL (Transistor Transistor Logic), ECL (emitter coupled logic), DTL (Diode Transistor Logic), CMOS (Complementary MOSFET) and so on. The detailed study of these families is in last chapter.

The answer for the next question is, the voltage levels are predefined by manufactures of the chip. User cannot change it.

For our Convenience we will take presently.

HIGH LOGIC = 1 = $V_{CC}$ = + 5 V.

LOW LOGIC = 0 = GND = 0V.

### Digital : Present :

Before we step learning digital Electronics, it is obvious that you should be aware with application(s) of it. Means where you are going to apply the knowledge of the subject.

You must have come across microprocessor based system, computers, digital multimeter, digital display Instruments etc. In short I will say it is becoming ***digital world***. The most wide application is ***Computers***. To learn inside of the computers this is the ***base***. Secondly, as digital is easy to implement, in 90% cases you will find that analog systems are replaced by digital.

Remember that in real world all the physical parameter is going to give you ***analog signal*** ONLY. T convert analog to digital we use ***ADC*** (analog to digital converter). Once we get digital form of the signa it is very easy to operate upon digital data and analyse.

## 1.3 Numbering System :

Recently it is found that, binary numbers are better suited to information processing machine t decimal number system. Now we are going to learn basic decimal system, binary system, octal system hex number system.

Scanned by CamScanner

### 1.3.1 Positional Number :

A number system is defined by digits or numerals. We can combined digits as per our requirement to represent full range. The number system with which we are normally familiar is ***Arabic Numerals***, consists of 10 digits (0, 1, 2, ... 9).

**Decimal System**

The 10 arabic digits can be combined in various ways to represent any number. Fundamental way of constructing a number is to form a sequence or string of digits in which consecutive digits represent consecutive power of 10. For example take 3 digit number → 786. The 786 represent, from left to right, hundreds (7), tens (8) and unit (6).

∴ We can decompose the number as ... (1)

$786 = 7 \times 10^2 + 8 \times 10^1 + 6 \times 10^0$

This method is decimal system. This is good example of positional notation, for numbers. Here you may ask, ***what do you mean by positional notation*** ? It means that, each digit of multidigit number has fixed value (or weight) determined by its POSITION. This number is also called as ***Weighted number*** system. For decimal number systems all the weights used are powers of the number 10, which along with number of available digit types, give the final decimal number. (Equation (1))

Presently we have shown only integer part. One may require to represent ***fractional*** part also. In decimal system, fractional part is denoted by sequences of digits whose weights are negative powers of 10. The integer part and fractional part represents the full number. Both are separated by special symbol, the ***Decimal Point.*** For example 1.414. This number can be decomposed. ... (2)

$1.414 = 1 \times 10^0 + 4 \times 10^{-1} + 1 \times 10^{-2} + 4 \times 10^{-3}$

**Number Base**

The decimal number notation shown in Equation (2), can be written in generalised form also. While writing the equation in generalised form quantity 10 is replaced by r, called **base** or **radix**, of the number system. We will represent number $x_2\ x_1\ x_0 .\ x_{-1}\ x_{-2}$ as ... (3)

$x_2\ x_1\ x_0 .\ x_{-1}\ x_{-2} = x_2 \times r^2 + x_1 \times r^1 + x_0 \times r^0 + x_{-1} \times r^{-1} + x_{-2} \times r^{-2}$

As compared with Equation (2), we have replace $10^i$ by $r^i$.

A positional number notation of this type is called a **base r or** *r – ary number system*. Number systems of our interest are listed as follows :

| System Name | Base r | Digits Employed |
|---|---|---|
| Decimal | 10 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 |
| Binary | 2 | 0, 1 |
| Ter nary | 3 | 0, 1, 2 |
| Octal | 8 | 0, 1, 2, 3, 4, 5, 6, 7, |
| Hexadecimal | 16 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F |

## 1.4 Binary Number System :

We have already seen that binary number system has base of 2 i.e. it has only two digits, namely '0' and '1'. We have also studied *bit*. Bit is nothing but *single* binary line. The weights for binary number can be given by

Scanned by CamScanner

| inary number → | $2^3$ | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | ..... |
|---|---|---|---|---|---|---|---|---|
| quivalent decimal → | 8 | 4 | 2 | 1 | 0.5 | 0.25 | 0.125 | ..... |

By putting r = 2 in Equation (3) we get equation for binary

$\therefore x_2 \times 2^2 + x_1 \times 2^1 + x_0 \times 2^0 + x_{-1} \times 2^{-1} + x_{-2} \times 2^{-2}$ ... (4)

Let's take one example.

If Binary number is 101, then $x_2\, x_1\, x_0 = 101$.

$\therefore \quad x_2 \times 2^2 + x_1 \times 2^1 + x_0 \times 2^0$

$= \quad 1 \text{ X } 2^2 \quad + \quad 0 \text{ X } 2^1 \quad + \quad 1 \text{ X } 2^0$

$= \quad 4 \quad + \quad 0 \quad + \quad 1 = 5 \rightarrow$ decimal.

$\therefore \quad (5)_{10} = (101)_2$ where 2 and 10 shows 'base'.

## 1.4.1 Decimal to Binary Conversion :

In this we will learn, how to convert given decimal number to its equivalent binary. For conversion from decimal to binary we use ***Double Dabble*** method. The steps to be followed are given as follows :

**Step 1** : Take Decimal number.
**Step 2** : Divide number by 2 and note down quotient and Remainder.
**Step 3** : Division of Quotient occurred, by 2.
**Step 4** : Repeat step 3, till quotient is no longer divisible by 2.
**Step 5** : Equivalent binary number is read from bottom (this bit becomes MSB), to top (this bit becomes LSB).

**Note :** (1) Here MSB means Most significant bit means left most bit. LSB means least significant bit means right most bit. Say for example binary number given is,

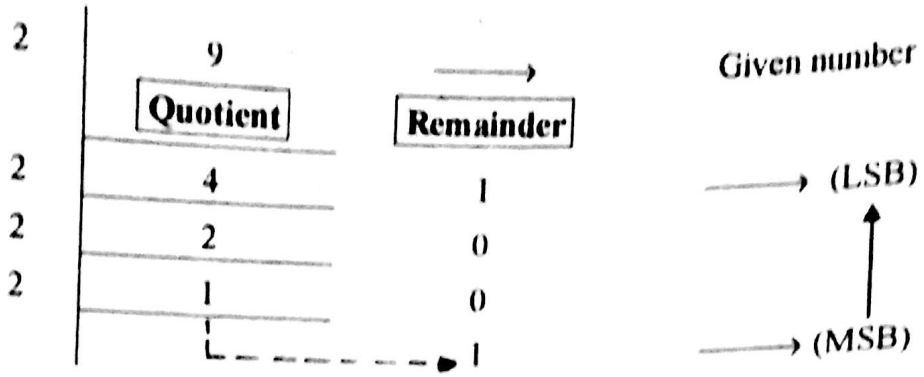
1 0 0 1 0
↑ ↑
MSB LSB

(2) The approach described below is valid for Integer part only. For fractional part different approach is used.

Scanned by CamScanner

**Ex. 2 :** Convert decimal 9 to binary. $(9)_{10} = (?)_2$

**Soln. :**

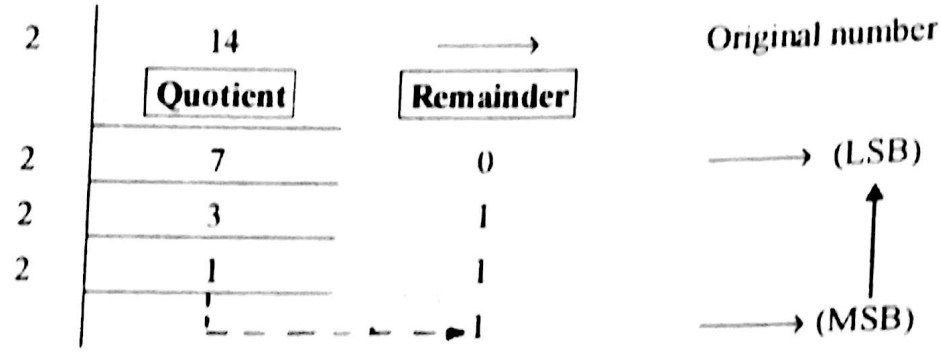



Write number from bottom to top.

$(9)_{10} = (1\ 0\ 0\ 1)_2$.

**Ex. 3 :** Convert decimal 14 to binary $(14)_{10} = (?)_2$

**Soln. :**




Write answer from bottom to top.

$(14)_{10} = (1\ 1\ 1\ 0)_2$

MSB LSB

Scanned by CamScanner

In above case we have taken only Integer number. In case of decimal fraction to binary conversion, steps followed are as follows :

**Step 1 :** Decimal fraction multiplied by 2.

**Step 2 :** Record (from the answer occurred in step 1), the number which is just before decimal point.

**Step 3 :** Again continue multiplication of fraction part occurred in step 2, and recording a number just before the decimal point. From the answer occured after multiplication.

**Step 4 :** Step 3 should be continued depending upon the number of significant bits (digits) required.

Let's solve some examples for the same.

**Ex. 5 :** Convert $(0.61)_{10}$ to equivalent binary fraction.

**Soln. :**

| Decimal fraction | | Base | | Answer | Recorded bit | |
|---|---|---|---|---|---|---|
| 0.61 | × | 2 | = | 1.22 | 1 | (MSB) |
| 0.22 | × | 2 | = | 0.44 | 0 | |
| 0.44 | × | 2 | = | 0.88 | 0 | top to bottom |
| 0.88 | × | 2 | = | 1.76 | 1 | |
| 0.76 | × | 2 | = | 1.52 | 1 | |
| 0.52 | × | 2 | = | 1.04 | 1 | (LSB) |

Now this fraction results in recurring binary value, hence a typical case we continue upto 6 significant bits. Normally 4 to 5 bits after fraction is more than enough.

$\therefore \quad (0.61)_{10} = (0.\ 1\ 0\ 0\ 1\ 1\ 1)_2$

↑ MSB ↑ LSB

**Ex. 6 :** Convert $(0.8)_{10}$ to equivalent binary.

**Soln. :**

| Decimal fraction | | Base | | Answer | Recorded bit | |
|---|---|---|---|---|---|---|
| 0.8 | × | 2 | = | 1.6 | 1 | (MSB) |
| 0.6 | × | 2 | = | 1.2 | 1 | |
| 0.2 | × | 2 | = | 0.4 | 0 | |
| 0.4 | × | 2 | = | 0.8 | 0 | |
| 0.8 | × | 2 | = | 1.6 | 1 | (LSB) |

Now the chain is repeating, therefore stop,

$\therefore \quad (0.8)_{10} = (0.\ 1\ 1\ 0\ 0\ 1)_2$

↑ MSB ↑ LSB.

**Ex. 7 :** Convert $(500\ .\ 21)_{10}$ to its equivalent binary.

**Soln. :** Here we have to operate upon 500 and 0.21 differently. First take 500 i.e. Integer part.

Scanned by CamScanner

| | 500 | |
|---|---|---|
| | **Quotient** | **Remainder** |
| 2 | 250 | 0 |
| 2 | 125 | 0 |
| 2 | 62 | 1 |
| 2 | 31 | 0 |
| 2 | 15 | 1 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |
| | 1 | 1 |

(LSB) ↑ (MSB)

$(500)_{10} = 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0$

Now for fractional part

| Decimal fraction | | Base | | Answer | Recorded bit | |
|---|---|---|---|---|---|---|
| 0.21 | × | 2 | = | 0.41 | 0 | → (MSB) |
| 0.41 | × | 2 | = | 0.82 | 0 | ↓ |
| 0.82 | × | 2 | = | 1.64 | 1 | ↓ |
| 0.64 | × | 2 | = | 1.28 | 1 | ↓ |
| 0.28 | × | 2 | = | 0.56 | 0 | ↓ |
| 0.56 | × | 2 | = | 1.12 | 1 | ↓ |
| 0.12 | × | 2 | = | 0.24 | 0 | → (LSB) |

$\therefore \quad (0.21)_{10} = (0.\ 0\ 0\ 1\ 1\ 0\ 1\ 0)_2$

Now combine both the answer.

$(500\ .\ 21)_{10} = (1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ .\ 0\ 0\ 1\ 1\ 0\ 1\ 0)_2$.

---

**Ex. 8 :** Convert $(37.31)_{10}$ to 10 bit binary.

**Soln. :** As mentioned in question total there has to 10 bits. First take Integer part.

| | 37 | |
|---|---|---|
| | **Quotient** | **Remainder** |
| 2 | 18 | 1 |
| 2 | 9 | 0 |
| 2 | 4 | 1 |
| 2 | 2 | 0 |
| 2 | 1 | 0 |
| | 1 | 1 |

→ (LSB) ↑ → (MSB)

$(37)_{10} = (1\ 0\ 0\ 1\ 0\ 1)_2$

Scanned by CamScanner

From Integer answer we conclude that there are total 6 bit. So you have to fit your fraction answer in 4 bits.

| Decimal fraction | | Base | | Answer | Recorded bit |
|---|---|---|---|---|---|
| 0.31 | × | 2 | = | 0.62 | 0 |
| 0.62 | × | 2 | = | 1.24 | 1 |
| 0.24 | × | 2 | = | 0.48 | 0 |
| 0.48 | × | 2 | = | 0.96 | 0 |

$\therefore (0.31)_{10} = (0.0100)_2$

$\therefore$ Total $(37.31)_{10} = (100101.0100)_2$.

**Shortcut Method**

Here we will learn shortcut method to convert decimal to binary. You are aware with binary positional weights i.e. ... 8, 4, 2, 1, 0.5 0.25 ... So if we want to find out binary of $(13)_{10}$.

$13 = 8 + 4 + 1$

8 4 2 1 ⟶ Weights

1 1 0 1 ⟶ Binary digits

$\therefore (13)_{10} = (1101)_2$.

Same way $(30)_{10} = 16 + 8 + 4 + 2$

$\therefore$ 16 8 4 2 1 ⟶ Weights

1 1 1 1 0 ⟶ Binary digits.

$(30)_{10} = (11110)_2$.

## 1.4.2 Binary to Decimal Conversion :

A binary number can be converted to its equivalent decimal form by multiplying the bits by th binary weights. The bits to the left of decimal point are multiplied by power $2^0$, $2^1$, $2^2$ and so on, fr LSB to MSB respectively. The bits to the right of the decimal are multiplied by negative power of 2 $2^{-1}$, $2^{-2}$, $2^{-3}$ and so on. Let's solve some examples to understand the concept clearly.

**Ex. 9**: Convert $(11100)_2$ to equivalent decimal. $(11100)_2 = (?)_{10}$

**Soln. :**

| | MSB | | | | LSB |
|---|---|---|---|---|---|
| Write binary number ⟶ | 1 | 1 | 1 | 0 | 0 |
| Weights of number ⟶ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

$\therefore (11100)_2 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$

$= 16 + 8 + 4 + 0 + 0$

$= 28$

$\therefore (11100)_2 = (28)_{10}$

**Ex. 10 :** Convert $(110010)_2$ to decimal.

**Soln. :**

| | | | | | | |
|---|---|---|---|---|---|---|
| Write binary number ⟶ | 1 | 1 | 0 | 0 | 1 | 0 |
| Weights ⟶ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

Scanned by CamScanner

$(1\ 1\ 0\ 0\ 1\ 0)_2 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 32 + 16 + 0 + 0 + 2 + 0$

$= 50$

$(1\ 1\ 0\ 0\ 1\ 0)_2 = (50)_{10}$

**Ex. 11 :** Convert $(0\ .\ 1\ 1\ 0\ 1)_2$ to decimal.

**Soln. :** This is example of binary fraction to decimal conversion.

Binary number $\longrightarrow$ 0 . 1 1 0 1

Weights $\longrightarrow$ $2^0$ . $2^{-1}$ $2^{-2}$ $2^{-3}$ $2^{-4}$

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

$= 0 + \frac{1}{2} + \frac{1}{4} + 0 + \frac{1}{16}$

$= \frac{8 + 4 + 1}{16} = \frac{13}{16}$

$= 0.8125$

$(0.1\ 1\ 0\ 1)_2 = (0\ .\ 8\ 1\ 2\ 5)_{10}$.

**Ex. 12 :** Convert $(1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ .\ 0\ 1\ 1\ 1\ 0)_2$ into decimal.

**Soln. :**

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

$= 128 + 64 + 0 + 0 + 8 + 0 + 2 + 1 + 0 + 0.250 + 0.125 + 0.0625 + 0.$

$= (203\ .\ 4375)_{10}$

$(1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ .\ 0\ 1\ 1\ 1\ 0)_2 = (203\ .\ 4375)_{10}$

Now let's write 0 to 15 decimal number and its equivalent binary number. From this Table we will derive some conclusion. The Table is as follows :

| Decimal Number | Binary Number | | | |
|---|---|---|---|---|
| | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 |

**Table 1**

Scanned by CamScanner

**Points :**

(1) $B_0$ bit is LSB and $B_3$ is MSB.

(2) $B_0$ bit toggles or changes state as decimal number increments by 1.

(3) $B_1$ bit changes (toggles) state when decimal is incremented by 2.

(4) $B_2$ changes state when decimal number is incremented by 4.

(5) Finally $B_3$ changes (toggles) state when decimal number is Incremented by eight times.

(6) From point 2, 3, 4 and 5 we conclude that changing also follows 1, 2, 4, 8 i.e. standard binary rule.

(7) While changing from $(0\ 0\ 0\ 0)_2$ to $(0\ 0\ 0\ 1)_2$ only one bit i.e. $B_0$ bit changes state.

(8) While changing from $(0\ 0\ 0\ 1)_2$ to $(0\ 0\ 1\ 0)_2$, two bits i.e. $B_0$ and $B_1$ changes. It means $B_0$ changes from 1 to 0 and $B_1$ changes from 0 to 1.

(9) While changing from $(0\ 0\ 1\ 1)_2$ to $(0\ 1\ 0\ 0)_2$, three bits changes the state i.e. $B_0$, $B_1$ and $B_2$.

(10) Finally when number $(0\ 1\ 1\ 1)_2$ changes to next state i.e. decimal either $(1\ 0\ 0\ 0)_2$, then four bits changes state i.e. $B_0$ to $B_3$. Here $B_0$, $B_1$ and $B_2$ changes from 1 to 0 and $B_3$ changes from 0 to 1.

(11) In number system it is always preferable that one bit changes at a time, because if two or more bits changes the state at a time, because of the speed difference in electronic component some FAULTY state MAY BE INTRODUCED in the sequence. We are going to elaborate this when we will study codes

### 1.4.3 Nibble, Byte and Word in Binary :

We have already studied bit in binary system.

**Nibble :** A group of 4 bits referred as *Nibble*. $B_3 - B_0$ represents nibble e.g. 1011, 1001, 1100 and so on.

**Byte :** A group of 8 bits or 2 nibbles referred as *Byte*. $B_7 - B_0$ represents Byte.

1 0 1 1  1 1 0 0  8 bits / 1 byte / 2 nibbles.

↓ ↓

$B_7$ $B_0$

**Word :** A group of 16 bits or 2 bytes or 4 nibbles referred as word.

e.g. 1 1 1 0  1 1 1 1  1 1 0 0  1 0 0 1 → 1 word / 2 bytes / 4 nibbles / 16 bits.

↓ ↓

$B_{15}$ $B_0$

**Double word :** A group of 32 bits or 2 words or 4 bytes or 8 nibbles is referred as *Double Word*.

## 1.5 Octal Number System :

During the initial stages of development in the field of digital number systems representation, th was a constant need to represent more numbers in Number system. The need arises because in binary have 0 and 1 only. So if you see the page written with 1, 0, 0, 1 string, after sometime your eye will '0' as 1 and '1' as '0' because it becomes monotonous job, and you will confuse your self.

Scanned by CamScanner

The octal number system has base of 8 i.e. it represents number from 0 to 7. Using these digits we represent a number in its octal form. To represent 0 to 7 we require ***three binary bits***. Therefore, it becomes easier to represent number and to performs arithmetic, Instead of using a large string of bits (binary digits).

The weights for actual number system can be given by

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Octal number | ⟶ ... | $8^3$ | $8^2$ | $8^1$ | $8^0$ | $8^{-1}$ | $8^{-2}$ | $8^{-3}$ | ..... |
| Equivalent Decimal | ⟶ ... | 512 | 64 | 8 | 1 | $\frac{1}{8}$ | $\frac{1}{64}$ | $\frac{1}{512}$ | ..... |

... (5)

By putting r = 8 in Equation (3) we get equation for octal.

..... $x_2 \times 8^2 + x_1 \times 8^1 + x_0 \times 8^0 + x_{-1} \times 8^{-1} + x_{-2} \times 8^{-2}$

Let's take an example if given octal number is $(x_2\ x_1\ x_0) = (357)_8$

$\therefore \quad x_2 \times 8^2 + x_1 \times 8^1 + x_0 \times 8^0$

$= \quad 3 \times 8^2 + 5 \times 8^1 + 7 \times 8^0$

$= \quad 192 + 40 + 7$

$= \quad 239 \longrightarrow$ decimal.

$\therefore \quad (357)_8 = (239)_{10}$

### 1.5.1 Decimal to Octal Conversion :

In this, decimal number will be given to you and you have to find out equivalent octal. The method used is ***Octal dabble*** method. The method is same as that of double dabble, only the change is instead of 2, divide number by 8. Secondly, as far as remainder is concern, it may be within 0 to 7. The steps to be followed are given as follows :

**Step 1 :** Take Decimal number.

**Step 2 :** Divide number by 2 and note down quotient and remainder. As stated previously, remainder within the range of 0 to 7 is valid.

**Step 3 :** Division of quotient occured, by 8.

**Step 4 :** Repeat step 3, till quotient is no longer divisible by 8.

**Step 5 :** Equivalent octal number is read from bottom (this digit becomes MSD), to top (this digit becomes LSD).

MSD – Most significant digit.

LSD – Least significant digit.

**Note :** Approach specified above is valid only for Integer part.

Let's first solve some examples to understand the concept.

Scanned by CamScanner

**Ex. 14 :** Convert $(2552)_{10}$ to its equivalent octal form.

**Soln. :**

| | | Quotient | Remainder | |
|---|---|---|---|---|
| 8 | 2552 | | | Original Number |
| 8 | | 319 | 0 | → (LSD) |
| 8 | | 39 | 7 | |
| 8 | | 4 | 7 | |
| | | | 4 | → (MSD) |

$(2552)_{10} = (4770)_8$.

Now let's learn steps to convert fractional part of decimal to its equivalent octal form. Steps are as follows :

**Step 1 :** Decimal fraction multiplied by 8.

**Step 2 :** Record (from the answer occured in step 1), the number which is just before decimal point.

**Step 3 :** Again continue multiplication of fraction part occurred in step 2, and recording a number just before the decimal point from the answer occurred after multiplication.

**Step 4 :** Step 3 should be followed depending upon number of significant digits required.

Let's solve some examples.

**Ex. 15 :** Convert $(0.32)_{10}$ to its equivalent octal form.

**Soln. :**

| Decimal fraction | | Base | | Answer | Recorded bit | |
|---|---|---|---|---|---|---|
| 0.26 | × | 8 | = | 2.56 | 2 | → MSD |
| 0.08 | × | 8 | = | 4.48 | 4 | |
| 0.64 | × | 8 | = | 3.84 | 3 | |
| 0.12 | × | 8 | = | 6.72 | 6 | → LSD |

$\therefore \quad (0.32)_{10} = (0.2436)_8$

MSD LSD

**Ex. 16 :** Convert $(0.26)_{10}$ to its equivalent octal form.

**Soln. :**

| Decimal fraction | | Base | | Answer | Recorded bit | |
|---|---|---|---|---|---|---|
| 0.26 | × | 8 | = | 2.08 | 2 | → MSD |
| 0.08 | × | 8 | = | 0.64 | 0 | |
| 0.64 | × | 8 | = | 5 12 | 5 | |
| 0.12 | × | 8 | = | 0.96 | 0 | |
| 0.96 | × | 8 | = | 7.68 | 7 | → LSD |

$(0.21)_{10} = (0.20507)_8$

Scanned by CamScanner

**Ex. 17 :** Convert $(3000.45)_{10}$ to its equivalent octal number.

**Soln. :** This problem covers both, Integer as well as fractional part. Let's take Integer first.

| | Quotient | Remainder | |
|---|---|---|---|
| 8 | 3000 | | |
| 8 | 375 | 0 | → (LSD) |
| 8 | 46 | 7 | |
| 8 | 5 | 6 | |
| | | 5 | → (MSD) |

$(3000)_{10} = (5670)_8$

| Decimal fraction | | Base | | Answer | Recorded bit | |
|---|---|---|---|---|---|---|
| 0.45 | × | 8 | = | 3.6 | 3 | → MSD |
| 0.6 | × | 8 | = | 4.8 | 4 | |
| 0.8 | × | 8 | = | 6.4 | 6 | |
| 0.4 | × | 8 | = | 3.2 | 3 | → LSD |

$\therefore (0.45)_{10} = (0.3463)_8$

Finally total answer $(3000.45)_{10} = (5670.3463)_8$

### 1.5.2 Octal to Decimal Conversion :

A octal number can be converted to its equivalent decimal form by multiplying the digits by their octal weight. The bits to the left of decimal point are multiplied by power $8^0$, $8^1$, $8^2$ and so on, from LSD to MSD respectively. The bits to the right, are multiplied with negative power of 8 i.e. $8^{-1}$, $8^{-2}$, $8^{-3}$ and so on. We will solve some examples for the same.

---

**Ex. 18 :** Convert $(746)_8$ to its equivalent decimal. $(746)_8 = (?)_{10}$

**Soln. :**

Write Octal number ⟶ 7 4 6

Weights of number ⟶ $8^2$ $8^1$ $8^0$

$= 7 \times 8^2 + 4 \times 8^1 + 6 \times 8^0$

$= 7 \times 64 + 32 + 6 \times 1 = 448 + 32 + 6$

$= 486$

$\therefore (746)_8 = (486)_{10}$

---

**Ex. 19 :** Convert $(0.123)_8$ to equivalent decimal. **(June 99, 1 Mark)**

**Soln. :**

Write octal number ⟶ 0 . 1 2 3

Weights ⟶ $8^0$ . $8^{-1}$ $8^{-2}$ $8^{-3}$

$= 0 \times 8^0 + 1 \times 8^{-1} + 2 \times 8^{-2} + 3 \times 8^{-3}$

$= 0 + \frac{1}{8} + \frac{2}{64} + \frac{3}{512}$

$= 0.1621$

$(0.123)_8 = (0.1621)_{10}$

Scanned by CamScanner

**Ex. 20 :** Convert $(312 . 196)_8$ to decimal.

**Soln. :**

Octal number ⟶ 3 2 1 1 9 6

Weights ⟶ $8^2$ $8^1$ $8^0$ $8^{-1}$ $8^{-2}$ $8^{-3}$

$= 3 \times 8^2 + 2 \times 8^1 + 1 \times 8^0 + 1 \times 8^{-1} + 9 \times 8^{-2} + 6 \times 8^{-3}$

$= 3 \times 64 + 2 \times 8 + 1 \times 1 + \frac{1}{8} + \frac{9}{64} + \frac{6}{512}$

$= 209.40234$

$(321 . 196)_8 = (209.40234)_{10}$

Now, let's see relationship of decimal number, binary number and octal. The full Table is shown as follows :

| Decimal Number | Binary Number | | | | Octal number |
|---|---|---|---|---|---|
| | $B_3$ | $B_2$ | $B_1$ | $B_0$ | |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 2 |
| 3 | 0 | 0 | 1 | 1 | 3 |
| 4 | 0 | 1 | 0 | 0 | 4 |
| 5 | 0 | 1 | 0 | 1 | 5 |
| 6 | 0 | 1 | 1 | 0 | 6 |
| 7 | 0 | 1 | 1 | 1 | 7 |
| 8 | 1 | 0 | 0 | 0 | 10 |
| 9 | 1 | 0 | 0 | 1 | 11 |
| 10 | 1 | 0 | 1 | 0 | 12 |
| 11 | 1 | 0 | 1 | 1 | 13 |
| 12 | 1 | 1 | 0 | 0 | 14 |
| 13 | 1 | 1 | 0 | 1 | 15 |
| 14 | 1 | 1 | 1 | 0 | 16 |
| 15 | 1 | 1 | 1 | 1 | 17 |

**Table 2**

## 1.6 Hexadecimal Number System :

This number system offers an advantage over octal system in the sense that four bits are grouped together to represent a hex digit. As a result more crunching is possible of binary digits. This number system is most widely used in microprocessor based system and computers.

The hexadecimal number system has base of 16 i.e. it represents from 0 to 15 designated as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 A, B, C, D, E, F.

The weights for here number system can be given by

$$\ldots \quad 16^3 \quad 16^2 \quad 16^1 \quad 16^0 \quad 16^{-1} \quad 16^{-3} \quad \ldots\ldots$$

By putting r = 16 in Equation 3 we get equation for hex.

$$\ldots + x_2 \times 16^2 + x_1 \times 16^1 + x_0 \times 16^0 + x_{-1} \times 16^{-1} + x_{-2} \times 16^{-2} + x_{-3} \times 16^{-3} + \ldots\ldots$$

For example if given hex number is $(1\ A\ F)_{16}$

Scanned by CamScanner

$$= 1 \times 16^2 + A \times 16^1 + F \times 16^0$$
$$= 256 + 160 + 15 = 431$$
$$(1\ A\ F)_{16} = (431)_{10}$$

### 1.6.1 Decimal to Hexadecimal (hex.) Conversion :

To convert decimal to hex, we are going to use ***hex dabble*** method. The method is same as that of double dabble. The only change is instead of 2, take base 16. Secondly, remainder can be within the range 0 to 15. The steps are as follows :

**Step 1** : Take decimal number.

**Step 2** : Divide number by 16 and note down quotient and remainder. Remainder should be between the range 0 to 15.

**Step 3** : Division of quotient, occurred by 16.

**Step 4** : Repeat step 3 till quotient is no longer divisible by 16.

**Step 5** : Equivalent hex numbers are read from bottom (this become MSD), to top (this becomes LSD).

**Note** : (1) 0 to 9 of hex is 0 – 9 of decimal.

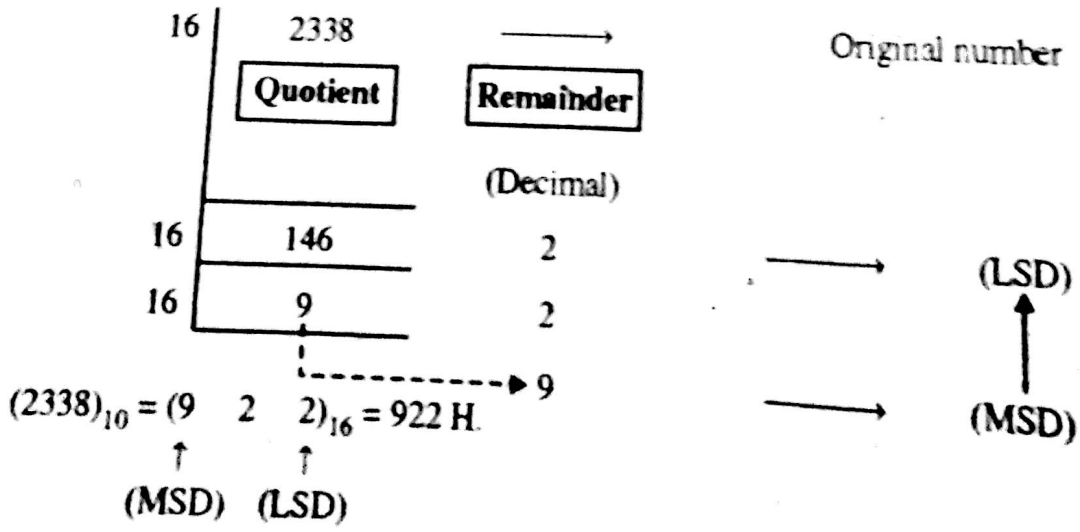
$(10)_{10}$ = A
$(11)_{10}$ = B
$(12)_{10}$ = C
$(13)_{10}$ = D
$(14)_{10}$ = E
$(15)_{10}$ = F

(2) To represent hex number we use 'H' or 'h' e.g. 12 A F H (12 A F h)

(3) The method mentioned above is valid only for Integer

**Ex. 21 :** Convert $(2338)_{10}$ to its equivalent hex format. $(2338)_{10} = (?)_{16}$ **(June 99, 1 Mark)**

**Soln. :**




Scanned by CamScanner

**Soln. :**

| | | Quotient | Remainder | (Decimal) | | (hex) | |
|---|---|---|---|---|---|---|---|
| 16 | 1023 | | | | ⟶ Original number | | |
| 16 | 63 | | | 15 | ⟶ | F | (LSD) |
| 16 | 3 | | | 15 | ⟶ | F | |
| | | | | 3 | ⟶ | 3 | (MSD) |

$(1023)_{10} = (3\ F\ F)_{16} = 3FFH.$

MSD LSD

COLLEGE OF Sc. IT 743

**Ex. 23 :** Convert $(461)_{10}$ to its equivalent hex format.

**Soln. :**

| | | Quotient | Remainder | (Decimal) | | (hex) | |
|---|---|---|---|---|---|---|---|
| 16 | 461 | | | | | | |
| 16 | 28 | | | 13 | ⟶ | D | (LSD) |
| 16 | 1 | | | 12 | ⟶ | C | |
| | | | | 1 | ⟶ | 1 | (MSD) |

$(461)_{10} = (1CD)_{16} = 1CDH.$

Now we will see how to convert fractional part of decimal number to its equivalent hex format. The method is same as previous methods i.e. multiply with base and record the digits.

**Step 1 :** Decimal fraction multiplied by 16.

**Step 2 :** Record (from the answer occurred in step 1), the number which is just before decimal point.

**Step 3 :** Again continue multiplication of fractional part occurred in step 2, and recording a number just before the decimal point from the answer occurred after multiplication.

**Step 4 :** Step 3 should be followed depending upon number of significant digits required.

Let's solve some example.

**Ex. 24 :** Convert $(0.122)_{10}$ to its equivalent hex. **(June 99, 1 Mark)**

**Soln. :**

| Decimal fraction | | Base | | Answer | Recorded (Decimal) | | Bit (hex) | |
|---|---|---|---|---|---|---|---|---|
| 0.122 | × | 16 | = | 1.952 | 1 | ⟶ | 1 | ⟶ (MSD) |
| 0.952 | × | 16 | = | 15.232 | 15 | ⟶ | F | |
| 0.232 | × | 16 | = | 3.712 | 3 | ⟶ | 3 | |
| 0.712 | × | 16 | = | 11.392 | 11 | ⟶ | B | |
| 0.392 | × | 16 | = | 6.272 | 6 | ⟶ | 6 | ↓ |
| 0.272 | × | 16 | = | 4.352 | 4 | ⟶ | 4 | ⟶ (LSD) |

$\therefore \quad (0.122)_{10} = (0.1\ F\ 3\ B\ 6\ 4)_{16} = 0.1\ F\ 3\ B\ 64\ H$

MSD LSD

Scanned by CamScanner

**Ex. 25 :** Convert $(0.48)_{10}$ to its equivalent hex.

**Soln. :**

| Decimal fraction | | Base | | Answer | Recorded Bit (Decimal) | | (hex) | |
|---|---|---|---|---|---|---|---|---|
| 0.48 | | | | | | | | → (MSD) |
| 0.68 | × | 16 | = | 7.68 | 7 | → | 7 | |
| 0.88 | × | 16 | = | 10.88 | 10 | → | A | |
| 0.08 | × | 16 | = | 14.08 | 14 | → | E | |
| 0.28 | × | 16 | = | 1.28 | 1 | → | 1 | |
| | × | 16 | = | 4.48 | 4 | → | 4 | → (LSD) |

$(0.48)_{10} = (0.7AE14)_{16}$.

**Ex. 26 :** Convert $(2003.31)_{10}$ to equivalent hex.

**Soln. :** In this problem we have to handle Integer and fractional part separately.

| 16 | 2003 | | | Original Number | |
|---|---|---|---|---|---|
| | Quotient | Remainder | | hex | |
| | | Decimal | | | |
| 16 | 125 | 3 | → | 3 | → LSD |
| 16 | 7 | 13 | → | D | |
| | | 7 | → | 7 | → MSD |

$(2003)_{10}$ = (7 D 3 H)

| Decimal fraction | | Base | | Answer | Recorded bit | |
|---|---|---|---|---|---|---|
| | | | | | Decimal | hex |
| 0.31 | × | 16 | = | 4.96 | (4) | (4) |
| 0.96 | × | 16 | = | 15.36 | (15) | (F) |
| 0.36 | × | 16 | = | 5.76 | (5) | (5) |
| 0.76 | × | 16 | = | 12.16 | (12) | (C) |
| 0.16 | × | 16 | = | 2.56 | (2) | (2) |

$\therefore$ $(2003.31)_{10} = (7\ D\ 3.4\ F\ 5\ C\ 2)_4$

## 1.6.2 Hexadecimal to Decimal Conversion :

The hexadecimal number can be converted to its equivalent decimal form by multiplying the digits by their hex weights. The bits to the left of decimal point are multiplied by power $16^0$, $16^1$, $16^2$ and so on, from LSD to MSD respectively. The bits to the right, are multiplied with negative power of 8 i.e. $16^{-1}$, $16^{-2}$, $16^{-3}$ and so on.

Let's solve some examples.

**Ex. 27 :** Convert $(2\ AF)_{16}$ to equivalent decimal.

**Soln. :**

| | | (M S D) | | (L S D) |
|---|---|---|---|---|
| Hex Number | → | 2 | A | F |
| | → | 2 | 10 | 15 |
| Weights | → | $16^2$ | $16^1$ | $16^0$ |

$= 2 \times 16^2 + A \times 16^1 + F \times 16^0$

$= 2 \times 256 + 10 \times 16 + 15 \times 1$

$= 512 + 160 + 15$

$= (687)_{10}$

Scanned by CamScanner

**Ex. 28 :** Convert $(0.F27)_{16}$ to its equivalent decimal form.

**Soln. :**

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | MSD ↓ | | | LSD ↓ |
| Hex Number | ⟶ | 0 | . F | 2 | 7 | |
| Weights | ⟶ | $16^0$ | . $16^{-1}$ | $16^{-2}$ | $16^{-3}$ | |

$$= 0 \times 16^0 + F \times 16^{-1} + 2 \times 16^{-2} + 7 \times 16^{-3}$$

$$= 0 + \frac{15}{16} + \frac{2}{16^2} + \frac{7}{16^3}$$

$$= (0.94702)_{10}$$

**Ex. 29 :** Convert $(10F.2F)_{16}$ to its equivalent decimal number.

**Soln. :**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Hex Number | ⟶ | 1 | 0 | F | . | 2 | F |
| | | 1 | 0 | 15 | . | 2 | 15 |
| Weight | ⟶ | $16^2$ | $16^1$ | $16^0$ | . | $16^{-1}$ | $16^{-2}$ |

$$= 1 \times 16^2 + 0 \times 16^1 + 15 \times 16^0 + 2 \times 16^{-1} + 15 \times 16^{-2}$$

$$= 271.1835937$$

$(10\ F.2F)_{16} = (271.1835937)_{10}$.

Now we will form a Table containing decimal, binary, octal, and Hex.

| Decimal Number | Binary Number | | | | | Octal Number | Hexadecimal Number |
|---|---|---|---|---|---|---|---|
| | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 0 | 2 | 2 |
| 3 | 0 | 0 | 0 | 1 | 1 | 3 | 3 |
| 4 | 0 | 0 | 1 | 0 | 0 | 4 | 4 |
| 5 | 0 | 0 | 1 | 0 | 1 | 5 | 5 |
| 6 | 0 | 0 | 1 | 1 | 0 | 6 | 6 |
| 7 | 0 | 0 | 1 | 1 | 1 | 7 | 7 |
| 8 | 0 | 1 | 0 | 0 | 0 | 10 | 8 |
| 9 | 0 | 1 | 0 | 0 | 1 | 11 | 9 |
| 10 | 0 | 1 | 0 | 1 | 0 | 12 | A |
| 11 | 0 | 1 | 0 | 1 | 1 | 13 | B |
| 12 | 0 | 1 | 1 | 0 | 0 | 14 | C |
| 13 | 0 | 1 | 1 | 0 | 1 | 15 | D |
| 14 | 0 | 1 | 1 | 1 | 0 | 16 | E |
| 15 | 0 | 1 | 1 | 1 | 1 | 17 | F |
| 16 | 1 | 0 | 0 | 0 | 0 | 20 | 10 |
| 17 | 1 | 0 | 0 | 0 | 1 | 21 | 11 |
| 18 | 1 | 0 | 0 | 1 | 0 | 22 | 12 |
| : | : | : | : | : | : | : | : |

Scanned by CamScanner

As shown hexadecimal number is from 0 to 9, A to F, 10 to 19, 1A to 1F, 20 to 29, 2A to 2F and so on. Octal will be from 0 to 7, 10 to 17, 20 to 27, 30 to 37 and so on.

## 1.7 Conversion :

Under this section we are going to study conversion of one type of numbering to other type i.e. Hex to octal, Hex to binary, binary to octal etc. Uptill now we have studied decimal to hex / binary / octal or Vice Versa.

### 1.7.1 Binary to Hex :

For conversion we group binary bits in ***group of 4 bits*** (Nibble) and we will find out corresponding hex digit.

Say for an example.

Given binary number is 1 1 0 0 0 1 1 0.

Now start grouping from LSB bit and travel right to left.

1 1 0 0  0 1 1 0  ⟶ Binary

MSB  LSB

Now refer Table 3. You will find that $(1\ 1\ 0\ 0)_2 = (C)_{16}$ and $(0\ 1\ 1\ 0)_2 = (6)_{16}$.

$\therefore$ $(1\ 1\ 0\ 0 \quad 0\ 1\ 1\ 0)_2$ = (C 6 H)

The above method was for Integer. For fraction, basic remains same i.e. you have to group 4 bits. But here starting point is after ***Fraction Point***. For example if given binary number is 0.01011101.

then 0. 0 1 0 1  1 1 0 1 ⟶ Binary

⇓ ⇓

0 . 5 D ⟶ Hex

$(0.\ 0\ 1\ 0\ 1 \quad 1\ 1\ 0\ 1)_2$ = (0. 5 D H).

---

**Ex. 30 :** Convert 1 1 0 0 1 1 0 . 0 1 1 0 1 0 to its equivalent Hex form.

**Soln. :** If you count Integer and fraction part of given number, you will find that we have 7 bits for Integer and 6 bits for fraction. Hex we have to add extra zeros. After adding extra zeros we get.

0 1 1 0  0 1 1 0 . 0 1 1 0  1 0 0 0 ⇒ binary

6  6 . 6  8 ⇒ hex.

Here adding zero's in Integer part at MSB bits and adding zero's in fractional part at LSB will not change the actual binary number.

$\therefore$ $(1\ 1\ 0\ 0\ 1\ 1\ 0.\ 0\ 1\ 1\ 0\ 1\ 0)_2$ = $(66\ .\ 68)_{10}$

(Leading zeros in Integer part and trailing zeros in fractional part are ***Don't Care*** for us.)

---

**Ex. 31 :** Convert $(1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ .\ 1\ 1\ 1\ 0)_2$ to hex.

**Soln. :**

dded ← 0 0 1 1  0 0 1 0  1 0 1 1  1 1 1 0 ⇒ binary

s  3  2  B  E ⇒ Hex

$(1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ .\ 1\ 1\ 1\ 0)_2$ = $(3\ 2\ B\ .\ E)_{16}$.

Scanned by CamScanner

**Ex. 32 :** Convert $(1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1)_2$ to hex

**Soln. :**

$$(\underbrace{0\ 0\ 1\ 1}_{\Downarrow}\ \underbrace{0\ 1\ 0\ 1}_{\Downarrow}\ \underbrace{1\ 1\ 0\ 1}_{\Downarrow}\ \underbrace{0\ 1\ 1\ 1}_{\Downarrow})_2$$

$$(3\qquad 5\qquad D\qquad 7)_{16}$$

$\therefore\ (1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1)_2 = (3\ 5\ D\ 7)_{16}$

## 1.7.2 Hex to Binary Conversion :

We know that one digit of hex is group of four bits of binary. So converting hex to binary is exactly reverse of what we have done in Section 1.7.1. Binary to hex conversion.

For example if HEX is $(A\ 3\ 2)_{16}$ then binary is

| A | 3 | 2 | ⟶ | hex | [Refer Table 3] |
|---|---|---|---|---|---|
| ⇓ | ⇓ | ⇓ | ⟶ | binary | |
| 1 0 1 0 | 0 0 1 1 | 0 0 1 0 | | | |

$\therefore\ (A\ 32\ H) = (1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2$

**Ex. 33 :** Convert $(F\ E\ A)_{16}$ to equivalent binary

**Soln. :** Refer Table 3 for conversion.

| F | E | A | ⟶ | hex |
|---|---|---|---|---|
| ⇓ | ⇓ | ⇓ | | |
| 1 1 1 1 | 1 1 1 0 | 1 0 1 0 | ⟶ | binary |

$\therefore\ (F\ E\ A)_{16} = F\ E\ A\ H = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0)_2$

Scanned by CamScanner

**Ex. 36 :** Convert (0 1 1 0 0 1) to its equivalent octal number.

**Soln. :**

$$\underline{0\ 1\ 1}\ \ \underline{0\ 0\ 1} \longrightarrow \text{Binary}$$
$$\Downarrow \qquad \Downarrow \longrightarrow \text{Grouping}$$
$$3 \qquad 1 \longrightarrow \text{Octal.}$$

$\therefore \quad (0\,1\,1\,0\,0\,1)_2 = (31)_8$

**Ex. 37 :** Convert $(0\,.\,1\,1\,0\,1\,0\,1)_2$ to octal.

**Soln. :**

$$0\ \ \underline{1\ 1\ 0}\ \ \underline{1\ 0\ 1} \longrightarrow \text{Binary}$$
$$\Downarrow \qquad \Downarrow \longrightarrow \text{Grouping}$$
$$0 \quad 6 \qquad 5 \longrightarrow \text{Octal}$$

$\therefore \quad (0\,.\,1\,1\,0\,1\,0\,1)_2 = (0\,.\,65)_8$

**Ex. 38 :** Convert $(1\,0\,0\,1\,0\,1\,0\,.\,0\,1\,1\,0\,0\,1\,0\,1)_2$ to octal.

**Soln. :**

$$\underline{0\ 0\ 1}\ \ \underline{0\ 1\ 0}\ .\ \underline{0\ 1\ 1}\ \ \underline{0\ 0\ 1}\ \ 0\ 1$$

We start combining groups of three bits in either direction from ***Fractional Point***. If required put zeros to extreme left or extreme right of the binary number to complete three bit group, as it helps in grouping but does not affect the number.

i.e.

$$\underline{0\ 0\ 1}\ \ \underline{0\ 0\ 1}\ \ \underline{0\ 1\ 0}\ .\ \underline{0\ 1\ 1}\ \ \underline{0\ 0\ 1}\ \ \underline{0\ 1\ 0} \longrightarrow \text{Binary}$$
$$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$$
$$1 \quad 1 \quad 2 \quad . \quad 3 \quad 1 \quad 2$$

$\therefore \quad (1\,0\,0\,1\,0\,1\,0\,.\,0\,1\,1\,0\,0\,1\,0\,1)_2 = (1\,1\,2\,.\,312)_8.$

**Ex. 39 :** Convert $(0\,.\,1\,0\,1\,1)_2$ to octal.

**Soln. :** $(0\,.\,1\,0\,1\,1)_2 = (0\,.\,\underline{1\,0\,1}\ \ \underline{1\,0\,0})_2 = (0\,.\,5\,4)_8.$

**Ex. 40 :** Convert $(1\,1\,0\,1\,1\,0\,1)_2$ to octal number.

**Soln. :** We put zeros to be left of number as it does not change the number. Grouping done from right to left i.e. LSB to MSB.

$\therefore \quad (1\,1\,0\,1\,1\,0\,1)_2 = (\underline{0\,0\,1}\ \ \underline{1\,0\,1}\ \ \underline{1\,0\,1})_2 = (1\,5\,5)_8.$

## 1.7.4 Octal to Binary Conversion :

This is exactly reverse of what we studied in section 1.7.3. In this case octal number will be given to you. You have to refer Table 2 and go on putting ***3 bit binary*** value for the same. Let's solve some examples.

**Ex. 41 :** Convert $(746)_8$ to binary.

**Soln. :** In binary $\left.\begin{matrix} 7 = 1\,1\,1 \\ 4 = 1\,0\,0 \\ 6 = 1\,1\,0 \end{matrix}\right\}$ 3 bit binary representation.

$\therefore \quad (746)_8 = (1\,1\,1\,1\,0\,0\,1\,1\,0)_2.$

Scanned by CamScanner

**Ex. 42 :** Convert $(0.543)_8$ to binary.

**Soln. :**

$$\begin{array}{cccc} 0 & 5 & 4 & 3 \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \\ 0 & .\ 1\,0\,1 & 1\,0\,0 & 0\,1\,1 \end{array} \longrightarrow \text{octal (Refer Table 2)}$$

$$\therefore\ (0.543)_8 = (0.101100011)_2 \longrightarrow \text{binary}$$

**Ex. 43 :** Convert $(1\,2\,3\,.\,5\,6\,7)_8$ to binary.

**Soln. :**

$$\begin{array}{cccccc} 1 & 2 & 3 & 5 & 6 & 7 \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow \\ 0\,0\,1 & 0\,1\,0 & 0\,1\,1 & 1\,0\,1 & 1\,1\,0 & 1\,1\,1 \end{array} \longrightarrow \text{Octal (Refer Table 2)}$$

$$\therefore\ (123.567)_8 = (001010011.101110111)_2 \longrightarrow \text{binary equivalent}$$

$$= (1010011.101110111)_2.$$

## 1.7.5 Hex to Octal Conversion :

4 bit grouping

To convert hex to octal, convert hex number to equivalent binary (i.e. represent each hex digit by four binary bits) and then combine groups of three bits (as each octal digit is combination of three bits.) Let's solve some examples.

**Ex. 44 :** Convert $(4\ CE)_{16}$ to its equivalent octal form.

**Soln. :**

$$\begin{array}{cccc} (\ 4 & C & E\ )_{16} & \longrightarrow \text{hex} \\ \Downarrow & \Downarrow & \Downarrow & \\ (0\,1\,0\,0 & 1\,1\,0\,0 & 1\,1\,1\,0)_2 & \longrightarrow \text{binary} \end{array}$$

Hex to binary Conversion.

$$\begin{array}{cccc} \underline{0\,1\,0} & \underline{0\,1\,1} & \underline{0\,0\,1} & \underline{1\,1\,0}\ )_2 \longrightarrow \text{binary} \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \\ (\ 2 & 3 & 1 & 3\ )_8 \longrightarrow \text{Octal} \end{array}$$

Binary to octal Conversation.

Note that grouping of bits in 'three' is done from LSB to MSB.

$\therefore\ (4\ CE)_{16} = (2\,3\,1\,3)_8.$

**Ex. 45 :** $(0\ .\ 12\ E)_{16} = (?)_8$

**Soln. :** The question is to convert hex to octal.

$$\begin{array}{cccc} (0\ .\ 1 & 2 & E\ )_{16} & \longrightarrow \text{hex} \\ \Downarrow & \Downarrow & \Downarrow & \\ (0\ .\ 0\,0\,0\,1 & 0\,0\,1\,0 & 1\,1\,1\,0)_2 & \longrightarrow \text{binary} \end{array}$$

hex to binary.

$$\begin{array}{cccc} (0\ .\ \underline{0\,0\,0} & \underline{1\,0\,0} & \underline{1\,0\,1} & \underline{1\,1\,0}\ )_2 \longrightarrow \text{binary} \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \\ (\ 0\ .\ 0 & 4 & 5 & 6)_8 \longrightarrow \text{Octal} \end{array}$$

binary to octal.

**Note that bits are grouped starting from the *fractional point* and moving right.**

$\therefore\ (0\ .\ 12E)_{16} = (0\ .\ 0456)_8.$

Scanned by CamScanner

**Ex. 46 :** Convert $(BB\ .\ 0A)_{16}$ to its equivalent octal form.

**Soln. :**

$($ B B 0 A $)_{16}$ ⟶ hex

$(1\ 0\ 1\ 1\quad 1\ 0\ 1\ 1\quad 0\ 0\ 0\ 0\quad 1\ 0\ 1\ 0)_2$ ⟶ binary

Binary to hex conversion.

Now put additional zeros to extreme left and right to group in three bits.

$(0\ 1\ 0\quad 1\ 1\ 1\quad 0\ 1\ 1\quad 0\ 0\ 0\quad 0\ 1\ 0\quad 1\ 0\ 0)_2$ ⟶ binary

$(2\quad 7\quad 3\quad 0\quad 2\quad 4)_8$ ⟶ octal

$(BB\ .\ OA)_{16} = (273\ .\ 024)_8$.

### 1.7.6 Octal to Hex Conversion :

In this, the operation is totally reverse than what we have done in section 1.7.5. First you have to convert octal to binary (group of 3 bits). After that convert binary to equivalent hex, (group of 4 bits). Let's solve some examples.

**Ex. 47 :** Convert $(371)_8$ to equivalent hexadecimal form.

**Soln. :**

$(3\quad 7\quad 1)_8$ ⟶ Octal

$(0\ 1\ 1\quad 1\ 1\ 1\quad 0\ 0\ 1)_2$ ⟶ Binary

Octal to binary Conversion

Binary number is 0 1 1 1 1 1 0 0 1. Group them into four bits, starting from right to left.

$(0\quad 1\ 1\ 1\ 1\quad 1\ 0\ 0\ 1)_2$ ⟶ binary

$(0\quad F\quad 9)_6$ ⟶ hex

binary to hex conversion

Now in above case, we have leading zero, which is of no use.

∴ Final answer is

$(371)_8 = (F9)_H = (F9)_{16}$.

**Ex. 48 :** Convert $(0\ .\ 207)_8$ to hex form.

**Soln. :**

$(0.\quad 2\quad 0\quad 7)_8$ ⟶ Octal

$(0.\quad 0\ 1\ 0\quad 0\ 0\ 0\quad 1\ 1\ 1)_2$ ⟶ binary

Octal to binary.

Binary number is 0 . 0 1 0 0 0 0 1 1 1. Now start grouping of 4 bits starting after ***fractional point*** and from left to right. Add additional zeros to extreme right if required.

1 0 0 0 1 1 1 0 . 1 1 1 0 0 1

You have to add extra zeros.

$(0\quad 0\ 1\ 0\ 0\quad 0\ 0\ 1\ 1\quad 1\ 0\ 0\ 0)_2$ → binary

$(0\quad 4\quad 3\quad 8)$ → hex

$(0.207)_8 = (0.438)_{16}$

binary to hex conversion

Scanned by CamScanner

**Ex. 49 :** Convert $(436.71)_8$ to its equivalent hex.

**Soln. :**

$$\begin{array}{cccccccc} (4 & 3 & 6 & . & 7 & 1)_8 & \rightarrow & \text{Octal} \\ \Downarrow & \Downarrow & \Downarrow & & \Downarrow & \Downarrow & & \\ (100 & 011 & 110 & . & 111 & 001)_2 & \rightarrow & \text{Binary} \end{array}$$

Octal to binary conversion

Binary number is $(100011110.111001)_2$

Now start grouping binary bits in group of 4 bits. Add extra zeros on extreme right or left if required.

1 0 0 0 | 1 1 1 0 . 1 1 1 0 | 0 1

You have to add extra zeros.

$$\begin{array}{cccccccc} 1\,0\,0\,0 & & 1\,1\,1\,0 & . & 1\,1\,1\,0 & 0\,1\,0\,0 & \rightarrow & \text{binary} \\ \Downarrow & & \Downarrow & & \Downarrow & \Downarrow & & \\ 8 & & E & . & E & 4 & \rightarrow & \text{hex} \end{array}$$

$\therefore (436.71)_8 = (8E.E4)_{16}$.

**Conclusion :**

Easy way to remember conversion rules is as follows :

(i) Decimal to binary – division by 2.

(ii) Decimal to octal – division by 8.

(iii) Decimal to hex – division by 16.

For converting a decimal number to any form, divide the decimal number successively by the base of that form, as illustrated above.

(iv) Binary to decimal – Multiply by power of 2.

(v) Octal to decimal – Multiply by power of 8.

(vi) Hex to decimal – Multiply by power of 16.

For converting any number system to decimal, multiply the digits by powers of that system.

(vii) Octal to binary – Represent each octal digit by three bits.

(viii) Hex to binary – Represent each hex digit by four bits.

(ix) Binary to octal – Combine groups of 3 bits.

(x) Binary to Hex – Combine groups of 4 bits.

Finally

(xi) Hex to octal – Convert hex to binary and binary to octal.

(xii) Octal to hex – Convert octal to binary and binary to hex.

## 1.8 Floating – Point Numbers :

All the numbering systems which we have studied so far fall into the category of base r fixed p numbers. Fixed point number means ***The fractional point*** (decimal point) is once finalised will be fi The fractional point, is fixed within each n bit number word. For example if my system is of 8 bit, define that my fractional point is at $5^{th}$ bit then binary format will be,

Scanned by CamScanner

$B_4\ B_3\ B_2\ B_1\ B_0\ .\ B_{-1}\ B_{-2}\ B_{-3}$

8 bits — Fractional point

Here you should note that once, my number of bits are finalised for a particular system, you cannot change it abruptly. So for above format minimum number we can represent is $('0')_2$ i.e.

$B_4\ B_3\ B_2\ B_1\ B_0\ .\ B_{-1}\ B_{-2}\ B_{-3} = (00000.000)_2$. Maximum number I can represent is $B_4\ B_3\ B_2\ B_1\ B_0\ .\ B_{-1}\ B_{-2}\ B_{-3} = (1\ 1\ 1\ 1\ 1\ .\ 1\ 1\ 1)_2$ i.e. $(31\ .\ 875)_{10}$

To increase the range of the numbers to be represented, the obvious solution is to increase number of bits. Let's take example for 32 bit system. Take that integer part is of 32 bit. Then maximum number you can represent is

$2^{32} - 1 = 4,\ 294,\ 967,\ 295 = 4.29 \times 10^9$.

But if we want to represent Angstroms number i.e. $4.07 \times 10^{26}\ A^0$ then it is not possible in 32 bit. Secondly minimum number I can represent in above case is $(0)_{10}$ = zero. But if I want to represent suppose mass of electron in kilogram i.e. $1.67 \times 10^{-27}$ kg, it is not possible.

Here you may argue to increase number of bits, but just increasing number of bits is not the solution because your system should support that many number of bits. So finally the conclusion is, for a ***fixed point number*** representation one cannot represent very large number and very small number.

**Remedy :**

The remedy over above problem is, to represent very large and very small numbers, we have to switch over to **Scientific Notation.** In scientific notation, number is in the form of $N = M \times 10^E$, where M and E are fixed points numbers called, **Mantissa** and **Exponent** respectively.

E is an integer that specifies the number of zeros to be appended to M, to obtain Floating point number N. Here you may ask question that why the ***Version of Scientific Number (notation)*** is called ***FLOATING POINT***. To understand this we will take simple example.

Let's say $(1\ 0\ 0\ 0\ .\ 2\ 3\ 4)_{10}$

This number can be represented as

| | | |
|---|---|---|
| $1000234$ | $\times\ 10^{-3}$ | $10.00234 \times 10^2$ |
| $100023.4$ | $\times\ 10^{-2}$ | $1.000234 \times 10^3$ |
| $1000234$ | $\times\ 10^{-1}$ | $0.1000234 \times 10^4$ |
| $1000.234$ | $\times\ 10^{0}$ | |
| $100.0234$ | $\times\ 10^{1}$ | |

In representation of $(1000\ .\ 234)_{10}$, you must have observed that ***decimal point*** is not fixed as such. It can ***FLOAT*** from number to number depending upon power of 10. Normally in scientific notation we represent number as $M \times B^E$. Therefore depending upon our requirement we set E and in M will allow ***decimal point*** to float. Therefore it is called as ***floating point numbering system***.

In $N = \pm M \times B^E$, B = base of floating point number system. We have taken base = 10 for decimal system. But for binary number it will be '2', B = 2. Let's see the format of floating point number system.

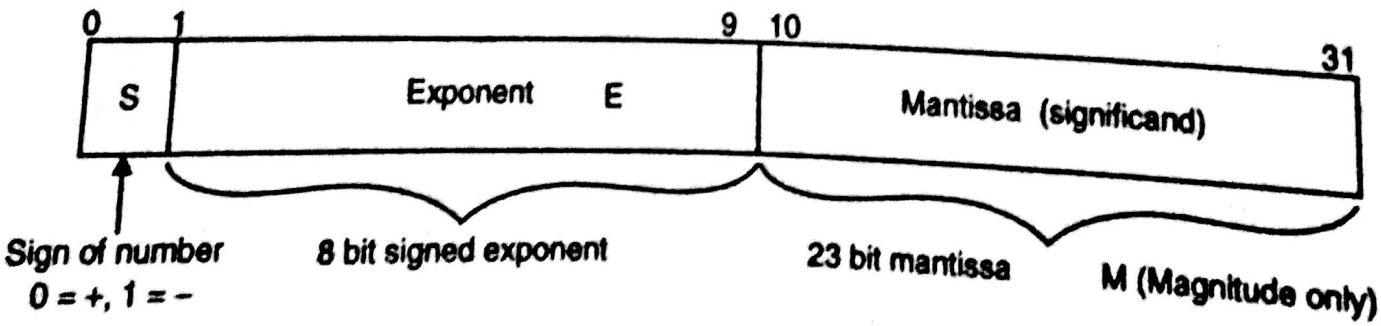



**Fig. 1.9 : Format of typical 32 bit floating point**

Scanned by CamScanner

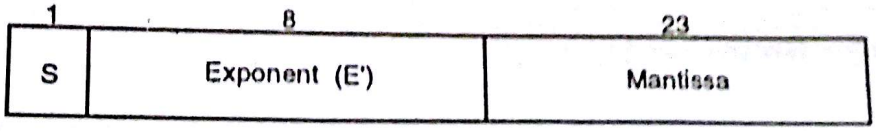
Fig. 1.9 shows 32 bit format for floating point. The sign S, indicates sign of mantissa. This bit is assigned left most position to maintain compatibility with the sign conversation for fixed point binary numbers. Eight bits are used for exponent E, with remaining 23 bits for mantissa. E represents signed binary integer, so that exponent value can range from – 128 to + 127.

Floating point numbers are standardised by IEEE. According to it we have to represent floating point number in the form

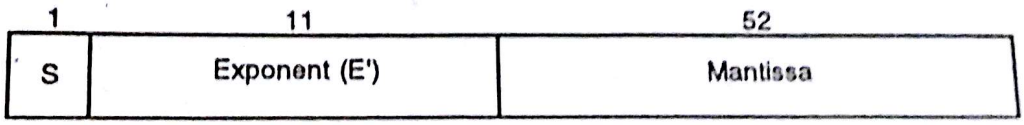
$\pm 1 . M \times 2^{E - 127}$ (Single Precision – 32 bit)

$\pm 1 . M \times 2^{E' - 1023}$ (Double Precision – 64 bit).

Here M = Mantissa, E = E' – 127. The format is shown in Fig. 1.10.




(a) Single precision




(b) Double precision

**Fig. 1.10 : IEEE 754 format**

$E = E' - 127 \therefore E' = E + 127$ E = Exponent of number.

The range for E' is 0 to 255.

Before we go deep into exponent first understand what is 1.M. Whenever any number is given to you, to represent it in floating point number system, you have to bring down to the format 1.M. Say for examples where the number given is $0 . 0\ 1\ 1\ 0 \times 2^6$. Presently the number has form 0 . M M = 0 1 1 0. But we want 1 . M. So now shift number to left side till you get 1 . M. While shifting the number you have to adjust exponent also.

$0 . 0\ 1\ 1\ 0 \times 2^6 \longrightarrow$ Actual number
$0 . 1\ 1\ 0 \times 2^5 \longrightarrow$ After 1st shift
$1 . 1\ 0 \times 2^4 \longrightarrow$ after 2nd shift.

} Shift operation:

Now stop here as we have 1 . M. Now M = 10.

$\therefore$ The floating point number has.

M = 10

S = plus (positive)

$E' = E + 127 = 4 + 127 = 131 = (1\ 0\ 0\ \ 0\ 0\ 11)_2$

$\therefore$ Number is as shown in Fig. 1.11.

Scanned by CamScanner

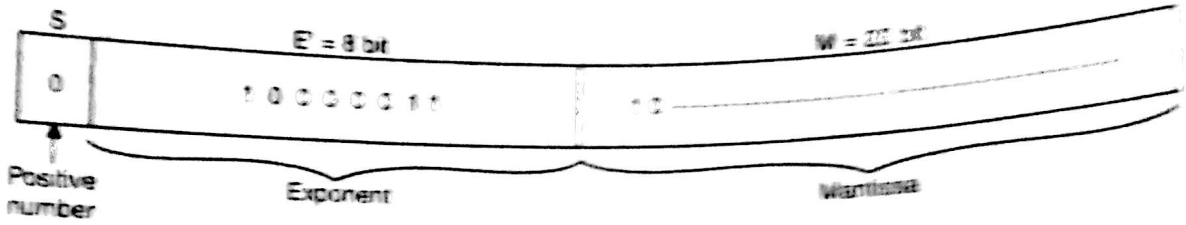



Fig. 1.11

Fig. 1.11 Floating point representation of $1.00 \times 2^6$

∴ Finally number is $(0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ \ldots)_2$

In hex we have $(4\ 3\ 8\ 0\ 0\ 0\ 0\ 0)_{16}$

### 1.8.1 De-Normalised – Normalised :

While representing $0.001 \times 2^5$ number in floating point, you have shifted the bits and adjusted exponent, till you get 1.M. The number we get after performing this operation is called 'Normalised Number' i.e. brought down to our requirement. Secondly if you see floating point number it has format = $1.M \times 2^{E-127}$. While representing it in IEEE format we simply forget '1' in 1.M. The answer for the same is, It is by default that we should have 1 before 'fractional point' Therefore the bit we don't take base (B) into account because it is default, we don't take '2' into account, only while representing it. But whenever we perform any operation, it will be taken into account.

From above discussion you must be now aware with de-normalised number. De-normalised number is 'RAW' number which doesn't follow 1.M format.

### Range :

Each floating point format (Single precision/double precision) has capability to represent the numbers within 'Certain' range. Generalised graphical representation is shown in Fig. 1.12.

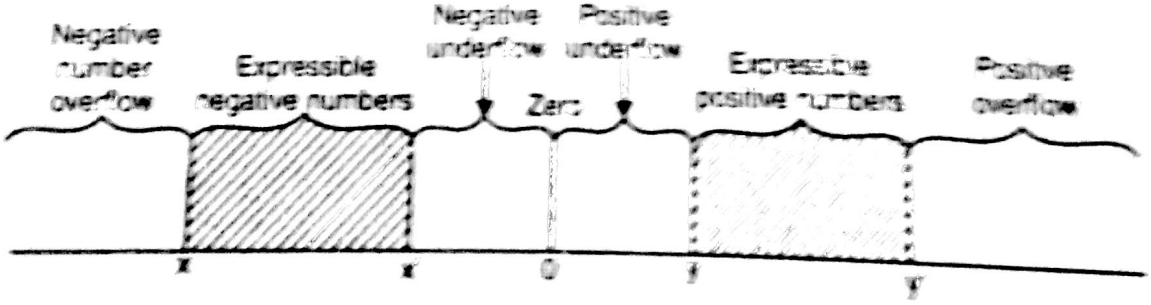



**Fig. 1.12 : Expressible numbers in floating point representation**

Fig. 1.12 shows the numbers that can be expressed (+ve or – ve), by shaded part. Below and above the shaded parts, numbers are not representable. Represent able range of numbers are x – x' and y – y'.

Five regions are not included in the group.

(1) Negative numbers, those are under negative overflow

(2) Negative numbers, those are under negative underflow

Scanned by CamScanner

| Significand (Mantissa) | Value |
|---|---|
| ≠ 0 | NAN |
| 0 | $(-1)^S \infty$ |
| – | $(-1)^S 2^{E-127} (1.M)$ |
| ≠ 0 | $(-1)^S 2^{-126} (0.M)$ |
| 0 | $(-1)^S 0$ |

**Double Precision (64 bits)**

| Significand (Mantissa) | Value |
|---|---|
| ≠ 0 | NAN |
| 0 | $(-1)^S \infty$ |
| – | $(-1)^S 2^{E-1023} (1.M)$ |
| ≠ 0 | $(-1)^S 2^{-1022} (0.M)$ |
| , 0 | $(-1)^S 0$ |

Scanned by CamScanner

$(1)_{10}$ = 0 0 . . 1 ⟶ Denormalised.

$(1)_{10}$ = $1.0 \times 2^0$ ⟶ Normalised.

∴ M = 0

E = E' – 127

0 = E' – 127 ∴ E' = 127 = $(0\ 1\ 1\ 1\ 1\ 1\ 1\ 1)_2$

| S | E' | Mantissa |
|---|---|---|
| 1 | 0 1 1 1 1 1 1 1 | 0 0 . . . . . 0 |
| 1 bit | 8 bits | 23 bits |

∴ Floating point number is $(1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ \ldots\ 0)_2$ and $(B\ F\ 8\ 0\ 0\ 0\ 0\ 0)_{16}$ i.e. hex.

**(iii)** $(0.5)_{10}$

$(0.5)_{10} = (0.1)_2 = 1.0 \times 2 - 2^{-1}$ ⟶ Normalised.

∴ As number is +ve S = 0

E = E' – 127

– 1 = E' – 127 ∴ E' = 126 = $(0\ 1\ 1\ 1\ 1\ 1\ 1\ 0)_2$

M = 0

| S | E' | |
|---|---|---|
| 0 | 0 1 1 1 1 1 1 0 | 0 0 . . . 0 |

Floating point Number is $(0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ \ldots\ 0)_2$

and $(3\ F\ 0\ 0\ 0\ 0\ 0\ 0)_{16}$ i.e. hex.

**(iv)** $(10)_{10}$

$(10)_{10} = (1\ 0\ 1\ 0)_2$ ⟶ De-normalised.

$1.0\ 1\ 0 \times 2^3$ ⟶ Normalised.

S = 0 ⟶ As number is +ve.

M = 0 1 0

E = E' – 127

3 = E' – 127 ∴ E' = 127 + 3 = $(130)_{10} = (1\ 0\ 0\ 0\ \ 0\ 0\ 1\ 0)_2$

| S | E' | Mantissa - M |
|---|---|---|
| 0 | 1 0 0 0 0 0 1 0 | 0 1 0 0 . . . 0 |

∴ Floating point number is $(0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ \ldots\ 0)_2$ and in hex 4 1 2 0 0 0 0 0 H.

**(v)** $(3.92 \times 10^2)_{10}$

= $(3920)_{10} = (1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0)_2$ ⟶ De-normalised.

∴ Normalised number ⟶ $1.1\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \times 2^8$

Here 5 bit = 0 ⟶ because number is +ve.

M = 1 0 0 0 0 0 0 0

E = E' – 127 = $(8)_{10}$ ∴ E' = $(135)_{10} = (1\ 0\ 0\ 0\ \ 0\ 1\ 1\ 1)_2$

| S | E | |
|---|---|---|
| 0 | 1 0 0 0 0 1 1 1 | 1 0 0 0 1 0 0 0 . . . . . 0 |

∴

Scanned by CamScanner

∴ Floating point number in binary is $(0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ \ldots\ 0)_2$ and in hex $43C40000H$.

## 1.9 Codes :

When numbers, letters or words are represented by a special group of symbols, we say that they are being encoded, and the group of symbols is called a ***Code***. Whenever you want that while communicating with your friend, nobody should know what you are communicating, you take help of ***Coded Language*** i.e. you speak in a code words such that only your friend can understand, nobody else will understand it. So **'coded'** message keeps your information secret.

Here you may ask, that in digital where the 'codes' are used. The answer is we are studying digital electronics, keeping in mind that in future we built microprocessor or computer. Now-a-days computers are used for scientific and business applications, or I say computers are used everywhere. Following are the points to use codes :

(1) To allow computer to perform fast scientific calculations like add, subtract, division, multiplication, log, antilog etc., at high speed and with more precision, one should use less HARDWARE. Therefore, we should have the codes which gives you all mathematical calculations with less number of steps.

(2) In computers (digital system) we use form of binary numbers for their internal operations, but the external world is decimal in nature. It means that conversion from binary to decimal and vice versa are performed often. We have seen that the conversions between decimal and binary can become long and complicated for large numbers. For this reason, a means of encoding decimal numbers that combines some features of both, the decimal and binary system is required.

(3) Computer also deals with Non-numerical Information like alphabet, punctuation marks and other special characters as well as numbers. To allow computer to handle all these non-numerical information, we have to have codes. The codes which represents non-numerical information are called ***alphanumeric Codes***.

(4) Presently I was talking about computers. User can have one or more number of computer of computers. If the user have two or more number of computers, he/she will insist that computers should ***Communicate*** with each other. The media for communication will be wires, wireless or fiber optics. The graphical presentation is shown in Fig. 1.13.

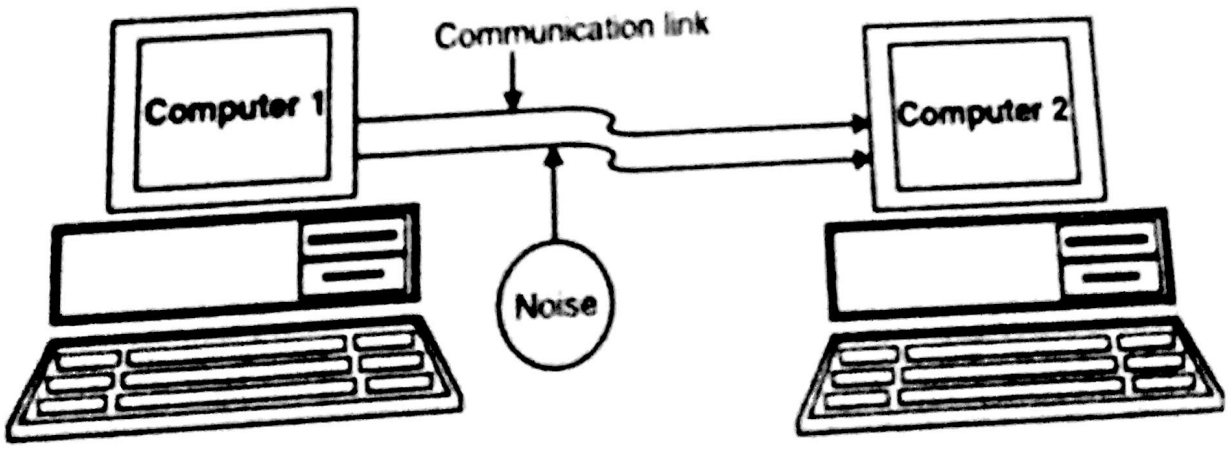



**Fig. 1.13 : Communication between computers**

Scanned by CamScanner

The problem involved in this is, when computers communicate with each other, the unwanted signal i.e. 'Noise' will come into picture and it may corrupt your data midway only. Because of this, corrupted data will be received by the receiver end of the computer. Therefore there has to be some means by which computer can ***detect error*** and if possible after detecting, can correct the problem. Therefore while performing communication between computers we use ***error detecting*** and ***error correcting codes***. Mainly we use parity method for error detecting and Hamming codes for detecting as well as correcting.

(5) Measurement Problem : Uptill now we have talked about computers. Digital can be also used to measure shaft's position without interfering with its motion. This is mainly required in industrial machine tools. The Fig. 1.14 will show you how digital method is implemented to measure angular position.

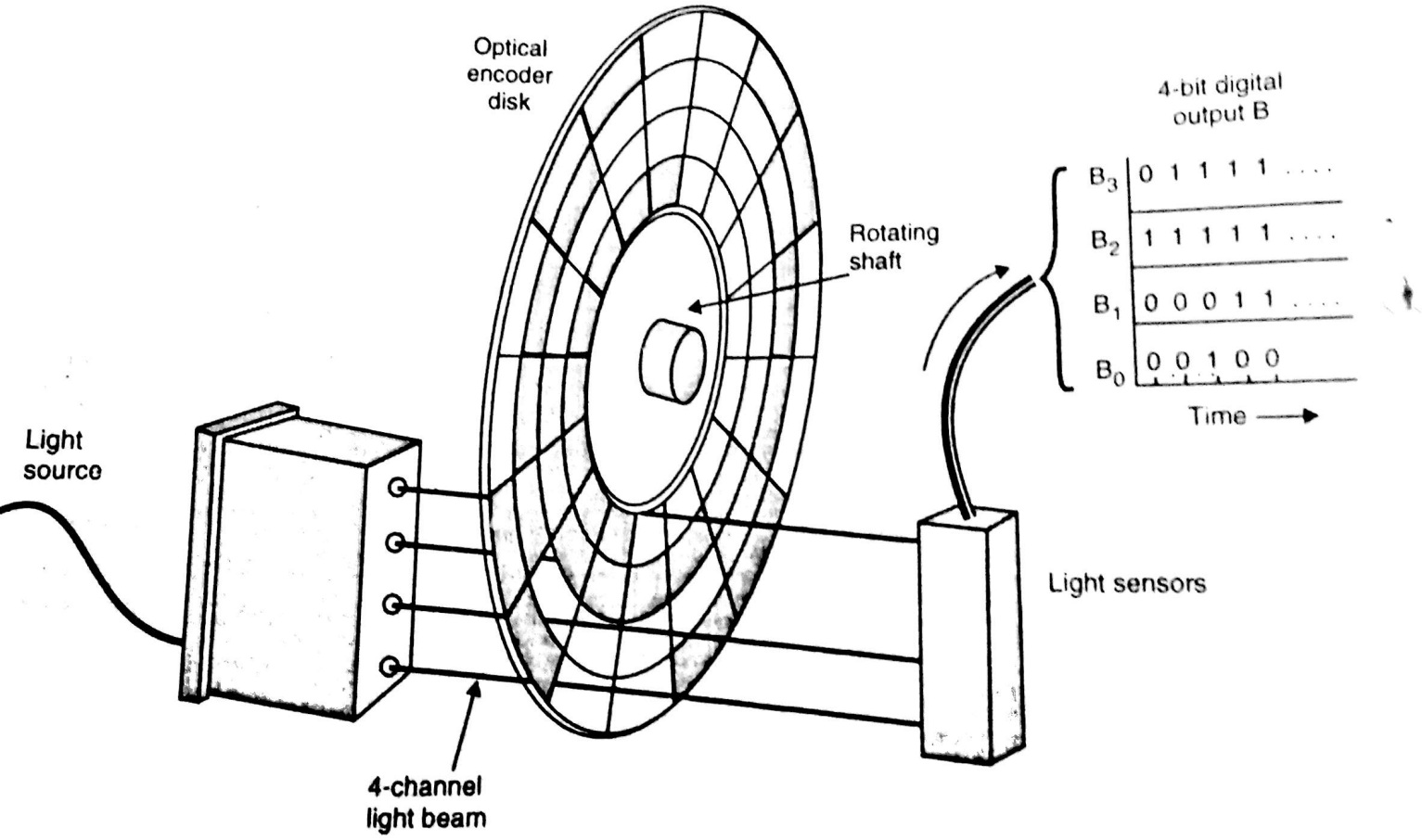



**Fig. 1.14 : Optical measurement of an angular position of rotating shaft**

As shown we are going to use optical technique for the above problem. A special disk, call an *Optical encoder*, through which a beam of light can be passed, is attached to rotating shaft. The light emerging from disk is sensed and processed by electronic circuits. If you observe figure carefully you will find that the encoder disk has translucent and opaque regions in patterns that vary with its position.

**Assumption :** When the light beam to a particular sensor is blocked, the output of the sensor will be logic 1. When light reaches to sensor, it will produce output logic zero

Scanned by CamScanner

$\therefore$ Sensor produces a pattern of binary signals $B_3\ B_2\ B_1\ B_0$.

The value of $B_3\ B_2\ B_1\ B_0$ represents current angular position of the disk. In short it provides 4 bit approximation of the angle.

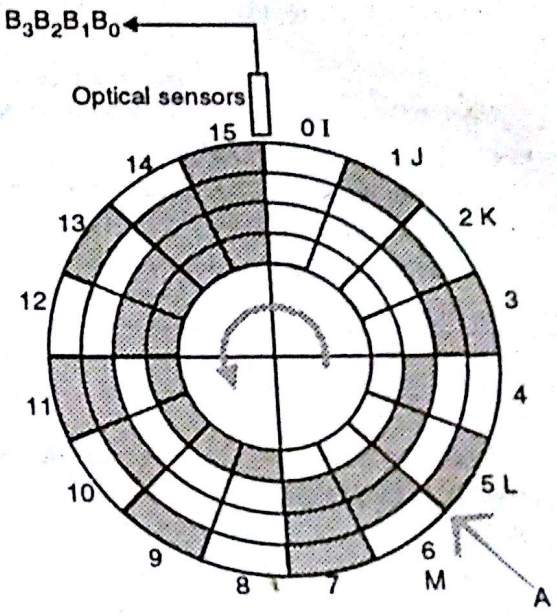



**Fig. 1.15 : Optical rotary encoders disk that uses ordinary binary code**

Fig. 1.15 shows optical encoder disk. The disk contain four concentric circle (tracks). Disk is divided in to 16 angular segments of equal size. ($22.5^0$). We have taken 16 division because 4 bit – $B_3\ B_2\ B_1\ B_0$ provides 16 binary states. If we want to increase accuracy, one can increase angular segments and can increase number of bits that will appear. If you observe figure, you will find that every segment has a unique pattern of translucent (0) and opaque (1) areas assigned to its four tracks. A 4 bit binary number is obtained by reading four tracks via light sensor. Let's consider that 'I' part is in front of light source. As all tracks of I are translucent you will get $B_3\ B_2\ B_1\ B_0 = (0\ 0\ 0\ 0)_2$. In segment 'J', outer track is dark, so you get $B_3\ B_2\ B_1\ B_0 = (0\ 0\ 0\ 1)_2$. The next 'K', segment is 0 0 1 0. Thus a complete revolution gives $B_3\ B_2\ B_1\ B_0$ = 0 0 0 0, 0 0 0 1, 0 0 1 0, 0 0 1 1 ... 1 1 1 1. After that again same chain will continue.

**Problem :** When shaft rotates and the ***boundary*** region of two segments passes optical sensor, some or all of the 4 bits $B_3\ B_2\ B_1\ B_0$ changes at slightly different times. The reasons are as follows :

***(1) Due to shaft wobble.***

***(2) Variations in response time of different light sensors.***

***(3) Other physical imperfection.***

**Note :** Two or more number of bits changing can cause problem, was specified, after Table 1, point 11. This is elaborated here.

Let's consider position A, which marks transition between segment L and M. Let's consider presently segment L is there. $\therefore\ B_3\ B_2\ B_1\ B_0 = (0\ 1\ 0\ 1)_2$. When A point rotates to sensor, actually $B_3\ B_2\ B_1\ B_0$ should change from $(0\ 1\ 0\ 1)_2$ to $(0\ 1\ 1\ 0)_2$.

But here if outer track perceived first, then $B_3\ B_2\ B_1\ B_0$ will change from.

$$(0\ 1\ 0\ 1)_2 \longrightarrow (0\ 1\ 0\ 0)_2 \longrightarrow (0\ 1\ 1\ 0)_2.$$

$\uparrow$

Outer track perceived first

Thus $(0\ 1\ 0\ 0)_2$ state, which is unwanted is introduced in the sequence. If optical sensor senses it, will produce error output.

Scanned by CamScanner

But if next track changes first, we have

$B_3\ B_2\ B_1\ B_0 = (0\ 1\ 0\ 1)_2 \longrightarrow (0\ 1\ 1\ 1)_2 \longrightarrow (0\ 1\ 1\ 0)_2$.

↑

next track perceived first.

Now it contains $(0\ 1\ 1\ 1)_2$ which is unwanted state. The most dangerous is when $B_3\ B_2\ B_1\ B$ changes from $(1\ 1\ 1\ 1)_2$ to $(0\ 0\ 0\ 0)_2$.

Therefore we should have such code which will not provide us spurious states in between. Here *Gra* ***Codes*** are used most widely.

We have covered five points to let you know the importance of the codes. The following chart wil inform you about different types of codes.

**CODES**

| Weighted Code | Non Weighted Code | Alphanumeric Code | Error detecting and Correcting Code |
|---|---|---|---|
| (A) BCD Codes | | | (1) Hamming Code. |
| (1) 8421 | (1) XS–3 | (1) ASCII | (2) Parity |
| (2) 7421 | (2) Gray Code | (2) EBCDIC | |
| (3) 5421 | | (3) Hollerith Code | |
| (4) 5211 | | | |
| (5) 4221 | | | |
| (6) 3321 | | | |
| (7) 2421 | | | |
| (8) $84\overline{21}$ | | | |
| (9) $74\overline{21}$ | | | |

Scanned by CamScanner

| | | |
|---|---|---|
| 11 | 0 0 0 1 | 0 0 0 1 |
| 12 | 0 0 0 1 | 0 0 1 0 |
| 13 | 0 0 0 1 | 0 0 1 1 |
| 14 | 0 0 0 1 | 0 1 0 0 |
| : | : | : |
| 19 | 0 0 0 1 | 1 0 0 1 |
| 20 | 0 0 1 0 | 0 0 0 0 |
| 21 | 0 0 1 0 | 0 0 0 1 |
| 22 | 0 0 1 0 | 0 0 1 0 |
| 23 | 0 0 1 0 | 0 0 1 1 |
| 24 | 0 0 1 0 | 0 0 1 1 |
| : | : | : |
| 29 | 0 0 1 0 | 1 0 0 1 |
| 30 | 0 0 1 1 | 0 0 0 0 |
| 31 | 0 0 1 1 | 0 0 0 1 |
| : | : | : |

Scanned by CamScanner

Say for an example if I want to represent $(8)_{10}$ then it will be represented as follows :

(a) 7 4 2 1 code

$(8)_{10} = 7 + 1 = 8$

1 0 0 1 ⟶ 7421 code.

$= 1 \times 7 + 4 \times 0 + 2 \times 0 + 1 \times 1 = 7 + 0 + 0 + 1 = 8.$

$(8)_{10} = (1\ 0\ 0\ 1)_{7421}$

(b) 5 4 2 1 Code

$(8)_{10} = 5 + 2 + 1$

∴ 5 4 2 1

1 0 1 1 ⟶ 5421 code.

$(8)_{10} = (1\ 0\ 1\ 1)_{5421}$

(c) 5211 Code.

$(8)_{10} = 5 + 2 + 1$

∴ (i) 5 2 1 1 (ii) 5 2 1 1

1 1 1 0 ← binary 5211 code → 1 1 0 1

∴ $(8)_{10} = (1\ 1\ 1\ 0)_{5211} = (1\ 1\ 0\ 1)_{5211}$ (both are valid).

But normally any code should have unique binary string for the value. Here we have two binary values so, *which one is correct*, this point I am going to elaborate within no time.

(d) 4 2 2 1

$(8)_{10} = 4 + 2 + 2$

∴ 4 2 2 1

1 1 1 0 ⟶ 4221 code.

$(8)_{10} = (1\ 1\ 1\ 0)_{4221}$

(e) 3 3 2 1

$(8)_{10} = 3 + 3 + 2 = 8$

∴ 3 3 2 1

1 1 1 0 ⟶ 3321 code

$(8)_{10} = (1\ 1\ 1\ 0)_{3321}$

(f) 2 4 2 1

$(8)_{10} = 2 + 4 + 2 = 8$

∴ 2 4 2 1

1 1 1 0 ⟶ 2421 code.

$(8)_{10} = (1\ 1\ 1\ 0)_{2421}$

(g) $8\ 4\ \bar{2}\ \bar{1}$

$(8)_{10} = 8.$

∴ $8\quad 4\quad \bar{2}\quad \bar{1}$

$1\quad 0\quad 0\quad 0 \longrightarrow 84\bar{2}\bar{1}$ code.

$(8)_{10} = (1\ 0\ 0\ 0)\ 8\ 4\ \bar{2}\ \bar{1}$

Scanned by CamScanner

| Decimal | 8 4 2 1 | 7 4 2 1 | 5 4 2 1 | 5 2 1 1 | 4 2 2 1 | 3 3 2 1 | 2 4 2 1 | 8 4 $\bar{2}$ $\bar{1}$ | 7 4 $\bar{2}$ $\bar{1}$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| 1 | 0 0 0 1 | 0 0 0 1 | 0 0 0 1 | 0 0 0 1 | 0 0 0 1 | 0 0 0 1 | 0 0 0 1 | 0 1 1 1 | 0 1 1 1 |
| 2 | 0 0 1 0 | 0 0 1 0 | 0 0 1 0 | 0 0 1 1 | 0 0 1 0 | 0 0 1 0 | 0 0 1 0 | 0 1 1 0 | 0 1 1 0 |
| 3 | 0 0 1 1 | 0 0 1 1 | 0 0 1 1 | 0 1 0 1 | 0 0 1 1 | 0 0 1 1 | 0 0 1 1 | 0 1 0 1 | 0 1 0 1 |
| 4 | 0 1 0 0 | 0 1 0 0 | 0 1 0 0 | 0 1 1 1 | 1 0 0 0 | 0 1 0 1 | 0 1 0 0 | 0 1 0 0 | 3 1 0 0 |
| 5 | 0 1 0 1 | 0 1 0 1 | 1 0 0 0 | 1 0 0 0 | 0 1 1 1 | 1 0 1 0 | 1 0 1 1 | 1 0 1 1 | 1 0 1 0 |
| 6 | 0 1 1 0 | 0 1 1 0 | 1 0 0 1 | 1 0 1 0 | 1 1 0 0 | 1 1 0 0 | 1 1 0 0 | 1 0 1 0 | 1 0 0 1 |
| 7 | 0 1 1 1 | 1 0 0 0 | 1 0 1 0 | 1 1 0 0 | 1 1 0 1 | 1 1 0 1 | 1 1 0 1 | 1 0 0 1 | 1 0 0 0 |
| 8 | 1 0 0 0 | 1 0 0 1 | 1 0 1 1 | 1 1 1 0 | 1 1 1 0 | 1 1 1 0 | 1 1 1 0 | 1 0 0 0 | 1 1 1 1 |
| 9 | 1 0 0 1 | 1 0 1 0 | 1 1 0 0 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 0 |

Scanned by CamScanner

Inverting the bits results in 1000 which is binary equivalent of $(5)_{10}$ where 5 is 9's complement of 4. To find 9's complement of number, you hav number from 9 and resulting number is 9's complement. So in 5211 you ac by changing 1 to 0 and vice versa.

Under this section I would like to also elaborate one more point which I

Under 5211 code we got 2 codes for $(8)_{10}$ i.e. 1110 and 1101. I told yo want specifically ***Reflection characteristic*** in 5211 then 1110 will give co will give code for $(1)_{10}$ = 0010. So that change has to be correctly done in th

We can also write down 2421 code as shown in Table 7.

| Decimal | 2 4 2 1 (a) | 2 4 2 1 (b) |
|---|---|---|
| 0 | 0 0 0 0 | 0 0 0 0 |
| 1 | 0 0 0 1 | 0 0 0 1 |
| 2 | 0 0 1 0 | 0 0 1 0 |
| 3 | 0 0 1 1 | 0 0 1 1 |
| 4 | 0 1 0 0 | 0 1 0 0 |
| 5 | 1 0 1 1 | 0 1 0 1 |
| 6 | 1 1 0 0 | 0 1 1 0 |
| 7 | 1 1 0 1 | 0 1 1 1 |
| 8 | 1 1 1 0 | 1 1 1 0 |
| 9 | 1 1 1 1 | 1 1 1 1 |

**Table 7**

As shown in Table 7, 2421 (a) is same as that in Table 6, i.e. self con written 2421 (b) which doesn't cater for reflecting characteristic. If you correct. So it is upto user which code he should use (a) or (b).

**Representing Number greater than 9 in BCD codes :**

Uptill now we have seen representation of 0 to 9 decimal in 2421, 5 decimal number greater than 9 ? how to represent it ? Let's take an example

Scanned by CamScanner

| Decimal | | | | | XS-3 Code $2^3$ | $2^2$ | $2^1$ | $2^0$ | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | 0 | 0 | 1 | 1 | 0 + 3 = 3 |
| 1 | | | | | 0 | 1 | 0 | 0 | 1 + 3 = 4 |
| 2 | | | | | 0 | 1 | 0 | 1 | 2 + 3 = 5 |
| 3 | | | | | 0 | 1 | 1 | 0 | 3 + 3 = 6 |
| 4 | | | | | 0 | 1 | 1 | 1 | 4 + 3 = 7 |
| 5 | | | | | 1 | 0 | 0 | 0 | 5 + 3 = 8 |
| 6 | | | | | 1 | 0 | 0 | 1 | 6 + 3 = 9 |
| 7 | | | | | 1 | 0 | 1 | 0 | 7 + 3 = 10 |
| 8 | | | | | 1 | 0 | 1 | 1 | 8 + 3 = 11 |
| 9 | | | | | 1 | 1 | 0 | 0 | 9 + 3 = 12 |
| 10 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | (1 + 3)(0 + 3) = 4/3 |
| 11 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | (1 + 3)(1 + 3) = 4/4 |
| 12 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | (1 + 3)(2 + 3) = 4/5 |
| 13 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | (1 + 3)(3 + 3) = 4/6 |
| . | | | . | | | | . | | . |
| . | | | . | | | | . | | . |
| . | | | . | | | | . | | . |
| 19 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | (1 + 3)(9 + 3) = 4/12 |
| 20 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | (2 + 3)(0 + 3) = 5/3 |
| 21 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | |
| . | | | . | | | | . | | . |
| . | | | . | | | | . | | . |
| . | | | . | | | | . | | . |
| 29 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | |

Scanned by CamScanner

$(4)_{10}$ in XS–3 $\longrightarrow$ (0 1 1 1)

Change 0 to 1 and 1 to 0, $\therefore$ we get (1 0 0 0) from (0 1 1 1).

1 0 0 0 in XS–3 is $(5)_1$ (Refer Table 8)

9's complement of 4 is = $(9)_{10} - (4)_{10} = (5)_{10}$

$\therefore$ XS–3 is self complementing code, because complement of XS–3 number represents 9's complement of decimal number.

### 1.11.2 Gray Code :

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

This Code I am going to explain you with reference to point (5) i.e. measurement problem under section 1.9. In that I have explained you about problem Involved in measuring angular position of shaft. We also concluded that we require one code in which we don't get ***Spurious*** (or unwanted) state. The answer to this problem is gray code. Let's see what is so special about gray.

In gray code ***every NEW CODE DIFFERS FROM THE PREVIOUS IN TERMS OF ONE SINGLE BIT***. In short at a time only ***one*** bit changes. We saw in binary system that 2 or more bits changes at a time and it creates problem. To understand advantage of change in only one bit first see that Table 9 which lists out gray code for equivalent binary.

| Decimal | Binary | Gray |
|---|---|---|
| 0 | 0 0 0 0 | 0 0 0 0 |
| 1 | 0 0 0 1 | 0 0 0 1 |
| 2 | 0 0 1 0 | 0 0 1 1 |
| 3 | 0 0 1 1 | 0 0 1 0 |
| 4 | 0 1 0 0 | 0 1 1 0 |
| 5 | 0 1 0 1 | 0 1 1 1 |
| 6 | 0 1 1 0 | 0 1 0 1 |
| 7 | 0 1 1 1 | 0 1 0 0 |
| 8 | 1 0 0 0 | 1 1 0 0 |
| 9 | 1 0 0 1 | 1 1 0 1 |
| 10 | 1 0 1 0 | 1 1 1 1 |
| 11 | 1 0 1 1 | 1 1 1 0 |
| 12 | 1 1 0 0 | 1 0 1 0 |
| 13 | 1 1 0 1 | 1 0 1 1 |
| 14 | 1 1 1 0 | 1 0 0 1 |
| 15 | 1 1 1 1 | 1 0 0 0 |

**Table 9**

Let's take 4 and 5, $G_3$ $G_2$ $G_1$ $G_0$.

$(4)_{10}$ $\longrightarrow$ gray code is 0 1 1 0

$(5)_{10}$ $\longrightarrow$ gray code is 0 1 1 1

If you observe $G_3$ $G_2$ $G_1$ for 4 and 5 are same , only $G_0$ changes.

Same way for 7 and 8,

Scanned by CamScanner

| | | $G_3$ | $G_2$ | $G_1$ | $G_0$ |
|---|---|---|---|---|---|
| $(7)_{10} \longrightarrow$ | gray code | 0 | 1 | 0 | 0 |
| $(8)_{10} \longrightarrow$ | gray code | 1 | 1 | 0 | 0 |

$G_0$, $G_1$, $G_2$ are same only $G_3$ changes.

So if you take any two consequent numbers, you will find that while switching over to next number only one bit changes at a time.

To measure angular position of shaft we use rotary optical disk. Disk is divided into 16 angular segments of equal size. Every segment has unique pattern of translucent (0) and opaque (1) areas assigned to its four tracks. The pattern is shown in Fig. 1.16.

The optical encoder has a pattern, which provides gray codes. Let's say $G_3\ G_2\ G_1\ G_0 = (5)_{10} = 0111$ (gray) i.e. we are presently at segment L. The point A is between L and M. When A point rotates to sensor, we should get $G_3\ G_2\ G_1\ G_0 = (6)_{10} = 01\ 01$ (gray). As we are using gray, even though shaft wobble or physical imperfection is there, we will get correct code.

Let's take a case that outer track perceived first. ∴ Initial condition is $(5)_{10} = 0111$ ($G_3\ G_2\ G_1\ G_0$) and as outer track perceived first we will get next code as 0111 ($G_3\ G_2\ G_1\ G_0$) only because, outer track is opaque for $(6)_{10}$ and $G_3\ G_2\ G_1$ are unchanged ; ∴ the code remains same. Same way if next track perceived first we will get $G_3\ G_2\ G_1\ G_0 = 0101 = (6)_{10}$ and this is correct code. So if we observe, when $G_3\ G_2\ G_1\ G_0$ code changes it is delayed slightly by imperfection, but this imperfection will not give us unwanted code.

Gray codes are also most widely used in K-map.

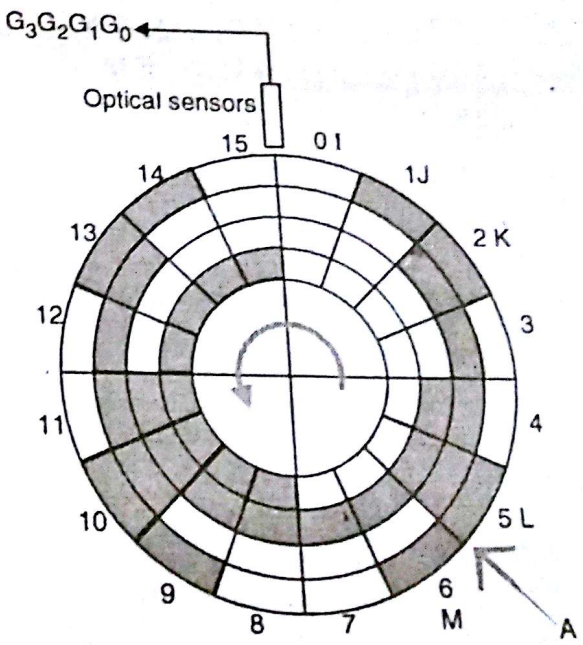



**Fig. 1.16 : A rotary optical encoder disk that uses, gray code**

## 1.12 Alphanumeric Codes :

To represent alphabet letters. Punctuation mark and other special character in computer system we require codes. These codes are called ***alphanumeric codes***.

Two most popular alphanumeric codes normally used are :

(i) American standard code for Information Interchange (ASCII).

(ii) Extended Binary Coded Decimal Interchange Code (EBCDIC).

### 1.12.1 ASCII Code (Pronounced as – key) :

This is an accepted standard character encoding scheme introduced in 1963 and used widely in many computer. It is basically 7 bit code and thus provides $2^7 = 128$ bit patterns. The complete ASCII code chart is illustrated in Table 10.

Scanned by CamScanner

| | | | |
|---|---|---|---|
| 6 | F | V | f |
| 7 | G | W | g |
| 8 | H | X | h |
| 9 | I | Y | i |
| : | J | Z | j |
| ; | K | [ | k |
| < | L | \ | l |
| = | M | ] | m |
| > | N | | n |
| ? | O | – | o |

CR – Carriage return

NAK – Negative acknowledge

Scanned by CamScanner

The ASCII code is represented by $b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1$. Take one example if we want to find A S C I I for 'A'.

'A' ⟶ $b_7\ b_6\ b_5$ $b_4\ b_3\ b_2\ b_1$

1 0 0 (from column) 0 0 0 1 (from Row)

∴ 'A' ⟶ ASCII $(1\ 0\ 0\ 0\ 0\ 0\ 1)_2$ OR $(41)_{16}$ hex.

Same way ASCII for LF (fine feed)

LF ⟶ $b_7\ b_6\ b_5$ $b_4\ b_3\ b_2\ b_1$

0 0 0 (from column) 1 0 1 0 (from Row)

∴ L . F ⟶ ASCII $(0\ 0\ 0\ 1\ 0\ 1\ 0)_2$ OR $(OA)_{16}$ hex.

## 1.12.2 EBCDIC (Pronounced *ebb – See – dick*) :

This character code was developed by IBM, which stands for Extended Binary Coded Decimal Interchange Code. EBCDIC employs all 8 bits of a byte, and so encoded a 256 character set, which is broadly similar to ASCII. Table 11 illustrates 8 bit EBCDIC code.

**Extended Binary Coded Decimal Interchange Code (EBCDIC)**

**Most Significant Digit (Hex)**

| LSD (Hex) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | DLE | DS | | SP | & | | | | | | | { | } | \ | 0 |
| 1 | SOH | DC1 | SOS | | | | / | | a | j | | | A | J | | 1 |
| 2 | STX | DC 2 | FS | SYN | | | | | b | k | s | | B | K | S | 2 |
| 3 | ETX | DC3 | , | | | | | | c | l | t | | C | L | T | 3 |
| 4 | PF | RES | BYP | PN | | | | | d | m | u | | D | M | U | 4 |
| 5 | HT | NL | LF | RS | | | | | e | n | v | | E | N | V | 5 |
| 6 | LC | BS | EOB | UC | | | | | f | o | w | | F | O | W | 6 |
| 7 | DEL | IL | PRE | EOT | | | | | g | p | x | | G | P | X | 7 |
| 8 | | CAN | | | | | | | h | q | y | | H | Q | Y | 8 |
| 9 | | EM | | | | | | | i | r | z | | I | R | Z | 9 |
| A | SMM | CC | SM | | ¢ | ! | \| | : | | | | | | | |
| B | VT | | | | . | $ | , | # | | | | | | | | |
| C | FF | IFS | | DC4 | < | . | % | @ | | | | | | | | |
| D | CR | IGS | ENQ | NAK | ( | ) | – | ' | | | | | | | | |
| E | SO | IRS | ACK | | + | ; | > | = | | | | | | | | |
| F | SI | IUS | BEL | SUB | \| | – | ? | " | | | | | | | | |

**Table 11**

Scanned by CamScanner

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| STX | DC2 | ' | २ | ए | ड (ड) | ड (ड) | ब्र (द) | | |
| ETX | DC3 | | ३ | क | ढ | ड़ (ड) | झ | | |
| EOT | DC4 | ॐ | ४ | | ढ | ड़ (ड) | ञ | | |
| ENQ | NAK | S | ५ | क (क) | ण (ण्ड) | ढ़ (ड) | ट्र (द) | | |
| ACK | SYN | : | ६ | क़ (क) | ण | ढ (ढ) | ६ | | |
| BEL | ETB | , | ७ | क्र (क) | ऩ | (ढ) | ६ | | |
| BS | CAN | ( | ८ | क्त (क) | ऋ | ण | ८ | | |
| HT | EM | ) | ९ | क्ष | | त | | | |
| LF | SUB | ! | : | ख | | | | | |
| VT | ESC | अ | ; | ख़ | | त्र | ८ | | |
| FF | FS | , | इ | ख्र | ऱ | | ९ | | |
| CR | GS | - | उ | ग | ट (ट) | | फ | | |
| SO | RS | . | ऊ (ऊ) | ग़ | ट्ट (ट) | द (द) | फ़ | | |
| SI | US | / | ? | | ड्ड (द) | (द) | DEL | | |

Scanned by CamScanner

| | | |
|---|---|---|
| NULL | – | Null |
| PE | – | Punch off |
| HT | – | Horizontal tab |
| LC | – | Lower case |
| DEL | – | Delete |
| RES | – | Restore |
| NL | – | New line |
| BS | – | Backspace |
| IL | – | Idle |
| BYP | – | Bypass |
| LF | – | Line feed |
| EOB | – | End of block |
| PRE | – | Prefix |
| SM | – | Set mode |
| PN | – | Punch on |
| RS | – | Reader stop |
| UC | – | Upper case |
| EOT | – | End of transmission |
| SP | – | Space. |

### 1.12.4 Hollerith Code :

Presently for storing the data / Information we use floppy, hard disk etc., for permanent storage. But before these devices, for storing the data *Punch Card* was used.

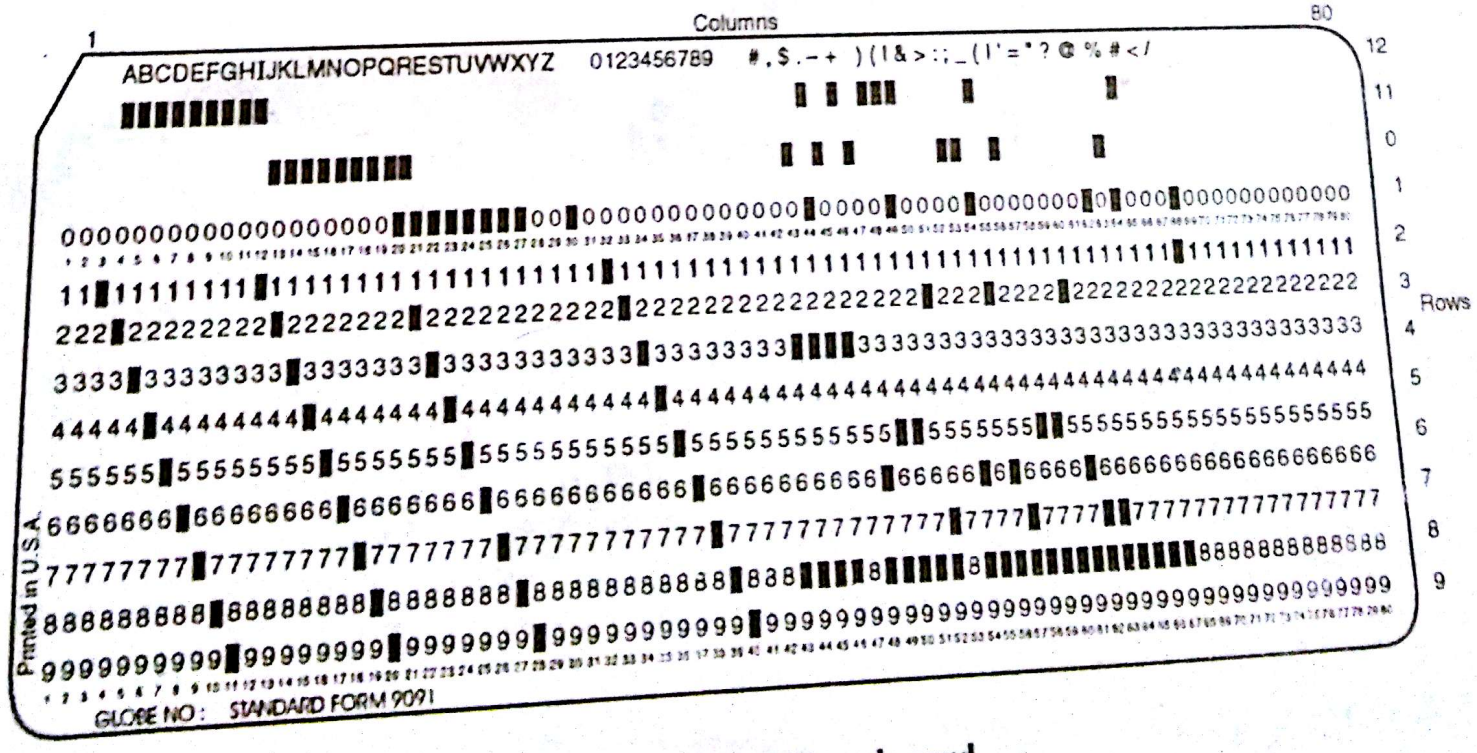



**Fig. 1.17 : Punch card**

Scanned by CamScanner

| International | American |
|---|---|
| . — | . — |
| — . . . | — . . . |
| — . — . | . . — . |
| — . . | — . . |
| . | . |
| . . — . | . — . |
| — — . | — — . |
| . . . . | . . . . |
| . . | . . |
| . — — — | — . — . |
| — . — | — . — |
| . — . . | ——— |
| — — | — — |
| — . | — . |
| — — — | . . |
| . — — . | . . . . . |
| — — . — | . . — . |
| . — . | . . . |
| . . . | . . . |
| — | — |
| . . — | . . — |
| . . . — | . . . — |
| . — — | . — — |
| — . . — | . — . . |
| — . — — | . . . . |
| — — . . | . . . . |

Table 13

Scanned by CamScanner

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | - | o | ● | ● | o | o | o | ● | P | 0 | o | ● | ● | o | o | ● | ● |
| B | ? | o | ● | o | o | ● | ● | ● | Q | 1 | o | ● | ● | ● | o | ● | ● |
| C | : | o | o | ● | ● | ● | o | ● | R | 4 | o | o | ● | o | ● | o | ● |
| D | WHO ARE YOU | o | ● | o | o | ● | o | ● | S | ' | o | ● | o | ● | o | o | ● |
| E | 3 | o | ● | o | o | o | o | ● | T | 5 | o | o | o | o | o | ● | ● |
| F | % | o | ● | o | ● | ● | o | ● | U | 7 | o | ● | ● | ● | o | o | ● |
| G | @ | o | o | ● | o | ● | ● | ● | V | = | o | o | ● | ● | ● | ● | ● |
| H | £ | o | o | o | ● | o | ● | ● | W | 2 | o | ● | ● | o | o | ● | ● |
| I | 8 | o | o | ● | ● | o | o | ● | X | / | o | ● | o | ● | ● | ● | ● |
| J | BELL | o | ● | ● | o | ● | o | ● | Y | 6 | o | ● | o | ● | o | ● | ● |
| K | ( | o | ● | ● | ● | ● | o | ● | Z | + | o | ● | o | o | o | ● | ● |
| L | ) | o | o | ● | o | o | ● | ● | CARRIAGE RETURN | | o | o | o | o | ● | o | ● |
| M | . | o | o | o | ● | ● | ● | ● | FIGURES | | o | ● | ● | o | ● | ● | ● |
| N | , | o | o | o | ● | ● | o | ● | LETTERS | | o | ● | ● | ● | ● | ● | ● |
| O | 9 | o | o | o | o | ● | ● | ● | LINE FEED | | o | o | ● | o | o | o | ● |

SPACE o o o ● o o ●

o START POLARITY (SPACING)

● STOP POLARITY (MARKING)

**Table 14 : Baudot code table**

Scanned by CamScanner

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

## 1.13 Error Detecting and Correcting Code :

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

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

(1) Communication between Computers.

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

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

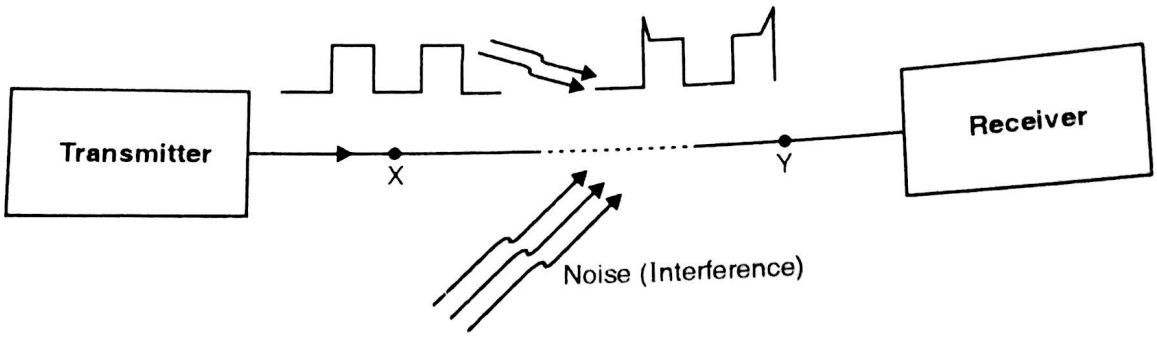



**Fig. 1.18 : Transmission data corrupted by noise**

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

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

### 1.13.1 Parity Bit :

➥ [ASKED IN EXAM - DEC. 98 III ]

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

(i) Even.

(ii) Odd.

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

Scanned by CamScanner

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

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

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

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

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

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

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

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

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

---

**Ex. 51 :** Attach even parity bit to following ASCII codes :

(1) 'H' 1 0 0 1 0 0 0

(2) 'L' 1 0 0 1 1 0 0

(3) 'O' 1 0 0 1 1 1 1

(4) 'A' 1 0 0 0 0 0 1

(5) 'a' 1 1 0 0 0 0 1

**Soln. :**

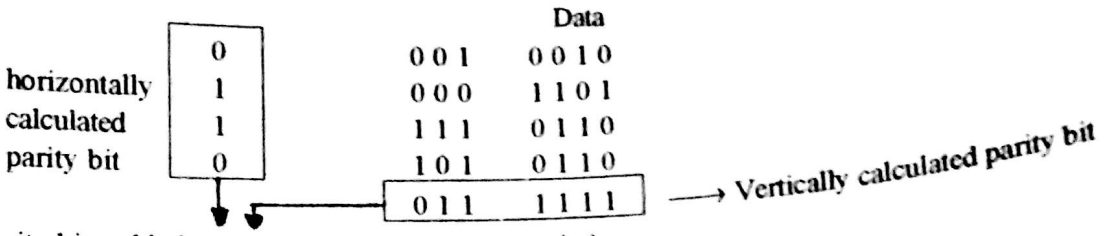
| ASCII | Parity bit | Code |
|---|---|---|
| 'H' | 0 | 1 0 0 1 0 0 0 |
| 'L' | 1 | 1 0 0 1 1 0 0 |
| 'O' | 1 | 1 0 0 1 1 1 1 |
| 'A' | 0 | 1 0 0 0 0 0 1 |
| 'a' | 1 | 1 1 0 0 0 0 1 |

To set even parity

## 1.13.2 Double Parity :

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

Scanned by CamScanner




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

Here Information is passed in blocks.

### 1.13.3 Checksum :

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

$A_1$ ⟶ data byte/ word

\+ $A_2$

\+ $A_3$ (Total 'n' number of data bytes / words)

\+ '

'

'

\+ $A_n$

Checksum

For example take $A_1$ and $A_2$ bytes

| | | | |
|---|---|---|---|
| $A_1$ | 0 1 0 0 | 1 0 0 0 |
| + $A_2$ | 0 0 1 1 | 1 1 0 0 |
| Checksum | 1 0 0 0 | 0 1 0 0 |

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

### 1.13.4 Five Bit Codes :

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

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

| Decimal | 8 6 4 2 1 | 6 3 2 1 0 | Shift Counter | 5 1 1 1 1 |
|---|---|---|---|---|
| 0 | 0 0 0 0 0 | 0 0 1 1 0 | 0 0 0 0 0 | 0 0 0 0 0 |
| 1 | 0 0 0 0 1 | 0 0 0 1 1 | 0 0 0 0 1 | 0 0 0 0 1 |
| 2 | 0 0 0 1 0 | 0 0 1 0 1 | 0 0 0 1 1 | 0 0 0 1 1 |
| 3 | 0 0 0 1 1 | 0 1 0 0 1 | 0 0 1 1 1 | 0 0 1 1 1 |
| 4 | 0 0 1 0 0 | 0 1 0 1 0 | 0 1 1 1 1 | 0 1 1 1 1 |
| 5 | 0 0 1 0 1 | 0 1 1 0 0 | 1 1 1 1 1 | 1 0 0 0 0 |
| 6 | 0 1 0 0 0 | 1 0 0 0 1 | 1 1 1 1 0 | 1 1 0 0 0 |
| 7 | 0 1 0 0 1 | 1 0 0 1 0 | 1 1 1 0 0 | 1 1 1 0 0 |
| 8 | 1 0 0 0 0 | 1 0 1 0 0 | 1 1 0 0 0 | 1 1 1 1 0 |
| 9 | 1 0 0 0 1 | 1 1 0 0 0 | 1 0 0 0 0 | 1 1 1 1 1 |

**Table 15**

Scanned by CamScanner

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

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

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

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

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

| Decimal | Ring Counter Code 9 8 7 6 5 4 3 2 1 0 | 5 4 3 2 1 0 | 50 – 4 3 2 1 0 | 4 out of 5 code 7 4 2 1 0 |
|---|---|---|---|---|
| 0 | 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 1 | 01 – 0 0 0 0 1 | 1 1 0 0 0 |
| 1 | 0 0 0 0 0 0 0 0 1 0 | 0 0 0 0 1 0 | 01 – 0 0 0 1 0 | 0 0 0 1 1 |
| 2 | 0 0 0 0 0 0 0 1 0 0 | 0 0 0 1 0 0 | 01 – 0 0 1 0 0 | 0 0 1 0 1 |
| 3 | 0 0 0 0 0 0 1 0 0 0 | 0 0 1 0 0 0 | 01 – 0 1 0 0 0 | 0 0 1 1 0 |
| 4 | 0 0 0 0 0 1 0 0 0 0 | 0 1 0 0 0 0 | 01 – 1 0 0 0 0 | 0 1 0 0 1 |
| 5 | 0 0 0 0 1 0 0 0 0 0 | 1 0 0 0 0 1 | 10 – 0 0 0 0 1 | 0 1 0 1 0 |
| 6 | 0 0 0 1 0 0 0 0 0 0 | 1 0 0 0 1 0 | 10 – 0 0 0 1 0 | 0 1 1 0 0 |
| 7 | 0 0 1 0 0 0 0 0 0 0 | 1 0 0 1 0 0 | 10 – 0 0 1 0 0 | 1 0 0 0 1 |
| 8 | 0 1 0 0 0 0 0 0 0 0 | 1 0 1 0 0 0 | 10 – 0 1 0 0 0 | 1 0 0 1 0 |
| 9 | 1 0 0 0 0 0 0 0 0 0 | 1 1 0 0 0 0 | 10 – 1 0 0 0 0 | 1 0 1 0 0 |

**Table 16**

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

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

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

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

### 1.13.6 Error Correcting Codes :

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

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

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

Scanned by CamScanner

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

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

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

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

m = n – k = Number of Parity bits.

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

**Conditions for parity bit :**

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

$P_1$ ⟶ 1, 3, 5, 7.

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

$P_2$ ⟶ 2, 3, 6, 7.

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

$P_4$ ⟶ 4, 5, 6, 7.

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

| 7 bit hamming format ⟶ | $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ |
|---|---|---|---|---|---|---|---|
| | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| | $B_3$ | $B_2$ | $B_1$ | $P_4$ | $B_0$ | $P_2$ | $P_1$ |
| | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| | 1 | 0 | 1 | $P_4$ | 1 | $P_2$ | $P_1$ |

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

*For $P_1$ bit*

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

*For $P_2$ bit*

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

*For $P_4$ bit*

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

∴ Finally we have 7 bit hamming code.

**Ans. :**

| $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |

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

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

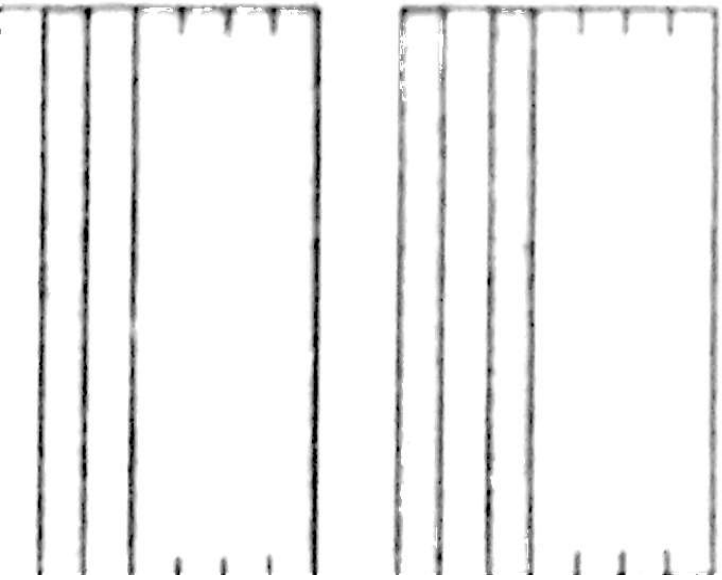
*Hamming Code Format*

| $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 | 1 |

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

Scanned by CamScanner

lly based on optical technique. Let's refer Fig. 1.19 to understand




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

Scanned by CamScanner

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

Scanned by CamScanner

## 1.15.2 Binary to BCD :

Assume 4 bit binary. Table 19 is as follows :

| Decimal | Binary $B_3$ | $B_2$ | $B_1$ | $B_0$ | 8 | 4 | 2 | 1 | BCD |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | | | | 0 0 0 0 |
| 1 | 0 | 0 | 0 | 1 | | | | | 0 0 0 1 |
| 2 | 0 | 0 | 1 | 0 | | | | | 0 0 1 0 |
| 3 | 0 | 0 | 1 | 1 | | | | | 0 0 1 1 |
| 4 | 0 | 1 | 0 | 0 | | | | | 0 1 0 0 |
| 5 | 0 | 1 | 0 | 1 | | | | | 0 1 0 1 |
| 6 | 0 | 1 | 1 | 0 | | | | | 0 1 1 0 |
| 7 | 0 | 1 | 1 | 1 | | | | | 0 1 1 1 |
| 8 | 1 | 0 | 0 | 0 | | | | | 1 0 0 0 |
| 9 | 1 | 0 | 0 | 1 | | | | | 1 0 0 1 |
| 10 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 0 0 0 |
| 11 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 0 0 1 |
| 12 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 0 1 0 |
| 13 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 0 1 1 |
| 14 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 1 0 0 |
| 15 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 1 0 1 |

Table 19

## 1.15.3 BCD to Binary :

It is exactly opposite to BCD code conversion as illustrated in Table 20.

| Decimal | 8 | 4 | 2 | 1 | BCD | Binary $B_3$ | $B_2$ | $B_1$ | $B_0$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | 0 0 0 0 | 0 | 0 | 0 | 0 |
| 1 | | | | | 0 0 0 1 | 0 | 0 | 0 | 1 |
| 2 | | | | | 0 0 1 0 | 0 | 0 | 1 | 0 |
| 3 | | | | | 0 0 1 1 | 0 | 0 | 1 | 1 |
| 4 | | | | | 0 1 0 0 | 0 | 1 | 0 | 0 |
| 5 | | | | | 0 1 0 1 | 0 | 1 | 0 | 1 |
| 6 | | | | | 0 1 1 0 | 0 | 1 | 1 | 0 |
| 7 | | | | | 0 1 1 1 | 0 | 1 | 1 | 1 |
| 8 | | | | | 1 0 0 0 | 1 | 0 | 0 | 0 |
| 9 | | | | | 1 0 0 1 | 1 | 0 | 0 | 1 |
| 10 | 0 | 0 | 0 | 1 | 0 0 0 0 | 1 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 1 | 0 0 0 1 | 1 | 0 | 1 | 1 |
| 12 | 0 | 0 | 0 | 1 | 0 0 1 0 | 1 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 1 | 0 0 1 1 | 1 | 1 | 0 | 1 |
| 14 | 0 | 0 | 0 | 1 | 0 1 0 0 | 1 | 1 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 | 0 1 0 1 | 1 | 1 | 1 | 1 |

Table 20

Scanned by CamScanner

## ess – 3) :

by adding three to original number. T

| Decimal | B C D | XS – 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 |

**Table 21**

CD code.

## o BCD :

to XS–3 table. The sam is illustrat

| Decimal | XS – 3 | B C D |
|---|---|---|
| 0 | 0 0 1 1 | 0 0 0 0 |
| 1 | 0 1 0 0 | 0 0 0 1 |
| 2 | 0 1 0 1 | 0 0 1 0 |
| 3 | 0 1 1 0 | 0 0 1 1 |
| 4 | 0 1 1 1 | 0 1 0 0 |
| 5 | 1 0 0 0 | 0 1 0 1 |
| 6 | 1 0 0 1 | 0 1 1 0 |
| 7 | 1 0 1 0 | 0 1 1 1 |
| 8 | 1 0 1 1 | 1 0 0 0 |
| 9 | 1 1 0 0 | 1 0 0 1 |

**Table 22**

are as follows :

Scanned by CamScanner

The EX–OR function is shown by symbol '⊕'.

i.e. $Y = A \oplus B$

Output, 1st bit, EX–OR, 2nd bit

| Inputs | | Output |
|---|---|---|
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

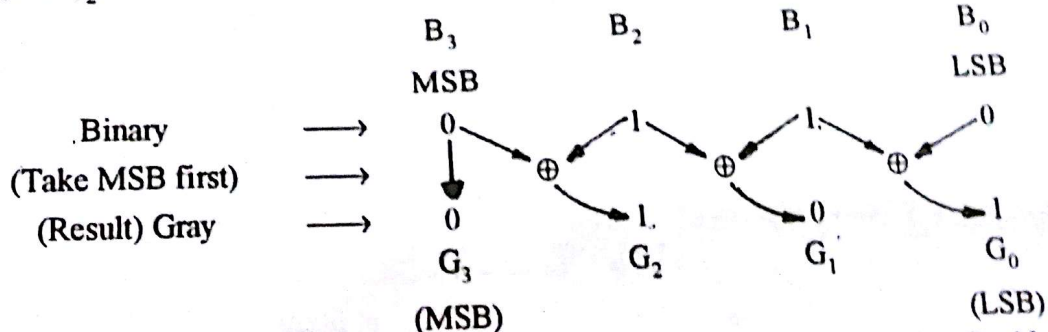
**Table 23**

First we will see EX–OR method to convert given binary number to equivalent gray.

**(a) By EX–ORing Method :**

Consider decimal 6 The equivalent binary is $(0110)_2$.

$(6)_{10} = (0110)_2$.




As shown above first you take MSB, so you get '0' at MSB position of gray i.e. $G_3$. Now start from MSB of binary and take next immediate bit i.e. 1. ∴ (MSB) ⊕ (next immediate bit) = 0 ⊕ 1 = 1 (Refer Table 23) = $G_2$.

Same way you take on next bits, i.e. $1 \oplus 1 = 0 = G_1$ and finally $1 \oplus 0 = 1 \rightarrow$ (LSB) = $G_0$.

∴ (0110) binary is equal to (0101) gray.

**(b) By Addition Method :**

Before we start for this you should know 'rules' of binary addition. At this stage I am going to give one table, you take it as granted.

Here we will take addition of single bit of two number, namely $A_0$ and $B_0$. Output is *Sum* and *Carry*.

| Inputs | | Outputs | |
|---|---|---|---|
| $A_0$ | $B_0$ | Sum | Carry |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

**Table 24**

Scanned by CamScanner

| | | |
|---|---|---|
| 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 |

Scanned by CamScanner

5421 – gray code same way you have 7421 – gray, 5211 - gray, XS–3 gray etc. The method for conversion is same what we have studied just now i.e. EX–OR / addition method. Following is Table which will show 2421 converted to equivalent 2421 – gray code.

| 2 4 2 1 | 2 4 2 1 – gray |
|---|---|
| 0 0 0 0 | 0 0 0 0 |
| 0 0 0 1 | 0 0 0 1 |
| 0 0 1 0 | 0 0 1 1 |
| 0 0 1 1 | 0 0 1 0 |
| 0 1 0 0 | 0 1 1 0 |
| 1 0 1 1 | 1 1 1 0 |
| 1 1 0 0 | 1 0 1 0 |
| 1 1 0 1 | 1 0 1 1 |
| 1 1 1 0 | 1 0 0 1 |
| 1 1 1 1 | 1 0 0 0 |

Table 26

743

## 1.15.7 Gray to Binary :

A gray number can be converted to binary step by step as follows :

**Step I** : MSB bit is kept as it is.

**Step II** : This bit is EX–ORed (added) with next bit from gray code.

**Step III** : The resulting bit in step II is EX–ORed with next bit from gray code.

**Step IV** : Step III is repeated till you reach LSB.

We will talk in terms of $B_3\ B_2\ B_1\ B_0$ and $G_3\ G_2\ G_1\ G_0$. Let's say given gray code is $G_3\ G_2\ G_1\ G$

(Refer Table for EX–OR Table 23).

(a) $B_3$ (binary MSB) = $G_3$ (MSB as it is)

(b) $B_3$ bit should be EX–ORed with next of gray, means $B_3 \oplus G_2$, this will give $B_2$.

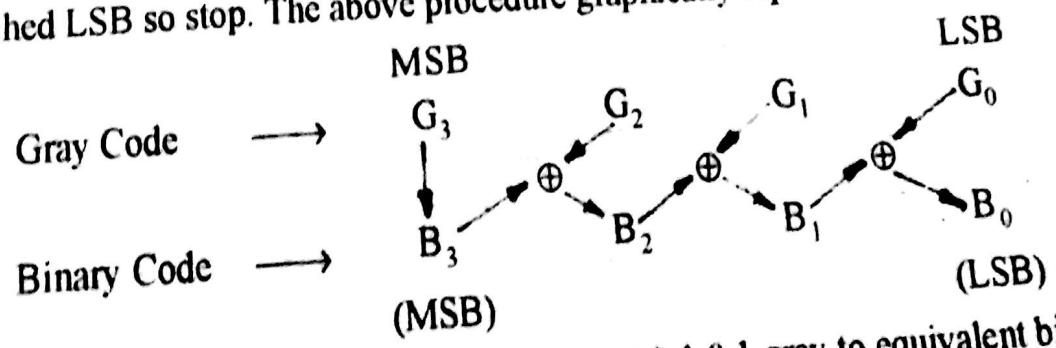
$\therefore \quad B_2 = B_3 \oplus G_2$.

(c) Resulting bit in step II is $B_2$, is EX–ORed with next bit from gray i.e. $G_1$. i.e. $B_2 \oplus G$ resulting bit is $B_1$, $B_2 \oplus G_1 = B_1$.

(a) Step III Continued

$\therefore \quad B_0 = B_1 \oplus G_0$.

Now we reached LSB so stop. The above procedure graphically represented as follows :




Let's take one example to understand the same convert 0 1 0 1 gray to equivalent binary.

Scanned by CamScanner

ou can perform by **Addition** method. For ac

gray to binary is shown in Table 27.

| Gray | | | | Binary | | | |
|---|---|---|---|---|---|---|---|
| $G_3$ | $G_2$ | $G_1$ | $G_0$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

Scanned by CamScanner

Now we are going to solve some examples.

## Solved Examples

**Ex. 52 :** Perform following Conversions :

(i) $(623 . 77)_8$ to binary and hex.

(ii) $(AF19)_{16}$ to Binary and octal.

(iii) $(11010111.110)_2$ to octal and hex.

(iv) $(AB . COF)_{16}$ to octal and decimal.

**Soln. :**

(i) $(623 . 77)_8 = (110 \quad 010 \quad 011 . \quad 111\ 111)_2$

$\therefore$ $\underline{0001}$ $\underline{1001}$ $\underline{0011}$ . $\underline{1111}$ $\underline{1100}$ $\longrightarrow$ binary

1 9 3 . F C $\longrightarrow$ hex.

$\therefore$ $(623 . 77) = (193 . FC)_{16}$.

(ii) $(A \quad F \quad 1 \quad 9)_{16}$ $\longrightarrow$ hex

$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$

$(1010 \quad 1111 \quad 0001 \quad 1001)_2$ $\longrightarrow$ Binary

hex to binary conversion.

$\underline{001}$ $\underline{010}$ $\underline{111}$ $\underline{100}$ $\underline{011}$ $\underline{001}$ $\longrightarrow$ binary

$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$

1 2 7 4 3 1 $\longrightarrow$ Octal

binary to octal conversion

(iii) $(1\ 1\ 0\ 1\ 0\ 1\ 1\ 1 . 1\ 1\ 0)_2 = (?)_8 = (?)_{16}$

$(\underline{011}$ $\underline{010}$ $\underline{111}$ . $\underline{110})_2$

$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$

(3 2 7 . $6)_8$ = $(327 . 6)_8$

$\underline{1101}$ $\underline{0111}$ . $\underline{1110}$ $\longrightarrow$ binary

D 7 . E $\longrightarrow$ hex

$(1\ 1\ 0\ 1\ 0\ 1\ 1\ 1 . 1\ 1\ 0)_2 = (327 . 6)_8 = (D7 . E)_{16}$

(iv)

$(A \quad B \quad . \quad C \quad 0 \quad F)_{16}$ $\longrightarrow$ Hex

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

$(1010 \quad 1011 \quad 1100 \quad 0000 \quad 1111)$ $\longrightarrow$ binary

hex to binary

$\underline{010}$ $\underline{101}$ $\underline{011}$ . $\underline{110}$ $\underline{000}$ $\underline{001}$ $\underline{111}$ $\longrightarrow$ binary

2 5 3 . 6 0 1 7 $\longrightarrow$ octal

= $(253 . 6017)_8$

**Decimal Number**

$= A \times 16^1 + B \times 16^0 + C \times 16^{-1} + 0 \times 16^{-2} + F \times 16^{-3}$

$= 10 \times 16 + 11 \times 1 + 12 \times \frac{1}{16} + 0 + \frac{15}{16^3}$

$= (171 . 07536)_{10}$.

Scanned by CamScanner

**Ex. 53 :** Perform following conversions :

(i) $(919 . 89)_{10}$ to binary.
(ii) $(204 . 1250)_{10}$ to hex.
(iii) $(625 . 625)_{10}$ to octal.
(iv) $(11001011 . 01110)_2$ to decimal.
(v) $(574)_8$ to binary and decimal.

**Soln. :**

**(i)** Integer 919.

| | 919 | | |
|---|---|---|---|
| | **Quotient** | **Remainder** | |
| 2 | 459 | 1 | ⟶ (LSB) |
| 2 | 229 | 1 | |
| 2 | 114 | 1 | |
| 2 | 57 | 0 | |
| 2 | 28 | 1 | |
| 2 | 14 | 0 | |
| 2 | 7 | 0 | |
| 2 | 3 | 1 | |
| 2 | 1 | 1 | |
| | | 1 | ⟶ (MSB) |

For fractional part.

| Decimal fraction | | base | | Answer | Recorded bit | |
|---|---|---|---|---|---|---|
| 0.89 | × | 2 | = | 1.78 | 1 | ⟶ (MSB) |
| 0.78 | × | 2 | = | 1.56 | 1 | |
| 0.56 | × | 2 | = | 1.12 | 1 | |
| 0.12 | × | 2 | = | 0.24 | 0 | |
| 0.24 | × | 2 | = | 0.48 | 0 | ⟶ (LSB) |

$\therefore \quad (919 . 89)_{10} = (1110010111 . 1110)_2$

**(ii)** Integer.

| | 204 | | | | |
|---|---|---|---|---|---|
| | **Quotient** | **Remainder** (Decimal) | | (hex) | |
| 16 | 12 | 12 | ⟶ | C | ⟶ (LSB) |
| | | 12 | ⟶ | C | ⟶ (LSB) |

*Fractional part.*

| Decimal fraction | | base | | Answer | Recorded bit (Decimal) (hex) |
|---|---|---|---|---|---|
| 0.125 | × | 16 | = | 2.000 | 2 ⟶ 2 |

$\therefore \quad (204 . 125)_{10} = (CC . 2)_{16}$

Scanned by CamScanner

**(iii)** $(625 . 625)_{10}$ to octal.

Integer

| | | | |
|---|---|---|---|
| 8 | 625 | ⟶ | Original Number |
| | Quotient | Remainder | |
| 8 | 78 | 1 | ⟶ (LSB) |
| 8 | 9 | 6 | |
| 8 | 1 | 1 | |
| | | 1 | ⟶ (MSB) |

Fractional part.

$0.625 \times 8 = 5$

$\therefore \quad (625 . 625)_{10} = (1161 . 5)_8$

**(iv)** $(11001011 . 01110)_2$ to decimal.

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

$= (202 . 4375)_{10}$

**(v)** $(574)_8$

$= 5 \times 8^2 + 7 \times 8^1 + 4 \times 8^0 = 5 \times 64 + 56 + 4 = (380)_{10}$

$\therefore \quad (574)_8 = (380)_{10}$

| | | | | |
|---|---|---|---|---|
| $(5$ | $7$ | $4)_8$ | ⟶ | Octal |
| ↓ | ↓ | ↓ | | Octal to binary |
| 1 0 1 | 1 1 1 | 1 0 0 | ⟶ | binary |

$\therefore \quad (574)_8 = (101111100)_2$

---

**Ex. 54 :** Perform following conversions :

(i) $(18)_{10}$ to XS–3.

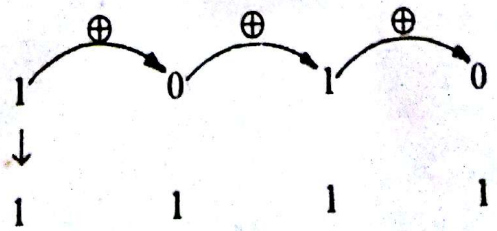
**Soln. :**

| | | | |
|---|---|---|---|
| $(1$ | $8)_{10}$ | ⟶ | decimal |
| $+3$ | $+3$ | ⟶ | Add 3 |
| 4 | 11 | ⟶ | XS–3 |
| ⇓ | ⇓ | | |
| 0 1 0 0 | 1 0 1 1 | ⟶ | XS – 3 (binary) |

$\therefore \quad (18)_{10} = (0100\ 1011)_{XS-3}$

**(ii) $(1010)_2$ to Gray Code.**

**Soln. :**



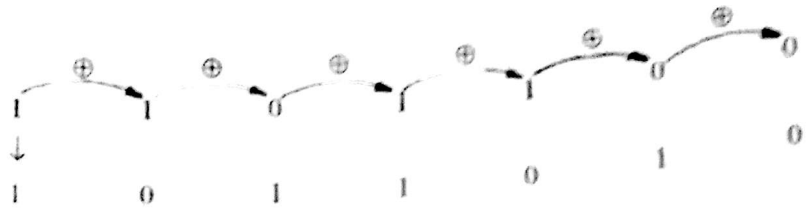

Scanned by CamScanner

**(iii) $(13)_{10}$ to 8421 BCD.**

**Soln. :** $(13)_{10} = (0001\ \ 0011)_{8421\ BCD}$

**(iv) $(1101100)_2$ to gray.**

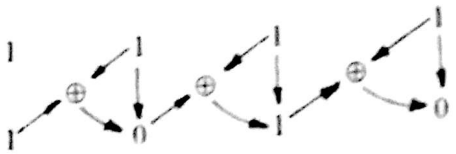
**Soln. :**




$\therefore \quad (1101100)_2 = 1011010$ gray

**(v) 1111 gray to binary.**

**Soln. :**




$\therefore \quad (1111)$ gray $= (1010)_2$

**(vi) $(847)_{10}$ to 8421 BCD, XS–3 and gray code.**

**Soln. :** (a) $(847)_{10} = 1000 \quad 0100 \quad \longrightarrow$ 8421 BCD

(b)

| | | | |
|---|---|---|---|
| | 8 | 4 | 7 ⟶ decimal |
| + | + | + | |
| | 3 | 3 | 3 ⟶ add 3 |
| | 11 | 7 | 10 XS–3 (Excess – 3) |
| | ↓ | ↓ | ↓ |
| | 1011 | 0111 | 1010 |

$\therefore \quad (847)_{10} = (1011 \quad 0111 \quad 1010)_{XS-3}$

(c) $(847)_{10}$ to gray.

For this first convert decimal to binary and then binary to gray.

| | 847 | | |
|---|---|---|---|
| | **Quotient** | **Remainder** | |
| 2 | 423 | 1 | → (LSB) |
| 2 | 211 | 1 | |
| 2 | 105 | 1 | |
| 2 | 52 | 1 | |
| 2 | 26 | 0 | |
| 2 | 13 | 0 | |
| 2 | 6 | 1 | |
| 2 | 3 | 0 | |
| 2 | 1 | 1 | |
| | | 1 | → (MSB) |

Scanned by CamScanner

| Decimal | 3 3 2 1 | 3 3 2 1 | 4 4 3 $\bar{2}$ | 4 4 3 $\bar{2}$ |
|---|---|---|---|---|
| 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| 1 | 0 0 0 1 | 0 0 0 1 | 0 0 1 1 | 0 0 1 1 |
| 2 | 0 0 1 0 | 0 0 1 0 | 0 1 0 1 | 1 0 0 1 |
| 3 | 0 0 1 1 | 0 1 0 0 | 0 0 1 0 | 0 0 1 0 |
| 4 | 0 1 0 1 | 1 0 0 1 | 0 1 0 0 | 1 0 0 0 |
| 5 | 1 0 1 0 | 0 1 1 0 | 1 0 1 1 | 0 1 1 1 |
| 6 | 1 1 0 0 | 1 0 1 1 | 1 1 0 1 | 1 1 0 1 |
| 7 | 1 1 0 1 | 1 1 0 1 | 1 0 1 0 | 0 1 1 0 |
| 8 | 1 1 1 0 | 1 1 1 0 | 1 1 0 0 | 1 1 0 0 |
| 9 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 |

Scanned by CamScanner

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

**Soln. :**

| Decimal | 7 4 2 1 | 8 4 2 1 |
|---|---|---|
| 0 | 0 0 0 0 | 0 0 0 0 |
| 1 | 0 0 0 1 | 0 0 0 1 |
| 2 | 0 0 1 0 | 0 0 1 0 |
| 3 | 0 0 1 1 | 0 0 1 1 |
| 4 | 0 1 0 0 | 0 1 0 0 |
| 5 | 0 1 0 1 | 0 1 0 1 |
| 6 | 0 1 1 0 | 0 1 1 0 |
| 7 | 1 0 0 0 | 0 1 1 1 |
| 8 | 1 0 0 1 | 1 0 0 0 |
| 9 | 1 0 1 0 | 1 0 0 1 |

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

(i) 1001

(ii) 1010

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

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

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

$1\ 0\ 0\ P_4\ 1\ P_2\ P_1$

Selection should be such that we get even parity for

(a) $P_4\ D_5\ D_6\ D_7$

$-\ 0\ 0\ 1 \Rightarrow P_4 = 1$

(b) $P_2\ D_3\ D_6\ D_7$

$-\ 1\ 0\ 1 \Rightarrow P_2 = 0$

(c) $P_1\ D_3\ D_5\ D_7$

$-\ 1\ 0\ 1 \Rightarrow P_1 = 0$

$\therefore$ The 7 bit even parity Hamming Code is

$7\ 0\ 0\ 1\ 1\ 0\ 0$

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

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

$1\ 0\ 1\ P_4\ 0\ P_2\ P_1$

Selection should be such that we get even parity for

(a) $P_4\ D_5\ D_6\ D_7$

$-\ 1\ 0\ 1 \Rightarrow P_4 = 0$

(b) $P_2\ D_3\ D_6\ D_7$

$-\ 0\ 0\ 1 \Rightarrow P_2 = 1$

(c) $P_1\ D_3\ D_5\ D_7$

$-\ 0\ 1\ 1 \Rightarrow P_1 = 0$

$\therefore$ 7 bit hamming code is.

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

$1\ 0\ 1\ 0\ 0\ 1\ 0$

Scanned by CamScanner

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

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

1 1 1 1 1 0 1

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

(a) $P_4\ D_5\ D_6\ D_7$

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

(b) $P_2\ D_3\ D_6\ D_7$

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

(c) $P_1\ D_3\ D_5\ D_7$

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

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

$\therefore$ Correct Code is 1111 111.

---

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

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

1 1 1 0 1 0 1

Bits $P_4\ D_5\ D_6\ D_7$

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

Bits $P_2\ D_3\ D_6\ D_7$

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

Bits $P_1\ D_3\ D_5\ D_7$

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

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

---

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

**Soln. :** The hamming code is,

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

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

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

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

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

Scanned by CamScanner

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

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

∴ Final code used is,

$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1 = 1\ 1\ 0\ 0\ 0\ 0\ 1$

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

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

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

This can be written as

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

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

∴ $2x^2 + x + 1 = 106$

∴ $2x^2 + x - 105 = 0$

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

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

**Ans.** $\boxed{x = 7}$

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

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

∴ $x^2 + 9x + 3 = 403$

∴ $x^2 + 9x - 400 = 0$

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

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

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

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

∴ $2x^2 + 2x - 200 = 0$

∴ $x^2 + x - 100 = 0$

Solve this using equation given.

∴ $\boxed{x = 10}$

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

**Soln. :** (a) BCD 8421

8 6 2 0 ⟶ decimal

1000 0110 0010 0000 ⟶ 8421 BCD.

(b) XS–3

8 6 2 0 ⟶ decimal

+ 3 3 3 3 ⟶ Add 3

11 9 5 3

↓ ↓ ↓ ↓

1011 1000 0101 0011 ⟶ XS–3

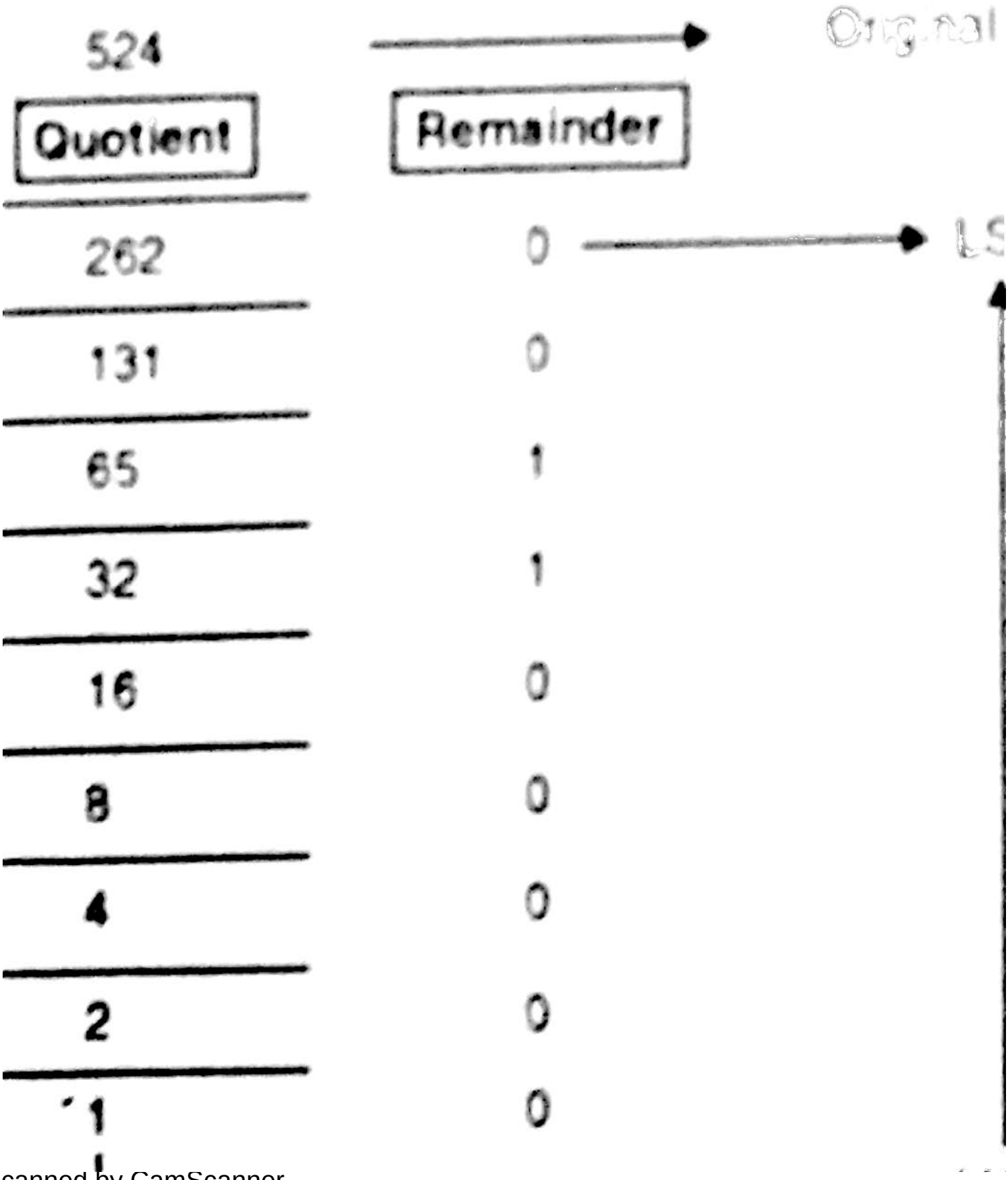
(c) 2421 Code.

8 6 2 0 ⟶ Decimal

1110 1100 0010 0000 ⟶ 2421 Code.

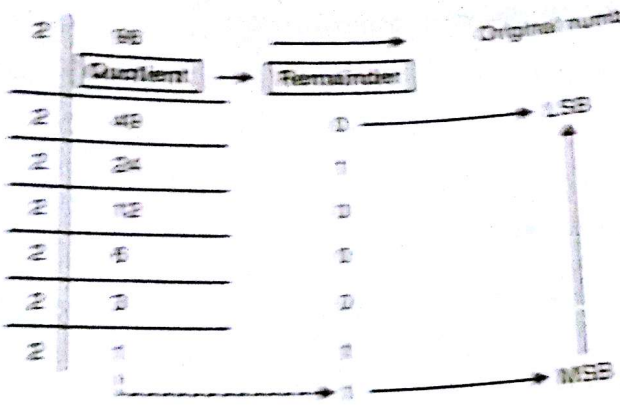
Scanned by CamScanner

## ›ase 10 numbers to base 2 numbers :




Scanned by CamScanner

(ii) $(98)_{10}$




...Ans.

$\therefore (98)_{10} = (1100010)_2$

2. Convert following octal numbers to decimal numbers : (May 96, 4 Marks)

(i) 745 (ii) 52

Ans. : (i) $(745)_8$

$= 7 \times 8^2 + 4 \times 8^1 + 5 \times 8^0$

$= 7 \times 64 + 32 + 5 \times 1$

$= 485$

...Ans.

$\therefore (745)_8 = (485)_{10}$

(ii) $(52)_8$

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

$= 40 + 2 \times 1$

$= 42$

...Ans.

$\therefore (52)_8 = (42)_{10}$

3 (a) Write short notes on : (May 96, 6 Marks)

(i) Excess– 3 code

(ii) Hamming code (May 96, Dec. 96, May 97, Dec. 97, 6 Marks)

Ans. : (i) Refer Section 1.11.3

(ii) Refer Section 1.13.6

4. Determine equivalent decimal number for the following binary numbers : (Dec. 96, 2 Marks)

(i) 1101.0101 (ii) 1010.1101.

Ans. : (i) $(1101.0101)_2$ :

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

$= 8 + 4 + 0 + 1 + 0 + 0.25 + 0 + 0.0625$

$= 13.3125$

...Ans.

$\therefore (1101.0101)_2 = (13.3125)_{10}$

Scanned by CamScanner

(ii) $(1010.1100)_2$

$$= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4}$$
$$= 8 + 0 + 2 + 0 + 0.5 + 0.25 + 0 + 0$$
$$= 10.75$$

$\therefore$ $\boxed{(1010.1100)_2 = (10.75)_{10}}$ ...Ans.

**5. Convert following base 8 numbers to base 2 numbers :** **(Dec. 96, 2 Marks)**

**(i) 764 (ii) 46**

**Ans. :** (i) $(764)_8$

In binary $7 = 1\ 1\ 1$, $6 = 1\ 1\ 0$, $4 = 1\ 0\ 0$ } 3 bit binary representation

$\therefore$ $\boxed{(764)_8 = (111110100)_2}$ ...Ans.

(ii) $(46)_8$

In binary $4 = 1\ 0\ 0$, $6 = 1\ 1\ 0$ } 3 bit binary representation

$\therefore$ $\boxed{(46)_8 = (100110)_2}$ ...Ans.

**6. Convert from BCD to Hex the following numbers :** **(Dec. 96, 4 Marks)**

**(i) 402 (ii) 83**

**Ans.:** (i) First convert the given decimal number to BCD.

$\therefore$ $(402)_{10} = (0\ 1\ 0\ 0 \quad 0\ 0\ 0\ 0 \quad 0\ 0\ 1\ 0)_{BCD}$

$\downarrow$ $\downarrow$ $\downarrow$

4 0 2

Now convert $(402)_{10}$ to Binary equivalent.

| 2 | 402 → Original number |
|---|---|
| | **Quotient** | **Remainder** |
| 2 | 201 | 0 → LSB |
| 2 | 100 | 1 |
| 2 | 50 | 0 |
| 2 | 25 | 0 |
| 2 | 12 | 1 |
| 2 | 6 | 0 |
| 2 | 3 | 0 |
| 2 | 1 | 1 |
| | | 1 → MSB |

$\therefore$ $(402)_{10} = (1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0)_2$

Scanned by CamScanner

Now convert the binary number to Hex making group of 4 bits from right to left.

| 0 0 0 1 | 1 0 0 1 | 0 0 1 0 |
|---|---|---|
| 1 | 9 | 2 |

$\therefore$ $(402)_{10} = (1\,1\,0\,0\,1\,0\,0\,1\,0)_2 = (192)_{Hex}$

$\therefore$ $(402)_{10} = (0\,1\,0\,0 \;\; 0\,0\,0\,0 \;\; 0\,0\,1\,0)_{BCD} = (1\,1\,0\,0\,1\,0\,0\,1\,0)_2$

$= (192)_{Hex}$ ...Ans.

(ii) First Convert given decimal number to BCD

$\therefore$ $(83)_{10} = (1\,0\,0\,0 \;\; 0\,0\,1\,1)_{BCD}$

$\downarrow \quad \downarrow$

$8 \quad 3$

Now convert $(83)_{10}$ to binary equivalent.

| 2 | 83 (Quotient) | Remainder | |
|---|---|---|---|
| 2 | 41 | 1 | → LSB |
| 2 | 20 | 1 | |
| 2 | 10 | 0 | |
| 2 | 5 | 0 | |
| 2 | 2 | 1 | |
| 2 | 1 | 0 | |
| | i | 1 | → MSB |

Original number

$\therefore$ $(83)_{10} = (1\,0\,1\,0\,0\,1\,1)_2$

Now convert the binary number to Hex, making group of 4bits from right to left.

| 0 1 0 1 | 0 0 1 1 |
|---|---|
| 5 | 3 |

$\therefore$ $(83)_{10} = (1\,0\,1\,0\,0\,1\,1)_2 = (53)_H$

$\therefore$ $(83)_{10} = (1\,0\,0\,0 \;\; 0\,0\,1\,1)_{BCD} = (1\,0\,1\,0\,0\,1\,1)_2$

$= (53)_H$ ...Ans.

**7. Write short note on : Gray code and its applications.** **(Dec. 96, May 97, 6 Marks)**

**Ans. :** Refer Section 1.11.2 and 1.15.6 [depicts 2 steps of converting binary to gray)

**8. Write short note on : Error correcting codes.** **(Dec. 96, Dec. 97, 6 Marks)**

**Ans. :** Refer Section 1.13.6

**. Determine equivalent decimal numbers for the following binary numbers :(May 97, 2 Marks)**

**(i) 1001.1010 (ii) 1110.1011**

Scanned by CamScanner

**Ans. :** (i) $(1001.1010)_2$

Write binary number → 1 0 0 1 . 1 0 1 0

Weights of binary number → $2^3$ $2^2$ $2^1$ $2^0$ . $2^{-1}$ $2^{-2}$ $2^{-3}$ $2^{-4}$

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

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

$= 9.625$

$\therefore$ $\boxed{(1001.1010)_2 = (9.625)_{10}}$ ...Ans.

(ii) $(1110.1011)_2$

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

$= 8 + 4 + 2 + 0 + 0.5 + 0 + 0.125 + 0.0625$

$= 14.6875$

$\therefore$ $\boxed{(1110.1011)_2 = (14.6875)_{10}}$ ...Ans.

**10. Convert the following base 8 numbers to base 2 numbers :** (May 97, 2 Marks)

**(i) 76 (ii) 457**

**Ans.:** (i) $(76)_8$

In binary $\left.\begin{matrix} 7 = 1\ 1\ 1 \\ 6 = 1\ 1\ 0 \end{matrix}\right\}$ 3 bit binary representation.

$\therefore$ $\boxed{(76)_8 = (1\ 1\ 1\ 1\ 1\ 0)_2}$ ...Ans.

(ii) $(457)_8$

In binary $\left.\begin{matrix} 4 = 1\ 0\ 0 \\ 5 = 1\ 0\ 1 \\ 7 = 1\ 1\ 1 \end{matrix}\right\}$ 3 bit binary representation

$\therefore$ $\boxed{(457)_8 = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1)_2}$ ...Ans.

**11. Convert from BCD to Hex the following numbers :** (May 97, 2 Marks)

**(i) 958 (ii) 77**

**Ans. :** (i) First convert the given decimal number to BCD

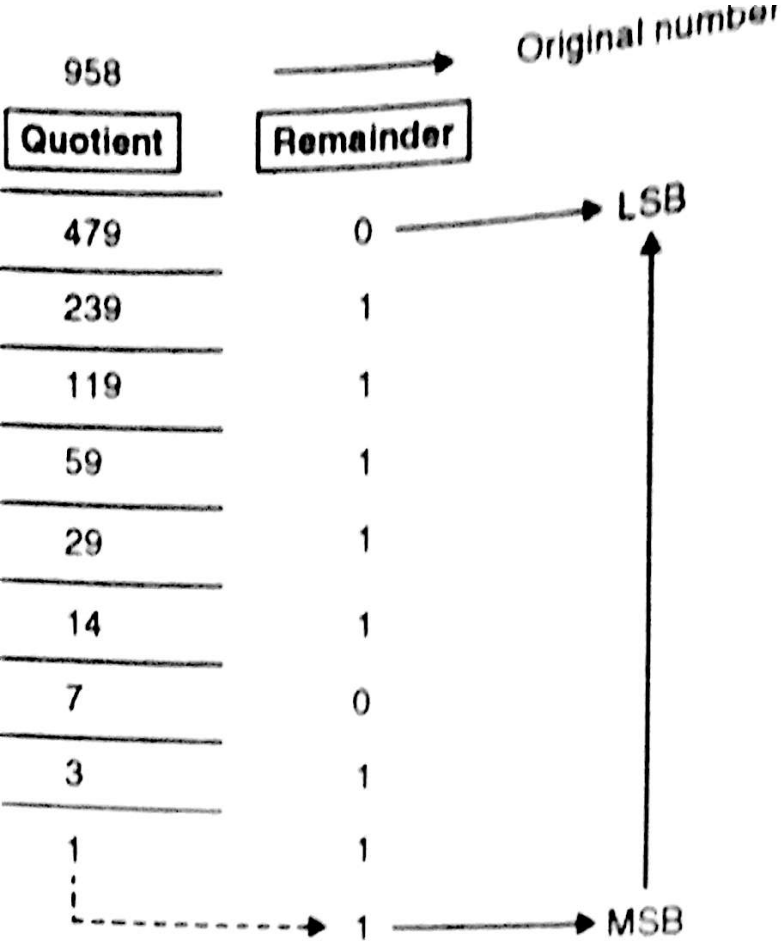
$\therefore$ $(958)_{10} = (1\ 0\ 0\ 1 \quad 0\ 1\ 0\ 1 \quad 1\ 0\ 0\ 0)_{BCD}$

$\downarrow$ $\downarrow$ $\downarrow$

9 5 8

Now convert $(958)_{10}$ to binary equivalent.

Scanned by CamScanner




$1\ 1\ 0\ 1\ 1\ 1\ 1\ 0)_2$

r to hex making group of 4 bits from right to left.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| | | B | | | | E | |

$1\ 1\ 0\ 1\ 1\ 1\ 1\ 0)_2 = (3BE)_{Hex}$

$0\ 0\ 0\ \ 0\ 0\ 1\ 1)_{BCD} = (1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0)_2$

$3E)_{Hex}$

**Ans**

Scanned by CamScanner

Now convert binary number to Hex making group of 4 bits from right to left.

$$\underbrace{0\ 1\ 0\ 0}_{4}\quad\underbrace{1\ 1\ 0\ 1}_{D}$$

$\therefore$ $(77)_{10} = (1\,0\,0\,1\,1\,0\,1)_2 = (4D)_{Hex}$

$\therefore$ $\boxed{(77)_{10} = (0\,1\,1\,1\ \ 0\,1\,1\,1)_{BCD} = (1\,0\,0\,1\,1\,0\,1)_2 = (4D)_{Hex}}$ ...Ans.

**12. Explain the difference between error detecting code and error correcting code.**

**(May 97, 6 Marks)**

**Ans. :** Refer Section 1.13.

**13. Determine equivalent decimal numbers for the following binary numbers:(Dec. 97, 2 Marks)**

**(i) 1100.1101 (ii) 1110.0101**

**Ans.:** (i) $(1100.1101)_2$

Given binary number is

$$= 1100.1101$$
$$= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$$
$$= 8 + 4 + 0 + 0 + 0.5 + 0.25 + 0 + 0.0625$$
$$= 12.8125$$

$\therefore$ $\boxed{(1100.1101)_2 = (12.8125)_{10}}$ ...Ans.

(ii) $(1110.0101)_2$

Given binary number is

$$= 1110.0101$$
$$= 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$$
$$= 8 + 4 + 2 + 0 + 0 + 0.25 + 0 + 0.0625$$
$$= 14.3125$$

$\therefore$ $\boxed{(1110.0101)_2 = (14.3125)_{10}}$ ...Ans.

**14. Convert following octal number to binary numbers : (Dec. 97, 2 Marks)**

**(i) 175 (ii) 36**

**Ans. :** (i) $(175)_8$

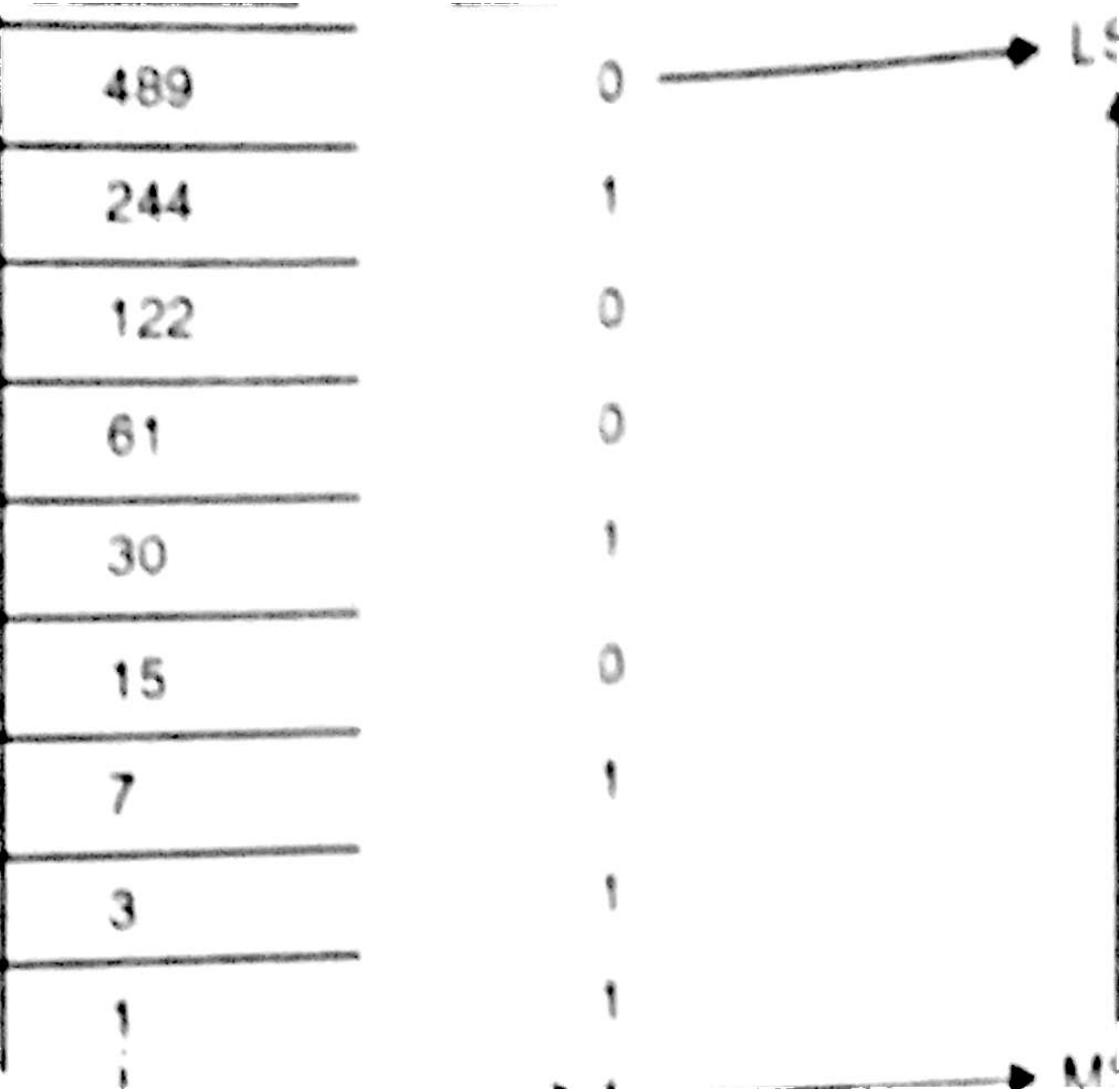
In binary $\left.\begin{matrix}1 = 0\,0\,1\\ 7 = 1\,1\,1\\ 5 = 1\,0\,1\end{matrix}\right\}$ 3 bit binary representation

Equivalent binary number is, $\boxed{(175)_8 = (001111101)_2}$ ...Ans.

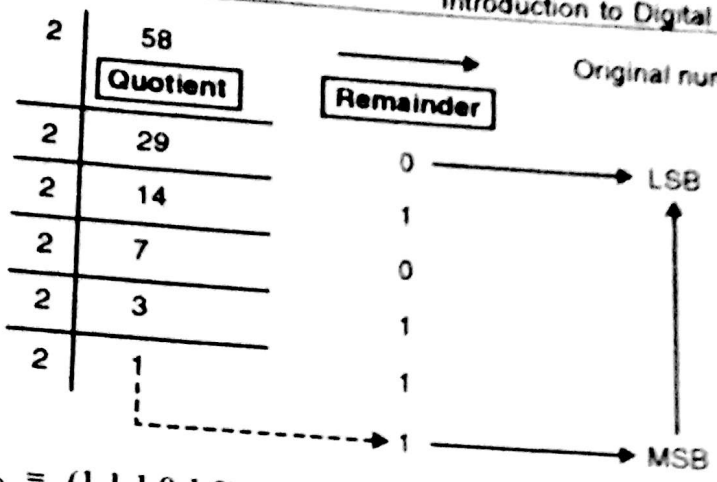
(ii) $(36)_8$

In binary $\left.\begin{matrix}3 = 0\,1\,1\\ 6 = 1\,1\,0\end{matrix}\right\}$ 3 bit binary representation

Scanned by CamScanner




Scanned by CamScanner




$\therefore \quad (58)_{10} = (1\ 1\ 1\ 0\ 1\ 0)_2$

Now convert binary to Hex making group of 4 bits from right to left.

$$\underbrace{0\ 0\ 1\ 1}_{3} \qquad \underbrace{1\ 0\ 1\ 0}_{A}$$

$\therefore \quad (58)_{10} = (1\ 1\ 1\ 0\ 1\ 0)_2 = (3A)_{Hex}$

$\therefore \quad \boxed{(58)_{10} = (0\ 1\ 0\ 1\ \ 1\ 0\ 0\ 0)_{BCD} = (1\ 1\ 1\ 0\ 1\ 0)_2 = (3A)_{Hex}}$ ...Ans.

**16. Determine equivalent decimal numbers for the following binary numbers:** (May 98, 2 Marks)

**(i) 1101.1011 (ii) 0110.1110**

**Ans. :** (i) $(1101.1011)_2$

$$= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$$
$$= 8 + 4 + 0 + 1 + 0.5 + 0.125 + 0.0625$$
$$= 13.6875$$

$\therefore \quad \boxed{(1101.1011)_2 = (13.6875)_{10}}$ ...Ans.

(ii) $(0110.1110)_2$

$$= 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4}$$
$$= 0 + 4 + 2 + 0 + 0.5 + 0.25 + 0.125 + 0$$
$$= 6.875$$

$\therefore \quad \boxed{(0110.1110)_2 = (6.875)_{10}}$ ...Ans.

**17. Convert the following numbers to binary numbers :** (May 98, 2 Marks)

**(i) $(123)_4$ (ii) $(230)_4$**

**Ans.: (i) $(123)_4$**

inary equivalent of $\left.\begin{matrix} 1 = 0\ 1 \\ 2 = 1\ 0 \\ 3 = 1\ 1 \end{matrix}\right\}$ 2 bit binary representation

$\boxed{(123)_4 = (011011)_2}$ ...An

Scanned by CamScanner

(ii) $(230)_4$

Binary equivalent of $\left.\begin{matrix} 2 = 1\ 0 \\ 3 = 1\ 1 \\ 0 = 0\ 0 \end{matrix}\right\}$ 2 bit binary representation ...Ans.

$\therefore$ $\boxed{(230)_4 = (101100)_2}$

**18. Certain number system has base 7. What is the hexadecimal equivalent of the minimum and maximum number that is expressed using the base 7 and four bits? (May 98, 4 Marks)**

**Ans.:** A number system having base 7 represents numbers from 0, 1, ... 6. But they are expressed in single bit.

So the minimum number in system which has base 7 and is expressed in 4 bits = $(0000)_7$

$\therefore$ Hexadecimal equivalent of minimum number having base 7 is ...Ans.

$\therefore$ $\boxed{(0000)_7 = (0000)_{16}}$

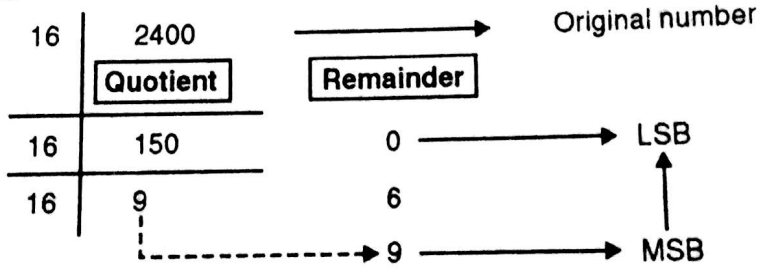
Similarly Maximum 4 bit number using base 7 is = $(6666)_7$

For hexadecimal equivalent first convert given number into decimal number.

$\therefore$ To find hexadecimal equivalent of $(6666)_7$ convert it into decimal number.

$$\therefore \quad (6666)_7 = 6 \times 7^3 + 6 \times 7^2 + 6 \times 7^1 + 6 \times 7^0$$
$$= 2058 + 294 + 42 + 6$$
$$= 2400$$
$$\therefore \quad (6666)_7 = (2400)_{10}$$

Now for converting decimal to hex number,




$\therefore$ $(2400)_{10} = (960)_{16}$

$\therefore$ $\boxed{\text{Hexadecimal equivalent of maximum number : } (6666)_7 = (960)_{16}}$ ...Ans.

**19. Determine equivalent decimal numbers for the following binary numbers: (Dec. 98, 2Marks)**

**(i) 1001.0111 (ii) 1010.1011**

**Ans. :** (i) $(1001.0111)_2$

$\therefore$ Equivalent decimal number is

$$= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$$
$$= 8 + 0 + 0 + 1 + 0 + 0.25 + 0.125 + 0.0625$$
$$= 9.4375$$

$\therefore$ $\boxed{(1001.0111)_2 = (9.4375)_{10}}$

...Ans.

Scanned by CamScanner

(ii) $(1010.1011)_2$

∴ Equivalent decimal number is

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

$$= 8 + 0 + 2 + 0 + 0.5 + 0 + 0.125 + 0.0625$$

$$= 10.6875$$

∴ $\boxed{(1010.1011)_2 = (10.6875)_{10}}$ ...Ans.

**20. What is the necessity of a parity code ?** **(Dec. 98, 3 Marks)**

**Ans. :** Refer Section 1.13 and 1.13.1.

**21. Determine equivalent hexadecimal numbers for the following decimal numbers :**

**(i) 2338 (ii) 0.122** **(June 99, 2 Marks)**

**Ans. :** (i) Refer Section 1.6.1, Ex. 21.

(ii) Refer Section 1.6.1, Ex. 24.

**22. Determine equivalent decimal numbers for the following octal numbers : (June 99, 2 Marks)**

**(i) 732 (ii) 0.123**

**Ans.:** (i) $(732)_8$

$$= 7 \times 8^2 + 3 \times 8^1 + 2 \times 8^0$$

$$= (474)_{10}$$

∴ $\boxed{(732)_8 = (474)_{10}}$ ...Ans.

(ii) 0.123

**Ans.:** Refer Section 1.5.2, Ex. 19.

**23. Perform the following conversions :** **(June 99, 3 Marks)**

**$(28)_{10}$ to 8421 BCD, XS – 3 and gray code.**

**Ans.:** $(28)_{10}$

(a) BCD 8421

2 8 → decimal

↓ ↓

$\boxed{0\ 0\ 1\ 0 \quad 1\ 0\ 0\ 0 \rightarrow \text{8421 BCD}}$ ...Ans.

(b) XS – 3

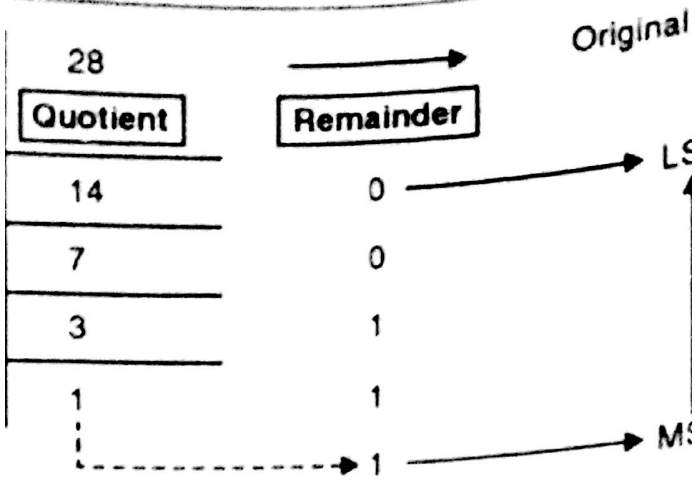
| | | | |
|---|---|---|---|
| | 2 | 8 | → decimal number |
| + | 3 | 3 | → Add 3 |
| | 5 | 11 | |
| | ↓ | ↓ | |

$\boxed{0\ 1\ 0\ 1 \quad 1\ 0\ 1\ 1 \rightarrow \text{XS} - 3}$ ...Ans.

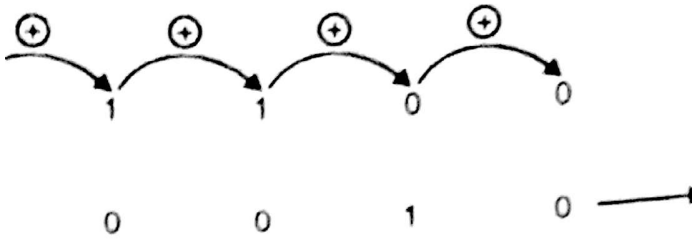
(c) $(28)_{10}$ to gray

For this first convert decimal to binary and then binary to gray.

Scanned by CamScanner




$= (11100)_2$




$_0 = (10010)_{gray}$

**4 – bit binary word into a seven bit even p**

5.7, Ex. 58. on Page 1.70.

**ing BCD number to hexadecimal number :**

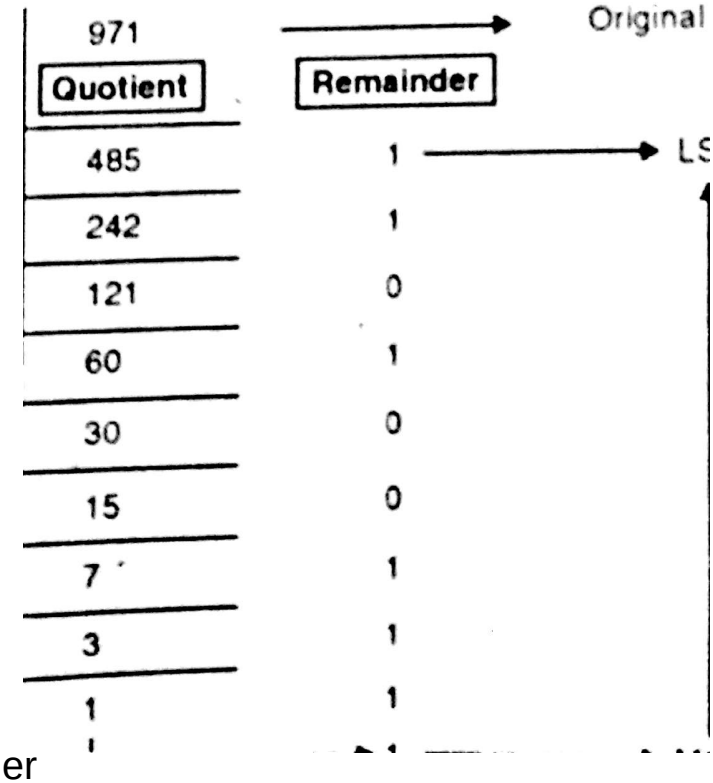
BCD

$= (1\ 0\ 0\ 1\ \ 0\ 1\ 1\ 1\ \ 0\ 0\ 0\ 1)_{BCD}$

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

$9 \quad 7 \quad 1$

binary equivalent.




Scanned by CamScanner

$\therefore$ $(971)_{10} = (1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1)_2$

Now convert binary to hex making group of 4 bits from right to left.

$\underset{3}{0\ 0\ 1\ 1}\ \underset{C}{1\ 1\ 0\ 0}\ \underset{B}{1\ 0\ 1\ 1}$

$\therefore$ $(971)_{10} = (1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1)_2 = (3CB)_{Hex}$

$\therefore$ $\boxed{(971)_{10} = (1\ 0\ 0\ 1\ \ 0\ 1\ 1\ 1\ \ 0\ 0\ 0\ 1)_{BCD} = (1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1)_2 = (3CB)_{Hex}}$ ...Ans.

**26. Perform the following conversions :**
**$(19)_{10}$ to 8421 BCD, XS – 3 and gray code.** **(Dec. 99, 3 Marks)**

**Ans. :** $(19)_{10}$

(a) BCD 8421

$$\begin{array}{cc} 1 & 9 \\ \downarrow & \downarrow \\ 0\ 0\ 0\ 1 & 1\ 0\ 0\ 1 \end{array} \longrightarrow \text{Decimal} \longrightarrow \text{8421 BCD}$$

$\therefore$ $\boxed{(19)_{10} = (0001\ 1001)_{BCD}}$ ...Ans.

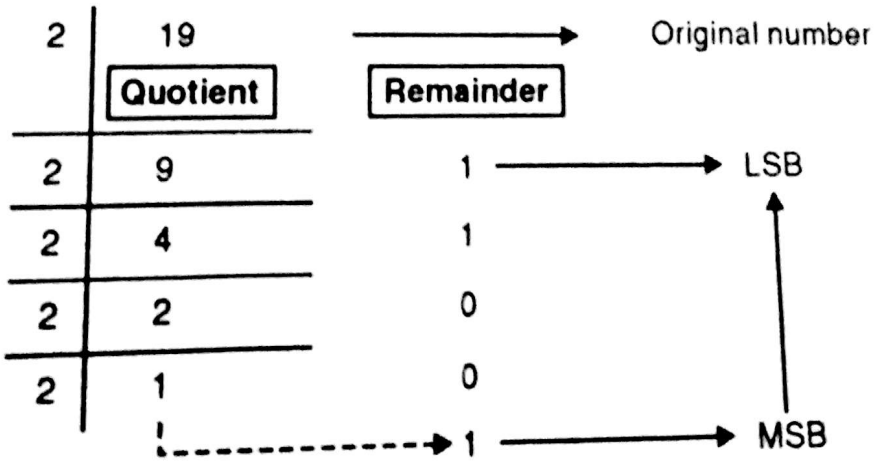
(b) XS – 3

$$\begin{array}{cccc} & 1 & 9 & \rightarrow \text{decimal} \\ + & 3 & 3 & \rightarrow \text{Add 3} \\ \hline & 4 & 12 & \\ & \downarrow & \downarrow & \\ & 0\ 1\ 0\ 0 & 1\ 1\ 0\ 0 & \rightarrow \text{XS} - 3 \end{array}$$

$\therefore$ $\boxed{(19)_{10} = (0100\ 1100)\ XS - 3}$ ...Ans.

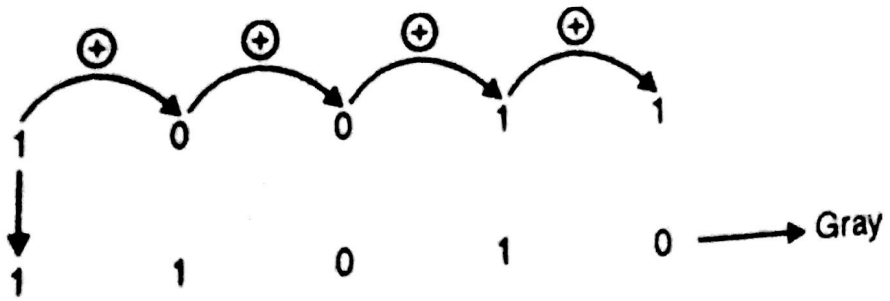
(c) Gray code

For this first convert decimal to binary and then binary to gray.




$\therefore$ $(19)_{10} = (10011)_2$

**Binary :**




Scanned by CamScanner

$= (11010)_{gray}$

## ng BCD number to its equivalent hexadecim

CD

$= (0110\ 1001\ 0111)_{BCD}$

s binary equivalent.

Original r

697

| Quotient | Remainder |
|---|---|
| 348 | 1 |
| 174 | 0 |
| 87 | 0 |
| 43 | 1 |
| 21 | 1 |
| 10 | 1 |
| 5 | 0 |
| 2 | 1 |
| 1 | 0 |

LS

M

Scanned by CamScanner

# Binary Arithmetic

2

**Syllabus :**

Signed number binary order, 1's and 2's Complement codes, Binary Arithmetic

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Binary addition / subtraction | 2.2 – 2.3 | – | ✓ |
| • Negative number representation in binary system | 2.4 | – | ✓ |
| • Binary subtraction using 1's and 2's complement method | 2.5 | – | ✓ |
| • Binary division / multiplication | 2.6 – 2.7 | – | ✓ |
| • Arithmetic in actual number system | 2.8 | – | ✓ |
| • Arithmetic in hexadecimal number system | 2.9 | – | ✓ |
| • BCD arithmetic | 2.10 | – | ✓ |
| • XS-3 arithmetic | 2.11 – 2.12 | – | ✓ |



**Comments**

- Binary arithmetic is base for the computers.
- All the concepts will be used in chapter - 5 for designing combinational logic circuits.

Scanned by CamScanner

## 2.1 Introduction :

In chapter 1, we studied about various numbering system like binary, octal and hex, codes etc. The main aim at that time was to develop numbering system which provides you less hardware to implement arithmetic and logical functions. In arithmetic normally we start with simple addition, subtraction, multiplication and division.

## 2.2 Binary Addition :

The rules for binary addition are given in Table 2.1. The Table shows addition of single bit of two given numbers.

| $A_0$ | | $B_0$ | | Sum | Carry |
|---|---|---|---|---|---|
| 0 | + | 0 | = | 0 | 0 |
| 0 | + | 1 | = | 1 | 0 |
| 1 | + | 0 | = | 1 | 0 |
| 1 | + | 1 | = | 0 | 1 |

Table 2.1

Carry

$A_0$
$+ B_0$
SUM
↓
$S_0$

As you are aware $A_0/B_0$ represents $2^0$ position of binary

$$\begin{array}{lll} & & A_0 \quad B_0 \\ A_0 = B_0 = 0 & \Rightarrow & 0 \times 2^0 + 0 \times 2^0 = 0 \\ A_0 = 0,\ B_0 = 1 & \Rightarrow & 0 \times 2^0 + 1 \times 2^0 = 1 \\ A_0 = 1,\ B_0 = 0 & \Rightarrow & 1 \times 2^0 + 0 \times 2^0 = 1 \\ A_0 = 1,\ B_0 = 1 & \Rightarrow & 1 \times 2^0 + 1 \times 2^0 = 1 + 1 = (2)_{10} \therefore \text{ in binary} \\ & & (2)_{10} = (10)_2 \qquad \therefore (\ \underset{\text{Carry}}{1} \quad \underset{\text{Sum}}{0}\ )_2 \end{array}$$

Let's perform some example to understand addition operation.

**Ex. 1 :** Add $(2)_{10}$ and $(3)_{10}$.

**Soln. :** $(2)_{10} = (0010)_2$, $(3)_{10} = (0011)_2$.

| | | (MSB) | | | (LSB) | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $(2)_{10}$ | $\Rightarrow$ | 0 | 0 | 1 | 0 | $\Rightarrow$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
| | | + | | | | | | | | |
| $(3)_{10}$ | $\Rightarrow$ | 0 | 0 | 1 | 1 | $\Rightarrow$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| Carry bits | $\Rightarrow$ | [0] | [0] | [1] | [0] | | | | | |
| | | 0 | 1 | 0 | 1 | | | | | |
| | | $(S_3)$ | $(S_2)$ | $(S_1)$ | $(S_0)$ | | | | | |

**Step I :** $A_0 + B_0 = 0 + 1 = 1$, carry = 0 (Refer Table 2.1), $\therefore S_0 = 1$.

**Step II :** $A_1 + B_1 = 1 + 1 = 0$, carry = 1

$\therefore$ carry is ***carried to next bits i.e. $A_2$, $B_2$ position***. The answer '0' we got should be with carry generated from previous stage.

Scanned by CamScanner

$\therefore \quad 0 + \boxed{0} = 0 \quad \therefore S_1 = 0$

**Step III :** $A_2 + B_2 = 0 + 0 = 0$, carry $\boxed{0}$

Here answer '0' should be added with carry generated from previous stage

$S_2 = 0 + \boxed{1} = 1, \quad \therefore S_2 = 1$

**Step IV :** Finally, $A_3 + B_3 = 0 + 0 = 0$, carry $\boxed{0}$

Thus answer is added with previous carry

$S_3 = 0 + \boxed{0} = 0, \quad \therefore S_3 = 0$

$\therefore$ Answer = $(2)_{10} + (3)_{10} = (5)_{10}$

$(0010)_2 + (0011)_2 = (0101)_2$

**Note :** (1) As this was first example, to understand thing clearly, we have also taken into account carry generated even though it is '0'. Normally if carry generated is '0', one should not consider, because ***ANY VALUE ADDED TO ZERO, IS VALUE ITSELF***

(2) Now every time you perform addition refer Table 2.1

**Ex. 2 :** Add $(5)_{10}$ and $(3)_{10}$

**Soln. :**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $(5)_{10}$ | $\Rightarrow$ | 0 | 1 | 0 | 1 | $\Rightarrow$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
| | + | | | | | | | | | |
| $(3)_{10}$ | $\Rightarrow$ | 0 | 0 | 1 | 1 | $\Rightarrow$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| Carry | $\Rightarrow$ | 1 | 1 | 1 | | | | | | |
| | | 1 | 0 | 0 | 0 | $\Rightarrow$ | $S_3$ | $S_2$ | $S_1$ | $S_0$ |

(1) Here adding $A_1$ and $B_1$ with carry is important steps

$A_1 + B_1 = 0 + 1 = 1$, carry '0'

But (Answer) + (Carry) = 1 + $\boxed{1}$ gives sum = 0

and carry = 1 $\therefore$ Finally $S_1 = 0$ and carry = 1

(2) Same way $A_2$ and $B_2$

$\therefore \quad A_2 + B_2 + \text{carry} = \underbrace{1 + 0} + 1$

$\therefore \quad \underbrace{1 + 0} = 1$ with carry '0' you can neglect this as it is zero

But 1 + 1 = 0 with carry 1

↑ ↑

Answer Carry

$\therefore$ Finally $S_2 = 0$ and carry 1

$(5)_{10} + (3)_{10} = (8)_{10}$ $(0101)_2 + (0011)_2 = (1000)_2$

Scanned by CamScanner

**Soln. :**

$$\begin{array}{cccccc} & & 1 & 1 & 0 & 0 \\ + & & 0 & 1 & 0 & 0 \\ \text{Carry} & \boxed{1} & \boxed{1} & & & \\ \downarrow & & & & & \\ 1 & 0 & 0 & 0 & 0 & \end{array}$$

$\therefore \quad (1100)_2 + (0100)_2 = (10000)_2$

**Ex. 4 :** Add two binary numbers $(11000)_2$ and $(10101)_2$.

**Soln. :**

$$\begin{array}{cccccc} & 1 & 1 & 0 & 0 & 0 \\ + & 1 & 0 & 1 & 0 & 1 \\ \text{Carry} \rightarrow & \boxed{1} & & & & \\ \downarrow & & & & & \\ 1 & 0 & 1 & 1 & 0 & 1 \end{array}$$

$(11000)_2 + (10101)_2 = (101101)_2$

**Ex. 5 :** Add $(10110)_2 + (0111)_2$

**Soln. :** $(10110)_2 = (22)_{10}$, $(0111)_2 = (7)_{10}$

$$\begin{array}{lcccccccccc} (22)_{10} & \Rightarrow & 1 & 0 & 1 & 1 & 0 & \Rightarrow & A_4 & A_3 & A_2 & A_1 & A_0 \\ + & & + & & & & & & & & & & \\ (7)_{10} & \Rightarrow & & 0 & 1 & 1 & 1 & \Rightarrow & 0 & B_3 & B_2 & B_1 & B_0 \\ & & \text{carry} \rightarrow & \boxed{1} & \boxed{1} & & & & & & & & \\ \hline (29)_{10} & & 1 & 1 & 1 & 0 & 1 & & S_4 & S_3 & S_2 & S_1 & S_0 \end{array}$$

$\therefore$ **Ans.** $(11101)_2 = (29)_{10}$

Here, $A_1 + B_1 = 1 + 1 = 0$ with carry 1

$\therefore \quad A_2 + B_2 + \text{Carry} = \underbrace{1 + 1}_{} + 1$

$= 0 + 1 = 1 = S_2$

$\downarrow$

(with carry '1')

$\therefore \quad$ Sum $= 1 = S_2$, carry $= 1$

**Ex. 6 :** Add two binary numbers :

$(1\ 1\ 01\ .\ 1\ 0\ 1)_2$ and $(0\ 1\ 0\ 1\ .\ 1\ 0\ 0)$.

**Soln. :** This part contains, fractional part also. The basic is going to remain same.

$$\begin{array}{cccccccccc} & 1 & 1 & 0 & 1 . & 1 & 0 & 1 & \Rightarrow & (13.625)_{10} \\ + & & & & & & & & & + \\ & 0 & 1 & 0 & 1 . & 1 & 0 & 0 & \Rightarrow & (5.5)_{10} \\ \text{Carry} \rightarrow & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & & & & & \\ \hline \underbrace{1}_{\text{Final carry}} & \underbrace{0 & 0 & 1 & 1 & 0 & 0 & 1}_{\text{answer}} & & (19.125)_{10} \end{array}$$

Scanned by CamScanner

$\therefore$ Total answer is $(1\,0\,0\,1\,1\,.\,0\,0\,1)_2 = (19.125)_{10}$

## 2.3 Binary Subtraction :

The rules for binary subtractor is given in Table 2.2. Table shows two, single bit binary subtraction.

| $A_0$ | | $B_0$ | | Difference | Borrow |
|---|---|---|---|---|---|
| 0 | – | 0 | = | 0 | 0 |
| 0 | – | 1 | = | 1 | 1 |
| 1 | – | 0 | = | 1 | 0 |
| 1 | – | 1 | = | 0 | 0 |

$$\begin{array}{rl} & A_0 \\ - & B_0 \\ \hline \text{(Borrow)} & \text{Difference} \end{array}$$

**Table 2.2**

As shown in Table 2.2, we have $A_0$, $B_0$ bit. The outputs are ***Difference*** and ***borrow***. Borrow is borrow from next bit and normally borrow is subtracted from next bit. (as you do in decimal)

(a) $A_0 = 0, B_0 = 0, \therefore A_0 - B_0 = 0 - 0 = 0$. Therefore difference = 0, borrow = 0.

(b) $A_0 = 0, B_0 = 1, \therefore A_0 - B_0 = 0 - 1 = 1$. Therefore difference 1 and borrow = 1

Just as in decimal number we borrow 10 if number being subtracted is greater than the other number, as the base is 10. Here we borrow 2, as base of binary is 2. Therefore as $A_0 - B_0 = 0 - 1$, we borrow 2 which generates ***borrow bit*** = 1 and 2 – 1 gives answer 1 and therefore difference is also 1.

(c) $A_0 = 1, B_0 = 0, \therefore A_0 - B_0 = 1 - 0 = 1$, Therefore difference = 1, borrow = 0

(d) $A_0 = 1, B_0 = 1, \therefore A_0 - B_0 = 1 - 1 = 0$, Therefore difference = 0, borrow = 0

Let's solve some examples based on same.

**Ex. 7 :** Perform $(3)_{10} - (2)_{10}$

**Soln. :**

$(3)_{10} \Rightarrow 0\ 0\ 1\ 1 \Rightarrow A_3\ A_2\ A_1\ A_0$

$(2)_{10} \Rightarrow 0\ 0\ 1\ 0 \Rightarrow B_3\ B_2\ B_1\ B_0$

Borrow → [0] [0] [0] [0]

↓

Final Borrow 0 0 0 1 $\Rightarrow$ $D_3\ D_2\ D_1\ D_0$

(1) $D_0 = A_0 - B_0 = 1 - 0 = 1$, borrow = 0

(2) $D_1 = A_1 - B_1 - 0 = 1 - 1 = 0$, borrow = 0

(3) $D_2 = A_2 - B_2 - 0 = 0 - 0 = 0$, borrow = 0

(4) $D_3 = A_3 - B_3 - 0 = 0 - 0 = 0$, borrow = 0

$\therefore (3)_{10} - (2)_{10} = (1)_{10}$

$\therefore$ **Ans. :** $= (0001)_2$

Scanned by CamScanner

**Soln. :** $(7)_{10} = (0\ 1\ 1\ 1)_2$, $(5)_{10} = (0101)_2$

| | | | | | |
|---|---|---|---|---|---|
| | | 0 | 1 | 1 | 1 |
| | − | 0 | 1 | 0 | 1 |
| Binary → | [0] | [0] | [0] | [0] | |
| | [0] | 0 | 0 | 1 | 0 |

↑ borrow

$\therefore (7)_{10} - (5)_{10} = (2)_{10}$

$\therefore$ **Ans. :** $(\underset{\uparrow}{0}\ 0\ 0\ 1\ 0)_2$

Final borrow

**Ex. 9 :** Subtract $(0101)_2$ from $(0110)_2$

**Soln :**

**Note :** Now onwards borrow = 0, will not be specified as, if you subtract 0 from any number, th answer is number itself

| MSB | | | LSB | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | ⇒ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
| 0 | 1 | 0 | 1 | ⇒ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| borrow → | | | [-1] | | | | [-1] | |
| [0] | 0 | 0 | 0 | 1 | | $D_3$ | $D_2$ | $D_1$ | $D_0$ |

↑ Final borrow

**Step I :** Start from LSB bit

**Step II :** $A_0 - B_0 = 0 - 1 = 1$ with borrow = 1

$\therefore D_0 = 1$

**Step III :** $D_1 = A_1 - B_1 - (\text{borrow}) = \underline{1 - 0} - 1$

$= 1 - 1 = 0$ (as $1 - 0 = 1$ with borrow 0)

$\therefore D_1 = 0$

**Step IV :** $A_2 - B_2 = 1 - 1 = 0$, with borrow '0'

**Step V :** $A_3 - B_3 = 0 - 0 = 0$, with borrow '0'

Scanned by CamScanner

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $(9)_{10}$ | $\Rightarrow$ | 1 | 0 | 0 | 1 | $\Rightarrow$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
| $-$ | | | | | | | | | | |
| $(10)_{10}$ | $\Rightarrow$ | 1 | 0 | 1 | 0 | $\Rightarrow$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| borrow | $\rightarrow$ | [-1] | [-1] | [0] | | | | | | |
| | [1] | 1 | 1 | 1 | 1 | | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
| | $\downarrow$ | | | | | | | | | |
| | final borrow | | | | | | | | | |

(a) $A_0 - B_0 = 1 - 0 = 1$ with borrow 0.

(b) $A_1 - B_1$ – (borrow from previous bit) = $\underline{0 - 1} - 0$

$1$ with borrow 1

$= 1 - 0 = 1$, with borrow 1.

(c) $A_2 - B_2$ – (borrow from previous bit)

$= \underbrace{0 - 0}_{0} - 1$

$= 0 - 1 = 1$ with borrow 1.

(d) $A_3 - B_3$ – (borrow from previous bit)

$= \underbrace{1 - 1}_{} - 1$

$= 0 - 1 = 1$ with borrow '1'.

As borrow = 1, answer is negative and it is not in true form. True form means actually it should show $(0001)_2 = (1)_{10}$ or to be specific $(-1)_{10}$.

**Ex. 11 :** Perform $(1011 . 010)_2 - (0110 . 101)_2$.

**Soln. :** This problem deals with fractional part.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 0 | 1 | 1 | 0 | 1 | 0 | $\Rightarrow$ | $(11.25)_{10}$ |
| | $-$ | | | | | | | | | |
| | | 0 | 1 | 1 | 0 | 1 | 0 | 1 | $\Rightarrow$ | $(6.625)_{10}$ |
| borrow $\longrightarrow$ | [0] | [-1] | | | [-1] | | [-1] | | | |
| | $\downarrow$ | | | | | | | | | |
| final borrow | | 0 | 1 | 0 | 0 | 1 | 0 | 1 | | $(4.625)_{10}$ |

**Ex. 12 :** Perform $(10101100)_2 - (00111000)_2$.

**Soln. :**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| $-$ | | | | | | | | |
| | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| borrow $\longrightarrow$ [0] | [-1] | [-1] | [-1] | | | | | |
| $\downarrow$ | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| final borrow | | | | | | | | |

$\therefore$ **Ans. :** $(0\ 1\ 1\ 1\ 0\ 1\ 0\ 0)_2$.

Scanned by CamScanner

## 2.4 Negative Number Representation in Binary System :

Until now, we have worked on binary numbers that were unsigned. Normally unsigned numbers are treated as ***Positive number***. But as you are aware, most numerical computation involves botl., positive and negative numbers. Therefore we should have method for representing ***sign (+/–)*** of the number. For representing signed magnitude numbers we have following methods :

(a) Sign magnitude code (Notation) OR SM representation.

(b) One's complement method (1's complement method).

(c) Two's complement method (2's complement method).

### 2.4.1 Signed Magnitude Code (SM Representation) :

In decimal, sign is normally indicated by + or 0 at the left of the number like $+ 1.414_{10}$, $- 3.514_{10}$. This notation is called signed magnitude notation, because magnitude follows the sign. The generalized format is shown as follows :

Number B = $\underbrace{B_{n-1}}_{\text{sign bit}}$ $\underbrace{B_{n-2} \quad B_{n-3} \quad \ldots \quad B_2 \quad B_1 \quad B_0}_{\text{magnitude } |B| \text{ bits}}$

Let's say for 8 bit the format is

$\underbrace{B_7}_{\text{sign bit 'S'}}$ $\underbrace{B_6 \quad B_5 \quad B_4 \quad B_3 \quad B_2 \quad B_1 \quad B_0}_{\text{Magnitude } |B|}$

Normally, S bit = 0 → denotes positive number

S bit = 1 → denotes negative number.

Let's say represent $(+120)_{10}$ using SM representation.

120 (Magnitude) = $B_6 \ldots\ldots B_0 = (1\ 1\ 1\ 1\ 0\ 0\ 0)_2$

Number is positive ∴ S bit = 0 ∴ $B_7 = 0$

| $B_7$ | $B_6$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| sign 'S' | magnitude | | | | | | |

To change from $(+ 124)_{10}$ to $(-124)_{10}$, simply change $B_7$ from 0 to 1. This process is called 'negation.'

$(-124)_{10} \Rightarrow$

| $B_7$ | $B_6$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| sign 'S' | Magnitude | | | | | | |

... (a)

If we want from $(-124)_{10}$ to + 124, again 'negate' $B_7$ from 1 to 0. So you get original number. The (a) represents $(-124)_{10}$ in sign magnitude but the same binary format is unsigned number system is $(148)_{10}$.

$(1\ 1\ 1\ 1\ 1\ 0\ 0\ 0)_{2SM} = (-124)_{10}$

where 2SM means, 2 is base, SM → signed magnitude

$(1\ 1\ 1\ 1\ 1\ 0\ 0\ 0)_2 = (148)_{10}$ (unsigned number).

Scanned by CamScanner

1 0 0 0

1 0 0 1

1 0 1 0

1 0 1 1

1 1 0 0

1 1 0 1

1 1 1 0

1 1 1 1

Scanned by CamScanner

ve number $\Rightarrow + (2^{n-1} - 1)$

ive number $\Rightarrow - (2^{n-1} - 1)$

t numbers represented will be.

ive number = $+ (2^{n-1} - 1) =$

ve number = $- (2^{n-1} - 1) =$

ows the full range for 4 bits.

| 1's complement notation | | | |
|---|---|---|---|
| $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 |

Scanned by CamScanner

**Conclusion :** Observing the Table 2.4 we conclude that there is again 'no' unique zero. We have +0 and -0. Therefore the system is not most widely used.

**Ex. 14 :** Find 1's complement of the following numbers
(i) $(-25)_{10}$ (ii) $(-125)_{10}$ (iii) $(-296)_{10}$

**Soln. :** First find out magnitude of 25, 125 and 296

| 2 | 25 | |
|---|---|---|
| | Quo | Rem |
| 2 | 12 | 1 |
| 2 | 6 | 0 |
| 2 | 3 | 0 |
| 2 | 1 | 1 |
| | 0 | 1 |

→ LSB, → MSB

| 2 | 125 | |
|---|---|---|
| | Quo | Rem |
| 2 | 62 | 1 |
| 2 | 31 | 0 |
| 2 | 15 | 1 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |
| | 0 | 1 |

→ LSB, → MSB

| 2 | 296 | |
|---|---|---|
| | Quo | Rem |
| 2 | 148 | 0 |
| 2 | 74 | 0 |
| 2 | 37 | 0 |
| 2 | 18 | 1 |
| 2 | 9 | 0 |
| 2 | 4 | 1 |
| 2 | 2 | 0 |
| 2 | 1 | 0 |
| | 0 | 1 |

→ LSB, → MSB

$(25)_{10} = (11001)_2$ $(125)_{10} = (1111101)_2$ $(296)_{10} = (100101000)_2$

∴ 1's complement of $(25)_{10}$ = $(00110)_2$ ⇒ $(-25)_{10}$
1's complement of $(125)_{10}$ = $(0000010)_2$ ⇒ $(-125)_{10}$
1's complement of $(296)_{10}$ = $(011010111)_2$ ⇒ $(-296)_{10}$

If we represent the numbers in 12 bit standard then,

$(25)_{10} = (0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 1\ \ 1\ 0\ 0\ 1)_2$
$(125)_{10} = (0\ 0\ 0\ 0\ \ 0\ 1\ 1\ 1\ \ 1\ 1\ 0\ 1)_2$
$(296)_{10} = (0\ 0\ 0\ 1\ \ 0\ 0\ 1\ 0\ \ 1\ 0\ 0\ 0)_2$

} Unused bits are represented as 0 since any number preceded by 0 does not affect its magnitude

∴ 1's complement of $(25)_{10}$ = $(1\ 1\ 1\ 1\ \ 1\ 1\ 1\ 0\ \ 0\ 1\ 1\ 0)_2 \Rightarrow (-25)_{10}$
1's complement of $(125)_{10}$ = $(1\ 1\ 1\ 1\ \ 1\ 0\ 0\ 0\ \ 0\ 0\ 1\ 0)_2 \Rightarrow (-125)_{10}$
1's complement of $(296)_{10}$ = $(1\ 1\ 1\ 0\ \ 1\ 1\ 0\ 1\ \ 0\ 1\ 1\ 1)_2 \Rightarrow (-296)_{10}$

**Note :** To get the original number, we take 1's complement again. This is obvious as complementing a number twice is the original number itself.

**Ex. 15 :** Find 1's complement of $(-1000)_{10}$.

**Soln. :** First find out binary for $(1000)_{10}$.

| | 1000 | |
|---|---|---|
| | **Quotient** | **Remainder** |
| 2 | 500 | 0 |
| 2 | 250 | 0 |
| 2 | 125 | 0 |
| 2 | 62 | 1 |
| 2 | 31 | 0 |
| 2 | 15 | 1 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |
| | 0 | 1 |

→ MSB

$(1000)_{10} = (1111101000)_2$
In 12 bit we get
$(1000)_{10} = (001111101000)_2$

Scanned by CamScanner

$(+1000)_{10} = (0\,0\,1\,1 \quad 1\,1\,1\,0 \quad 1\,0\,0\,0)_2$

1's complement $= (1\,1\,0\,0 \quad 0\,0\,0\,1 \quad 0\,1\,1\,1)_2 \Rightarrow (-1000)_2$.

### 2.4.3 2's Complement Notation (2C Notation) :

In 1C code we have +0 and –0. Let's say if we add 6 to 0 i.e. $(0)_{10} + (6)_{10} = (6)_{10}$.

If we take all 0 form of zero we get answer

$(0\,0\,0\,0)_2 + (0\,1\,1\,0)_2 = (0\,1\,1\,0)_2 \Rightarrow (6)_{10}$

But if we take all 1 form of zero we get,

$(1\,1\,1\,1)_2 + (0\,1\,1\,0)_2 = (0\,1\,0\,1)_2 \Rightarrow (5)_{10}$

Five ! here 'one' must be added to correct the answer. This will unnecessary increase hardware. Therefore we require one code which provides us ***unique*** zero and this we get using 2's complement notation (2C code).

2's complement of a number = 1's complement of number + 1

For example, take $(+6)_{10}$ = $(0110)_2$

1's complement of $(6)_{10}$ = $(1001)_2$

2's complement of $(6)_{10}$ = 1's complement of $(6)_{10}$ + 1

$$\begin{array}{rl} = & 1\;0\;0\;1 \\ + & \underline{\qquad\quad 1} \\ & (1\,0\;1\,0)_2 \Rightarrow (-6)_{10} \text{ in 2C code.} \end{array}$$

Let's take '0'. Represent '0' in 2C form.

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

1's complement of $(0)_{10} = (1\,1\,1\,1)_2$

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

$$\begin{array}{lccccc} & & 1 & 1 & 1 & 1 \\ & & & & & 1 \\ \text{Carry} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} \\ \hline & & 0 & 0 & 0 & 0 \end{array}$$

ignore final carry

$\therefore$ $(+0) = (0000)_2$ and 2's complement of $(0)_{10} = (0000)_2$

$\therefore$ $(+0)_{10}$ and $(-0)_{10}$ values are same.

$\therefore$ 2's complement has unique zero.

$\therefore$ this code is most widely used.

### Range of representing number in 2's complement method :

In 2's complement method binary number of 'n' bits represent

maximum positive numbers $\Rightarrow + (2^{n-1} - 1)$

maximum negative numbers $\Rightarrow - (2^{n-1})$

$\therefore$ for 4 bit binary number

maximum positive number $= + (2^{n-1} - 1) = + (2^{4-1} - 1) = + (8 - 1) = (+7)_{10}$

maximum negative number $= - (2^{n-1}) = - (2^{4-1}) = - (2^3) = (-8)_{10}$

Scanned by CamScanner

We will see total number for 4 bits, in Table 2.5.

| Decimal Number | Binary Number | | | | |
|---|---|---|---|---|---|
| | $B_3$ | $B_2$ | $B_1$ | $B_0$ | |
| $+7 = (2^3 - 1)$ | 0 | 1 | 1 | 1 | positive numbers |
| + 6 | 0 | 1 | 1 | 0 | |
| + 5 | 0 | 1 | 0 | 1 | |
| + 4 | 0 | 1 | 0 | 0 | |
| + 3 | 0 | 0 | 1 | 1 | |
| + 2 | 0 | 0 | 1 | 0 | |
| + 1 | 0 | 0 | 0 | 1 | |
| 0 | 0 | 0 | 0 | 0 | unique zero |
| − 1 | 1 | 1 | 1 | 1 | negative numbers |
| − 2 | 1 | 1 | 1 | 0 | |
| − 3 | 1 | 1 | 0 | 1 | |
| − 4 | 1 | 1 | 0 | 0 | |
| − 5 | 1 | 0 | 1 | 1 | |
| − 6 | 1 | 0 | 1 | 0 | |
| − 7 | 1 | 0 | 0 | 1 | |
| $-7 = -(2^3)$ | 1 | 0 | 0 | 0 | |

**Table 2.5**

**Ex. 16 :** Find 2's complement of following numbers :

(i) $(-37)_{10}$, (ii) $(-51)_{10}$, (iii) $(-130)_{10}$.

**Soln. :** First we have to find out binary equivalent of magnitude.

(i)

| | Quo. | Rem. | |
|---|---|---|---|
| 2 | 37 | | |
| 2 | 18 | 1 | LSB |
| 2 | 9 | 0 | |
| 2 | 4 | 1 | |
| 2 | 2 | 0 | |
| 2 | 1 | 0 | |
| | 1 | | MSB |

(ii)

| | Quo. | Rem. | |
|---|---|---|---|
| 2 | 51 | | |
| 2 | 25 | 1 | LSB |
| 2 | 12 | 1 | |
| 2 | 6 | 0 | |
| 2 | 3 | 0 | |
| 2 | 1 | 1 | |
| | 1 | | MSB |

(iii)

| | Quo. | Rem. | |
|---|---|---|---|
| 2 | 130 | | |
| 2 | 65 | 0 | LSB |
| 2 | 32 | 1 | |
| 2 | 16 | 0 | |
| 2 | 8 | 0 | |
| 2 | 4 | 0 | |
| 2 | 2 | 0 | |
| 2 | 1 | 0 | |
| | 1 | | MSB |

$(37)_{10} = (1\ 0\ 0\ 1\ 0\ 1)_2$ $(51)_{10} = (1\ 1\ 0\ 0\ 1\ 1)_2$ $(130)_{10} = (1\ 0\ 0\ 0\ 0\ 0\ 1\ 0)_2$

**8 bit representation of three numbers**

$(37)_{10} = (0\ 0\ 1\ 0\ 0\ 1\ 0\ 1)_2$ $(51)_{10} = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)_2$ $(130)_{10} = (1\ 0\ 0\ 0\ 0\ 0\ 1\ 0)_2$

**(a) 2's complement of $(37)_{10}$ = 1's complement of $(37)_{10}$ + 1**

$$= 1\ 1\ 0\ 1\ \ 1\ 0\ 1\ 0 \rightarrow \text{1's complement of } (37)_{10}$$

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

$$1\ 1\ 0\ 1\ \ 1\ 0\ 1\ 1 \rightarrow (-37)_{10}$$

Scanned by CamScanner

$\therefore$ 2's complement of $(37)_{10}$ $= (-37)_{10} = (1\ 1\ 0\ 1\ 1\ 0\ 1\ 1)_2$

(b) Same way,

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1's complement of $(51)_{10}$ | $\Rightarrow$ | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| + 1 | $\Rightarrow$ | + | | | | | | | 1 |
| 2's complement of $(51)_{10}$ | | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 $\rightarrow (-51)_{10}$ |

$\therefore$ $(-51)_{10} = (1\ 1\ 0\ 0\ 1\ 1\ 0\ 1)_2$ in 2's complement notation.

(c) 1's complement of $(-130)_{10}$ is $(0111\ 1101)_2 = (-130)_{10}$ in 1's complement. As seen here even though magnitude fits, the representation is illegal. This is because the smallest negative number representable in 8 bit i.e. $(-128)_{10}$.

$\therefore$ 8 bits are not sufficient to represent $-130$. $\therefore$ we have to use 12 bits.

$\therefore$ $(+130)_{10} = (0\ 0\ 0\ 0\ \ 1\ 0\ 0\ 0\ \ 0\ 0\ 1\ 0)_2$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1's complement of $(130)_{10}$ | $\Rightarrow$ | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| + 1 | + | | | | | | | | | | | | 1 |
| 2's complement of $(130)_{10}$ | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 $\Rightarrow (-130)_{10}$ |

$(-130)_{10}$ in 2's complement = $(1\ 1\ 1\ 1\ \ 0\ 1\ 1\ 1\ \ 1\ 1\ 1\ 0)_2$.

**Properties of Two's Complement :**

Two's complement numbers have several properties that make them particularly attractive for use in digital circuit.

(1) The first and importnat property is 2C code has a unique representation of zero. (Refer Table 2.5). So plus and minus zero have the same 2C code. This makes it easier to detect a zero result and avoid some arithmetic problem.

(2) In SM notation we saw that negative numbers have sign bit 1, while non-negative numbers have sign bit 0. Thus the sign of a 2C number is always given by its leftmost bit, with same standard convention i.e. 0 = positive and 1 = negative . This conclusion can be done by observing Table 2.5 and secondly from the fact that first step i.e. one's complement changes the sign of the number. In next step i.e. addition of 1 changes the sign of the number again if it generates carry into sign position, this only happens when number is zero.

In Ex. 16 (c) we saw 2C of $(130)_{10}$, the first argument is already given to you that for 8 bit maximum limit is $(-128)$.

$\therefore$ $(-130)_{10}$ cannot fit. Secondly by observing 2's complement string also we conclude i.e.

$(-130)_{10} = (0\ 1\ 1\ 1\ \ 1\ 1\ 0\ 1)_2$

Initially it is zero, so number has to be positive, but we are writing $(-130)_{10}$

$\therefore$ It should contain 1 because of this we switched over to 12 bit.

(3) 2's complement of 2's complemented number is same. It means that you have 2's complemented number, you again find out 2's complement of 2C number. You will get original value.

Say for an example :

$(4)_{10} = (0\ 1\ 0\ 0)_2$

| | | | | | | |
|---|---|---|---|---|---|---|
| 1's complement of $(4)_{10}$ | $\Rightarrow$ | 1 | 0 | 1 | 1 | |
| + 1 | $\Rightarrow$ | + | | | | 1 |
| 2's complement of $(4)_{10}$ | $\Rightarrow$ | 1 | 1 | 0 | 0 | $\Rightarrow (-4)_{10}$ |

Now you perform 2's complement operation on $(-4)_{10}$

Scanned by CamScanner

$(-4)_{10} = (1\ 1\ 0\ 0)_2$

1's complement of $(-4)_{10}$ $\Rightarrow$ 0 0 1 1

\+ 1 $\Rightarrow$ + 1

2's complement of $(4)_{10}$ $\Rightarrow$ 0 1 0 0 $\Rightarrow (+4)_{10}$

(4) One can perform X + Y and X – Y on n bit 2C integers directly using module $2^n$ addition and subtraction.

## 2.5 Binary Subtraction using 1's and 2's Complement Method :

In direct binary subtraction, the implementation becomes complicated. Therefore to perform subtraction easily we represent number to be subtracted, by ***negative number representation***. The basic concept is A – B can be written as A + (–B). So basic building block will be simple addition, but one of the operand is represented by negative numbering i.e. 1's complement (1C) or 2's complement (2C). We have studied both the numbering system.

### 2.5.1 Subtraction using 1's Complement Method :

**Rules** : In this method the minuend is added to one's complement of the subtrahend and the following action is taken :

(i) If an end around carry is generated it is added to the result and the result is positive and in its true form

(ii) If an end around carry is not generated the result is negative and in it's one's complement form.

Let's see some examples.

**Ex. 17** : Perform $(6)_{10} - (4)_{10}$ using 1's complement method.

**Soln.** : $(+6)_{10} - (4)_{10} = (+6)_{10} + (-4)_{10}$

∵ We have to represent $(-4)_{10}$ in negative number representation.

$(+6)_{10} = (0\ 1\ 1\ 0)_2$

$(+4)_{10} = (0\ 1\ 0\ 0)_2$. To find 1's complement simply complement the bits

$\therefore (-4)_0 = (1\ 0\ 1\ 1)_2$

$(+6)_{10} \Rightarrow$ 0 1 1 0

\+ $(-4)_{10} \Rightarrow$ 1 0 1 1

carry → 1 1 1

0 0 0 1

end around + 1

0 0 1 0 $\Rightarrow (0\ 0\ 1\ 0)_2 = (+2)_{10}$

$\therefore$ Answer $(+2)_{10}$ is in true form as end around carry is present.

**Ex. 18** : Perform $(4)_{10} - (6)_{10}$ using 1's complement method.

**Soln.** : $(+4)_{10} - (6)_{10} = (+4)_{10} + (-6)_{10}$

$(+4)_{10} = (0\ 1\ 0\ 0)_2$

$(+6)_{10} = (0\ 1\ 1\ 0)_2$ To find 1's complement of $(+4)_{10}$

↓ ↓ ↓ ↓ complement

$(-6)_{10} = (1\ 0\ 0\ 1)_2$

Scanned by CamScanner

$\therefore$ $(+4)_{10} \Rightarrow 0\ 1\ 0\ 0$

+

$(-6)_{10} \Rightarrow 1\ 0\ 0\ 1$

carry → 0

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

+ → 0

**Ans.** $1\ 1\ 0\ 1$

(carry bit = 0 ; $\therefore$ carry not generated),

As stated in point (ii) when no end around carry is generated answer is in 1's complemented form. Therefore to regain answer in true form complement it.

**Ans. :** $(1\ 1\ 0\ 1)_2 \Rightarrow (-2)_{10}$

$\downarrow\ \downarrow\ \downarrow\ \downarrow$ complement

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

---

**Ex. 19 :** Perform $(-4)_{10} - (6)_{10}$ using 1's complement method.

**Soln. :** $(-4)_{10} - (6)_{10} = (-4)_{10} + (-6)_{10}$

In this case both numbers are negative.

$\therefore$ we have to find out 1's complement of both the numbers.

$(+4)_{10} = (0\ 1\ 0\ 0)_2$ $(+6)_{10} = (0\ 1\ 1\ 0)_2$

$\downarrow\ \downarrow\ \downarrow\ \downarrow$ $\downarrow\ \downarrow\ \downarrow\ \downarrow$

$(-4)_{10} = (1\ 0\ 1\ 1)_2$ $(-6)_{10} = (1\ 0\ 0\ 1)_2$ using 1C

$\therefore$ $(-4)_{10} \Rightarrow 1\ 0\ 1\ 1$

+

$(-6)_{10} \Rightarrow 1\ 0\ 0\ 1$

carry → 1 $\quad$ 1 1

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

+ → 1

$0\ 1\ 0\ 1$

Here rules (i) and (ii) will not work as both the numbers given are ***negative*** . The final answer we have to complement to get in true form.

$(-10)_{10} = (0\ 1\ 0\ 1)_2$ perform 1's complement

$\downarrow\ \downarrow\ \downarrow\ \downarrow$ complement

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

$\therefore$ Even though carry generated we have to complement the final answer.

$\therefore$ ***It is essential that we have to inform hardware that both the numbers are negative .***

Exs. 17, 18 and 19 contains three typical cases :

(i) Positive number and smaller negative number.

(ii) Positive number and larger negative number.

Scanned by CamScanner

(iii) Two negative number.

Now we are going to see case (iv) that is of equal and opposite number.

**Ex. 20 :** Perform $(6)_{10} - (6)_{10}$ using 1's complement method.

**Soln. :** $(6)_{10} = (0\ 1\ 1\ 0)_2$

$\therefore \quad (-6)_{10} = (1\ 0\ 0\ 1)_2$

$(6)_{10} \Rightarrow 0\ 1\ 1\ 0$

$+ \qquad +$

$(-6)_{10} \Rightarrow 1\ 0\ 0\ 1$

carry → [0]

$1\ 1\ 1\ 1$

→ 0

$1\ 1\ 1\ 1$

Carry not generated, therefore answer is in 1's complement form.

$\therefore (-0)_{10} = (1\ 1\ 1\ 1)_2$

$(+0)_{10} = (0\ 0\ 0\ 0)_2$

Basically $(1\ 1\ 1\ 1)_2$ is also correct as in 1's complement $(-0)_{10} = (1\ 1\ 1\ 1)_2$.

## 2.5.2 Binary Subtraction using 2's Complement Method :

**Rules :** (i) Minuend is added to two's complement of subtrahend.

(ii) If carry is generated, result is positive and in its true form.

(iii) If no carry is generated, result is negative and is in its two's complement form.

(iv) Carry is always discarded.

**Note :** Concluding that answer is in true or 2C form can be done, by observing sign bit.

Here also we will study four cases :

(i) Positive number and smaller negative number.

(ii) Positive number and larger negative number.

(iii) Two negative numbers.

(iv) Equal and opposite numbers.

**Ex. 21 :** Perform $(6)_{10} - (4)_{10}$ using 2's complement method.

**Soln. :** $(+6)_{10} - (4)_{10} = (+6)_{10} + (-4)_{10}$.

$(+6)_{10} = (0\ 1\ 1\ 0)_2$, $(+4)_{10} = (0\ 1\ 0\ 0)_2$

For finding 2's complement of $(4)_{10}$ to get $(-4)_{10}$

1's complement of $(4)_{10} \Rightarrow 1\ 0\ 1\ 1$

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

2's complement of $(4)_{10}$ $\quad 1\ 1\ 0\ 0$ $(-4)_{10}$

Scanned by CamScanner

$\therefore$ $(+6)_{10}$ $\Rightarrow$ 0 1 1 0

$+$ $(-4)_{10}$ $\Rightarrow$ 1 1 0 0

carry $\boxed{1}$ $\boxed{1}$

$\uparrow$

Discard carry

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

Sign bit (0 = positive, 1 = negative ). So answer is in true form.

**Ex. 22 :** Perform $(+4)_{10} - (+6)_{10}$ using 2C method.

**Soln. :** $(+4)_{10} - (+6)_{10} = (+4)_{10} + (-6)_{10}$

$\therefore$ $(+4)_{10} = (0100)_2$ , $(+6)_{10} = (0110)_2$

1's complement of $(6)_{10}$ $\Rightarrow$ 1 0 0 1

$+$ 1 $+$ 1

2's complement of $(6)_{10}$ 1 0 1 0 $\Rightarrow (-6)_{10}$

$(+4)_{10}$ $\Rightarrow$ 0 1 0 0

$+$ $+$

$(-6)_{10}$ $\Rightarrow$ 1 0 1 0

Carry $\boxed{0}$

carry not generated

1 1 1 0 $\Rightarrow$ answer

$\uparrow$ sign bit

As carry = 0, answer is in 2's complement form therefore to get final answer, complement $(1\ 1\ 1\ 0)_2$ and add '1'.

1's complement of $(1110)_2$ $\Rightarrow$ 0 0 0 1

$+$ 1 $+$ 1

2's complement of $(1110)_2$ 0 0 1 0 $\Rightarrow (+2)_{10}$

So answer is verified.

$(4)_{10} + (-6)_{10} = (-2)_{10} = (1110)_{2C}$

**Ex. 23 :** Perform $(-4) - (6)_{10}$ using 2C method.

**Soln. :** $(-4)_{10} - (6)_{10} = (-4)_{10} + (-6)_{10}$

$(+4)_{10} = (0\ 1\ 0\ 0)_2$ $(+6)_{10} = (0\ 1\ 1\ 0)_2$

$(-4)_{2C} = (1\ 1\ 0\ 0)_{2\,C}$ $(-6)_{2\,C} = (1\ 0\ 1\ 0)_{2C}$

$(-4)_{10}$ $\Rightarrow$ 1 1 0 0

$+$ $(-6)_{10}$ $\Rightarrow$ 1 0 1 0

carry $\boxed{1}$ 0 1 1 0

Actually our answer should be $(-10)_{10}$, $(-10)_{10}$ doesn't fit within 4 bits. Therefore we have to switch bits.

$(+4)_{10} = (0\ 0\ 1\ 0\ 0)_2$ $(+6)_{10} = (0\ 0\ 1\ 1\ 0)_2$

$(-4)_{2C} = (1\ 1\ 1\ 0\ 0)_2$ $(-6)_{2C} = (1\ 1\ 0\ 1\ 0)_2$

Scanned by CamScanner

$(-4)_{10}$ ⇒ 1 1 1 0 0
+ $(-6)_{10}$ ⇒ 1 1 0 1 0
carry → [1] [1]
1 0 1 1 0
↑
sign bit

Discard carry

As we are aware with the fact that last bit in 2C code represents sign. 1 = Negative, 0 = Positive.

∴ Even though carry generated, but as sign bit = 1, we have to find out 2's complement of $(1\ 0\ 1\ 1\ 0)_2$.

1's complement of $(10110)_2$ ⇒ 0 1 0 0 1
+ 1 + 1
2's complement of $(10110)_2$ 0 1 0 1 0 ⇒ $(+10)_{10}$
↑
positive

---

**Ex. 24** : Perform $(+6) - (+6)_{10}$ using 2C.

**Soln. :** $(+6)_{10} - (+6)_{10} = (+6)_{10} + (-6)_{10}$

∴ $(+6)_{10} = (0\ 1\ 1\ 0)_2$

$(-6)_{10} = (1\ 0\ 1\ 0)_{2C}$

$(+6)_{10}$ ⇒ 0 1 1 0
+ +
$(-6)_{10}$ ⇒ 1 0 1 0
carry → [1] [1] [1]
0 0 0 0

Discard carry. Therefore answer = $(0\ 0\ 0\ 0)_2 = (0)_{10}$

The result is obvious + 0 as expected.

**Note :** While performing 1C or 2C, both, numbers have to have same number of bits.

## 2.6 Binary Multiplication :

The multiplication in binary is done in same manner as the multiplication of decimal numbers. The process is simple, since multiplier digits are always '1' or '0' and so, we are always multiplying by '0' or '1'. First see the rules for multiplication.

**Rule :** Here we are going to multiply two numbers of 'single' bit i.e. $A_0$ and $B_0$. Table 2.6 shows the rules.

Scanned by CamScanner

| $A_0$ | $B_0$ | $A_0 \times B_0$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Table 2.6

$A_0 \times B_0 \Rightarrow A_0 B_0$

The following example shows generalized multiplication for 4 bit number – say $A_3 - A_0$ and $B_3 - B_0$

$$
\begin{array}{llllll}
 & & A_3 & A_2 & A_1 & A_0 \\
\times & & B_3 & B_2 & B_1 & B_0 \\
\hline
 & & A_3 \times B_0 & A_2 \times B_0 & A_1 \times B_0 & A_0 \times B_0 \\
 & A_3 \times B_1 & A_2 \times B_1 & A_1 \times B_1 & A_0 \times B_1 & X \\
 & A_3 \times B_2 & A_2 \times B_2 & A_1 \times B_2 & X & X \\
+ & A_3 \times B_3 & A_2 \times B_3 & A_1 \times B_3 & A_0 \times B_3 & X \\
\hline
 & A_3 \times B_3 & A_3 \times B_2 + A_2 \times B_3 & A_3 \times B_1 + A_2 \times B_2 + A_1 \times B_3 & A_3 \times B_0 + A_2 \times B_1 + A_1 \times B_2 + A_0 \times B_3 & A_2 \times B_0 + A_1 \times B_1 + A_0 \times B_2 \quad A_1 \times B_0 + A_0 \times B_1 \quad A_0 \times B_0
\end{array}
$$

Fig. 2.1

**Note :** (1) X is normally taken as '0'.
(2) The Fig. 2.1 shows, finally addition, while performing addition refer Table 2.1, and take 'carry' into account.

*Let us see some examples.*

**Ex. 25** : Perform multiplication of $(9)_{10}$ and $(11)_{10}$.

**Soln. :** $(9)_{10} = (1001)_2$ $(11)_{10} = (1011)_2$

$$
\begin{array}{llllllll}
 & & & 1 & 0 & 0 & 1 & \\
 & & & 1 & 0 & 1 & 1 & \\
\hline
+ & & & 1\times1 & 0\times1 & 0\times1 & 1\times1 & \Rightarrow \\
 & & 1\times1 & 0\times1 & 0\times1 & 1\times1 & X & \Rightarrow \\
+ & 1\times0 & 0\times0 & 0\times0 & 1\times0 & X & X & \Rightarrow \\
 & 1\times1 & 0\times1 & 0\times1 & 1\times1 & X & X & X \Rightarrow \\
 & & & & & & \text{carry} & \rightarrow
\end{array}
$$

$$
\begin{array}{ccccccc}
 & & 1 & 0 & 0 & 1 & \\
 & 1 & 0 & 0 & 1 & X & \\
0 & 0 & 0 & 0 & X & X & \\
1 & 0 & 1 & X & X & X & X \\
\boxed{1} & \boxed{1} & & & & & \\
\hline
1 & 1 & 0 & 0 & 0 & 1 & 1
\end{array}
$$

$\therefore$ $(9)_{10} \times (11)_{10} = (99)_{10} \Rightarrow (1\,1\,0\,0\,0\,1\,1)_2$.

**Ex. 26 :** Perform $(10110)_2 \times (101)_2$.

**Soln. :** $(10110)_2 = (22)_{10}$ $(101)_2 = (5)_{10}$.

Scanned by CamScanner

$$
\begin{array}{rccccccc}
 & & & 1 & 0 & 1 & 1 & 0 \\
 & & \times & & & 1 & 0 & 1 \\
\hline
 & & & 1 & 0 & 1 & 1 & 0 \\
 & & 0 & 0 & 0 & 0 & 0 & X \\
+ & 1 & 0 & 1 & 1 & 0 & X & X \\
 & & \boxed{1} & & & & & \\
\hline
 & 1 & 1 & 0 & 1 & 1 & 1 & 0
\end{array}
$$

$\therefore \quad (22)_{10} \times (5)_{10} = (110)_{10} = (110\ 1110)_2$

**Note :** It is not essential in multiplication that both the numbers should have equal number of bits. If you want to make equal number you can add zeros.

$(22)_{10} = (1\ 0\ 1\ 1\ 0)_2 \Rightarrow$ 5 bits

$(5)_{10} = (0\ 0\ 1\ 0\ 1)_2 \Rightarrow$ 5 bits by adding zeros.

The answer is going to remain same, even though you add zeros.

**Ex. 27 :** Perform $(11001)_2 \times (110)_2$.

**Soln. :**

$$
\begin{array}{rccccccc}
 & & & 1 & 1 & 0 & 0 & 1 \\
 & & \times & & & 1 & 1 & 0 \\
\hline
 & & & 0 & 0 & 0 & 0 & 0 \\
 & & 1 & 1 & 0 & 0 & 1 & X \\
 & 1 & 1 & 0 & 0 & 1 & X & X \\
\boxed{1} & \boxed{1} & & & & & & \\
\hline
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{array}
$$

$\therefore$ **Ans. :** $(10010110)_2$.

**Ex. 28 :** Perform (11010 . 101) × (10010 . 1).

**Soln. :** In this example, fractional part is also involved. Still procedure is going to remain same. After getting multiplied and added you calculate position of ***fractional point*** in both the numbers and as you do in decimal system, you put ***dot*** in the final answer.

$$
\begin{array}{rccccccccc}
 & & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \quad \Rightarrow 26.625 \\
 & \times & 1 & 0 & 0 & 1 & 0 & 1 & & \times 18.5 \\
\hline
 & & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \quad \Rightarrow 492.5625 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X \\
 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & X \quad X \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X \quad X \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X \quad X \quad X \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & X & X \quad X \quad X \quad X \\
 & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & & \boxed{1} & \\
\hline
1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \quad 0 \quad 0 \quad 1
\end{array}
$$

Scanned by CamScanner

In the first number we have 3 bits after fractional point. In $2^{nd}$ we have 1 bit after fractional point.
Therefore total 3 + 1 = 4

Now start from right of the final answer, calculate 4 bits and put fractional point (dot).

$\therefore$ Answer = $(1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0.\underbrace{1\ 0\ 0\ 1}_{4\ \text{bits}})_2 = (492.5625)_{10}$

(June '99, 2 Marks)

**Ex. 29** : Perform $(10110.01)_2 \times (101.1)_2$

**Soln.** :

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| | × | | | 1 | 0 | 1 | 1 | |
| | | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| | 1 | 0 | 1 | 1 | 0 | 0 | 1 | X |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | X | X |
| | 1 | 1 | 1 | 1 | 1 | | | |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |

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

## 2.7 Binary Division :

The process for dividing one binary number (dividend) by another (divisor) is same as that we follow in decimal number system. The method implemented is ***subtract the divisor from dividend***. If no borrow is generated set quotient to logic '1' else to logic '0'.

Let's see some examples.

**Ex. 30** : Perform $(1111)_2 \div (10)_2$

**Soln.** : $(1111)_2 = (15)_{10}$

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

```
 10 ) 1 1 1 1 ( 1 1 1  → Quotient
      1 0
      -----
      0 1 1
        1 0
        -----
        0 1 1
          1 0
          -----
          0 1   → Remainder
```

$\therefore$ **Quotient** = $(111)_2 = (7)_{10}$

**Remainder** = $(01)_2 = (1)_2$.

**Ex. 31** : Perform $(30)_{10} \div (3)_{10}$

Scanned by CamScanner

**Soln. :** $(30)_{10} = (11110)_2$, $(3)_{10} = (00011)_2$.

$\therefore$ 11 ) 1 1 1 1 0 ( 1 0 1 0

1 1

added zero

0 0 1

Here 001 is less than 11

$\therefore$ we have to take one more bit down and put '0' in quotient

$\therefore$ we get

0 0 1 1

1 1

0 0 0 0

Finally, last '0' in dividend that we put in Quotient (shown by dotted line).

$\therefore$ Quotient = $(1\,0\,1\,0)_2$

Remainder = $(0\,0\,0\,0)_2$

---

**Ex. 32 :** Perform 1101001 + 101. **(Dec. 99, 2 Marks)**

**Soln. :** $(1101001)_2 = (105)_{10}$ $(101)_2 = (5)_{10}$

$\therefore$ 1 0 1 ) 1 1 0 1 0 0 1 ( 1 0 1 0 1

− 1 0 1

1

If we bring only 1 bit down we get 0011

0 0 1 1 0

which is less than 101. $\therefore$ we take two bit

1 0 1

and add '0' in quotient.

0 0 1 0 1

here again we take two bits down and

1 0 1

add '0' in Quotient.

0 0 0

$\therefore$ Quotient = $(10101)_2$, Remainder = $(00000)_2$.

---

**Ex. 33 :** Perform $(9)_{10} + (3)_{10}$.

**Soln. :** $(9)_{10} = (1001)_2$, $(3)_{10} = 0011$.

1 1 ) 1 0 0 1 ( 1 1

− 0 1 1

Here we have two bits '10' in dividend and '10' is smaller than '11' i.e. divisor therefore we take 3 bits of dividend.

0 0 1 1

− 1 1

0 0

$\therefore$ Quotient = $(11)_2 = (3)_{10}$

Remainder = $(0)_2 = (0)_{10}$.

Scanned by CamScanner

**Soln. :**

```
0 1 1 ) 1 0 0 0 0 0 1 ( 1 0 1 0 1
      - 0 1 1 ↓ ↓
        0 0 1 0 0
          0 1 1 ↓ ↓
          0 0 1 0 1
            0 1 1
            0 1 0  → Remainder
```

**Quotient** = $(10101)_2 = (21)_{10}$

**Remainder** = $(010)_2 = (2)_{10}$

---

***Ex. 35 :*** Perform $(10)_{10} \div (4)_{10}$

**Soln. :** $(10)_{10} = (1010)_2$, $(4)_{10} = (0100)_2$

```
1 0 0 ) 1 0 1 0 ( 1
      - 1 0 0
        0 0 1 0
```

→ Here '10' is less than '100'. Therefore same as in decimal number we have to imagine fraction point and take '0' i.e. next bit.

$\therefore$

```
1 0 0 ) 1 0 1 0 . 0 ( 1 0 1
      - 1 0 0 ↓   ↓
        0 0 1 0   0
          1 0     0
          0 0 . 0
```

*Now, here remainder* = $(0000)_2 = (0)_{10}$

About quotient we have $(101)_2$ but how to put fractional point. The concept is same as that of decimal. Presently in dividend we have *one* bit after fraction point.

$\therefore$ *Quotient* = $(10.1)_2 = (2.5)_{10}$

$(10)_{10} \div (4)_{10} = (2.5)_{10}$

---

***Ex. 36 :*** Perform 1101001.101 ÷ 101. **(June 99, 2 Marks)**

**Soln. :**

```
1 0 1 ) 1 1 0 1 0 0 1 . 1 0 1 ( 1 0 1 0 1 . 0 0 1
      - 1 0 1 ↓ ↓
        0 0 1 1 0
          1 0 1 ↓ ↓
          0 0 1 0 1
            1 0 1 ↓ ↓ ↓
            0 0 0 1 0 1
                  1 0 1
                  0 0 0
```

$\therefore$ **Quotient** = $(10101.001)_2$

Scanned by CamScanner

## 2.8.1 Addition : ...mber System :

**Rule :** Since the base of the number system is '8', a *Carry* is carried forward if partial sum exceeds '8'.
The required digit of partial sum = Partial sum – 16.

Let's take one example,

Add $(6)_8$ and $(7)_8$.

$$\begin{array}{r} (6)_8 \\ +\ (7)_8 \\ \hline (13)_{10} \end{array}$$

$\therefore$ $(13)_{10} - (8)_{10} = 5$ with carry = 1.

$$\begin{array}{r} \therefore\ (6)_8 \\ +\ (7)_8 \\ \text{carry } \boxed{1} \\ \hline 5 \end{array}$$

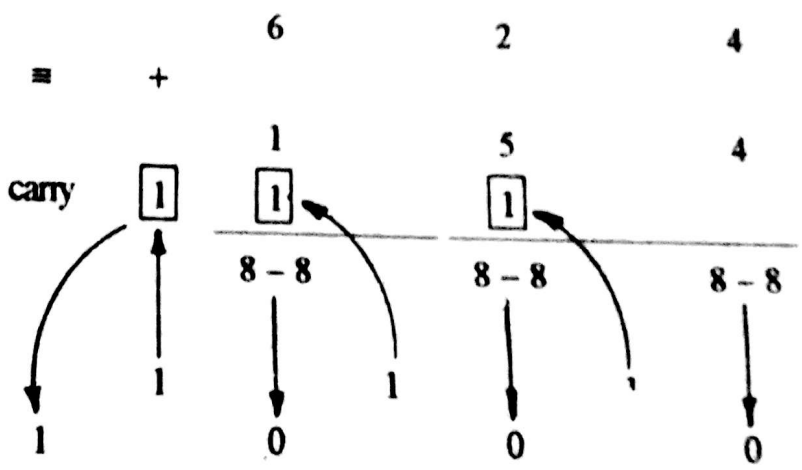
$\therefore$ Answer = $(15)_8$.

**Ex. 37 :** Perform $(624)_8 + (154)_8$.

**Soln. :**

$$\begin{array}{r} 6\ \ 2\ \ 4 \\ +\ \ \ \ \ \\ 1\ \ 5\ \ 4 \\ \text{carry } \boxed{1}\ \boxed{1}\ \boxed{1} \\ \hline (1\ \ 0\ \ 0\ \ 0)_8 \end{array}$$

$\equiv$



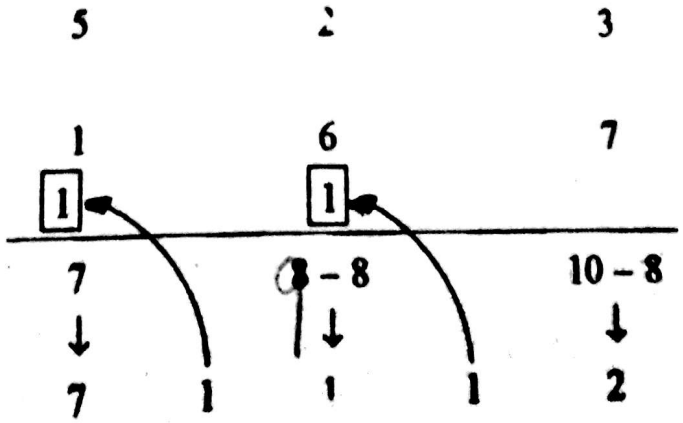

$\therefore$ $(624)_8 + (154)_8 = (1000)_8$

**38 :** Perform $(523)_8 + (167)_8$.

**n. :**

$$\begin{array}{r} 5\ \ 2\ \ 3 \\ +\ \ \ \ \ \\ 1\ \ 6\ \ 7 \\ \boxed{1}\ \boxed{1} \\ \hline 7\ \ 1\ \ 2 \end{array}$$

$\equiv$



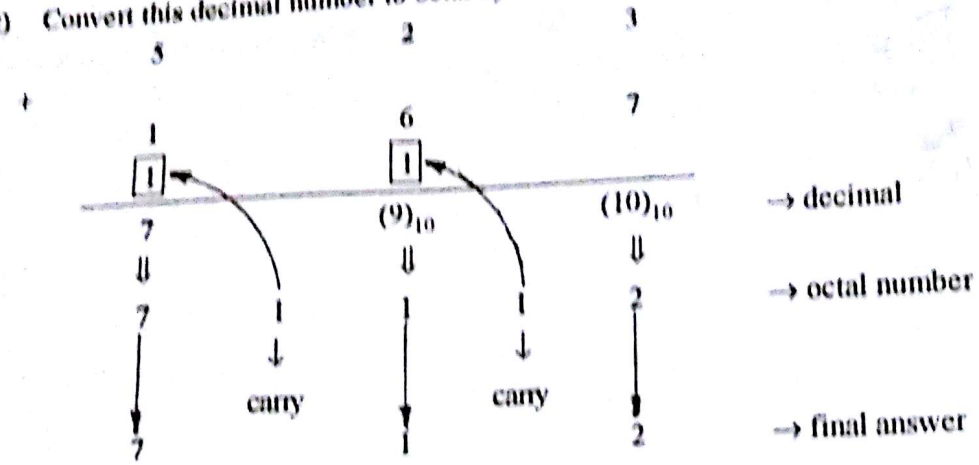

... (712) = Answer

Scanned by CamScanner

If you can do this, no problem. Otherwise you can follow one more simpler method.

(1) You add two octal number i.e. purely decimal.

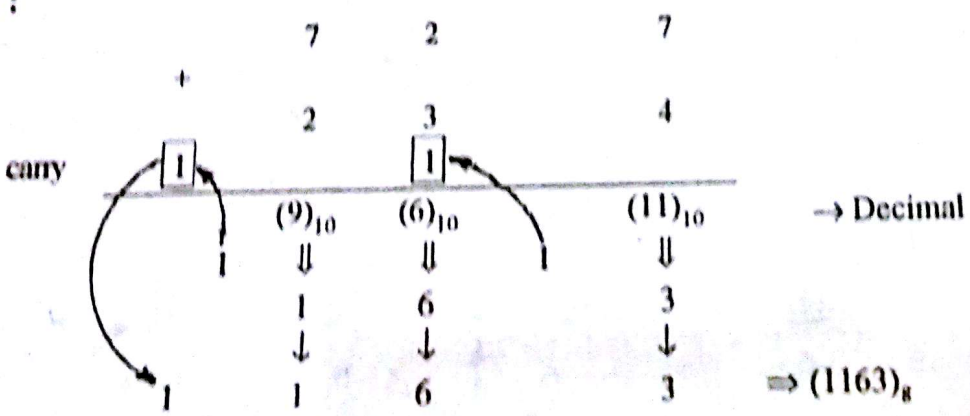
(2) Convert this decimal number to octal by referring Table 2.2, Chapter – 1.




$(523)_8 + (167)_8 = (712)_8$.

**Ex. 39 :** Add $(727)_8$ with $(234)_8$.

**Soln. :**




$\therefore \quad (727)_8 + (234) = (1163)_8$.

There is one more method to verify your answer. The method is

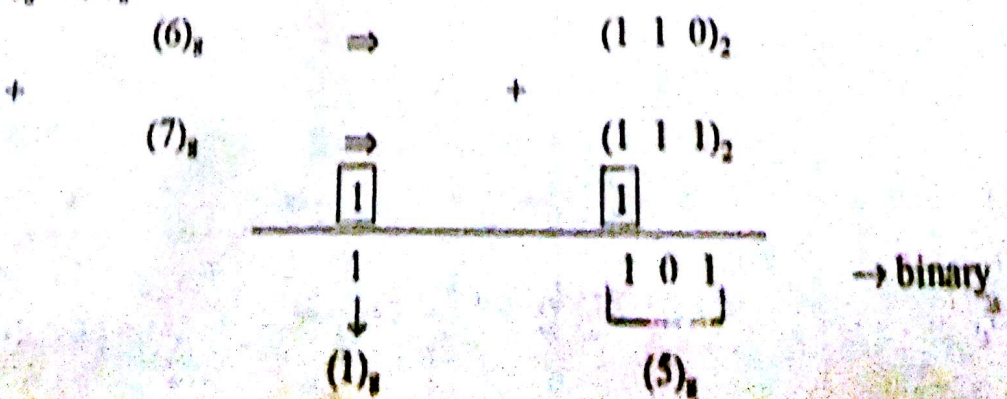
***(1) Convert octal to its equivalent binary.***

***(2) Perform binary addition (Refer Table 2.1 for the same).***

***(3) If you want answer back in octal, convert binary to octal.***

Let's see same examples that we have solved just now.

(i) $(6)_8 + (7)_8$




Ans. : (15)8

Scanned by CamScanner

(ii) $(624)_8 + (154)_8$

$(624)_8 \Rightarrow$ 1 1 0 | 0 1 0 | 1 0 0

\+ \+

$(154)_8$ 0 0 1 | 1 0 1 | 1 0 0

Carry: [1] | [1] [1] [1] | [1] [1] [1]

1 | 0 0 0 | 0 0 0 | 0 0 0 ⇒ binary

1 | 0 | 0 | 0 → octal

∴ **Ans. :** $(1000)_8$.

(iii) $(523)_8 + (167)_8$

$(523)_8 \Rightarrow$ 1 0 1 | 0 1 0 | 0 1 1

\+ \+

$(167)_8$ 0 0 1 | 1 1 0 | 1 1 1

carry: [1] [1] | [1] [1] | [1] [1]

1 1 1 | 0 0 1 | 0 1 0 → binary

⇓ ⇓ ⇓

$(7)_8$ $(1)_8$ $(2)_8$

∴ **Ans. :** $(712)_8$.

(iv) $(727)_8 + (234)_8$

$(727)_8 \Rightarrow$ 1 1 1 | 0 1 0 | 1 1 1

\+ \+

$(234)_8$ 0 1 0 | 0 1 1 | 1 0 0

carry: [1] | [1] | [1] [1] [1]

1 | 0 0 1 | 1 1 0 | 0 1 1 → Binary

↓ ⇓ ⇓ ⇓

1 | 1 | 6 | 3 → Octal

∴ **Ans. :** $(1163)_8$.

## 2.8.2 Subtraction :

Now we are going to perform octal subtraction.

(i) By direct method.

(ii) By converting octal to binary and then performing subtraction and then back converting answer from binary to octal.

**(i) By Direct Method :**

**Rule :** If the Minuend is less than subtrahend, we borrow 8 and carry '1'.

Say for an example, perform $(751)_8 - (273)_8$

[8] [8] ⟶ borrow

7 5 1 $(13)_{10}$ $(9)_{10}$

−

2 7 3 $(3)_{10}$ $(8)_{10}$

[1] [1] carry

4 5 6

Scanned by CamScanner

**Step I :** While performing 1 – 3 we found '1' is smaller, therefore we added 8 in 1. i.e. borrow from previous digit. At the same time we added carry '1' in next digit of subtrahend i.e. 7, we get 7 + 1 = 8.

Now, (8 + 1) – 3 = 9 – 3 = 6.

**Step II :** Now we have to perform 5 – (7 + 1) where 1 is carry. Again we found that 5 is less than 8, therefore we borrowed 8 from next digit so we got (8 + 5) = 13. At the same time we added carry '1' in next digit of subtracted i.e. 2. Therefore we got (2 + 1) = 3.

Finally, we performed (8 + 5) – (7 + 1) = (13) – (8) = 5.

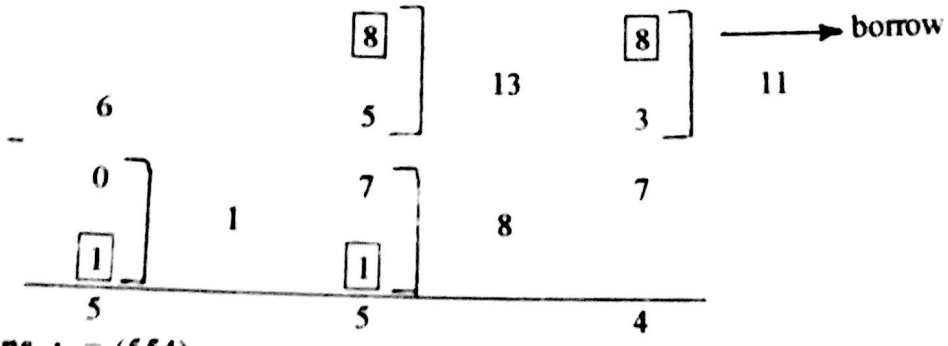
**Step III :** Finally we have to perform 7 – (2 + 1) = 7 – (3) = 4.

∴ **Ans. :** = $(456)_8$.

---

**Ex. 40 :** Perform $(653)_8 - (77)_8$.

**Soln. :**

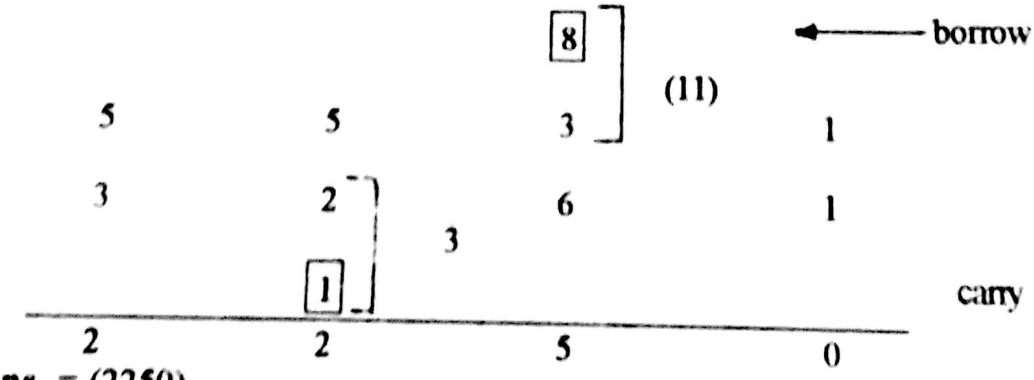



∴ **Ans. :** = $(554)_8$.

---

**Ex. 41 :** Perform $(5531)_8 - (3261)_8 = (?)_8$.

**Soln. :**




∴ **Ans.** = $(2250)_8$.

*(ii) By converting octal to binary. We will take same examples to cross check answer.*

*(a)*

$(751)_8 \Rightarrow$ 1 1 1 1 0 1 0 0 1

–

$(273)_8 \Rightarrow$ 0 1 0 1 1 1 0 1 1

-1 -1 -1 -1 -1 → (borrow)

1 0 0 1 0 1 1 1 0 → binary

4 5 6 → octal

∴ **Ans. :** = $(456)_8$

Scanned by CamScanner

(b) $(653)_8 - (77)_8$

$(653)_8$ $\Rightarrow$ 1 1 0 1 0 1 0 1 1

$-$

$(77)_8$ $\Rightarrow$

1 1 1 1 1 1 1 1

−1 −1 −1 −1 −1 (borrow)

1 0 1 1 0 1 1 0 0 → binary

5 5 4 → octal

∴ **Ans.** : $= (554)_8$

(c) $(5531)_8 - (3261)_8$

$(5531)_8$ $\Rightarrow$ 1 0 1 1 0 1 0 1 1 0 0 1

$-$

$(3261)_8$

0 1 1 0 1 0 1 1 0 0 0 1

−1 −1 −1 → (borrow)

0 1 0 0 1 0 1 0 1 0 0 0 → binary

2 2 5 0 → octal

∴ **Ans.** : $= (2250)_8$

Until now we saw the direct method and binary conversion method to perform subtraction is octal. We can also use 1's complement (1C) and 2's complement (2C) method for subtraction.

Let's see how to find ***negative number*** in octal.

## 2.8.3 1's Complement to Represent Negative Octal Number :

Let's take number $(5)_8$ to represent in 1's complement. the first method is to subtract any given octal number from $(7)_8$. This will directly give you 1's complement.

1's complement of $(5)_8 = (7)_8 - (5)_8 = (2)_8$

2nd method is very simple.

(a) Convert given octal to binary.

(b) Complement binary number.

(c) Convert binary to octal.

$(5)_8 \Rightarrow (1 \quad 0 \quad 1)_2$ ⟶ Step (a)

↓ ↓ ↓ Complement → Step (b)

$(0 \quad 1 \quad 0)_2$ →

⇓

$(2)_8$ ⟶ Step (c)

LIBRARY 743

**Therefore answer from 1st and 2nd method both is matching. Let's perform all previous examples using 1's complement method. I am going to prefer method 1 of finding 1's complement, because you are already aware with 2nd method.**

**(i) Perform $(751)_8 - (273)_8$.**

**Soln. : $(751)_8 - (273)_8 = (751)_8 + (-273)_8$**

Scanned by CamScanner

$(-273)_8$ = 7 7 7
− 2 7 3
$(5\ 0\ 4)_8$ → 1's complement of $(273)_8$

$(751)_8$ + $(-273)_8$ = 7 5 1 + 5 0 4

$(12)_{10}$ $(5)_{10}$ $(5)_{10}$ → decimal

carry → 1 4 5 5

carry end around carry

4 5 6

∴ **Ans.** : = $(456)_8$

---

(ii) Perform $(653)_8 - (77)_8$

**Soln. :**

1's complement of $(77)_8$ = 7 7 7 − 0 7 7 = 7 0 0 ⇒ $(-77)_8$

$(653)_8 - (077)_8 = (653)_8 + (-077)_8$

6 5 3 +

7 0 0

$(13)_{10}$ $(5)_{10}$ $(3)_{10}$ → decimal

$(15)_8$ $(5)_8$ $(3)_8$ → octal

carry 1

5 5 4 ∴ **Ans.** = $(554)_8$

---

(iii) $(5531)_8 - (3261)_8$

**Soln. :** $(5531)_8 + (-3261)_8$

$(-3261)_8$ = 7 7 7 7 − 3 2 6 1 = 4 5 1 6 ⇒ 1's complement of $(3261)_8$

5 5 3 1 → octal +

4 5 1 6 → octal

$(9)_{10}$ $(10)_{10}$ $(4)_{10}$ $(7)_{10}$ → decimal +

$(10)_{10}$ $((1)2)_8$ $(4)_8$ $(7)_8$

carry 1 2 2 4 7

1

Scanned by CamScanner

$$\begin{array}{cccc} & 2 & 2 & 4 & 7 \\ + & & & & 1 \\ \hline & 2 & 2 & 4 & 8 \end{array}$$

→ end around carry

→ decimal

$\therefore$ 2 2 4 $((1)\,0)_8$ (carry generated from LSD is taken to next digit)

↑ carry

$\therefore$ $(2\ 2\ 5\ 0)_8$ = Answer.

### 2.8.4 2's Complement to Represent Negative Octal Number :

2's complement is nothing but 1's complement plus 1. To achieve this we have two methods :

**First :**
(1) Subtract number from 7 7 ... 7. (The numbers of digit 7 depends upon no. of digits of number). Say 4 digits, we have 7777. This is done to achieve 1's complement.
(2) Add 1 to 1's complemented number. You will get 2C code.

**Second :**
(1) Convert octal number to binary.
(2) Perform 1's complement on binary number.
(3) Add 1 to it.
(4) Convert binary to octal.

**Example :** To represent $(723)_8$ in 2's complement form.

**Soln. : Method 1 :**

(i) We have 3 digits, therefore subtract from 777.

$$\begin{array}{cccc} & 7 & 7 & 7 \\ - & 7 & 2 & 3 \\ \hline & 0 & 5 & 4 \end{array}$$

→ 1's complement of $(723)_8$

1's complement of $(723)_8$ ⇒ 0 5 4

\+ +

1 ⇒ 1

2's complement of $(723)_8$ 0 5 5 ⇒ $(-723)_8$ in 2C code

**Method 2 :**

$(723)_8 = (111\ 010\ 011)_2$

1's complement of $(723)_8 = (000\ 101\ 100)_2$

1'C of $(723)_8$ ⇒ 0 0 0 1 0 1 1 0 0

\+ +

1 1

2'C of $(723)_8$ 0 0 0 1 0 1 1 0 1 → binary $(-723)$

⇓ ⇓ ⇓

0 5 5 → $(-723)_8$

Now we are going to perform subtraction using 2's complement method. We are going to see some examples for the same.

Scanned by CamScanner

**Ex. 42 :** Perform $(751)_8 - (273)_8$ using 2's complement method.

**Soln. :** $(751)_8 - (273)_8 = (751)_8 + (-273)_8$

Now represent $(273)_8$ in 2's complement form to get $(-273)_8$

$$\begin{array}{rl} \text{1C of } (273)_8 & = \quad 7\ 7\ 7 \\ & - \quad 2\ 7\ 3 \\ \hline & \quad 5\ 0\ 4 \end{array}$$

$$\begin{array}{rcl} \text{1C of } (273)_8 & \Rightarrow & 5\ 0\ 4 \\ + \quad 1 & \Rightarrow & + \quad 1 \\ \hline \text{2 C of } (273)_8 & \Rightarrow & 5\ 0\ 5 \end{array} \Rightarrow (-273)_8 \text{ in 2C}$$

$$\begin{array}{rcl} (751)_8 & \Rightarrow & 7\ 5\ 1 \\ + & & + \\ (-273)_8 & \Rightarrow & 5\ 0\ 5 \\ \text{carry} & & 1\ 4\ 5\ 6 \end{array} \quad \begin{array}{l} \rightarrow \text{octal directly} \\ \therefore \textbf{Ans.} = (456)_8 \end{array}$$

↑ Ignore carry

---

**Ex. 43 :** Perform $(653)_8 - (77)_8$ in 2C method.

**Soln. :** $(653)_8 - (77)_8 = (653)_8 + (-77)_8$

$$\begin{array}{rl} \text{1 C of } (077)_8 & = \quad 7\ 7\ 7 \\ & - \quad 0\ 7\ 7 \\ \hline & \quad 7\ 0\ 0 \end{array}$$

$$\begin{array}{rcl} \text{1'C of } (077)_8 & \Rightarrow & 7\ 0\ 0 \\ + \quad 1 & + & \quad 1 \\ \hline \text{2'C of } (077)_8 & \Rightarrow & 7\ 0\ 1 \end{array} \Rightarrow (-77)_8 \text{ in 2C}$$

$$\begin{array}{rcl} (653)_8 & \Rightarrow & 6\ 5\ 3 \\ + & & + \\ -(077)_8 & \Rightarrow & 7\ 0\ 1 \\ \text{carry} & \longrightarrow & \boxed{1}\ 5\ 5\ 4 \\ \text{(ignore it)} & \longrightarrow & \end{array}$$

$\therefore$ **Ans.** $= (554)_8$.

---

**Ex. 44 :** Perform $(5531)_8 - (3261)_8$.

**Soln. :** $(5531)_8 - (3261)_8 = (5531)_8 + (-3261)_8$.

$$\begin{array}{rl} \text{1 C of } (3261)_8 & = \quad 7\ 7\ 7\ 7 \\ & - \quad 3\ 2\ 6\ 1 \\ \hline & \quad 4\ 5\ 1\ 6 \end{array}$$

$$\begin{array}{rcl} \text{1 C of } (3261)_8 & \Rightarrow & 4\ 5\ 1\ 6 \\ + \quad 1 & + & \quad 1 \\ \hline \text{2'C of } (3261)_8 & & 4\ 5\ 1\ 7 \end{array} \Rightarrow \text{2C of } (3261)_8$$

Scanned by CamScanner

$(5531)_8$ ⇒ 5 5 3 1

$(-3261)_8$ ⇒ 4 5 1 7

1 1 1 → carry

Ignore this carry

2 2 5 0

∴ **Ans.** = $(2250)_8$.

**Note :** In simple subtraction , 1's complement and 2's complement examples taken are identical so that you can compare three methods.

## 2.8.5 Multiplication :

Now we are going to learn multiplication in octal system. If you directly multiply octal number it becomes bit complicated therefore normal procedure followed is as follows :

**Step I :** Convert octal to binary. (both, multiplier and multiplication).

**Step II :** Perform ***simple binary multiplication*** (Refer Table 2.6).

**Step III :** After performing multiplication, whatever answer you get in binary, convert it to equivalent octal.

Let's see some examples for the same.

**Ex. 45 :** Perform $(12)_8 \times (7)_8$.

**Soln. :**

**Step I :** Convert both octal number to binary :

$(12)_8 = (0\ 0\ 1\ 0\ 1\ 0) = (1\ 0\ 1\ 0)_2$

$(7)_8 = (1\ 1\ 1)_2 = (1\ 1\ 1)_2$

**Step II :** Perform binary multiplication :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | 1 | 0 | 1 | 0 | |
| | | | | 1 | 1 | 1 | |
| | | | 1 | 0 | 1 | 0 | |
| | | 1 | 0 | 1 | 0 | X | |
| | 1 | 0 | 1 | 0 | X | X | |
| 1 | 1 | 1 | | | | | |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | → Binary |
| 1 | 0 | | | 6 | | | → $(106)_8$ |

To cross check.

$(12)_8 \times (7)_8 = (1 \times 8^1 + 2 \times 8^0) \times [7 \times 8^0] = (10)_{10} \times (7)_{10} = (70)_{10}$

Consider result i.e. $(106)_8$

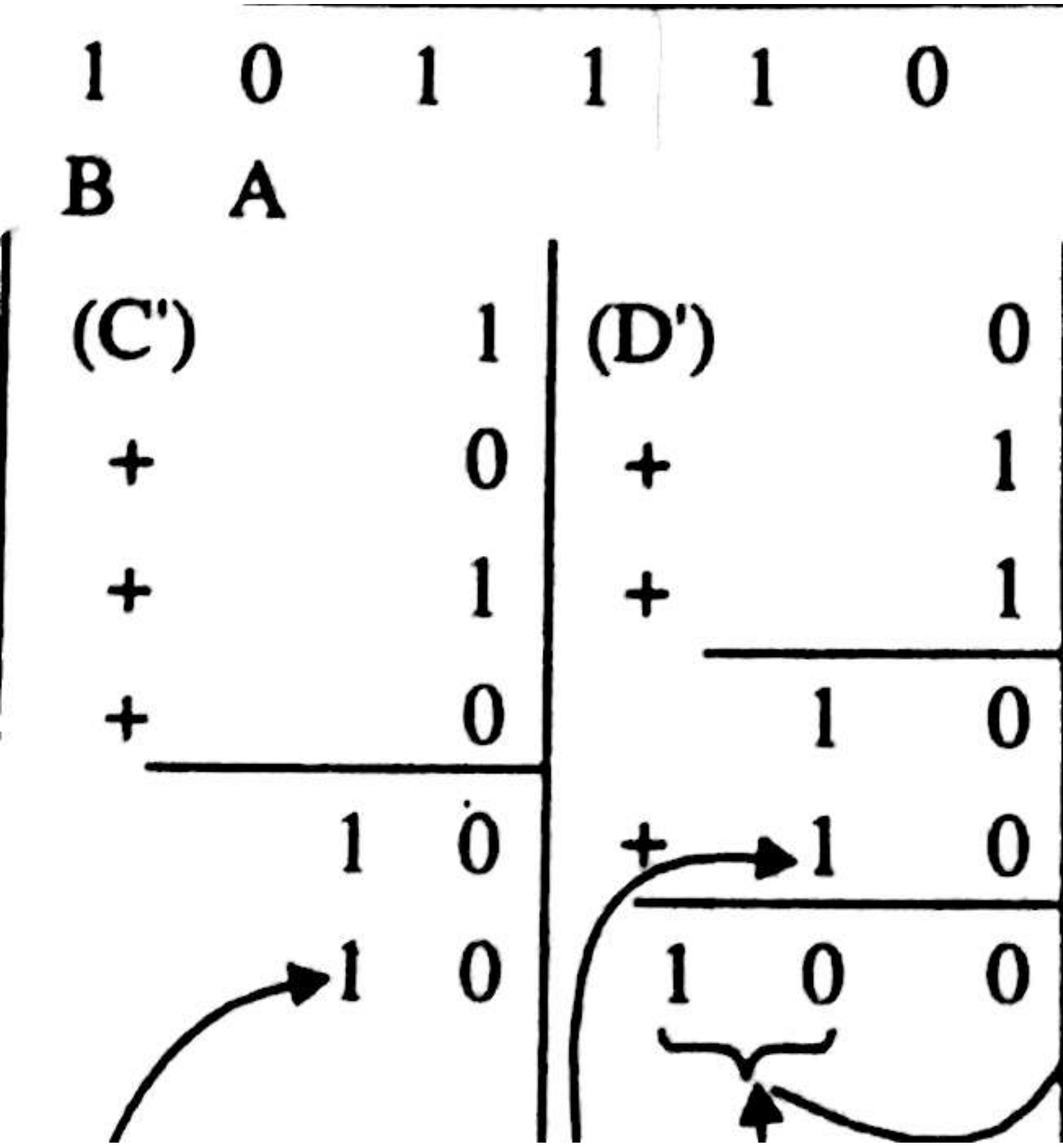
$(106)_8 = 1 \times 8^2 + 0 \times 8^1 + 6 \times 8^0 = (64 + 0 + 6)_{10} = (70)_{10}$.

**Ex. 46 :** Perform $(725)_8 \times (33)_8$.

**Soln. :**

**Step I :** Convert given octal to binary

Scanned by CamScanner




Scanned by CamScanner

**Step II :**

$$
\begin{array}{cccccccccccc}
 & & & & & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
 & & & & \times & & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
 & & & & & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
 & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X \\
 & & & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & X & X \\
 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X & X \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X & X & X \\
 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & X & X & X & X & X \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & X & X & X & X & X \\
 & & & \boxed{1} & \boxed{1} & \boxed{1} & & & & \boxed{1} & & & \\
\hline
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & \rightarrow \text{binary}
\end{array}
$$

**Step III :** Convert binary to octal.

$$
\begin{array}{cccccccccccccc}
(1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & \cdot 0 & 1)_2 & \rightarrow \text{binary} \\
 & \Downarrow & & & \Downarrow & & & \Downarrow & & & \Downarrow & & & \\
(1 & 5 & & & 4 & & & 7 & & & 1 & & )_8 & \text{octal}
\end{array}
$$

$\therefore$ **Ans.** = $(1547.1)_8$

To verify convert $(42.4)_8$ and $(31.2)_8$ in decimal

$(42.4)_8 = (34.5)_{10}$, $(31.2)_8 = (25.25)_{10}$

$\therefore$ $(34.5)_{10} \times (25.25)_{10} = (871.25)_{10}$

Now convert $(1547.1)_8$ to decimal you will get $(871.25)_{10}$.

### 2.8.6 Division :

For division in octal system, we follow the same steps i.e.

**Step I :** Convert octal to binary.
**Step II :** Perform binary division.
**Step III :** Convert given binary quotient and remainder to octal.

## 2.9 Arithmetic in Hexadecimal Number System :

### 2.9.1 Addition :

**Rule :** Since the base of the hex number system is 16, a carry (or logic 1) is carried forward if the p sum exceeds 16 and the required digit of partial sum = Partial sum – 16. Steps performe adding hex numbers is,

**Step I** : Convert EACH hex digit to equivalent decimal.
**Step II** : Add the two numbers.
**Step III** : Convert it (answer) to equivalent hex number and adjust carry, if any.

**For example, add $(A)_{16}$ and $(9)_{10}$**

Scanned by CamScanner

$$\begin{array}{rcll} & A & \rightarrow & (10)_{10} \\ + & & & \\ & 9 & \rightarrow & (9)_{10} \\ \hline & & & (19)_{10} \end{array}$$

→ Directly you can find that $(19)_{10} = (13)_{16}$

OR perform $(19)_{10} - (16)_{10} = 3$ → partial sum.

$\therefore$ $(19)_{10} = 1 \quad 3$

↑ ↑

carry partial sum

$\therefore$

$$\begin{array}{rl} & A \\ + & 9 \\ \text{carry} & \boxed{1} \\ \hline & 1\ 3 \end{array}$$

$\therefore$ **Ans. :** $= (13)_{16}$

Take one more example of two digits.

Add $(C2)_H$ and $(3E)_H$

$$\begin{array}{lcll} C\ 2 & \rightarrow & (12)_{10} & (2)_{10} \\ 3\ E & \rightarrow & (3)_{10} & (14)_{10} \\ \hline & & (15)_{10} & (16)_{10} \end{array}$$

Convert each here digit to equivalent decimal

$\therefore$ $(15)_{10}$ $(16)_{10}$

⇓

$16 - 16 = 0$ with carry (1)

We have,

$$\begin{array}{rll} & (15)_{10} & (16)_{10} \\ + & 1 & \\ \hline & (16)_{10} & (0)_{16} \end{array}$$

⇓

$(0)_{16}$

$\boxed{1}$

↑

carry

$\therefore$ Total answer is $(100)_{16}$. These all steps are represented in only ONE step.

$$\begin{array}{lllll} & C & 2 & \Rightarrow & (12)_{10} \quad (2)_{10} \\ + & & & + & \\ & 3 & E & \Rightarrow & (3)_{10} \quad (14)_{10} \\ & \text{carry} & \rightarrow & \boxed{1} & \boxed{1} \\ \hline & & & 1 & (16)_{10} - (16)_{10} = 0 \quad (16)_{10} - (16)_{10} = 0 \\ & & & \downarrow & \downarrow \quad \downarrow \\ & & \therefore & (1 & 0 \quad 0 \quad )_{16} \end{array}$$

**Ans. :** $= (100)_{16} = 100$ H

**: One should subtract $(16)_{10}$ if and only if total sum is equal or greater than $(16)_{10}$.**

Scanned by CamScanner

**Ex. 48 :** Add following hex numbers :

(i) $(DDCC)_{16} + (BBAA)_{16} = (?)_{16}$

(ii) $A5H + 2CH = (?)_{16}$

(iii) $FE1 + CA9 = (?)_{16}$

**Soln. :**

**(i)**

| | | | | | |
|---|---|---|---|---|---|
| | D D C C → | $(13)_{10}$ | $(13)_{10}$ | $(12)_{10}$ | $(12)_{10}$ |
| + | B B A A → | $(11)_{10}$ | $(11)_{10}$ | $(10)_{10}$ | $(10)_{10}$ |
| Carry | | 1 | 1 | 1 | 1 |
| | | 1 | 25 − 16 = 9 | 26 − 16 = 9 | 23 − 16 = 7 | 22 − 16 = 6 |
| | | ↓ | ↓ | ↓ | ↓ | ↓ |
| ∴ | Ans. | (1 | 9 | 9 | 7 | $6)_{16}$ |
| | | = | $(19976)_{16}$ | | | |

**(ii)**

| | | | | |
|---|---|---|---|---|
| | A 5 → | $(10)_{10}$ | $(5)_{10}$ | |
| + | 2 C → | $(2)_{10}$ | $(12)_{10}$ | |
| | carry → | 1 | | |
| | | $(13)_{10}$ | 17 − 16 = 1 | |
| | | ⇓ | ⇓ | |
| | | $(D)_{16}$ | $(1)_{16}$ | |

$\therefore$ Ans = $(D1)_{16}$

**(iii)**

| | | | | |
|---|---|---|---|---|
| | FE1 → | $(15)_{10}$ | $(14)_{10}$ | $(1)_{10}$ |
| | CA9 → | $(12)_{10}$ | $(10)_{10}$ | $(9)_{10}$ |
| | carry | 1 | 1 | |
| | | 1 | 28 − 16 = $(12)_{10}$ | 24 − 16 = $(8)_{10}$ | $(10)_{10}$ |
| | | ⇓ | ⇓ | ⇓ | ⇓ |
| | | 1 | C | 8 | A |

$\therefore$ **Ans. :** = $(1\ C\ 8\ A)_{16}$

**49 :** Perform following additions :

(i) $(DADA)_{16} + (BABA)_{16}$

(ii) FA7H + 1FDH

(iii) DDCCH + 1234H

(iv) 5678H + 4456H

Scanned by CamScanner

**Soln. :**

(i)

| | | | | | |
|---|---|---|---|---|---|
| | D A D A | → | $(13)_{10}$ | $(10)_{10}$ | $(13)_{10}$ | $(10)_{10}$ |
| + | | | + | | | |
| | B A B A | → | $(11)_{10}$ | $(10)_{10}$ | $(11)_{10}$ | $(10)_{10}$ |
| carry → | | → | 1 | 1 | 1 | |
| | | | 25 – 16 = 9 | 21 – 16 = 5 | 25 – 16 = 9 | 20 – 16 = 4 |
| | | | ↓ | ↓ | ↓ | ↓ |
| | | | 9 | 5 | 9 | 4 |
| | | 1 | | | | |

**Ans. :** $(19594)_{16}$

(ii)

| | | | | | | |
|---|---|---|---|---|---|---|
| | F A 7 | → | $(15)_{10}$ | $(10)_{10}$ | $(7)_{10}$ | |
| + | | | + | | | |
| | 1 F D | → | $(1)_{10}$ | $(15)_{10}$ | $(13)_{10}$ | |
| | | → | 1 | 1 | | – carry |
| | | | 17 – 16 = 1 | 26 – 16 = 10 | 20 – 16 = 4 | |
| | | | ↓ | ↓ | ↓ | |
| | | | 1 | A | 4 | |
| | | 1 | | | | |

∴ **Ans.** = $(11A4)_{16}$

(iii)

| | | | | | | |
|---|---|---|---|---|---|---|
| | D D C C | → | $(13)_{10}$ | $(13)_{10}$ | $(12)_{10}$ | $(12)_{10}$ |
| + | | | + | | | |
| | 1 2 3 4 | → | $(1)_{10}$ | $(2)_{10}$ | $(3)_{10}$ | $(4)_{10}$ |
| | | | 1 | 1 | 1 | |
| | | | $(15)_{10}$ | 16 – 16 = 0 | 16 – 16 = 0 | 16 – 16 = 0 |
| | | | ↓ | ↓ | ↓ | ↓ |
| | | | F | 0 | 0 | 0 |

∴ **Ans.** = $(F000)_{16}$

(iv)

| | | | | |
|---|---|---|---|---|
| | 5 | 6 | 7 | 8 |
| + | | | | |
| | 4 | 4 | 5 | 6 |
| | $(9)_{10}$ | $(10)_{10}$ | $(12)_{10}$ | $(14)_{10}$ |
| | ↓ | ↓ | ↓ | ↓ |
| | 9 | A | C | E |

. **Ans.** = $(9ACE)_{16}$.

ne more method is there to perform hex addition. The steps are as explained follows :

Convert hex to binary.

Scanned by CamScanner

(ii) Perform binary addition (Refer Table 1)
(iii) Convert final answer in binary to hex.
We will see two examples for the same.
(i) Add $(A)_{16}$ and $(8)_{16}$

$(A)_{16} \rightarrow (1\ 0\ 1\ 0)_2$
\+
$(8)_{16} \rightarrow (1\ 0\ 0\ 0)_2$
carry [1]

[1] [0 0 1 0] → binary
1 2 → 12 H is the answer

(ii) Perform C2H + 3EH.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| C2H → | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| + 3EH → | | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| | [1] | [1] | [1] | [1] | [1] | [1] | [1] | | Carry |
| | [1] | [0 | 0 | 0 | 0] | [0 | 0 | 0 | 0] binary |
| | 1 | 0 | | | | 0 | | | hex |

## 2.9.2 Subtraction :

Now we are going to study subtraction in hex numbers. We are going to study three method of subtraction.

(i) Direct subtraction.
(ii) 1C method for subtraction.
(iii) 2C method for subtraction.

**(i) Direct Subtraction :**

**Rule :** If the minuend is less than subtrahend, we borrow 16 and take carry '1'. (Reference : See 'octal subtraction', the difference is in octal we borrow '8', in hex we borrow 16).

We will see one examples for the same.
(a) Perform $(A)_{16} - (8)_{16}$

$(A)_{16} \rightarrow (10)_{10}$
\-
$(8)_{16} \rightarrow (8)_{10}$
$(2)_{10} \Rightarrow (2)_{16}$

∴ **Ans. :** $= (2)_{16}$

(b) Perform 73H – 1CH

$(16)_{10}$ → Borrow
$(19)_{10}$
(73 H) → $(7)_{10}$ $(3)_{10}$ → Operand 1
\- \-
(1 CH) → $(1)_{10}$ $(12)_{10}$ → Operand 2
\+ $(2)_{10}$
$(1)_{10}$ → Carry

$(7)_{10} - (2)_{10} = 5$ $(19 - 12) = 7$
↓ ↓
5 7

∴ **Ans. :** = 57 H

Scanned by CamScanner

(c) Perform $(A65)_H - (777)_H$.

$$
\begin{array}{lllllll}
 & & & & & & (16) \\
 & & & & & + & \\
A & 6 & 5 & \rightarrow & (10)_{10} & & (6)_{10} \\
- & & & & & & \\
7 & 7 & 7 & \rightarrow & + \; (7)_{10} & & (7)_{10} \\
 & & & & \boxed{1}_{10} \; (8)_{10} & + & \boxed{1}_{10} \; (8) \\
\hline
 & & & & 10 - 8 = (2)_{10} & & 22 - 8 = (14)_{10} \\
 & & & & \downarrow & & \downarrow \\
 & & & & 2 & & E
\end{array}
$$

$\therefore$ **Ans.** $= (2EE)_{16}$ OR 2EEH

(d) Perform $(2C48)_{16} - (9AA)_{16}$

$$
\begin{array}{lllllll}
 & & & & & & (16) \\
 & & & & & + & \\
2 & C & 4 & 8 & \rightarrow (2)_{10} & (12)_{10} & (4)_{10} \\
- & & & & & & \\
0 & 9 & A & A & \rightarrow (0)_{10} & + \; (9)_{10} & + \; (10)_{10} \\
 & & & & & \boxed{1}_{10} \; (10)_{10} & + \; \boxed{10}_{10} \\
\hline
 & & & & 2 - 0 = (2)_{10} & 12 - 10 = (2)_{10} & 20 - 1 \\
 & & & & \downarrow & \downarrow & \downarrow \\
 & & & & 2 & 2 & 9
\end{array}
$$

$\therefore$ **Ans.** $= (229E)_{16}$.

In this also you have one more method. The steps to be followed :

(1) Convert given hex number to equivalent binary.

(2) Perform binary subtraction (Refer Table 2.2)

(3) Convert binary answer to equivalent hex.

Scanned by CamScanner

$(73)_{10} \rightarrow (0\ 1\ 1\ 1\ 0\ 0\ 1\ 1)_2 \rightarrow$ Operand 1

$-$

$(1C)_{10} \rightarrow (0\ 0\ 0\ 1\ 1\ 1\ 0\ 0)_2 \rightarrow$ Operand 2

$\boxed{-1}\ \boxed{-1}\ \boxed{-1} \rightarrow$ borrow

$0\ 1\ 0\ 1 \quad 0\ 1\ 1\ 1 \rightarrow$ binary answer

$\Downarrow \qquad \Downarrow$

$5 \qquad 7 \rightarrow$ hex

**Ans.** $= (57)_{16}$

(c) Perform $(A65)_{16} - (777)_{16}$

$(A\ 6\ 5)_{10} \rightarrow (1\ 0\ 1\ 0\ \ 0\ 1\ 1\ 0\ \ 0\ 1\ 0\ 1)_2 \rightarrow$ Operand 1

$-$ $-$

$(7\ 7\ 7)_{16} \rightarrow (0\ 1\ 1\ 1\ \ 0\ 1\ 1\ 1\ \ 0\ 1\ 1\ 1)_2 \rightarrow$ Operand 2

$\boxed{1}\ \boxed{1}\ \boxed{1}\ \boxed{1}\ \ \boxed{-1}\ \boxed{-1}\ \boxed{-1}\ \boxed{-1}\ \ \boxed{-1}\ \boxed{-1}$

$0\ 0\ 1\ 0 \quad 1\ 1\ 1\ 0 \quad 1\ 1\ 1\ 0 \rightarrow$ binary answer

$\Downarrow \qquad \Downarrow \qquad \Downarrow$

$2 \qquad E \qquad E \rightarrow$ hex

$\therefore$ **Ans.** $= (2EE)_{16}$

## 2.9.3 One's Complement Method for Subtraction in Hex Numbers :

Under this section we are going to perform subtraction of hex number using 1C code. 1C code is used to represent negative numbers. Here you may ask how to find 1C of hex number. I have described two methods.

**Method I :** Subtract the number, whose 1C code we want to find out, from FF ..... FFH. Numbers of FFF ..... digits depends upon numbers of digits of hex number.

Say for an example

Find 1C code for 123H

In 123H we have 3 digits

Therefore we have to perform following operation :

$F\ F\ F \rightarrow \quad (15)_{10} \quad (15)_{10} \quad (15)_{10}$

$-$ $-$

$1\ 2\ 3 \rightarrow \quad (1)_{10} \quad (2)_{10} \quad (3)_{10}$

$(14)_{10} \quad (13)_{10} \quad (12)_{10}$

$\downarrow \qquad \downarrow \qquad \downarrow$

$E \qquad D \qquad C$

$\therefore \quad (-123)_{16}$ in 1C code is $(EDC)_{16}$.

**Method II :** In 2nd method you have to perform following steps :

Scanned by CamScanner

**Step II** : 'Complement' binary number.
**Step III** : Convert complemented binary number to its equivalent hex.

Say for an example,
Convert $(A)_{16}$ to its equivalent 1C code.

$(A)_{16} \Rightarrow (1\ 0\ 1\ 0)_2 \rightarrow$ binary (Step I)

$(0\ 1\ 0\ 1)_2 \rightarrow$ complement (Step II)

$(5)_{16} \rightarrow$ binary to hex (Step III)

$\therefore$ $(-A)_{16}$ in 1C code is $(5)_{16}$

Take one more example
Convert (12)H to its equivalent 1C code

$(12)_H \Rightarrow (0\ 0\ 0\ 1\ 0\ 0\ 1\ 0)_2 \rightarrow$ binary (Step I)

$(1\ 1\ 1\ 0\ 1\ 1\ 0\ 1)_2 \rightarrow$ complement (Step II)

E D $\rightarrow$ binary to hex

$\therefore$ 1C of $(+12)_H \rightarrow (ED)_H \rightarrow (-12)_H$

We will solve same examples that we have solved, to check for correct answer.

(a) Perform $(A)_{16} - (8)_{16}$ using 1C method.

**Soln. :** $(A)_{16} - (8)_{16} = (A)_{16} + (-8)_{16}$

$\therefore$ Find 1C of $(8)_{16}$

1C of $(8)_{16}$ = F $\rightarrow (15)_{10}$

$-$ 8 $\rightarrow (8)_{10}$

$(7)_{10} \rightarrow (7)_{16}$

$\therefore$ $(-8)_{16}$ in 1C = $(7)_{16}$

$(A)_{16} \Rightarrow (A)_{16} \Rightarrow (10)_{10}$

$+ \qquad +$

$(-8)_{10} \Rightarrow (7)_{16} \Rightarrow 1 \quad (7)_{10}$

End around carry $\rightarrow$ $17 - 16 = 1$

$\rightarrow 1$

$(2)_{10} \rightarrow$

$\therefore$ **Ans.** = $(2)_{16}$

(b) Perform $(73)_{16} - (1C)_{16}$ using 1C method.

**Soln. :** $(73)_{16} - (1C)_{16} = (73)_{16} + (-1C)_{16}$

1C of $(1C)_{16} \Rightarrow$ F F $\rightarrow$ $(15)_{10}$ $(15)_{10}$

$-$ $-$

1 C $\rightarrow$ $(1)_{10}$ $(12)_{10}$

$(14)_{10}$ $(3)_{10}$

Scanned by CamScanner

$(73)_{16}$ ⇒ $(73)_{16}$ ⇒ $(7)_{10}$ $(3)_{10}$

$+$ $+$ $+$

$(-1C)_{16}$ ⇒ $(E3)_{16}$ ⇒ $(14)_{10}$ $(3)_{10}$

1

$21 - 16 = 5$ $(6)_{10}$

End around carry → 1

$(5)_{10}$ $(7)_{10}$ ⇒ $(57)_{16}$

(c) Perform $(A65)_H - (777)_H$ using 1C method

**Soln. :** $(A65)_{16} - (777)_{16} = (A65)_{16} + (-777)_{16}$

1 C of $(777)_{16}$ ⇒ F F F → $(15)_{10}$ $(15)_{10}$ $(15)_{10}$

$-$

7 7 7 → $(7)_{10}$ $(7)_{10}$ $(7)_{10}$

$(8)_{10}$ $(8)_{10}$ $(8)_{10}$

↓ ↓ ↓

8 8 8 → hex

$\therefore$ $(-777)_{16}$ in 1C = $(888)_{16}$.

A 6 5 → A 6 5 → $(10)_{10}$ $(6)_{10}$ $(5)_{10}$

$+$ $+$

$(-7\ 7\ 7)$ → 8 8 8 → $(8)_{10}$ $(8)_{10}$ $(8)_{10}$

carry → 1

$18 - 16 = (2)_{10}$ $(14)_{10}$ $(13)_{10}$

$(1)_{10}$

$(2)_{10}$ $(14)_{10}$ $(14)_{10}$

↓ ↓ ↓

2 E E → hex

$\therefore$ **Ans. :** $= (2EE)_{16}$

(d) Perform $(2C48)_{16} - (9AA)_{16}$ using 1C method.

**Soln. :** $(2C48)_{16} - (9AA)_{16} = (2C48)_{16} + (-9AA)_{16}$

Here, $(2C48)_{16}$ is four digit where 9AA is 3 digit, so make $(9AA)_{16}$, four digit by inserting extra '0', therefore number is $(09AA)_{16}$.

Here we have to find complement (1C) of $(09AA)_{10}$

1C of $(09AA)_{10}$ → F F F F → $(15)_{10}$ $(15)_{10}$ $(15)_{10}$ $(15)_{10}$

$-$

0 9 A A → $(0)_{10}$ $(9)_{10}$ $(10)_{10}$ $(10)_{10}$

$(15)_{10}$ $(6)_{10}$ $(5)_{10}$ $(5)_{10}$

↓ ↓ ↓ ↓

F 6 5 5 → hex

$\therefore$ $(-9AA)_{16}$ in 1C code = $(F655)_{16}$

Scanned by CamScanner

| | | | | | | |
|---|---|---|---|---|---|---|
| + (2C48) | → | 2C48 | → $(2)_{10}$ | $(12)_{10}$ | $(4)_{10}$ | $(8)_{10}$ |
| (−09AA) | → | F655 | → $(15)_{10}$ | $(6)_{10}$ | $(5)_{10}$ | $(5)_{10}$ |
| Carry → | | | 1 | 1 | | |
| | | | 18 − 16 = 2 | 18 − 16 = 2 | $(9)_{10}$ | $(13)_{10}$ |
| End around carry → | | | | | | (1) |
| | | | 2 | 2 | $(9)_{10}$ | $(14)_{10}$ |
| | | | ↓ | ↓ | ↓ | ↓ |
| | hex → | | 2 | 2 | 2 | E |

∴ **Ans.** = $(229E)_{16}$.

In all examples we have directly worked on hex number. There is one more method. The steps to be performed in the method are as follows :

**Step I** : Convert hex to binary. (Convert both the given number).
**Step II** : Find out 1C of negative number. (i.e. 1C of operand 2).
**Step III** : Add operand 1 (and 1C of operand 2. Adjust end around carry).
**Step IV** : Now whatever the binary answer is there, you convert it to its equivalent hex. This is achieved by grouping four bits.

For this you can take any previous example and verify your answer by going through this method.

### 2.9.4 Two's Complement Method for Subtraction in Hex Number :

Under this section we are going to perform subtraction of hex number using 2's complement (2C) method. 2C code is proper method for representing negative numbers. To convert given number in 2C code following methods are there :

**Method 1 :** Subtract the number, whose 2C code we want to find out, from FFF ..... FH. Numbers of FFF ..... FH digits depends upon, numbers of digits in hex number. After word add '1' into it. This will give you 2C code. Say for an example, 234H. This number has 3 digits.

1C code for $(234)_{16}$ = F F F → $(15)_{10}$ $(15)_{10}$ $(15)_{10}$

−

2 3 4 → $(2)_{10}$ $(3)_{10}$ $(4)_{10}$

$(13)_{10}$ $(12)_{10}$ $(11)_{10}$

Now add '1' in 1C code

$(13)_{10}$ $(12)_{10}$ $(11)_{10}$ → 1 code

\+ 1

$(13)_{10}$ $(12)_{10}$ $(12)_{10}$

↓ ↓ ↓

D C C → 2 C code for $(234)_{16}$

∴ $(-234)_{16}$ in 2C = $(DCC)_{16}$.

**Method 2 :** Perform following steps to find 2C code in hex :

**Step I** : Convert hex number to its equivalent binary.
**Step II** : *Complement* binary number. This is nothing but 1's complement.
**Step III** : Add '1' in 1C code.
**Step IV** : Convert final binary answer to its equivalent hex. Say for an example, represent $(234)_{16}$ in code.

Scanned by CamScanner

$(234)_{16} \rightarrow (0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0)_2 \rightarrow$ binary number

$\downarrow$ $\downarrow$

$(1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1)_2 \rightarrow$ 1 C code

$+$ 1

$\boxed{1}\ \boxed{1}$

$\underbrace{1\ 1\ 0\ 1}_{D}\ \underbrace{1\ 1\ 0\ 0}_{C}\ \underbrace{1\ 1\ 0\ 0}_{C} \rightarrow$ 2C in binary group 4 bits

$\rightarrow$ 2C in hex

Now we are going to perform subtraction in hex using 2C method. We are going to solve same examples that we have solved in 1C. You are aware with the fact that 2C = 1C + 1. So in previous examples we have already found 1C code. So I am simply going to add '1' and get 2C code.

(a) Perform $(A)_{16} - (8)_{16}$ using 2C method

**Soln. :** $(A)_{16} - (8)_{16} = (A)_{16} + (-8)_{16}$

$\therefore$ 2C of $(8)_{16}$ will gives $(-8)_{16}$

2C of $(8)_{16}$ = 1C of $(8)_{16}$ + 1 = $(7)_{10} + (1)_{10} = (8)_{10} \Rightarrow (-8)_{16}$

$(A)_{16} \rightarrow (10)_{10}$

$+$

$(-8)_{16} \rightarrow (8)_{10}$

carry $\boxed{1}$

$\uparrow$ $18 - 16 = 2$

discard carry

**Ans. :** $(2)_{10}$

(b) Perform $(73)_{16} - (1C)_{16}$ in 2C code.

**Soln. :** $(73)_{16} - (1C)_{16} = (73)_{16} + (-1C)_{16}$

2C for $(1C)_{16}$ = 1C of $(1C)_{16}$ + 1

= $(E3)_{16} + 1 = (E4)_{16} \Rightarrow (-1C)_{16}$

$(73)_{16} \rightarrow 7\ 3 \rightarrow (7)_{10}\ (3)_{10}$

$+$ $+$

$(-1C)_{16} \rightarrow E\ 4 \rightarrow (14)_{10}\ (4)_{10}$

$\boxed{1}$

$21 - 16 = 5\ (7)_{10}$

Discard carry

$\downarrow$ $\downarrow$

5 7 $\rightarrow$ hex

$\therefore$ **Ans.** = $(57)_{16}$.

(c) Perform $(A65)_{16} - (777)_{16}$ using 2C code.

**Soln. :** $(A65)_{16} - (777)_{16} = (A65)_{16} + (-777)_{16}$

2C for $(777)_{16}$ = 1C of $(777)_{16}$ $\rightarrow$ 8 8 8

$+$ 1 $+$ 1

8 8 9 $\Rightarrow (-777)_{16}$

Scanned by CamScanner

$(A65)_{16} \rightarrow (A65)_{16} \rightarrow (10)_{10} \quad (6)_{10} \quad (3)_{10}$

$+$

$(-777)_{16} \rightarrow (889)_{16} \rightarrow (8)_{10} \quad (8)_{10} \quad (9)_{10}$

$\boxed{1}$

Discard carry

$18 - 16 = 2 \quad (14)_{10} \quad (14)_{16}$

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

$2 \quad E \quad E \quad \rightarrow$ hex

**Ans. :** $(2EE)_{16}$

(d) Perform $(2C48)_{16} - (9AA)_{16}$.

**Soln. :** $(2C48)_{16} - (9AA)_{16} = (2C48)_{16} + (-9AA)_{16}$

1 C of $(9AA)_{16} \rightarrow$ F 6 5 5

$+ \qquad +$

1 $\rightarrow$ 1

2 C of $(9AA)_{16} \rightarrow$ F 6 5 6 $\rightarrow (-9AA)_{16}$

$(2C48)_{16} \rightarrow (2C48) \rightarrow (2)_{10} \quad (12)_{10} \quad (4)_{10} \quad (8)_{10}$

$+ \qquad +$

$(-9AA)_{16} \rightarrow (F656) \rightarrow (15)_{10} \quad (6)_{10} \quad (5)_{10} \quad (6)_{10}$

$\boxed{1} \qquad \boxed{1}$

Discard carry

$18 - 16 = 2 \quad 18 - 16 = 2 \quad (9)_{10} \quad (19)_{10}$

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

$2 \quad 2 \quad 9 \quad E$

$\therefore$ **Ans. :** $= (229E)_{16}$

### 2.9.5 Multiplication :

If you directly multiply hex numbers the thing becomes bit complicated. Therefore it will be better if one follows following steps :

| | |
|---|---|
| **Step I :** | Convert hex to binary. (Multiplier and Multiplicand both). |
| **Step II :** | Perform ***Simple binary multiplication***. For this refer Table 2.6. |
| **Step III :** | After performing multiplication, whatever answer you get in binary, convert it to equivalent hex. This you can achieve by grouping 4 binary bits. Add extra zeros where required. |

Let's solve some examples for the same.

**Ex. 50 :** Multiply $(72)_{16}$ and $(39)_{16}$.

**Soln. :**

**Step I :** Convert hex to binary,

$(72)_{16} = (0\ 1\ 1\ 1\ \ 0\ 0\ 1\ 0)_2 = (1\ 1\ 1\ 0\ 0\ 1\ 0)_2$

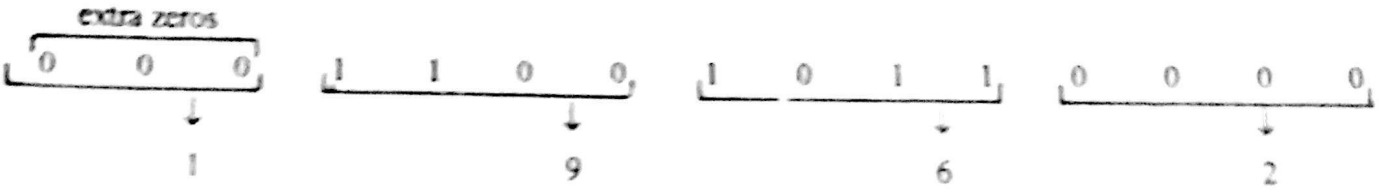
$(39)_{16} = (0\ 0\ 1\ 1\ \ 1\ 0\ 0\ 1)_2 = (1\ 1\ 1\ 0\ 0\ 1)_2$

**Step II :** Perform binary multiplication :

Scanned by CamScanner

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |
| | | | | | | | 1 | 1 | 1 | 0 | 0 | 1 | |
| | | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |
| | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | |
| | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | |
| | | | 1 | 1 | 1 | 0 | 0 | 1 | 0 | X | X | X | |
| | | 1 | 1 | 1 | 0 | 0 | 1 | 0 | X | X | X | X | |
| | 1 | 1 | 1 | 0 | 0 | 1 | 0 | X | X | X | X | X | |
| | [10] | [10] | [1] | [1] | [1] | [1] | [1] | | | | | | |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | → binary |

**Step III :** Convert binary to hex




∴ **Ans.** = $(1962)_{16}$

---

**Ex. 51 :** $(A2C)_{16} \times (B42)_{16}$

**Soln. : Step I :** Convert hex to binary

$(A2C)_{16} = (1010 \quad 0010 \quad 1100)_2$

$(B42)_{16} = (1011 \quad 0100 \quad 0010)_2$

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | |
| | | | | | | | | × | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | X | |
| | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | |
| | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X |
| | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X |
| | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X | X |
| | | | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | X | X | X | X | X | X | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | | |
| | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | X | X | X | X | X | X | X | | | | |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | X | X | X | X | X | X | X | X | | | | |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | |

743

$= (728358)_{16}$

---

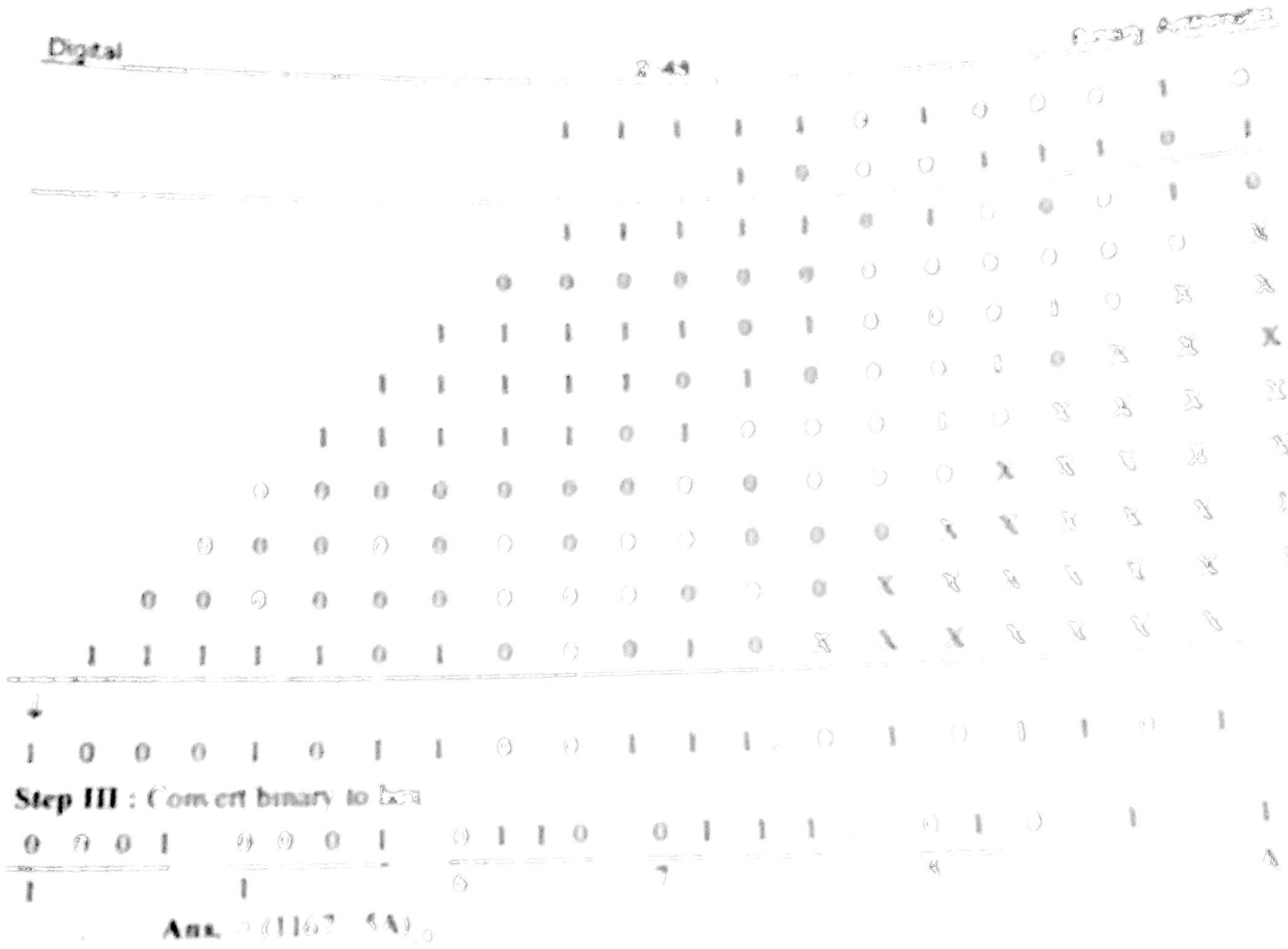
**Ex. 52 :** Perform $(FA2)_{16} \times (11D)_{16}$

**Soln. : Step I :** Convert hex to binary

$FA2 = (1111 \quad 1010 \quad 0010)_2 = (111 \quad 110[illegible] \quad 001)_2$

$11D = (0001 \quad 0001 \quad 1101)_2 = (10001 \quad 1101)_2$

Scanned by CamScanner



**Step III** : Convert binary to hex

**Ans.**

### 2.9.6 Division :

To perform division in hex number you have to follow steps which are described as follows :

**Step I** : Convert hex number to equivalent binary

**Step II** : Perform ***binary division***

**Step III** : Convert final answer in binary i.e. Quotient and Remainder to equivalent hex

So the Step I and Step III both are same as that we performed in multiplication. Step II is already taught to you.

Now we are going to solve some examples which will cover binary, octal, hex, addition, subtraction, multiplication and division. This is for your practice.

## Examples

**Ex. 53 :** Convert following decimal numbers into binary numbers :

(i) – 128, (ii) 257

**Soln. :** We have one positive and negative number. For negative number we have to present in si magnitude, 1's complement and 2's complement method.

**(i) – 128, First find out +128, use double dabble method.**

Scanned by CamScanner

| | | | |
|---|---|---|---|
| 2 | 128 | | |
| 2 | 64 | 0 | → LSB |
| 2 | 32 | 0 | |
| 2 | 16 | 0 | |
| 2 | 8 | 0 | |
| 2 | 4 | 0 | |
| 2 | 2 | 0 | |
| 2 | 1 | 0 | |
| | | 1 | → MSB |

$(+ 128)_{10} = (1\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0)_2$

(a) SM representation : We will use 12 bits

$(1\ \ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0)_2$

↑

sign bit

(b) 1's complement :

$(0\ 0\ 0\ 0\ \ 1\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0)_2 \rightarrow (+ 128)_{10}$

$(1\ 1\ 1\ 1\ \ 0\ 1\ 1\ 1\ \ 1\ 1\ 1\ 1)_2 \rightarrow (- 128)_{10}$ in 1C

(c) 2's complement

1's complement → 1 1 1 1 0 1 1 1 0 1 1 1

\+ \+

1 1

2's complement $(1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0)_2$

∴ $(- 128)_{10}$ in 2C = $(1111\ 1000\ 0000)_2$

(ii) $(257)_{10}$

| | | | |
|---|---|---|---|
| 2 | 257 | | |
| 2 | 128 | 1 | → LSB |
| 2 | 64 | 0 | |
| 2 | 32 | 0 | |
| 2 | 16 | 0 | |
| 2 | 8 | 0 | |
| 2 | 4 | 0 | |
| 2 | 2 | 0 | |
| | 1 | 0 | |
| | | 1 | → MSB |

$(+ 257)_{10} = \boxed{0}\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 1$

↑

sign bit

**Ex. 54 :** Solve (i) A5H + 2CH

(ii) 1101 × 1001.

Scanned by CamScanner

**Soln. :** (i)

| | | | | |
|---|---|---|---|---|
| | A 5 | → | $(10)_{10}$ | $(5)_{10}$ |
| + | | | + | |
| | 2 C | → | $(2)_{10}$ | $(12)_{10}$ |
| | | | 1 | |
| | | | $(13)_{10}$ | $17 - 16 = 1$ |
| | | | ⇓ | ↓ |
| | | | D | 1 |

∴ **Ans. :** = $(D1)_{16}$

**(ii)**

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | 1 | 1 | 0 | 1 |
| | | × | 1 | 0 | 0 | 1 |
| | | | 1 | 1 | 0 | 1 |
| | | 0 | 0 | 0 | 0 | X |
| | 0 | 0 | 0 | 0 | X | X |
| 1 | 1 | 0 | 1 | X | X | X |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |

∴ **Ans.** = $(1110101)_2$.

---

**Ex. 55 :** Determine 2's complement of (i) 8FH, (ii) FFH.

**Soln. :** (i)

| | | | | | |
|---|---|---|---|---|---|
| 1's complement | → | F F | → | $(15)_{10}$ | $(15)_{10}$ |
| − | | − | | | |
| | 8 F | | | $(8)_{10}$ | $(15)_{10}$ |
| | | | | $(7)_{10}$ | $(0)_{10}$ $(70)_{16}$ |

| | | | | | |
|---|---|---|---|---|---|
| | 1's complement | → | 7 | 0 | |
| + | 1 | + | 1 | | ∴ $(-8FH) = (71\ H)_{2C}$ |
| | 2's complement | | 7 | 1 | |

**(ii)**

| | | | | |
|---|---|---|---|---|
| 1's complement | → | F | F | H |
| − | | F | F | H |
| | | 0 | 0 | |

| | | | | | |
|---|---|---|---|---|---|
| | 1's complement | → | 0 | 0 | |
| + | | + | | | |
| | 1 | | 1 | | ∴ $(-FFH) = (01H)_{2C}$ |
| | 2's complement | | 0 | 1 | |

2C → 2's complement.

Scanned by CamScanner

**Ex. 56 :** Determine decimal equivalent of following in signed magnitude form :

(i) 8DH, (ii) FFH

**Soln. :**

(i) 8DH → 1 | 0 0 0 1 1 0 1 | Magnitude = $(13)_{10}$ $= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

↑ sign

1 = negative

∴ $(8DH)_{SM} \rightarrow (-13)_{10}$

(ii) FFH → 1 | 1 1 1 1 1 1 1 | Magnitude

↑ sign

negative

$$\text{Magnitude} = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$
$$= 64 + 32 + 16 + 8 + 4 + 2 + 1$$
$$= 127$$

∴ Decimal number is $(-127)_{10}$.

---

**Ex. 57 :** Perform FA9 – 1FD

**Soln. :**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | $(16)_{10}$ | $(26)_{10}$ | $(16)_{10}$ | $(25)_{10}$ | → borrow |
| F A 9 → | $(15)_{10}$ | | $(10)_{10}$ | | $(9)_{10}$ | | → Operand 1 |
| 1 F D → | $(1)_{10}$ | $(2)_{10}$ | $(15)_{10}$ | $(16)_{10}$ | $(13)_{10}$ | | → Operand 2 |
| | $(1)_{10}$ | | $(1)_{10}$ | | | | → Carry |
| | $15 - 2 = (13)_{10}$ | | $26 - 16 = (10)_{10}$ | | $25 - 13 = (12)_{10}$ | | |
| | ↓ | | ↓ | | ↓ | | |
| | D | | A | | C | | |

∴ **Ans.** = $(DAC)_{16}$.

---

**Ex. 58 :** Divide $(11101110)_2$ by $(1001)_2$.

**Soln. :**

```
1 0 0 1 ) 1 1 1 0 1 1 1 0 ( 1 1 0 1 0
          1 0 0 1 ↓
          -------
            1 0 1 1
            1 0 0 1 ↓ ↓
            -------
            0 0 1 0 1 1
                1 0 0 1 ↓
                -------
                0 0 1 0 0
```

∴ Quotient = $(11010) \rightarrow (26)_{10}$

Remainder = $(100)_2 \rightarrow (4)_{10}$.

Scanned by CamScanner

**Ex. 59 :** Carry out $(756)_8 + (125)_8 - (176)_8 = (\ ?\ )_8$

**Soln. :**

| | | | |
|---|---|---|---|
| | 7 | 5 | 6 |
| | 1 | 2 | 5 |
| [1] | [1] | [1] | |
| | 9 − 8 = 1 | 8 − 8 = 0 | 11 − 8 = 3 |
| | ↓ | ↓ | ↓ |
| | 1 | 2 | 3 |

$\therefore \quad (756)_8 + (125)_8 = (1103)_8$

| | | | | |
|---|---|---|---|---|
| | $(8)_{10}$ | $(8)_{10}$ | (8) | → borrow |
| | $(0)_{10}$ | $(8)_{10}$ | $(11)_{10}$ | |
| 1 | $(1)_{10}$ | 0 | (3) | → operand 1 |
| − | | | | |
| 0 | 1 | 7 | 6 | → operand 2 |
| [1] | [1] | [1] | | → carry |
| | $(1)_{10}$ | $(2)_{10}$ | $(8)_{10}$ | |
| 1 − 1 = 0 | 9 − 2 = 7 | 8 − 8 = 0 | 11 − 6 = 5 | |
| ↓ | ↓ | ↓ | ↓ | |
| 0 | 7 | 0 | 5 | |

$\therefore$ **Ans.** $= (0705)_8 = (705)_8$.

**. 60 :** Carry out the following arithmetic operation and represent result in same format :
(i) $(621)_8 - (217)_8$, (ii) $(F4D2)_{16} - (91FA)_{16}$

**ln. : (i)**

| | | | | |
|---|---|---|---|---|
| | | $(8)_{10}$ | | → borrow |
| | | | $(9)_{10}$ | |
| 6 | 2 | 1 | | → operand 1 |
| − | | | | |
| 2 | 1 | 7 | | → operand 2 |
| | | | $(2)_{10}$ | |
| | [1] | | | → carry |
| 4 | 2 − 2 = 0 | 9 − 7 = 2 | | |
| ↓ | ↓ | ↓ | | |
| 4 | 0 | 2 | | |

**Ans.** $= (402)_8$.

Scanned by CamScanner

(ii)

F 4 D 2 → $(15)_{10}$ $(4)_{10}$ $(16)_{10}$ $(13)_{10}$ $(29)_{10}$ $(16)_{10}$ $(2)_{10}$ $(18)_2$ → borrow

− −

9 1 F A → $(9)_{10}$ $(1)_{10}$ $(2)_{10}$ $(15)_{10}$ $(16)_{10}$ $(10)_{10}$

[1] [1] → carry

$(6)_{10}$ $4 - 2 = 2$ $29 - 16 = (13)_{10}$ $18 - 10 = (8)_{10}$

↓ ↓ ↓ ↓

6 2 D 8

∴ **Ans.** = $(62D8)_{16}$

---

**Ex. 61 :** Using 2's complement perform following :

(i) $(42)_{10} - (68)_{10}$. (ii) $(25)_{10} - (16)_{10}$.

**Soln. :** (i) $(42)_{10} = (101010)_2$

$= (0010\ 1010)_2$

$(68)_{10} = 1000100$

$= (0100\ 0100)_2$

1's complement of $(68)_{10}$ → 1 0 1 1 1 0 1 1

\+ 1 + 1

2's complement of $(68)_{10}$ 1 0 1 1 1 1 0 0 → $(-68)_{10}$ in 2C

$(42)_{10}$ → 0 0 1 0 1 0 1 0

+

$(-68)_{10}$ → 1 0 1 1 1 1 0 0

[1] [1] [1]

1 1 1 0 0 1 1 0 ⇒ $(-26)_{10}$

↑

as 1 – no. is negative

∴ **Ans.** is in 2's complement form

1 1 1 0 0 1 1 0

↓ ↓ ↓ → complement

0 0 0 1 1 0 0 1

\+ 1

(0 0 0 1 1 0 1 $0)_2$ ⇒ $(+26)_{10}$

(ii)

$(25)_{10} = (11001)_2$ $(16)_{10} = (10000)_2$

1's complement of $(16)_{10}$ → 0 1 1 1 1

\+ 1 + 1

2's complement of $(16)_{10}$ 1 0 0 0 0 ⇒ $(-16)_{10}$

Scanned by CamScanner

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $(25)_{10}$ | → | | 1 | 1 | 0 | 0 | 1 | |
| + | | | + | | | | | | |
| | $(-16)_{10}$ | → | | 1 | 0 | 0 | 0 | 0 | |
| | | | $\boxed{1}$ | | | | | | |
| | | discard | | 0 | 1 | 0 | 0 | 1 | $\Rightarrow (+9)_{10}$ |

**Ex. 62 :** Solve following :

(i) 1101.01 × 100.1 (ii) 111110.1 ÷ 1010

**Soln. :** (i)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 1 | 0 | 1 | . | 0 | 1 |
| | | | | 1 | 0 | 0 | . | 1 | |
| | | | 1 | 1 | 0 | 1 | | 0 | 1 |
| | | 0 | 0 | 0 | 0 | 0 | | 0 | X |
| | 0 | 0 | 0 | 0 | 0 | 0 | | X | X |
| 1 | 1 | 0 | 1 | 0 | 1 | X | | X | X |
| | | $\boxed{1}$ | | | | | | | |
| 1 | 1 | 1 | 0 | 1 | 1 | . | 1 | 0 | 1 |

(ii)

```
1010 ) 111110.10 ( 110.01
     - 1010
       1011
     - 1010
       00010 10
     -    1010
          0000
```

Quotient = $(110.01)_2$ , Remainder = 0

**Ex. 63 :** Subtract $(723)_8$ and $(237)_8$

**Soln. :**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | $(8)_{10}$ | $(10)_{10}$ | $(8)_{10}$ | $(11)_{10}$ | → | borrow |
| | 7 | 2 | | 3 | | → | operand 1 |
| − | 2 | 3 | $(4)_{10}$ | 7 | | → | operand 2 |
| | $\boxed{1}$ | $\boxed{1}$ | | | | → | carry |
| | 4 | 10 − 4 = 6 | | 11 − 7 = 4 | | | |
| | ↓ | ↓ | | ↓ | | | |
| | 4 | 6 | | 4 | | | |

∴ **Ans.** = $(464)_8$

Scanned by CamScanner

**Ex. 64 :** (a) Carry out following operations and represent in the same form :

(i) $(10101100)_2 - (00111000)_2$

(ii) $(3987)_{16} - (2111)_{16}$.

**Soln. :**

(i)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| − | | | | | | | | |
| | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| | [−1] | [−1] | [−1] | | | | | |
| | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

(ii)

| | | | | |
|---|---|---|---|---|
| | 3 | 9 | 8 | 7 |
| − | | | | |
| | 2 | 1 | 1 | 1 |
| | 1 | 8 | 7 | 6 |

$\Rightarrow (1876)_{16}$

---

(b) Using 2's complement method, perform :

(i) $(99)_{10} - (156)_{10}$

(ii) $(3128)_{10} - (5692)_{10}$.

**Soln. :**

(i) $(99)_{10} = (1100011)_2 = (01100011)_2$

$(156)_{10} = (010011100)_2$

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1's complement of $(156)_{10}$ | | | | | | | | $\rightarrow$ | 0 1 1 0 0 0 1 1 |
| + | | | | | | | | 1 | + | 1 |
| | 2's complement of $(156)_{10}$ | | | | | | | | | 0 1 1 0 0 1 0 0 $\rightarrow (-156)_{10}$ |
| | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | $\rightarrow (99)_{10}$ |
| + | | | | | | | | | | |
| | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | $\rightarrow (-156)_{10}$ |
| | | [1] | [1] | | | | | | | |
| | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | $\Rightarrow (-57)_{10}$ |
| | ↑ | | | | | | | | | |

This shows answer is negative and in 2's complement form.

$\therefore$ 2's complement of $(111000111)_2 = (00111001)_2 = (57)_{10}$

(ii) $(3128)_{10} = (110000111000)_2$

$(5692)_{10} = (1011000111100)_2$

2's complement of $(5692)_{10} = 0100111000100$

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $(3128)_{10}$ | = | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| | + | | | | | | | | | | | | | |
| $(-5692)_{10}$ | = | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| | | ↑ | | | | | | | | | | | | |

$\Rightarrow (-2564)_{10}$

**This indicates answer is in 2's complement form.**

carry

Scanned by CamScanner

$\therefore \quad (+2564)_{10} = (010100000100)_2$

---

**Ex. 65 :** (a) Perform following :

(i) $(E38)_{16} - (7AE)_{16} = (?)_{16}$ (ii) $(5531)_8 -$

**Soln. :**

**(i)**

$(16)_{10}$

E 3 8 → $(14)_1$ $(3)_{10}$

0

7 A E → $(7)_{10}$ $(10)_{10}$

$(8)_{10}$

1 1

$14 - 8 = (6)_{10}$ $19 - 11$

↓

6

$\therefore$ **Ans.** $= (68A)_{16}$ .

**(ii)**

(8)

5 5 3

3 2 6

(3)

1

Scanned by CamScanner

(ii)

$$\begin{array}{rr} & (54)_{10} \\ - & (15)_{10} \\ \hline & (19)_{10} \end{array} \Rightarrow (13)_{16}$$

(c) Express the following numbers to sign magnitude, 1's and 2's complement form :

(i) – 67, (ii) – 102.

**Soln. :** (i) $(+67)_{10} = (1000011)_2$, $(102)_{10} = (1100110)$

In sign magnitude $(-67)_{10}$ = 1 (↑ sign bit (Negative)) 1 0 0 0 0 1 1 (Magnitude)

$(-102)_{10}$ = 1 (↑ sign bit) 1 1 0 0 1 1 0 (Magnitude)

(ii) In 1C code

$(-67)_{10} = (1011\ 1100)_2$

$(-102)_{10} = (1001\ 1001)_2$

(iii) In 2C code

$$\begin{array}{rr} (-67)_{10} = & 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0 \\ + & 1 \\ \hline & (1\ 0\ 1\ 1\ 1\ 1\ 0\ 1)_2 \end{array} \qquad \begin{array}{rr} (-102)_{10} = & 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1 \\ + & 1 \\ \hline & (1\ 0\ 0\ 1\ 1\ 0\ 1\ 0)_2 \end{array}$$

**Ex. 66 :** (a) After converting to binary, perform the following subtraction by using 2's complement and give the answer in hexadecimal form :

(i) $(ABC)_{16} - (476)_8$

(ii) $(31)_{10} - (52)_8$.

**Soln. :** (i) $(ABC)_{16} = (1010\ 1011\ 1100)_2$

$(476)_8 = (0001\ 0111\ 1110)_2$

$$\begin{array}{rr} & 1'C \text{ of } (476)_8 \\ + & 1 \\ \hline & 2'C \text{ of } (476)_8 \end{array} \rightarrow \begin{array}{r} 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1 \\ + \quad 1 \\ \hline (1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0)_2 \end{array} \rightarrow (-476)_8$$

$$\begin{array}{lcccccccccccc} (ABC)_{16} \rightarrow & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ + \quad + & & & & & & & & & & & & \\ (-476)_8 \rightarrow & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ \boxed{1} & \boxed{1} & \boxed{1} & & \boxed{1} & & & & & & & & \\ \hline & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ & & 9 & & & 7 & & & & E & & & \end{array} \Rightarrow (97E$$

discard carry

Scanned by CamScanner

(ii) $(31)_{10} = (11111)_2$

$(52)_8 = (101010)_2$

1'C of $(52)_8$ → 0 1 0 1 0 1

\+ 1 → 1

2'C of $(52)_8$ → 0 1 0 1 1 0 ⇒ $(-52)_8$

0 0 1 1 1 1 1 → $(-31)_8$

\+

1 0 1 0 1 1 0 → $(-52)_8$

1 1 1 1 1

1 1 1 0 1 0 1 → Take 2'c complement

∴ 0 0 0 1 0 1 1 → 2C of the answer ⇒ (0BH)

∴ 1 1 1 0 1 0 1 ⇒ (− 0 B H).

---

(b) After converting to binary perform following subtraction using 2's complement form $(356)_{10} - (62)_{10}$

**Soln. :**

| 2 | 356 | |
|---|---|---|
| | Quo. | Rem. |
| 2 | 178 | 0 → LSB |
| 2 | 89 | 0 |
| 2 | 44 | 1 |
| 2 | 22 | 0 |
| 2 | 11 | 0 |
| 2 | 5 | 1 |
| 2 | 2 | 1 |
| 2 | 1 | 0 |
| | 1 | → MSB |

| 2 | 62 | |
|---|---|---|
| | Quo. | Rem. |
| 2 | 31 | 0 → LSB |
| 2 | 15 | 1 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |
| | 1 | → MSB |

$(356)_{10}$ = $(0001\ 0110\ 0100)_2$

Added zeroes to make 12 bit

$(62)_{10}$ = $(111110)_2$

= $(0000\ 0011\ 1110)_2$

1 C of $(62)_{10}$ → 1 1 1 1 1 1 0 0 0 0 0 1

\+ \+

1 → 1

2 C of (62) 1 1 1 1 1 1 0 0 0 0 1 0

$(356)_{10}$ → 0 0 0 1 0 1 1 0 0 1 0 0

\+ \+

$(-62)_{10}$ → 1 1 1 1 1 1 0 0 0 0 1 0

1 1 1 1 1 1

Discard 0 0 0 1 0 0 1 0 0 1 1 0 ⇒ $(+294)_{10}$

Scanned by CamScanner

**Ex. 67 :** Perform following :

(i) $(36)_8 \times (22)_8$

(ii) $(43)_8 \times (12)_8$

(iii) $(63)_8 \times (21)_8$

(iv) $(DA)_{16} - (AF)_{16}$

(v) $(AC)_{16} - (8F)_{16}$

(vi) $(8F)_{16} - (CA)_{16}$

**Soln. :** (i) $(36)_8 = (011\ \ 110)_2 = (11110)_2$

$(22)_8 = (010\ \ 010)_2 = (10010)_2$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 1 | 1 | 1 | 0 |
| | | | | | × | 1 | 0 | 0 | 1 | 0 |
| | | | | | | 0 | 0 | 0 | 0 | 0 |
| | | | | | 1 | 1 | 1 | 1 | 0 | X |
| | | | | 0 | 0 | 0 | 0 | 0 | X | X |
| | | | 0 | 0 | 0 | 0 | 0 | X | X | X |
| | | 1 | 1 | 1 | 1 | 0 | X | X | X | X |
| carry → | | 1 | 1 | 1 | 1 | | | | | |
| | | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| | | ↓ | | ↓ | | ↓ | | | ↓ | |
| | | 1 | | 0 | | 3 | | | 4 | |

$\Rightarrow (1034)_8$

(ii) $(43)_8 = (100\ \ 011)_2$

$(12)_8 = (001\ \ 010)_2 = (1010)_2$

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 0 | 0 | 0 | 1 | 1 | | |
| | | | × | | | 1 | 0 | 1 | 0 | | |
| | | | | 0 | 0 | 0 | 0 | 0 | 0 | | |
| | | | 1 | 0 | 0 | 0 | 1 | 1 | X | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | X | X | | |
| | 1 | 0 | 0 | 0 | 1 | 1 | X | X | X | | |
| | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | | |
| | | 5 | | 3 | | | 6 | | | $(536)_8$ | |

(iii) $(63)_8 = (110\ \ 011)_2$

$(21)_8 = (010\ \ 001)_2 = (10001)_2$

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 1 | 0 | 0 | 1 | 1 | | |
| | | | × | | | 1 | 0 | 0 | 0 | 1 | |
| | | | | 1 | 1 | 0 | 0 | 1 | 1 | | |
| | | | 0 | 0 | 0 | 0 | 0 | 0 | X | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | X | X | | |
| | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | | |
| 1 | 1 | 0 | 0 | 1 | 1 | X | X | X | X | | |
| | | | 1 | 1 | | | | | | | |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | |
| 1 | | 5 | | 4 | | | 3 | | | $\Rightarrow (1543)_8$ | |

Scanned by CamScanner

(iv)

$$
\begin{array}{lcl}
D\ A & \rightarrow & (13)_{10} \quad (16)_{10} \quad (10)_{10} \quad (26)_{10} \\
-\ A\ F & \rightarrow & (10)_{10} \quad (1)_{10} \quad (11)_{10} \quad (15)_{10} \\
\hline
(2\ B)_{16} & & 13 - 11 = (2)_{10} \quad 26 - 15 = (11)_{10}
\end{array}
$$

$$\downarrow \qquad\qquad \downarrow$$

$$2 \qquad\qquad B \qquad = (2B)_{16}$$

(v)

$$
\begin{array}{lcl}
A\ C & \rightarrow & (10)_{10} \quad (16)_{10} \quad (12)_{10} \quad (28)_{10} \\
-\ 8\ F & \rightarrow & (8)_{10} \quad (1) \quad (9) \quad (15)_{10} \\
\hline
 & & 10 - 9 = 1 \quad (28 - 15) = (13)_{10}
\end{array}
$$

$$\downarrow$$

$$1 \qquad = (1DH)$$

(vi)

$$
\begin{array}{lcl}
C\ A & \rightarrow & (12)_{10} \quad (10)_{10} \\
-\ 8\ F & \rightarrow & (8)_{10} \quad (15)_{10} \\
\hline
 & & 3 \quad 11 \quad \rightarrow (3B)_{16}
\end{array}
$$

$\therefore$ **Ans.** $= (-3B)_{16}$.

**Ex. 68 :** Perform binary subtraction using 2's complement representation of negative number.

(i) 10111 – 11100.

**Soln. :**

$$
\begin{array}{rcl}
 & 1'C \text{ of } (11100)_2 & = (00011)_2 \\
+ & 1 & + \quad 1 \\
\hline
 & 2'C \text{ OF } (11100)_2 & (00100)_2
\end{array}
$$

$$
\begin{array}{rcccccc}
 & & 1 & 0 & 1 & 1 & 1 \\
+ & & 0 & 0 & 1 & 0 & 0 \\
 & & & \boxed{1} & & & \\
\hline
 & & 1 & 1 & 0 & 1 & 1
\end{array}
$$

$\rightarrow$ **Ans.** is in 2's complement form $(-5)_{10}$

$\therefore$ $0\ 0\ 1\ 0\ 1 \rightarrow (5)_{10}$

**Ex. 69 :** Perform following :

(i) $(71.2)_8 + (33.7)_8$

(ii) $(AB2.D)_{16} + (74.A)_{16}$.

Scanned by CamScanner

**Soln. :** (i)

$$\begin{array}{ccccc} & & \boxed{1} & & \\ & 7 & 1 & . & 2 \\ \boxed{1} & 3 & 3 & . & 7 \\ \hline & 10-8=2 & 5 & . & 9-8=1 \\ \downarrow & \downarrow & \downarrow & & \downarrow \\ 1 & 2 & 5 & & 1 \end{array}$$

$\therefore$ **Ans.** $= (125.1)_8$

(ii)

$$\begin{array}{lcccc} A\ B\ 2\ .\ D & \rightarrow & (10)_{10} & (11)_{10} & (2)_{10} & (13)_{10} \\ + & & & & & \\ 7\ 4\ .\ A & \rightarrow & & (7)_{10} & (4)_{10} & (10)_{10} \\ & & \boxed{1} & & \boxed{1} & \\ \hline & & (11)_{10} & 18-16=(2)_{10} & (7)_{10} & 23-16=(7)_{10} \\ & & \downarrow & \downarrow & \downarrow & \downarrow \\ & & B & 2 & 7 & 7 \end{array} = (B27.7)_{16}$$

**Ex. 70 :** Perform following :

(i) 101101 – 1011

(ii) (1010.0111) – (0011.0110)

(iii) $(727.24)_8 - (143.4)_8$

(iv) $(A2C.6A)_{16} - (8BB.76)_{16}$

(v) $(F2F.7)_{16} - (753.A1)_{16}$.

**Soln. :**

(i)

$$\begin{array}{r} 1\ 0\ 1\ 1\ 0\ 1 \\ -\quad 1\ 0\ 1\ 1 \\ \boxed{-1} \\ \hline 1\ 0\ 0\ 0\ 1\ 0 \end{array}$$

(ii)

$$\begin{array}{r} 1\ 0\ 1\ 0\ .\ 0\ 1\ 1\ 1 \\ -\quad 0\ 0\ 1\ 1\ .\ 0\ 1\ 1\ 0 \\ \boxed{-1}\ \boxed{-1}\ \boxed{-1} \\ \hline 0\ 1\ 1\ 1\ .\ 0\ 0\ 0\ 1 \end{array}$$

(iii)

$$\begin{array}{rcccccc} & & (8) & & & (8) & \\ & 7 & 2 & 7 & . & 2 & 4 \\ - & & & & & & \\ & 1 & 4 & 3 & . & 4 & 0 \\ & (1) & & (1) & & & \\ \hline & 5 & 6 & 3 & . & 6 & 4 \end{array} \Rightarrow (563.64)_8$$

(iv)

$$\begin{array}{lcccccc} & & & (16)_{10} & & (16)_{10} & (16)_{10} \\ A\ 2\ C\ .\ 6\ A & \Rightarrow & (10)_{10} & (2)_{10} & (12)_{10} & . & (6)_{10} & (10)_{10} \\ - & & & & & & & \\ 8\ B\ B\ .\ 7\ C & \Rightarrow & (8)_{10} & (11)_{10} & (11)_{10} & . & (7)_{10} & (12)_{10} \\ & & (1)_{10} & & (1)_{10} & & (1)_{10} & \\ \hline & & (1) & (7)_{10} & 0 & . & (14)_{10} & 26-12=(14)_{10} \\ & & \downarrow & \downarrow & \downarrow & & \Downarrow & \downarrow \\ & & 1 & 7 & 0 & & E & E \end{array}$$

Scanned by CamScanner

$(F\ 2\ F\ .\ 7)_{16}$ → $(15)_{10}$ $(2)_{10}$ $(15)_{10}$ $(7)_{10}$ $(0)_{10}$

$(7\ 5\ 3\ .\ A\ 1)_{16}$ → $(7)_{10}$ $(5)_{10}$ $(3)_{10}$ $(10)_{10}$ $(1)_{10}$

7 13 11 12 15

$(7$ D B C $F)_{16}$

## 2.10 BCD Arithmetic :

Now we are going to learn addition/subtraction operation in BCD codes. You are aware that BCD means 0 – 9 represented in binary from $(0000)_2$ to $(1001)_2$. These codes are valid. The codes from $(1010)_2$ to $(1111)_2$ are ***invalid*** as far as BCD is concern.

### 2.10.1 Addition :

For adding two binary coded decimal numbers, we are going to follow the same rule i.e. binary addition (Table 2.1). Under this we are going to study important 3 cases. Presently to understand concept I am using ***single digit.*** But the same rule is valid for multi-digit BCD code or packed BCD.

**Case – I :** Add $(3)_{10}$ and $(4)_{10}$ BCD

$(4)_{10}$ → $(0\ 1\ 0\ 0)_2$

\+

$(3)_{10}$ → $(0\ 0\ 1\ 1)_2$

0 $0\ 1\ 1\ 1$

carry is '0' magnitude of the bits is < 9

**Case – II :** Add $(7)_{10}$ and $(6)_{10}$.

$(7)_{10}$ → $(0\ 1\ 1\ 1)_2$

\+

$(6)_{10}$ → $(0\ 1\ 1\ 0)_2$

0 $1\ 1\ 0\ 1$

carry is '0'

But magnitude is > 9 i.e. this answer is not having ***Valid*** BCD code.

Actually, $(7)_{10} + (6)_{10} = (13)_{10}$ and in BCD,

$(13)_{10} = (0\ 0\ 0\ 1 \quad 0\ 0\ 1\ 1)_{BCD}$

1 3

∴ Here to achieve BCD answer we have to add $(6)_{10}$ or $(0110)_2$ to the answer.

$1\ 1\ 0\ 1$

\+

$0\ 1\ 1\ 0$

1 $0\ 0\ 1\ 1$

1 3

Scanned by CamScanner

Here you may ask question, why 'ONLY' 6 is added, why not any other number. The answer is very simple, if you start counting binary code i.e. $(1010)_2$ to $(1111)_2$ i.e. the codes which are ***invalid***, you will find those are '6'. Let's see the codes, 1010, 1011, 1100, 1101, 1110, 1111, all 6 are invalid. So to ***SKIP*** these states we add 6.

**Conclusion :** If we add two BCD numbers and if the answer is ***Invalid*** BCD code (i.e. > 9), simply add 6 to it.

**Case III :** Add $(9)_{10}$ and $(8)_{10}$

$(9)_{10} \rightarrow (1\ 0\ 0\ 1)_2$

$+$ $+$

$(8)_{10} \rightarrow (1\ 0\ 0\ 0)_2$

$(17)_{10}$ $\boxed{1}$ $0\ 0\ 0\ 1$

↑

carry is '1' valid BCD code

Actually we should get $(00001\ \ 01111)_{BCD}$. But the above answer shows us.

1 0001 i.e. $(11)_{10}$ which is wrong.

↑

carry

Secondly, answer is also not having invalid BCD. But as here carry is generated (Carry = 1), add (6) into it.

$$\therefore \quad \begin{array}{r} 1\ 0\ 0\ 0\ 1 \\ +\ \ 0\ 1\ 1\ 0 \\ \hline 1\ 0\ 1\ 1\ 1 \end{array} \quad \Rightarrow \quad (17)_{10}$$

**Final Conclusion :**

Studying case I/II/III finally we conclude that if answer contains invalid BCD code (i.e. number > $9)_{10}$) OR carry is generated, you add $(6)_{10}$ or $(0110)_2$. Don't add $(6)_{10}$ if number is ≤ 9 OR carry not

Scanned by CamScanner

As invalid BCD is in least significant digit add '6' only to that digit only.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| + | | | | | | | | |
| | | | | | 0 | 1 | 1 | 0 |
| | | | | $\boxed{1}$ | $\boxed{1}$ | $\boxed{1}$ | | |
| | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| | 7 | | | | 4 | | | |

$\therefore$ **Ans.** = $(74)_{10}$

(iii) Add $(82)_{10}$ and $(34)_{10}$

| | | | | |
|---|---|---|---|---|
| | $(82)_{10}$ | → | 1 0 0 0 | 0 0 1 0 |
| + | | + | | |
| | $(34)_{10}$ | → | 0 0 1 1 | 0 1 0 0 |
| | $(116)_{10}$ | | 1 0 1 1 | 0 1 1 0 |
| | | | Invalid BCD (MSD) | Valid BCD (LSD) |

As MSD has invalid BCD code add '6' only to MSD.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| + | | | | | | | | |
| | 0 | 1 | 1 | 0 | | | | |
| | $\boxed{1}$ | $\boxed{1}$ | | | | | | |
| $\boxed{1}$ | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| ↓ | ↓ | | | | ↓ | | | |
| 1 | 1 | | | | 6 | | | |

$\Rightarrow (116)_{10}$ = Answer

(iv) Add $(88)_{10}$ and $(77)_{10}$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $(88)_{10}$ | → | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| + | | + | | | | | | | | |
| | $(77)_{10}$ | → | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| | $(165)_{10}$ | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | | MSD | | | | LSD | | | |

Here MSD and LSD both contains invalid BCD, therefore add 6 to BOTH THE DIGITS.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| | $\boxed{1}$ | $\boxed{1}$ | $\boxed{1}$ | $\boxed{1}$ | $\boxed{1}$ | $\boxed{1}$ | | |
| $\boxed{1}$ | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| ↓ | ↓ | | | | ↓ | | | |
| 1 | 6 | | | | 5 | | | |

$\therefore$ **Ans.** = $(165)_{10}$

(v) Convert to BCD and add $(11)_{10} + (9)_{10}$.

Scanned by CamScanner

**Soln. :**

$(11)_{10}$ → 0 0 0 1 0 0 0 1

$(09)_{10}$ → 0 0 0 0 1 0 0 1

$(20)_{10}$ 0 0 0 1 1 0 1 0

+

0 1 1 0 → add 6 to LSD

1 1 1 1

0 0 1 0 | 0 0 0 0

↓ ↓

2 0

⇒ ∴ Ans $(20)_{10}$

(vi) Convert following decimal number to BCD. Add them. $(5337)_{10} + (7538)_{10}$.

**Soln. :**

$(5337)_{10}$ → 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 1

+

$(7538)_{10}$ → 0 1 1 1 0 1 0 1 0 0 1 1 1 0 0 0

1 1 1 1 1 1

12875 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 1

Invalid BCD codes, ∴ add 6 to it individually

1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 1

+

0 1 1 0 0 1 1 0

1 1 1 1

1 | 0 0 1 0 | 1 0 0 0 | 0 1 1 1 | 0 1 0 1

↓ ↓ ↓ ↓ ↓

1 2 8 7 5

∴ **Ans.** = $(12875)_{10} = (1\ 0010\ 1000\ 0111\ 0101)_{BCD}$.

## 2.10.2 BCD Subtraction :

For BCD subtraction nine's complement (9C) of the subtrahend is added to minuend. Say if you want to subtract $Y_3\ Y_2\ Y_1\ Y_0$ from $X_3\ X_2\ X_1\ X_0$, then first find out 9's complement of $Y_3\ Y_2\ Y_1\ Y_0$. Let's say nine's complement of $Y_3\ Y_2\ Y_1\ Y_0$ is $Z_3\ Z_2\ Z_1\ Z_0$.

Then add $Z_3\ Z_2\ Z_1\ Z_0$ with $X_3\ X_2\ X_1\ X_0$

Aim → $X_3\ X_2\ X_1\ X_0 - Y_3\ Y_2\ Y_1\ Y_0$

= $X_3\ X_2\ X_1\ X_0$ + (9's complement of $Y_3\ Y_2\ Y_1\ Y_0$)

= $X_3\ X_2\ X_1\ X_0 + Z_3\ Z_2\ Z_1\ Z_0$

**How to find 9's complement of given number ?**

To find 9's complement of given numbers we have two methods.

Scanned by CamScanner

**Method I :** Subtract given number from $(9)_{10}$, you will get 9's complement. Let's say we want to find 9C of $Y_3\ Y_2\ Y_1\ Y_0$ then operation to be performed is

$(1001)_2 - (Y_3\ Y_2\ Y_1\ Y_0) = Z_3\ Z_2\ Z_1\ Z_0$

Let's say $Y_3\ Y_2\ Y_1\ Y_0 = (0011)_2 = (3)_{10}$

then

| | | | | | |
|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | $\rightarrow (9)_{10}$ |
| − | | | | | − |
| | 0 | 0 | 1 | 1 | $\rightarrow (3)_{10}$ |
| | 0 | 1 | 1 | 0 | $\rightarrow (6)_{10}$ |

$\therefore$ $(6)_{10}$ or $(0110)_2$ is 9C of $(0011)_2$ or $(3)_{10}$.

**Method II :**

**Step I** : Find ***one's complement*** of given number.

**Step II** : Add $(10)_{10}$ or $(1010)_2$ to one's complement number.

Find 9's complement of $(0011)_2$ or $(3)_{10}$.

**Step I** : Find one's complement of $(3)_{10}$

i.e. 0 0 1 1

↓ ↓ ↓ ↓ → complement

( 1 1 0 0 )$_2$

**Step II** : Add $(10)_{10}$ or $(1010)_2$ to one's complemented code

$\therefore$

| | | | | |
|---|---|---|---|---|
| | 1 | 0 | 1 | 0 |
| + | | | | |
| | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |

↑ ignore carry — 9's complement of $(3)_{10}$ i.e. $(6)_{10}$ or $(0110)_2$

**Note :** Normally 2nd method is preferred as it is very easy to implement in digital hardware.

**Rules for BCD Subtraction :**

(1) (a) First find out 9's complement of subtrahend.

(b) Add minuend and 9's complement of subtrahend.

(c) If the sum of the numbers is an invalid BCD OR if a carry is generated from MSB, we have to add $(6)_{10}$ or $(0110)_2$. After adding $(011)_2$ if carry is generated it has to be added to sum, therefore carry is called ***end around carry***. The result we are going to get will be positive and is in its true form.

(2) (a) First find out 9's complement of subtrahend.

(b) Add minuend and 9's complemented number.

(c) If the sum of two numbers, is valid BCD, the result is negative and in its nine's complement.

Scanned by CamScanner

9's complement of $(3)_{10}$ = $(9)_{10} - (3)_{10}$

= $(6)_{10} = (0110)_2$

(2) Add minuend and 9's complemented number

$(8)_{10}$ → 1 0 0 0

+

$(-3)_{10}$ → 0 1 1 0

$(5)_{10}$ 1 1 1 0 → invalid BCD

+ 0 1 1 0 → ∴ add $(6)_{10}$ or $(0110)_2$

carry 1 1 1

0 1 0 0

(end around carry) → 1

0 1 0 1 → $(5)_{10}$

∴ As carry generated answer is in true form.

(ii) Perform $(3)_{10} - (8)_{10}$

**Soln. :** (1) 9's complement of $(8)_{10}$ = $(9)_{10} - (8)_{10}$

= $(1)_{10} = (0001)_2$

(2) Add minuend and 9's complemented number

$(3)_{10}$ → 0 0 1 1

+

$(-8)_{10}$ → 0 0 0 1

$(-5)_{10}$ 0 1 1

0 1 0 0 → $(-5)_{10}$

carry

As carry is not generated, answer is in 9's complement therefore to achieve the answer in true form, subtract answer from $(9)_{10}$ or $(1001)_2$.

$(9)_{10}$ → 1 0 0 1

−

$(4)_{10}$ → 0 1 0 0

1

0 1 0 1 → $(+5)_{10}$

∴ $(4)_{10}$ or $(0100)_2$ is 9's complement of $(5)_{10}$.

(iii) Perform $(83)_{10} - (21)_{10}$.

**Soln. :** (1) Find 9's complement of $(21)_{10}$

9's complement of $(21)_{10} = (99)_{10} - (21)_{10}$

= 1 0 0 1 1 0 0 1

−

0 0 1 0 0 0 0 1

0 1 1 1 1 0 0 0 → $(78)_{10}$ (BCD)

Scanned by CamScanner

(2) Add minuend and 9's complemented number

$(83)_{10}$ → 1 0 0 0 0 0 1 1

+

$(-21)_{10}$ → 0 1 1 1 1 0 0 0

$(+62)_{10}$ 1 1 1 1 1 0 1 1 → invalid BCD

0 1 1 0 0 1 1 0 → add 6

carry → [1] 0 1 1 0 0 0 0 1

1

0 1 1 0 0 0 1 0 → $(62)_{10}$ BCD

(iv) Perform $(21)_{10} - (83)_{10}$

**Soln. :** (1) Find 9's complement of $(83)_{10}$

$= (99)_{10} - (83)_{10} = (16)_{10} = (0001 \quad 0110)_{BCD}$

(a) Add both the numbers

$(21)_{10}$ → 0 0 1 0 0 0 0 1

+

$(-83)_{10}$ → 0 0 0 1 0 1 1 0

$(-62)_{10}$

carry → [0] 0 0 1 1 0 1 1 1

As carry is not generated answer is negative and in its 9's complement form

$\therefore \quad (00110111) = (37)_{10}$

$\therefore \quad (99)_{10} - (37)_{10} = (62)_{10} = (0110 \quad 0010)_{10}$

$(37)_{10}$ is 9's complement of $(62)_{10}$

## 2.11 XS–3 Arithmetic :

The steps involved In XS – 3 additions are as follows :

(i) Convert given number to equivalent XS – 3 format.

(ii) Now add both, XS - 3 numbers.

(iii) After adding if carry is generated, add $(3)_{10}$ or $(0011)_2$ to the sum of two digits.
If carry not generated subtract $(3)_{10}$ or $(0011)_2$ from sum of two digits.

(iv) Subtraction of $(3)_{10}$ is equivalent to adding $(13)_{10}$ or $(1101)_2$.

Let's see some examples.

(i) Add $(3)_{10}$ and $(5)_{10}$ in XS – 3 .

**Soln. :** $(3)_{10}$ in XS – 3 is $(0110)_2$ $\left(\text{or } (6)_{10}\right)$

$(5)_{10}$ in XS – 3 is $(100)_2$ $\left(\text{or } (8)_{10}\right)$

$(3)_{10}$ → 0 1 1 0

+

$(5)_{10}$ → 1 0 0 0

$(8)_{10}$ [0] 1 1 1 0

↑

Carry = 0, ∴ subtract $(3)_{10}$ from the sum

Scanned by CamScanner

$$\begin{array}{r} 1\ 1\ 1\ 0 \\ -\ 0\ 0\ 1\ 1 \\ \hline 1\ 0\ 1\ 1 \end{array} \Rightarrow (8)_{10} \text{ in XS-3}$$

OR

$$\begin{array}{r} 1\ 1\ 1\ 0 \\ +\ \boxed{1}\ 1\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 1 \end{array}$$

ignore

**Note :** $(1101)_2$ is nothing but $(-3)_{10}$ in 2's complement form.

(ii) Add $(8)_{10}$ and $(4)_{10}$.

**Soln. :** $(8)_{10}$ in XS–3 $\rightarrow (1011)_2$

$(4)_{10}$ in XS–3 $\rightarrow (0111)_2$

Add both.

$$\begin{array}{rcl} (8)_{10} & \rightarrow & 1\ 0\ 1\ 1 \\ + & & \\ (4)_{10} & \rightarrow & 0\ 1\ 1\ 1 \\ \hline (12)_{10} & & \boxed{1}\ \boxed{1}\ \boxed{1} \\ & & \boxed{1}\ 0\ 0\ 1\ 0 \end{array}$$

As carry = 1 add $(0011)_2$

carry

$$\begin{array}{r} 0\ 0\ 1\ 1 \\ \hline 0\ 1\ 0\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0\ 0\ 1\ 1 \\ \hline 0\ 1\ 0\ 1 \\ \hline 2 \end{array}$$

$\rightarrow$ XS-3 code for 1 and 2

$\rightarrow (12)_{10}$ in XS-3

(iii) Add $(38)_{10}$ and $(44)_{10}$.

**Soln. :**

$$\begin{array}{rcl} & & \text{MSD} \quad \text{LSD} \\ (38)_{10} & \rightarrow & 0\ 1\ 1\ 0 \quad 1\ 0\ 1\ 1 \\ + & & \\ (44)_{10} & \rightarrow & 0\ 1\ 1\ 1 \quad 0\ 1\ 1\ 1 \\ 82 & \rightarrow & \boxed{0}\ \boxed{1}\ \boxed{1}\ \boxed{1}\ \boxed{1} \quad \boxed{1}\ \boxed{1}\ \boxed{1} \\ & & 1\ 1\ 1\ 0 \quad \boxed{C=1} \quad 0\ 0\ 1\ 0 \\ & & + \\ & & 1\ 1\ 0\ 1 \quad 0\ 0\ 1\ 1 \\ \text{ignore} & \rightarrow & \boxed{1} \quad (1\ 0\ 1\ 1)_2 \quad (0\ 1\ 0\ 1)_2 \\ & & 1\ 0\ 1\ 1 \quad 0\ 1\ 0\ 1 \end{array} \Rightarrow (82)_{10} \text{ in XS-3}$$

Here MSD doesn't give carry ∴ subtract $(3)_{10}$

As LSD provides carry, add $(3)_{10}$ or (0011)

## 2.12 XS–3 Subtraction :

The steps involved in XS–3 subtraction are as follows :

(i) Convert given number to its equivalent XS–3 format.

(ii) Complement the subtrahend (find 1C of subtrahend).

(iii) Add 1C of subtrahend and minuend.

(iv) If end around carry is generated

(a) Result is positive

(b) Add carry also.

Scanned by CamScanner

(c) If carry generated from digit add $(3)_{10}$ else add $(13)_{10}$.

else (a) Result is negative

(b) You can add carry also.

(c) If carry generated from digit, add $(3)_{10}$ else add $(3)_{10}$.

(d) Take 1's complement (1C) of the result.

Let's see some examples.

(i) Perform $(8)_{10} - (3)_{10}$ in XS–3.

**Soln. :** $(8)_{10} \rightarrow (1\ 0\ 1\ 1)_2$

$(3)_{10} \rightarrow (0\ 1\ 1\ 0)_2$

1's complement of $(3)_{10} = (1001)_2$

Add $(8)_{10} \rightarrow$ 1 0 1 1

\+

$(-3)_{10} \rightarrow$ 1 0 0 1

1 1

carry → [1] 0 1 0 0

1

0 1 0 1 Add $(3)_{10}$

0 0 1 1

[1] [1] [1]

1 0 0 0 ⇒ $(5)_{10}$ in XS–3

(ii) Perform $(3)_{10} - (8)_{10}$

**Soln. :** $(3)_{10} - (8)_{10} = (3)_{10} + (-8)_{10}$

∴ $(3)_{10} \rightarrow (0110)$ in XS–3

$(8)_{10} \rightarrow (1011)$ in XS–3

1's complement of $(8)_{10} \rightarrow (0100)$ XS–3

Add

$(3)_{10} \rightarrow$ 0 1 1 0

\+

$(-8)_{10} \rightarrow$ 0 1 0 0

[1]

$(-5)_{10}$ [0] 1 0 1 0

no carry → 0

1 0 1 0

\+ 1 1 0 1

ignore carry → [1] 0 1 1 1

↓ ↓ ↓ ↓

$(-5)_{10}$ in XS-3 (1C form) 1 0 0 0

no carry add $(13)_{10}$

now take 1's complement

⇒ $(5)_{10}$ in XS–3

Scanned by CamScanner

(iii) Perform $(53)_{10} - (28)_{10}$ in XS–3.

**Soln. :** $(53)_{10} = 1000\ \ 0110 \rightarrow$ XS–3

$(25)_{10} = 0101\ \ 1011 \rightarrow$ XS–3

1'C of $(28)_{10} = 1010\ \ 0100$

| | | | MSD | | LSD | |
|---|---|---|---|---|---|---|
| + | $(53)_{10}$ | → | 1 0 0 0 | | 0 1 1 0 | |
| | $(-28)_{10}$ | → | 1 0 1 0 | | 0 1 0 0 | |
| | $(25)_{10}$ | | | | [1] | |
| | | [1] | 0 0 1 0 | | 1 0 1 0 | |
| | end around | → | | | 1 | |
| | | | 0 0 1 0 | | 1 0 1 1 | To MSD add $(3)_{10}$ |
| | | + | 0 0 1 1 | | 1 1 0 1 | To LSD add $(13)_{10}$ |
| | ignore carry | → [1] | 0 1 0 1 | | 1 0 0 0 | $\Rightarrow$ XS – 3 for $(25)_{10}$ |
| | | | 2 | | 5 | |

## Exercise

1. Perform addition operation without converting : $A6_H + 2B_H$ **(May 96, 4 Marks)**
2. Represent $(39)_{10}$ in : **(May 96, 4 Marks)**
   (i) 1's complement representation
   (ii) 2's complement representation
3. Perform addition without converting : $B7_H + 8A_H$ **(Dec. 96, 4 Marks)**
4. Represent $(67)_{10}$ in: **(Dec. 96, 4 Marks)**
   (i) 1's complement representation
   (ii) 2's complement representation
5. Perform addition without converting **(May 97, 4 Marks)**
   $A5_H + B6_H$
6. Represent $(67)_{10}$ in : **(May 97, 4 Marks)**
   (i) 1's complement represenation (ii) 2's complement representation.
7. Perform addition without converting : **(Dec. 97, 4 Marks)**
   $CA_H + 6E_H$
8. Represent $(89)_{10}$ in : **(Dec. 97, 4 Marks)**
   (i) 1's complement representation (ii) 2's complement representation.
9. Perform the following operations directly without converting to decimal : **(May 98, 9 Marks)**
   (i) $(112)_4 \times (211)_4$ (ii) $(BC)_H - (5D)_H$ (iii) $(176)_8 + (532)_8$
10. Represent $(61)_{10}$ in : **(May 98, 4 Marks)**
    (i) 1's complement representation
    (ii) 2's complement representation.

Scanned by CamScanner

## 2.13 Examples from University Papers :

**1. Perform addition operation without converting : $A6_H + 2B_H$.** **(May 96, 4 Marks)**

**Ans. :**

$$\begin{array}{rlll} & A6 \rightarrow & (10)_{10} & (6)_{10} \\ + & 2B \rightarrow & (2)_{10} & (11)_{10} \\ \hline & & (12)_{10} & (17)_{10} \end{array}$$

} Convert each digit here to equivalent decimal

$\therefore$ $(17)_{10} = 17 - 16 = 1$ with carry (1)

We have,

$$\begin{array}{rll} & (12)_{10} & (17)_{10} \\ + & 1 & \\ \hline & (13)_{10} & (1)_{16} \end{array}$$

Now $(13)_{10} = (D)_{16}$

$\therefore$ Total answer of the sum is

$= (D1)_{16}$

$\therefore$ $\boxed{(A6)_H + (2B)_H = (D1)_H}$ ...Ans.

**2. Represent $(39)_{10}$ in :** **(May 96, 4 Marks)**

**(i) 1's complement representation**

**(ii) 2's complement representation.**

**Ans. :** (i) 1's complement representation of $(39)_{10}$

First find out binary equivalent of $(39)_{10}$

| 2 | 39 → Original number | |
|---|---|---|
| | Quotient | Remainder |
| 2 | 19 | 1 → LSB |
| 2 | 9 | 1 |
| 2 | 4 | 1 |
| 2 | 2 | 0 |
| 2 | 1 | 0 |
| | 1 → MSB | |

$\therefore$ $(39)_{10} = (100111)_2$

Now 1's complement of $\quad 1\ 0\ 0\ 1\ 1\ 1$

$\downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow$

$= \quad 0\ 1\ 1\ 0\ 0\ 0$

$\therefore$ 1's complement representation of

$\boxed{(39)_{10} = (011000)_2}$ ...Ans.



Scanned by CamScanner

(ii) 2's complement of $(39)_{10}$ is

1's complement of $(39)_{10}$ + 1

$$\therefore \quad \begin{array}{cccccccc} & 0 & 1 & 1 & 0 & 0 & 0 & \rightarrow \text{1's complement} \\ + & & & & & & 1 & \\ \hline & 0 & 1 & 1 & 0 & 0 & 1 & \rightarrow \text{2's complement} \end{array}$$

$\therefore$ 2's complement representation of $(39)_{10} = 011001$

...Ans.

**3. Perform addition without converting : $B7_H + 8A_H$** **(Dec. 96, 4 Marks)**

**Ans. :**

$$\begin{array}{lll} B7 & \rightarrow & (11)_{10} \quad (7)_{10} \\ 8A & \rightarrow & (8)_{10} \quad (10)_{10} \\ \hline & & (19)_{10} \quad (17)_{10} \end{array}$$

$$= (19)_{10} \quad (17)_{10}$$

$$\downarrow$$

$$17 - 16 = 1 \text{ with carry } 1$$

$\therefore$ $19 + 1 = (20)_{10}$

$= (20)_{10}$

$\downarrow$

$(20)_{10} - (16)_{10} = (4)_{16}$ with carry 1

$\therefore$ Total answer of sum $= (141)_H$

$\therefore$ $\boxed{(B7)_H + (8A)_H = (141)_H}$

...Ans.

**4. Represent $(67)_{10}$ in** **(Dec. 96, May 97, 4 Marks)**

**(i) 1's complement representation**

**(ii) 2's complement representation.**

**Ans. :** (i) 1's complement representation of $(67)_{10}$

First find out binary equivalent of $(67)_{10}$

| 2 | 67 → Original number | |
|---|---|---|
| | Quotient | Remainder |
| 2 | 33 | 1 → LSB |
| 2 | 16 | 1 |
| 2 | 8 | 0 |
| 2 | 4 | 0 |
| 2 | 2 | 0 |
| 2 | 1 | 0 |
| | | 1 → MSB |

$\therefore$ $(67)_{10} = (1000011)_2$

Scanned by CamScanner

Now 1's complement of

| 1 | 0 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |

$\therefore$ 1's complement of $(67)_{10} = (0111100)_2$

(ii) 2's complement of $(67)_{10}$ is ...Ans

1's complement of $(67)_{10}$ + 1

| $\therefore$ | 0 | 1 | 1 | 1 | 1 | 0 | 0 | → 1's complement |
|---|---|---|---|---|---|---|---|---|
| + | | | | | | | 1 | |
| | 0 | 1 | 1 | 1 | 1 | 0 | 1 | → 2's complement |

$\therefore$ 2's complement representation of

$$(67)_{10} = (0111101)_2$$

...Ans.

**5. Perform addition without converting : $A5_H + B6_H$** **(May 97, 4 Marks)**

**Ans. :** $(A5)_H + (B6)_H$

A 5 → $(10)_{10}$ $(5)_{10}$

B 6 → $(11)_{10}$ $(6)_{10}$

Carry 1

1 21 − 16 = 5 $(11)_{10}$

$(1)_{16}$ $(5)_{16}$ $(B)_{16}$

$= (15B)_{16}$

$\therefore$ $(A5)_H + (B6)_H = (15B)_H$ ...Ans.

**6. Perform addition without converting : $CA_H + 6E_H$** **(Dec. 97, 4 Marks)**

**Ans. :** $(CA)_H + (6E)_H$

C A → $(12)_{10}$ $(10)_{10}$

6 E → $(6)_{10}$ $(14)_{10}$

Carry 1 1

1 19 − 16 = 3 24 − 16 = 8

$(1)_{16}$ $(3)_{16}$ $(8)_{16}$

$= (138)_{16}$

Scanned by CamScanner

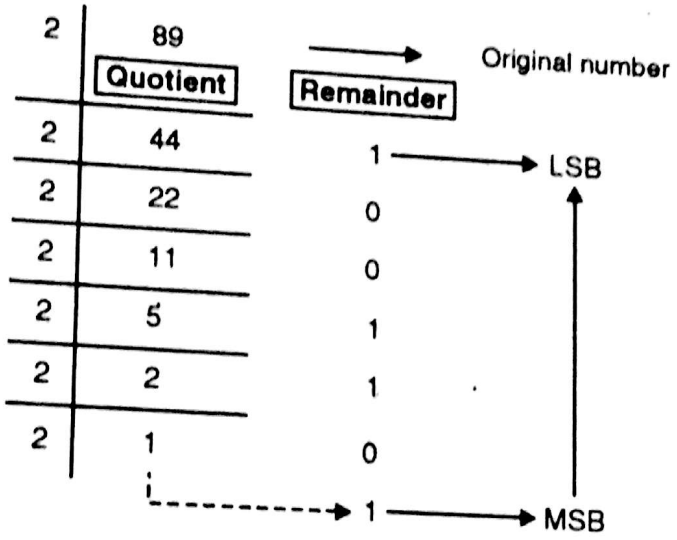
$\therefore$ $\boxed{(CA)_H + (6E)_H = (138)_H}$

**7. Represent $(89)_{10}$ in :**

**(i) 1's complement representation (ii) 2's complement representation.** **(Dec. 97, 4 Marks**

**Ans. :** (i) 1's complement representation of $(89)_{10}$.

First find out binary equivalent of $(89)_{10}$.




$\therefore$ $(89)_{10} = (1011001)_2$

1's complement of 1 0 1 1 0 0 1

↓ ↓ ↓ ↓ ↓ ↓ ↓

0 1 0 0 1 1 0

$\boxed{\therefore \text{ 1's complement representation of } (89)_{10} = (0100110)_2}$ **...Ans.**

(ii) 2's complement of $(89)_{10}$ is

1's complement of $(89)_{10}$ + 1

$\therefore$ 0 1 0 0 1 1 0 ← 1's complement

\+ 1

0 1 0 0 1 1 1 ← 2's complement

$\boxed{\therefore \text{ 2's complement representation of } (89)_{10} = (0100111)_2}$ **...Ans.**

**8. Perform the following operations directly without converting to decimal : (May 98, 9 Marks)**

**(i) $(112)_4 \times (211)_4$ (ii) $(BC)_H - (5D)_H$ (iii) $(176)_8 + (532)_8$**

**Ans.:** (i) $(112)_4 \times (211)_4$

First convert these numbers into binary form

As base is 4 each digit in number is represented using 2 bits.

$\therefore$ $(112)_4 = (010110)_2$

$(211)_4 = (100101)_2$

Scanned by CamScanner

Perform binary multiplication

```
                0 1 0 1 1 0
              ×
                1 0 0 1 0 1
              -------------
                0 1 0 1 1 0
              0 0 0 0 0 0 X
            0 1 0 1 1 0 X X
          0 0 0 0 0 0 X X X
        0 0 0 0 0 0 X X X X
      0 1 0 1 1 0 X X X X X
          [1] [1]   [1]
      -----------------------
      0 1 1 0 0 1 0 1 1 1 0
          3   0   2   3   2
```

Convert the binary number to base 4 number, group 2 bits

$\boxed{\therefore (112)_4 \times (211)_4 = (30232)_4}$ ...Ans.

(ii) $(BC)_H - (5D)_H$

```
    A 1
  - B C  ⇒   A   1C
    5 D  ⇒   5   D
             -----
             5   F
```

$= (5F)_H$

$\boxed{\therefore \quad (BC)_H - (5D)_H = (5F)_H}$ ...Ans.

(iii) $(176)_8 + (532)_8$

```
    1 7 6
  + 5 3 2
    [1] [1]
  -------
    7 3 0
```

$\boxed{\therefore (176)_8 + (532)_8 = (730)_8}$ ...Ans.

**9. Represent $(61)_{10}$ in :** **(May 98, 4 Marks)**

**(i) 1's complement representation**

**(ii) 2's complement representation**

**Ans. :**

**(i) 1's complement representation of $(61)_{10}$**

Scanned by CamScanner

| | Quotient | Remainder |
|---|---|---|
| 2 | 61 | 1 |
| 2 | 30 | 0 |
| 2 | 15 | 1 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |

Original number → LSB … MSB

$\therefore (61)_{10} = (111101)_2$

1's complement of 1 1 1 1 0 1

↓ ↓ ↓ ↓ ↓ ↓

0 0 0 0 1 0

**$\therefore$ 1's complement representation of $(61)_{10} = (000010)_2$** ...Ans.

(ii) 2's complement representation of $(61)_{10}$ is

1's complement of $(61)_{10}$ + 1

1's complement is 0 0 0 0 1 0

\+ 1

0 0 0 0 1 1 → 2's complement

**$\therefore$ 2's complement representation of $(61)_{10} = (000011)_2$** ...Ans.

**10. Add the two numbers $(AC)_H$ and $(D9)_H$ directly. Convert the hexadecimal system obtained, in a number system with radix four.** **(Dec. 98, 4 Marks)**

**Ans. :**

$(AC)_H + (D9)_H$

AC → $(10)_{10}$ $(12)_{10}$

D9 → $(13)_{10}$ $(9)_{10}$

Carry 1 1

1 24 − 16 = 8 21 − 16 = 5

$(1)_{16}$ $(8)_{16}$ $(5)_{16}$

$= (185)_{16}$

$\therefore (AC)_H + (D9)_H = (185)_H$

**Now to convert $(185)_H$ to a number system with radix 4.**

Scanned by CamScanner

First we have to convert $(185)_H$ to decimal form,

$$= 1 \times 16^2 + 8 \times 16^1 + 5 \times 16^0$$

$$= 256 + 128 + 5$$

$$= (389)_{10}$$

$$\therefore \quad (185)_{16} = (389)_{10}$$

Now convert $(389)_{10}$ to number system with radix 4 by division method.

| 4 | 389 | → | Original number |
|---|---|---|---|
| | Quotient | Remainder | |
| 4 | 97 | 1 → | LSB |
| 4 | 24 | 1 | |
| 4 | 6 | 0 | |
| 4 | 1 | 2 | |
| | ↓ | 1 → | MSB |

$$\therefore \quad (389)_{10} = (12011)_4$$

$$\therefore \quad (AC)_H + (D9)_H = (185)_H$$

$$(185)_H = (389)_{10} = (12011)_4$$

...Ans.

**11. Two binary numbers A and B are added and subtracted as shown : (Dec. 98, 2 Marks)**

$$(A)_2 + (B)_2 = (1100)_2$$

$$(A)_2 - (B)_2 = (0010)_2$$

**then find $(A)_2$ and $(B)_2$.**

**Ans. :** To get the values of $(A)_2$ and $(B)_2$ ; First add the two equations

$$(A)_2 + (B)_2 = 1100 \quad \rightarrow \text{eqn. (a)}$$

$$+ \quad (A)_2 - (B)_2 = 0010 \quad \rightarrow \text{eqn. (b)}$$

$$2(A)_2 = 1110$$

$$\therefore \quad 2(A)_2 = (1110)_2$$

Now to get $(A)_2$, $(1110)_2 = (14)_{10}$

$$\therefore \quad (A)_{10} = 14/2 = (7)_{10}$$

$\therefore$ In Binary value of

$$(A)_2 = (0111)_2$$

...Ans.

Now substitute this value in Equation (a)

$$\therefore \quad (0111)_2 + (B)_2 = (1100)_2,$$

Scanned by CamScanner

$$(B)_2 = \begin{array}{cccc} 1 & 1 & 0 & 0 \\ - \; 0 & 1 & 1 & 1 \\ \boxed{-1} & \boxed{-1} & \boxed{-1} & \\ \hline 0 & 1 & 0 & 1 \end{array}$$

$\therefore \quad (B)_2 = (0101)_2$ ...Ans.

**12. Represent $(38)_{10}$ in 2's complement representation.** **(June 99, 1 Mark)**

**Ans. :** First find out 1's complement of $(38)_{10}$.

| 2 | 38 → Quotient | Remainder | Original number |
|---|---|---|---|
| 2 | 19 | 0 → | LSB |
| 2 | 9 | 1 | |
| 2 | 4 | 1 | |
| 2 | 2 | 0 | |
| 2 | 1 | 0 | |
| | | 1 → | MSB |

$\therefore \quad (38)_{10} = (100110)_2$

1's complement $\quad \begin{array}{cccccc} 1 & 0 & 0 & 1 & 1 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}$

$= (011001)_2$

Now 2's complement of $(38)_{10}$ is 1's complement of $(38)_{10} + 1$

$\therefore$

$$\begin{array}{cccccc} 0 & 1 & 1 & 0 & 0 & 1 \rightarrow \text{1's complement} \\ + & & & & & 1 \\ & & & & \boxed{1} & \\ \hline 0 & 1 & 1 & 0 & 1 & 0 \rightarrow \text{2's complement} \end{array}$$

$\therefore$ 2's complement of $(38)_{10} = (011010)_2$ ...Ans.

**(June 99, 2 Marks)**

**13. Perform the following multiplication :**

$$(10110.01)_2 \times (101.1)_2$$

**Ans.:** Refer Section 2.6 (Ex. 29). **(June 99, 2 Marks)**

**14. Perform the following division :**

$$(1101001.101) \div (101)$$

**Ans.:** Refer Section 2.7 (Ex. 36). **(June 99, 2 Marks)**

**15. Perform the following addition : $FB8_H + D89_H$.**

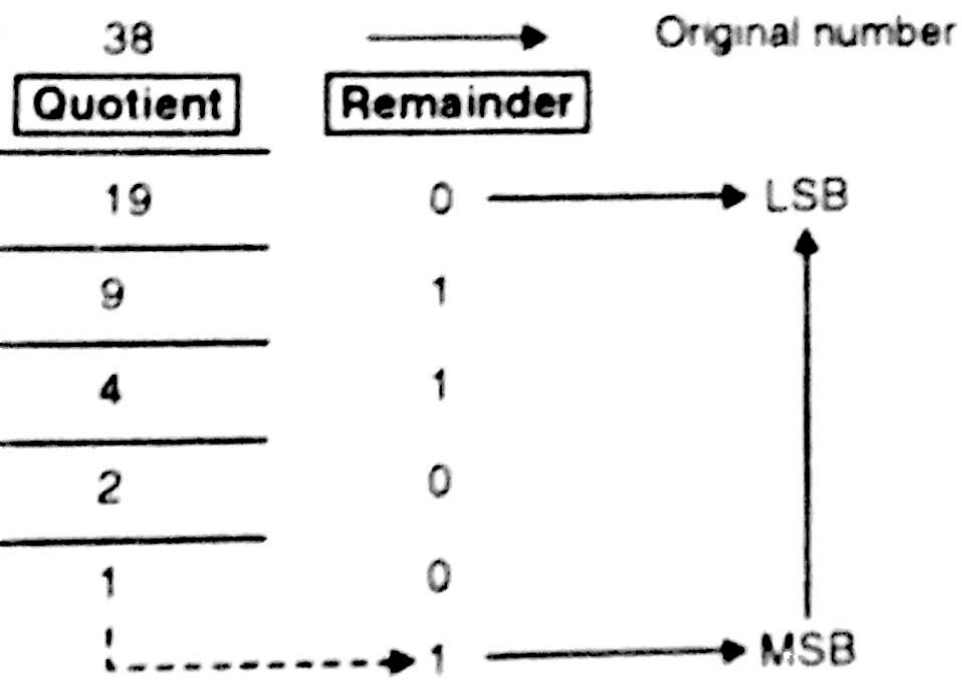
Scanned by CamScanner

$$(B)_2 = \begin{array}{cccc} 1 & 1 & 0 & 0 \\ - \; 0 & 1 & 1 & 1 \\ \boxed{-1} & \boxed{-1} & \boxed{-1} & \\ \hline 0 & 1 & 0 & 1 \end{array}$$

$= \boxed{(0101)_2}$ ...Ans.

**complement representation.** **(June 99, 1 Mark)**

mplement of $(38)_{10}$




$(100110)_2$

s complement

| 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 0 | 1 | 1 | 0 | 0 | 1 |

$(011001)_2$

$_{10}$ is 1's complement of $(38)_{10}$ + 1

1 0 0 1 → 1's complement

1

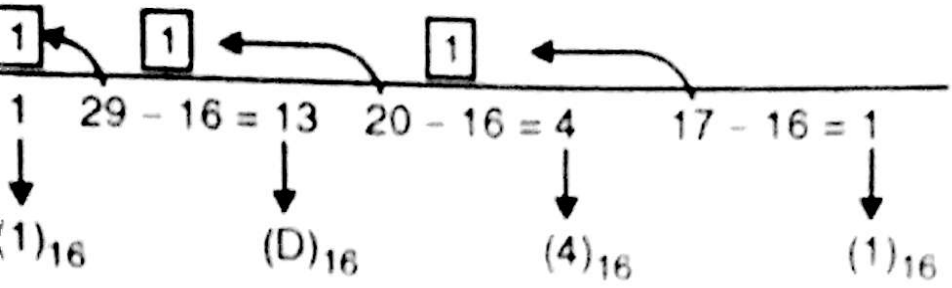
$\boxed{1}$

Scanned by CamScanner

$+ (D89)_H$

$\rightarrow$ $(15)_{10}$ $(11)_{10}$ $(8)_{10}$

$\rightarrow$ $(13)_{10}$ $(8)_{10}$ $(9)_{10}$




$(1)_{16}$ $(D)_{16}$ $(4)_{16}$ $(1)_{16}$

$D41)_{16}$

$)41)_H$

**...Ans.**

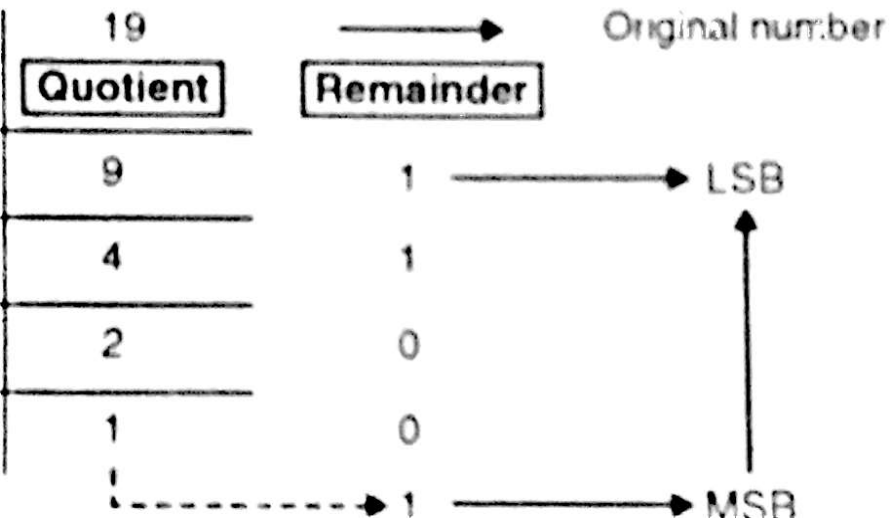
**(Dec. 99, 2 Marks)**

**resentation**

**presentation.**

representation of $(19)_{10}$

lent of $(19)_{10}$




Scanned by CamScanner

**17. Perform the following operations directly without converting to decimal :**

**(i)** $(C9)_{11} - (8D)_{11}$ **(Dec. 99, 3 Marks)**

**(ii)** $(232)_4 \times (12)_4$ **(Dec. 99, 3 Marks)**

**(iii)** $(257)_8 + (423)_8$ **(Dec. 99, 3 Marks)**

**(iv)** $(1101001)_2 \div (101)_2$ **(Dec. 99, 2 Marks)**

**Ans.: (i)**

$(C9)_H - (8D)_H$

$(C9)_H \rightarrow (12)_{10}$

$(8D)_H \rightarrow (8)_{10} + (1)_{10} = (9)_{10}$

$(16)_{10}$ → Borrow

$(25)_{10}$

$(9)_{10}$ → Operand 1

$(13)_{10}$ → Operand 2

→ Carry

$(12)_{10} - (9)_{10} = 3$ $\quad (25 - 13) = 12$

$(3)_H$ $\quad (C)_H$

$= (3C)_H$

$\therefore (C9)_H - (8D)_H = (3C)_H$ **...Ans.**

**(ii)** $(232)_4 \times (12)_4$

First convert these numbers into binary.

As base is 4 each digit in the number is represented using 2 bits.

$\therefore \quad (232)_4 = (101110)_2$

$(12)_4 = (0110)_2$

**Perform binary multiplication**

$$
\begin{array}{r}
1\ 0\ 1\ 1\ 1\ 0 \\
\times \\
\qquad 0\ 1\ 1\ 0 \\
\hline
0\ 0\ 0\ 0\ 0\ 0 \\
1\ 0\ 1\ 1\ 1\ 0\ X \\
1\ 0\ 1\ 1\ 1\ 0\ X\ X \\
0\ 0\ 0\ 0\ 0\ 0\ X\ X\ X \\
\hline
1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0
\end{array}
$$

1 1 1 1 1

Convert the binary number to base 4 number, group 2 bits

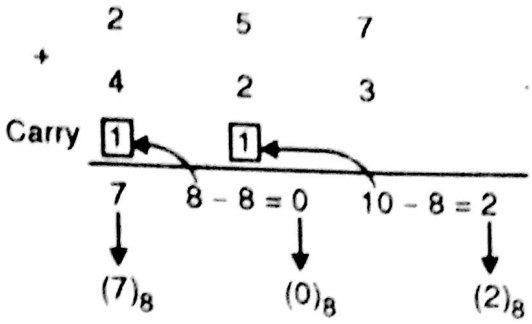
1 0 1 1 0

**...Ans.**

$\therefore (232)_4 \times (12)_4 = (10110)_4$

Scanned by CamScanner

(iii) $(257)_8 + (423)_8$




$\therefore (257)_8 + (423)_8 = (702)_8$ ...Ans.

(iv) $(1101001)_2 + (101)_2$

**Ans.:** Refer Section 2.7, Ex. 32.

Scanned by CamScanner

**Syllabus :**

Binary Logic Functions, Boolean laws, Truth tables, Associative and distributive properties, Demorgan's theorem, Realization of switching functions using logic gates.

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Introduction to Logic | 3.1 – 3.2 | ✓ | – |
| • Boolean Theorems | 3.3 | ✓ | ✓ |
| • Boolean Laws | 3.4 | ✓ | ✓ |
| • Demorgan's Theorem | 3.5 | ✓ | ✓ |
| • Perfect Induction | 3.6 | – | ✓ |
| • Reduction of logic expression using boolean algebra | 3.7 – 3.8 | – | ✓ |
| • EX-OR / EX-NOR gate | 3.9 | ✓ | ✓ |
| • Universal Logic gates | 3.10 | ✓ | ✓ |
| • Implementation using universal logic gates | 3.11 – 3.13 | – | ✓ |
| • Input bubbled logic | 3.13 | ✓ | ✓ |
| • Assertion level | 3.14 | ✓ | – |



**Comments**

- Base of digital electronics.
- Don't proceed further till your concepts are crystal clear.
- Concentriate more on section 3.3 to 3.10

Scanned by CamScanner

## 3.1 Introduction :

This chapter covers most important part in digital circuit because all digital electronic equipments are based on the use of ***digital logic*** circuits. Logic circuits are based on the use of ***logic equations***. To understand digital electronics, you must understand logic equations and the logic circuits made up of logic gates. In digital, circuits operates in binary mode i.e. '0' or '1'. This characteristic of logic circuit allows us to use logic equations. Logic equations might also be known as logical algebra, ***Boolean algebra (after the English Mathematician George Boole)***, computer algebra or computer logic. Boolean algebra is a tool for the analysis and design of digital systems.

### Basics of Logical Algebra :

Logical algebra differs from conventional algebra in three respects :

1. The symbols used in logical algebra (usually letters), do not represent numerical values.
2. Arithmetic operations are not performed in logical algebra. In Boolean algebra there are no fractions, decimals, negative numbers, square roots, cube roots, logarithms, imaginary numbers and so on.
3. Third and very important point is logical algebra allows only two possible values '0' or '1' for any variable.

Thus logical algebra is ideally suited to any system of intelligence based on two opposite states, such as ON or OFF. The Table 3.1 shows probable cases.

| Logic 0 | Logic 1 |
|---|---|
| False | True |
| OFF | ON |
| Low | High |
| No | Yes |
| Open Switch | Closed Switch |

**Table 3.1**

Thus logical algebra is well suited to express the opening and closing of electrical switches, the presence or absence of electrical pulses, or the polarity or amplitude relationship of pulses. Because only two values are possible, Boolean algebra is relatively easy to work with as compared with ordinary algebra.

Actually in Boolean algebra there are only three basic operations :

1. Logical addition, also called ***OR addition*** or ***OR operation***. The common symbol for this operation is plus sign (+).
2. Logical multiplication, also called ***AND multiplication*** or ***simply AND operation.*** The common symbol for this operation is multiplication sign (·).
3. Logical complementation or inversion, also called the ***NOT operation***. The symbol for the same is over bar ( ¯ ).

## 3.2 Logic :

Now we are going to learn three basic logical operations.

Scanned by CamScanner

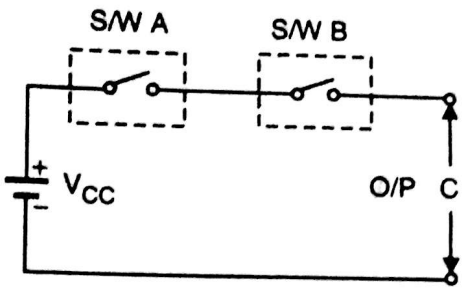
| Inputs | | Output |
|---|---|---|
| Maruti Car | Black Colour | Answer for Eligibility |
| F | F | F |
| F | T | F |
| T | F | F |
| T | T | T |

Scanned by CamScanner

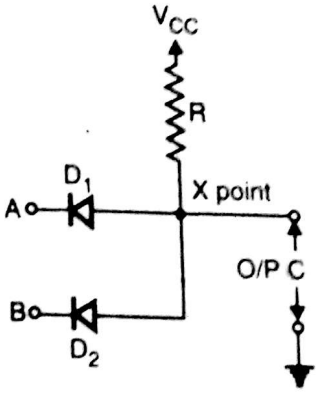
**Conclusion :** In AND function, if either of the condition is false (logic 0) then the output is false (logic 0).

In Electronics we have device which performs AND function, called AND gate. Symbol is shown in Fig. 3.1.

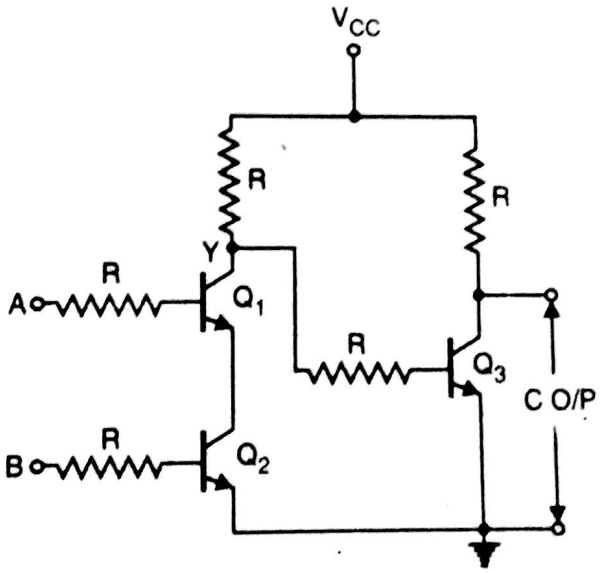
Now we will see how to implement AND function electrically i.e. using switches, diodes and transistor.




(a) AND using switches




(b) AND using diodes




(c) AND using transistor

**Fig. 3.2**

## AND Using Switches

We are going to follow standard specified in Table 3.1, i.e. logic 0 = open, logic 1 = closed. Logic 0 = LOW = 0V, logic 1 = HIGH = $V_{CC}$.

**Case I** ⇒ A = open B = open, ∴ O/P C = 0V = Logic 0.

**Case II** ⇒ A = open B = closed. but path is still incomplete. ∴ O/P C = 0V = Logic 0.

**Case III** ⇒ A = closed B = open, path is not complete, ∴ O/P C = logic 0 = 0V.

**Case IV** ⇒ A = closed, B = closed, ∴ path is complete, ∴ O/P C = $V_{CC}$ = HIGH = logic 1.

**So if we write truth table we get.**

Scanned by CamScanner

| O/P C (logic) | | A | B | O/P C |
|---|---|---|---|---|
| 0 | | 0 | 0 | 0 |
| 0 | ⇒ | 0 | 1 | 0 |
| 0 | | 1 | 0 | 0 |
| 1 | | 1 | 1 | 1 |

**(b)**

**Table 3.4**

+ 5V ⇒ HIGH.

tion correctly we will assume that dio
it, is equal to zero. Refer Fig. 3.2 (b).

diodes $D_1$ and $D_2$ both will conduct.
***forward voltage drop of the dio***
gic 0.

logic 0 B = 0V = logic '0', O/P C = 0

= +5V, B = 0V = logic 0, ∴ diode $D_1$
ge at point X is again ***forward voltag***
C = 0V = logic '0'.

Scanned by CamScanner

**Case I** → A = 0V = B = logic 0, ∴ transistor $Q_1$ and $Q_2$ both are cut-off. Voltage at point Y is logic 1 i.e. $V_{CC}$. This will drive transistor $Q_3$ in saturation, ∴ O/P C = 0V = logic 0.

**Case II** → A = logic 0 = 0V, B = logic 1 = $V_{CC}$. Transistor $Q_2$ will conduct but $Q_1$ is OFF. ∴ Again voltage at point Y is logic 1 i.e. $V_{CC}$. This will drive transistor $Q_3$ in saturation, ∴ O/P C = 0V = logic 0.

**Case III** → A = $V_{CC}$ = logic 1, B = 0V = logic 0. Transistor $Q_1$ is ON, $Q_2$ is OFF. ∴ $V_Y$ = logic 1, ∴ transistor $Q_3$ is in saturation, ∴ O/P C = 0V = logic 0

**Case IV** → A = B = $V_{CC}$ = logic 1. Both transistors $Q_1$ and $Q_2$ will be in saturation, provides $V_Y$ = 0 = logic 0. Therefore transistor $Q_3$ is cut-off. As $Q_3$ is OFF, O/P C = $V_{CC}$ = logic 1.

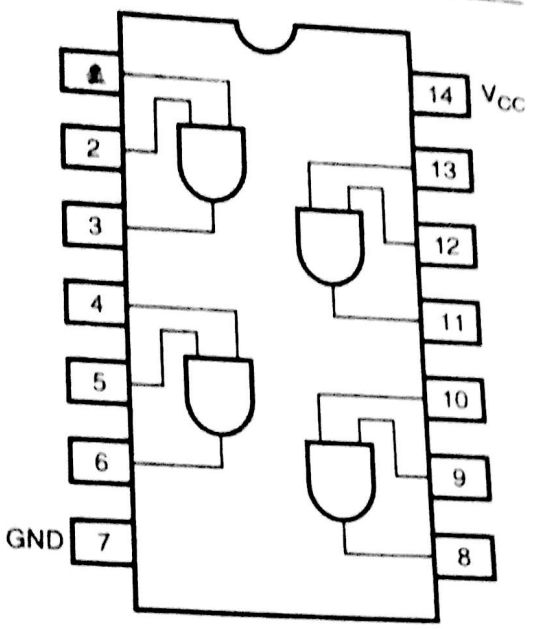
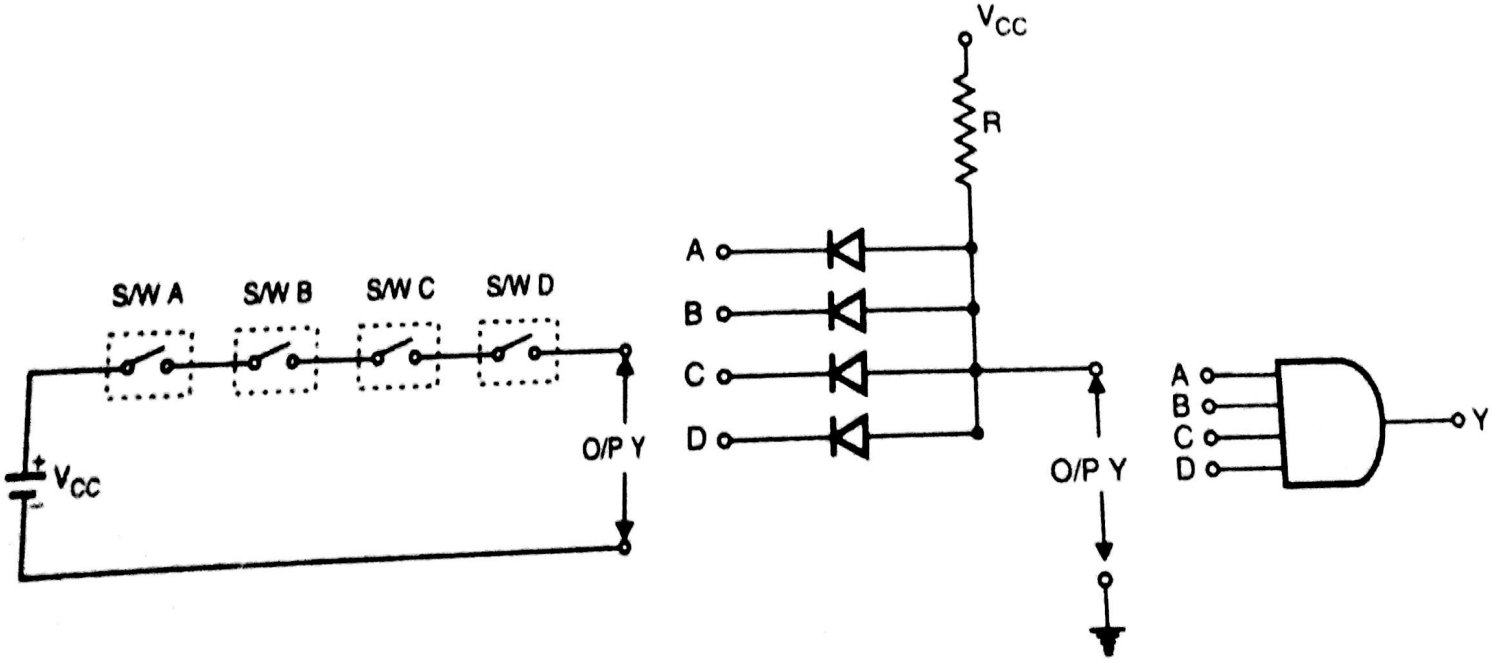



**Fig. 3.2 (A) : Pin diagram of a 7408**

If we summarise all these cases we will get truth Table as shown in Table 3.4 (b).

The AND gate we get in a standard package, i.e. Integrated Circuit (IC) of 14 pin dual-in-line (DIP) package. The same is shown in Fig. 3.2 (A) – 7408.

Uptill now what we have discussed was 2 input (i.e. A and B) AND gate. One can have two or more numbers of inputs also i.e. A, B, C, D, E, F, G, H, and so on. Table 3.5 shows truth table for 4 input AND gate. Fig. 3.3 shows 4 input AND using switches and diodes with symbol.




**(a) 4 input AND using switches** **(b) 4 input AND using diodes** **(c) Symbol**

**Fig. 3.3**

Scanned by CamScanner

| Inputs | | | | Output |
|---|---|---|---|---|
| A | B | C | D | Y |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**Table 3.5**

16 input states.

---

ic 1 then only Output is 1.

e or more inputs are '0'.

formed exactly like ordinary multipli

Scanned by CamScanner

∴ Maruti **OR** Fiat, either or both are valid.

We have 'OR' operator. Now let's make table generalised.

Maruti ⟶ Variable A

Fiat ⟶ Variable B

Required o/p ⟶ Variable C

| A | **OR** | B | = | C |
|---|---|---|---|---|
| A | + | B | = | C |

+ ⇒ OR operator (+ is Not addition)

∴ Truth Table is,

| **Inputs** | | **Output** |
|---|---|---|
| **A** | **B** | **C** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Table 3.7**




**Fig. 3.3 (A) : Symbol of OR gate**

**Conclusion :** In OR, when either of the input is HIGH (True), O/P is HIGH (True).

In electronics we have device which performs OR function, called OR gate. The symbol for 2 input OR gate is shown in Fig. 3.3 (A).

Now we will implement OR function using switches, diodes and transistor.

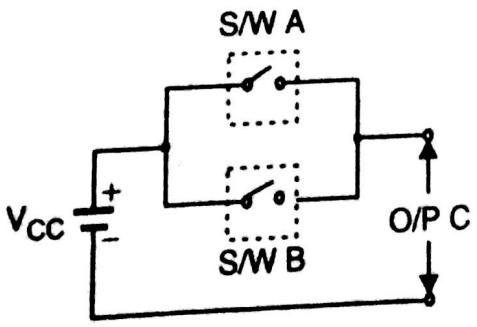



**Fig. 3.4 (a) : OR using switches**

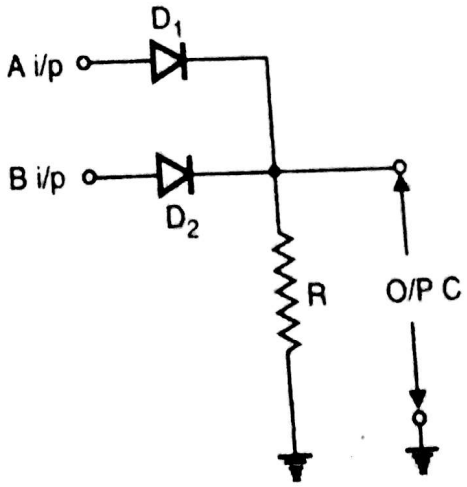



**Fig. 3.4 (b) : OR using diodes**

Scanned by CamScanner

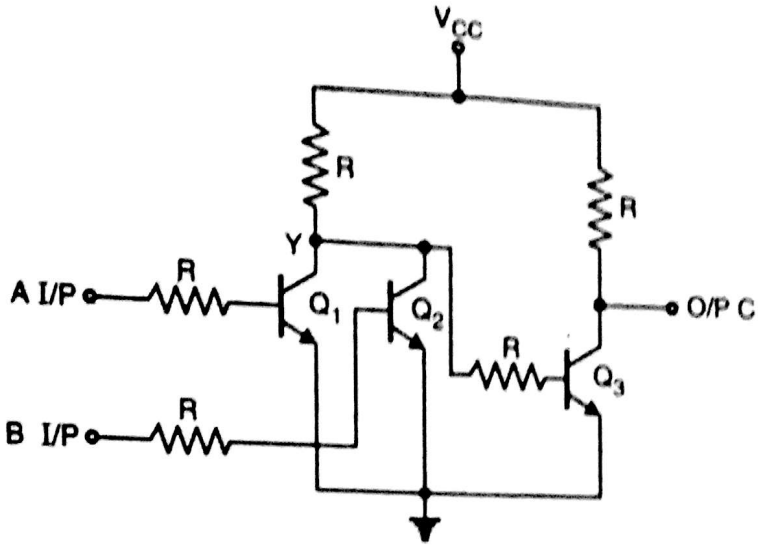



**Fig. 3.4 (c) : OR using transistor**

## OR Using Switches

Refer Fig. 3.4 (a). We have two switches A and B connected in parallel. Output is variable 'C'. Switch open = logic 0, switch closed = logic 1. 0V = logic 0 and $V_{CC}$ = HIGH = logic 1.

**Case I** ⇒ Switch A = B = open, ∴ Circuit is not completed, we have O/P C = 0V = logic 0.

**Case II** ⇒ Switch A = open, B = closed. The electrical connection is completed via switch B, ∴ O/P C = $V_{CC}$ = logic 1.

**Case III** ⇒ Switch A = closed, B = open. The electrical connection is completed via switch A, ∴ O/P C = $V_{CC}$ = logic 1.

**Case IV** ⇒ Switch A = B = closed, Electrical connection is completed via both switches, ∴ O/P C = $V_{CC}$ = logic 1.

If we combine these four cases, we are going to get truth table (T. T) as shown in Table 3.7.

## OR Using Diodes

**Assumption :** Diodes are 'ideal', ∴ drop across diode is approximately 0V. 0V = LOW = Logic 0, $V_{CC}$ = HIGH = logic 1.

**Case I** ⇒ A = B = 0V = logic 0. ∴ diodes $D_1$ and $D_2$ both are cut-off. ∴ O/P C = 0V = logic 0

**Case II** ⇒ A = 0V, B = $V_{CC}$, ∴ diode $D_1$ = OFF, $D_2$ = ON. As $D_2$ is ON there will be $V_{CC}$ appearing across resistance R as forward voltage drop of diode is assumed to be '0'. ∴ O/P C = HIGH = logical 1.

**Case III** ⇒ A = $V_{CC}$, B = 0, ∴ diode $D_1$ = ON, $D_2$ = OFF. As $D_1$ is ON, $V_{CC}$ will appear across resistance R as diode drop is assumed to be 0. ∴ O/P C = HIGH = logical 1.

**Case IV** ⇒ A = B = $V_{CC}$ = HIGH, ∴ both the diodes will conduct and output will be logical 1, as

Scanned by CamScanner

If we combine all four cases, we will get truth Table as shown in Table 3.7.

## OR Using Transistor

In digital, transistor (s) is either OFF or ON (cut-off or saturation). This is achieved by adjustin resistance's in the circuit. Let's refer Fig. 3.4 (c), and study behaviour of circuit.

**Case I** ⇒ A = B = 0V = logic 0. ∴ Transistor $Q_1$ and $Q_2$ both are off, provides HIGH logic at point Y. As $V_Y$ = HIGH, transistor $Q_3$ is ON and operates in saturation. ∴ O/P C ≅ 0V = logic '0'.

**Case II** ⇒ A = 0, B = $V_{CC}$ = HIGH. ∴ Transistor $Q_1$ = OFF and $Q_2$ = ON. The combined effect is $V_Y \cong 0V$. As $V_Y$ = 0V, transistor $Q_3$ = OFF, ∴ O/P C = HIGH = $V_{CC}$ = logic 1.

**Case III** ⇒ A = $V_{CC}$ = HIGH, B = 0. ∴ Transistor $Q_1$ = ON and $Q_2$ = OFF. The combined effect is $V_Y \cong 0V$. As $V_Y$ = 0V, transistor $Q_3$ = OFF, ∴ O/P C = HIGH = $V_{CC}$ = logic 1.

**Case IV** ⇒ A = B = $V_{CC}$ = HIGH, ∴ both the transistor $Q_1 = Q_2$ = ON. ∴ $V_Y$ = 0. ∴ Transistor $Q_3$ = OFF and O/P C = HIGH = $V_{CC}$ = logic 1.

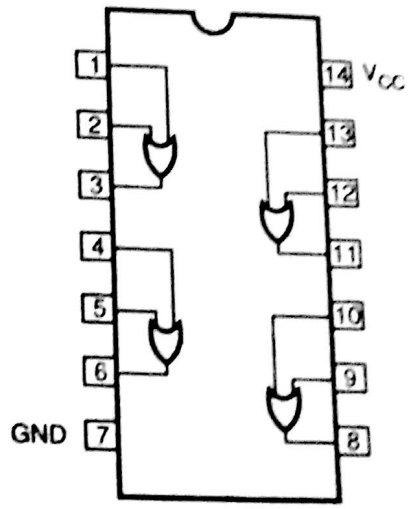
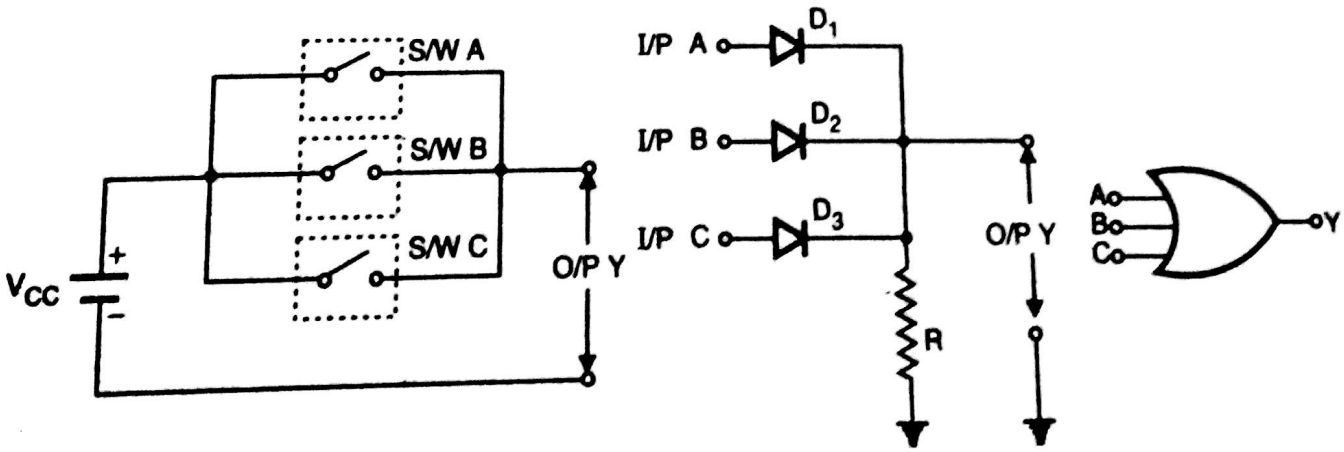



**Fig. 3.5 : Pin diagram of IC 7432**

After combining these four cases, we are going to get Truth table same as Table 3.7.

The OR gate we get in standard package, i.e. DIP. The IC number is 7432. Refer Fig. 3.5

Uptill now we have discussed 2 input OR gate. We can have more than 2 inputs for OR gate. Th Fig. 3.6 shows 3 input OR gate using switches and diodes with symbol.




**(a) OR using 3 switches** **(b) OR using diodes** **(c) Symbol**

**Fig. 3.6**

**The truth table is as shown in Table 3.8.**

Scanned by CamScanner

| Inputs | | | Output |
|---|---|---|---|
| A | B | C | Y |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**Table 3.8**

**Conclusion :** When either of the terminal is HIGH output is HIGH.

$Y = A + B + C$.

Summary of OR function.

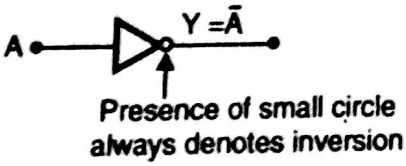
(1) The OR operation produces result 1 when any of the input variable is 1.

(2) The OR operation produces result 0 only when all input variables are '0'.

(3) With OR operation, $1 + 1 = 1$, $1 + 1 + 1 + 1 = 1$, $1 + 1 + \ldots 1 + 1 = 1$ only.

### 3.2.3 NOT Operation (an inverter) (Complementation) :

NOT operation operates on ***single*** variable. It does not operate on multiple input variables like AND, OR gate. The Fig. 3.7 shows symbol for an Inverter.







**(a) Symbol of an inverter** **(b) Symbol of an inverter**

**Fig. 3.7**

As shown in Fig. 3.7, when NOT gate is excited with input variable A, O/P $Y = \bar{A}$, (Y equal to 'Inverse' of A') or (Y equals complement of A). Over bar ( ¯ ) represents NOT operation. The truth Table is shown in Table 3.9.

*NOT of '1' is 0.*

*NOT of '0' is 1.*

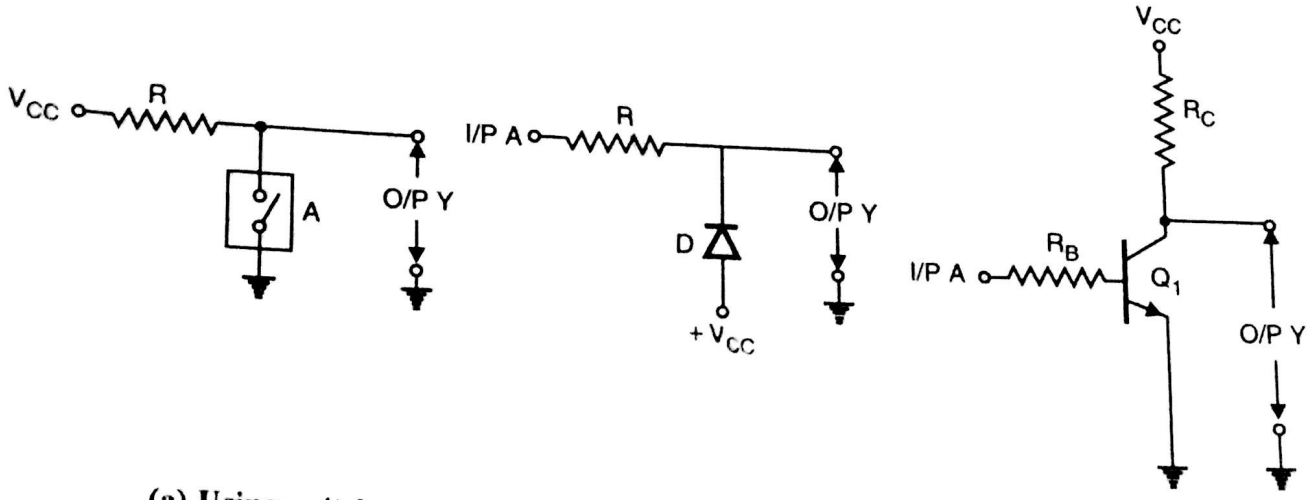
**NOT operation is referred as Inversion or complementation.**

**Let's implement Inverter using switches and transistor.**

| Input | Output |
|---|---|
| A | $Y = \bar{A}$ |
| 0 | 1 |
| 1 | 0 |

**Table 3.9**

Scanned by CamScanner




(a) Using switch (b) Using diodes (c) Using transistor

**Fig. 3.8**

## NOT Using Switch

**Case I :** When A = open means we are applying logic '0', ∴ O/P Y is equal to $V_{CC}$ = HIGH = logic 1.

**Case II :** When A = closed means we are applying logic '1', then O/P Y is equal to LOW = 0V = logic 0, because Y is shorted to ground.

## NOT Using Transistor

**Case I :** When I/P A = HIGH = $V_{CC}$ = logic 1, $Q_1$ will be ON, ∴ O/P Y is approximately equal to 0V = LOW = logic 0.

**Case II :** When I/P A = LOW = 0V = logic 0, $Q_1$ will be cut-off. ∴ O/P Y is equal to $V_{CC}$ = logic 1.

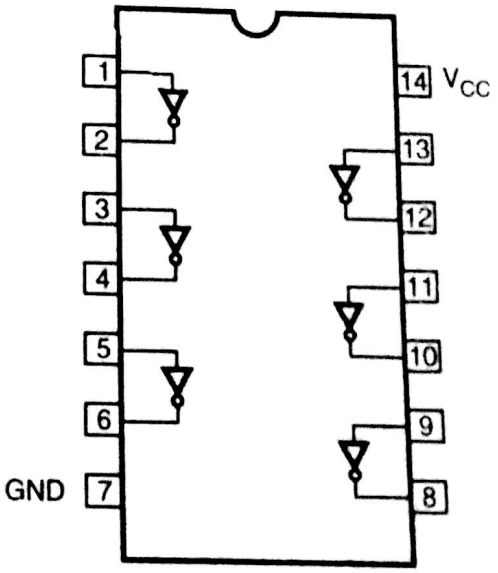



**Fig. 3.9 : Pin diagram of 7404**

If we make truth table for two cases, for NOT using switch / transistor, we will get truth table identical to Table 3.9.

The INV we get in standard DIP package. The IC number is 7404.

## Summary of Logic Gates :

The rules for AND OR and INV can be summarized as follows :

| OR | AND | NOT (INV) |
|---|---|---|
| 0 + 0 = 0 | 0 · 0 = 0 | $\bar{0} = 1$ |
| 0 + 1 = 1 | 0 · 1 = 0 | $\bar{1} = 0$ |
| 1 + 0 = 1 | 1 · 0 = 0 | |
| 1 + 1 = 1 | 1 · 1 = 1 | |

Scanned by CamScanner

## 3.3 Boolean Theorems :

Boolean theorems help to simplify circuits using Boolean algebra. We are going to derive Boolean theorems basically from AND, OR and INV (Inverter), taking some typical cases

### AND Laws

i) **One input fixed at logic '0' :**

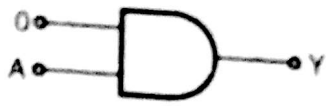



**Fig. 3.10**

Refer Fig. 3.10. The truth Table of AND gate states that when one input is fixed at logic '0' irrespective of other input, at I/O logic, output is always '0'

$\therefore$ $\boxed{A \cdot 0 = 0}$

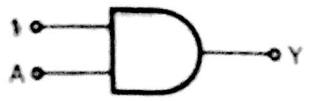



**Fig. 3.11**

ii) **One input fixed at logic 1 :**

Refer Fig. 3.11. When one input of AND gate is at logic level '1'. Output follows input variable i.e. $Y = 0$ if $A = 0$, $Y = 1$ if $A = 1$

$\therefore$ $\boxed{A \cdot 1 = A}$

iii) **Both inputs are having common input OR both inputs are shorted to single variable :**

Refer Fig. 3.12

(a) If $A = 0$, both inputs are at logic '0', i.e. $Y = 0$

(b) If $A = 1$, both inputs are at logic '1', i.e. $Y = 1$

$\therefore$ $\boxed{A \cdot A = A}$

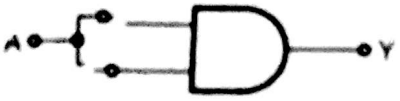



**Fig. 3.12**

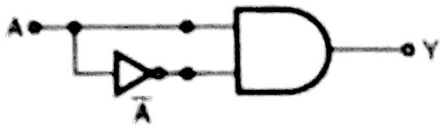



**Fig. 3.13**

iv) **If one input is negated (complement) of the other :**

Refer Fig. 3.13. Irrespective of value of A i.e. logic '0' or '1', one input will be always at '0', $\therefore$ O/P of AND gate will be always logic '0'

$\therefore$ $\boxed{A \cdot \bar{A} = 0}$

### OR Laws

(i) **One input fixed at logic '0' :**




**Fig. 3.14**




**Fig. 3.15**

Scanned by CamScanner

Refer Fig. 3.14

(a) If A = 0, both inputs are at logic '0', ∴ Y = 0.

(b) If A = 1, One input of OR gate is at logic 1, ∴ Y = 1 means O/P follows input.

∴ $\boxed{A + 0 = A}$

**(ii) One input fixed at logic 1 :**

Refer Fig. 3.15 for OR gate, if either of the input is at logic 1 output will be always at logic '1'.

∴ $\boxed{A + 1 = 1}$

**(iii) Both inputs are shorted to single variable :**




**Fig. 3.16**

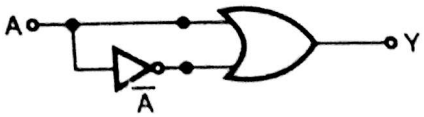



**Fig. 3.17**

Refer Fig. 3.16.

(a) If A = 0, both inputs of OR gate will be at logic '0', ∴ Y = 0.

(b) If A = 1, both inputs of OR gate will be at logic '1', ∴ Y = 1.

∴ $\boxed{A + A = A}$

**(iv) If one input is negation (complement) of other :**

Refer Fig. 3.17.

Irrespective of value of A, i.e. logic 0 or 1, one input of OR gate will be always at logic 1, ∴ O/P will be always at logic 1.

∴ $\boxed{A + \bar{A} = 1}$

## Inversion Law

(i) Refer Fig. 3.18, If input is A, output $Y = \bar{A}$.

(ii) Refer Fig. 3.19. If input A is given to 1st Inverter $x = \bar{A}$.




**Fig. 3.18**




**Fig. 3.19**

The output of first gate given to 2nd gate, ∴ $Y = \bar{x}$, but as $x = \bar{A}$, $Y = \bar{x} = \bar{\bar{A}} = A$.

**Conclusion :** Double Inversion of variable is variable itself.

$\boxed{\bar{\bar{A}} = A}$

Scanned by CamScanner

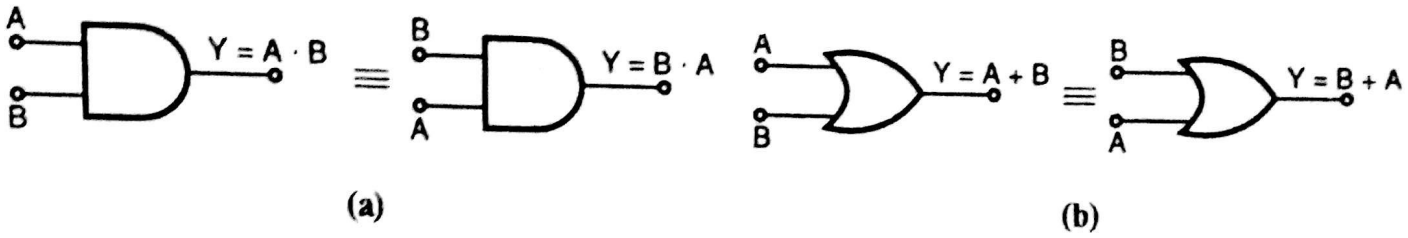
## 3.4 Boolean Laws :

### Commutative Law

Any binary operation that satisfies law $X \cdot Y = Y \cdot X$ is referred to as ***Commutative Operation***.

i.e. $\boxed{A \cdot B = B \cdot A}$ and $\boxed{A + B = B + A}$

Changing position of variables on the gate does not affect its operation. The same is shown in Fig. 3.20.




(a) (b)

**Fig. 3.20**

The same thing can be proved using ***Perfect Induction Method***. In this method one can check validity of a Boolean expression for all possible combinations of logic '0' and '1's.

| | | | | | |
|---|---|---|---|---|---|
| | A | 0 | 0 | 1 | 1 |
| | B | 0 | 1 | 0 | 1 |
| A · B | | 0 | 0 | 0 | 1 |
| B · A | | 0 | 0 | 0 | 1 |

$\therefore$ $\boxed{A \cdot B = B \cdot A}$

| | | | | | |
|---|---|---|---|---|---|
| | A | 0 | 0 | 1 | 1 |
| | B | 0 | 1 | 0 | 1 |
| A + B | | 0 | 1 | 1 | 1 |
| B + A | | 0 | 1 | 1 | 1 |

$\therefore$ $\boxed{A + B = B + A}$

### Associative Law

This law states that the order in which the operation is performed is irrelevant as their effect is the same.

For example $(x \cdot y) \cdot z = x \cdot (y \cdot z)$

also $(x + y) + z = x + (y + z)$.

This is diagramatically represented in Figs. 3.21 (a) and (b).

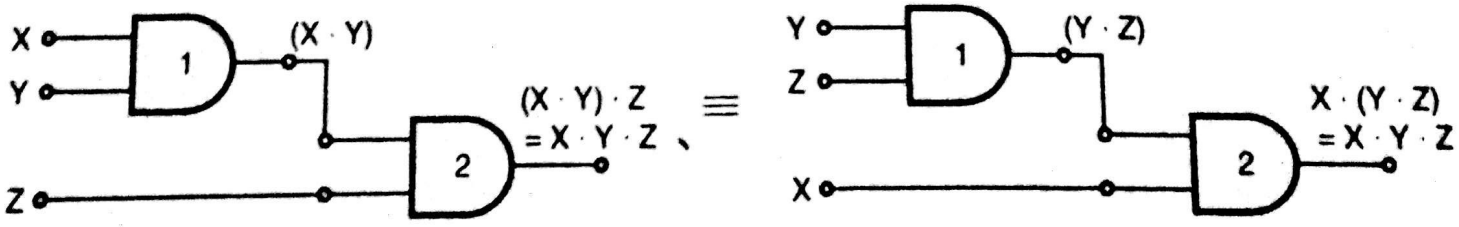



**Fig. 3.21 : (a)**

Scanned by CamScanner

Need to prove this.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | A | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | | B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | C | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| A | · | B | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| B | · | C | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| (A·B) | · | C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| A | · | (B·C) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

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

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| C | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| A + B | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| B + C | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| (A + B) + C | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| A + (B + C) | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Scanned by CamScanner

$\therefore$ $\boxed{A \cdot (B + C) = (A \cdot B) + (A \cdot C) = AB + AC}$

Same way you have

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | → LSB |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| C | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | → MSB |
| B · C | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |
| A + B | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | |
| A + C | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | |
| A + (B · C) | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | } both are matching |
| (A+B) · (A+C) | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | |

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

## 3.5 De Morgan's Theorem :

Before I start explaining you theorem, let's see few definitions :

(i) **Literal :** A variable in a Boolean expression, appearing in uncomplemented or complemented form. For example in the expression $Y = (A + \bar{B}\ C)$, we have two terms A and $\bar{B}C$. The first term has a signal literal A while the second term $\bar{B}C$ has two literals $\bar{B}$ and C.

(ii) **Product Term :** A product is solely product of literal, for example $\bar{A}\bar{B}C$ is a product term. A product (AND) of 'n' Boolean variables (3 variables in above case), uncomplemented or complemented, but not repeated, in a Boolean function of n variables is referred to as a standard product term or **'Minterm'**.

(iii) **Sum of Term :** A sum term is solely the sum of literals. For example $A + \bar{B} + \bar{C}$ is a sum term. A sum (OR) of 'n' Boolean variables, uncomplemented or complemented, but not repeated, in a Boolean function of 'n' variables is referred to as a standard term or **'Maxterm'**.

(iv) **Standard Product Term :** A product term which consists of one literal from every variable in the function domain. For example in a four variable ABCD the standard product term will be $\bar{A}\ \bar{B}\ C\ D$, $A\bar{B}\bar{C}D$, $\bar{A}\bar{B}\bar{C}D$, $\bar{A}\bar{B}CD$ etc.

(v) **Standard Sum Term :** A sum term which consists of one literal from every variable in the function domain. For example in a domain of three variables A, B, C, the standard sum terms would be $(A + B + C)$, $(A + B + \bar{C})$, $(A + \bar{B} + C)$ etc.

(vi) **Sum of Products :** A Boolean expression, expressed as sum of product term i.e. as an OR of AND terms containing uncomplemented or complemented variables, for example

$AB + BC + CA$

$\bar{A}BC + AB + AB\bar{C} + ACD$ etc.

Scanned by CamScanner

**(vii) Product of Sum :** A Boolean function expressed as product of sum terms i.e. as an AND of OR terms containing uncomplemented or complemented variables for example (A + C) (B + A + D) etc.

**(viii) Standard Sum of Products :** This form contains one standard product term or minterm for each *one* entry in the truth Table for the expression. This form can be represented as the OR (Sum) of a group of ANDs (the product terms) of Boolean variables, uncomplemented or complemented, for example

$f(A, B, C, D) = A \bar{B} C$ in its standard SOP form would include D also i.e. $A\bar{B}C(D + \bar{D}) = A\bar{B}C\bar{D} + A\bar{B}CD$ (As $D + \bar{D} = 1$).

**(ix) Standard Product of Sum :** This form contains one standard term or maxterm for each *Zero* entry in the truth Table for the expression. This form can be represented as AND (Product) of the group of OR's (the sum term) of Boolean expression, uncomplemented or complemented, for example

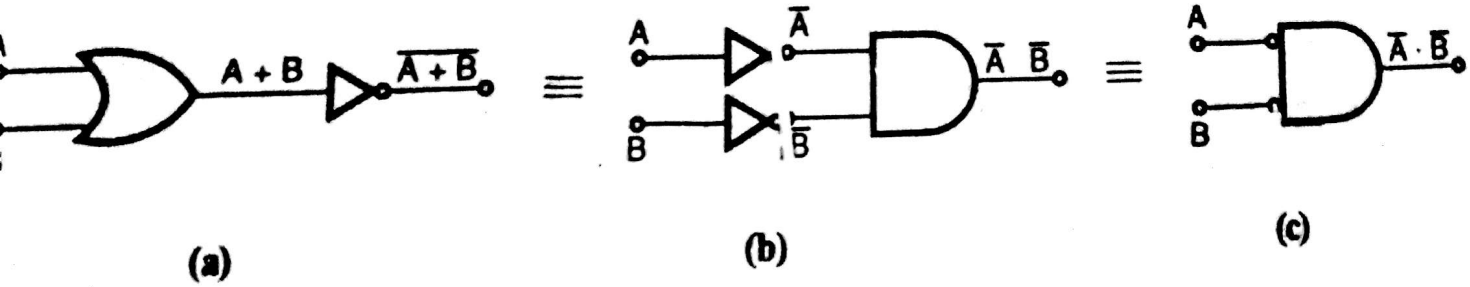
$$f(A, B, C) = (A + C)$$
$$= (A + C + B\bar{B}) \quad (B\bar{B} = 0 \text{ Refer AND law})$$
$$= (A + B + C) \cdot (A + \bar{B} + C)$$

**Note :** While writing, product (AND) term, we normally put 'dot'. But now onward 'dot' may or may not be present because $A \cdot B$ can be also written as AB, $A \cdot B \cdot C = ABC$, $\bar{A} \cdot B \cdot \bar{C} = \bar{A} B \bar{C}$ and so on.

One of the most powerful identity used in Boolean algebra is Demorgan's theorem. Demorgan formulated two theorems to simplify expressions i.e. they provide method of expressing the complement of a complex expression in terms of complement of Individual component or in short theorem allows removal of individual variable from under a NOT sign for e.g. $\overline{A + B + C} \cdot \overline{A B C}$, $\overline{AB + C}$ etc. The first Demorgan's theorem states that ***the complement of sum of products equal to the product of complement of individual variable*** i.e. $\overline{A + B} = \bar{A} \cdot \bar{B}$

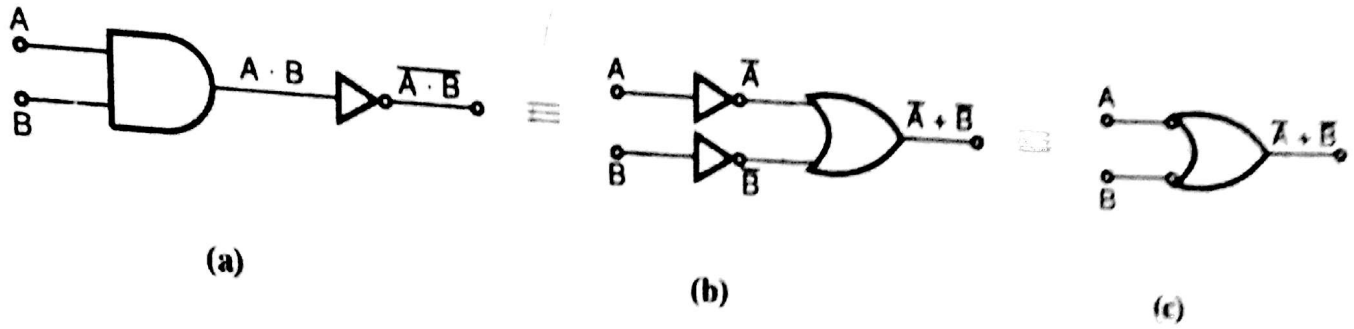
The second Demorgan's theorem states that ***the complement of a product equals the sum of the complement of individual variable*** i.e. $\overline{A \cdot B} = \bar{A} + \bar{B}$

| | | |
|---|---|---|
| $\overline{A + B}$ | $= \bar{A} \cdot \bar{B}$ | Refer Fig. 3.22 |
| $\overline{A \cdot B}$ | $= \bar{A} + \bar{B}$ | Refer Fig. 3.23 |




(a) (b) (c)

Scanned by CamScanner




(a) (b) (c)

**Fig. 3.23**

Steps involved in Demorgan's theorem.

**Step I** : Complement entire function. Say given function is $\overline{A + B}$.

$\therefore$ Again complement $\overline{\overline{A + B}} = A + B$.

**Step II** : Change all AND to ORs and all ORs to AND.

$\therefore$ $A + B$ will become $A \cdot B$.

**Step III** : Complement each of Individual variable.

$\therefore$ We will get $\overline{A} \cdot \overline{B}$ $\quad \therefore \quad \overline{A + B} = \overline{A} \cdot \overline{B}$

The above procedure is called ***Demorganisation***. Let's prove it by perfect Induction method

| | | | | | |
|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | |
| B | 0 | 0 | 1 | 1 | |
| $\overline{A}$ | 1 | 0 | 1 | 0 | |
| $\overline{B}$ | 1 | 1 | 0 | 0 | |
| $\overline{A} \cdot \overline{B}$ | 1 | 0 | 0 | 0 | ⟶ |
| $A + B$ | 0 | 1 | 1 | 1 | These two are matching |
| $\overline{A + B}$ | 1 | 0 | 0 | 0 | ⟶ $\therefore \overline{A + B} = \overline{A} \cdot \overline{B}$ |

Same way

| | | | | | |
|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | |
| B | 0 | 0 | 1 | 1 | |
| $\overline{A}$ | 1 | 0 | 1 | 0 | |
| $\overline{B}$ | 1 | 1 | 0 | 0 | |
| $\overline{A} + \overline{B}$ | 1 | 1 | 1 | 0 | ⟶ These two are matching. |
| $A \cdot B$ | 0 | 0 | 0 | 1 | |
| $\overline{A.B}$ | 1 | 1 | 1 | 0 | ⟶ $\therefore \overline{A \cdot B} = \overline{A} + \overline{B}$ |

## Deriving More Laws from Basic Laws

**Now we are going to derive more laws from the basic laws.**

Scanned by CamScanner

(i) $A + AB = A$

**Proof :** $A + AB$

$= A \cdot 1 + AB$ (AND law (ii))

$= A \cdot (1 + B)$ (Distributive law)

$= A \cdot 1$ (OR law (i))

$= A$ (AND law (ii))

(ii) $A + \bar{A}B = A + B$

**Proof :** $A + \bar{A}B$

$= A \cdot 1 + \bar{A}B$ (AND law (i))

$= A \cdot (B + 1) + \bar{A}B$ (OR law (ii))

$= AB + A \cdot 1 + \bar{A}B$ (Distributive law)

$= AB + A + \bar{A}B$ (AND law (ii))

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

$= A + B \cdot 1$ (OR law (iv))

$= A + B$ (AND law (ii))

Same way we have

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

(iii) $A \cdot (A + B) = A$.

**Proof :** $A \cdot (A + B)$

$= A \cdot A + A \cdot B$ (Distributive law)

$= A + A \cdot B$ (AND law (iii))

$= A \cdot 1 + A \cdot B$ (AND law (ii))

$= A \cdot (1 + B)$ (Distributive law)

$= A \cdot 1$ (OR law (iv))

$= A$ (AND law (ii))

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

**Proof :** $A \cdot (\bar{A} + B)$

$= A \cdot \bar{A} + AB$ (Distributive law)

$= 0 + AB$ (AND law (iv))

$= AB$ (OR law (i))

Scanned by CamScanner

Summary of Boolean laws and Theorems Let's summaries all the laws.

## LAWS

| | | | | |
|---|---|---|---|---|
| (1) | $A \cdot 0 = 0$ | AND law | (12) $A + 0 = A$ | OR law |
| (2) | $A \cdot 1 = A$ | | (13) $A + 1 = 1$ $\bar{A} + 1 = 1$ | |
| (3) | $A \cdot A = A$ | | (14) $\bar{A} + A = A$ | |
| (4) | $A \cdot \bar{A} = 0$ | | (15) $A + \bar{A} = 1$ | |
| | Commutative law | | | |
| (5) | $A \cdot B = B \cdot A$ | | (16) $A + B = B + A$ | |
| | Associative law | | | |
| (6) | $A \cdot (B.C) = (A \cdot B) \cdot C$ | | (17) $A + (B + C) = (A + B) + C$ | |
| | Distributive law | | | |
| (7) | $A \cdot (B + C) = A \cdot B + A \cdot C$ | | (18) $A + (B \cdot C) = (A + B) \cdot (A + C)$ | |
| | More laws | | | |
| (8) | $A \cdot (A + B) = A$ | | (19) $A + AB = A$ | |
| (9) | $A \cdot (\bar{A} + B) = AB$ | | (20) $A + \bar{A}B = A + B$ | |
| (10) | $\bar{\bar{A}} = A$ | | (21) $\bar{A} + AB = \bar{A} + B$ | |
| | | | (22) $\bar{A} + A\bar{B} = \bar{A} + \bar{B}$ | |
| | Demorgan theorem | | | |
| (11) | $\overline{A \cdot B} = \bar{A} + \bar{B}$ | | (23) $\overline{A + B} = \bar{A} \cdot \bar{B}$ | |

**Table 3.10**

## Duality Theorem

The duality theorem states that starting with a particular relation, the other Boolean relation can be derived by,

(i) Replacing OR sign, with AND sign.

(ii) Replacing AND sign, with OR sign.

(iii) Complementing 0's and 1's in the expression.

Scanned by CamScanner

## 3.6 Perfect Induction :

As I have specified this is the method using which one can check the validity of a Boolean expressi for all possible combinations of logic 0 and 1's. Let's see few examples

(i) $(A + B)(\bar{A} + B)(A + \bar{B})(\bar{A} + \bar{B}) = 0$

**Soln. :** Since only two variables A and B are given, we have :

(i) A = 0, B = 0

$$\begin{aligned} LHS &= (0 + 0)(\bar{0} + 0)(0 + \bar{0})(\bar{0} + \bar{0}) \\ &= 0\,(1 + 0)(0 + 1)(1 + 1) \\ &= 0 \cdot 1 \cdot 1 \cdot 1. \\ &= 0 = RHS \end{aligned}$$

(ii) A = 0, B = 1

$$\begin{aligned} LHS &= (0 + 1)(\bar{0} + 1)(0 + \bar{1})(\bar{0} + \bar{1}) \\ &= 1\,(1 + 1)(0 + 0)(1 + 0) \\ &= 1 \cdot 1 \cdot 0 \cdot 1 \\ &= 0 = RHS \end{aligned}$$

(iii) A = 1, B = 0

$$\begin{aligned} LHS &= (1 + 0)(\bar{1} + 0)(1 + \bar{0})(\bar{1} + \bar{0}) \\ &= 1\,(0 + 0)(1 + 1)(0 + 1) \\ &= 1 \cdot 0 \cdot 1 \cdot 1. \\ &= 0 = RHS \end{aligned}$$

(iv) A = 1, B = 1

$$\begin{aligned} LHS &= (1 + 1)(\bar{1} + 1)(1 + \bar{1})(\bar{1} + \bar{1}) \\ &= 1\,(0 + 1)(1 + 0)(0 + 0) \\ &= 1 \cdot 1 \cdot 1 \cdot 0. \\ &= 0 = RHS \end{aligned}$$

∴ The given expression is valid.

---

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

**Soln. :** Since three variables A, B, C are given, we have :

(i) A = B = C = 0

$$0\,0 + \bar{0}\,\bar{0} + \bar{0}\,0\,0 = 0\,0\,\bar{0} + \bar{0}\,\bar{0} + 0\,0$$
$$0 + 1\,1 + 1\,0\,0 = 0\,0\,1 + 1\,1 + 0$$
$$0 + 1 + 0 = 0 + 1 + 0 \quad \text{True.}$$

(ii) A = 0, B = 0, C = 1

$$0\,0 + \bar{0}\,\bar{0} + \bar{0}\,0\,1 = 0\,0\,\bar{1} + \bar{0}\,\bar{0} + 0\,1$$
$$0 + 1\,1 + 1\,0\,1 = 0\,0\,0 + 1\,1 + 0$$
$$0 + 1 + 0 = 0 + 1 + 0 \quad \text{True}$$

Scanned by CamScanner

(iii) A = 0, B = 1, C = 0.

$$0\,1 + \bar{0}\bar{1} + \bar{0}\,1\,0 = 0\,1\,\bar{0} + \bar{0}\,\bar{1} + 1\,0$$
$$0 + 1\,0 + 1\,1\,0 = 0\,1\,1 + 1\,0 + 1\,0$$
$$0 + 0 + 0 = 0 + 0 + 0 \quad \text{True}$$

(iv) A = 0, B = 1, C = 1

$$0\,1 + \bar{0}\,\bar{1} + \bar{0}\,1\,1 = 0\,1\,\bar{1} + \bar{0}\,\bar{1} + 1\,1$$
$$0 + 1\,0 + 1\,1\,1 = 0\,1\,0 + 1\,0 + 1$$
$$0 + 0 + 1 = 0 + 0 + 1 \quad \text{True}$$

(v) A = 1, B = 0, C = 0.

$$1\,0 + \bar{1}\,\bar{0} + \bar{1}\,0\,0 = 1\,0\,\bar{0} + \bar{1}\,\bar{0} + 0\,0$$
$$0 + 0\,1 + 0\,0\,0 = 1\,0\,1 + 0\,1 + 0$$
$$0 + 0 + 0 = 0 + 0 + 0 \quad \text{True}$$

(vi) A = 1, B =0, C = 1.

$$1\,0 + \bar{1}\,\bar{0} + \bar{1}\,0\,1 = 1\,0\,\bar{1} + \bar{1}\,\bar{0} + 0\,1$$
$$0 + 0\,1 + 0\,0\,1 = 1\,0\,0 + 0\,1 + 0$$
$$0 + 0 + 0 = 0 + 0 + 0 \quad \text{True}$$

(vii) A = 1, B = 1, C = 0

$$1\,1 + \bar{1}\,\bar{1} + \bar{1}\,1\,0 = 1\,1\,\bar{0} + \bar{1}\,\bar{1} + 1\,0$$
$$1 + 0\,0 + 0\,1\,0 = 1\,1\,1 + 0\,0 + 1\,0$$
$$1 + 0 + 0 = 1 + 0 + 0 \quad \text{True}$$

(viii) A = 1, B = 1, C = 1

$$1\,1 + \bar{1}\,\bar{1} + \bar{1}\,1\,1 = 1\,1\,\bar{1} + \bar{1}\,\bar{1} + 1\,1$$
$$1 + 0\,0 + 0\,1\,1 = 1\,1\,0 + 0\,0 + 1$$
$$1 + 0 + 0 = 0 + 0 + 1 \quad \text{True}$$

∴ Given expression is verified

## 3.7 Reduction of Logic Expression using Boolean Algebra :

In this section, Boolean expression will be given to you and you will be asked to minimize Or reduce it. What do you mean by reduction or minimisation ? If you have Boolean expression which is not minimised and if you built ***hardware*** or circuit with the help of that expression, then number of Integrated Circuits (ICs) required will be more. As hardware is more ***reliability*** reduces and also cost of product increases. Therefore one has to ***minimise*** expression in such a way that hardware required will be less. Basically we have three technique for minimisation :

(1) Reduction of logic expression using Boolean algebra.
(i.e. we are going to use the Boolean laws, and we will reduce the expression.)

(2) Karnaugh Map (K–Map)
(3) Quine Mc clusky method. } We will study in later chapters.

Let's solve some examples to reduce Boolean expression using Boolean laws.

Scanned by CamScanner

**Ex. 1 :** Reduce following :

(i) A A B C

(ii) A B B B C

(iii) $A \cdot A \cdot A \cdot \bar{A}$

(iv) $A \cdot 1 \cdot A$

(v) W + W + X

(vi) W + X + X + X

(vii) W + X + 1

(viii) W + W + X + X + X

(ix) AA + BB.

(x) $A\bar{A}$ + BBB

(xi) BC + $B\bar{B}$C. (Note : Refer Table 3.10).

**Soln. :** (i) $A \cdot A \cdot B \cdot C$

$= A \cdot B \cdot C = A\ B\ C$

(ii) $A \cdot B\ B\ B \cdot C$ (Law (3)) $\longrightarrow A \cdot A = A$

$= A\ B\ C$

(iii) $A\ A\ A\ \bar{A}$ (Law (3)) $\longrightarrow \underline{B.B}\ B = \underline{B\ B} = B$

$= A\ \bar{A}$

$= 0$ (Law (3)) $\longrightarrow A \cdot A\ A = A\ A = A$

(Law (4)) $\longrightarrow A \cdot \bar{A} = 0$

(iv) $A \cdot 1 \cdot A$

$= A \cdot A \cdot 1$ (Law (6)) $\longrightarrow$ Associative (Commutative)

$= A \cdot 1$ (Law (3))

$= A$ (Law (2))

(v) $W + W + X$

$= W + X$

(vi) $W + \underline{X + X} + X$ (Law (14)) $\longrightarrow A + A = A$

$= W + \underline{X + X}$ (Law (14))

$= W + X$ (Law (14))

(vii) $W + \underline{X + 1}$

$= W + 1$ (Law (13))

$= 1$ (Law (13))

(viii) $\underline{W + W} + \underline{X + X + X}$

$= W + \underline{X + X}$ (Law (14))

$= W + X$

(ix) $A\ A + B\ B$

$= A + B$ (Law (3))

(x) $A \cdot \bar{A} + B\ B\ B$

$= A \cdot \bar{A} + B$ (Law (3))

$= 0 + B$ (Law (4))

$= B + 0 = B$ (Law (12))

(xi) $BC + \underline{B\bar{B}}\ C$

$= BC + 0 \cdot C$ (Law (4))

Scanned by CamScanner

$= BC + 0$ (Law (1))

$= BC$ (Law (12))

**OR**

$B\,C + B\bar{B}C$

$= BC\left(1 + \bar{B}\right)$ (Law (7))

$= BC\,(1)$ (Law (13))

$= B \cdot C \cdot 1 = B\,C$ (Law (2))

**Ex. 2 :** Reduce following :

(i) $A + \bar{A}B + AB$

(ii) $A\bar{B} + \bar{A}B + AB + \bar{A}\bar{B}$

(iii) $A\bar{B}D + A\bar{B}D + \bar{B}\bar{D}$

(iv) $W\bar{X}(W + Y) + WY\left(\bar{W} + \bar{X}\right)$

(v) $(AB + C) \cdot (AB + D)$.

**Soln. :**

(i) $A + \bar{A}\,B + A\,B$

$= A + A\,B + \bar{A}\,B$ Commutative

$= A\,(1 + B) + \bar{A}\,B$ (Law (7))

$= A \cdot 1 + \bar{A}\,B$ (Law (13))

$= A + \bar{A}\,B$ (Law (2))

$= A + B$ (Law (20))

(ii) $\underset{(2)}{A\bar{B}} + \underset{(1)}{\bar{A}B + AB} + \underset{(2)}{\bar{A}\bar{B}}$

(Adjust such that you get variable common)

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

$= B\left(\bar{A} + A\right) + \bar{B}\left(\bar{A} + A\right)$ (Law (7))

$= B \cdot (1) + \bar{B} \cdot (1)$ (Law (15))

$= B + \bar{B}$ (Law (2))

$= 1$ (Law (15)).

(iii) $A\bar{B}D + A\bar{B}D + \bar{B}\,\bar{D}$

$= A\bar{B}D + \bar{B}\bar{D}$ (Law (14))

$= \bar{B}\left(AD + \bar{D}\right)$ (Law (7))

$= \bar{B}\left(\bar{D} + AD\right)$ (Law (16)). commutative.

$= \bar{B}\left(\bar{D} + A\right)$ (Law (21))

Scanned by CamScanner

(iv) $W\bar{X}(W + Y) + WY(\bar{W} + \bar{X})$

$= W\bar{X} \cdot W + W\bar{X} \cdot Y + WY\bar{W} + WY\bar{X}$

$= W\bar{X} + \underline{WY\bar{X}} + W\bar{W}Y + \underline{WY\bar{X}}$ (Law (7))

$= W\bar{X} + WY\bar{X} + W \cdot \bar{W}Y$ (Law (3))

$= W\bar{X} + WY\bar{X} + 0 \cdot Y$ (Law (14))

$= W\bar{X}(1 + Y) + 0$ (Law (4))

$= W\bar{X} \cdot 1$ (Law (7) and (1))

$= W\bar{X}$ (Law (13) and (12))

(v) $(AB + C)(AB + D)$ (Law (2))

$= AB \cdot AB + AB \cdot D + AB \cdot C + C \cdot D$

$= AB + ABD + ABC + CD$ (law (7))

$= AB(1 + D + C) + CD$ (Law (3))

$= AB \cdot 1 + CD$ (Law (7))

$= AB + CD$ (Law (13))

(Law (2)).

**Ex. 3 :** Prove following :

(i) $AB + ABC + A\bar{B} = A.$

(ii) $(B + A)(B + D)(A + C)(C + D) = BC + AD$

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

**Soln. :**

(i) $AB + ABC + A\bar{B}$ (L. H. S.)

$= A \cdot B \cdot (C + 1) + A\bar{B}$ (Law (7))

$= A \cdot B \cdot 1 + A\bar{B}$ (Law (13))

$= A \cdot B + A\bar{B}$ (Law (2))

$= A \cdot (B + \bar{B})$ (Law (7))

$= A \cdot 1$ (Law (15))

$= A$ (R. H. S.) (Law (2))

(ii) $\underline{(B + A)(B + D)}\ \underline{(A + C)(C + D)}$ (L. H. S.)

$= (BB + BD + AB + AD)(AC + AD + CC + CD)$ (Law 7)

$= (B + BD + AB + AD)(AC + AD + C + CD)$ (Law 3)

$= [B(1 + D) + AB + AD][C(1 + A) + AD + CD]$ (Law 7)

$= [B + AB + AD][C + AD + CD]$ (Law 13)

Scanned by CamScanner

$= [B(1 + A) + AD][C(1 + D) + AD]$ (Law 7)

$= [B + AD][C + AD]$ (Law 13)

$= BC + BAD + ADC + AD \cdot AD$ (Law 7)

$= BC + BAD + ADC + AD$ (Law 3)

$= BC + AD \quad (B + C + 1)$ (Law 7)

$= BC + AD$ (R. H. S.) (Law 13)

(iii) $\left(\bar{A} + B\right)\bar{A}\bar{B}\bar{C}$ (L. H. S.)

$= \bar{A}\bar{B}\bar{C} \cdot \bar{A} + \bar{A}\bar{B}\bar{C} \cdot B$ (Law 7)

$= \bar{A}\bar{B}\bar{C} + 0$ (Law 3 and 4)

$= \bar{A}\bar{B}\bar{C}$ (Law 12)

$= \overline{A + B + C}$ (Law 11)

**Ex. 4 :** Prove following

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

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

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

**Soln. :**

(i) $ABC + ACD + \bar{C}A$ (L. H. S)

$= ABC + \left(\bar{A} + \bar{C}\right)D + \bar{C}A$ (Law 11)

$= AD + CD + A\left(\bar{C} + \bar{B}C\right)$ (Law 7)

$= AD + CD + A\left(\bar{C} + \bar{B}\right)$ (Law 22)

$= AD + CD + A\bar{C} + A\bar{B}$ (Law 7)

$= R.H.S$

(ii) $\overline{(A + AB)}\,\overline{(C + \bar{D})}$ (L. H. S)

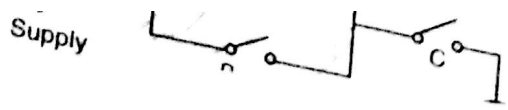
$= \overline{(A + B)(C + \bar{D})}$ (Law 20)

$= \overline{(\bar{A} + \bar{B})}\,\overline{(C + \bar{D})}$ (Law 11)

$= \bar{A}\bar{B} + \bar{C}\bar{\bar{D}}$ (Law 23)

$= \bar{A}\bar{B} + \bar{C}D$ (Law 10)

Scanned by CamScanner






(iii) $(A + B)(\bar{A} + C)(B + C)$ (L. H. S)

$= (A + B)(\bar{A}B + \bar{A}C + BC + CC)$ (Law 7)

$= (A + B)(\bar{A}B + \bar{A}C + BC + C)$ (Law 3)

$= (A + B)[\bar{A}B + C(\bar{A} + B + 1)]$ (Law 7)

$= (A + B)(\bar{A}B + C)$ (Law 13)

$= A \cdot \bar{A}B + AC + \bar{A}B \cdot B + BC$ Law (7)

$= 0 + AC + \bar{A}B + BC$ (Law 4 and 3)

$= AC + \bar{A}B + BC + 0$ (Law 12)

$= AC + \bar{A}B + BC + A\bar{A}$ (Law 4)

$= C(A + B) + \bar{A}(B + A)$ (Law 7)

$= C(A + B) + \bar{A}(A + B)$ (Law 16)

$= (A + B)(\bar{A} + C)$ (R. H. S.) (Law 18)

**Ex. 5 :** Simplify the following

(i) $F_1 = A\bar{C} + B\bar{C}D + A\bar{B}C + ACD$

(ii) $F_2 = B + A\bar{B} + ACD + A\bar{C}$

(iii) $F_3 = \overline{\overline{A + B + \overline{CD}}}$

**Soln. :**

(i) $F_1 = A\bar{C} + B\bar{C}D + A\bar{B}C + ACD$

$= A[\bar{C} + CD] + B\bar{C}D + A\bar{B}C$ (Law 7)

$= A[\bar{C} + D] + B\bar{C}D + A\bar{B}C$ (Law 21)

$= A\bar{C} + AD + B\bar{C}D + A\bar{B}C$ (Law 7)

$= A(\bar{C} + \bar{B}C) + AD + B\bar{C}D$ (Law 7)

$= A(\bar{C} + \bar{B}) + AD + B\bar{C}D$ (Law 22)

$= A\bar{C} + A\bar{B} + AD + B\bar{C}D$ (Law 7)

$= A[\bar{C} + \bar{B} + D] + B\bar{C}D$ (Law 7)

(ii) $F_2 = B + A\bar{B} + ACD + A\bar{C}$ (Law 7 and 20)

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

Scanned by CamScanner

$= A + B + A(\bar{C} + D)$ (Law 21)

$= B + A(\bar{C} + D + 1)$ (Law 7)

$= B + A + (1)$ (Law 13)

$= B + A$ (Law 2)

$= A + B$ (Law 16)

(iii) $A + \overline{B + \overline{CD}}$

$= \bar{A} \cdot \left( \overline{B + \overline{CD}} \right)$ (Law 23)

$= \bar{A} \cdot (B + \bar{C} + \bar{D})$ (Law 10 and 11)

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

**Ex. 6 :** Reduce $F = AB + \overline{AC} + A\bar{B}C\,(AB + C)$

**Soln. :** $AB + \overline{AC} + A\bar{B}C\,(AB + C)$

$= AB + \bar{A} + \bar{C} + A\bar{B}C \cdot AB + A\bar{B}C \cdot C$ (Law 7 and 11)

$= AB + \bar{A} + \bar{C} + 0 + A\bar{B}C$ (Law 4)

$= AB + \bar{A} + \left[ \overline{\bar{C} + C \cdot A\bar{B}} \right]$ (Law 12)

$= AB + \bar{A} + \bar{C} + A\bar{B}$ (Law 21)

$= A(B + \bar{B}) + \bar{A} + \bar{C}$ (Law 7)

$= A + \bar{A} + \bar{C}$ (Law 15)

$= 1 + \bar{C}$ (Law 15)

$= 1$ (Law 13)

**Ex. 7 :** Reduce following expressions

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

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

(iii) $\overline{(A + \overline{BC})} \left( A\bar{B} + \overline{ABC} \right)$

(iv) $ABC + \overline{\bar{A}\bar{B}} + AC$

(v) $\overline{(\bar{A}\bar{B} + ABC)(\bar{A}\bar{B}C)}$

***Soln. :***

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

$= AB + A(\bar{B} \cdot \bar{C})$ Demorgan (Law 23)

Scanned by CamScanner

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

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

$= A(B + \bar{C})$ (Law (7))

(ii) $\overline{ABC}\ \overline{(A+B+C)}$ (Law (20))

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

$= (\bar{A} + \bar{B} + \bar{C})(\bar{A}\bar{B}\bar{C})$ (Law (23))

$= \bar{A}\bar{B}\bar{C} \cdot \bar{A} + \bar{A}\bar{B}\bar{C} \cdot \bar{B} + \bar{A}\bar{B}\bar{C} \cdot \bar{C}$ (Law (11))

$= \bar{A}\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}\bar{C}$ (Law (7))

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

$= \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$ (Law (14))

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

(iii) $\overline{(A + \overline{BC})}\ \left(A\bar{B} + \overline{ABC}\right)$

$= \left(\bar{A} \cdot \overline{\overline{BC}}\right)(A\bar{B} + \bar{A} + \bar{B} + \bar{C})$ (Law (23))

$= (\bar{A}BC)\ (\bar{A} + \bar{B} + \bar{B} + \bar{C})$ (Law (21))

$= (\bar{A}BC)(\bar{A} + \bar{B} + \bar{B}) + \bar{C}$ (Law (14))

$= \bar{A}BC \cdot \bar{A} + \bar{A}BC \cdot \bar{B} + \bar{A}BC \cdot \bar{C}$ (Law (3) and (4))

$= \bar{A}BC + 0 + 0$ (Law (12))

$= \bar{A}BC.$

(iv) $\overline{[ABC + \bar{A}\bar{B}]} + AC$

$= \left(\overline{\overline{ABC + \bar{A}\bar{B}}}\right) \cdot \overline{AC}$ (Law 23)

$= (ABC + \bar{A}\bar{B}) \cdot (\bar{A} + \bar{C})$ (Law 11)

$= ABC \cdot \bar{A} + ABC \cdot \bar{C} + \bar{A}\bar{B}\bar{A} + \bar{A}\bar{B}\bar{C}$ (Law (4) and (3))

$= 0 + 0 + \bar{A}\bar{B} + \bar{A}\bar{B}\bar{C}$ (Law (12) and (6))

$= \bar{A}\bar{B}(1 + \bar{C}) = \bar{A}\bar{B} \cdot 1$ (Law 13)

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

COLLEGE OF COMMERCE
LIBRARY
BSC IT
743

Scanned by CamScanner

(v) $\overline{(AE) + ABC}\,(A\bar{B}C)$

$= \overline{(\bar{A} + \bar{B} + ABC)}\,(A\bar{B}C)$ (Law (11))

$= \overline{(\bar{A} \cdot A\bar{B}C + A\bar{B}C\bar{B} + ABC \cdot A\bar{B}C)}$ (Law (7))

$= \overline{(0 + A\bar{B}C + 0)}$ (Law (4) and (3))

$= A\bar{B}C$ (Law 12)

$= \bar{A} + B + \bar{C}$ (Law 11)

**Ex. 8 :** Prove that $\bar{X}(X + Y) + \bar{Z} + ZY = Y + \bar{Z}$

**Soln. :** L. H. S

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

$= \bar{X} \cdot X + \bar{X} \cdot Y + \bar{Z} + ZY$ (Law 7)

$= 0 + \bar{X} \cdot Y + \bar{Z} + Y$ (Law 4 and 21)

$= Y + \bar{X} \cdot Y + \bar{Z}$

$= Y(1 + \bar{X}) + \bar{Z}$ (Law 7)

$= Y \cdot (1) + \bar{Z}$ (Law 13)

$= Y + \bar{Z}$ (Law 2)

**Ex. 9 :** Prove that $WX + X\bar{Y} + YZ + X\bar{Z} = X + YZ$

**Soln. :** L. H. S.

$= WX + X\bar{Y} + YZ + X\bar{Z}$ (Law 7)

$= XW + X(\bar{Y} + \bar{Z}) + YZ$ (Law 11)

$= XW + X(\overline{YZ}) + YZ$

$= XW + YZ + (\overline{YZ}) \cdot X$ (Law 20)

$= XW + YZ + X$ (Law 7)

$= X(W + 1) + Y(Z)$ (Law 13)

$= X + YZ$

**Ex. 10 :** Simplify following :

(i) $(A + \bar{B} + AB)(A + \bar{B})(\bar{A}B)$

$= (A + \bar{B} + A)(A \cdot \bar{A}B + \bar{B} \cdot \bar{A}B)$ (Law (21) and (7))

Scanned by CamScanner

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

$= (A + \bar{B}) \cdot 0$ (Law (14) and (4))

$= 0$. (Law (13))

(ii) $(AB + C)(AB + CD)$

$= ABAB + ABCD + ABC + CCD$

$= AB + ABCD + ABC + CD$ (Law 3)

$= AB(1 + CD + C) + CD$ (Law 7)

$= A \cdot B \cdot 1 + CD$ (Law 13)

$= AB + CD$. (Law 2)

(iii) $A\bar{B} + \bar{A}\bar{B} + \overline{AB} + AB$

(Law 15)

$= A\bar{B} + \bar{A}\bar{B} + 1$

$= 1$ ($AB = x \therefore \bar{x} + x = 1$)

(iv) $XW + (XY + Y)X$ (Law 13)

$= XW + XXY + XY$

$= XW + XY + XY$ (Law 7)

$= XY + XW$. (Law 3)

(v) $\overline{(AB + ABC)(A\bar{B}C)}$

$= \overline{AB(1 + C)(A\bar{B}C)}$ (Law 7)

$= \overline{(AB)(A\bar{B}C)}$ (Law 13)

$= \overline{ABA\bar{B}C}$ (Law 4) $\longrightarrow B\bar{B} = 0$

$= \bar{0}$

$= 1$

Uptill now we have solved some examples but we haven't drawn final circuit. Now let's solve some examples to minimise Boolean expressions and finally draw the circuit.

**Ex. 11 :** Using rules of Boolean algebra, solve example and implement using gates. Draw circuit diagram neatly :

(i) $(A + BC)\left(B + \bar{C}A\right)$

(ii) $Y = A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + ABCD + A\bar{B}CD$

(iii) $Y = \overline{\overline{AD + \overline{CD}} \cdot \overline{BD + CD}}$

Scanned by CamScanner

**Soln. :** (i) $(A + BC)\left(B + \bar{C}A\right)$

$= AB + A \cdot A\bar{C} + BC \cdot B + BC \cdot \bar{C}A$

$= AB + A\bar{C} + BC + 0$

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

The circuit is as shown in Fig. 3.24.

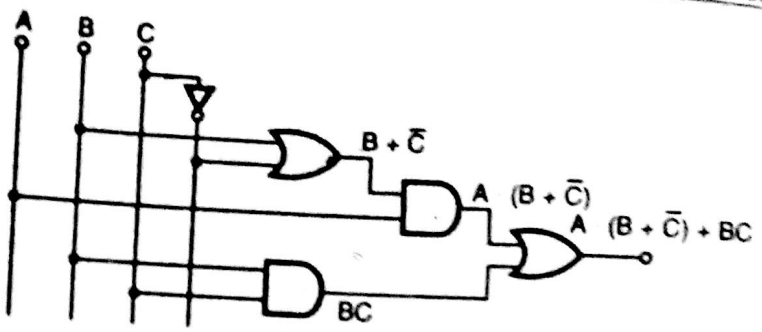



Fig. 3.24

(ii) $AB\bar{C}\bar{D} + AB\bar{C}D + ABC\bar{D} + ABCD + A\bar{B}CD$

$= AB\bar{C}\left(\bar{D} + D\right) + ABC\,(D + D) + A\bar{B}CD$ (Law 7)

$= AB\bar{C} \cdot 1 + ABC \cdot 1 + A\bar{B}CD$ (Law 15)

$= AB\bar{C} + ABC + A\bar{B}CD$ (Law 2)

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

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

$= A\left(B + \bar{B}\,CD\right)$

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

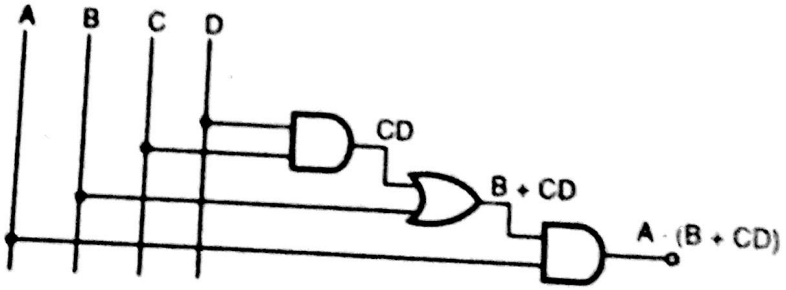



Fig. 3.25

(iii) $\overline{\overline{AD + \bar{C}\bar{D}} \cdot \overline{BD + CD}}$

$= \overline{\overline{AD + \bar{C}\bar{D}}} + \overline{\overline{BD + CD}}$

$= AD + \bar{C}\bar{D} + BD + CD$

$= BD + AD + \bar{C}\bar{D} + CD = AD + BD + 1 = 1$

---

**12 :** *Simplify expression and draw circuit.*

(i) $Y = A\left(B + C\left(\overline{AB + AC}\right)\right)$

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

Scanned by CamScanner

(iii) $Y = C\left(\overline{ABC} + A\bar{B}C\right)$

(iv) $Y = \left(\overline{\overline{(\bar{A}+B)} + \overline{(A+\bar{B})}}\right) + \left(\overline{\overline{(\bar{A}\,B)}\;\overline{(A\,\bar{B})}}\right)$

**Soln. :** (i) $Y = \left(AB + C\left(\overline{AB + AC}\right)\right)$

$= A\left(B + C\left[\overline{AB} \cdot \overline{AC}\right]\right)$ Demorgan

$= A\left[B + C\left(\left(\bar{A}+\bar{B}\right)\cdot\left(\bar{A}+\bar{C}\right)\right)\right]$ Demorgan

$= A\left[B + C\left(\bar{A}\cdot\bar{A} + \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B}\bar{C}\right)\right]$

$= A\left[B + C\left(\bar{A}\left(1 + \bar{C} + \bar{B}\right) + \bar{B}\bar{C}\right)\right]$

$= A\left[B + C\left(\bar{A}\cdot 1 + \bar{B}\bar{C}\right)\right]$

$= A\left[B + \bar{A}C + \bar{B}\bar{C}\cdot C\right]$

$= A\left[B + \bar{A}C + 0\right]$

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

$= AB + 0 = AB$

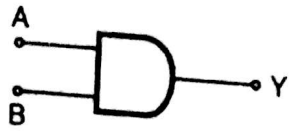



Fig. 3.26

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

$= X\cdot\bar{X} + XY + XZ + \bar{X}Z + YZ + Z\cdot Z$

$= 0 + XY + XZ + \bar{X}Z + YZ + Z$

$= XY + Z\left[X + \bar{X} + Y + 1\right]$

$= XY + Z\,[1] = XY + Z$

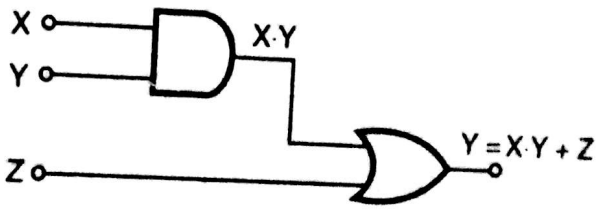



Fig. 3.27

(iii) $Y = C\left(\overline{ABC} + A\bar{B}C\right)$

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

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

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

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

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

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

Scanned by CamScanner

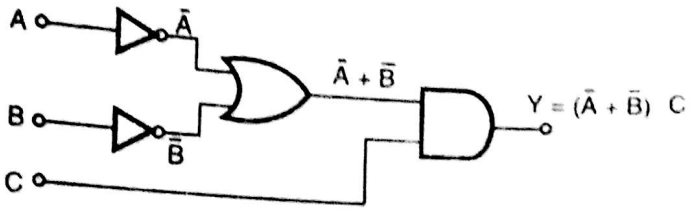



**Fig. 3.28**

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

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

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

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

$= 0 + \underline{\bar{A}\bar{B}} + AB + 0 + \underline{\bar{A}B} + A\bar{B}$

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

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

No gates required, directly connect O/p to $V_{CC}$.

---

**Ex. 13 :** Write dual of the expression given below and prove its validity by Boolean algebra.

$(a + b)(\bar{a} + c)(b + c) = (a + b)(\bar{a} + c)$.

**Soln. :** To find dual we replace AND by OR and Vice Versa, and complement 0's and 1's.

$\therefore \quad ab + \bar{a}c + bc = ab + \bar{a}c$

R. H. S. $ab + \bar{a}c = ab \cdot 1 + \bar{a}c \cdot 1 = ab(c + 1) + \bar{a}c(b + 1) = abc + ab + \bar{a}bc + \bar{a}c$

$= abc + \bar{a}bc + ab + \bar{a}c = bc(a + \bar{a}) + ab + \bar{a}c$

$= ab + \bar{a}c + bc \longrightarrow$ (L. H. S)

## 3.8 Deriving Boolean Expression from Given Circuit :

*Under this section circuit will be given to you and you have to find out or derive Boolean expression. The method to write Boolean is very simple. You start from i/p side (normally left), and go on writing o/p expression of each gate till you reach final o/p terminal (normally Y). Here switch n/w will be also given to you. I hope you remember AND/OR circuit using switches.*

*Let's see some examples.*

---

***14 :** Write down equation for output Y for the circuit shown in Fig. 3.28 (A).*

Scanned by CamScanner

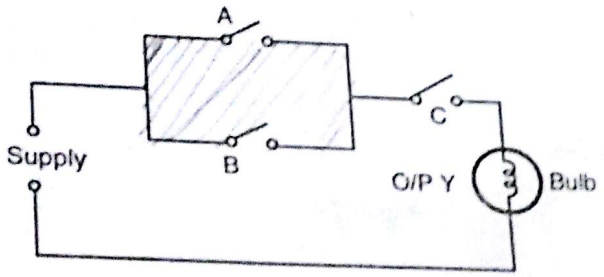



Fig. 3.28 (A)

**Soln. :** As shown in Fig. 3.28 (A) switches A and B forms OR gate, and C is in series forms AND gate.

∴ equation is $Y = (A + B) \cdot C$

Refer Fig. 3.29.

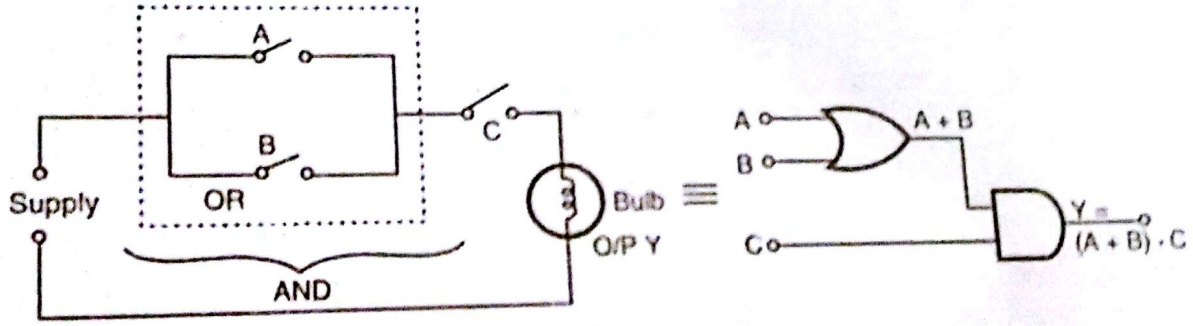



Fig. 3.29

**Ex. 15 :** Write down equation for switch circuit shown in Fig. 3.30.

**(i)**

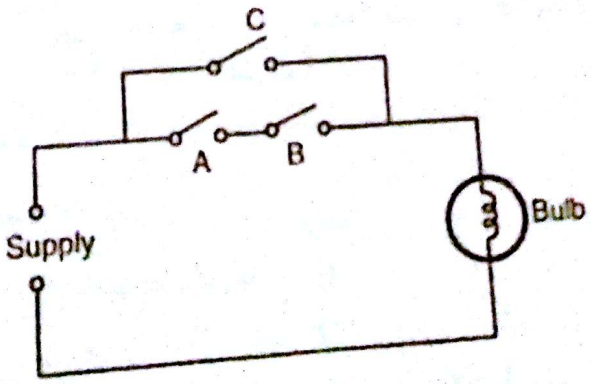



Fig. 3.30

**Soln. :** $Y = AB + C$ i.e. [A AND B] OR C will allow the bulb to glow.

Equivalent circuit is shown in Fig. 3.31.

Scanned by CamScanner

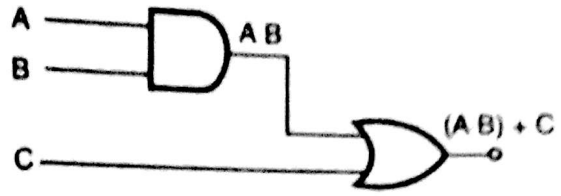



Fig. 3.31

**(ii)**

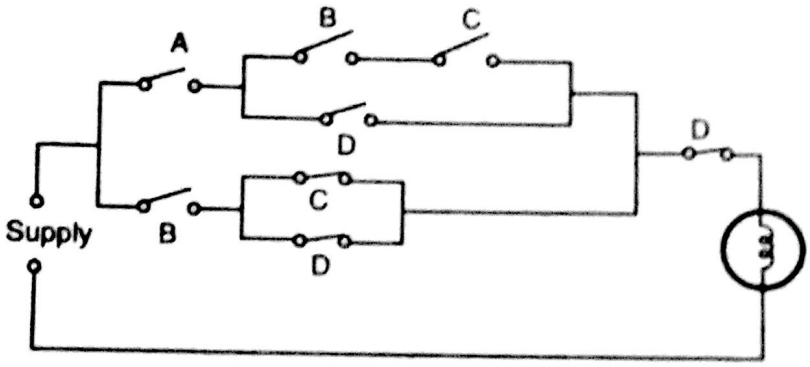



Fig. 3.32

**Soln. :** Always start from inner loops.

$$\therefore \quad Y = \bar{D}\left[A(BC + D) + B\left(\bar{C} + \bar{D}\right)\right] = \bar{D}\left[ABC + AD + B\bar{C} + B\bar{D}\right]$$

$$= ABC\bar{D} + B\bar{C}\bar{D} + B\bar{D}$$

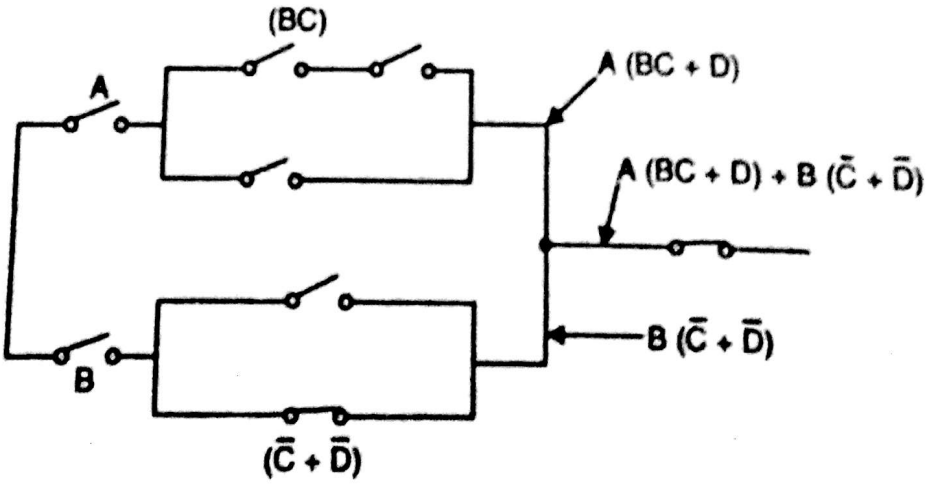



Fig. 3.33

Scanned by CamScanner

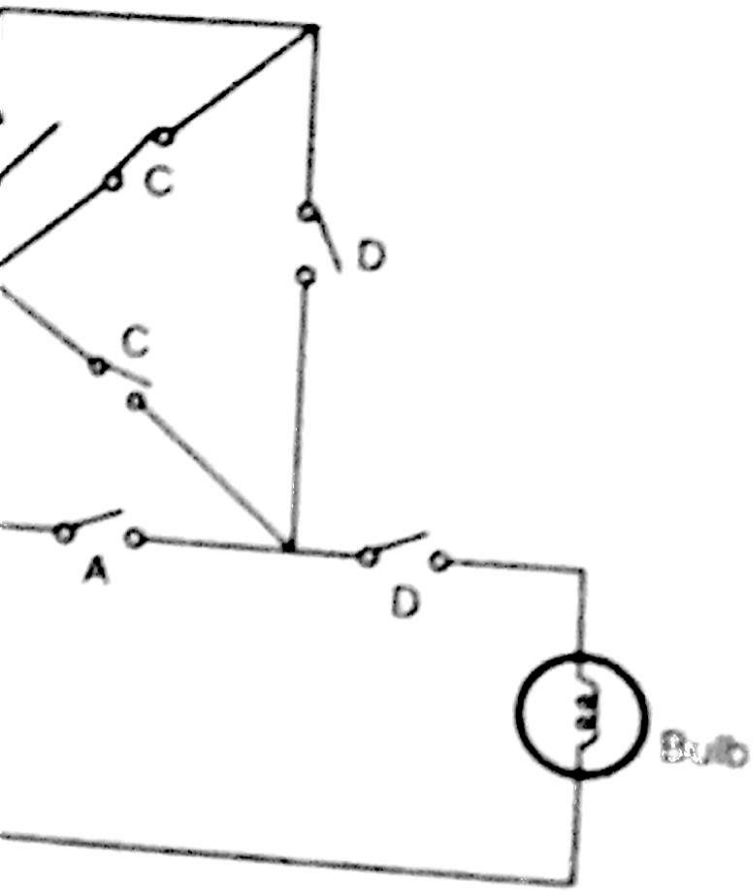



Fig. 3.34

n as follows

Scanned by CamScanner

**(iv)**

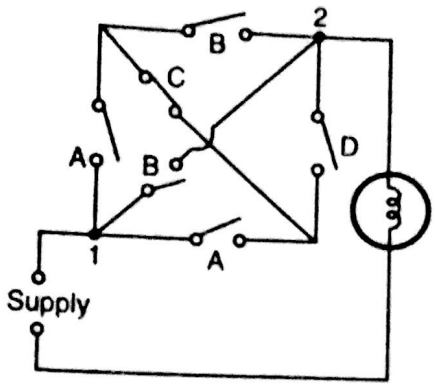



**Fig. 3.36**

**Soln. :** Rearranging the circuit we get,

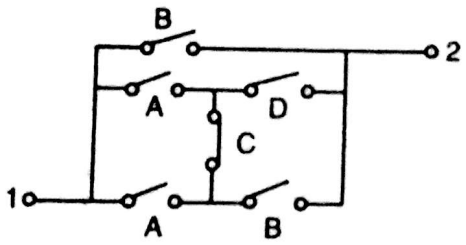



**Fig. 3.37**

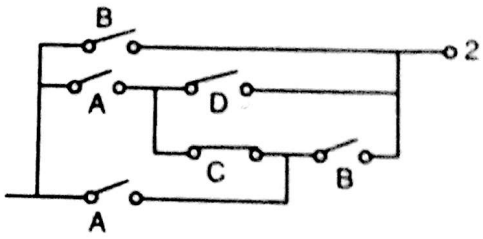



**Fig. 3.38**

Starting from input we have,

$$\bar{B} \text{ OR } \{A \text{ AND } [D \text{ OR } (\bar{C} \text{ AND } B)]\} \text{ OR}$$

$$\{A \text{ AND } [B \text{ OR } (\bar{C} \text{ AND } D) \text{ OR } (\bar{C} \text{ AND } A \text{ AND } \bar{B})]\}$$

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

**Ex. 16 :** Write down Boolean expressions for given digital circuit.

***(i)***

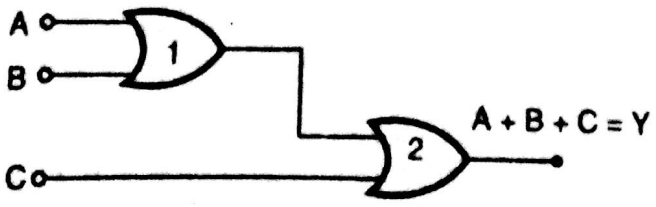



**Fig. 3.39**

Scanned by CamScanner

**Soln. :** Output of 1st of gate = A + B $\boxed{\therefore Y = A + B + C}$

(ii)

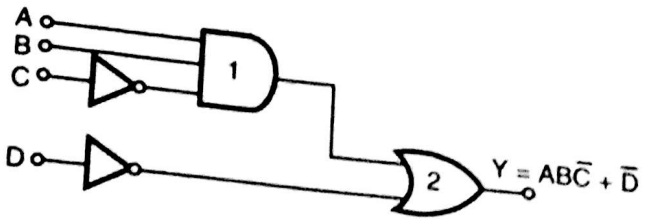



**Fig. 3.40**

**Soln. :** Input to AND 1 is A, B and $\bar{C}$ $\therefore$ O/p of AND 1 = $AB\bar{C}$.

Input to OR 2 is o/p of AND 1 and $\bar{D}$.

$\therefore$ Final $\boxed{Y = AB\bar{C} + \bar{D}}$

(iii)

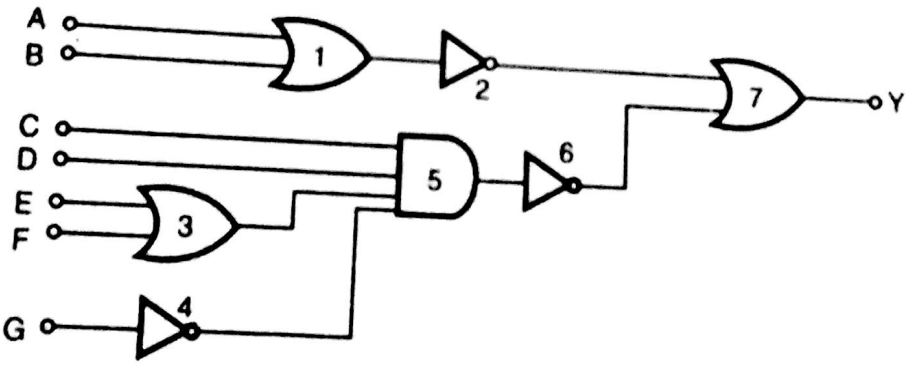



**Fig. 3.41**

(1) O/p of OR1 is A + B.

(2) O/p of INV2 is $\overline{A + B}$

(3) O/p of OR3 is E + F.

(4) O/p of AND5 is $\underline{CD\,(E + F)\,\bar{G}}$

(5) O/p of INV6 is $CD\,(E + F)\bar{G}$

$\therefore$ Final o/p is i.e. $\boxed{Y = \overline{A + B} + \overline{CD\bar{G}\,(E + F)}}$

Scanned by CamScanner

(iv) Write the equation for output X for the circuit shown in Fig. 3.42. Minimise the equation using Boolean identity.

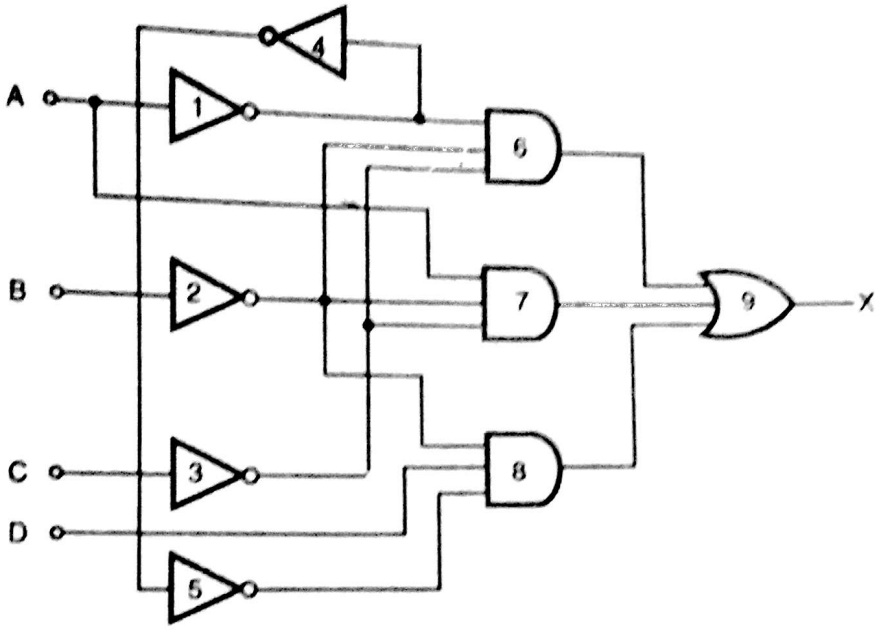



Fig. 3.42

**Soln. :**

(1) O/p of INV1 is $\longrightarrow \bar{A}$

(2) O/p of INV2 is $\longrightarrow \bar{B}$

(3) O/p of INV3 is $\longrightarrow \bar{C}$

(4) O/p of INV4 is $\longrightarrow A$

(5) O/p of INV5 is $\longrightarrow \bar{A}$

$\therefore$ O/p of AND6 $\longrightarrow \bar{A}\bar{B}\bar{C}$

O/p of AND7 $\longrightarrow A\bar{B}\bar{C}$

O/p of AND8 $\longrightarrow \bar{A}D\bar{B}$

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

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

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

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

$\therefore$ Circuit is as shown in Fig. 3.43.

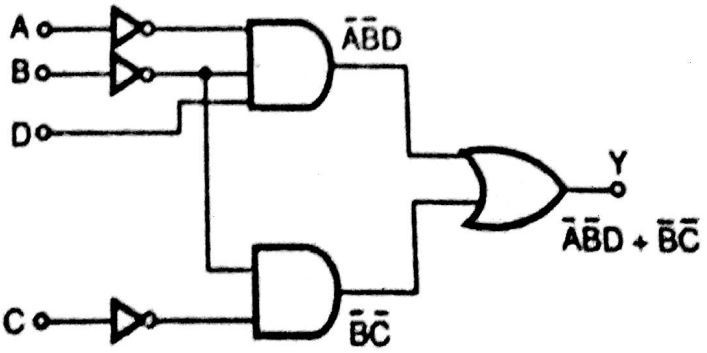



Fig. 3.43

Scanned by CamScanner

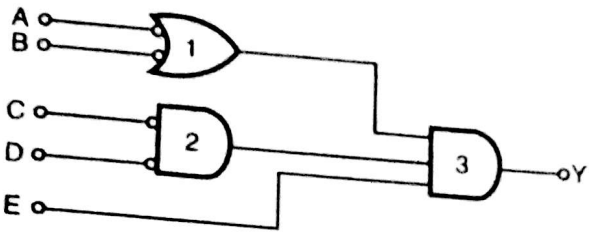



Fig. 3.44

**Soln. :** Here bubble represents Inversion

$\therefore$ O/p of gate 1 $\longrightarrow \bar{A} + \bar{B}$

O/p of gate 2 $\longrightarrow \bar{C} \cdot \bar{D}$

O/p of gate 3 $\longrightarrow Y = \left(\bar{A} + \bar{B}\right) \cdot \left(\bar{C} \cdot \bar{D}\right) \cdot E$

$= \bar{C}\,\bar{D}\,E\left(\bar{A} + \bar{B}\right)$

## 3.9 Exclusive OR Gate–Exclusive NOR Gate :

Exclusive OR gate is also termed as EX–OR or XOR gate. The Fig. 3.45 (a) shows symbol of EX–OR gate and Fig. 3.45 (b) shows truth Table of the same.




(a)

| Inputs | | Output |
|---|---|---|
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 → $\bar{A}B$ |
| 1 | 0 | 1 → $A\bar{B}$ |
| 1 | 1 | 0 |

(b)

Fig. 3.45

As shown in truth Table O/P Y is function of minterm 1 and 2. i.e. $Y = A\bar{B} + \bar{A}B$. This can be also written as $A \oplus B$. Where $\oplus \longrightarrow$ sign for EX – ORing.

**Basic Property :**

The property of the gate is O/P is HIGH only if 'odd' number of inputs are at logic '1'. If 'Even' numbered are present O/P is logic '0'.

Fig. 3.46 shows equivalent circuit for EX–OR using AND, OR and Inverter.

Scanned by CamScanner

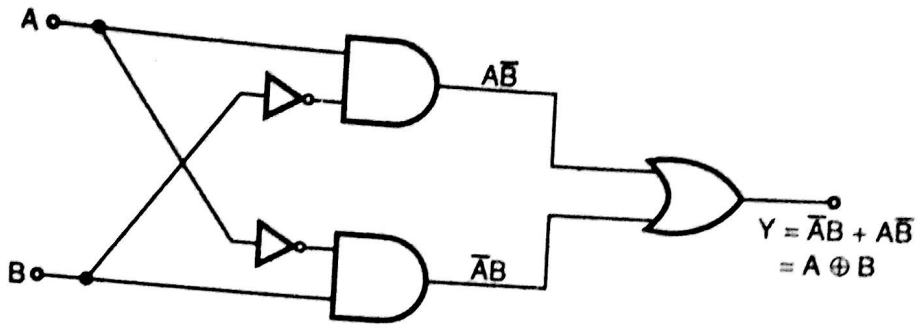
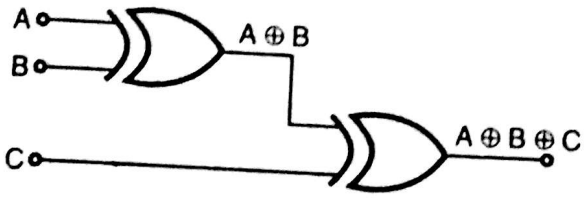



**Fig. 3.46**

Presently, we have seen truth Table of 2 input XOR gate. We can increase number of inputs by cascading the gates. The circuit and truth Table is shown in Fig. 3.47 for 3 input XOR.




(a)

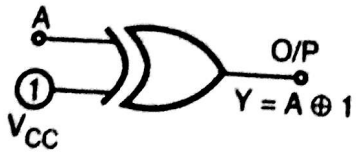
| Inputs | | | Output |
|---|---|---|---|
| A | B | C | Y |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

(b)

**Fig. 3.47**

**EX–OR Law :** Now we will see laws for EXOR.

**Case I : One terminal tied to $V_{CC}$.**




(a)

| Inputs | | Output |
|---|---|---|
| A | Logic 1 | Y |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

(b)

**Fig. 3.48**

Fig. 3.48 (a) shows one of the terminal of XOR is tied to $V_{CC}$. Available input is given to 2nd terminal.

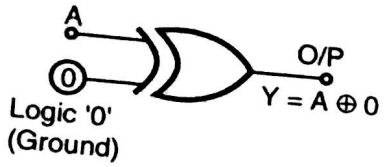
(a) When A = 0 , O/P $Y = A \oplus 1 = 0 \oplus 1 = 1$

(b) When A = 1 , O/P $Y = A \oplus 1 = 1 \oplus 1 = 0$

$\therefore$ Conclusion is O/P $Y = \bar{A}$ when one terminal tied to $V_{CC}$. The truth Table is also shown in Fig. 3.48 (b). $\boxed{Y = A \oplus 1 = \bar{A}}$

Scanned by CamScanner

**Case II : One terminal tied to GND.**




(a)

| Inputs | | Output |
|---|---|---|
| A | Logic '0' | Y |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

(b)

**Fig. 3.49**

Fig. 3.49 (a) shows that one terminal of XOR tied to GND (logic '0') and A variable is given to another terminal.

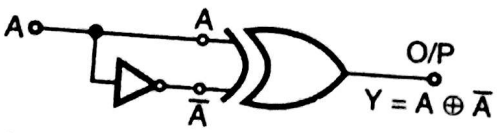
(a) When A = 0, O/P Y = A ⊕ 0 = 0 ⊕ 0 = 0

(b) When A = 1, O/P Y = A ⊕ 0 = 1 ⊕ 0 = 1

∴ O/P follows inputs.

∴ $\boxed{Y = A \oplus 0 = A}$

**Case III : One I/P is inversion of other.**




(a)

| Inputs | | Output |
|---|---|---|
| A | $\bar{A}$ | Y |
| 0 | 1 | 1 |
| 1 | 0 | 1 |

(b)

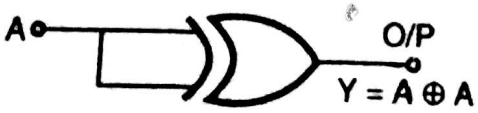
**Fig. 3.50**

(a) When A = 0, O/P Y = A ⊕ $\bar{A}$ = 0 ⊕ 1 = 1.

(b) When A = 1, O/P Y = A ⊕ $\bar{A}$ = 1 ⊕ 0 = 1

∴ $\boxed{Y = A \oplus \bar{A} = 1}$

**ıse IV : Both inputs are shorted.**




(a)

| Inputs | | Output |
|---|---|---|
| A | A | Y |
| 0 | 0 | 0 |
| 1 | 1 | 0 |

(b)

**Fig. 3.51**

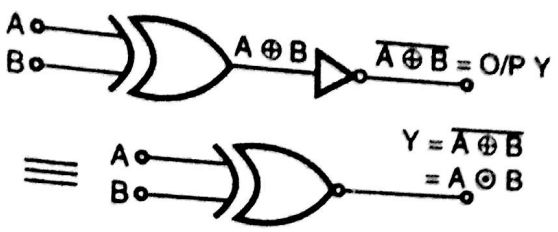
a) When A = 0, O/P Y = A ⊕ A = 0 ⊕ 0 = 0.

b) When A = 1, O/P Y = A ⊕ A = 1 ⊕ 1 = 0.

$\boxed{Y = A \oplus A = 0}$

Scanned by CamScanner

**Case V : Inverting O/P OF EX–OR gate.**




(a)

| Input | | Output |
|---|---|---|
| A | B | $Y = \overline{A \oplus B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(b)

**Fig. 3.52**

The O/P of EX–OR gate is Inverted using 'NOT' gate. $\therefore Y = \overline{A \oplus B} = \overline{A\bar{B} + \bar{A}B}$

Let's simplify the same.

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

$$= \bar{A} \cdot A + \bar{A}\bar{B} + AB + B\bar{B} = 0 + \bar{A}\bar{B} + AB + 0$$

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

As O/P is Inverted it is also called as EX–NOR gate, where 'N' in EX–NOR represents 'NOT' gate. The truth Table is shown in Fig. 3.52 (b). EX–NOR symbol is shown in Fig. 3.52 (a).

$\therefore$ O/P Y for EX–NOR = $\overline{A \oplus B} = A \odot B$

**Property of EX–NOR gate :**

Property of EX–NOR is exactly Invert of EX–OR i.e.

(1) $Y = A \odot 1 = A$ (2) $Y = A \odot 0 = \bar{A}$

(3) $Y = A \odot \bar{A} = 0$ (4) $Y = A \odot A = 1$

WE get EX-OR gate in DIP form. The IC number is 7486.

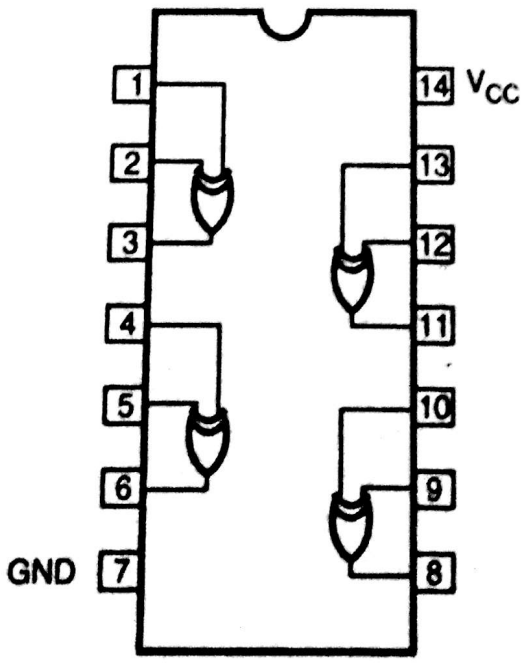



**Fig. 3.53 : 7486**

Scanned by CamScanner

Let's implement some circuit using EX-OR and EX-NOR.

**Ex. 17 :** Implement the following functions using X-OR gates and AND gates only :

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

**Soln. :** $F = A\bar{B}(C\bar{D} + \bar{C}D) + \bar{A}B(C\bar{D} + \bar{C}D)$

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

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

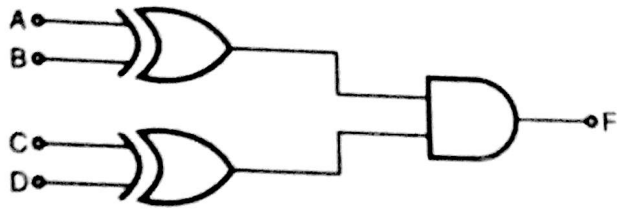



Fig. 3.54

**Ex. 18 :** Show that the expression, $f = \left(\overline{A\bar{B}}\right)\left(\overline{\bar{A}B}\right)$ is equivalent to the exclusive OR function

**Soln. :** $f = \left(\overline{A\bar{B}} \cdot \overline{\bar{A}B}\right) = \overline{A\bar{B}} + \overline{\bar{A}B}$

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

**Ex. 19 :** Implement following using EX-OR

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

**Soln. :** $F = \bar{A}B\bar{C} + \bar{A}\bar{B}C + A\bar{C}D + AC\bar{D} = \bar{A}\left(B\bar{C} + \bar{B}C\right) + A\left(\bar{C}D + C\bar{D}\right)$

$= \bar{A} \cdot (B \oplus C) + A \cdot (C \oplus D)$ ... (1)

For EX-NOR gate complement equation twice

$= \overline{\bar{A}(B \oplus C) + A(C \oplus D)}$ Use Demorgan

$= \overline{\bar{A}(B \oplus C)}\left(\overline{A(C \oplus D)}\right)$

$= \dfrac{\left[\bar{\bar{A}} + \left(\overline{B \oplus C}\right)\right]\left[\bar{A} + \left(\overline{C \oplus D}\right)\right]}{}$

$= \dfrac{(A + (B \odot C))\left(\bar{A} + (C \odot D)\right)}{}$ ... (2)

Figure for Equations (1) and (2) are shown in Figs. 3.55 (a) and (b).

Scanned by CamScanner

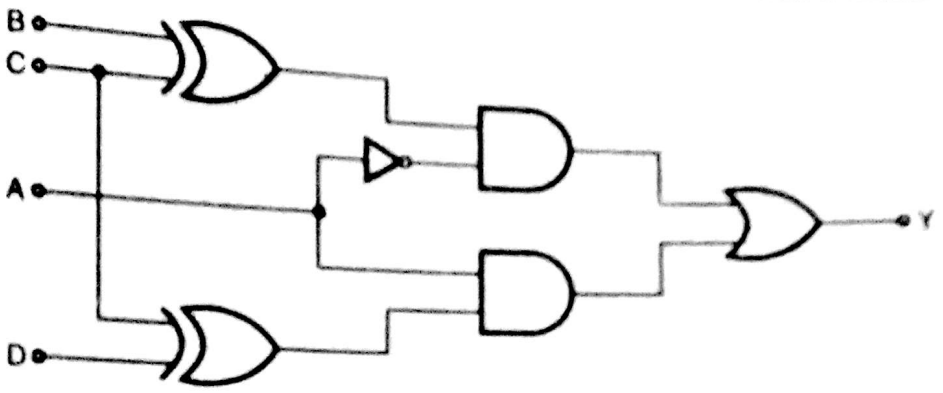



Fig. 3.55 (a)

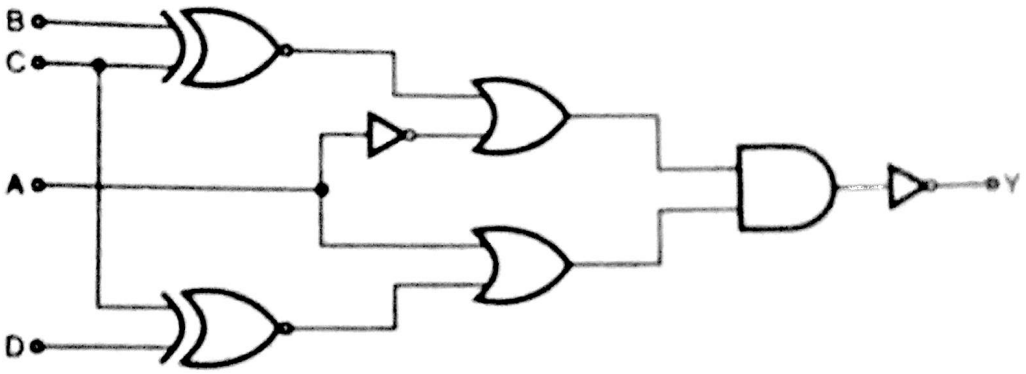



Fig. 3.55 (b)

**Ex. 20 :** Show how you can implement,

$Y = A \oplus B \oplus C \oplus D$, using three Ex–OR gates

**Soln. :** Using X – OR gates.

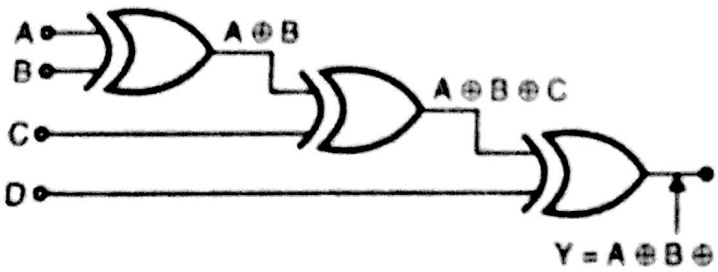



Fig. 3.56 (a)

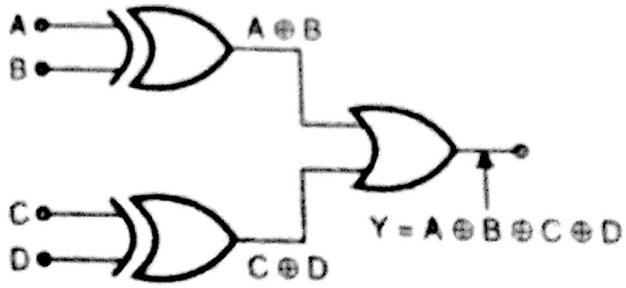



Fig. 3.56 (b)

Or, $Y = (A \oplus B) \oplus (C \oplus D)$

## 3.10 Universal Logic Gates :

It is possible to implement AND, OR, INV (NOT), XOR, XNOR gates using one member sets NAND and NOR. Thus, using these gates, one can realize any given logical expression. As a result they are known as ***Universal logic gates***. Let's first see symbol and truth Table.

Scanned by CamScanner




(a) Symbol for NAND gate

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(b) Truth table

Fig. 3.57

By observing truth Table you can conclude that NAND is nothing but AND-Inverter (NOT)




(a) Symbol for NOR gate

| Inputs | | Output |
|---|---|---|
| A | B | Y |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

(b) Truth table

Fig. 3.58

NOR is nothing but OR-NOT combination

## NAND Laws

NAND Laws will be exactly complement of AND Laws. Let's write it down.

(1) $\overline{A \cdot 0} = 1$ (One input of NAND fixed at logic 0)

(2) $\overline{A \cdot 1} = \overline{A}$ (One input of NAND fixed at logic 1)

(3) $\overline{A \cdot A} = \overline{A}$ (Both inputs are shorted to single variable. O/P is complement of input A so it is an INV)

(4) $\overline{A \cdot \overline{A}} = 1$ (If one input is complement of the other)

## NOR Laws

Same way we can write laws for NOR

(1) $\overline{A + 0} = \overline{A}$ (One input of NOR fixed at logic 0)

(2) $\overline{A + 1} = 0$ (One input of NOR fixed at logic 1)

(3) $\overline{A + A} = \overline{A}$ (Both inputs are shorted to single variable)

Scanned by CamScanner

(3) $\overline{A + A} = \bar{A}$ (Both inputs are shorted to single variable)

(4) $A + \bar{A} = 0$ (If one input is complement of the other)

## 3.11 Implementation of other Gates using Universal Logic Gates :

### 3.11.1 Use NAND to Implement all Gates :

**(i) NOT gate :**

Equation for NOT gate is $Y = \bar{A}$. Inverter acts only on single variable. But NAND has two inputs, ∴ we have to short inputs of NAND gate to make it single variable.

∴ $Y = \overline{A \cdot B} = \overline{A \cdot A} = \bar{A}$ (Law 3). Single variable B = A.

∴ O/P Y is complement of input. Observe Fig. 3.59.

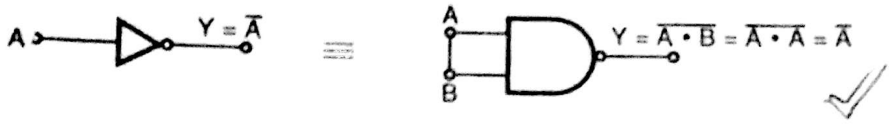



**Fig. 3.59**

**(ii) AND gate :**

Equation for AND is $Y = \overline{A \cdot B}$ ...(3.1)

Equation for NAND is $Y = \overline{A \cdot B}$ ...(3.2)

Observing these two equation we conclude that, if we Invert O/P of NAND we can get AND gate.

i.e. $Y = \overline{\overline{A \cdot B}} = A \cdot B$

But here we are going to use NAND only. An Inverter using NAND can be implemented just by starting two inputs. (Law 3 of NAND).

∴ Circuit is as shown in Fig. 3 60.

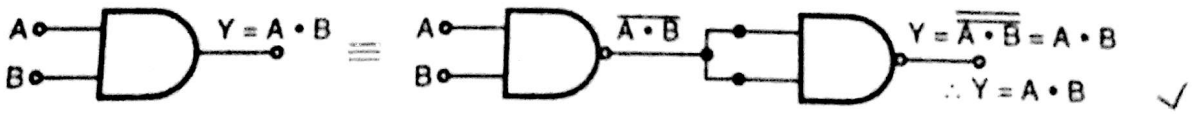



**Fig. 3.60**

**(iii) OR gate using NAND only :**

Equation for OR is $Y = A + B$ ...(3.3)

Equation for NAND is $Y = \overline{A \cdot B}$ (use demorgan to simplify).

∴ O/P of NAND is $Y = \overline{A.B}$

$= \bar{A} + \bar{B}$ ...(3.4)

Observe Equations (3.3) and (3.4), the only difference is inputs are complemented, ∴ if we already provide complemented inputs, O/P will be same as that of OR gate.

Fig 3.61 shows the circuit

Scanned by CamScanner

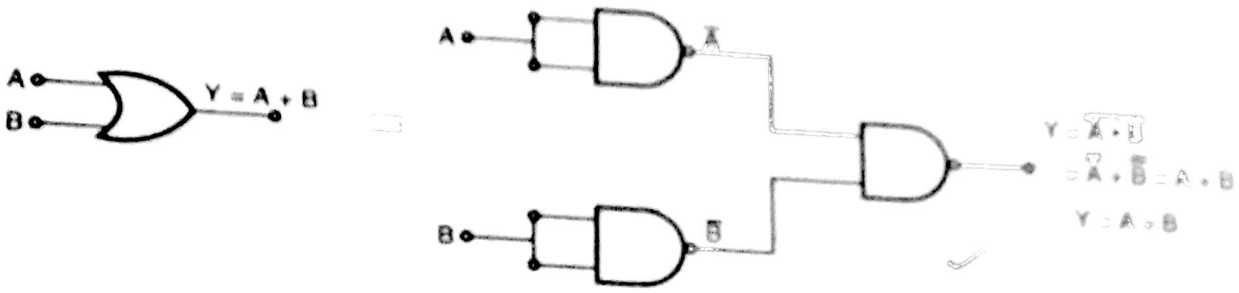



Fig. 3.61

### 3.11.2 Use NOR to Implement all the Gates :

**(i) NOT gate :** Equation for NOT is, $Y = \bar{A}$ (single variable)

$\therefore$ It is obvious that we should short input terminals of NOR also

$\therefore$ $Y = \overline{A + B}$ , as B = A, $Y = \overline{A + A} = \bar{A}$ (law 3 of NOR gate)

The circuit is as shown in Fig. 3.62.

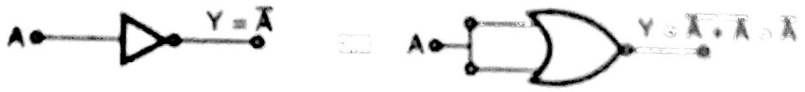



Fig. 3.62

**(ii) AND gate :**

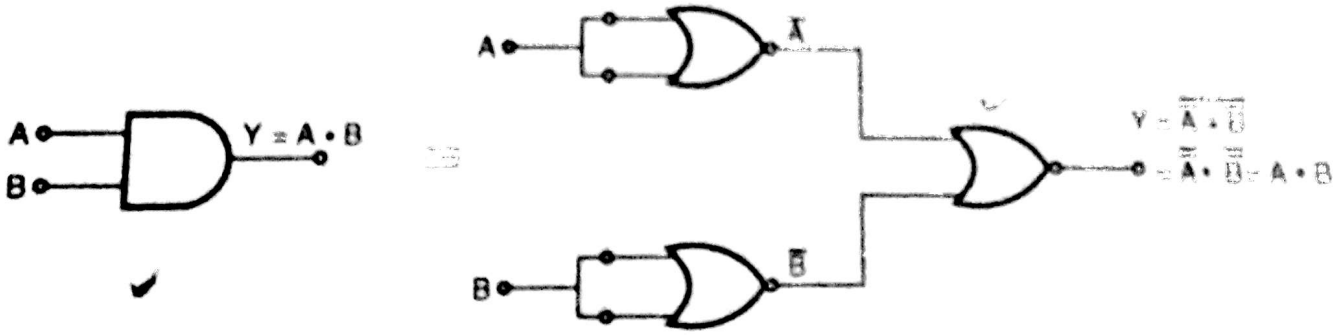



Fig. 3.63

Equation for AND is $Y = A \cdot B$ ... (3.5)

Equation for NOR is $Y = \overline{A + B}$ Use Demorgan's theorem.

$\therefore$ $Y = \bar{A} \cdot \bar{B}$ ... (3.6)

Compare Equation (3.5) and (3.6) you can conclude that inputs to NOR are complemented. $\therefore$ we should provide already complemented input. Refer Fig. 3.63.

**(iii) OR gate :** Simply complement O/P of NOR, you will get OR gate.

Scanned by CamScanner

$Y = \overline{\overline{A + B}} = A + B$ refer Fig. 3.64.

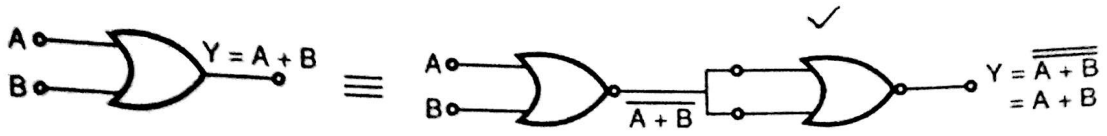



**Fig. 3.64**

Now we are going to implement given expressions using NAND and NOR gate.

## 3.12 Implementation Using NAND Gate

There are two methods to implement any given Boolean expression using NAND gate.

**Method I :** The steps are as follows :

| | | |
|---|---|---|
| **Step I** | : | Draw AND, OR and INV circuit for given Boolean expression. |
| **Step II** | : | You have already built standard blocks for AND, OR and INV using NAND. Simply go on replacing it by standard block. |
| **Step III** | : | Check for any double Inversion. Remove if any. Because double inversion $\left(\overline{\overline{A}}\right)$ is variable itself (A). |

Let's see example for same.

Implement IJ + KL using NAND only.

The circuit using AND/OR is shown in Fig. 3.65.

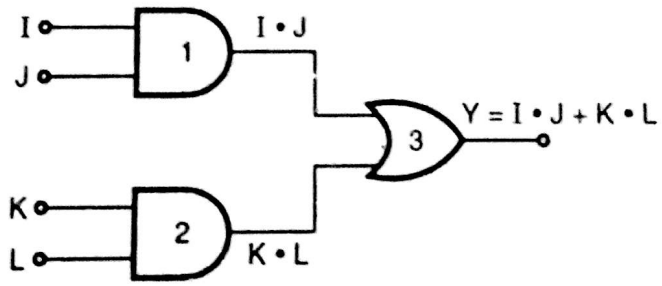



**Fig. 3.65**

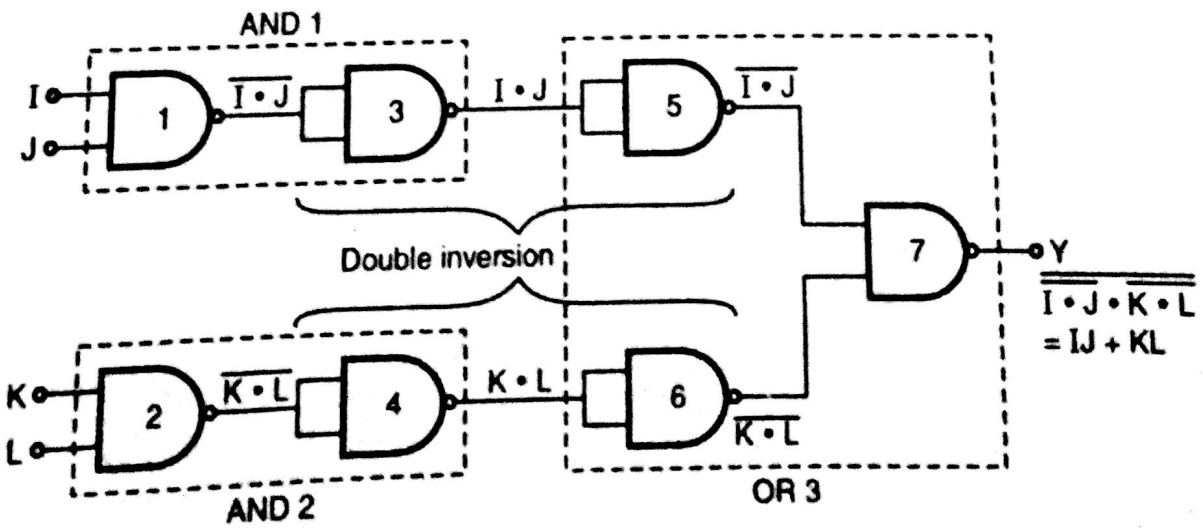



**Fig. 3.66**

Scanned by CamScanner

)R by standard block as shown in Fig. 3.66. If you observe Fig. 3.66 NAND ersion. ∴ These two gates can be eliminated. ∴ we get reduced form as shown

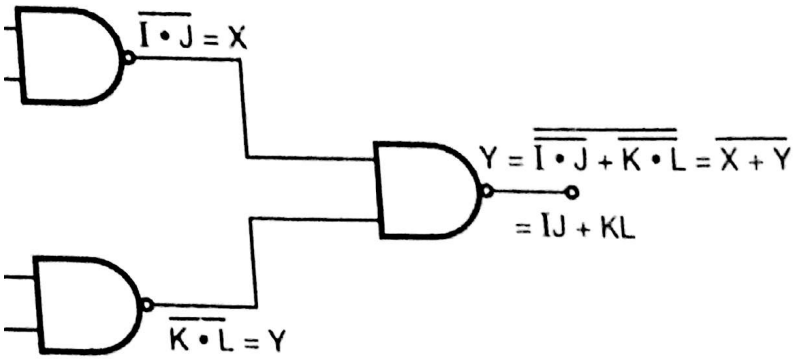



**Fig. 3.67**

nod you have to simplify equation and bring down it to form $\overline{A \cdot B}$, so that it by NAND gate. Only you should take one precaution that you don't manipulate

ression, you simply ***double complement*** the equation, for e.g. take IJ + KL.

$$\overline{I \cdot J} \cdot \overline{K \cdot L} = \overline{\overline{I \cdot J} \cdot \overline{K \cdot L}}$$

$$= \overline{X \cdot Y} \quad \left(X = \overline{I \cdot J}, Y = \overline{K \cdot L}\right)$$

⟶ NANAD 1

⟶ NAND 2

NAND 3

vn in Fig. 3.67.

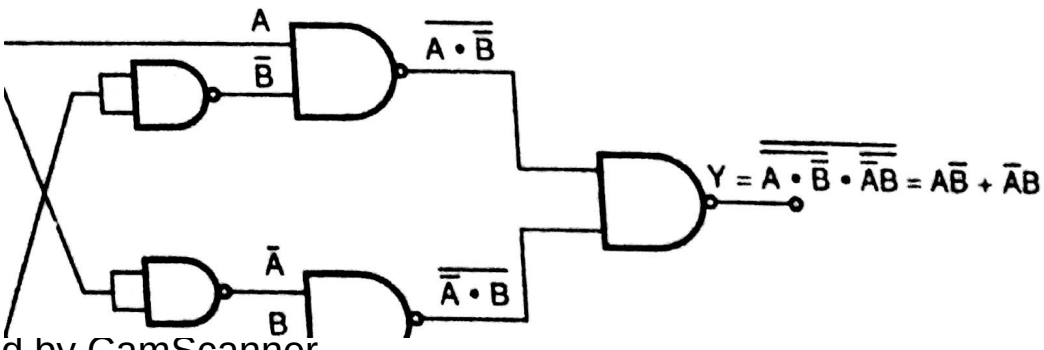
---

following expression using NAND gate :

+ $\bar{A}B$ (ii) $Y = X + \bar{W}Z$ (iii) $Y = \overline{C \cdot (A + B)} \cdot D$

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

shown in Fig. 3.68 (a).




Scanned by CamScanner

...changed to single variable)



(ii) $Y = X + \bar{W}Z = X + \bar{W}Z = \bar{X} \cdot \bar{W}Z$ The circuit is as shown in Fig. 3.68 (b).

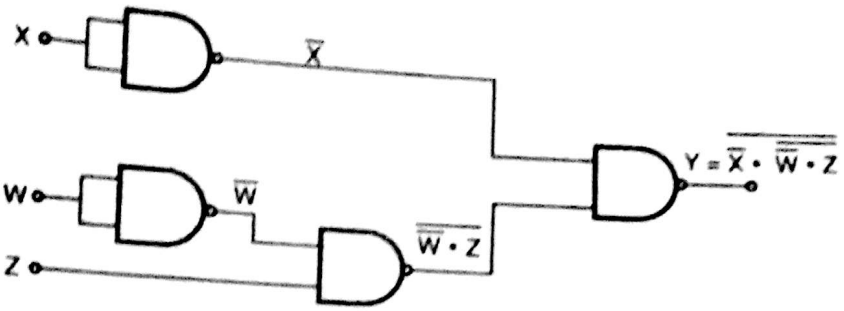



**Fig. 3.68 (b)**

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

(i In the above equation if we double complement equation and simplify it, AND will be converted to OR. ∴ the steps are

$$Y = \left[\bar{C}\left(\overline{A + B}\right)\right] D = \left[\bar{C} + \bar{A} \cdot \bar{B}\right] \cdot D$$

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

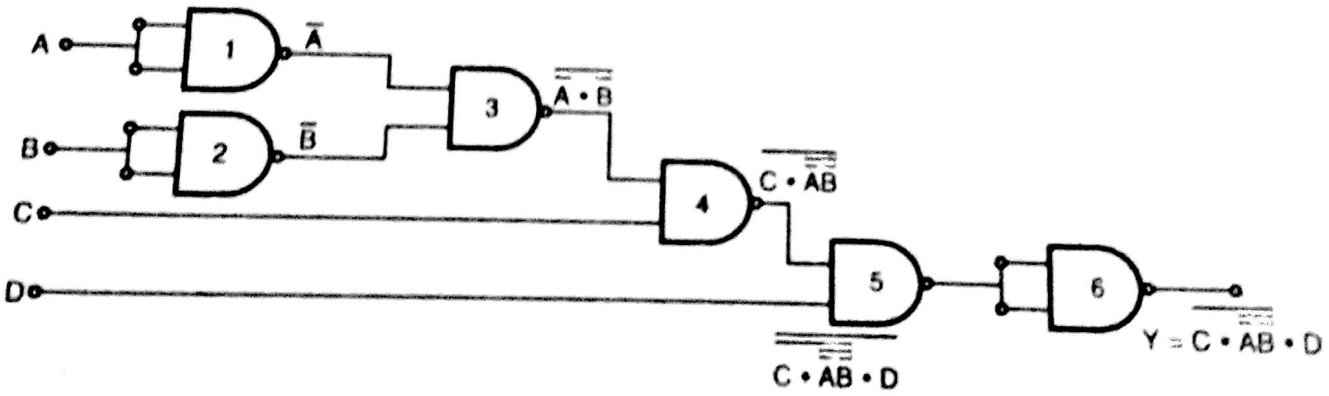



**Fig. 3.68 (c)**

*NAND 6 is used to adjust last sign of complement*

## 3.13 Implementation Using NOR Gate :

There are two methods to implement any given Boolean expression using NOR gate.

***Method I*** : This method is same as we implemented for NAND gate.

***Step I*** : Draw AND/OR/INV circuit for given Boolean expression.

***Step II*** : We have already built standard block for AND/OR/INV using NOR gate. Simply replace it by standard block.

***Step III*** : Check for double Inversion, remove if any.

Scanned by CamScanner

For example say

Y = I J + K L, Implement this using NOR only.

Fig. 3.69 shows AND/OR/INV circuit. Fig. 3.70 shows equivalent NOR circuit.

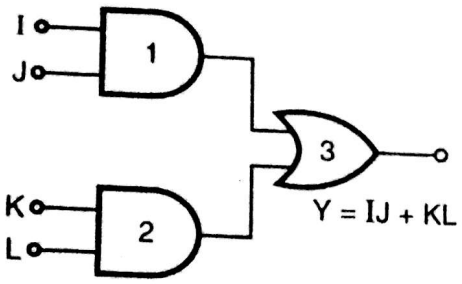



Fig. 3.69

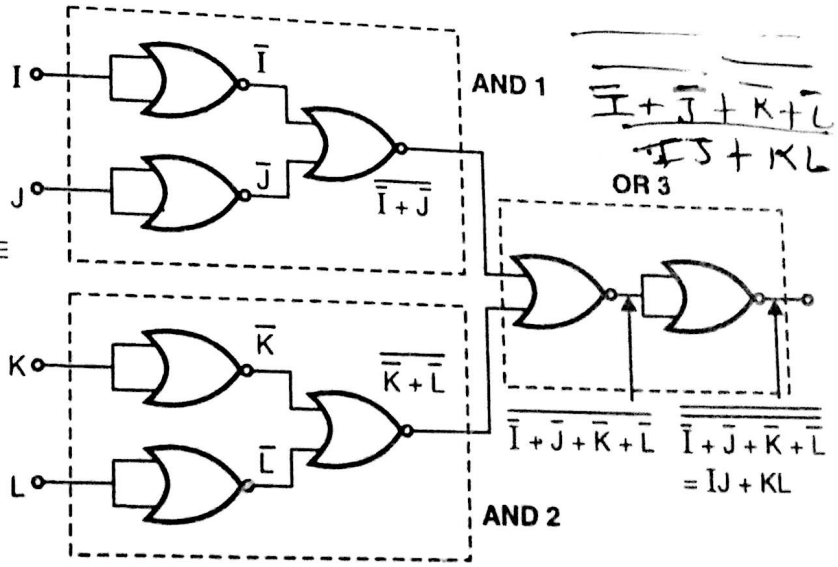



Fig. 3.70

**Method II :** In this case you have to bring down equation to $\overline{A + B}$ form by simplification. To achieve this again you double complement the equation. For example

$$Y = IJ + KL = \overline{\overline{IJ + KL}} = \overline{\overline{IJ} \cdot \overline{KL}} = \overline{\left[\bar{I} + \bar{J}\right] \cdot \left[\bar{K} + \bar{L}\right]} = \overline{\bar{I} + \bar{J}} + \overline{\bar{K} + \bar{L}}$$

$$= IJ + KL$$

The diagram is as shown in Fig. 3.70.

NAND and NOR in DIP Package.

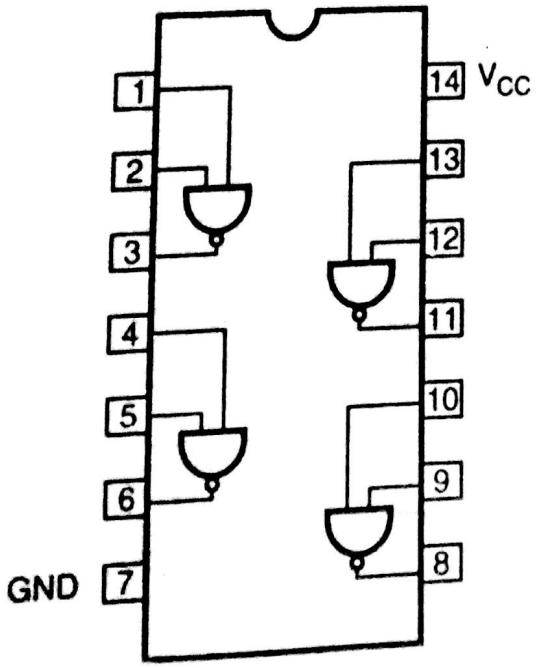



Fig. 3.70 (A) : 7400 → NAND

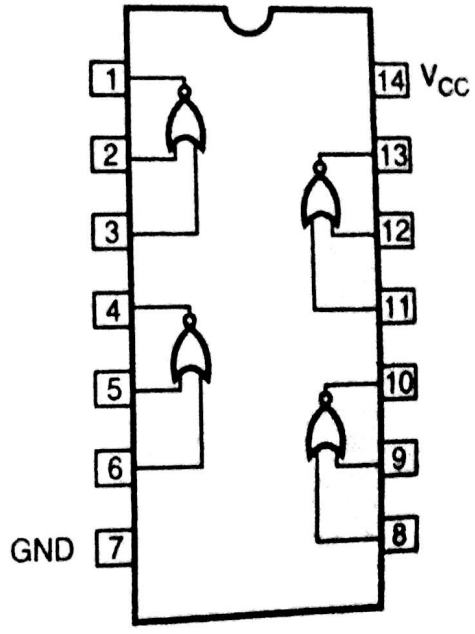



Fig. 3.70 (B) : 7402 → NOR

Scanned by CamScanner

say

L, Implement this using NOR only.

ws AND/OR/INV circuit. Fig. 3.70 shows equivalent NOR circuit.

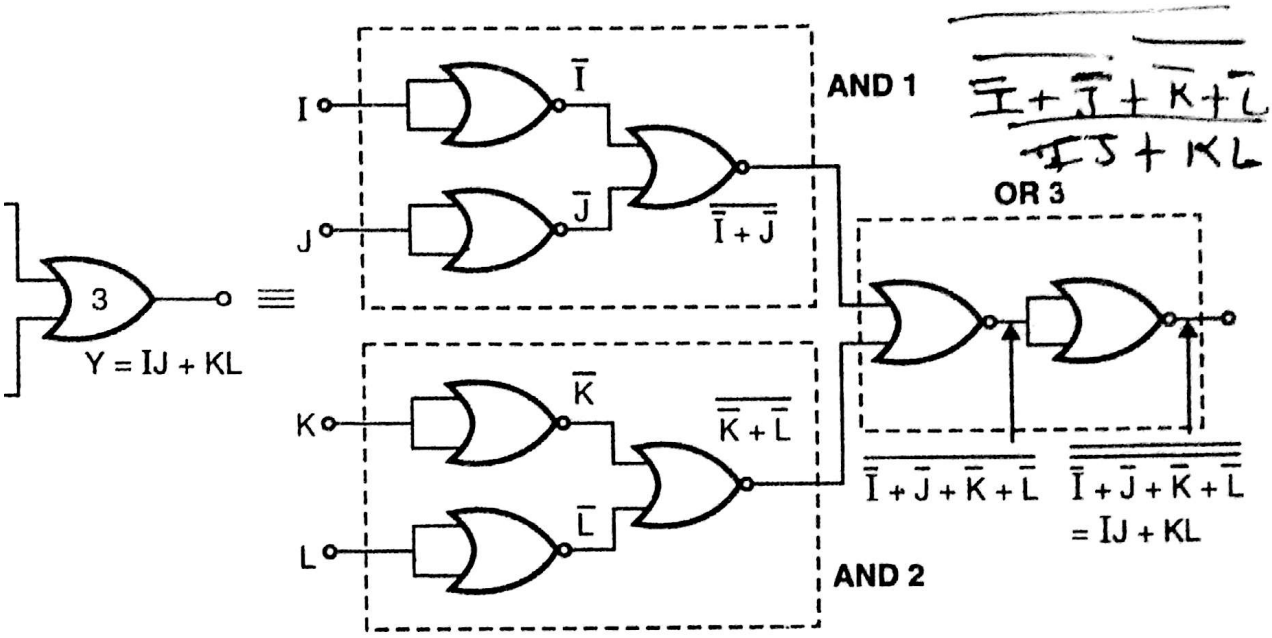



**Fig. 3.69**

**Fig. 3.70**

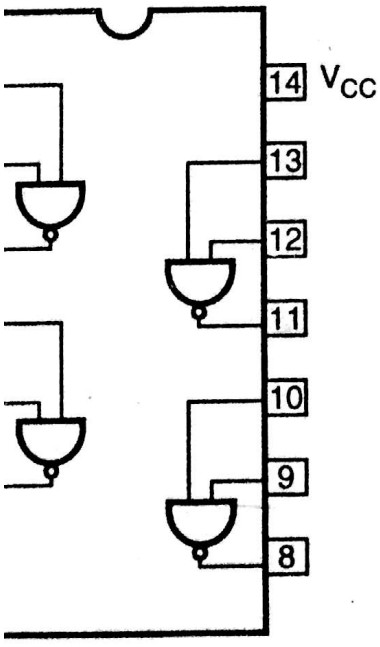
his case you have to bring down equation to $\overline{A+B}$ form by simplification. To achieve uble complement the equation. For example

$$KL = \overline{\overline{IJ+KL}} = \overline{\overline{IJ}\cdot\overline{KL}} = \left[\bar{I}+\bar{J}\right]\cdot\left[\bar{K}+\bar{L}\right] = \bar{I}+\bar{J}+\bar{K}+\bar{L}$$

KL

is as shown in Fig. 3.70.

NOR in DIP Package.




**'0 (A) : 7400 → NAND**

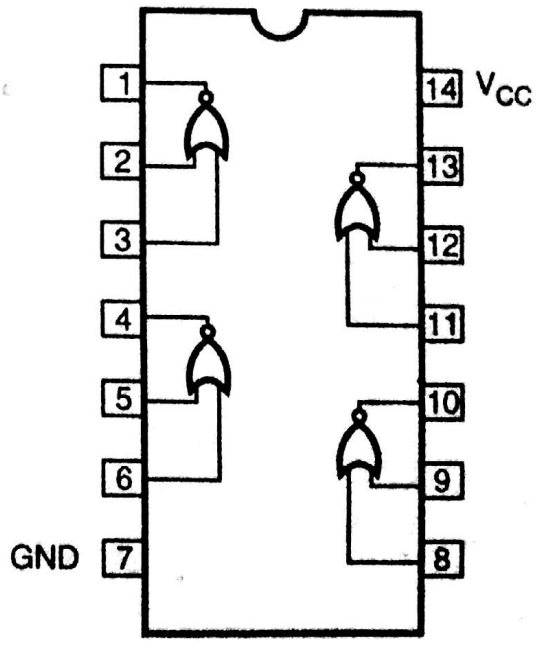



**Fig. 3.70 (B) : 7402 → NOR**

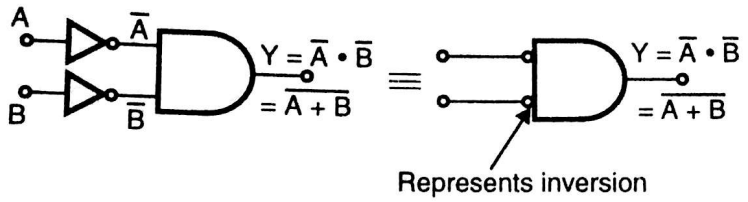
Scanned by CamScanner

## 3.14 Input Bubbled Logic :

In the discussion so far, we saw that an Inverter or a bubble was placed at the output of NAND, NOR and X – NOR gates. Normally only bubble also represents Inversion (complement). But placing bubbles at input of gate, affects its operation.

### Input Bubbled AND gate

Let's refer Fig. 3.71.




(a) (b)

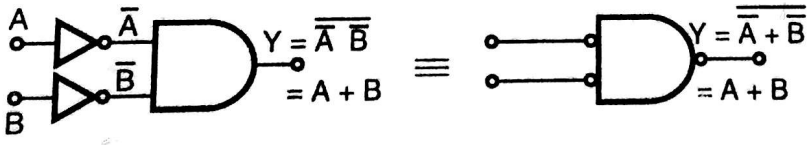
| Inputs | | | | Output |
|---|---|---|---|---|
| A | B | $\bar{A}$ | $\bar{B}$ | Y |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |

(c) Truth table

**Fig. 3.71**

Fig. 3.71 (a) shows inputs A and B are Inverted and then given to AND gate. The same circuit in modified representation can be shown as ***bubbled input***. The total gate is then referred as ***Bubbled AND gate***. The O/P of bubbled AND is $Y = \bar{A} \cdot \bar{B} = \overline{A + B}$. $\overline{A + B}$ is equation of NOR gate, ∴ bubbled AND is equal to NOR gate.

Now take NAND gate and apply bubbled input.




(a) (b)

| Inputs | | | | Output |
|---|---|---|---|---|
| A | B | $\bar{A}$ | $\bar{B}$ | Y |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |

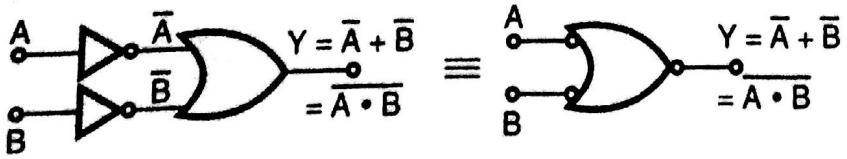
(c) Truth table

**Fig. 3.72**

Fig. 3.72 (c) show that truth Table of bubbled NAND is equal to OR gate.

### Input Bubbled OR gate

Fig. 3.73 shows two input bubbled OR gate.




(a) (b)

| Inputs | | | | Output |
|---|---|---|---|---|
| A | B | $\bar{A}$ | $\bar{B}$ | Y |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |

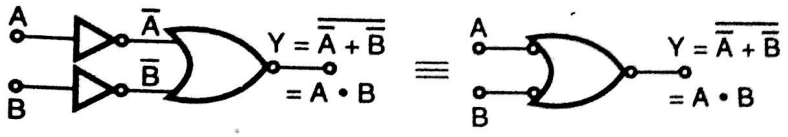
(c) Truth table

**Fig. 3.73**

Scanned by CamScanner

As shown bubbled OR gate is nothing but NAND gate. This conclusion you can derive by observing truth Table.

Now let's take bubbled NOR.




(a) (b)

| Inputs | | | | Output |
|---|---|---|---|---|
| A | B | $\bar{A}$ | $\bar{B}$ | Y |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

(c) Truth table

Fig. 3.74

Bubbled NOR is nothing but AND gate.

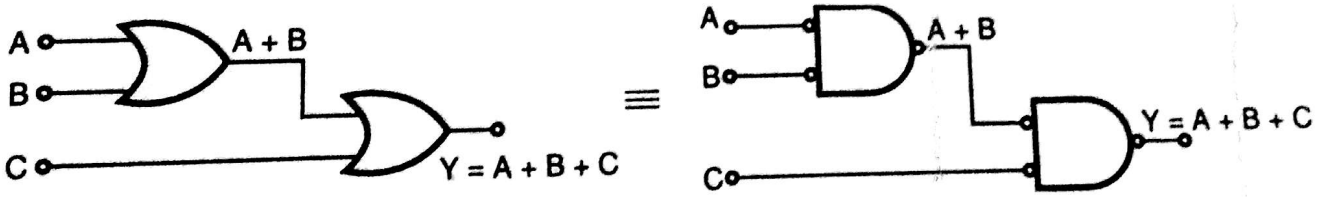
**Conclusion :**

(1) Input bubbled AND is equivalent to NOR gate.
(2) Input bubbled NAND is equivalent to OR gate.
(3) Input bubbled OR is equivalent to NAND gate.
(4) Input bubbled NOR is equivalent to AND gate.

**Ex. 22 :** Represent following expressions in terms of input bubbled AND and OR gate :

(i) $Y = A + B + C$.

**Soln. :** First draw actual circuit and then replace it by equivalent bubbled gate.

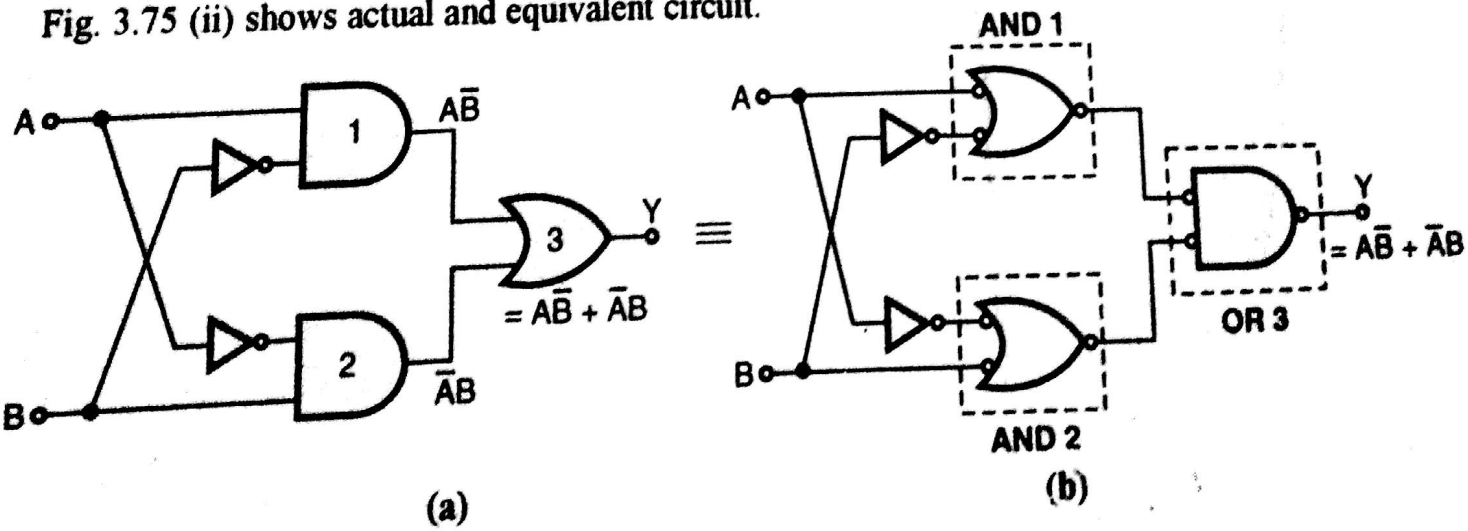



(a) Actual circuit (b) Bubbled logic circuit

Fig. 3.75 (i)

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

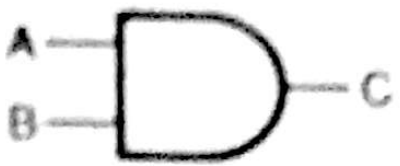
Fig. 3.75 (ii) shows actual and equivalent circuit.




(a) (b)

Scanned by CamScanner

**AND**




$C = AB$







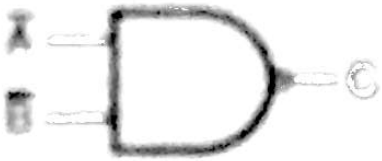



**OR**




$C = A + B$

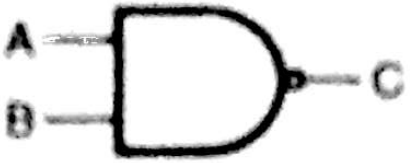






**NAND**




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







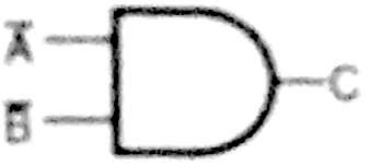



**NOR**




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

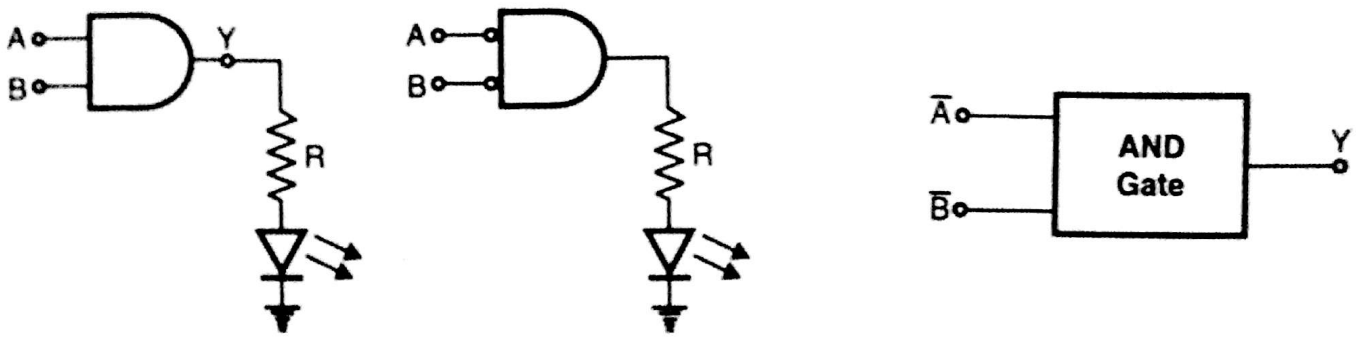






Scanned by CamScanner




(a) (b) (c)

**Fig. 3.76**

As shown when either A or B, or both are 'LOW' O/P is logic '0' and LED is OFF. But when both puts are ***asserted HIGH*** O/P will be HIGH and LED will lit.

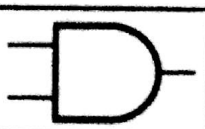
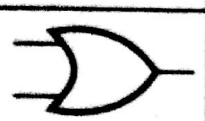
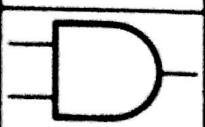
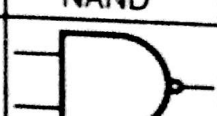
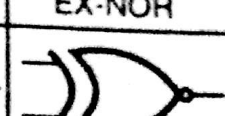
Here I would like to define ***Assertion level***. It is a level necessary to cause an event to occur.

But in some circuit assertion level is 'active low'. i.e. event occurs when there will be 'LOW' logic esent at input terminal. Refer Fig. 3.76 (b). As shown input to AND get is bubbled, therefore when ther of the input, or both are at HIGH logic, output is 'LOW', therefore LED is OFF. But if both the puts are asserted with 'LOW' signal. O/P will be HIGH and LED will glow. Normally ***active low*** inputs e represented by providing 'over bar' to it, as shown in Fig. 3.76 (c).

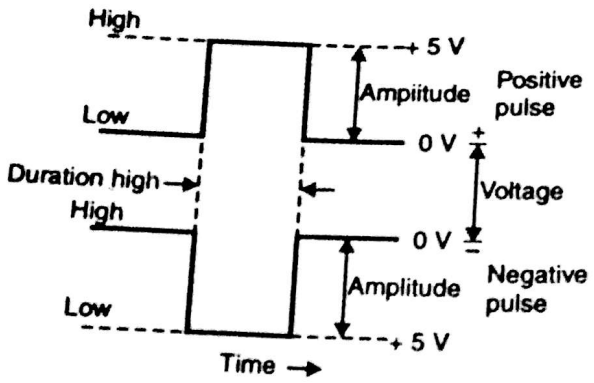
The way we have ***active low inputs***, we can also have ***active low output***. Presently for Fig. 3.76 (a), ), (c) output is active HIGH. To convert active HIGH to active low, simply put ***Inverter (NOT)*** gate at e output. Then you can label O/P Instead of Y, $\bar{Y}$ · $\bar{Y}$ represent ***active low output***.

**Positive Logic :** It is logic in which all the Boolean variables and Boolean functions behaves as scribed i.e. logic '0' corresponds to 0V and logic 1 corresponds to +5V.

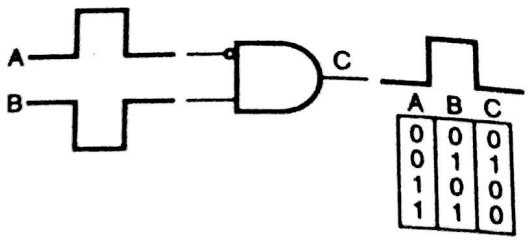
**Negative Logic :** In this normal meaning of binary is interchanged i.e. logic '0' corresponds to +5V d logic 1 corresponds to 0V. Thus all the Boolean variables and Boolean functions behave as if they are mplements.

| Inputs | | Positive logic | | | | | |
|---|---|---|---|---|---|---|---|
| | |  |  |  |  |  |  |
| A | B | AND | OR | NAND | NOR | EX-OR | EX-NOR |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| A | B | OR | AND | NOR | NAND | EX-NOR | EX-OR |
| Inputs | |  |  |  |  |  |  |

Scanned by CamScanner




(b)




(c)

**Fig. 3.77**

The Fig. 3.77 illustrated full Table showing negative and positive logic.

As shown positive logic AND is equivalent to negative logic OR, positive logic NOR is equivalent to Negative logic NAND and so on.

Normally negative logic is rarely used.

Scanned by CamScanner

## 3.16 Examples from University Papers :

**1. Prove the following using Boolean theorems :**

$$\left[(A + \bar{A}B)\,(A + \bar{A}\,\bar{B})\right]\ \left[(CD + \bar{C}\,\bar{D}) + (C \oplus D)\right] = A$$

**(May 96, 4 Marks)**

**Ans.:** L.H.S. $= [(A + \bar{A}B)\,(A + \bar{A}\bar{B})]\,[CD + \bar{C}\,\bar{D} + (C \oplus D)]$

Simplifying the above equations,

$$= [A \cdot A + A\bar{A}\bar{B} + \bar{A}BA + \bar{A}B \cdot \bar{A}\bar{B}]\,[(CD + \bar{C}\bar{D}) + (C \oplus D)]$$

Now according to boolean laws.

(i) $A \cdot A = A$ (ii) $A \cdot \bar{A} = 0$ (iii) $\bar{A} \cdot \bar{A} = \bar{A}$

and as per output equation of EX–NOR gate

$$CD + \bar{C}\bar{D} = \overline{C \oplus D}$$

$\therefore$ The above equation becomes,

$$= [A + 0 + 0 + \bar{A} \cdot 0]\left[\overline{C \oplus D} + (C \oplus D)\right]$$

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

If $C \oplus D = A$

then $(\overline{C \oplus D}) + (C \oplus D) = \bar{A} + A$

And according to Boolean law $A + \bar{A} = 1$

$$= [A]\,[\bar{A} + A]$$

$$= [A]\,[1]$$

$$= A$$

L.H.S. = [A] = R.H.S. → Hence Proved.

**...Ans.**

**2. Prove the following expression using Boolean theorems :**

**(Dec., 96, 4 Marks)**

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

**Ans.:** Start from R.H.S.

$$\therefore \quad (A + B)\,\overline{AB} = A \cdot \overline{AB} + B \cdot \overline{AB}$$

Now according to Demorgan theorem

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

$\therefore$ Using this in above equation

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

Scanned by CamScanner

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

Now using Boolean laws, $A\bar{A} = 0$ and $B\bar{B} = 0$

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

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

$\therefore$ R.H.S. $= A\bar{B} + \bar{A}B =$ L.H.S. $\rightarrow$ Hence Proved

...Ans.

**3. Prove the following expression using Boolean theorems :** **(May 97, 4 Marks)**

$$\mathbf{A(B + C) + \overline{\overline{AB} + \overline{BC}} + \bar{A}(B + C) = B(AC + 1) + C}$$

**Ans.:** Start from L.H.S.

$$A(B + C) + \left(\overline{\overline{AB} + \overline{BC}}\right) + \bar{A}(B + C) = AB + AC + \left(\overline{\overline{AB} + \overline{BC}}\right) + \bar{A}B + \bar{A}C$$

For simplifying the above equation use Demorgan theorem

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

$$= AB + AC + AB \cdot BC + \bar{A}B + \bar{A}C$$

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

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

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

$$= ABC + B + C \qquad (\because A + A = 1)$$

$$= B(AC + 1) + C$$

L.H.S. $= B(AC + 1) + C =$ R.H.S. $\rightarrow$ Hence proved

...Ans.

**4. Simplify the following expression using Boolean theorems :** **(Dec. 97, 4 Marks)**

$$\mathbf{(A + \bar{A}\,\bar{B})\,(A + \bar{A}B)\,(CD + \bar{C}\,\bar{D}\,A + C\bar{D} + \bar{C}DA)}$$

**Ans.:** Simplifying the equation,

$$(A \cdot A + A\bar{A}B + \bar{A}\bar{B} \cdot A + \bar{A}\bar{B} \cdot \bar{A}B)\,(CD + \bar{C}\bar{D}A + C\bar{D} + \bar{C}DA)$$

According to Boolean laws

$A \cdot A = A$, $A \cdot \bar{A} = 0$ and $B \cdot \bar{B} = 0$

$$= (A + 0 + 0 + 0)\,(CD + C\bar{D} + \bar{C}\bar{D}A + \bar{C}DA)$$

$$= (A)\,[C(D + \bar{D}) + A\bar{C}\,(D + \bar{D})]$$

As $D + \bar{D} = 1$

$$= A[C \cdot 1 + A\bar{C} \cdot 1]$$

Scanned by CamScanner

Now as $A \cdot 1 = A$

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

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

$= AC + A\bar{C} \quad (\because A \cdot A = A)$

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

$= A \quad (\because C + \bar{C} = 1)$

$\therefore \quad (A + \bar{A}\,\bar{B})\,(A + \bar{A}B)\,(CD + \bar{C}\,\bar{D}\,A + C\bar{D} + \bar{C}DA) = A$

...Ans.

**5. Let 'C' is the control input and 'A' is the signal input applied to a 2–input gate of any type. When gate is 'enabled', the uncomplemented or complemented form of a signal 'A' appears as the gate output when the output is not of this type, gate is called as 'disabled'. For example, a function of 'OR' gate may be explained by a function table.(May 98, 12 Marks)**

| Control input 'C' | Output of 'OR' gate |
|---|---|
| 0 | A |
| 1 | 1 |

**Write function tables derived from truth–table for the following gates :**

**(i) AND gate** **(ii) NAND gate**

**(iii) NOR gate** **(iv) EX–OR gate.**

**Ans.:** According to the given data 'C' is control input and 'A' is signal input applied to 2–input gate See Fig. Q.3.5(a)).

(i) The function table for the AND gate is as shown in Table 1.

The output equation for AND gate is $Y = A \cdot B$ (where Y is output and A, B are 2 inputs.)

| Control input 'C' | Output of 'AND' gate |
|---|---|
| 0 | 0 |
| 1 | A |

**Table 1**

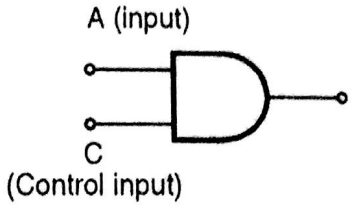



**Fig. Q.3.5(a)**

[Here Second input is A]

This gate is 'enabled' when the control input is high.

(ii) The function table for 'NAND' gate is as shown in Table 2.

The output equation for 'NAND' gate is

$Y = \overline{A \cdot B}$ (where 'Y' is output and A, B are two inputs.)

Scanned by CamScanner

| Control input 'C' | Output of 'NAND' gate |
|---|---|
| 0 | 1 |
| 1 | $\bar{A}$ |

Table 2

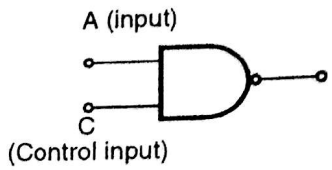



Fig. Q.3.5(b)

[Here Second input is A]

This gate is enabled when control input is high (1).

(iii) The function table for 'NOR' gate is as shown in Table 3.

The output equation for 'NOR' gate is

$Y = \overline{A + B}$ (where 'Y' is output and A, B are two inputs.)

| Control input 'C' | Output of 'NOR' gate |
|---|---|
| 0 | $\bar{A}$ |
| 1 | 0 |

Table 3

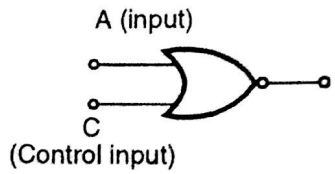



Fig. Q.3.5(c)

[Here Second input is A]

This gate is 'enabled' when control input is 'low'.

(iv) The function table for EX–OR gate is as shown in Table 4.

The output equation for EX–OR gate is,

$Y = A \oplus B = \bar{A}B + A\bar{B}$ (where 'Y' is output and A, B are inputs)

| Control input 'C' | Output of 'EX-OR' gate |
|---|---|
| 0 | A |
| 1 | $\bar{A}$ |

Table 4

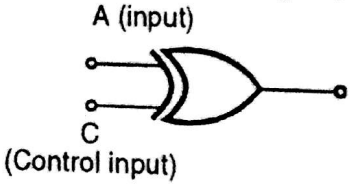



Fig. Q.3.5(d)

[Here A is Second input.]

Here gate is enabled when input is low and high.

**6. Using above function tables, prove the following :** **(May 98, 4 Marks)**

**(i) NAND gate is enabled keeping control input 'high' whereas NOR is enabled keeping it 'LOW '.**

**(ii) Ex–OR gate is always enabled.**

**Ans.:** (i) Using the function table for 'NAND' gate in (Q. 5) we can say, if control input of 'NAND' gate is '0' its output is always high and if control input of 'NAND' gate is 1, its output is complemented

Scanned by CamScanner

form of signal A. Therefore we can say that NAND gate is enabled keeping control input high (1). Similarly using the function table for 'NOR' gate in Q. 5 we can say,

If control input of 'NOR' gate is low (0), its output is complemented form of signal A and if control input of NOR gate is high (1) its output is always one. Therefore we can say that 'NOR' gate is enabled keeping control input low.

(ii) Using the function table for EX–OR gate in Q. 5 we can say,

If control input of EX–OR gate is zero its output is uncomplemented form of signal A and if control input of EX–OR gate is 1, its output is complemented form of signal A. Therefore we can say that EX–OR gate is always enabled.

**7. Simplify the following expressions using boolean theorems :** **(May 98, 6 Marks)**

**(i)** $\left[(A + \bar{A}B)\,(A + \overline{\bar{A}B})\right]\left[(CD + \bar{C}\bar{D}) + (C \oplus D)\right]$

**(ii)** $\bar{X}\,\bar{Y}\,\bar{Z} + \bar{X}Y\bar{Z} + X\overline{Y\bar{Z}} + XY\bar{Z}$

**(iii)** $XY + \bar{X}\bar{Y}Z + \overline{XY}\,Z$

**Ans.:** (i) Refer the answer of Q. 1.

(ii) $\bar{X}\,\bar{Y}\,\bar{Z} + \bar{X}\,Y\bar{Z} + X\,\overline{YZ} + XY\bar{Z}$

$$= \bar{X}\,\bar{Z}(Y + \bar{Y}) + X\,\overline{YZ} + XY\bar{Z}$$

According to boolean law $Y + \bar{Y} = 1$

$$= \bar{X}\,\bar{Z} \cdot 1 + X\,\overline{YZ} + XY\bar{Z}$$

According to Demorgan theorem

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

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

$$= \bar{X}\bar{Z} + X\bar{Y} + X\bar{Z} + XY\bar{Z}$$

$$= \bar{X}\,\bar{Z} + X\bar{Y} + X\bar{Z}(1 + Y)$$

As $(1 + A = 1)$, $= \bar{X}\,\bar{Z} + X\bar{Y} + X\bar{Z}$

$$= \bar{X}\,\bar{Z} + X\bar{Z} + X\bar{Y}$$

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

As $(A + \bar{A} = 1)$ $= \bar{Z} \cdot 1 + X\bar{Y}$

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

$$\therefore \quad \bar{X}\,\bar{Y}\,\bar{Z} + \bar{X}\,Y\,\bar{Z} + X\,\overline{YZ} + XY\bar{Z} = X\bar{Y} + \bar{Z}$$

**...Ans.**

Scanned by CamScanner

(iii) $XY + \bar{X}\,\bar{Y}Z + \overline{XY}\,Z$

According to Demorgan theorem

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

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

$$= XY + \bar{X}\,\bar{Y}Z + \bar{X}Z + \bar{Y}Z$$

$$= XY + \bar{X}Z\,(\bar{Y} + 1) + \bar{Y}Z$$

$$= XY + \bar{X}Z \cdot 1 + \bar{Y}Z \qquad (\because \bar{A} + 1 = 1)$$

$$= XY + \bar{X}Z + \bar{Y}Z$$

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

$$\therefore XY + \bar{X}\,\bar{Y}Z + \overline{XY}\,Z = XY + Z(\bar{X} + \bar{Y})$$

...Ans.

**8. Implement the following using 2 input NOR gates :** **(May 98, 4 Marks)**

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

**(ii) $Z = A \oplus B$**

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

We have to implement the above output using NOR gates only.

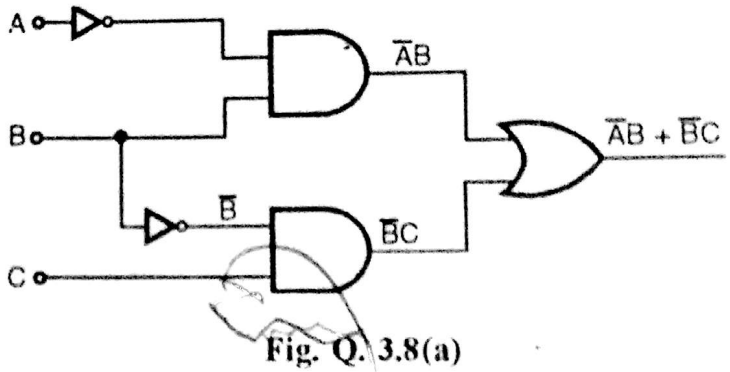



Fig. Q. 3.8(a)

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

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

$$= \left[\overline{\overline{A + \bar{B}}}\right] + \left[\overline{\overline{B + \bar{C}}}\right]$$

$\therefore$ Circuit will be (See Fig. Q.3.8(b).

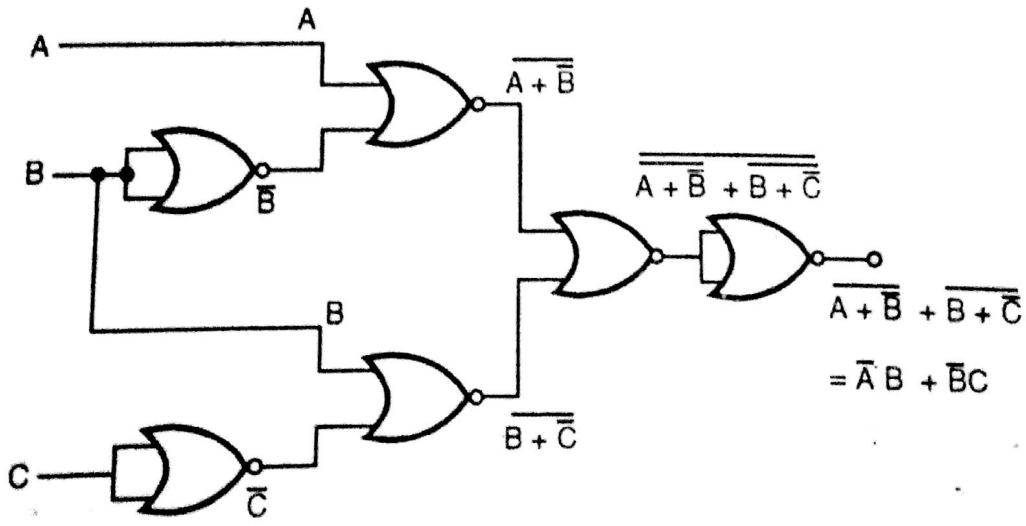



Fig. Q. 3.8 (b)

Scanned by CamScanner

$$Z = A \oplus B$$

n of EX–OR gate

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

$$Z = \overline{\bar{A}B} + \overline{A\bar{B}} \qquad (\because \bar{\bar{A}} = A)$$

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

ing the above using NOR gates,

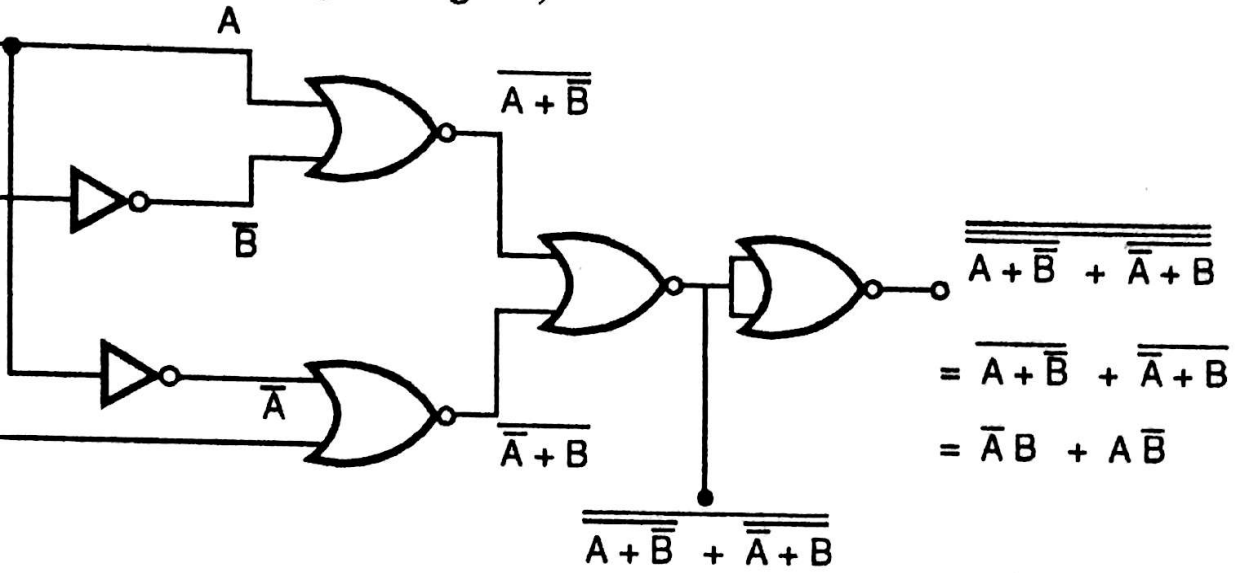
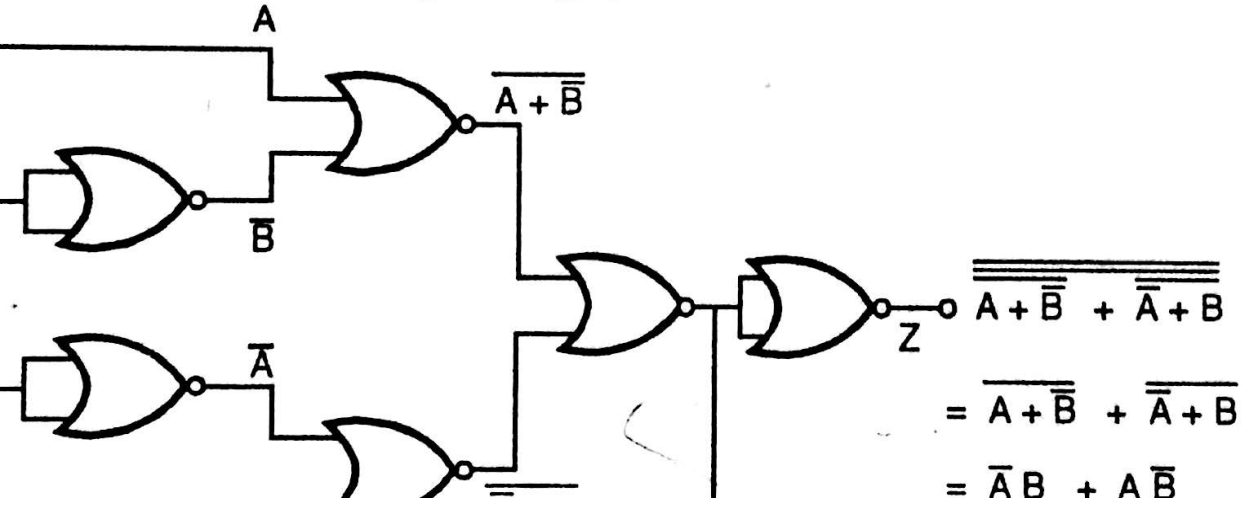



**Fig. Q.3.8(c)**

g INVERTER by NOR gate we get,




Scanned by CamScanner

$Y = \bar{A}B + A\bar{B}$ can be written as.

$Y = \overline{\bar{A}B + A\bar{B}}$ ; Simplify this, we get,

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

implemented using NAND gates. Output of NAND gate is Y

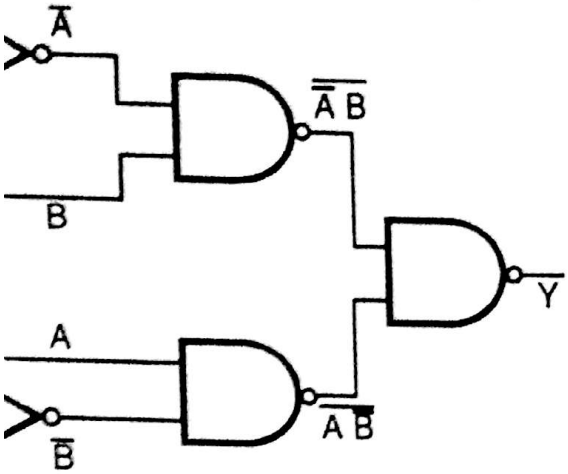



Fig. Q.3.10 (a)

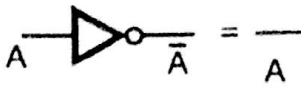



Fig. Q.3

can be represented as single input NAND gate as shown in Fig.

). 3.10(a) can be represented as,

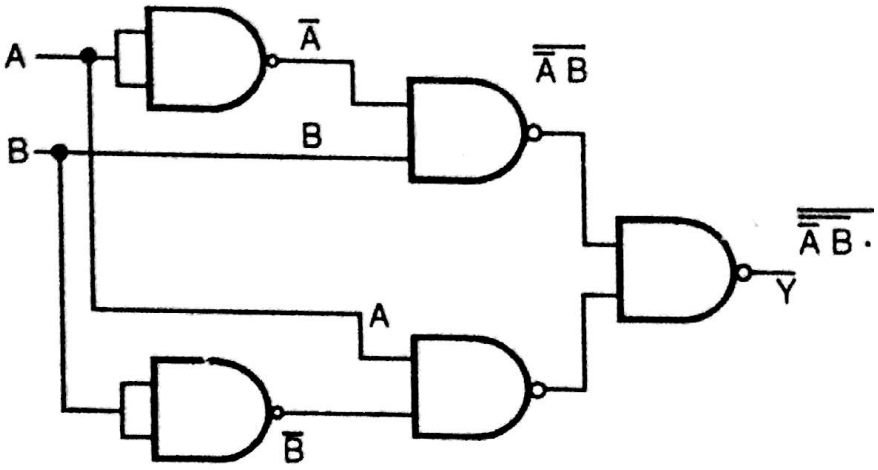



Fig. Q. 3.10 (b)

table of a comparator and an EX–OR gate are as follows. A anc

al.

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| A |
|---|
| 0 |
| 0 |
| 1 |
| 1 |

Scanned by CamScanner

**11. Prove the following expression using Boolean theorems :** **(June 99, 4 Marks)**

**$(B + A)(B + D)(A + C)(C + D) = BC + AD$**

**Ans.:** $(B + A)(B + D)(A + C)(C + D) = BC + AD$

L.H.S. $= (B \cdot B + B \cdot D + A \cdot B + A \cdot D)(A \cdot C + A \cdot D + C \cdot C + C \cdot D)$

According to boolean law $A \cdot A = A$

$= (B + BD + AB + AD)(AC + AD + C + CD)$

$= [B(1 + D + A) + AD][C(1 + D + A) + AD]$

$= [B \cdot 1 + AD][C \cdot 1 + AD]$

$= BC + BAD + CAD + AD$

$= BC + AD[B + C + 1]$

$= BC + AD$

$\therefore$ L.H.S. $= BC + AD =$ R.H.S. $\rightarrow$ Hence proved.

**...Ans.**

**12. Prove the following using Boolean Theorems :**

**(i)** $[(C + \bar{C}D)(C + \bar{C}\bar{D})][(AB + \bar{A}\bar{B} + A \oplus B)] = C$ **(Dec. 99, 3 Marks)**

**(ii)** $\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC = AB + AC + BC$ **(Dec. 99, 3 Marks)**

**Ans.:** (i) $[(C + \bar{C}D)(C + \bar{C}\bar{D})][(AB + \bar{A}\bar{B} + A \oplus B)] = C$

L.H.S. $= [C \cdot C + C \cdot \bar{C}\bar{D} + \bar{C}D \cdot C + \bar{C}D \cdot \bar{C}\bar{D}][AB + \bar{A}\bar{B} + A\bar{B} + \bar{A}B)$

As per boolean law,

$A \cdot A = A$ and $A \cdot \bar{A} = 0$

$= [C + 0 + 0 + 0][AB + \bar{B}(A + \bar{A}) + \bar{A}B]$

$= C[AB + \bar{B}(1) + \bar{A}B]$ $(\because A + \bar{A} = 1)$

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

$= C[B + \bar{B}]$

$= C[1]$

$= C$

$\therefore$ L.H.S. $= C =$ R.H.S. $\rightarrow$ Hence proved.

**...Ans.**

Scanned by CamScanner

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

According to boolean laws,

$A + A = A \qquad \therefore \quad A + A + A = A$

Therefore adding two terms of ABC to L.H.S of the above equation

$\therefore$ L.H.S of the above equation becomes

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

Rearranging the terms, $= \bar{A}BC + ABC + A\bar{B}C + ABC + AB\bar{C} + ABC$

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

As $A + \bar{A} = 1$ $= BC + AC + AB$

$= AB + AC + BC$

$\therefore$ L.H.S. = AB + AC + BC = R.H.S. $\rightarrow$ Hence proved. ...Ans.

❑ ❑ ❑

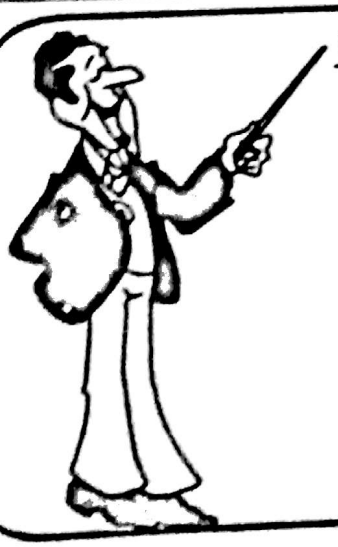
Scanned by CamScanner

# Minterm, Maxterm and Karnaugh Map 4

**Syllabus :**

Switching equations, canonical logic forms, sum of product and product of sums, Karnaugh maps ; Two, three and four variable Karnaugh maps. Simplification of expressions, Quine–Mc Cluskey minimization techniques.

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Minterms and sum of minterm form | 4.2 | – | ✓ |
| • Maxterm and product of maxterm form | 4.3 | – | ✓ |
| • Algebric construction of maxterm using boolean expressions | 4.4 | – | ✓ |
| • Reduction technique using K–map | 4.5 | – | ✓ |
| • Grouping of variables in K–map | 4.6 | ✓ | ✓ |
| • K-map for POS form | 4.7 | ✓ | ✓ |
| • Minimising boolean expression using K–map | 4.8 | – | ✓ |
| • Quine MC-clusky method or tabular method | 4.9 | – | ✓ |



**Comments**

- Knowledge of K–map is required, throughout the subject.
- K–map is important from all the angle.
- Concentrate on 4.5/6/7/8.

Scanned by CamScanner

## 4.1 Introduction :

We have studied logic gates and boolean algebra. Using boolean theorem we have manipulated logical expression(s). The aim behind manipulation was, to reduce logical expression considerably. To realize logical expression, we use logic gates ; so by reducing logical expression, we are reducing number of gates required. This indirectly will result in the reduction in cost and increase in reliability of the circuit.

In this chapter, we will concentrate on one more technique of reducing logical expression i.e. Karnaugh map (K-map). But before that let's introduce yourself to maxterms and minterms.

## 4.2 Minterms and Sum of Miniterm Form :

O/P = 1

Uptill now we have solved many problems. In that one Boolean expression was given to you and you were supposed to minimise it. But here, you may ask, from where do we get actual Boolean expression which we are suppose to minimise ?

The answer of the question is very simple. In actual practice you will decide some variable A, B, C, D etc., depending upon number of inputs from different sensors. After getting particular ***combination*** of input you want that O/P terminal should give you logic 1. But for invalid combination of inputs, you require O/P to be low (logic 0). Thus again you are going to formulate truth Table. Here for understanding I will take example of 'OR' gate. In OR gate, the truth table is,

| | Decimal | Inputs | | Output | |
|---|---|---|---|---|---|
| | | A | B | Y | |
| $m_0$ | 0 | 0 | 0 | 0 | $\rightarrow \bar{A}\bar{B} \rightarrow \bar{0}\cdot\bar{0} = 1\cdot 1 = 1$ |
| $m_1$ | 1 | 0 | 1 | 1 | $\rightarrow \bar{A}B \rightarrow \bar{0}\cdot 1 = 1\cdot 1 = 1$ |
| $m_2$ | 2 | 1 | 0 | 1 | $\rightarrow A\bar{B} \rightarrow 1\cdot\bar{0} = 1\cdot 1 = 1$ |
| $m_3$ | 3 | 1 | 1 | 1 | $\rightarrow AB \rightarrow 1\cdot 1 = 1\cdot 1 = 1$ |

Variables should be adjusted in such a way that you should get logical 1.

**Table 4.1**

So you require O/P to be high, when A = 0, B = 1 ; A = 1 B = 0 and A = B = 1. So whenever you require output of circuit to be '1' that particular term is selected. As shown in Table 4.1, on the left side, we have written $m_0$, $m_1$, $m_2$ and $m_3$; where m stands for ***minterm***. In general we have $m_i$ where i varies from '0' to some finite value. The function $m_i$ is called the $i^{th}$ ***minterm function***, or simply the $i^{th}$ *minterm*.

So from Table 4.1 we conclude that our output is '1' for $m_1$, $m_2$, $m_3$. ∴ Function f(x) can be written in the following ***sum of minterms*** form.

$f(x) = m_1 + m_2 + m_3$ (where '+' denotes OR sign). Here we are suppose to select minterms those generates O/P logic 1 only.

$$f(A, B) = m_1 + m_2 + m_3 \quad \ldots(1)$$

∴ O/P Y is also function of $m_1$, $m_2$, $m_3$ which is written as $Y = \bar{A}B + A\bar{B} + AB$ ...(2)

We have selected only those terms which has O/P Y = 1. Now let's see

**Case I** : A = 0, B = 1.

$$\therefore Y = \bar{A}B + A\bar{B} + AB = \bar{0}\cdot 1 + 0\cdot\bar{1} + 0\cdot 1 = 1\cdot 1 + 0\cdot 0 + 0$$
$$= 1 + 0 + 0 = 1$$

**Case II** : A = 1 B = 0

$$Y = \bar{A}B + A\bar{B} + AB = \bar{1}\cdot 0 + 1\cdot\bar{0} + 1\cdot 0 = 0\cdot 0 + 1\cdot 1 + 0 = 0 + 1 + 0 = 1$$

Scanned by CamScanner

**Case III** : $A = 1, B = 1$

$Y = \bar{A}B + A\bar{B} + AB = \bar{1} \cdot 1 + 1 \cdot \bar{1} + 1 \cdot 1 = 0 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 = 0 + 0 + 1 = 1$

**Case IV** : $A = 0, B = 0$

$Y = \bar{A}B + A\bar{B} + AB$ (This form is sum of product (SOP) form)

$= \bar{0} \cdot 0 + 0 \cdot \bar{0} + 0 \cdot 0 = 1 \cdot 0 + 0 \cdot 1 + 0 = 0 + 0 + 0 = 0.$

We get here $Y = 0$ because we haven't selected $\bar{A}\bar{B}$ term, because we ***don't require it***.

Now, but at this stage, you will remind me that equation for OR is $Y = A + B$, but we have, $Y = \bar{A}B + A\bar{B} + AB$. So now let's reduce it (minimise the equation).

$$Y = \underline{\bar{A}B} + A\bar{B} + \underline{AB} = B(\bar{A} + A) + A\bar{B} = B \cdot 1 + A\bar{B}$$

$$= B + \bar{B}A = B + A = A + B \quad \therefore \boxed{Y = A + B} \rightarrow \text{equation for OR (reduced).}$$

The equation $f(x) = m_1 + m_2 + m_3$ (for OR gate)

or generalised $f(x) = m_a(x) + m_b(x) + \ldots + m_k(x)$ is nothing but ***SUM OF MINTERM*** form. This can be also written as,

$f(A, B) = \Sigma m\ (1, 2, 3)$.

$\Sigma$ = Summation (logical ORing).

m = minterm.

1, 2, 3 = nqthing but numerical value showing $m_1$, $m_2$ and $m_3$.

f (A, B) = function of variable A and B.

f (A, B) = $\Sigma$m (1, 2, 3) is also called as ***Numeric Form***. Let's take one more example to understand the concept.

---

**Ex. 1 :** f (A, B, C) = $\Sigma$m (0, 1, 4, 7).

**Soln. :**

(a) f (A, B, C) → means we have three variables.

$\therefore$ Number of decimal state $= 2^n$ n = Number of bits or variables. $\therefore 2^n = 2^3 = 8$.

(b) Here $\Sigma$m (0, 1, 4, 7) means

$f(x) = m_0 + m_1 + m_4 + m_7$ i.e. O/P is HIGH only at ABC = $(0\ 0\ 0)_2$, $(0\ 0\ 1)_2$, $(1\ 0\ 0)_2$, $(1\ 1\ 1)_2$.

For remaining all minterms i.e. 2, 3, 5, 6 ($m_2$, $m_3$, $m_5$, $m_6$) O/P is Low.

Let's make truth table.

| | Decimal | Inputs | | | Output | |
|---|---|---|---|---|---|---|
| | | A | B | C | Y | |
| $m_0$ | 0 | 0 | 0 | 0 | 1 | $\rightarrow \bar{A}\bar{B}\bar{C}$ |
| $m_1$ | 1 | 0 | 0 | 1 | 1 | $\rightarrow \bar{A}\bar{B}C$ |
| $m_2$ | 2 | 0 | 1 | 0 | 0 | |
| $m_3$ | 3 | 0 | 1 | 1 | 0 | |
| $m_4$ | 4 | 1 | 0 | 0 | 1 | $\rightarrow A\bar{B}\bar{C}$ |
| $m_5$ | 5 | 1 | 0 | 1 | 0 | |
| $m_6$ | 6 | 1 | 1 | 0 | 0 | |
| $m_7$ | 7 | 1 | 1 | 1 | 1 | $\rightarrow ABC$ |

**Table 4.2**

Scanned by CamScanner

$m_0 = (0\ 0\ 0)_2 = \bar{A}\,\bar{B}\,\bar{C}$ $\therefore$ We can write minterm $\bar{A}\bar{B}\bar{C} = \bar{0} \cdot \bar{0} \cdot \bar{0} = 1 \cdot 1 \cdot 1 = 1$

$m_1 = (0\ 0\ 1)_2 = \bar{A}\,\bar{B}\,C$ $\therefore$ We can write $\bar{A}\,\bar{B}\,C = \bar{0} \cdot \bar{0} \cdot 1 = 1 \cdot 1 \cdot 1 = 1$

$m_4 = (1\ 0\ 0)_2 = A\,\bar{B}\,\bar{C}$ $\therefore$ We can write $A\,\bar{B}\,\bar{C} = 1 \cdot \bar{0} \cdot \bar{0} = 1 \cdot 1 \cdot 1 = 1$

$m_7 = (1\ 1\ 1)_2 = A\,B\,C$ $\therefore$ We can write $A\,B\,C = 1 \cdot 1 \cdot 1 = 1$

$\therefore\ Y = m_0 + m_1 + m_4 + m_7 = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC$. Now minimise it

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

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

**Ex. 2 :** Convert following numerical form into sum–of–minterms OR sum of product form :

(i) $F(A, B, C, D) = \Sigma m\ (1, 2, 3, 6, 7, 15)$.

**Soln. :** (a) There are four variables A, B, C, D. $\therefore$ No. of states are $= 2^n = 2^4 = 16 = 0$ to 15.

(b) $\Sigma m\ (1, 2, 3, 6, 7, 15)$ means Ouput Y is high only for $m_1$, $m_2$, $m_3$, $m_6$, $m_7$ and $m_{15}$. For remaining states Ouput = 0. Let's write down truth table.

(c) Truth Table is,

| | Decimal | Inputs | | | | Output | |
|---|---|---|---|---|---|---|---|
| | | A | B | C | D | Y | |
| $m_0$ | 0 | 0 | 0 | 0 | 0 | 0 | |
| $m_1$ | 1 | 0 | 0 | 0 | 1 | 1 | $\rightarrow \bar{A}\,\bar{B}\,\bar{C}D$ |
| $m_2$ | 2 | 0 | 0 | 1 | 0 | 1 | $\rightarrow \bar{A}\bar{B}C\bar{D}$ |
| $m_3$ | 3 | 0 | 0 | 1 | 1 | 1 | $\rightarrow \bar{A}\bar{B}CD$ |
| $m_4$ | 4 | 0 | 1 | 0 | 0 | 0 | |
| $m_5$ | 5 | 0 | 1 | 0 | 1 | 0 | |
| $m_6$ | 6 | 0 | 1 | 1 | 0 | 1 | $\rightarrow \bar{A}BC\bar{D}$ |
| $m_7$ | 7 | 0 | 1 | 1 | 1 | 1 | $\rightarrow \bar{A}BCD$ |
| $m_8$ | 8 | 1 | 0 | 0 | 0 | 0 | |
| $m_9$ | 9 | 1 | 0 | 0 | 1 | 0 | |
| $m_{10}$ | 10 | 1 | 0 | 1 | 0 | 0 | |
| $m_{11}$ | 11 | 1 | 0 | 1 | 1 | 0 | |
| $m_{12}$ | 12 | 1 | 1 | 0 | 0 | 0 | |
| $m_{13}$ | 13 | 1 | 1 | 0 | 1 | 0 | |
| $m_{14}$ | 14 | 1 | 1 | 1 | 0 | 0 | |
| $m_{15}$ | 15 | 1 | 1 | 1 | 1 | 1 | $\rightarrow ABCD$ |

**Table 4.3**

Scanned by CamScanner

Let's solve some examples to get acquainted with procedure.

**Ex. 2 (A) :** Represent minterms for following expressions. OR represent the following expressions in numerical form :

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

**Soln. :** Here F (A, B, C, D) directly informs you that you have to add two more variables i.e. B and C. There are two methods to know the minterm.

**Method I :** This method we have already seen.

$\therefore$ Write $Y = \bar{A}D$

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

$$= \left(\bar{A}B + \bar{A}\bar{B}\right) \cdot \left(DC + D\bar{C}\right)$$

$$= \bar{A}BCD + \bar{A}B\bar{C}D + \bar{A}\bar{B}CD + \bar{A}\bar{B}\bar{C}D$$

$$= \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BCD \quad \ldots(4)$$

$$\Downarrow \qquad \Downarrow \qquad \Downarrow \qquad \Downarrow$$

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

$$\downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow$$

$$1\,(m_1) \qquad 3\,(m_3) \qquad 5\,(m_5) \qquad 7(m_7)$$

$\therefore$ $F(A, B, C, D) = \sum m\,(1, 3, 5, 7)$.

**Method II :** In this ***unknown variables*** are replaced by $x_i$ and known variables are replaced by 0 or 1 logic depending upon given expression. In our case we know that we should have four variables A, B, C, D.

But expression is $\bar{A}D$, $\therefore$ B and C both are missing.

| MSB | $x_1$ | $x_0$ | LSB |
|---|---|---|---|
| A | B | C | D |
| 0 | | | 1 |

A should be put '0' because it is $\bar{A}$.

D should be put '1' because it is D.

Now as 2 variables are unknown we can have $2^2$ i.e. 4 combinations. $\therefore$ Final Table is,

| MSB | $x_1$ | $x_0$ | LSB | | Equivalent |
|---|---|---|---|---|---|
| A | B | C | D | | Decimal |
| 0 | 0 | 0 | 1 | $\longrightarrow$ | 1 |
| 0 | 0 | 1 | 1 | $\longrightarrow$ | 3 |
| 0 | 1 | 0 | 1 | $\longrightarrow$ | 5 |
| 0 | 1 | 1 | 1 | $\longrightarrow$ | 7 |

$\therefore$ **Finally** $F(A, B, C, D) = \sum m\,(1, 3, 5, 7) \longrightarrow$ numerical form.

**Note :** In this case there was only one expression, $\therefore$ we have prepare only one Table. But if we have more expression, we have to make that many number of Tables.

**Ex. 2(B) :** Represent the following expressions in numerical form :

(i) $F(A, B, C, D) = \bar{A}BC + BD$.

Scanned by CamScanner

(1) The first term is $\bar{A}\,B\,C = \bar{0} \cdot 1 \cdot 1$

$\therefore$ A = 0 B = 1 C = 1 (D is missing)

| MSB | | | LSB ($x_0$) | |
|---|---|---|---|---|
| A | B | C | D | |
| 0 | 1 | 1 | 0 | $\longrightarrow$ $(6)_{10}$ |
| 0 | 1 | 1 | 1 | $\longrightarrow$ $(7)_{10}$ |

(2) From second term BD = 1 · 1.

$\therefore$ B = 1 D = 1, A and C variables are missing $\therefore$ Table is,

| MSB $x_1$ | | $x_0$ | LSB | |
|---|---|---|---|---|
| A | B | C | D | |
| 0 | 1 | 0 | 1 | $\longrightarrow$ $(5)_{10}$ |
| 0 | 1 | 1 | 1 | $\longrightarrow$ $(7)_{10}$ |
| 1 | 1 | 0 | 1 | $\longrightarrow$ $(13)_{10}$ |
| 1 | 1 | 1 | 1 | $\longrightarrow$ $(15)_{10}$ |

$\therefore$ Total minterms will be considering both the terms $F(A, B, C, D) = \Sigma m\,(5, 6, 7, 13, 15)$

---

(ii) $F(A, B, C, D) = B\bar{C} + D$. $n$ = unknown variable $= 2^n$

**Soln. :** Let's take term one by one.

(1) $B\bar{C} = 1 \cdot \bar{0}$ $\therefore$ B = 1 C = 0.

| MSB $x_1$ | | | LSB $x_0$ | |
|---|---|---|---|---|
| A | B | C | D | |
| 0 | 1 | 0 | 0 | $\longrightarrow$ $(4)_{10}$ |
| 0 | 1 | 0 | 1 | $\longrightarrow$ $(5)_{10}$ |
| 1 | 1 | 0 | 0 | $\longrightarrow$ $(12)_{10}$ |
| 1 | 1 | 0 | 1 | $\longrightarrow$ $(13)_{10}$ |

(2) D $\therefore$ D = 1

| MSB $x_2$ | $x_1$ | $x_0$ | LSB | |
|---|---|---|---|---|
| A | B | C | D | |
| 0 | 0 | 0 | 1 | $\longrightarrow$ $(1)_{10}$ |
| 0 | 0 | 1 | 1 | $\longrightarrow$ $(3)_{10}$ |
| 0 | 1 | 0 | 1 | $\longrightarrow$ $(5)_{10}$ |
| 0 | 1 | 1 | 1 | $\longrightarrow$ $(7)_{10}$ |
| 1 | 0 | 0 | 1 | $\longrightarrow$ $(9)_{10}$ |
| 1 | 0 | 1 | 1 | $\longrightarrow$ $(11)_{10}$ |
| 1 | 1 | 0 | 1 | $\longrightarrow$ $(13)_{10}$ |
| 1 | 1 | 1 | 1 | $\longrightarrow$ $(15)_{10}$ |

$\therefore$ $F(A, B, C, D) = \Sigma m\,(1, 3, 4, 5, 7, 9, 11, 12, 13, 15)$

Scanned by CamScanner

**(iii)** $F(A, B) = A + \bar{B}$

**Soln. :** (a) Term A = 1

| A | B |
|---|---|
| 1 | 0 |
| 1 | 1 |

$\longrightarrow (2)_{10}$

$\longrightarrow (3)_{10}$

(b) Term $\bar{B} = \bar{0}$ $\therefore$ B = 0.

| A | B |
|---|---|
| 0 | 0 |
| 1 | 0 |

$\longrightarrow (0)_{10}$

$\longrightarrow (2)_{10}$

$\therefore$ $F(A, B) = \Sigma m\,(0, 2, 3)$

**Note :** An all minterm equation such as Equation (3) and Equation (4) is a standard way of representing any logical function. It is called the CANONICAL SUM OF MINTERMS FORM OR CANONICAL DISJUNCTIVE FORM (disjunction is another term for OR). This type of Boolean expression is simply an OR Sum of all the minterms, with each minterm expressed as an AND product term containing a literal for every primary input variable.

## 4.3 Maxterm and Product of Maxterm Form :

By forming the duals of ***Minterm form*** we obtain ***Maxterm function***. As we have seen in minterm we consider output logical '1'. But in maxterm we consider output logical '0'. We will take same example i.e. of OR gate. The truth Table is,

| | Decimal | Inputs | | Output | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | A | B | Y | | | | | |
| $M_0$ | 0 | 0 | 0 | 0 | $\longrightarrow$ | $A + B$ | = | $0 + 0$ | = 0 |
| $M_1$ | 1 | 0 | 1 | 1 | $\longrightarrow$ | $A + \bar{B}$ | = | $0 + \bar{1}$ | = 0 |
| $M_2$ | 2 | 1 | 0 | 1 | $\longrightarrow$ | $\bar{A} + B$ | = | $\bar{1} + 0$ | = 0 |
| $M_3$ | 3 | 1 | 1 | 1 | $\longrightarrow$ | $\bar{A} + \bar{B}$ | = | $\bar{1} + \bar{1}$ | = 0 + 0 = 0 |

**Table 4.4**

If you observe, we have written terms in such a way that final answer you should get is '0' i.e. when A = 0, B = 0 we have directly taken A + B. If you compare it with minterm you will find that, in that we used to write 'product' term i.e. $\bar{A} \cdot \bar{B}$ and in such a way that we get answer '1'.

But in maxterm we should write down terms in 'sum' form and finally we are going to put it ***Product of SUM (POS)*** form. Secondly for maxterm we use capital letter 'M'.

We are going to select only those term which will provide output '0' logic. $\therefore$ only $M_0$ maxterm provides '0' output.

$\therefore$ f(x) function can be written in the following ***Product of maxterm*** (product of sum) form as follows :

$$f(x) = M_0$$

$$f(A, B) = M_0$$

OR $$f(A, B) = \pi M(0)$$

**Note :** In minterm we have used '$\Sigma$' sign. For maxterm we use '$\pi$' sign. M $\longrightarrow$ maxterm.

$\therefore$ output $\boxed{Y = A + B}$ $\longrightarrow$ equation for OR gate.

Take one more example of AND gate. Truth table for AND gate is as follows :

Scanned by CamScanner

| | Decimal | Inputs | | Output | |
|---|---|---|---|---|---|
| | | A | B | Y | |
| $M_0$ | 0 | 0 | 0 | 0 | $\longrightarrow A + B$ |
| $M_1$ | 1 | 0 | 1 | 0 | $\longrightarrow A + \bar{B}$ |
| $M_2$ | 2 | 1 | 0 | 0 | $\longrightarrow \bar{A} + B$ |
| $M_3$ | 3 | 1 | 1 | 1 | $\longrightarrow \bar{A} + \bar{B}$ |

**Table 4.5**

∴ For AND gate output Y is function f(x)

$f(x) = M_0 \cdot M_1 \cdot M_2$ (Product of maxterm form)

$f(A, B) = M_0 \cdot M_1 \cdot M_2$

$f(A, B) = \pi M (0, 1, 2)$ (Numeric form). ...(5)

∴ output $Y = (A + B) \cdot (A + \bar{B}) \cdot (\bar{A} + B)$ (Product of sum form (POS))

Let's simplify because as far as we know output Y for AND gate is A. B.

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

$$= (A + A\bar{B} + A \cdot B + 0)(\bar{A} + B) = [A(1 + \bar{B} + B)](\bar{A} + B)$$

$$= [A(1)](\bar{A} + B) = A(\bar{A} + B) = A \cdot \bar{A} + AB = 0 + AB \quad \therefore \boxed{Y = AB}$$

Let's solve some examples.

---

**Ex. 3 :** Simplify following :

(i) $F(A, B, C) = \pi M (0, 2, 4, 6)$

**Soln. :** (a) In the problem maxterms are given. Maxterm gives you POS form.

(b) Number of variables are three → A, B, C. ∴ $2^3 = 8$ states are there. You have to make truth table. Showing logic '0' output for maxterm 0, 2, 4, 6 and logic 1 for remaining.

(c) Truth Table is,

| | Decimal | Inputs | | | Output | |
|---|---|---|---|---|---|---|
| | | A | B | C | Y | |
| $M_0$ | 0 | 0 | 0 | 0 | 0 | $\longrightarrow A + B + C$ |
| $M_1$ | 1 | 0 | 0 | 1 | 1 | $\longrightarrow A + B + \bar{C}$ |
| $M_2$ | 2 | 0 | 1 | 0 | 0 | $\longrightarrow A + \bar{B} + C$ |
| $M_3$ | 3 | 0 | 1 | 1 | 1 | $\longrightarrow A + \bar{B} + \bar{C}$ |
| $M_4$ | 4 | 1 | 0 | 0 | 0 | $\longrightarrow \bar{A} + B + C$ |
| $M_5$ | 5 | 1 | 0 | 1 | 1 | $\longrightarrow \bar{A} + B + \bar{C}$ |
| $M_6$ | 6 | 1 | 1 | 0 | 0 | $\longrightarrow \bar{A} + \bar{B} + C$ |
| $M_7$ | 7 | 1 | 1 | 1 | 1 | $\longrightarrow \bar{A} + \bar{B} + \bar{C}$ |

**Table 4.6**

Scanned by CamScanner

Function $f(x) = \pi M (0, 2, 4, 6)$

$f(A, B, C) = M_0 M_2 M_4 M_6$ (Product of maxterm)

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

$$= [A \cdot A + A\bar{B} + AC + BA + B\bar{B} + BC + CA + C\bar{B} + C \cdot C] (\bar{A} + B + C) (\bar{A} + \bar{B} + C)$$

$$= [A + A\bar{B} + AC + AB + 0 + BC + AC + \bar{B}C + C] (\bar{A} + B + C) (\bar{A} + \bar{B} + C)$$

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

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

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

$$= (A + C) (\bar{A} + \bar{A}\bar{B} + \bar{A}C + \bar{A}B + 0 + BC + \bar{A}C + \bar{B}C + C)$$

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

$$= (A + C) [\bar{A} \cdot 1 + C \cdot 1] = (A + C) (\bar{A} + C)$$

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

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

$$= C (1 + A + \bar{A}) = C \cdot 1 = C.$$

We can also use short cut method to solve the example. Here one need not write down full Table. You directly take those term, specified in the expression. For our case the terms are (0, 2, 4, 6), and write down maxterm, the Table is shown below.

$F (A, B, C) = \pi M (0, 2, 4, 6)$

| | | A | B | C | | | | |
|---|---|---|---|---|---|---|---|---|
| ∴ | 0 = | 0 | 0 | 0 | → | $A + B + C$ | $0 + 0 + 0$ | = 0 |
| | 2 = | 0 | 1 | 0 | → | $A + \bar{B} + C$ | $0 + \bar{1} + 0$ | = 0 |
| | 4 = | 1 | 0 | 0 | → | $\bar{A} + B + C$ | $\bar{1} + 0 + 0$ | = 0 |
| | 6 = | 1 | 1 | 0 | → | $\bar{A} + \bar{B} + C$ | $\bar{1} + \bar{1} + 0$ | = 0 |

## 4.4 Algebraic Construction of Maxterm using Boolean Expressions :

We can algebraically derive product of maxterm expression for a function using the properties of Boolean algebra. For example let's say we have equation for AND gate.

$$Y = A \cdot B$$

We can introduce any variable which is missing into term AB by ORing the factor $x.\bar{x}$ (where x = any variable) ($x \cdot \bar{x} = 0$, $AB + x\bar{x} = AB + 0 = AB$, means we are not playing with actual equation.)

After adding the term $x\bar{x}$ use law of distribution and simplify unless and untill you get maxterm. Say for an example Y = AB can be also written as,

$$Y = AB + 0 = AB + B\bar{B}$$

$$= (AB + B) (AB + \bar{B}) \longrightarrow \text{Distributive Law}$$

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

Scanned by CamScanner

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

$= [A + AB + B]\left[AB + B + \bar{A}\right]\left[\bar{B} + AB\right]$

$= [A(1 + B) + B]\left[B(A + 1) + \bar{A}\right]\left[\bar{B} + AB\right]$

$= [A + B]\left[B + \bar{A}\right]\left[\bar{B} + AB\right]$

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

$\therefore$ Finally we have $Y = (A + B) \cdot (\bar{A} + B) \cdot (A + \bar{B})$ ...(6)

$\Downarrow \quad \Downarrow \quad \Downarrow$

$0 + 0 \quad \bar{1} + 0 \quad 0 + \bar{1}$

$AB = (00)_2 \quad AB = (10)_2 \quad AB = (01)_2$

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

$\therefore \quad Y = M_0 \cdot M_2 \cdot M_1$

$= M_0 \cdot M_1 \cdot M_2$ , where M = Maxterm.

Compare Equation (6) with Equation (5). Both are same. Thus we have to break down equation so that we get maxterms ***ONLY.***

Let's solve some examples to clear the idea.

---

**Ex. 4 :** Express following functions as product of maxterm :

(i) $F(W, X, Y, Z) = \bar{Y}Z + WX\bar{Y} + WX\bar{Z} + \bar{W}\bar{X}Z$

**Soln. :** $F = \bar{Y}Z + WX\bar{Y} + WX\bar{Z} + \bar{W}\bar{X}Z$

Double Complement full equation.

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

$$= \overline{\bar{Y}Z \cdot WX\bar{Y} \cdot WX\bar{Z} \cdot \bar{W}\bar{X}Z}$$

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

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

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

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

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

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

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

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

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

Scanned by CamScanner

Remove common term.

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

| | | | | |
|---|---|---|---|---|
| $W+X+Y+\bar{Z}$ | $\Rightarrow$ | $0+0+0+\bar{1}$ | $\Rightarrow$ $(0\,0\,0\,1)_2$ | $\Rightarrow$ $(1)_{10}$ |
| $\bar{W}+X+Y+\bar{Z}$ | $\Rightarrow$ | $\bar{1}+0+0+\bar{1}$ | $\Rightarrow$ $(1\,0\,0\,1)_2$ | $\Rightarrow$ $(9)_{10}$ |
| $W+\bar{X}+Y+\bar{Z}$ | $\Rightarrow$ | $0+\bar{1}+0+\bar{1}$ | $\Rightarrow$ $(0\,1\,0\,1)_2$ | $\Rightarrow$ $(5)_{10}$ |
| $\bar{W}+\bar{X}+Y+\bar{Z}$ | $\Rightarrow$ | $\bar{1}+\bar{1}+0+\bar{1}$ | $\Rightarrow$ $(1\,1\,0\,1)_2$ | $\Rightarrow$ $(13)_{10}$ |
| $\bar{W}+\bar{X}+Y+Z$ | $\Rightarrow$ | $\bar{1}+\bar{1}+0+0$ | $\Rightarrow$ $(1\,1\,0\,0)_2$ | $\Rightarrow$ $(12)_{10}$ |
| $\bar{W}+\bar{X}+\bar{Y}+Z$ | $\Rightarrow$ | $\bar{1}+\bar{1}+\bar{1}+0$ | $\Rightarrow$ $(1\,1\,1\,0)_2$ | $\Rightarrow$ $(14)_{10}$ |
| $W+X+\bar{Y}+\bar{Z}$ | $\Rightarrow$ | $0+0+\bar{1}+\bar{1}$ | $\Rightarrow$ $(0\,0\,1\,1)_2$ | $\Rightarrow$ $(3)_{10}$ |

$\therefore \quad Y = M_1 \cdot M_3 \cdot M_5 \cdot M_9 \cdot M_{12} \cdot M_{13} \cdot M_{14}$

**Ex . 4(A) :** Express the following functions as product of maxterm :

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

**Soln. :** First simplify this,

$$\therefore \quad Y = \bar{A}D + BD + \bar{B}D = \bar{A}D + D(B+\bar{B}) = \bar{A}D + D \cdot 1 = D(1+\bar{A}) = D \cdot 1 = D$$

$$\therefore \quad Y = D.$$

$$Y = D + 0 = D + A \cdot \bar{A} = (D+A)(D+\bar{A}) = (D+A+B\bar{B})(D+\bar{A}+B\bar{B})$$

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

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

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

| | | | | |
|---|---|---|---|---|
| $A+B+C+D$ | $\Rightarrow$ | $0+0+0+0$ | $\Rightarrow$ $(0\,0\,0\,0)_2$ | $\Rightarrow$ $(0)_{10}$ |
| $A+B+\bar{C}+D$ | $\Rightarrow$ | $0+0+\bar{1}+0$ | $\Rightarrow$ $(0\,0\,1\,0)_2$ | $\Rightarrow$ $(2)_{10}$ |
| $A+\bar{B}+C+D$ | $\Rightarrow$ | $0+\bar{1}+0+0$ | $\Rightarrow$ $(0\,1\,0\,0)_2$ | $\Rightarrow$ $(4)_{10}$ |
| $A+\bar{B}+\bar{C}+D$ | $\Rightarrow$ | $0+\bar{1}+\bar{1}+0$ | $\Rightarrow$ $(0\,1\,1\,0)_2$ | $\Rightarrow$ $(6)_{10}$ |
| $\bar{A}+B+C+D$ | $\Rightarrow$ | $\bar{1}+0+0+0$ | $\Rightarrow$ $(1\,0\,0\,0)_2$ | $\Rightarrow$ $(8)_{10}$ |
| $\bar{A}+B+\bar{C}+D$ | $\Rightarrow$ | $\bar{1}+0+\bar{1}+0$ | $\Rightarrow$ $(1\,0\,1\,0)_2$ | $\Rightarrow$ $(10)_{10}$ |
| $\bar{A}+\bar{B}+C+D$ | $\Rightarrow$ | $\bar{1}+\bar{1}+0+0$ | $\Rightarrow$ $(1\,1\,0\,0)_2$ | $\Rightarrow$ $(12)_{10}$ |
| $\bar{A}+\bar{B}+\bar{C}+D$ | $\Rightarrow$ | $\bar{1}+\bar{1}+\bar{1}+0$ | $\Rightarrow$ $(1\,1\,1\,0)_2$ | $\Rightarrow$ $(14)_{10}$ |

$\therefore \quad Y = M_0\, M_2\, M_4\, M_6\, M_8\, M_{10}\, M_{12}\, M_{14}.$

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

Scanned by CamScanner

**Soln. :** $Y = A + B + 0 = A + \bar{B} + C\bar{C} = (A + \bar{B} + C)(A + \bar{B} + \bar{C})$

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

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

$\Downarrow$ $(0\ 1\ 0\ 0)_2$ $\Downarrow$ 4 — $\Downarrow$ $(0\ 1\ 0\ 1)_2$ $\Downarrow$ 5 — $\Downarrow$ $(0\ 1\ 1\ 0)_2$ $\Downarrow$ 6 — $\Downarrow$ $(0\ 1\ 1\ 1)_2$ $\Downarrow$ 7

$\therefore Y = M_4 \cdot M_5 \cdot M_6 \cdot M_7$

---

**(iii)** $F(A, B, C) = A + BC$

**Soln. :** $Y = \overline{A + BC} = \bar{A} \cdot \overline{BC} = \bar{A} \cdot (\bar{B} + \bar{C})$

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

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

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

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

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

$\bar{A} + B + C \Rightarrow (1\ 0\ 0)_2 \Rightarrow (4)_{10}$

$\bar{A} + B + \bar{C} \Rightarrow (1\ 0\ 1)_2 \Rightarrow (5)_{10}$

$\bar{A} + \bar{B} + C \Rightarrow (1\ 1\ 0)_2 \Rightarrow (6)_{10}$

$A + \bar{B} + \bar{C} \Rightarrow (0\ 1\ 1)_2 \Rightarrow (3)$

$\bar{A} + \bar{B} + \bar{C} \Rightarrow (1\ 1\ 1)_2 \Rightarrow (7)_{10}$

$\therefore Y = \overline{M_3 \cdot M_4 \cdot M_5 \cdot M_6 \cdot M_7}$

## 4.5 Reduction Technique using Karnaugh Map :

***Why do we need K-map ?***

We have reduced logical expressions, using boolean algebra. To achieve, MINIMUM reduction of logical expression, one has to understand boolean laws or theorems. So while simplifying the boolean expressions, we have to think precisely and take next step carefully. This may lead to uncertainty about the final reduced expression we achieve. In short, we require a method ; which will reduce this uncertainty. The answer is **Karnaugh map**.

Scanned by CamScanner

Thus, K Map (Karnaugh Map) is a graphical means for representing Boolean or logical expressions so that they can be minimized or simplified as SOP or POS form. It is also known as veitch diagram as it was proposed by E. W. Veitch.

### 4.5.1 Standard 2/3/4 Variable K-maps :

K-map basically uses ***graphical chart***. Refer Fig. 4.1

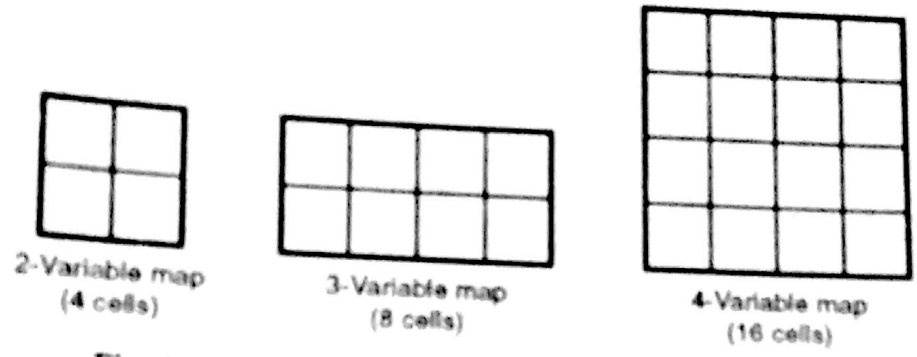



**Fig. 4.1 : Outlines of 2, 3 and 4 variable Karnaugh maps**

Each box is called cells. The number of boxes depends upon 'n' variables

n = 2, boxes will be $2^n = 2^2 = 4$

n = 3, boxes will be $2^n = 2^3 = 8$

n = 4, boxes will be $2^n = 2^4 = 16$

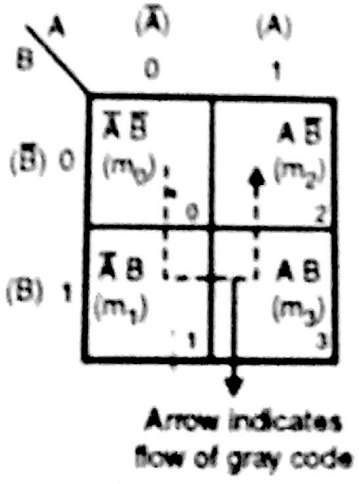
Let's first observe standard 2, 3, 4 and 5 variable K-map

#### 2 Variable K–Map

The following Map I shows 2 variable K-Map

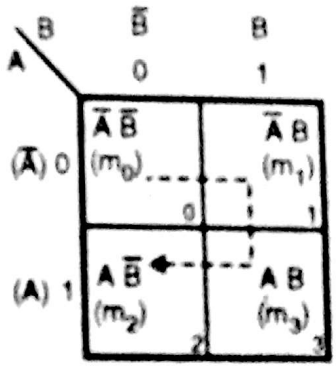
| Minterm | Decimal | A | B |
|---|---|---|---|
| $m_0$ | 0 | 0 | 0 |
| $m_1$ | 1 | 0 | 1 |
| $m_2$ | 2 | 1 | 0 |
| $m_3$ | 3 | 1 | 1 |

(a)




(b)

OR




(c)

**Map I**

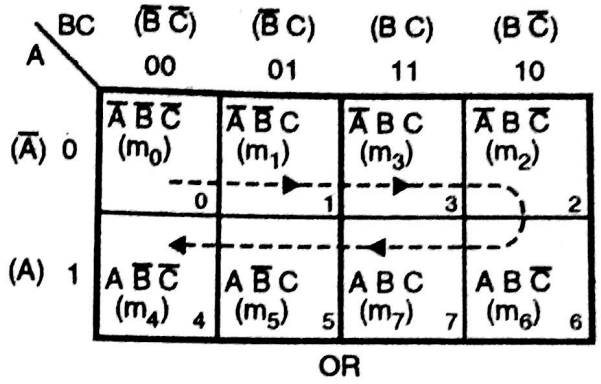
Two variable indicates 4 states. Here A = MSB (by default). The dotted line in K Map indicates flo of gray code.

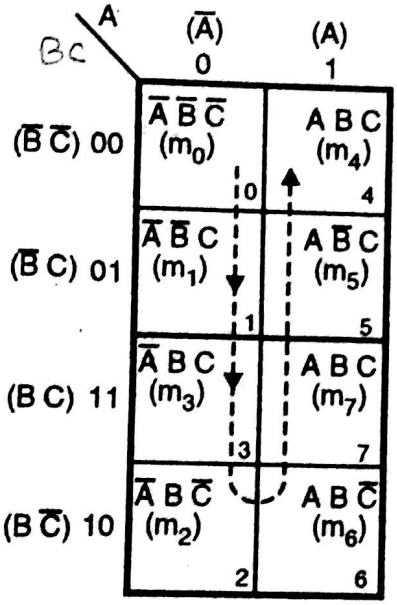
Scanned by CamScanner

## 3 Variable K–Map

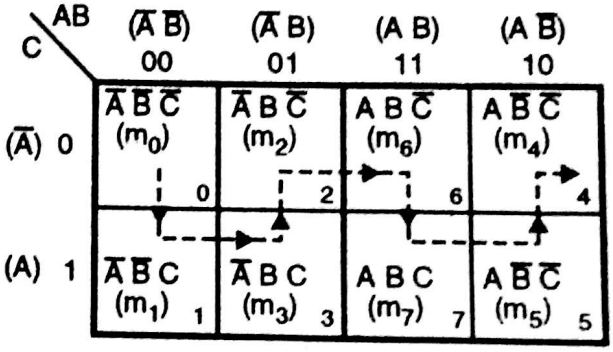
The Map II shows 3 variable K–Map :

| Minterm | Decimal | MSB A | B | LSB C |
|---|---|---|---|---|
| $m_0$ | 0 | 0 | 0 | 0 |
| $m_1$ | 1 | 0 | 0 | 1 |
| $m_2$ | 2 | 0 | 1 | 0 |
| $m_3$ | 3 | 0 | 1 | 1 |
| $m_4$ | 4 | 1 | 0 | 0 |
| $m_5$ | 5 | 1 | 0 | 1 |
| $m_6$ | 6 | 1 | 1 | 0 |
| $m_7$ | 7 | 1 | 1 | 1 |




(a)




(b)

**Map II**

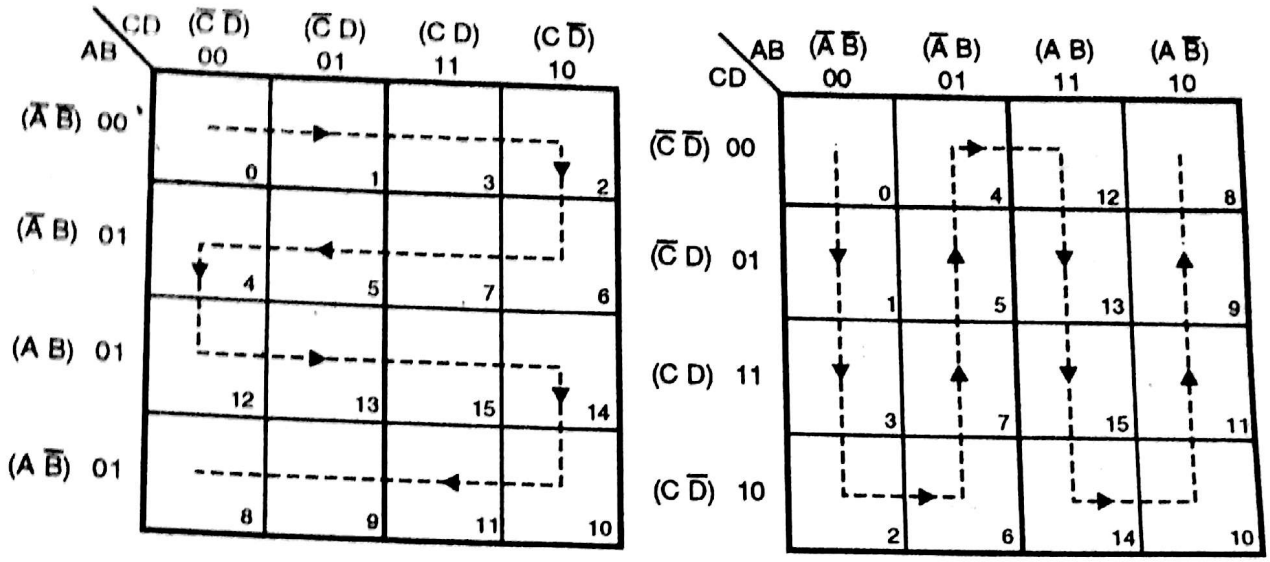
Total eight boxes are required to represent 8 states. Total 3 way of K–Map is shown. It is upto y that which one you remember very easily. The dotted line with arrow indicates flow of gray code.

## 4 Variable K–Map

Map III shows basic skeleton for 4 variable K–Map.

| Decimal | MSB A | B | C | LSB D |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| ⋮ | | | | |
| 14 | 1 | 1 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 |

Scanned by CamScanner




**Map III**

Dotted line indicates flow of gray code.

## 5 Variable K–Map

The way five variable K–Map is represented is slightly different from above case, the basic skeleton is as shown in Map IV (on the next page).

Here MSB bit A has two states '0' or '1'.

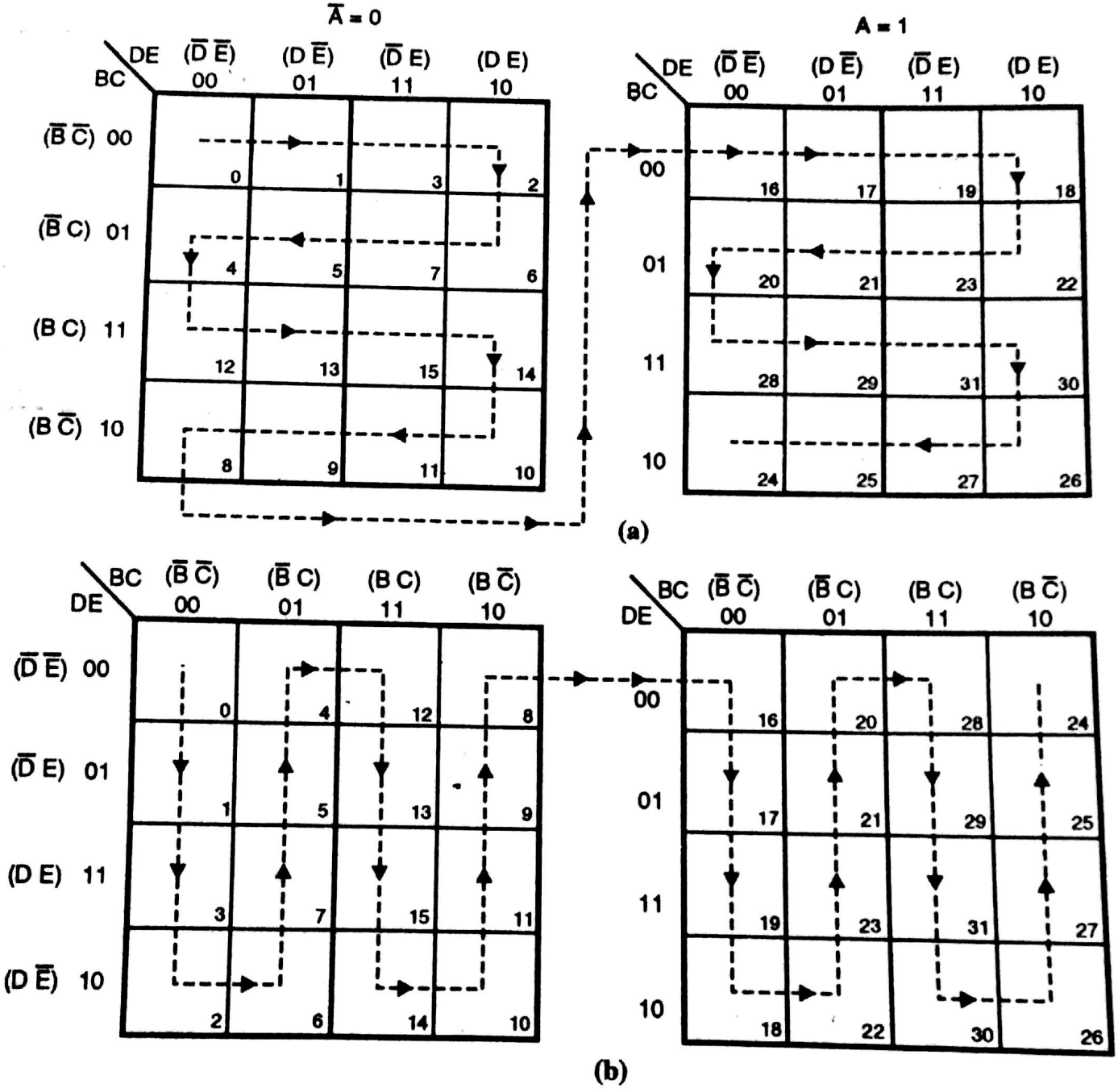
Under $\overline{A}$ = 0 and A = 1, two four variable K–Map are drawn. The numbering sequence is as in four variable K–Map. Only when A = 1 then ABCDE = 10000 or 1XXXX, then starting number is '16' (decimal) not '0' (decimal)

**Note :**

1. The K-map, we have drawn are specifically for product terms i.e. minterms.
2. As seen, each row and column has variable or variables, with its complement.
3. When we move from one cell to next along any row or column, one and only one variable in the product term changes.

   The change is either to complemented or uncomplemented form. THE IMPORTANT POINT HERE IS, IN K-MAP WE USE 'GRAY CODES' WHERE ONE BIT CHANGES AT A TIME. It is a key factor in the way in which K-map works.
4. We have shown 2 or 3 ways of drawing K-map, for 2, 3, 4 and 5 variables. You select only 'ONE' pattern from that, and follow the same throughout your study.

Scanned by CamScanner



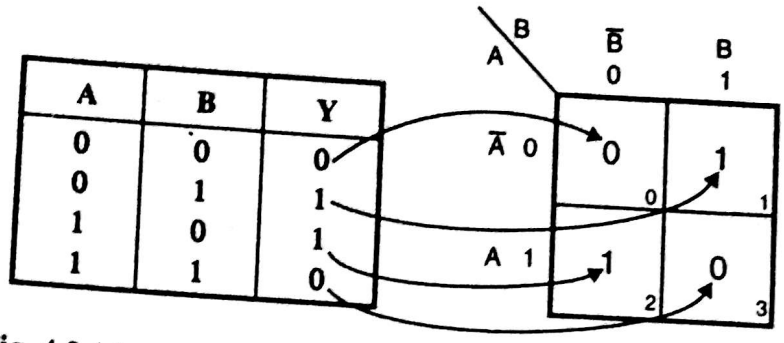

**Map IV**

## 4.5.2 Transferring Truth Table to K-map :

Here I would like to show you, how we should 'MAP' truth Table in K-map.

For understanding purpose, I will show you example for 2 and 3 variables. But the same is valid for 4 and 5 variables also.

Scanned by CamScanner




| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Fig. 4.2 (a) : Representation of 2-variable truth Table on K-map**

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

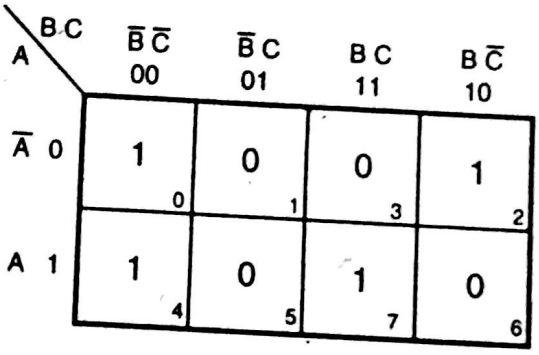



**Fig. 4.2 (b) : Representation of 3-variable truth Table on K-map**

Till this point, we have studied about basic K-map and transferring truth Table to K-map. Now we have to derive SOP form, and that is also in minimized form. For that we have to study ***Grouping in K-map***.

## 4.6 Grouping of Variables in K–map :

In K–Map '0' or '1' are grouped together to get simplified Boolean expression. If you group '0', you are grouping maxterms so equation you get is in POS form. But when you group '1', you are grouping minterms; so equation you get has SOP form. Before we start, let's see some definations.

**Pairs** : The group of two adjacent '0' (or 1) is called pair.

**Quad** : The group of four adjacent '0' (or 1) is called quad.

**Octets** : The group of eight adjacent '0' (or 1) is called octet.

For grouping the variables, there are some standard rules. We will study the same now.

Scanned by CamScanner

**Note** : In K-Map one should see that you try to cover more number of 1s (or '0' s). The grouping follows binary rule, i.e. you can group 64, 32, 16, 8, 4, 2 or 1 variable. You can not group 3 or 5, 6, 7 or 9 to 15 and so on.

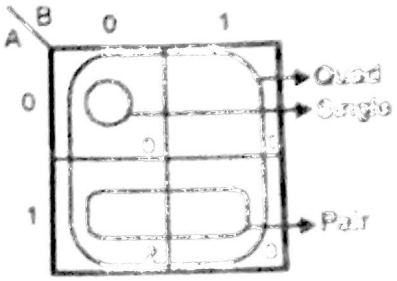
### 4.8.1 Grouping of Variables in Two Variable K–Map :

Map V shows grouping of variables in 2 variable K–Map.

Variables are grouped horizontally or vertically but not diagonally. If all variable in K–Map having logic level '1' are grouped, final expression is always '1' (or $V_{CC}$)

But if all variable in K–Map having logic level '0' are grouped final expression is always '0' (or GND).

Let's see example for the same.




**Map V**

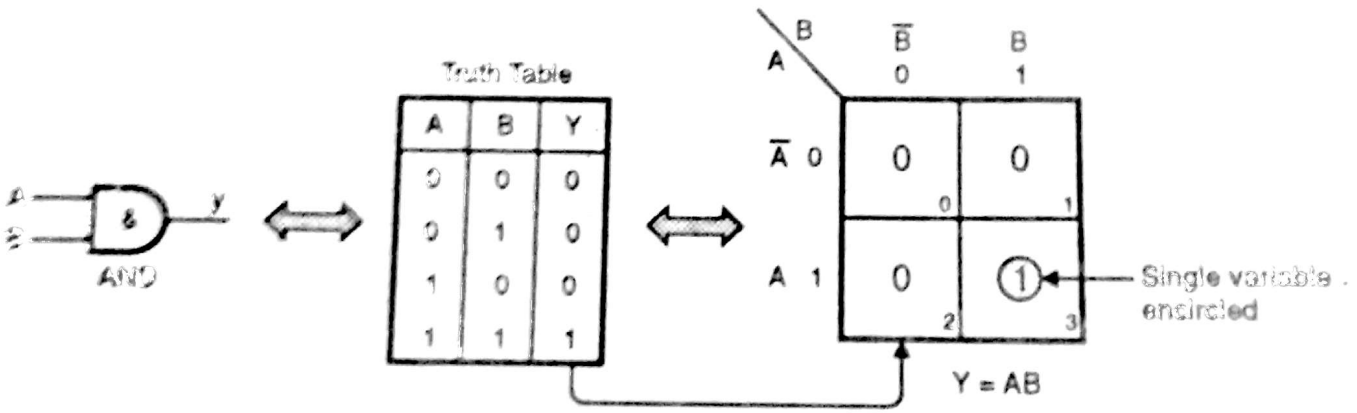
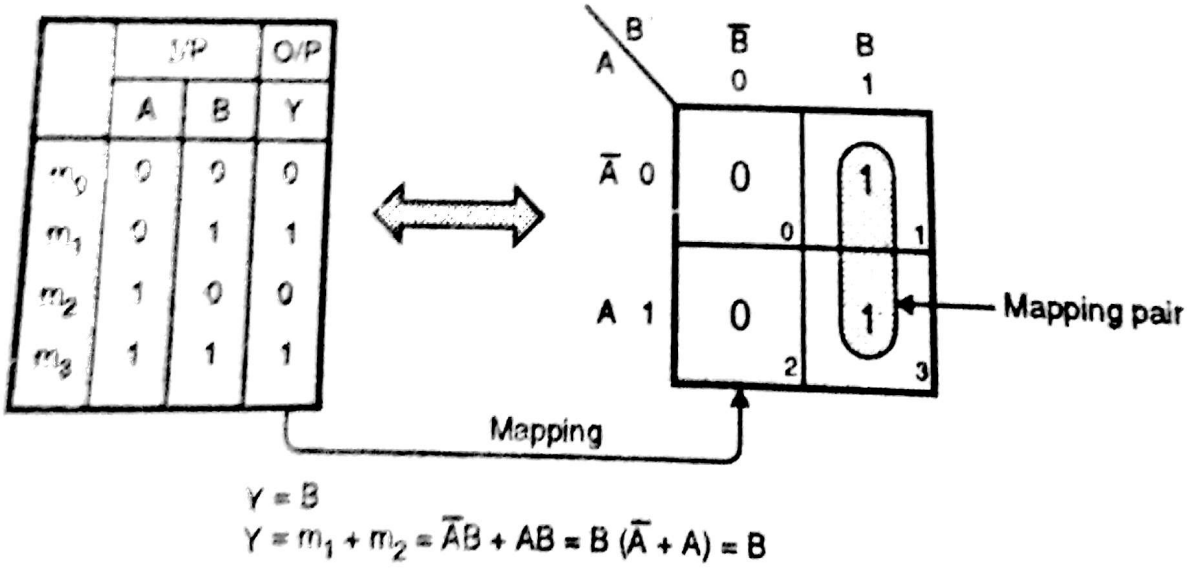



**Fig. 4.3 : K-map for input AND gate**

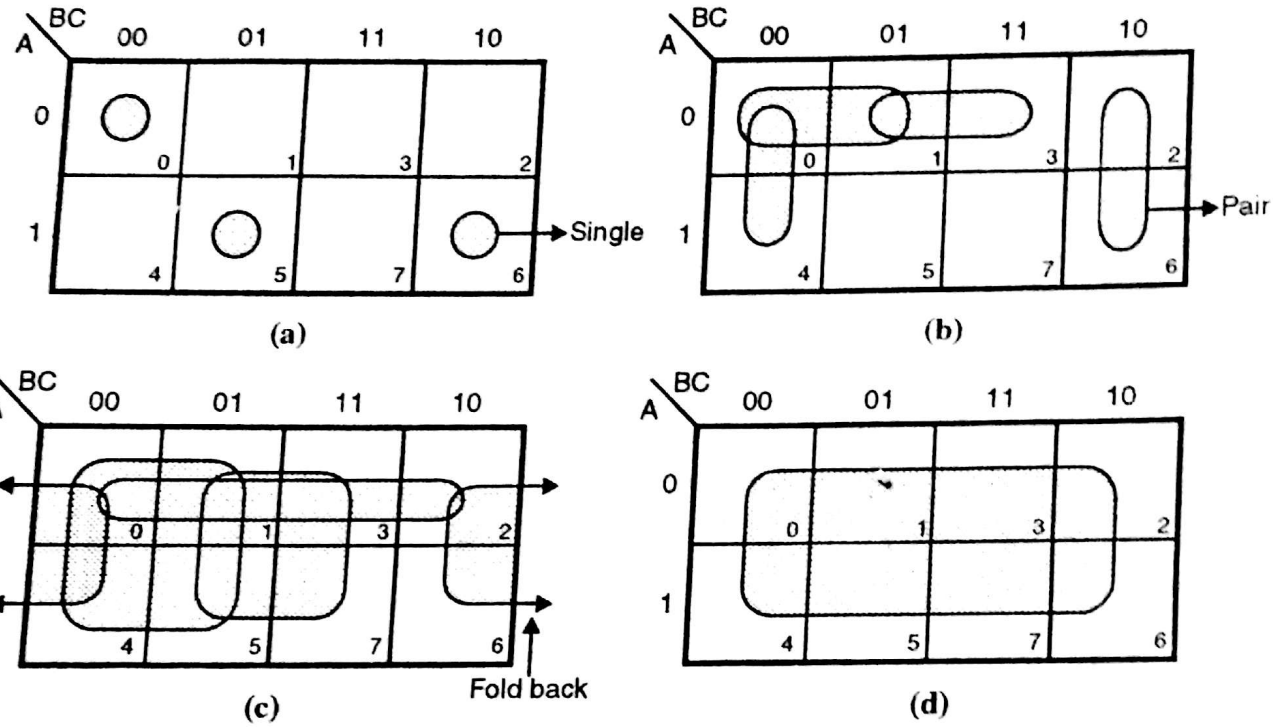
Suppose we have following truth Table , then the K-map will be,




Scanned by CamScanner

### 4.6.2 Grouping of Variables in Three Variable K–Map :

Map VI shows how variables are grouped in three variable K–Map.




**Map VI**

Map VI (c) shows an interesting case. If the K–Map is folded such that its two end meet, a quad is formed. This is referred to as a ***foldback***.

---

**Ex. 5 :** Find the expression for the following K–Maps :

*(i)*

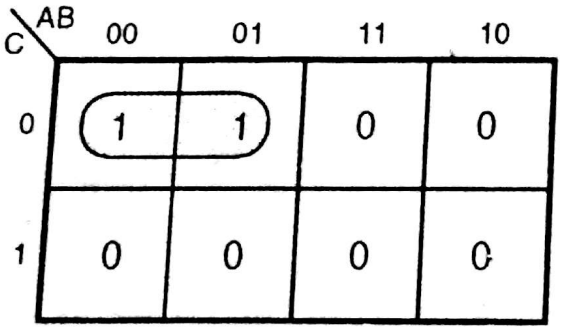



**Fig. Ex. 4.5 (i)**

**Soln. :** Refer Fig. Ex. 4.5 (i), since 1's are grouped, the final expression will be in SOP form.

As you are aware with the fact that in K–Map we are not supposed to consider those variables, that changes the state i.e. $\bar{B}$ to B or B to $\bar{B}$. We should consider only those variables those remains constant within the grouping block.

Scanned by CamScanner

Refer Fig. Ex. 4.5 (i) (a), as shown $\bar{C}$ remains unchanged (vertically). Now go horizontally the variables are $\bar{A}\bar{B}$ and $\bar{A}B$. Here $\bar{A}$ is constant but $\bar{B}$ changes to B.

$\therefore$ Neglect $\bar{B}$ and B because,

$$\bar{A}\bar{B} + \bar{A}B = \bar{A}(\bar{B} + B)$$
$$= \bar{A} \cdot 1 = \bar{A}.$$

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

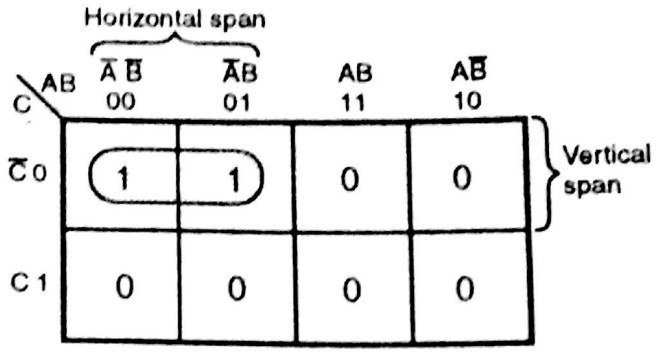



**Fig. Ex. 4.5 (i) (a)**

(ii) Refer Fig. Ex. 4.5 (ii).

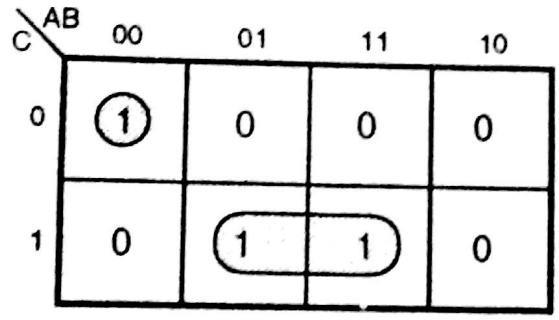



**Fig. Ex. 4.5 (ii)**

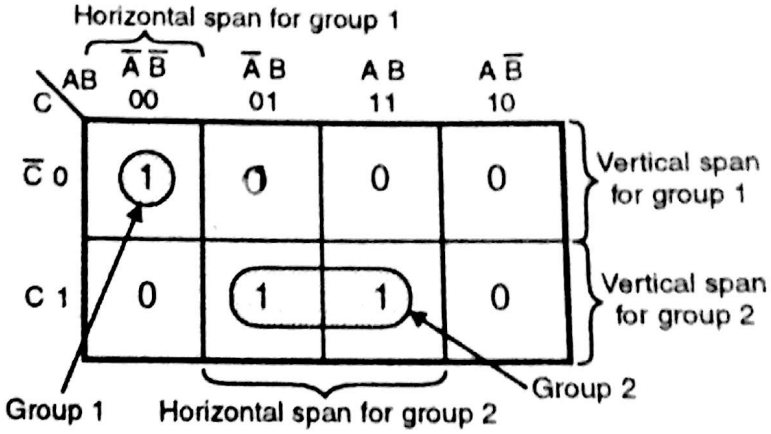



**Fig. Ex. 4.5 (ii) (a)**

**Soln.** : Refer Fig. Ex. 4.5 (ii) (a)

$$Y = \bar{A}\bar{B}\bar{C} + BC.$$
$$\downarrow \quad \downarrow$$

group 1 group 2

If you observe group 1, only one is encircled, $\therefore$ no change of variable is encountered and you have to select all the variables covering that box. i.e. Horizontally $\bar{A}\bar{B}$, vertically $\bar{C}$, $\therefore$ **Ans.** = $\bar{A}\bar{B}\bar{C}$.

For 2nd group vertical span is constant i.e. C.

But for horizontal we have two term $\bar{A}B$ and AB. So $\bar{A}$ changes to A. So leave it.

$\therefore$ For 2nd group horizontally we have $\longrightarrow$ B

vertically we have $\longrightarrow$ C $\therefore$ **Ans.** = BC.

**Finally total output Y = 1st group + 2nd group (SOP form)** $\boxed{Y = \bar{A}\bar{B}\bar{C} + BC}$

(iii) Refer Fig. Ex. 4.5 (iii)

Scanned by CamScanner

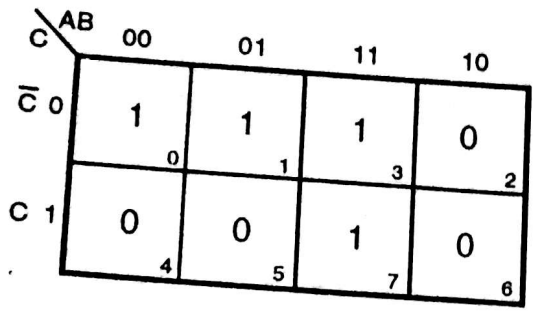



Fig. Ex. 4.5 (iii)

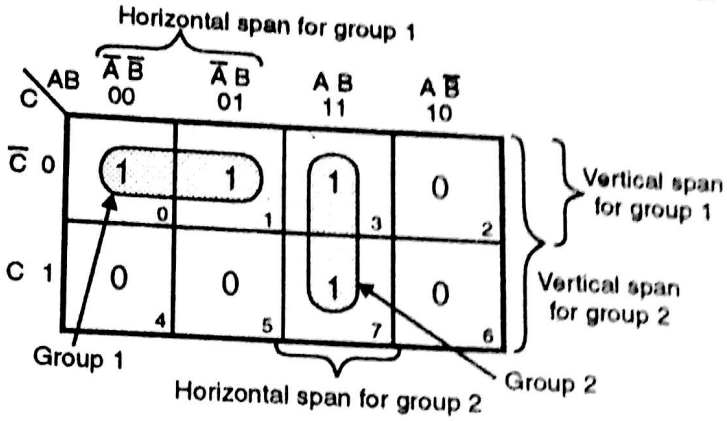



Fig. Ex. 4.5 (iii) (a)

**Soln.** : Refer Fig. Ex. 4.5 (iii) (a)

For group 1 : Vertical span $\longrightarrow \bar{C}$

Horizontal span $\longrightarrow \bar{A}\bar{B}, \bar{A}B$ $\therefore$ Finally $\rightarrow \bar{A}$.

Group 1 $\rightarrow \bar{A}\bar{C}$

group 2 : Vertical span $\longrightarrow \bar{C}, C \rightarrow$ So $\bar{C} + C = 1$.

Horizontal span $\longrightarrow$ AB only.

$\therefore$ Finally group 2 $\longrightarrow AB \cdot 1 \rightarrow AB$.

$\therefore$ output $Y = \underset{\text{group 1}}{\bar{A}\bar{C}} + \underset{\text{group 2}}{AB}$

(iv) Refer Fig. Ex. 4.5 (iv)

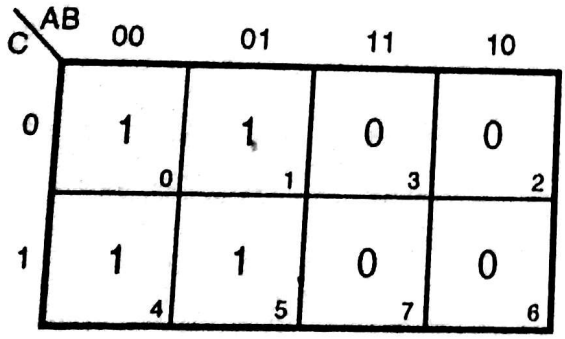



Fig. Ex. 4.5 (iv)

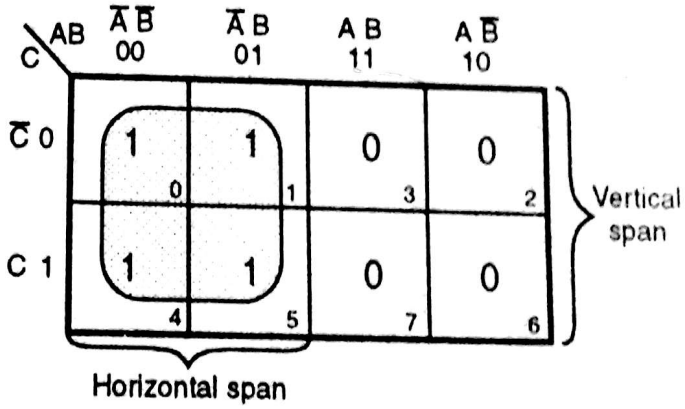



Fig. Ex. 4.5 (iv) (a)

**Soln.** : Refer Fig. Ex. 4.5 (iv) (a)

$\therefore$ Horizontal span $\longrightarrow \bar{A}\bar{B}, \bar{A}B \longrightarrow \bar{A}$ only

Vertical span $\longrightarrow \bar{C}, C \longrightarrow 1$

$\therefore$ **Ans.** $= \bar{A} \cdot 1 = \bar{A}$ $\therefore$ $\boxed{\text{output } Y = \bar{A}}$

Scanned by CamScanner

(v) Refer Fig. Ex. 4.5 (v)

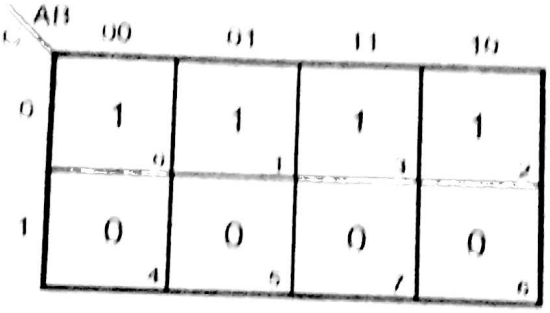



Fig. Ex. 4.5 (v)

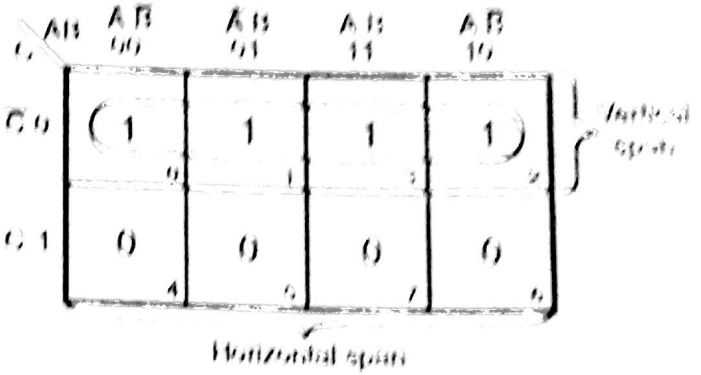



Fig. Ex. 4.5 (v) (a)

**Soln.** : Refer Fig. Ex. 4.5 (v) (a)

∴ Horizontal span → AB, AB, AB, AB

Both the variables are changing.

∴ $AB + AB + AB + AB = A(B + B) + A(B + B) = A \cdot 1 + A \cdot 1 = A + A = 1$

Vertically → C only.

∴ $\boxed{\text{output } Y = C}$

(vi) Refer Fig. Ex. 4.5 (vi)

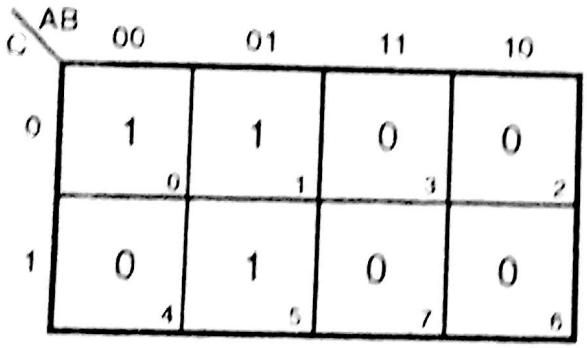



Fig. Ex. 4.5 (vi)

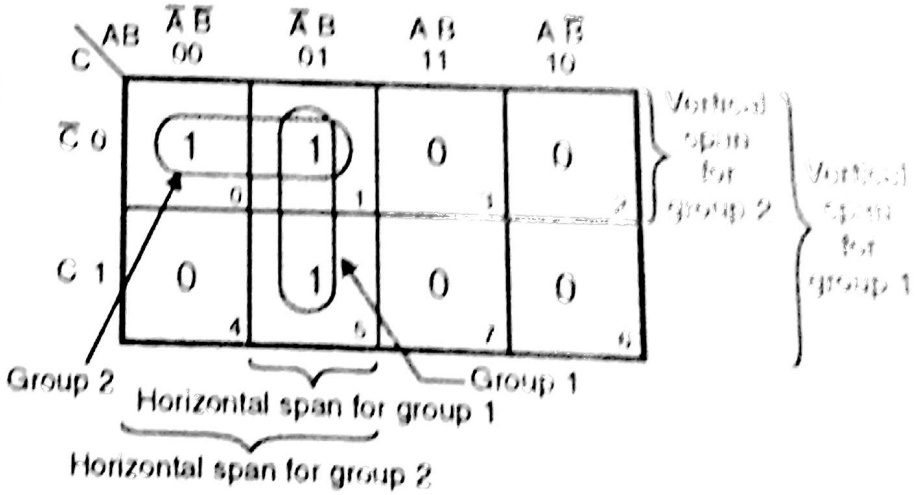



Fig. Ex. 4.5 (vi) (a)

**Soln.** : Refer Fig. Ex. 4.5 (vi) (a). This is specific case of overlap.

**Group 1 :** From vertical span → $\bar{C}, C$ → 1. From Horizontal span → $\bar{A}B$.

∴ group 1 → $\bar{A}B$.

**Group 2 :** From vertical span → $\bar{C}$ only. From Horizontal span → $\bar{A}B$, $\bar{A}\bar{B}$ → $\bar{A}$ only.

∴ group 2 → $\bar{A}\bar{C}$. ∴ $\boxed{\text{output } Y = \bar{A}B + \bar{A}\bar{C}}$

(vii) Refer Fig. Ex. 4.5 (vii)

Scanned by CamScanner

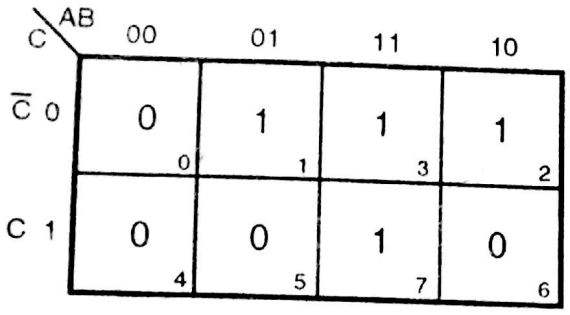



Fig. Ex. 4.5 (vii)

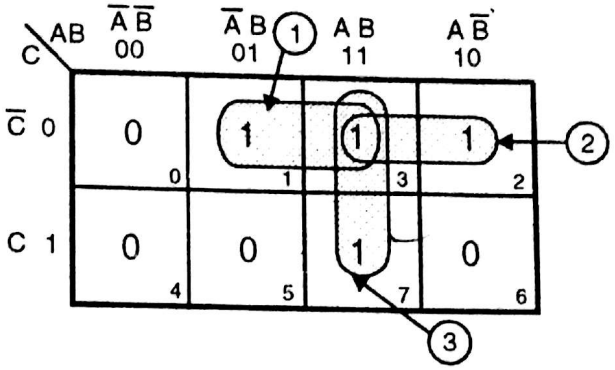



Fig. Ex. 4.5 (vii) (a)

**Soln. :** Refer Fig. Ex. 4.5 (vii) (a). Now groups are directly numbered.

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

(1) (2) (3)

---

(viii) Refer Fig. Ex. 4.5 (viii)

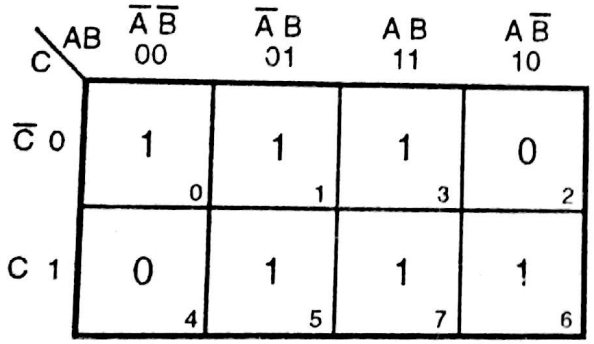



Fig. Ex. 4.5 (viii)

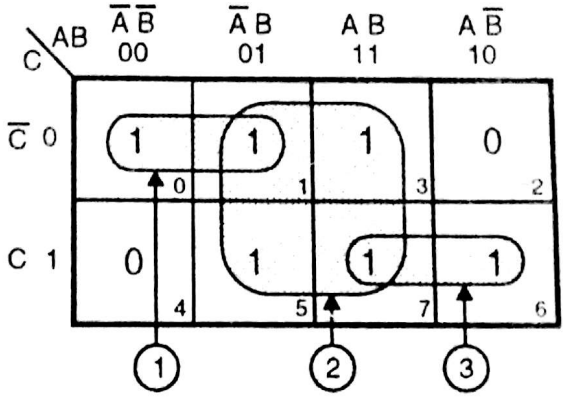



Fig. Ex. 4.5 (viii) (a)

**Soln. :** Refer Fig. Ex. 4.5 (viii) (a).

output $Y = \bar{A}\bar{C} + B + AC$

(1) (2) (3)

---

(ix) Refer Fig. Ex. 4.5 (ix).

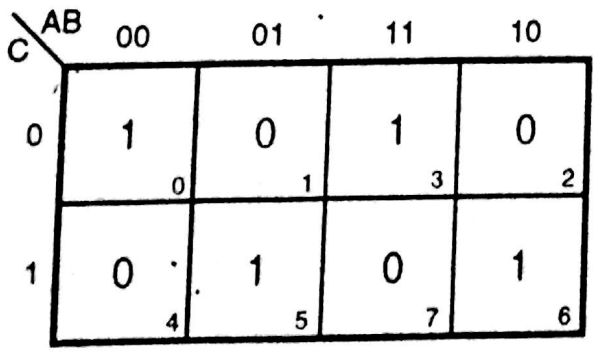



Fig. Ex. 4.5 (ix)

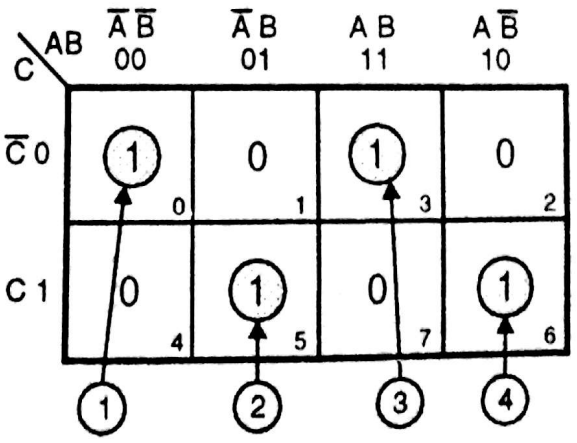



Fig. Ex. 4.5 (ix) (a)

Scanned by CamScanner

**Soln. :** Refer Fig. Ex. 4.5 (ix) (a).

output $Y = \underset{(1)}{\bar{A}\bar{B}\bar{C}} + \underset{(2)}{\bar{A}BC} + \underset{(3)}{AB\bar{C}} + \underset{(4)}{A\bar{B}C}$

$= \bar{A}\left(\bar{B}\bar{C} + BC\right) + A\left(B\bar{C} + \bar{B}C\right) = \bar{A}\,(A \odot B) + A\,(A \oplus B)$

(x) Refer Fig. Ex. 4.5 (x)

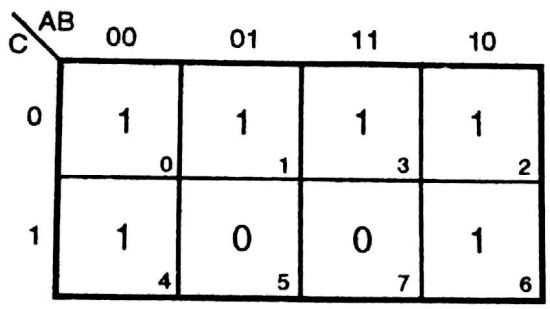



**Fig. Ex. 4.5 (x)**

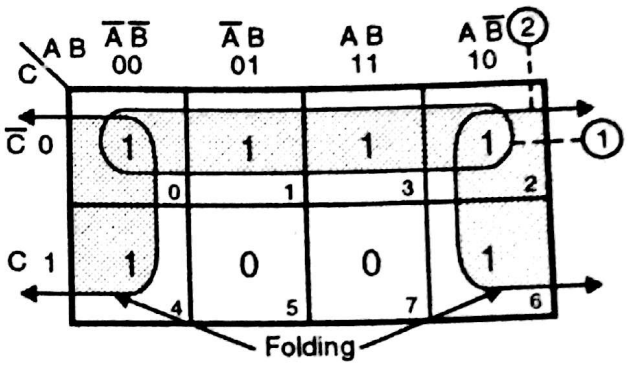



**Fig. Ex. 4.5 (x) (a)**

**Soln. :** Refer Fig. Ex. 4.5 (x) (a)

This is Interesting case of folding.

(a) For group 1, Answer is $\bar{C}$.

(b) For group 2, we have covered 1's of boxes 0, 4, 2, 6.

These boxes forms 'Quad'.

$\therefore$ From Horizontal span we have $\longrightarrow A\bar{B},\ \bar{A}\bar{B}$

$\longrightarrow A\bar{B} + \bar{A}\bar{B}$

$\longrightarrow \bar{B}\left(A + \bar{A}\right) \rightarrow \bar{B} \cdot 1 \rightarrow \bar{B}$

From vertical span we have $\rightarrow \bar{C} + C = 1$

$\therefore$ for group 2, answer is $\bar{B}$

$\therefore$ Finally

| output $Y = \underset{(1)}{\bar{C}} + \underset{(2)}{\bar{B}}$ |
|---|

(xi) Refer Fig. Ex. 4.5 (xi)

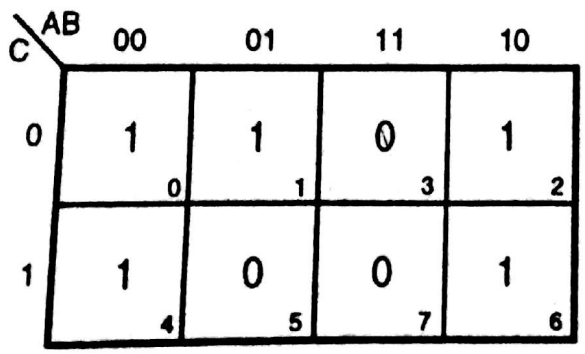



**Fig. Ex. 4.5 (xi)**

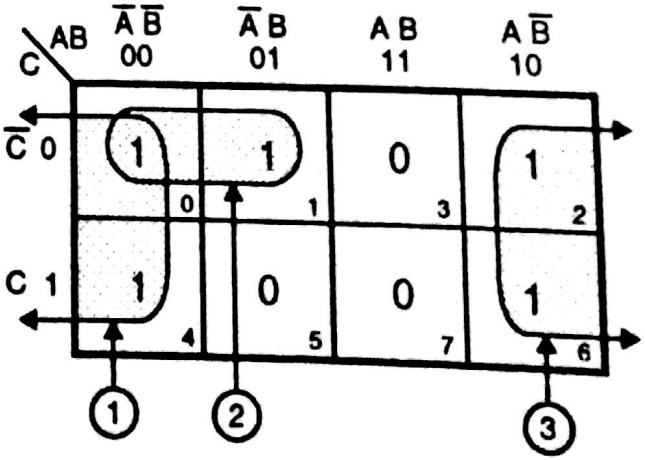



**Fig. Ex. 4.5 (xi) (a)**

Scanned by CamScanner

**Soln.** : Refer Fig. Ex. 4.5 (xi) (a).

output $Y = \underset{\downarrow}{B} + \underset{\downarrow}{\bar{A}\bar{C}}$

$\therefore$ (1) (2)

In Ex. 5 we have covered many different cases for 3 variable K–Map. Now we are going to see more cases for 2 variables, 4 variables. But basic is going to remain same.

### 4.6.3 Grouping of Variables in Four Variable K–Map :

Grouping of variables in four variable K – Map is as shown in Map VII. (next page)

Different ways of grouping under pair, quad and octet are shown.

Uptill now we have studied different grouping. Here you may ask question that after grouping how you are suppose to write down Boolean expression ?

You follow steps as specified :

**Step I :** First encircle required number of 1's for SOP form.

**Step II :** Now take encircling of 1's one by one. In one K–Map you may have one or more encirclements.

**Step III :** Let's say you have select one of the group. Now carefully observe, 'SPAN' of encircle i.e. how many ***boxes*** it has covered horizontally and vertically. Both the directions are important because in both direction we have written boolean expressions.

Now take ***vertical direction*** and see any variable changing the state. Here changing the state means x to $\bar{x}$ or $\bar{x}$ to x. If variable is changing neglect it. You have to consider only those variables, those are remaining constant. Suppose you got variable I J.

Now select ***horizontal direction***, follow same above rule. Suppose you got variable K only.

$\therefore$ Final term is ***I J K***.

The variable may be complemented or uncomplemented.

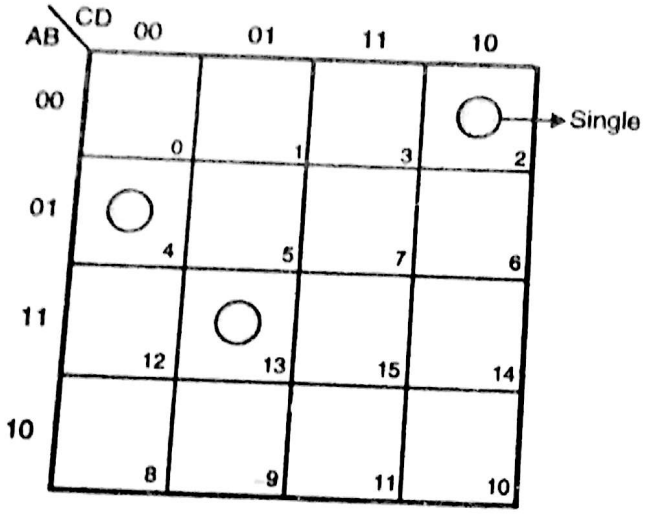
**Step IV :** Encircling 1's, ***once more*** is allowed, but you should check that it should not form something called as ***Redundancy***.

The block is redundant or encircling that '1' is not required, as it will on simplification, get cancelled out. This would have resulted in additional hardware. (This specific case is mentioned in Ex. 16, Ex. 17).

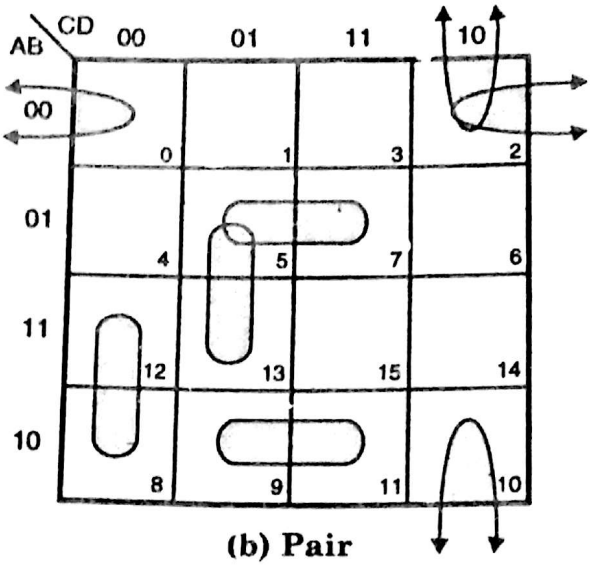
**Important Note :**

**You have to obtain boolean expression using K–map in such a way that further simplification or reduction is not at all possible.**

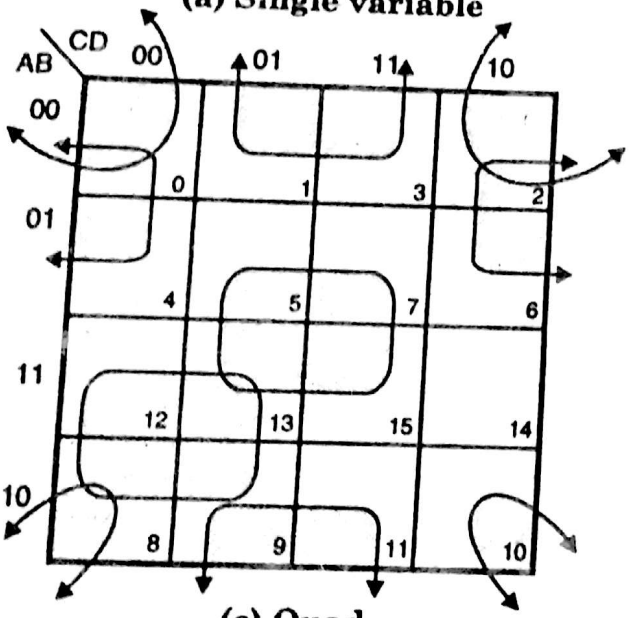
Scanned by CamScanner



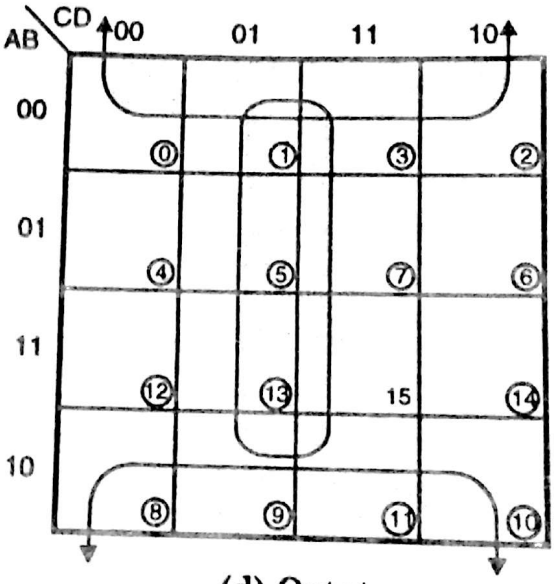

(a) Single variable



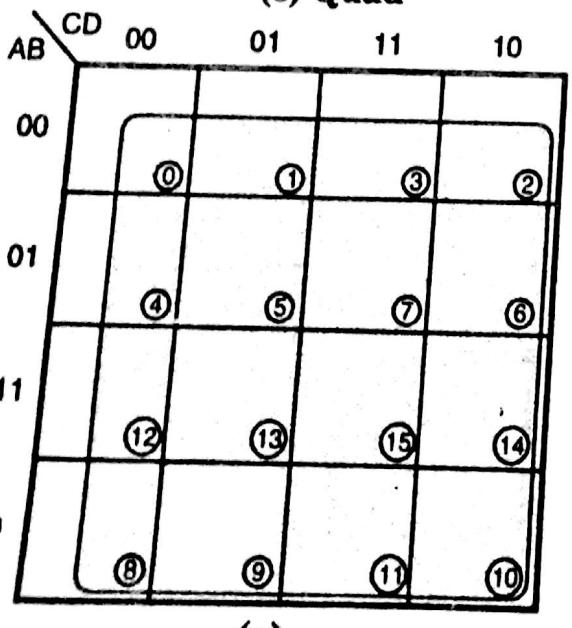

(b) Pair




(c) Quad




(d) Octet




(e)

**Map VII**

Scanned by CamScanner

**Ex. 6 :** (i) Find expression for the following K-map

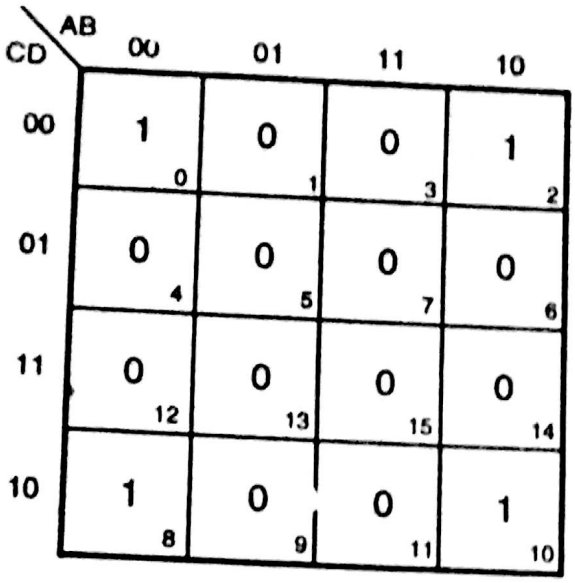



**Fig. Ex. 4.6 (i)**

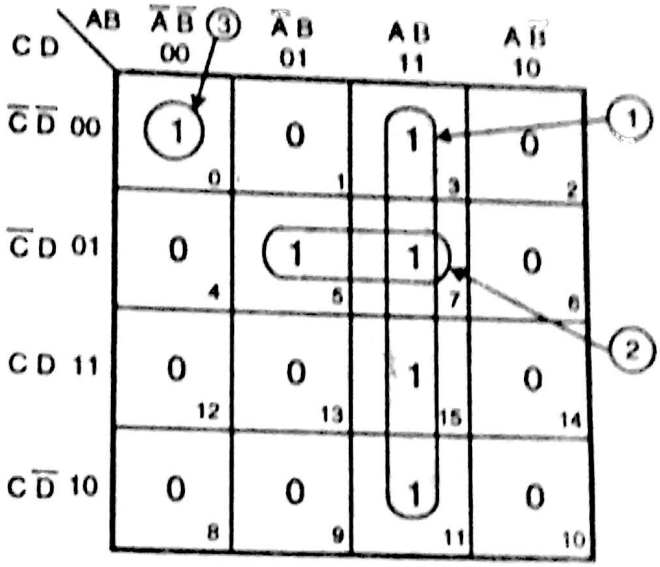



**Fig. Ex. 4.6 (i) (a)**

**Soln. :** Grouping is shown in Fig. Ex. 4.6 (i) (a)

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

(1) (2) (3)

**(ii)**

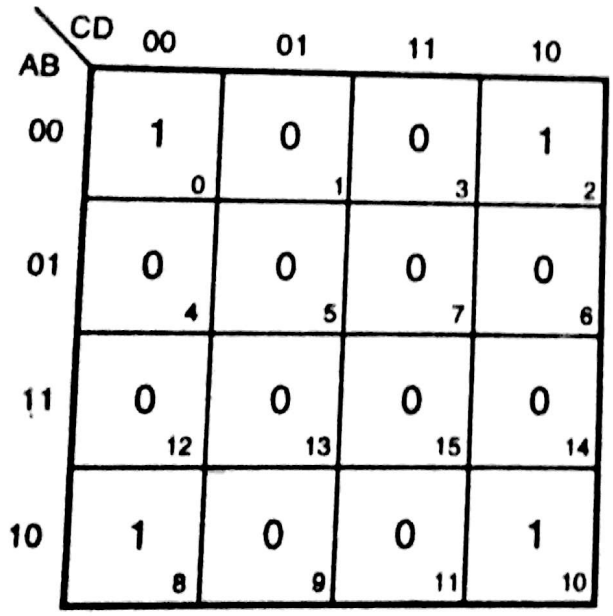



**Fig. Ex. 4.6 (ii)**

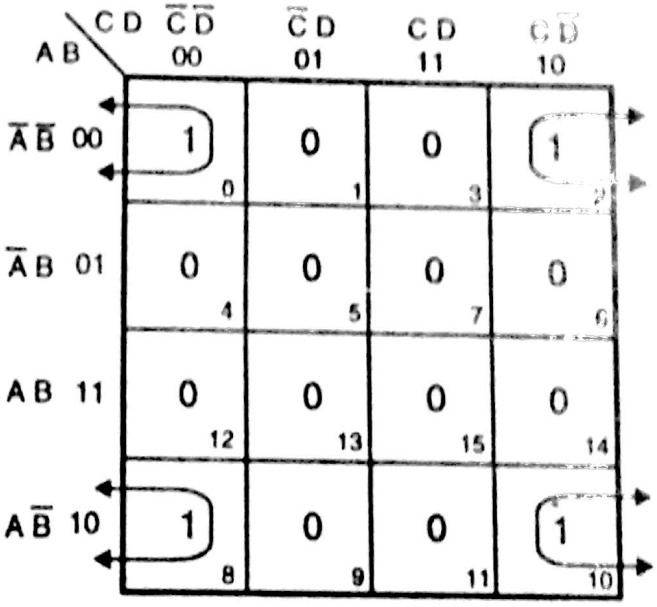



**Fig. Ex. 4.6 (ii) (a)**

**Soln. :** Fig. Ex. 4.6 (ii) (a) shows grouping.

For vertical we have $\rightarrow \bar{C}\bar{D} + C\bar{D} \rightarrow \bar{D}$

horizontal we have $\rightarrow \bar{A}\bar{B} + A\bar{B} \rightarrow \bar{B}$

$\therefore$ output $Y = \bar{B}\bar{D}$

Uptill now we have seen that directly filled K-Map with encirclement was given. This was given just to know different ways of encircling 1's. But now question is how to fill up the K-Map? To understand this we will solve some more examples.

Scanned by CamScanner

**Ex. 7 :** F (A, B, C, D) = $\Sigma$m (0, 1, 2, 5, 13, 15) Draw K-Map and find minimised Boolean expression.

**Soln. :** From example we can conclude that K-Map required will be of 4 variables (A, B, C, D). The conclusion can be also made up from highest state. Highest state = 15. To represent 15 we require minimum 4 bits.

**Step 1 :** Now first draw basic skeleton of K-Map (Refer Map III)

**Step 2 :** Presently box is empty. Given is '$\Sigma$m' it means ***minterm*** ∴ write down '1' in the box whose number is specified in the example i.e. 0, 1, 2, 5, 13 and 15

**Step 3 :** Remaining boxes of K-Map should be filled up with zeros

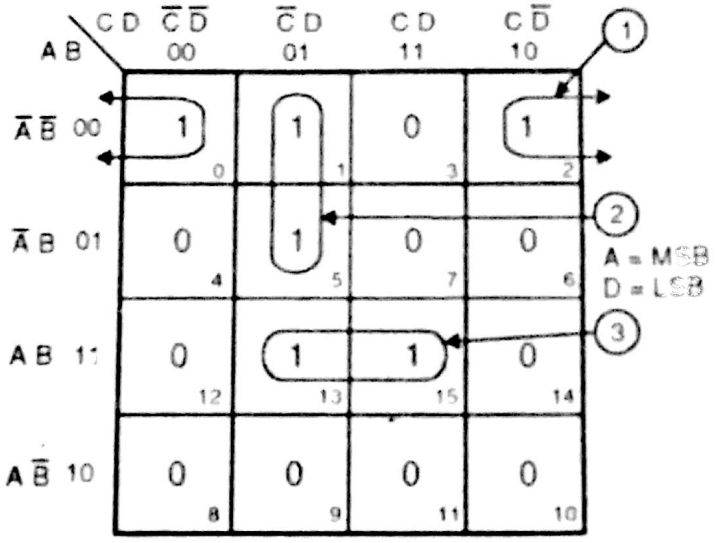



**Fig. Ex. 4.7 (a)**

**Ans. :** $Y = \underset{(3)}{ABD} + \underset{(2)}{\bar{A}CD} + \underset{(1)}{\bar{A}\bar{B}D}$

**Ex. 8 :** F (A, B, C, D) = $\Sigma$m (1, 3, 5, 9, 11, 13).

**Soln. :** Procedure is same as that defined in Ex. 7.

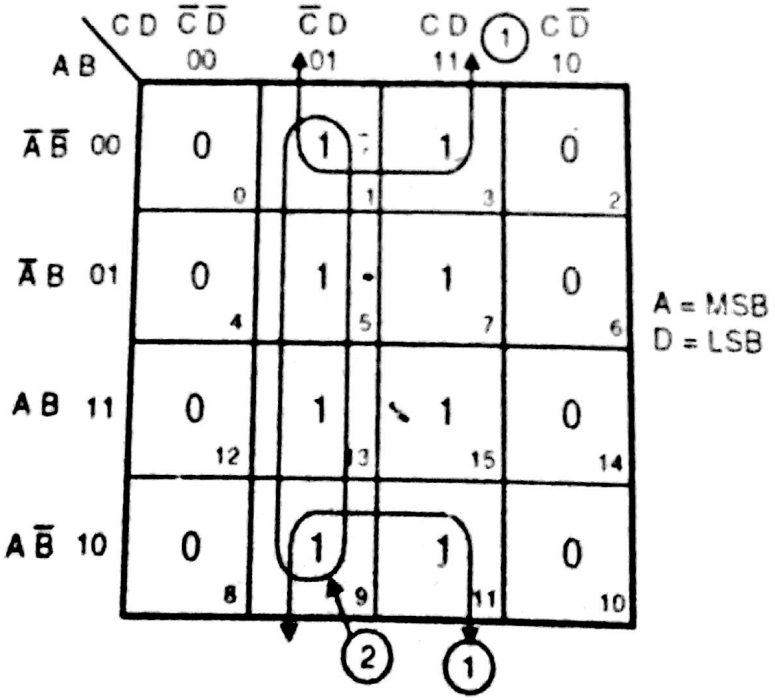



**Fig. Ex. 4.8 (a)**

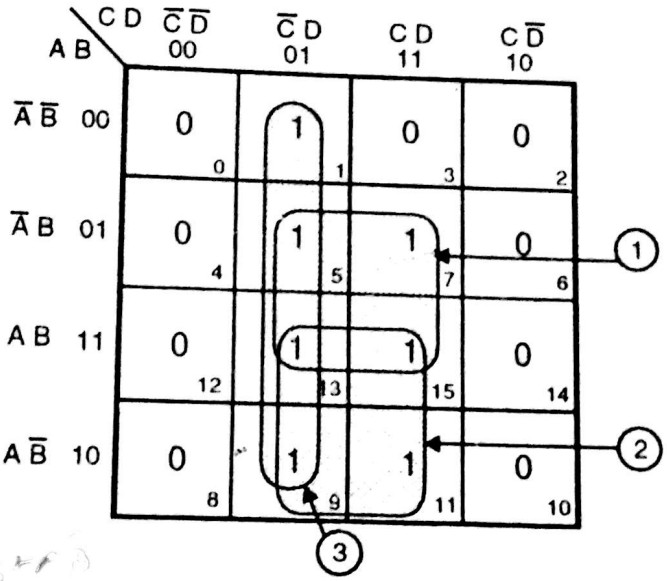
Scanned by CamScanner

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

(1) (2)

**Ex. 9 :** F (A, B, C, D) = Σm (1, 5, 7, 9, 11, 13, 15).

**Soln. :** Procedure is same as that defined in Ex. 7.




F (A, B, C, D) = $\bar{C}D + AD + BD$ ..... (a)

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

**Fig. Ex. 4.9 (a)**

Let's draw the circuit.

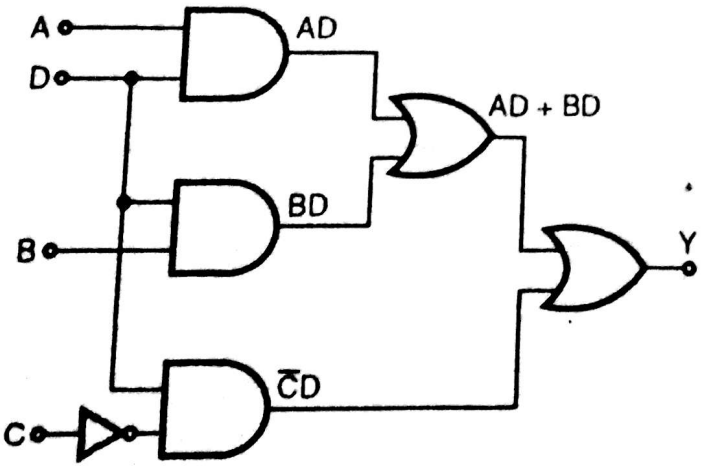



**Fig. Ex. 4.9 (b)**

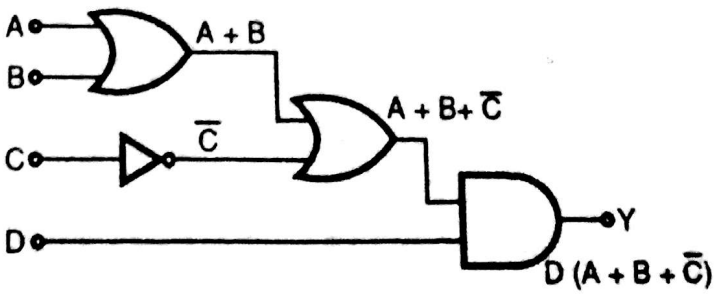



**Fig. Ex. 4.9 (c)**

**Ex. 10 :** F (A, B, C, D) = Σm (1, 3, 4, 5, 7, 9, 11, 13, 15).

**Soln. :** Procedure is same as that followed in Ex. 7.

Scanned by CamScanner

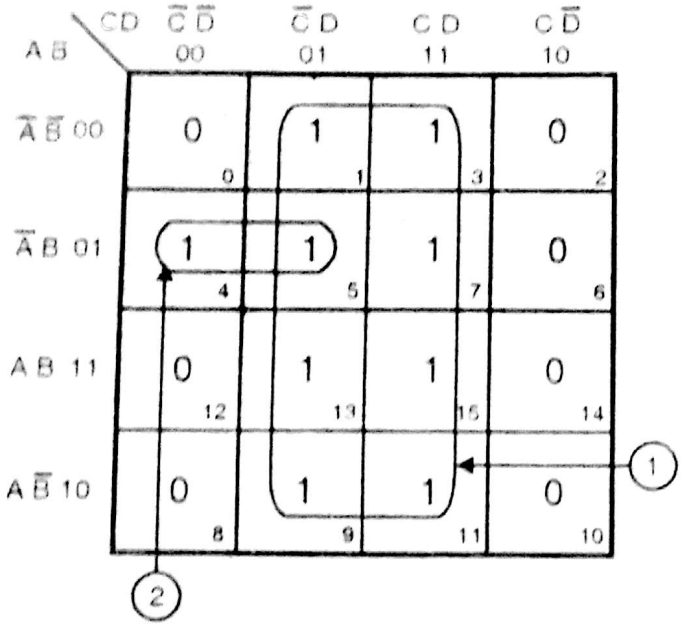



Fig. Ex. 4.10 (a)

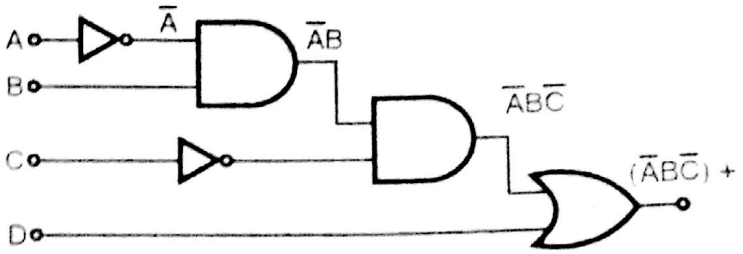


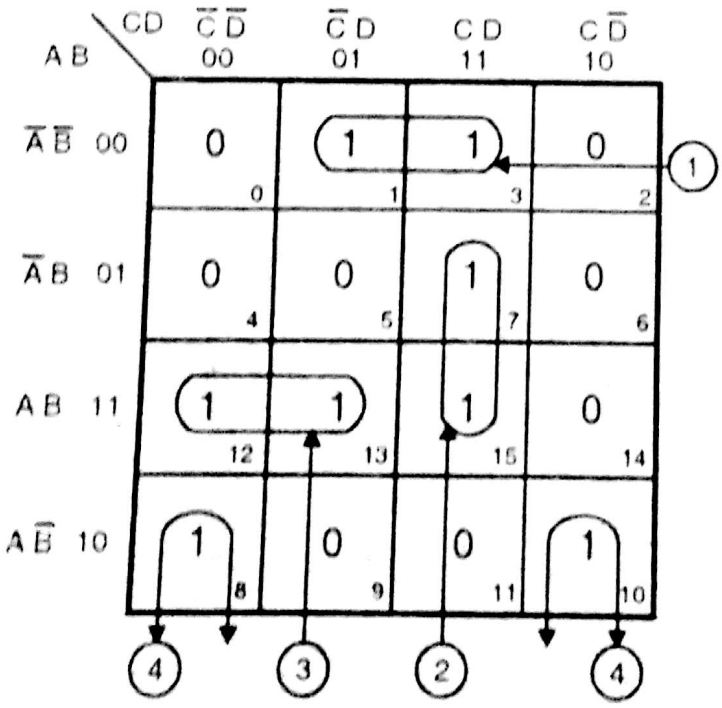

Fig. Ex. 4.10 (b)

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

**Ex. 11** : Simplify following expression using K–Map and realise using AND–OR realisation

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

**Soln.** : Procedure is same as that followed in Ex. 7




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

Fig. Ex. 4.11 (a)

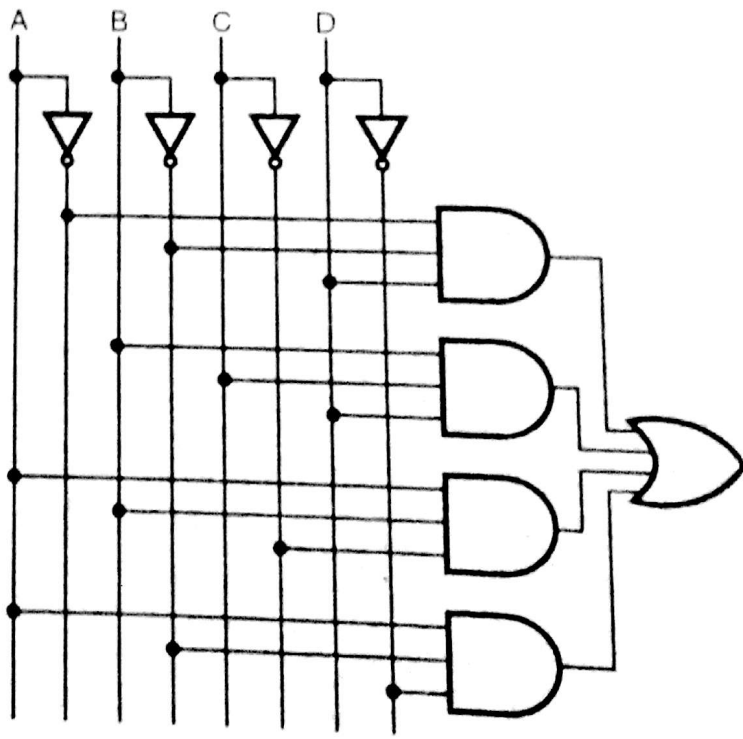


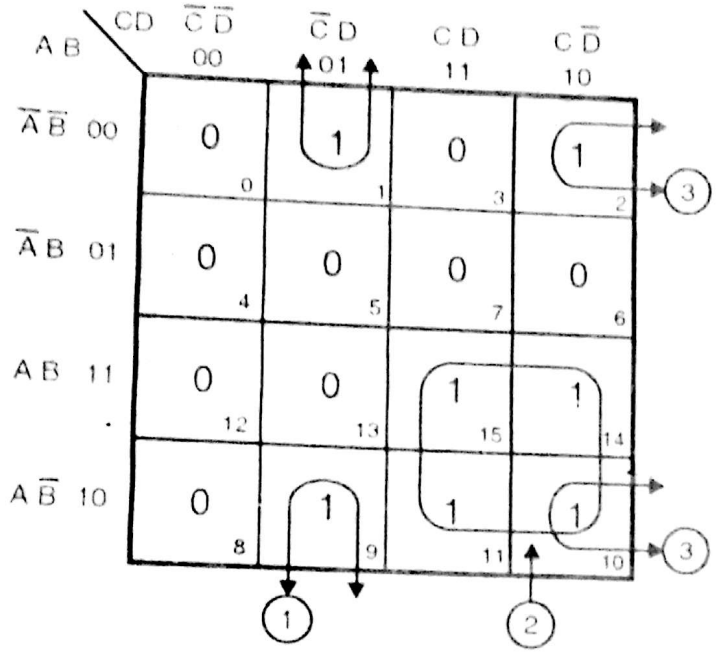

Fig. Ex. 4.11 (b)

**Ex.12 :** Simplify following expression using K–Map

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

Scanned by CamScanner

**Soln. :**




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

(1) (2) (3)

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

$$= B\,(C \oplus D) + A\,C$$

**Fig. Ex. 4.12 (a)**

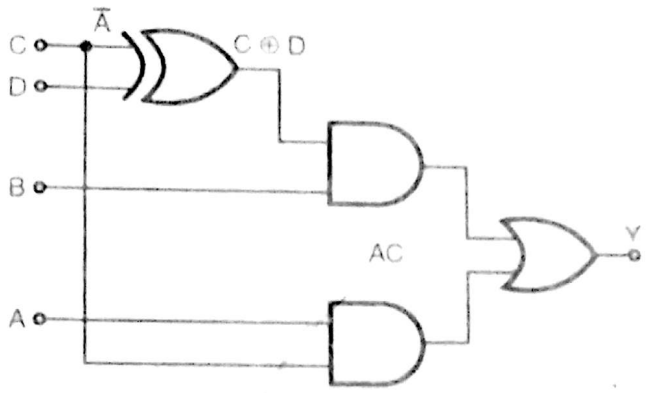
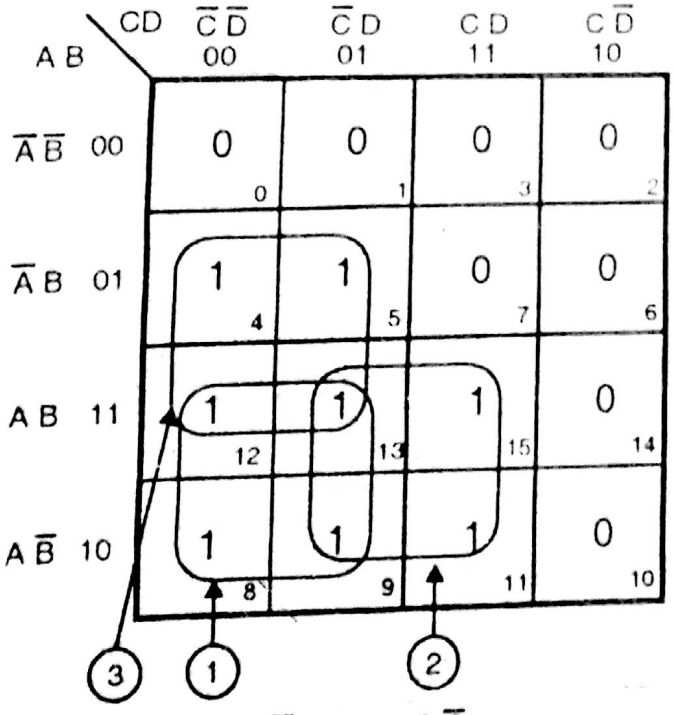



**Fig. Ex. 4.12 (b)**

**Ex. 13 :** Simplify following expression and realize using 2 input gates

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

**Soln. :**




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

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

**Fig. Ex. 4.13 (a)**

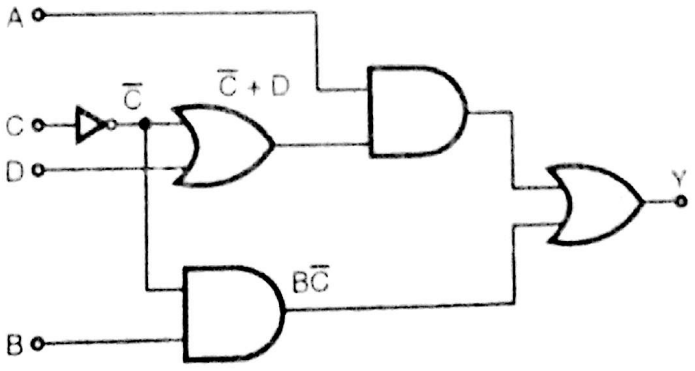



**Fig. Ex. 4.13 (b)**

Scanned by CamScanner

**Ex. 14 :** F(A, B, C, D, E) = Σm (0, 1, 7, 9, 11, 13, 15, 16, 17, 23, 25, 27). Simplify using 5 variable K–Map.

**Soln. :** (1) Draw basic skeleton for 5 variable K–Map. (Refer Map IV (a) and (b)).

(2) 'Σm' is for minterm. ∴ go on filling '1's in the boxes, whose number is specified, i.e. 0, 1, 7, 9 ... 27.

(3) Now start encircling number of '1's.

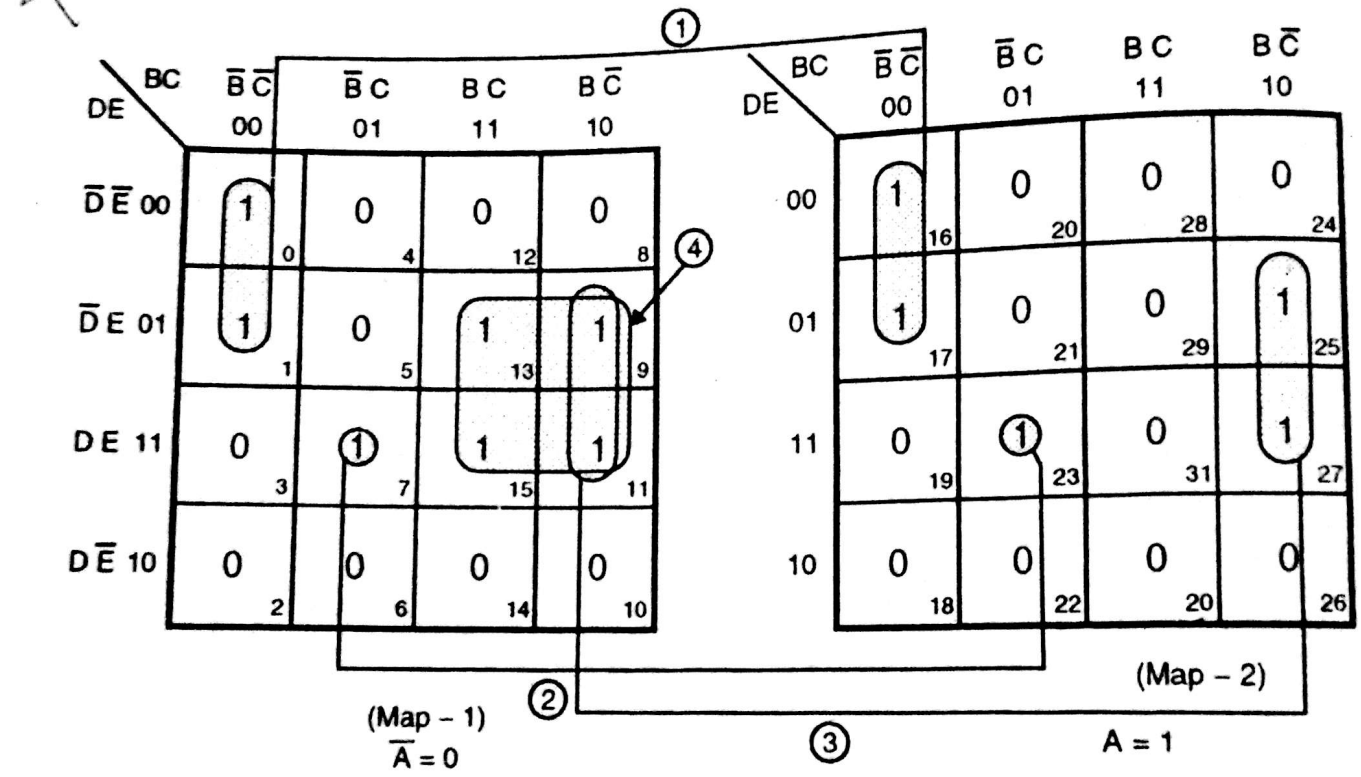



**Fig. Ex. 4.14 (a)**

As shown in Fig. Ex. 4.14 (a) we have two maps, map 1 and 2. While encircling the number of '1's, you should think of overlap of map 2 over map 1. ∴ Box 0, 1, 3, 2, 4, 5, 7, 6 and so on will be under 16, 17, 19, 18, 20, 21, 23, 22 ... and so on. Now if in the above box we have 1 and exactly below it in map 1 we have '1'. You can encircle those '1's and combine. Say for an example '1' s of box 0, 1 and 16, 17 are coming exactly below. So total it will form Quad. Now 2, 1's from map 1 and 2, 1's from map 2, ∴ there is change of maps ∴ $\bar{A}$ changes to A.

∴ While writing equation for group 1 it will be

Vertically → $\bar{D}\bar{E} + \bar{D}E \rightarrow \bar{D}$

Horizontally → $\bar{B}\bar{C}$

∴ group 1 → $\bar{B}\bar{C}\bar{D}$

For group 2, vertically → DE

horizontally → $\bar{B}C$ and as map changes A is cancelled.

∴ group 2 → $\bar{B}CDE$.

For group 3, vertically → $\bar{D}E + DE \rightarrow E$

horizontally → $B\bar{C}$, map 1 and 2 both are considered.

∴ group 3 → $B\bar{C}E$.

For group 4, vertically → $\bar{D}E + DE \rightarrow E$

Scanned by CamScanner

horizontally $\rightarrow BC + B\bar{C} \rightarrow B$

and as there is no map change ever, grouping is in map–1 only.

$\therefore$ group 4 $\rightarrow \bar{A}BE$

$\therefore$ output $Y = \underset{(1)}{\bar{B}\bar{C}\bar{D}} + \underset{(2)}{\bar{B}CDE} + \underset{(3)}{B\bar{C}E} + \underset{(4)}{\bar{A}BE} = \bar{B}(\bar{C}\bar{D} + CDE) + BE(\bar{A} + \bar{C})$

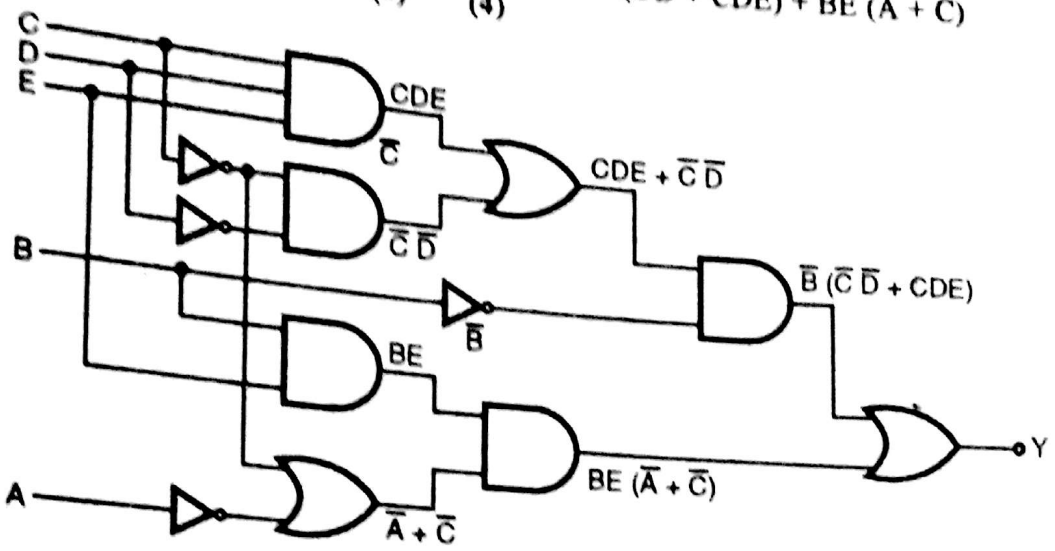
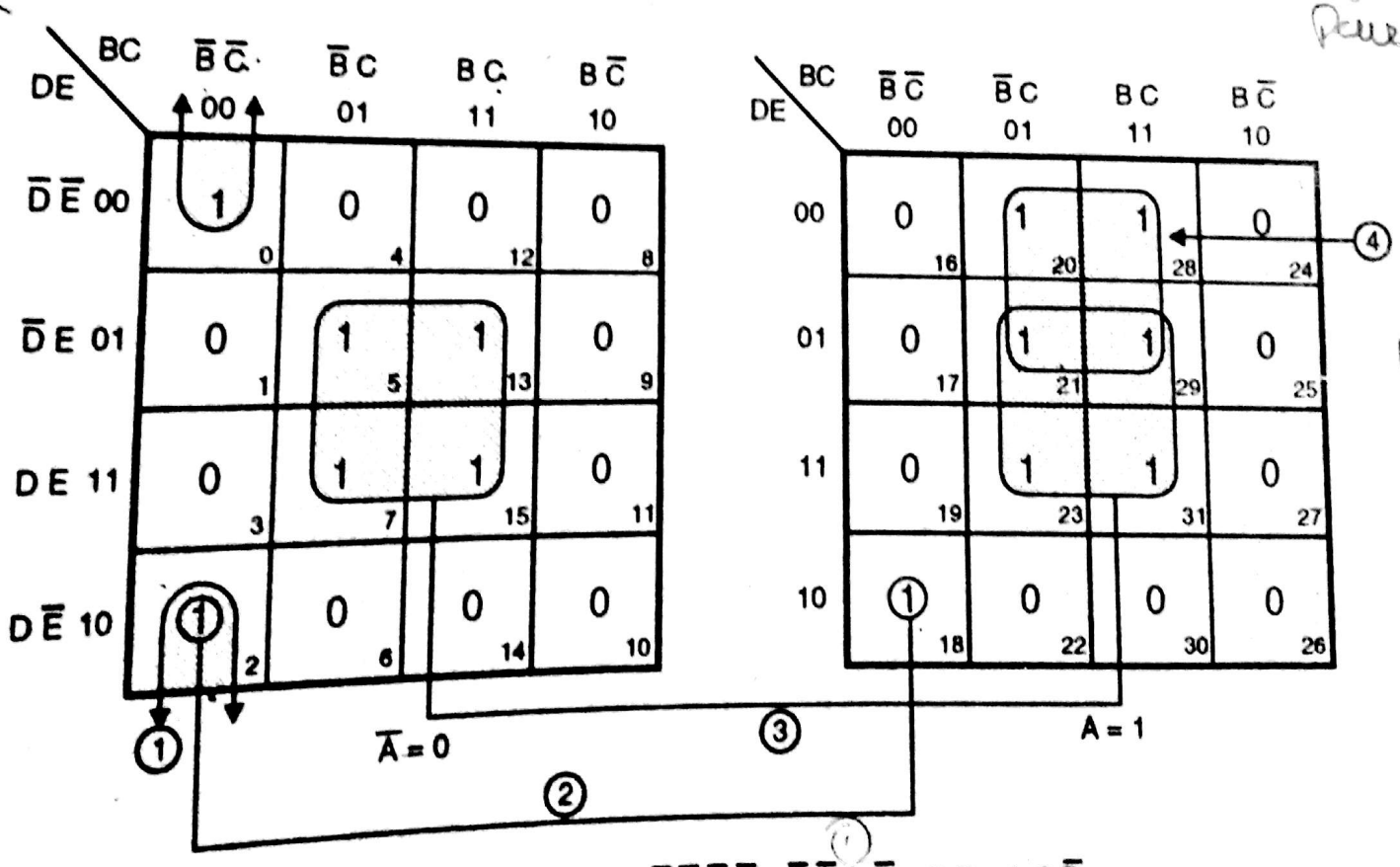



**Fig. Ex. 4.14 (b)**

**Ex. 15 :** F (A, B, C, D, E) = $\Sigma m$ (0, 2, 5, 7, 13, 15, 18, 20, 21, 23, 28, 29, 31). Simplify this using K–Map.

**Soln. :** K–Map will be five variables. Procedure is as shown in Ex. 14.




$$F(A, B, C, D, E) = \bar{A}\bar{B}\bar{C}\bar{E} + \bar{B}\bar{C}D\bar{E} + CE + AC\bar{D}$$
$$= \bar{B}\bar{C}\bar{E}(\bar{A} + D) + C(E + A\bar{D})$$

**Fig. Ex. 4.15 (a)**

Scanned by CamScanner

$$F(A, B, C, D, E) = \underset{(1)}{\bar{A}\bar{B}\bar{C}\bar{E}} + \underset{(2)}{\bar{B}\bar{C}\bar{D}\bar{E}} + \underset{(3)}{CE} + \underset{(4)}{A\bar{C}\bar{D}}$$

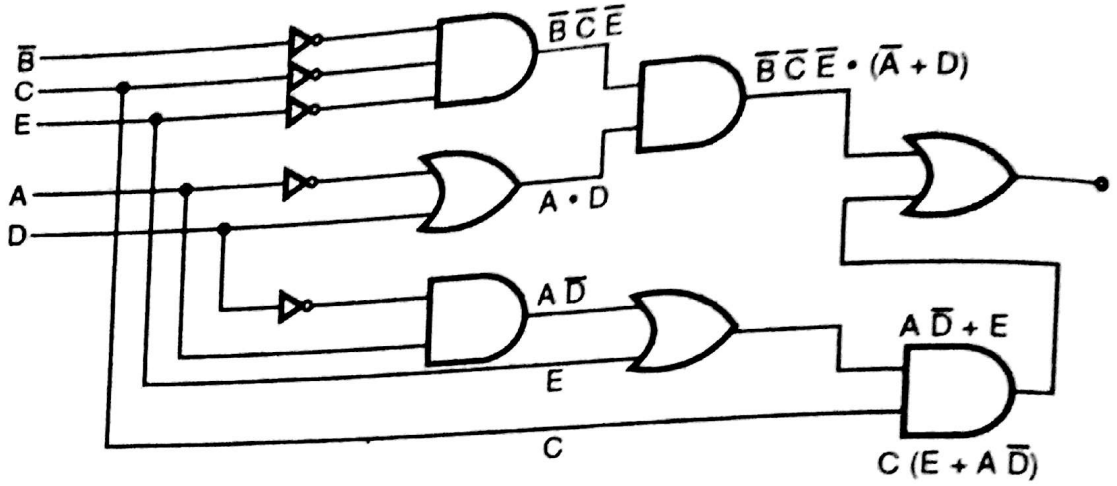



**Fig. Ex. 4.15 (b)**

**Ex. 16 :** F(A, B, C, D) = Σm (1, 5, 6, 7, 11, 12, 13, 15).
Simplify this using K–Map.

**Soln. :** K–Map is four variables, Procedure is as shown in Ex. 7.

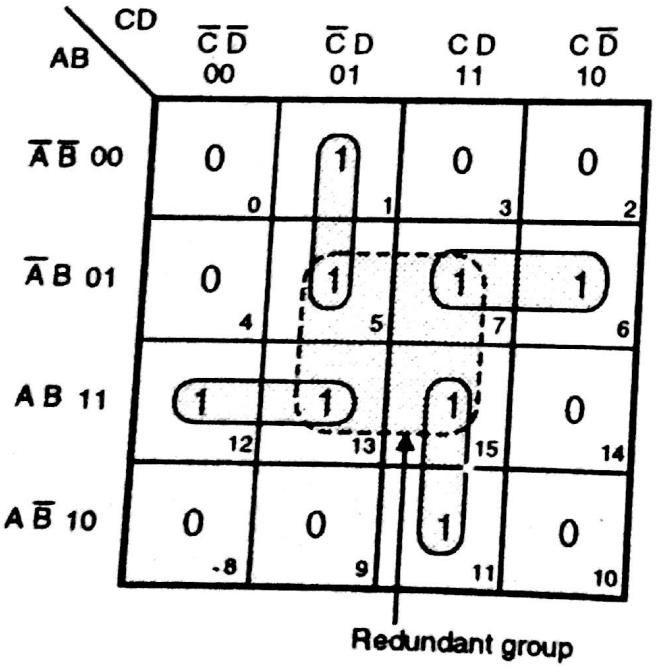



**Fig. Ex. 4.16 (a)**

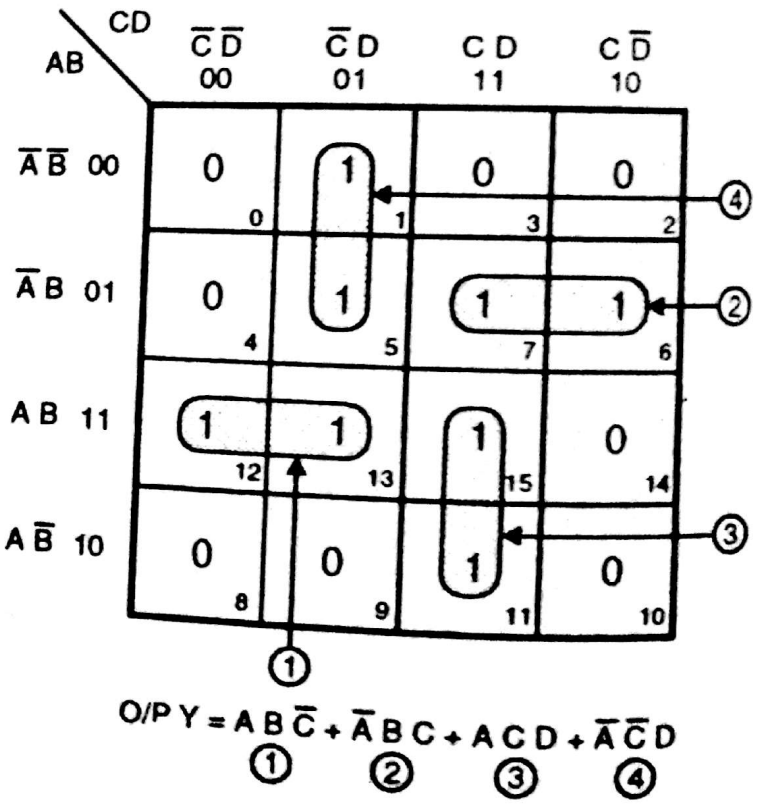



**Fig. Ex. 4.16 (b)**

Scanned by CamScanner

As shown in Fig. Ex. 4.16 (a) first you will try to cover maximum number of 1's. ∴ you will select group 5, 7, 13, 15. After that you will make group of 1, 5 – 7, 6 – 15, 11 – 12, 13 i.e. four groups.

But the moment you make these four groups all 1's are covered. Refer Fig. Ex. 4.16 (b). Here four groups are shown.

∴ Encircling 5, 7, 13, 15, i.e. forming quad becomes ***Redundant group*** which is not required

∴ Finally Fig. Ex. 4.16 (b) is correct grouping.

**Ex. 17 :** F(A, B, C, D) = Σm (0, 1, 2, 5, 13, 15).

**Soln. :** Draw four variable K–Map and fill the boxes.

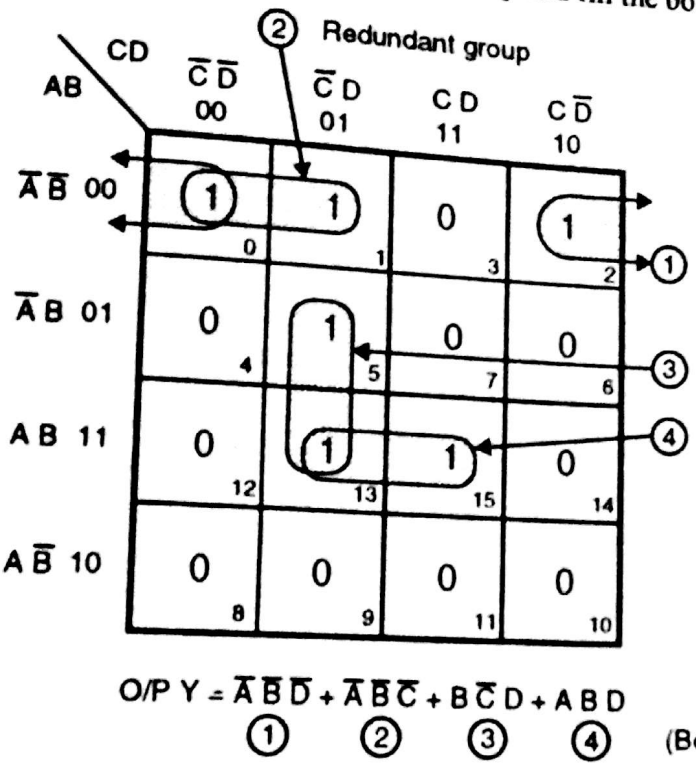



**Fig. Ex. 4.17 (a)**

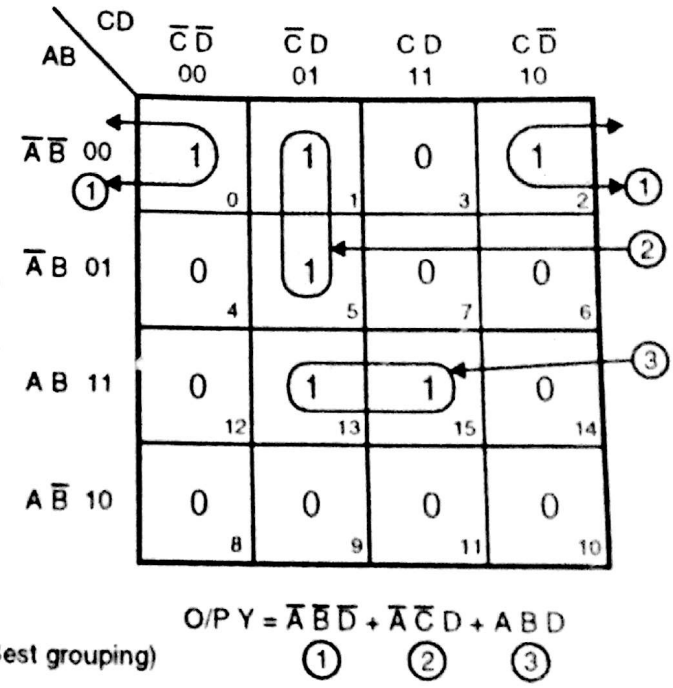



**Fig. Ex. 4.17 (b)**

As shown in Fig. Ex. 4.17 (a), you will start grouping 0,1 – 5,13 – 13,15 – 0,2. Here you are going to get four terms. But you can still simplify this by grouping 1's as shown in Fig. Ex. 4.17 (b). In that we are going to get only 3 terms, ∴ it is the best encircling.

From Fig. Ex. 4.17 (a) → output $Y = \bar{A}\bar{B}\bar{D} + \bar{A}\bar{B}\bar{C} + B\bar{C}D + ABD$ ...(x)

(b) → output $Y = \bar{A}\bar{B}\bar{D} + \bar{A}\bar{C}D + ABD$ ...(y)

We will start simplifying Equation (x) and come down to Equation (y).

$Y = \bar{A}\bar{B}D + \bar{A}\bar{B}\bar{C} + B\bar{C}D + ABD.$

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

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

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

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

$= \bar{A}\bar{B}D + \bar{A}\bar{C}D + ABD.$ → Now further simplification is not possible.

**Note :** Therefore using K–Map also you have to obtain Boolean expression in such a way that further simplification is literally impossible.

Scanned by CamScanner

**Ex. 18 :** Reduce following expression using 5 variable K–Map using SOP form :

$F(A, B, C, D, E) = \Sigma m (0, 1, 2, 3, 10, 12, 13, 14, 20, 21, 22, 23, 26, 27, 28, 29, 30, 31)$

**Soln. :** Procedure is same as that followed in Ex. 14.

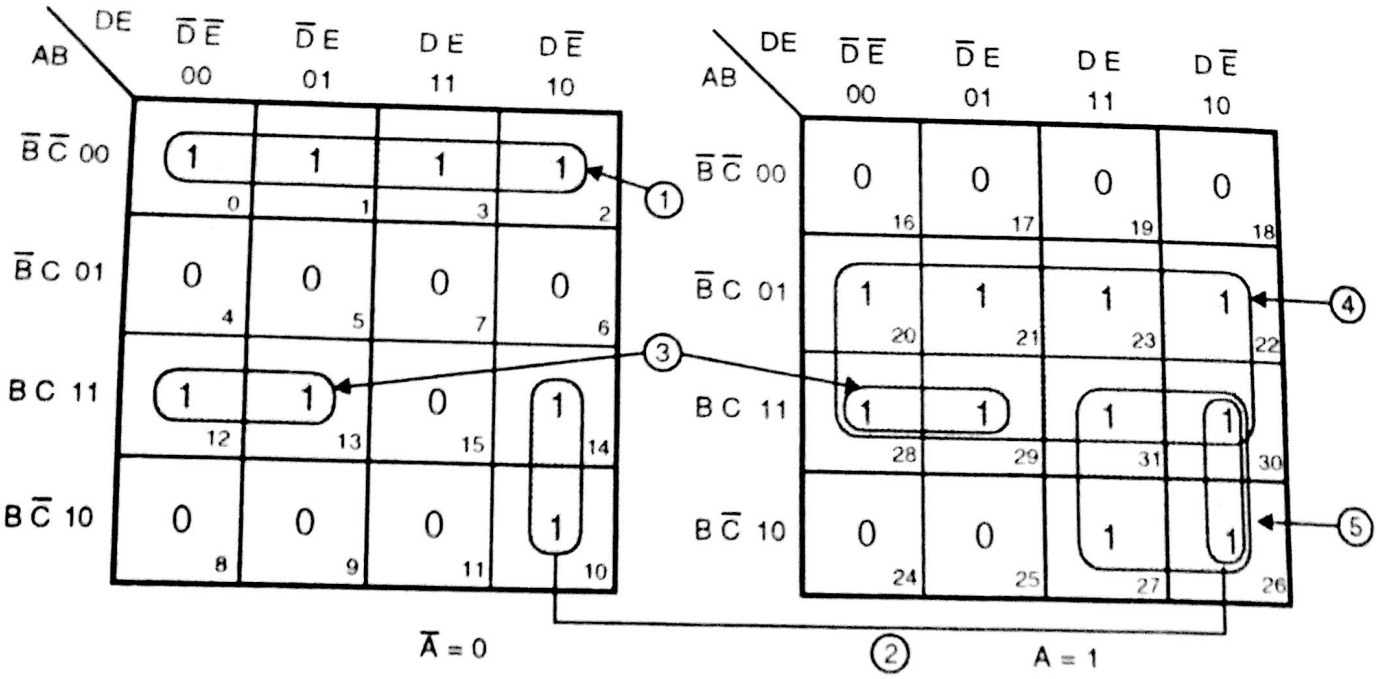



**Fig. Ex. 4.18 (a)**

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

## 4.6.4 Don't Care Condition :

In logic circuits, for some cases it may happen that for certain input conditions no specific output level is defined i.e. HIGH or LOW. This case may be present because these input conditions will never occur. In short, there will be certain combinations of input levels where we ***don't care*** whether the output is HIGH or LOW.

Normally don't care states are specified by 'd' for example say $f(A, B, C) = \Sigma m (0, 1, 3) + d (5, 6)$. Here '$\Sigma$m' is as usual minterm, 'd' → don't care condition. In above function states 5 and 6 are don't care.

Now you will ask how to represent don't care in K–Map ? Usually in K–Map we put 'x' (don't care !) sign in the box whose number is specified by the function under 'd'.

As far as circuit design of such function is concerned you (the circuit designer) is free to make the output of ***don't care*** condition either '0' or '1', in order to produce simplest output expression. Normally when you write SOP form, you normally force don't care output state to 1 and for POS form, you normally force don't care output state to 0.

Let's take one example to understand concept clearly.

---

**Ex. 19 :** $F(A, B, C, D) = \Sigma m (1, 4, 8, 12, 13, 15) + d (3, 14)$, simplify.

**Soln. : For this problem we have drawn two K–Map. The first one doesn't encircles 'X' (don't care condition) and other one encircles the 'X'. Fig. Ex. 4.19 (b) and (d) are the hardware design.**

**Fig. Ex. 4.19 (b) (i) shows logic circuit for output Y, we achieve from Fig. Ex. 4.19 (a) (i). In that we require 9 AND gates and 3 OR gates In Fig Ex. 4 19 (i) (d) logic circuit for output Y derived from Fig.**

Scanned by CamScanner

Ex. 4.19 (i) (c), is drawn. In that we required 7 AND gates and 3 OR gates. 2 AND gates are saved if we cover 'X' (don't care conditions).

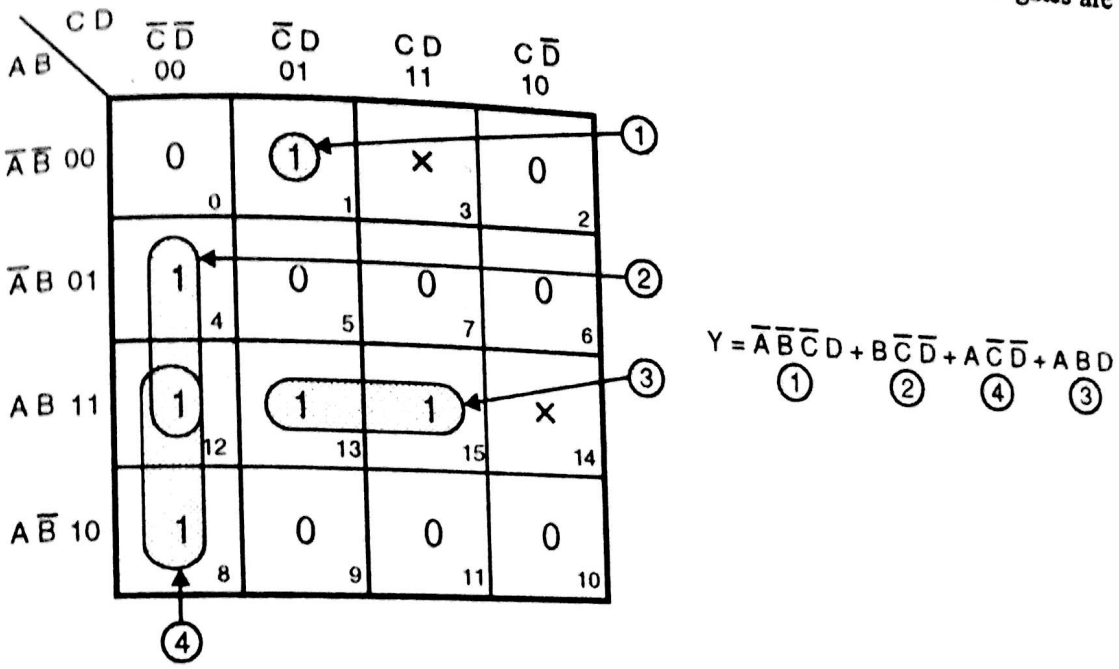



**Fig. Ex. 4.19 (i) (a)**

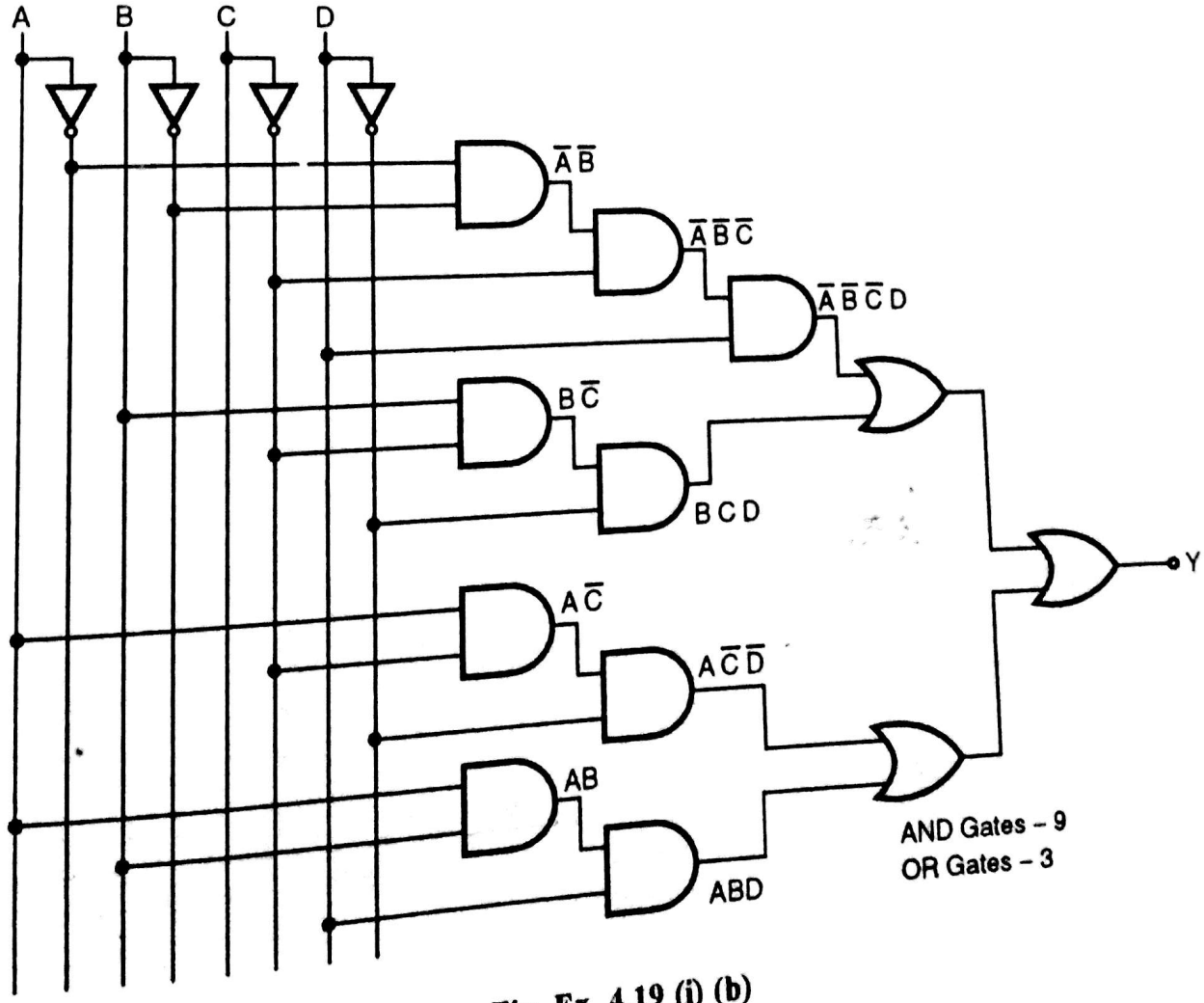



**Fig. Ex. 4.19 (i) (b)**

Scanned by CamScanner

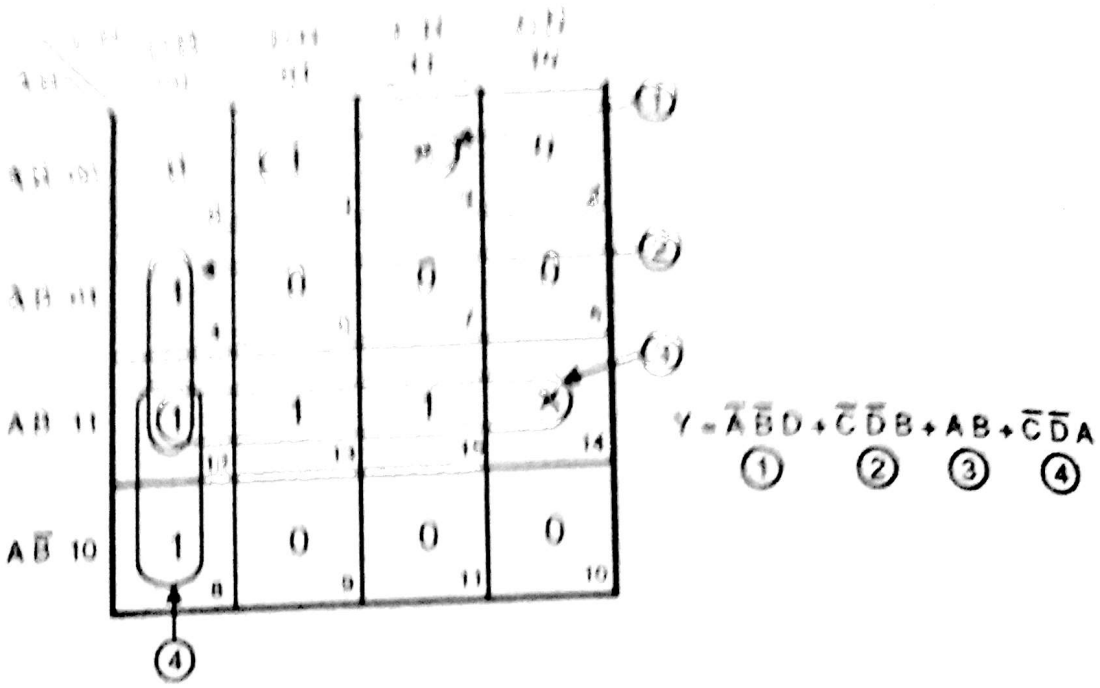



Fig. Ex. 4.19 (i) (c)

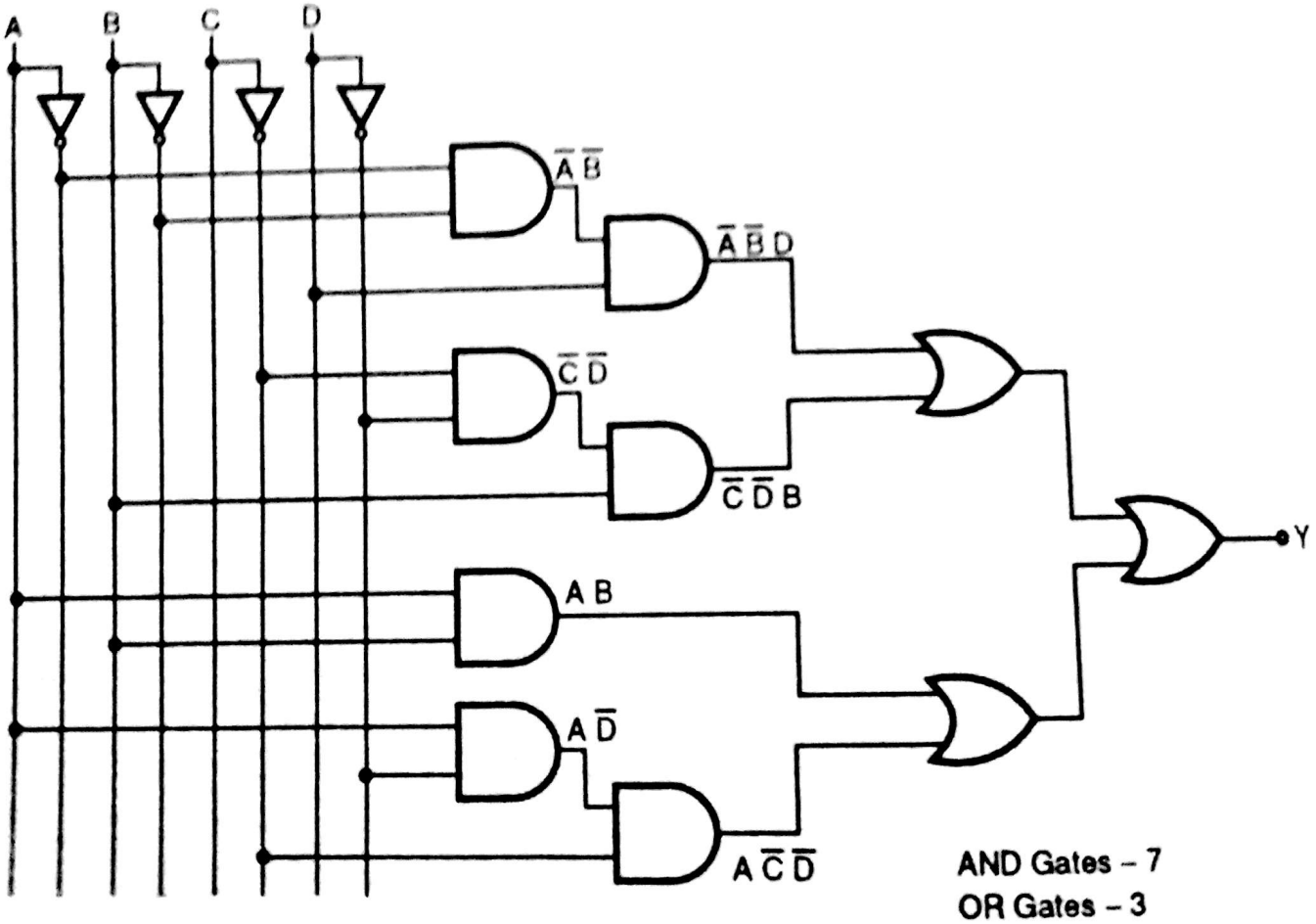



**Fig. Ex. 4.19 (i) (d)**

Here, I would like to tell you one thing that it is ***not*** always necessary that once you encircle one 'X', corresponding 'X' SHOULD BE ENCIRCLED. It is purely your choice that how many 'X's you convert to 1, by encircling in minterm Or 1's group. Let's see one example to understand the concept.

---

Find reduced SOP form for following equation :

$F(A, B, C, D) = \Sigma m(1, 3, 7, 11, 15) + \Sigma d(0, 2, 5, 8, 14)$.

Scanned by CamScanner

**Soln.** : Draw K–Map for 4 variable, and fill 1's, 0's and X's.

As shown in Fig. Ex. 4.19 (ii) (a) we have two groups.

group 1 covers → 0, 1, 3, 2. group 2 covers → 3, 7, 5, 11.

In that, 0, 2 are 'X' (don't care condition); but by encircling that in group 1. We have forced 'X' to logical '1'.

After forming group 1/2, we have encircled all 1's. ∴ One need not encircle 'X' in 5, 8, 14 box of K–Map.

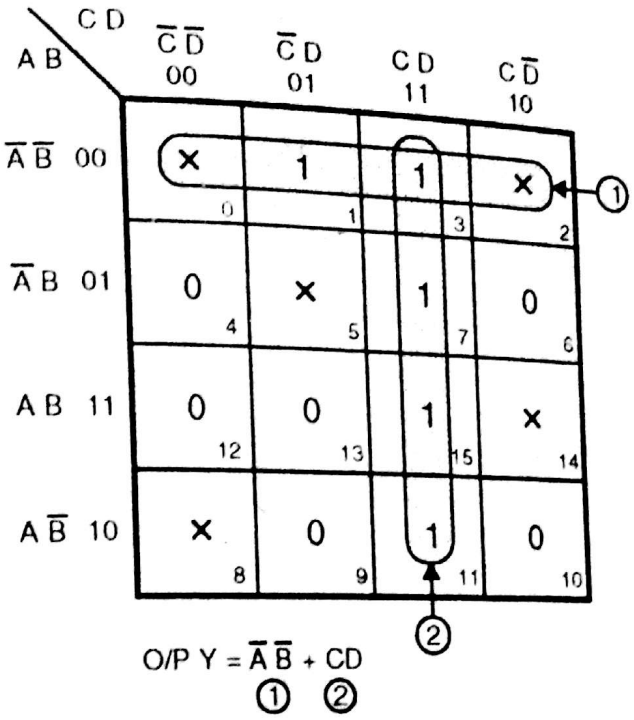



**Fig. Ex. 4.19 (ii) (a)**

**Ex. 20 :** Simplify following :

$F(A, B, C, D) = \Sigma m(0, 1, 5, 9, 13, 14, 15) + d(3, 4, 7, 10, 11)$.

**Soln. :** Draw 4 variable K–Map and fill all the boxes.

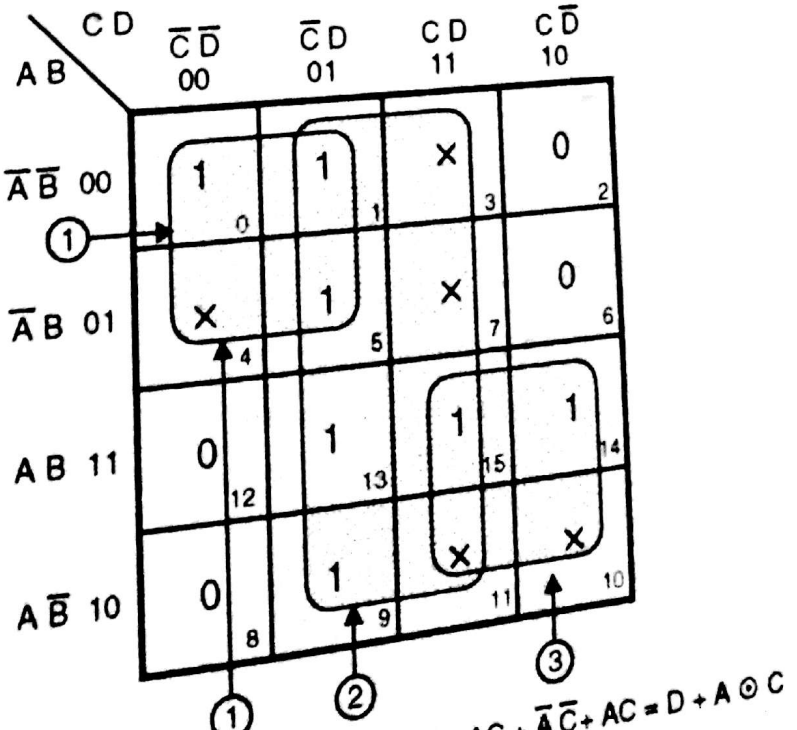
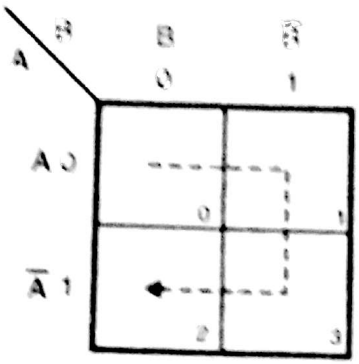



**Fig. Ex. 4.20 (a)**

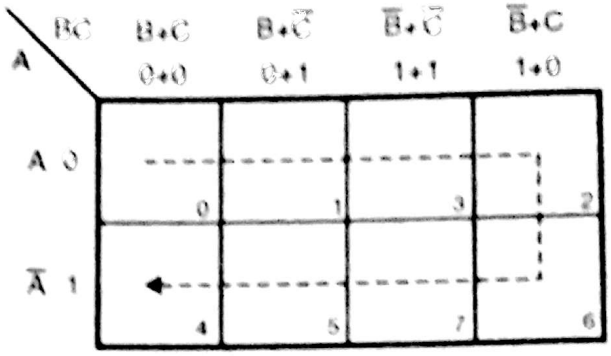
Scanned by CamScanner

## 4.7 K–Map for POS Form :

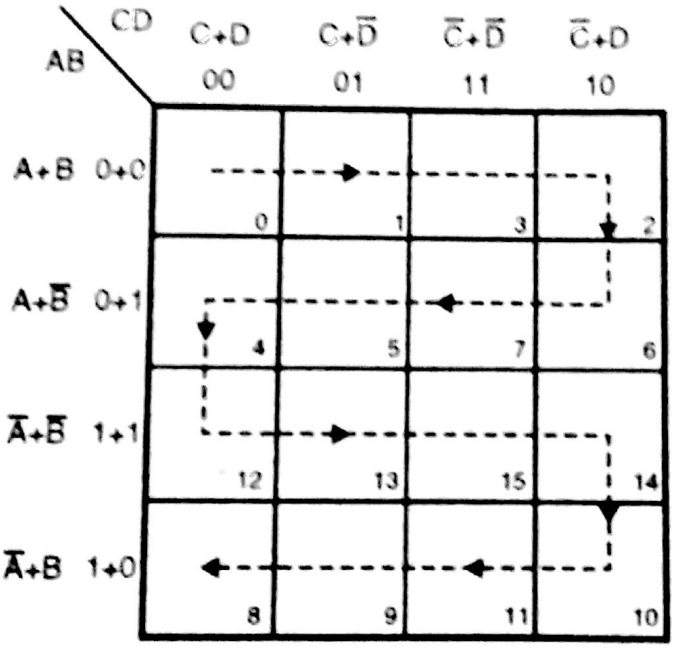
We have solved different examples where we encircled '1' i.e. we have taken into account minterm. Finally we will solve some examples related to maxterm. The basic K-map skeleton to be used for maxterm is as shown in Fig. 4.4




(a) 2 variable K-map




(b) 3 variable K-map




(c) 4 variable K–map

**Fig. 4.4**

If you observe K–Map correctly and compare it with Map I, II and III. The basic difference you will find that,

(1) In minterm AND operation was used. But here OR operation is used.

(2) In minterm variables are adjusted in such a way that final answer is '1' for example 0 0 0 0 (A B C D)

give minterm $\overline{A}\overline{B}\overline{C}\overline{D}$, i.e. $\overline{0} \cdot \overline{0} \cdot \overline{0} \cdot \overline{0} = 1$.

But in maxterm sum should give you logic '0', for example ABCD = 0000 then maxterm will be A + B + C + D, i.e. 0 + 0 + 0 + 0 = 0. If ABCD = 0010 then maxterm will be

Scanned by CamScanner

$A + B + \bar{C} + D = 0 + 0 + \bar{1} + 0 = 0.$

(3) Minterm K–Map gives SOP form. Maxterm K–Map gives POS form.

(4) Now in maxterm K–map you have to encircle '0', not '1's

(5) Rules for encircling '0's are same as that of 1's. You will be more clear with examples.

**Ex. 21 :** Find expressions for following K–Map :

(i)

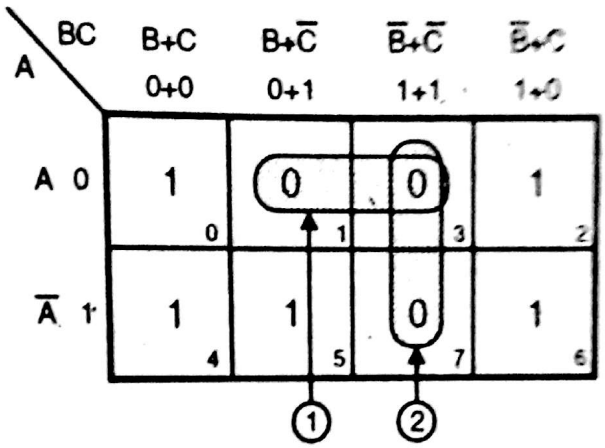



**Fig. Ex. 4.21 (i)**

**Soln. :** Grouping 1 covers '0's in 1, 3. Grouping 2 covers '0's in 3, 7.
Neglect those variables which are changing through vertical or horizontal sum.

$\therefore$ For group 1 $\rightarrow$ $(A + \bar{C})$; group 2 $\rightarrow$ $(\bar{B} + \bar{C})$

$\therefore$ output $Y = (A + \bar{C}) \cdot (\bar{B} + \bar{C})$ (POS form)
(1) (2)

(ii)

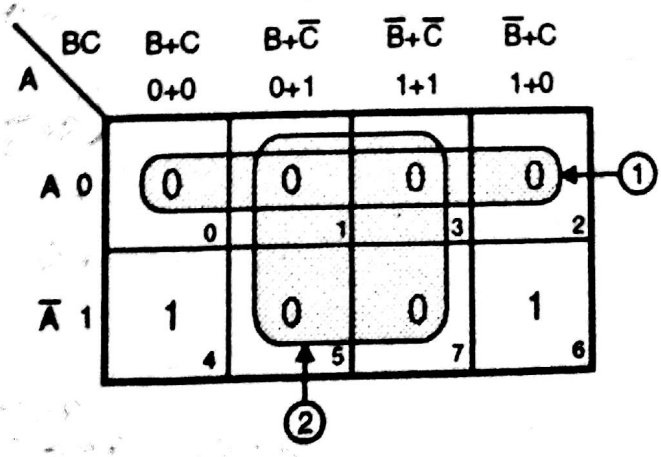



**Fig. Ex. 4.21 (ii)**

**Soln. :** Group 1 $\rightarrow$ A Group 2 $\rightarrow \bar{C}$ $\therefore$ output $Y = (A) \cdot (\bar{C})$

Scanned by CamScanner

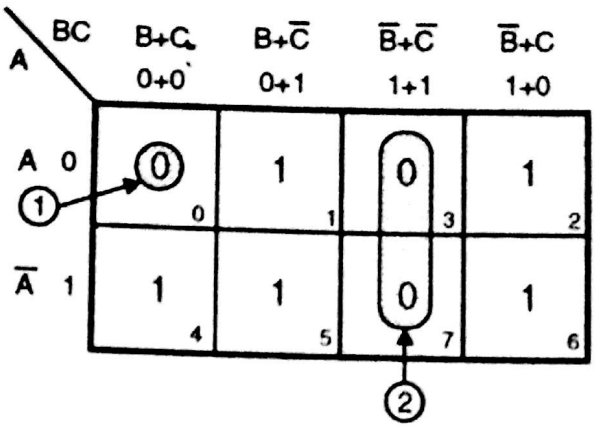



**Fig. Ex. 4.21 (iii)**

**Soln.** : Output $Y = (A + B + C) \cdot \left(\bar{B} + \bar{C}\right)$

(1) (2)

(iv)

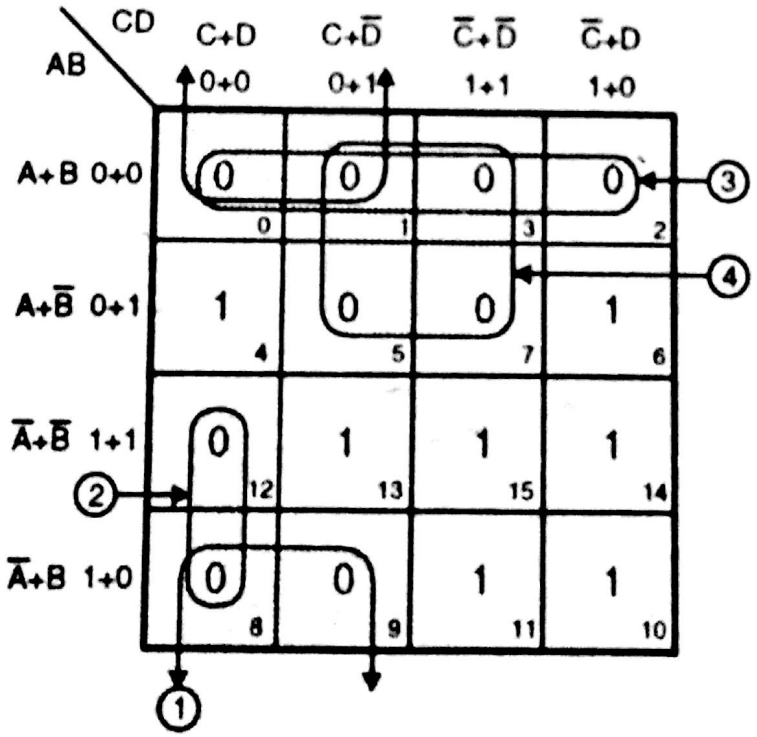



**Fig. Ex. 4.21 (iv)**

**Soln.** : output $Y = (B + C)\left(\bar{A} + C + D\right)(A + B)\left(A + \bar{D}\right)$

(1) (2) (3) (4)

Scanned by CamScanner

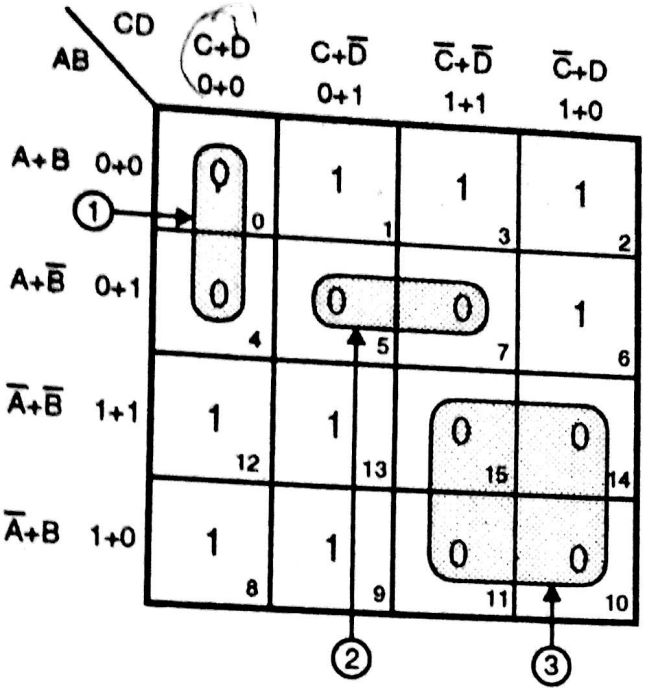



Fig. Ex. 4.21 (v)

**Soln. :** Output $Y = (A + C + D)\underset{(1)}{} \underbrace{(A + \bar{B} + \bar{D})}_{(2)} \underbrace{(\bar{A} + \bar{C})}_{(3)}$

**(vi)** Refer Fig. Ex. 4.21 (vi)

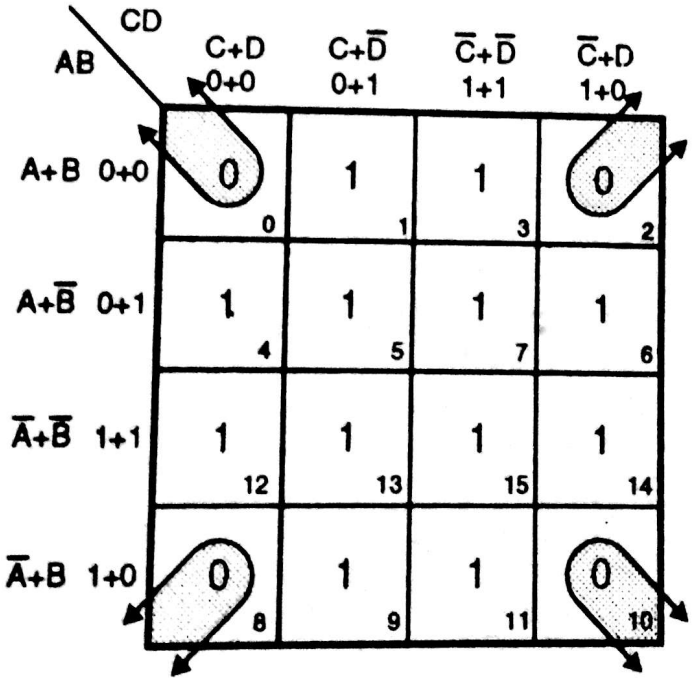



Fig. Ex. 4.21 (vi)

**Soln. : Output Y = (B + D).**

**(vii)** Refer Fig. Ex. 4.21 (vii).

Scanned by CamScanner

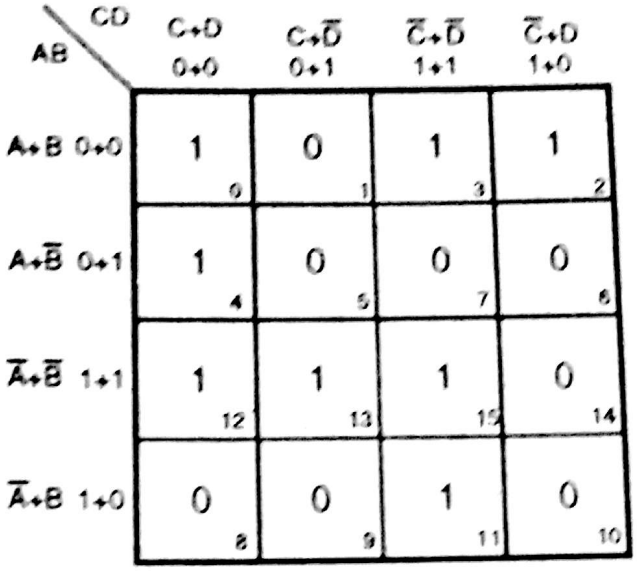



**Fig. Ex. 4.21 (vii)**

**Soln. :** In this case 2 way of encircling is shown. Both are correct and both gives minimum hardware.

We have seen some examples where direct encircling of the '0's was given to you. Now you will be provided with the function expression, and you have to give minimised Boolean expression in POS form.

You are already aware with the fact that 'πM' represents maxterm. Maxterm provides output 'Y' equal to zero. The steps you have to follow in Fig. Ex. 4.21.

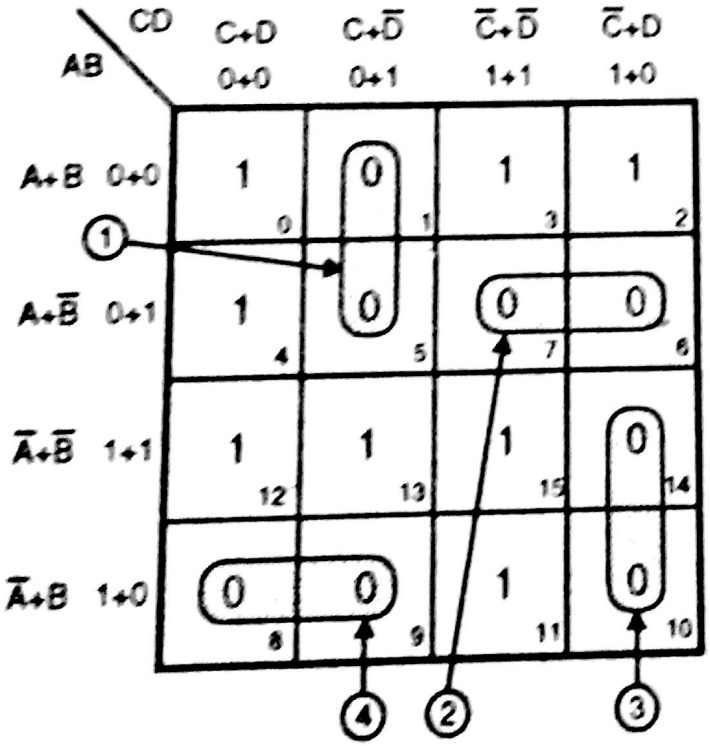



**Fig. Ex. 4.21 (vii) (a)**

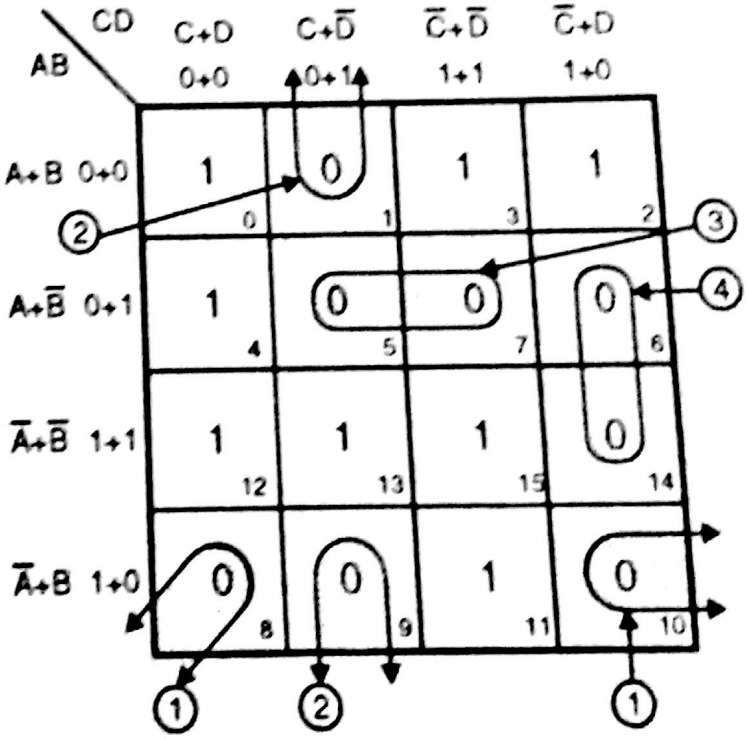



**Fig. Ex. 4.21 (vii) (b)**

$$\text{Output } Y_{(\text{Fig. Ex. 4.21 (vii) (a)})} = (A + C + \bar{D})\,(A + \bar{B} + \bar{C})\,(\bar{A} + \bar{C} + D)\,(\bar{A} + B + C)$$
$$(1) \qquad (2) \qquad (3) \qquad (4)$$

$$\text{Output } Y_{(\text{Fig. Ex. 4.21 (vii) (b)})} = (\bar{A} + B + D)\,(B + C + \bar{D})\,(A + \bar{B} + \bar{D})\,(\bar{B} + \bar{C} + D)$$
$$(1) \qquad (2) \qquad (3) \qquad (4)$$

Scanned by CamScanner

| | | |
|---|---|---|
| **Step I** | : | Read maxterm expression. |
| **Step II** | : | From the number of variables given decide the 'n' variable K–Map you are going to require. Draw basic skeleton of it. |
| **Step III** | : | Now go on filling '0' in the boxes specified by the function. |
| **Step IV** | : | Remaining all boxes should be filled up with logic '1'. |
| **Step V** | : | Now go on grouping '0's. |

**Ex. 22 :** Find expression for following and implement using logic gates :

(i) $F(A, B, C, D) = \pi M(0, 3, 4, 7)$

**Soln. :**

(1) The example is for maxterm.

(2) No. of variables are three.

$\therefore$ 3 variable K–Map is required.

(3) Draw basic skeleton of K–Map for 3 variable K–Map.

(4) Now go on filling '0' in the boxes specified in the example i.e. 0, 3, 4 and 7.

(5) Remaining all boxes should be filled by '1'.

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

(1) (2)

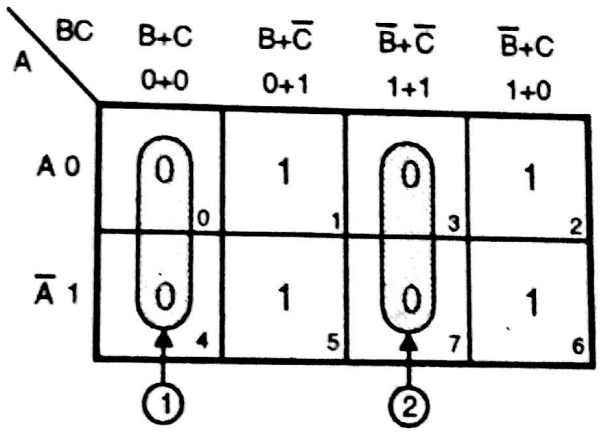



**Fig. Ex. 4.22 (i)**

(ii) $F(A, B, C) = \pi m(0, 1, 2, 4, 5)$

$Y = (B)\quad (A + C)$

(1) (2)

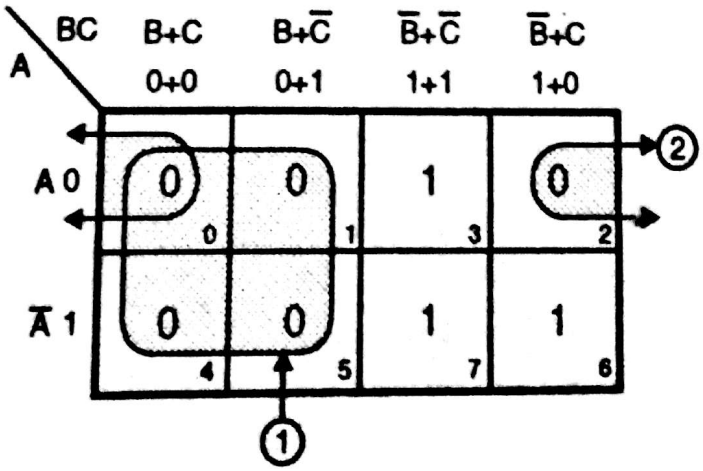
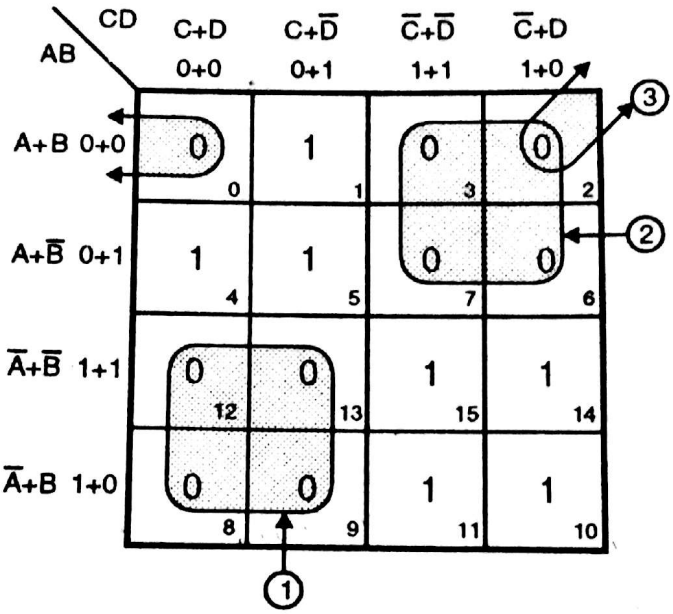


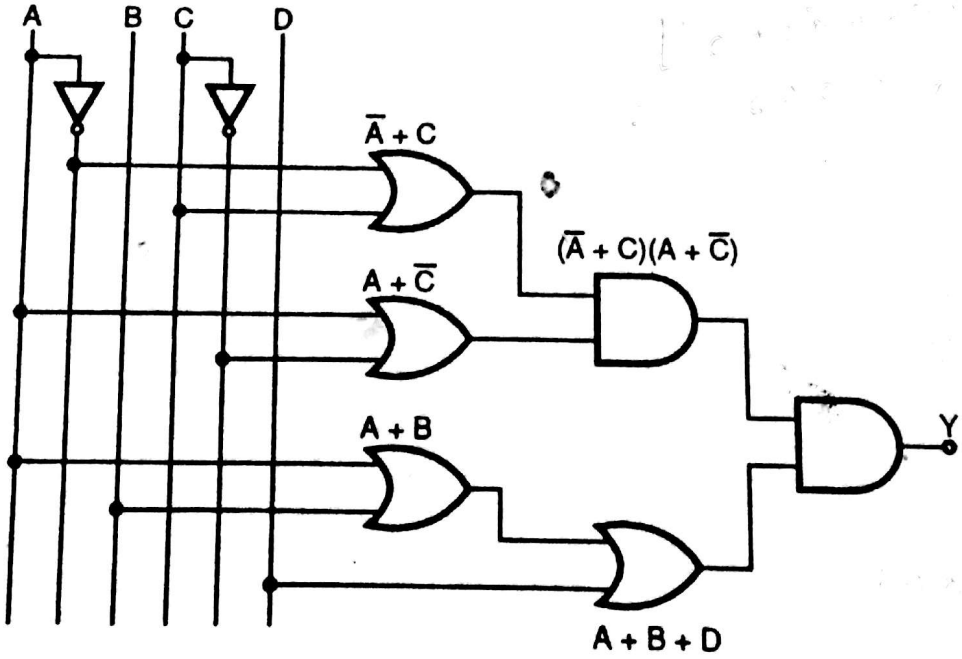

**Fig. Ex. 4.22 (ii)**

Scanned by CamScanner

(iii) F (A, B, C, D) = π M (0, 2, 3, 6, 7, 8, 9, 12, 13).




(a)




(b)

**Fig. Ex. 4.22 (iii)**

**Soln. :** $Y = (\bar{A} + C)\ (A + \bar{C})\ (A + B + D)$

(1) (2) (3)

v) F (A, B, C) = πM (2, 3, 4, 5, 6, 7)

Scanned by CamScanner

**Soln. :**

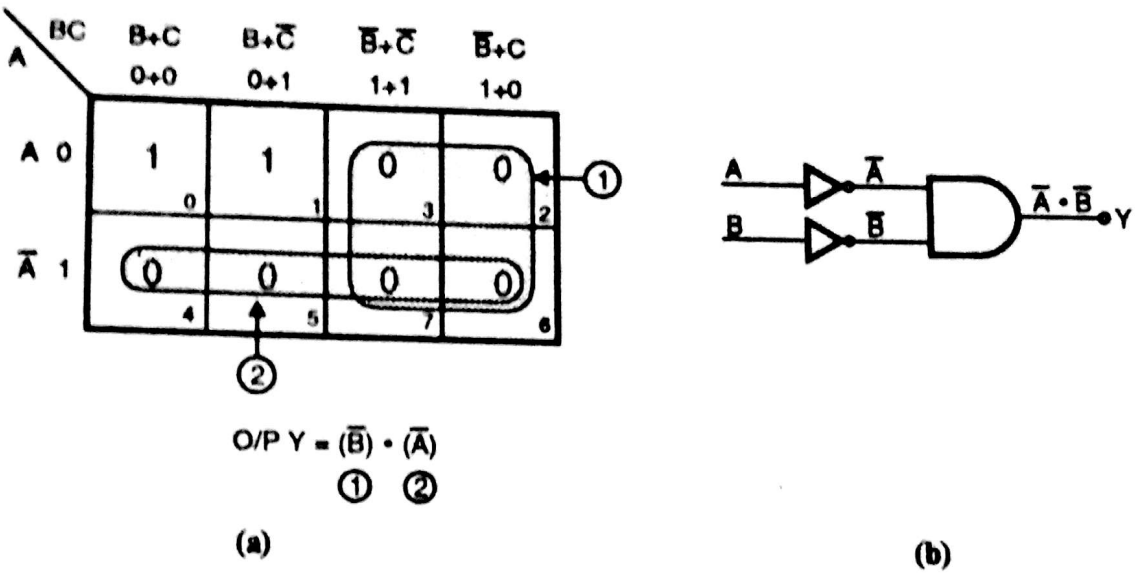



**Fig. Ex. 4.22 (iv)**

(v) F (A, B, C, D) = πM (0, 2, 4, 5, 7, 8, 9, 10, 13, 14, 15)

**Soln. :**

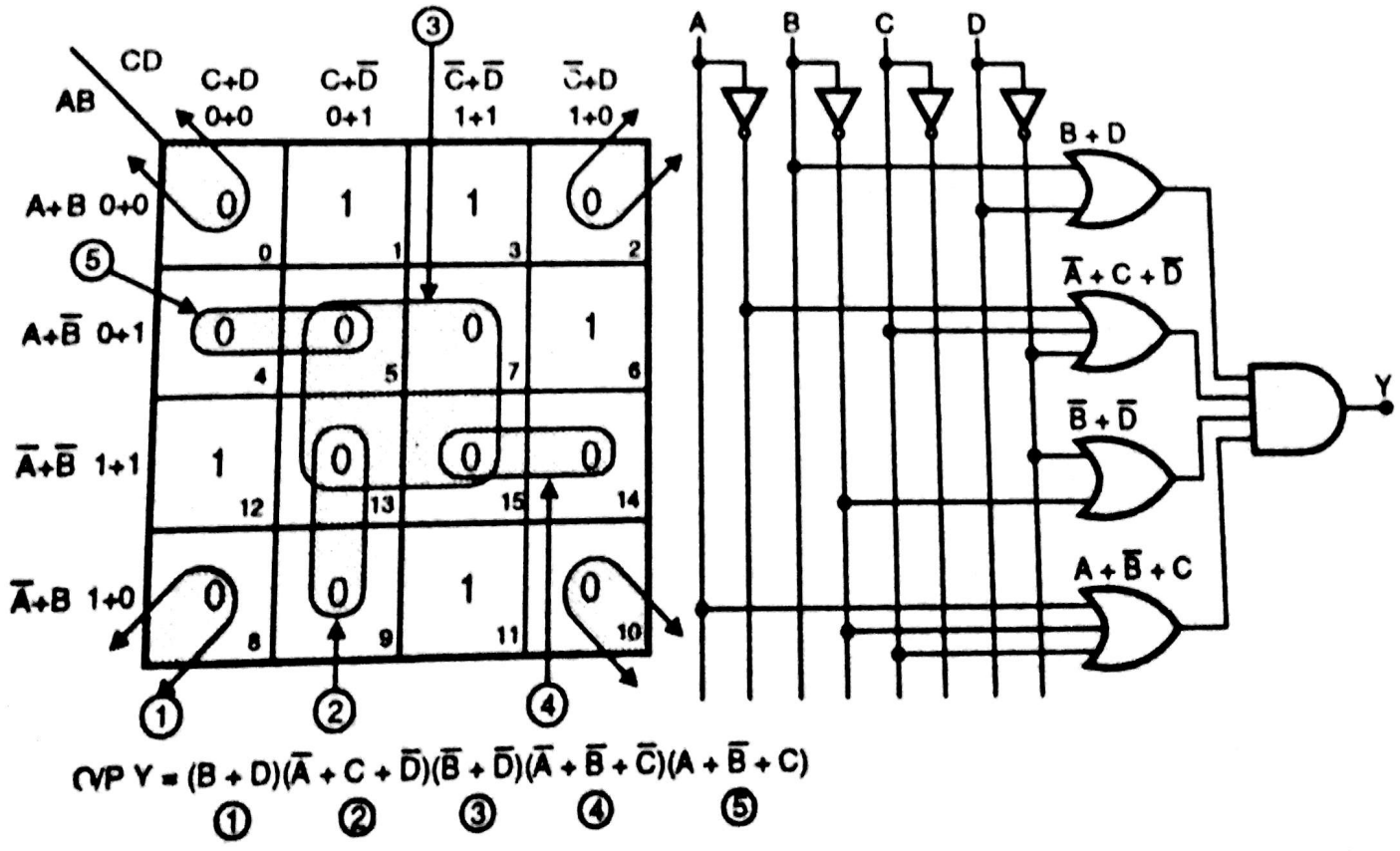



**Fig. Ex. 4.22 (v)**

(vi) F (A, B, C, D, E) = πM (2, 3, 5, 8, 9, 10, 11, 17, 18, 19, 21, 24, 25, 26, 27, 29)

Scanned by CamScanner

**Soln. :**

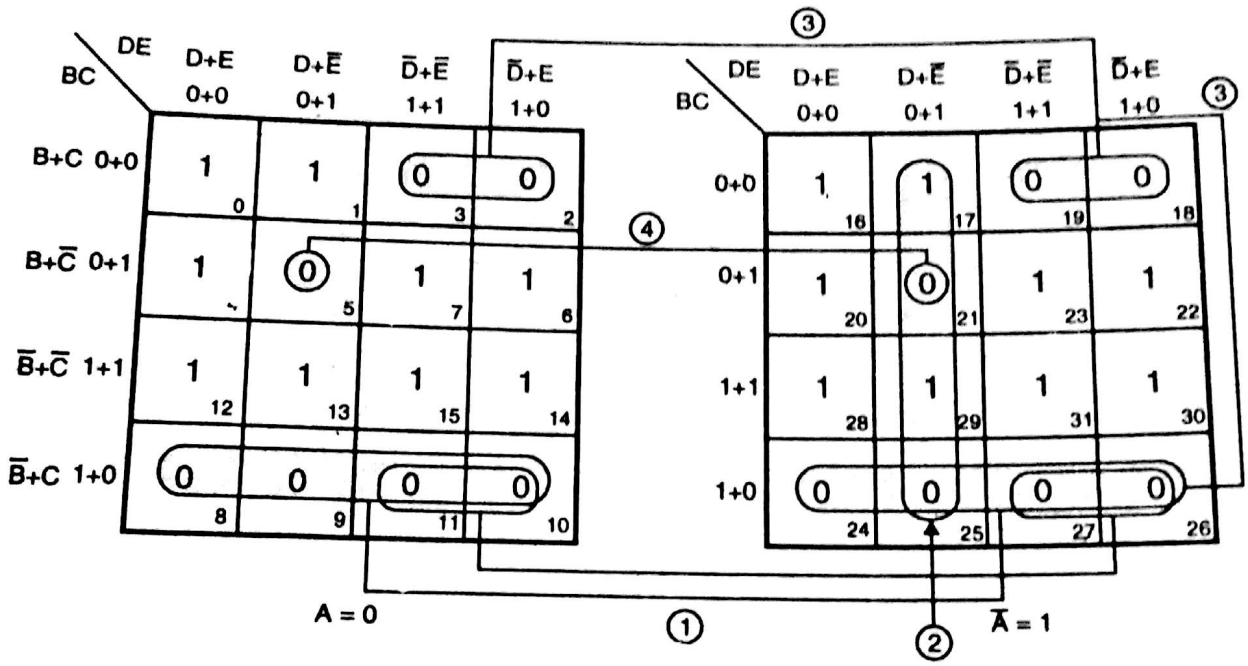
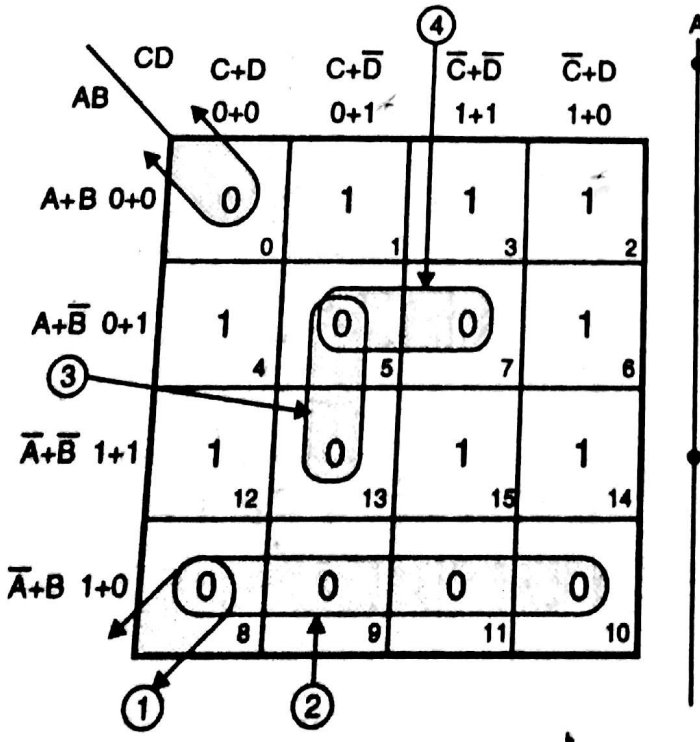


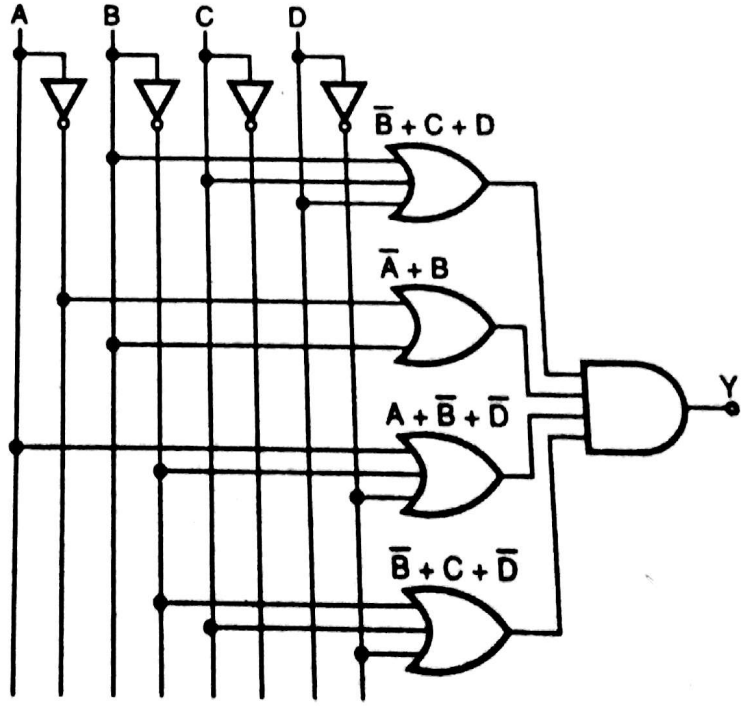

**Fig. Ex. 4.22 (vi)**

Output $Y = (\bar{B} + C) \cdot (\bar{A} + D + \bar{E})\ (C + \bar{D})\ (B + \bar{C} + D + \bar{E})$
(1) (2) (3) (4)

**(vii)** $F(A, B, C, D) = \pi M\ (0, 5, 7, 8, 9, 10, 11, 13)$

**Soln. :**



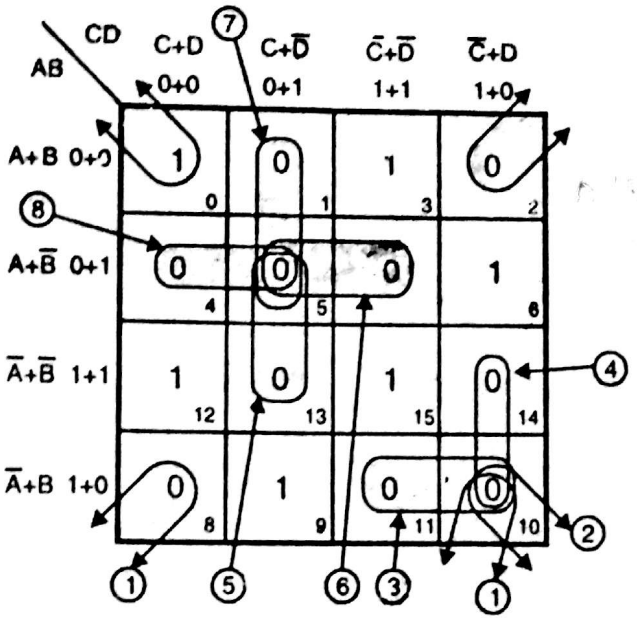

O/P $Y = (B + C + D)(\bar{A} + B)(A + \bar{B} + \bar{D})(\bar{B} + C + \bar{D})$
① ② ④ ③

**(a)** **Fig. Ex. 4.22 (vii)** **(b)**

Scanned by CamScanner

(viii) F (A, B, C, D) = πM (1, 2, 4, 5, 7, 8, 10, 11, 13, 14)

**Soln. :**



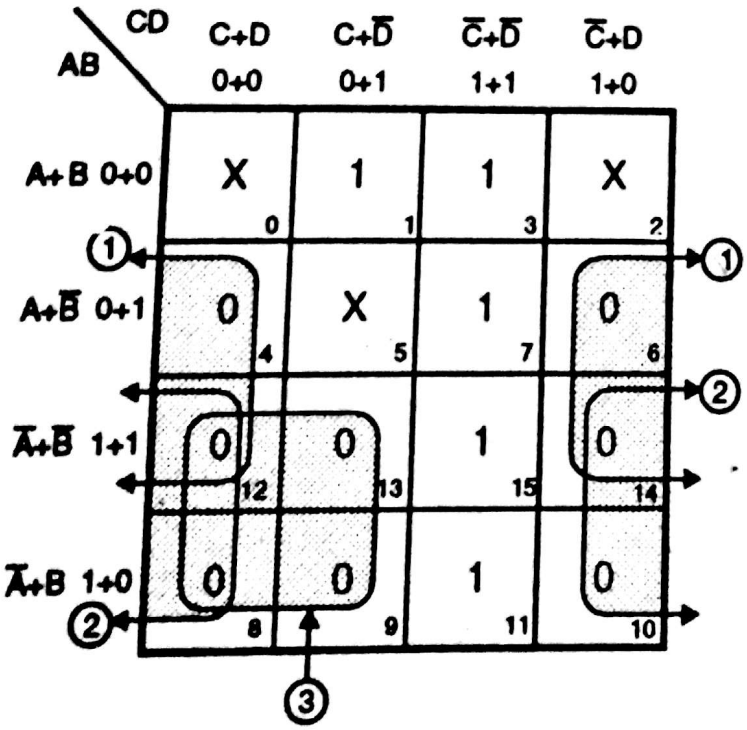

O/P Y = $(\bar{A} + B + D)(B + \bar{C} + D)(\bar{A} + B + \bar{C})(\bar{A} + \bar{C} + D)(\bar{B} + C + \bar{D})(A + \bar{B} + \bar{D})(A + C + \bar{D})(A + B + C)$

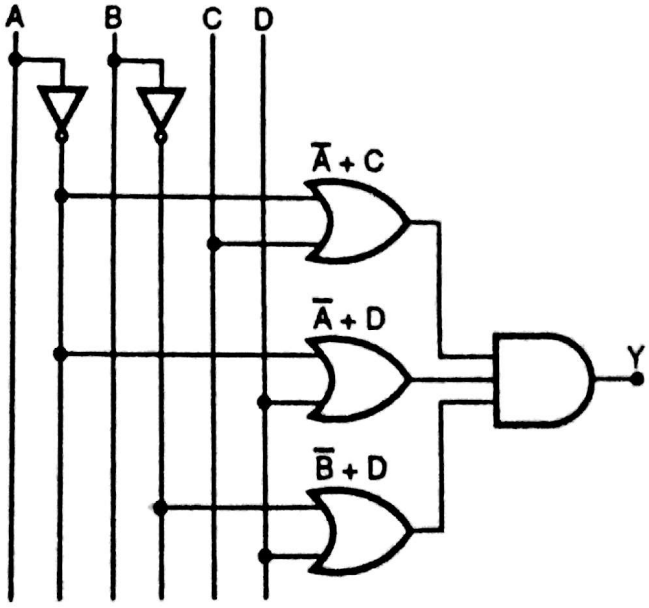
① ② ③ ④ ⑤ ⑥ ⑦ ⑧

**Fig. Ex. 4.22 (viii)**

(ix) F (A, B, C, D) = πM (4, 6, 8, 9, 10, 12, 13, 14) + d (0, 2, 5).

**Soln. :**




O/P Y = $(\bar{A} + C)(\bar{A} + D)(\bar{B} + D)$ [Without taking X into account]

③ ② ①

O/P Y = $(\bar{A} + C) \cdot (D)$ [Considering X in O and D as 0]

(a) (b)

**Fig. Ex. 4.22 (ix)**

Scanned by CamScanner

This example includes don't care condition ('X') also. The steps are same as explained i.e. write 'X' in the boxes numbered (0, 2, 5). Fill '0' in 4, 6, 8, 9, 10, 12, 13, 14. Group '0' and if required (if circuit minimises) also group 'X' into, i.e. force that 'X' of box to logic '0'. This is the same way, you were forcing 'X' to logic '1' in case of SOP form (minterms).

Now we are going to solve some examples in which you will be given with a function expression mentioning ***minterms*** (not maxterms) and you have to find out POS form (not SOP form). Method is very easy, from given minterm function, you find out maxterm. Let's say for example, given function is,

$F = \Sigma m\,(1, 3, 7, 11, 15) \rightarrow$ specifies minterms. Therefore remaining numbers are maxterms.

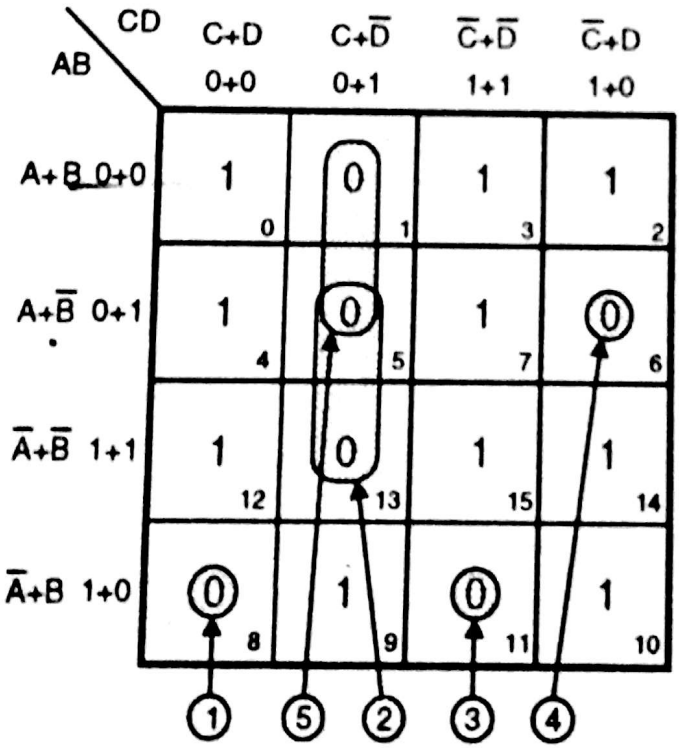
i.e. $F = \pi M\,(0, 2, 4, 5, 6, 8, 9, 10, 12, 13, 14)$. (i.e. reverse of minterm). Now next step is simple, once you get maxterm function, simply map it in K–map, group it, and find Boolean expressions. Let's solve same examples.

---

**Ex. 23 :** Find expression in POS form for given function.

(i) $F\,(A, B, C, D) \triangleq \Sigma m\,(0, 2, 3, 4, 7, 9, 10, 12, 14, 15)$.

**Soln. :**




$$O/P\ Y = (\bar{A} + B + C + D)(\bar{B} + C + \bar{D})(\bar{A} + B + \bar{C} + \bar{D})(A + \bar{B} + \bar{C} + D)(A + C + \bar{D})$$

① ② ③ ④ ⑤

**Fig. Ex. 4.23 (i) : (a)**

Scanned by CamScanner

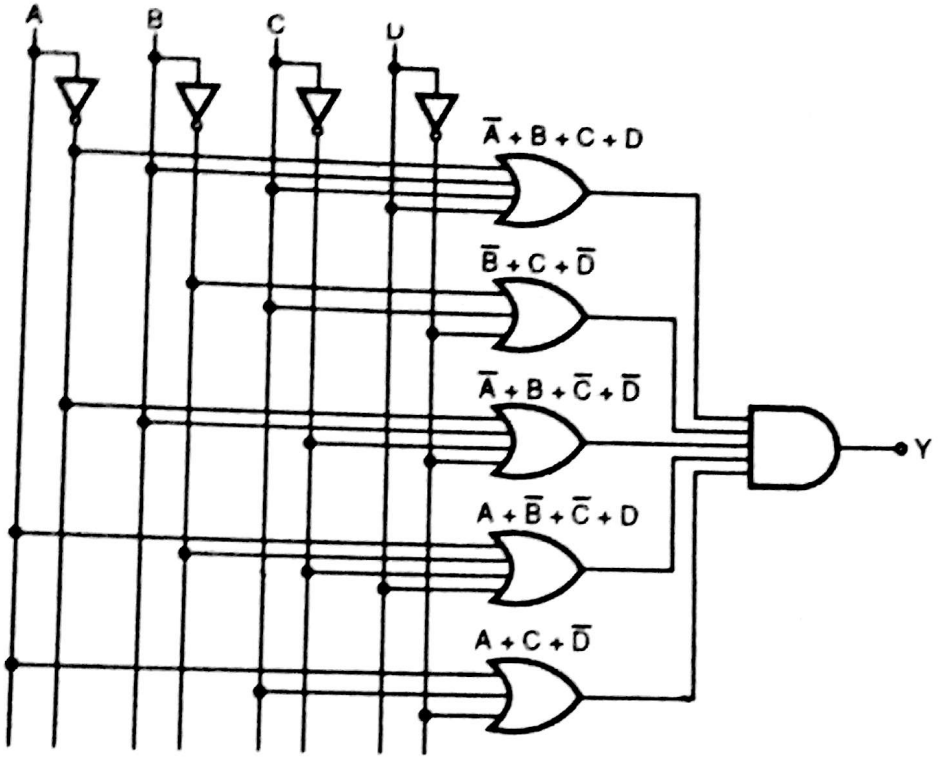
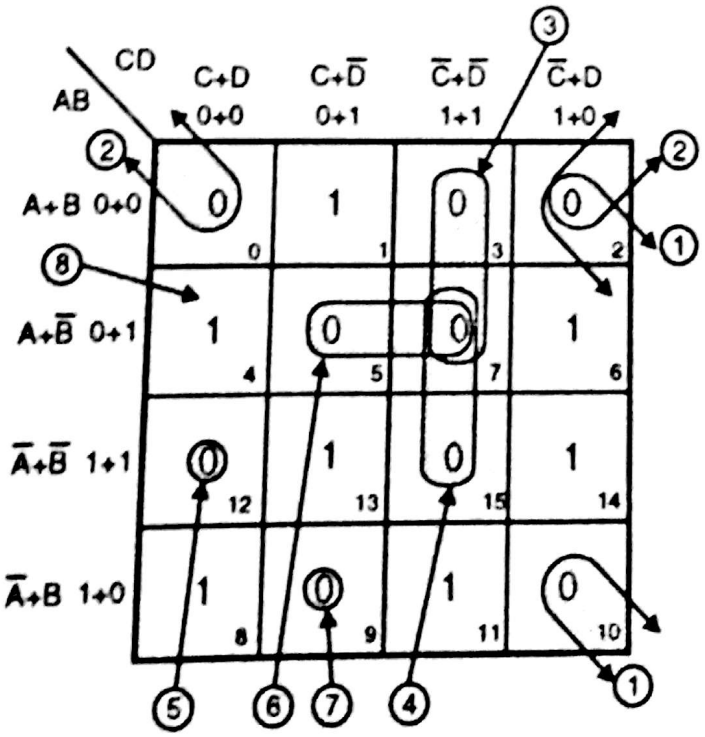



**Fig. Ex. 4.23 (i) : (b)**

**(ii)** F (A, B, C, D) = $\Sigma$m (1, 4, 6, 8, 11, 13, 14)

**Soln. :** First find out function using maxterm.

$\therefore$ F (A, B, C, D) = $\pi$M (0, 2, 3, 5, 7, 9, 10, 12, 15). Now map maxterm in 4 variable K–map.




O/P Y = $(B+\overline{C}+D)(A+B+D)(A+\overline{C}+\overline{D})(\overline{B}+\overline{C}+\overline{D})(\overline{A}+\overline{B}+C+D)(A+\overline{B}+\overline{D})(\overline{A}+B+C+\overline{D})$

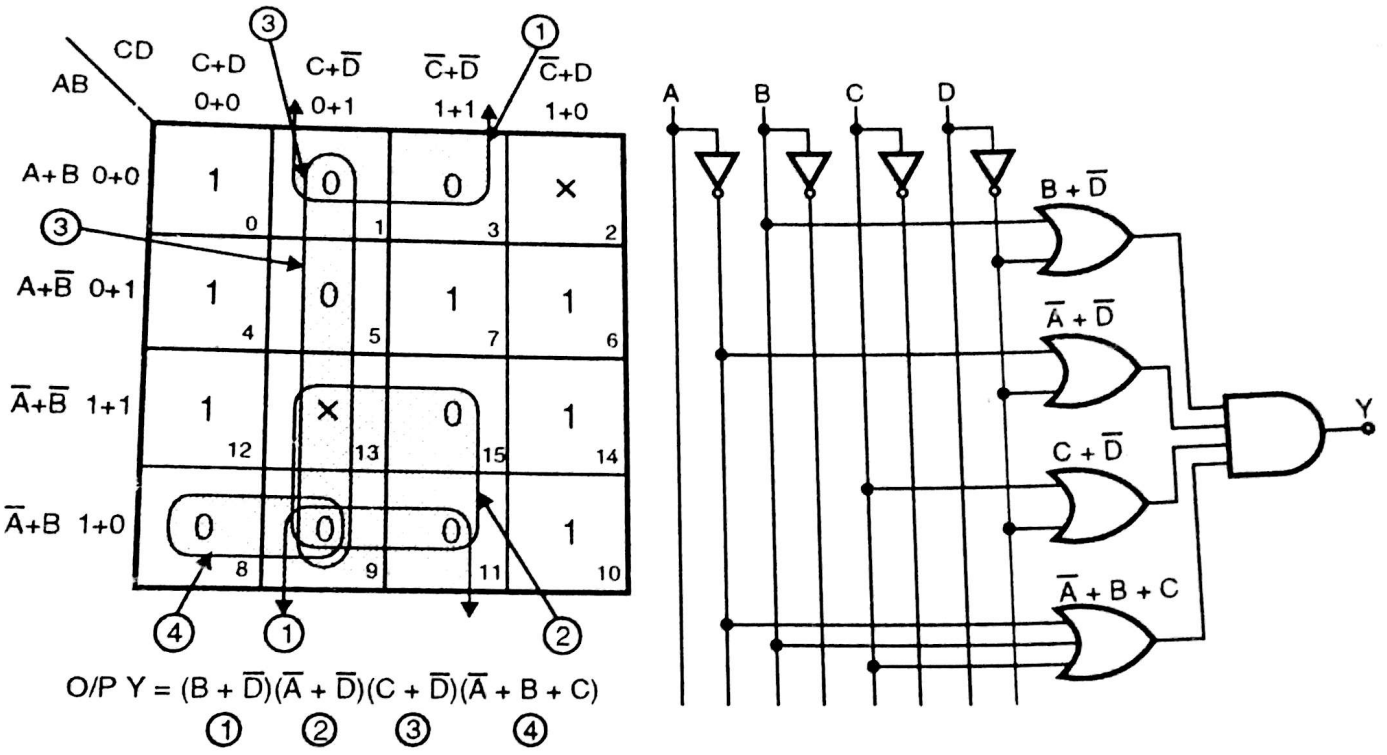
① ② ③ ④ ⑤ ⑥ ⑦

**Fig. Ex. 4.23 (ii)**

Scanned by CamScanner

**(iii)** $F(A, B, C, D) = \Sigma m(0, 4, 6, 7, 10, 12, 14) + d(2, 13)$

**Soln. :**




**(a)** **Fig. Ex. 4.23 (iii)** **(b)**

In this problem you should retain don't care condition as it is and only remaining numbers are selected as maxterm.

$$F(A, B, C, D) = \pi M(1, 3, 5, 8, 9, 11, 15) + d(2, 13).$$

## 4.8 Minimise Boolean Expression using K-Map or Obtaining K-Map for given Boolean Expression :

In this section we are going to ***reverse engineer*** the K-map, means uptill now directly decimal value of minterm was given to you and you were finding the Boolean expressions for the same. But now you will be given Boolean expression and using K-map you have to find out position of minterms, and if required you have to minimise it.

The procedure is very simple. The steps are as follows :

**Step I** : Observe Boolean expression carefully and conclude either it is two, three, four or five variable K-map.

**Step II** : Depending upon number of variables, you draw the basic skeleton of K-map.

Scanned by CamScanner

**Step III :** Now take terms of Boolean expression one by one, and analyse. Now suppose you have 4 variable K-map and term given is $B\bar{C}$. Then it is very easy to find where are the minterms.

Refer Fig. 4.5 (a) first you have to find out, where B is constant and put 'slash' (or darken that area). Same way you find out the 'span' for variable $\bar{C}$ and darken it. The area which is common to both B and $\bar{C}$, you have to simply put 1's over there. Refer Fig. 4.5 (b).

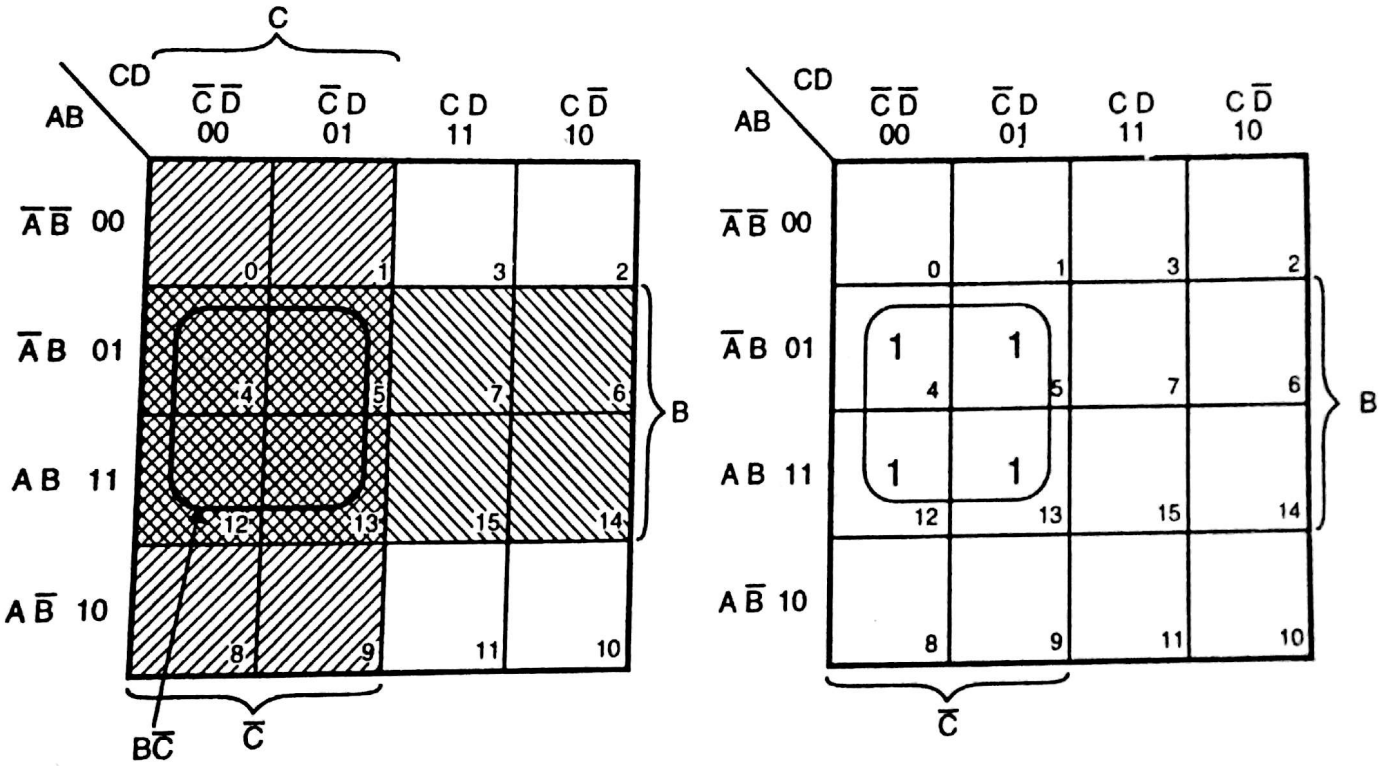



Fig. 4.5

**Step IV :** Repeat same step for all the terms and finally you will get K-map with all minterms (logic '1'). Remaining boxes of K-map should be filled up with logic '0'.

**Step V :** Minimise it, i.e. write down 'new' Boolean expression if the same is specified in the problem.

Let's solve one example to understand the concept.

**Ex. 24 :** Reduce following expression using K-map :

(i) $F = ABCD + A\bar{C}D + B\bar{C} + A\bar{B}CD$.

**Soln. :** (1) Observing the expression we conclude that it has 4 variables.

(2) Draw basic skeleton for 4 variable K–map (minterm).

Scanned by CamScanner

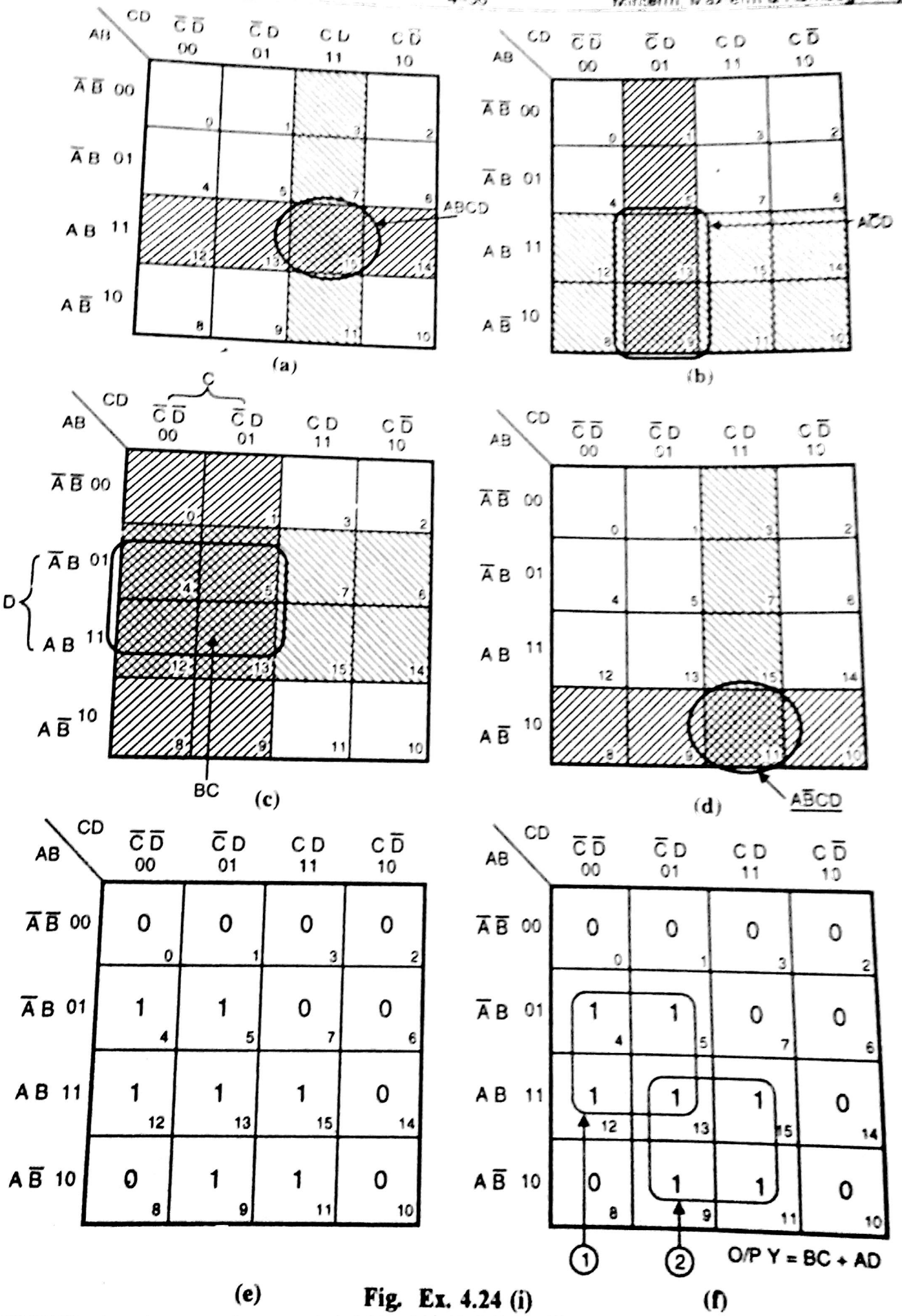



**Fig. Ex. 4.24 (i)**

Scanned by CamScanner

Take term, one by one.

Fig. Ex. 4.24 (i) (a) → K map for term ABCD (b) → K map for term ACD

(c) → K map for term BC (d) → K map for term ABCD

Fig. Ex. 4.24 (i) (e) shows final combination of Fig. (a), (b), (c), (d).

∴ Finally we can write down,

$F(A, B, C, D) = \Sigma m (4, 5, 9, 11, 12, 13, 15)$

Now reduce it. Refer Fig. Ex. 4.24 (i) (f).

∴ Minimised Boolean expression is, $\boxed{Y = BC + AD}$

---

(ii) Obtain 'K' map for $f(A, B, C, D) = A\bar{B} + BC + C\bar{D}$ **(May 96, 6 Marks)**

**Soln. :**

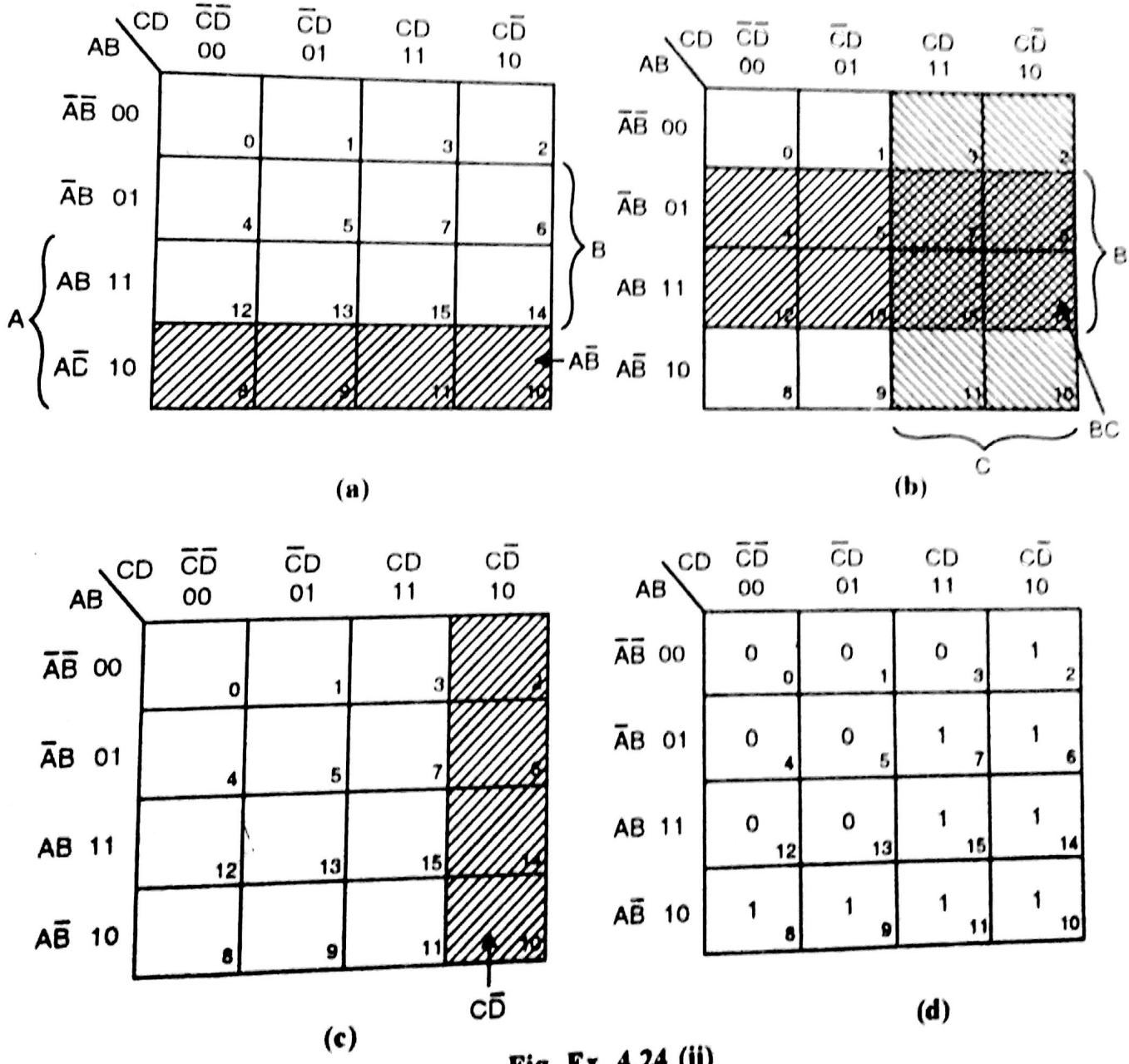



**Fig. Ex. 4.24 (ii)**

Scanned by CamScanner

(a) › shows map for AB, it covers minterms 8, 9, 11, 10
(b) › shows map for BC, it covers minterms 7, 6, 15, 14
(c) › shows map for CD, it covers minterms 2, 6, 14, 10

Finally, we can write

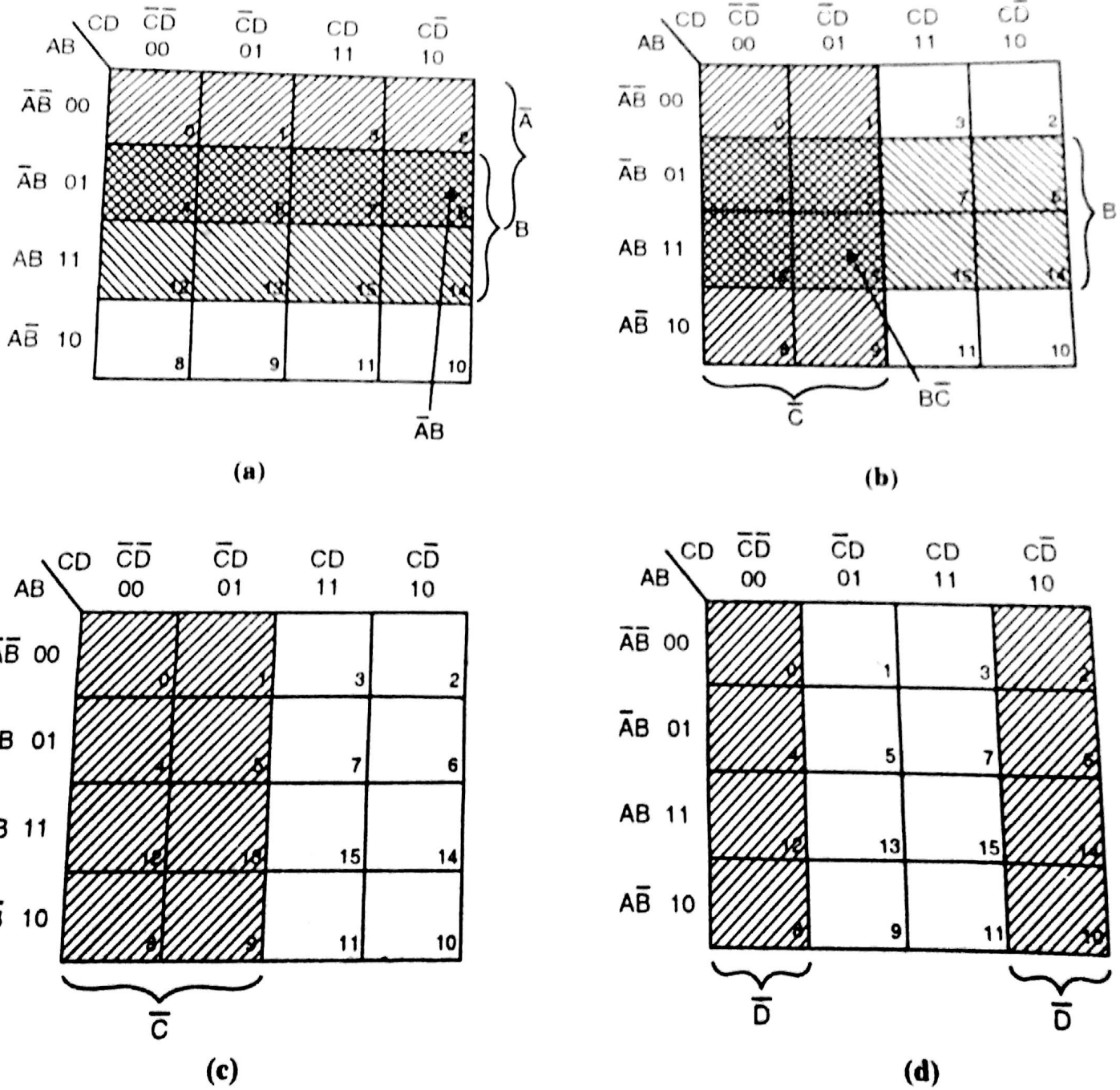
$f(A, B, C, D) = \Sigma m (2, 6, 7, 8, 9, 10, 11, 14, 15)$

**(iii)** Obtain K-map for following expression **(Dec. 96, 6 Marks)**

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

**Soln. :** $f(A, B, C, D) = \bar{A}B + BC + CD = AB + BC + C + D$ (Demorgan's theorem)

Now draw K-map for $\bar{A}B$, $BC$, $C$, $D$




(a) (b) (c) (d)

**Fig. Ex. 4.24 (iii)**

Scanned by CamScanner

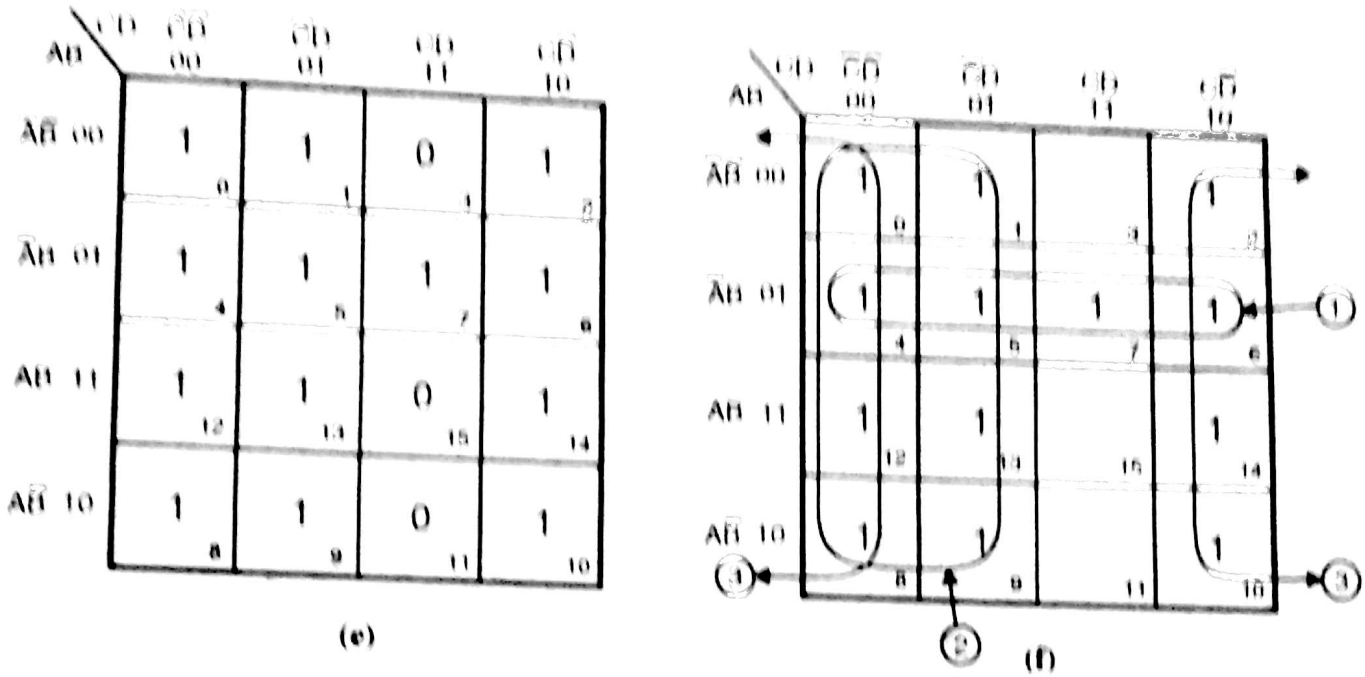



Fig. Ex. 4.24 (iii)

Let's simplify $= AB + C(1 + B) + D$

$= AB + C + D$

(1) (2) (3)

(iv) Draw K-map for following function and simplify. **(May 97, 6 Marks)**

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

**Soln. :** We have to draw 3 variable K-map.

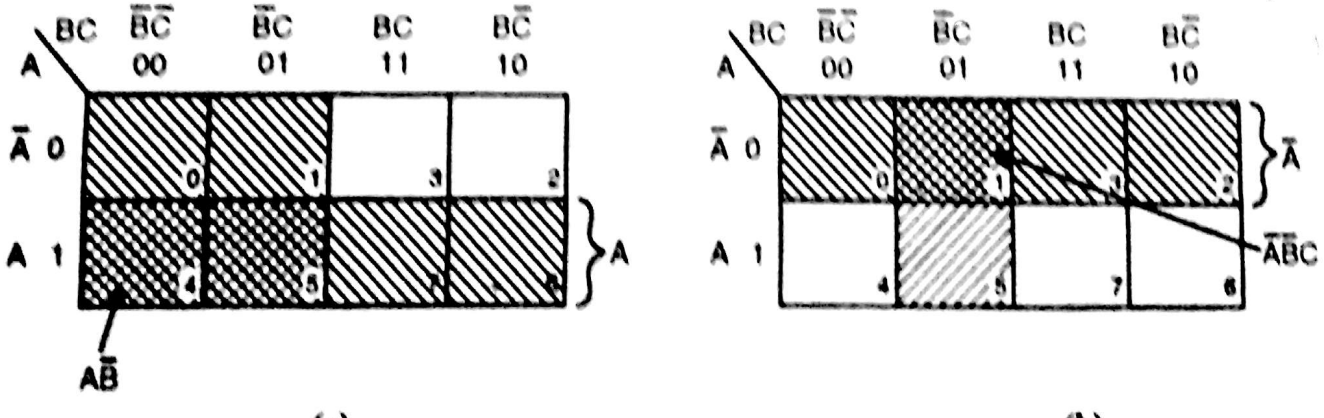
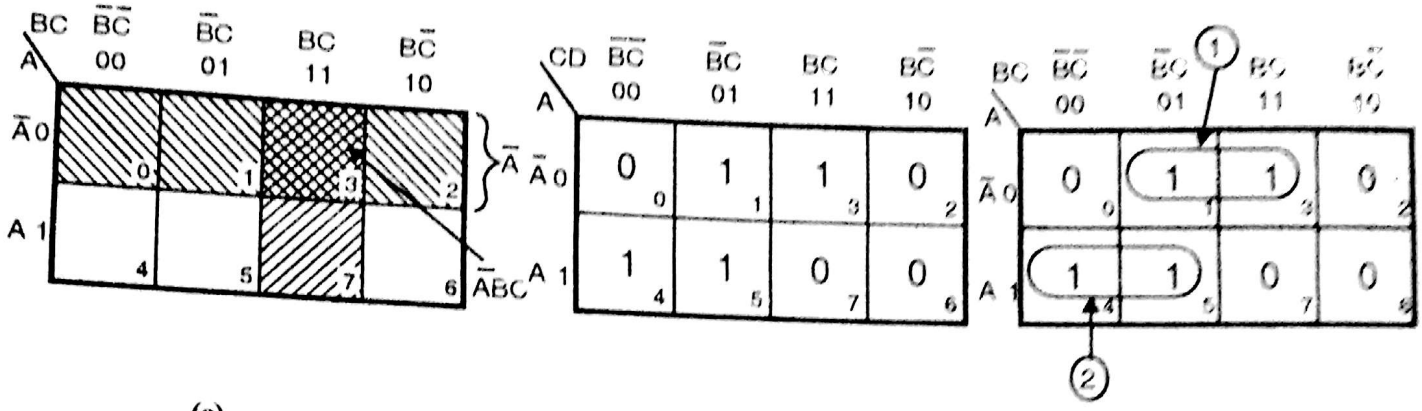



Fig. Ex. 4.24 (iv)

Scanned by CamScanner




(c) (d) (e)

**Fig. Ex. 4.24 (iv)**

Fig. Ex. 4.24 (iv)

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

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

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

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

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

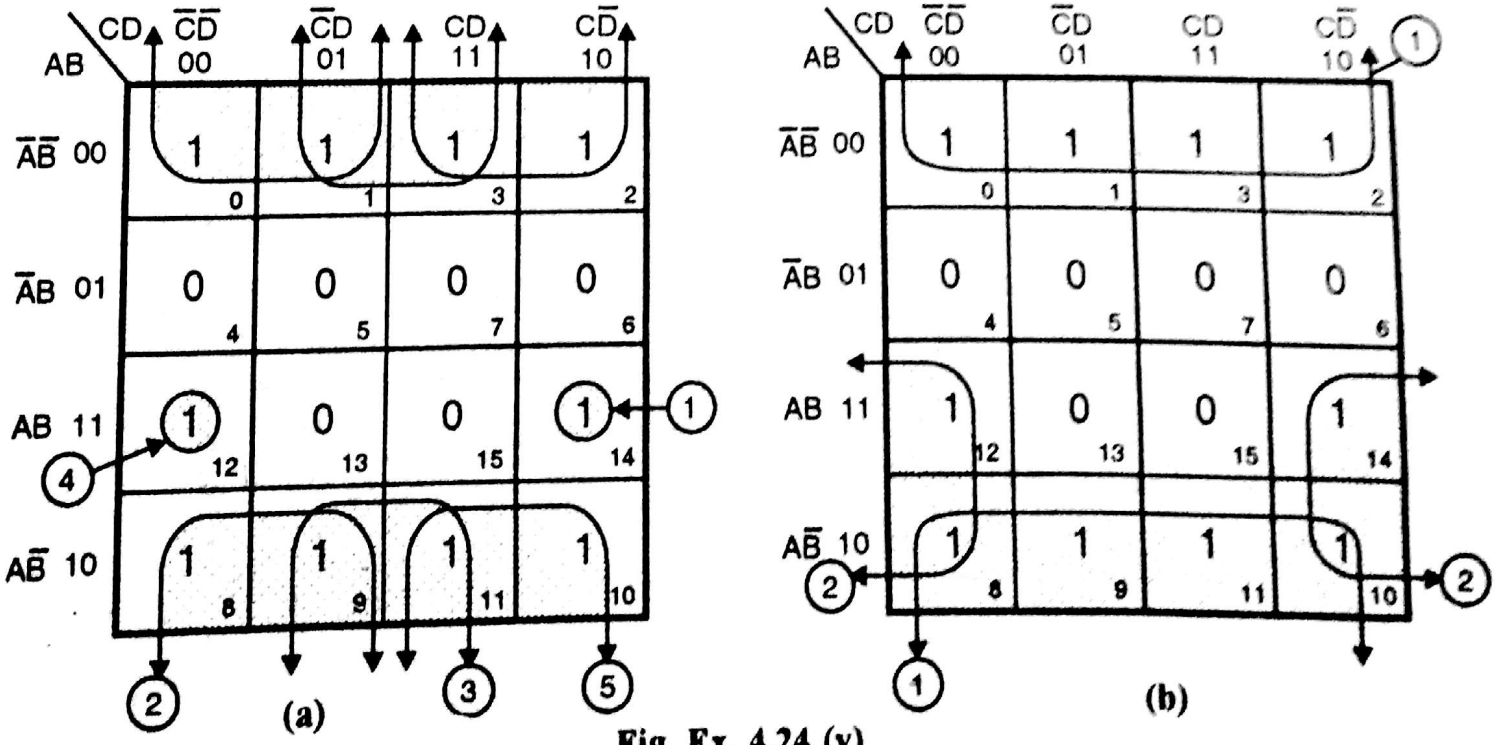
∴ **Ouput $Y = A\bar{B} + \bar{A}C$**

---

(v) Draw K-map for following expression : **(June 99, 6 Marks)**

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

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




(a) (b)

**Fig. Ex. 4.24 (v)**

Scanned by CamScanner

Group 1 → $ABCD$

2 → $B\bar{C}$

3 → $\bar{B}D$

4 → $AB\bar{C}\bar{D}$

5 → $\bar{B}C$

Newly mapped K-map

Group 1 → $\bar{B}$

2 → $A\bar{D}$

∴ (Ouput) $Y = \bar{B} + A\bar{D}$

**Conclusions about K-map :**

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

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

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

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

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

## 4.9 Quine Mc Cluskey Method or Tabular Method :

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

**Prime Implicant :**

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

Let's take one example,

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

Remaining minterms not appearing are,

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

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

**Code Word :**

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

Scanned by CamScanner

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

Say for example,

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

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

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

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

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

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

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

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

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

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

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

Let's solve one example to understand it better.

**(i)** Find minimal SOP expression for the function :

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

**Soln. : Step I :** Listing of all the minterm.

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

**Table 4.7**

Scanned by CamScanner

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

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

**Table 4.8**

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

| | | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
|---|---|---|---|---|---|---|
| $m_0$ | = | 0 | 0 | 0 | 0 | 0 |
| $m_8$ | = | 1 | 0 | 0 | 0 | 0 |

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

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

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

While comparing $m_8$ with minterms you will find only two match

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

These two terms should be written under group 2.

Now group 3 will start.

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

Scanned by CamScanner

**First Reduction Table**

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

**Table 4.9**

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

**First Reduction Table**

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

**Table 4.10 (a)**

Scanned by CamScanner

| Second Reduction Table | | | | | | |
|---|---|---|---|---|---|---|
| Group | Minterm | Binary Equivalent | | | | |
| | | $B_4$ A | $B_3$ B | $B_2$ C | $B_1$ D | $B_0$ E |
| 2 | $m_8, m_{10}, m_{12}, m_{14}$ | 0 | 1 | X | X | 0 |
| | $m_8, m_{12}, m_{10}, m_{14}$ | 0 | 1 | X | X | 0 |
| 3 | $m_{17}, m_{19}, m_{25}, m_{27}$ | 1 | X | 0 | X | 1 |
| | $m_{17}, m_{21}, m_{25}, m_{29}$ | 1 | X | X | 0 | 1 |
| | $m_{17}, m_{25}, m_{19}, m_{27}$ | 1 | X | 0 | X | 1 |
| | $m_{17}, m_{25}, m_{21}, m_{29}$ | 1 | X | X | 0 | 1 |

**Table 4.10 (b)**

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

Final reduction from Table 4.10 (b) is,

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

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

**Cover Table :**

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

**Table 4.11**

Scanned by CamScanner

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

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

$\downarrow$

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

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

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

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

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

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

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

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

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

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

$\therefore$ Final solution is,

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

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

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

('X' terms are not considered).

---

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

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

**Soln. :**

**Step I :** List all minterms :

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

**Table 4.12**

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

Scanned by CamScanner

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

**Table 4.13**

**Step III :** Form first reduction Table.

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

**Table 4.14**

**Step IV :** Now list second reduction Table :

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

**Table 4.15**

Scanned by CamScanner

Finally reducing this Table 4.15, we get,

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

**Table 4.16**

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

**Step V :** Cover Table

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

**Table 4.17**

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

So, ***reduced cover Table.***

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

**Table 4.18**

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

∴ Final reduction,

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

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

**) Minimise expression using Quine–Mc Cluskey method.**

Scanned by CamScanner

**Soln. :** First list out all the minterms.

**Step I :**

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

**Table 4.19**

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

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

**Table 4.20**

**Step III :** Now prepare 1st reduction Table.

**First Reduction Table**

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

**Table 4.21**

Scanned by CamScanner

Further reduction is not possible.

**Step IV :** Cover Table :

$P_A$ = X010
$P_B$ = X100
$P_C$ = 10X0
$P_D$ = 1X00
$P_E$ = 101X
$P_F$ = 110X
$P_G$ = 1X11
$P_H$ = 11X1
$P_I$ = 0001

| | $m_1$ | $m_2$ | $m_4$ | $m_8$ | $m_{10}$ | $m_{11}$ | $m_{12}$ | $m_{13}$ | $m_{15}$ |
|---|---|---|---|---|---|---|---|---|---|
| $P_A$ | | X | | | X | | | | |
| $P_B$ | | | X | | | | X | | |
| $P_C$ | | | | X | X | | | | |
| $P_D$ | | | | X | | | X | | |
| $P_E$ | | | | | X | X | | | |
| $P_F$ | | | | | | | X | X | |
| $P_G$ | | | | | | X | | | X |
| $P_H$ | | | | | | | | X | X |
| $P_I$ | X | | | | | | | | |

**Table 4.22**

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

Therefore final function is,

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

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

---

(iii) Simplify following using tabular method :

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

**Soln. :**

**Step I :**

| Minterms | Binary |
|---|---|
| $m_{20}$ | 0 1 0 1 0 0 |
| $m_{28}$ | 0 1 1 1 0 0 |
| $m_{52}$ | 1 1 0 1 0 0 |
| $m_{60}$ | 1 1 1 1 0 0 |

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

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

**Step III :** First Reduction Method.

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

Scanned by CamScanner

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

**Step V :** Final reduction gives us,

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

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

(iv) Reduce using Quine Mc Cluskey method :

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

**Soln. :** List all the minterms.

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

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

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

**First level implicants (Reduction Table)**

Scanned by CamScanner

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

**Second Reduction Table**

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

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

Prime Implicants

$P_A = X010$

$P_B = 01X0$

$P_C = X100$

$P_D = 10X0$

$P_E = X111$

$P_F = 11X1$

$P_G = 0X1X$

$P_H = 1X0X$

| | $m_2$ | $m_3$ | $m_4$ | $m_6$ | $m_7$ | $m_8$ | $m_9$ | $m_{10}$ | $m_{12}$ | $m_{13}$ | $m_{15}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $P_A$ | X | | | | | | | X | | | |
| $P_B$ | | | X | X | | | | | | | |
| $P_C$ | | | X | | | | | | X | | |
| $P_D$ | | | | | | X | | X | | | |
| $P_E$ | | | | | X | | | | | | X |
| $P_F$ | | | | | | | | | | X | X |
| $P_G$ | X | X | | X | X | | | | | | |
| $P_H$ | | | | | | X | X | | X | X | |
| | ✔ | ✔ | | ✔ | ✔ | ✔ | ✔ | | ✔ | ✔ | |

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

Therefore reduced cover Table is.

Scanned by CamScanner

$P_A = X010$

$P_B = 01X0$

$P_C = X100$

$P_D = 10X0$

$P_E = X111$

$P_F = 11X1$

| | $m_4$ | $m_{10}$ | $m_{15}$ |
|---|---|---|---|
| $P_A$ | | X | |
| $P_B$ | X | | |
| $P_C$ | X | | |
| $P_D$ | | X | |
| $P_E$ | | | X |
| $P_F$ | | | X |

Therefore we select $P_A$, $P_B$ and $P_E$.

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

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

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

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

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

Scanned by CamScanner

## 4.10 Examples from University Papers :

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

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

**Ans.:** Refer Section 4.6.3, Ex. 11.

2. **Obtain K–map for :**

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

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

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

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

**Ans. :** First list out all minterms.

**Step I :**

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

**Table 1**

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

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

**Table 2**

Scanned by CamScanner

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

**Table 3**

**Step IV :** Now list second reduction table from Table 3

Second Reduction Table

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

**Table 4**

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

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

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

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

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

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

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

**Ans. :** As highest state = 15

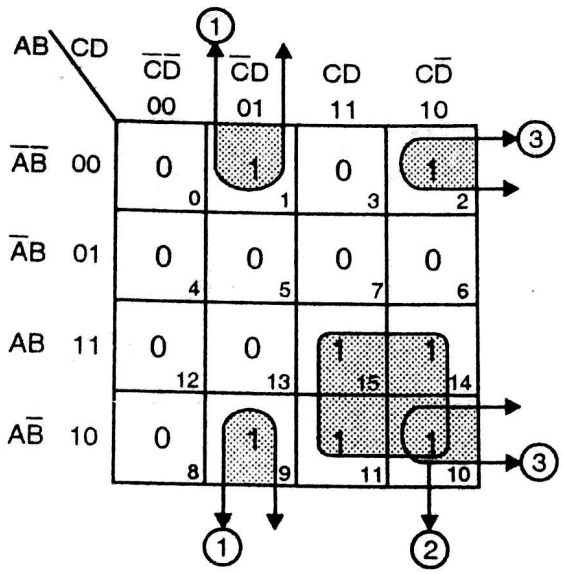
**Therefore K–map required will be of 4 variables (A, B, C, D).**

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

Scanned by CamScanner

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

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




After making different groups of ones. Write down output equation

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

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

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

Implementing above using Equation (1)

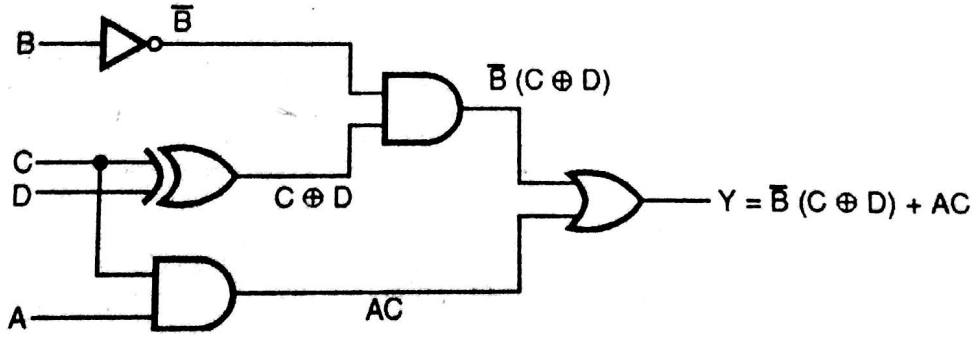



**Fig. Q. 4.4**

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

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

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

**6. Minimise the following expression using Quine McCluskey tabular method :**

$$f(A, B, C, D) = \Sigma\, m(1, 3, 4, 5, 6, 8, 12, 14, 15)$$ **(Dec. 96, 10 Marks)**

Scanned by CamScanner

**Ans. :**

**Step I :** List all the minterms

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

**Table 5**

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

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

**Table 6**

**Step III : Using Table 6 prepare first reduction table**

Scanned by CamScanner

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

**Table 7**

**Step IV :** From Table 7 from second reduction table.

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

**Table 8**

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

**Step V :** Cover table

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

**Table 9**

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

Scanned by CamScanner

So reduced cover table

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

**Table 10**

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

∴ Final reduction,

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

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

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

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

**Ans.:** Refer Section 4.6.3, Ex. 13.

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

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

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

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

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

**Ans. :**

**Step I :** List all the minterms.

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

**Table 11**

Scanned by CamScanner

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

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

**Table 12**

**Step III :** Using Table 12 prepare first reduction table.

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

**Table 13**

**Step IV :** Using Table 13 list second reduction

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

**Table 14**

Scanned by CamScanner

Finally reducing this Table 14, we get

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

**Table 15**

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

**Step V :** Cover table

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

**Table 16**

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

So reduced cover table.

| Prime Implicants | | | | | | Minterms | |
|---|---|---|---|---|---|---|---|
| | | | | | | $m_1$ | $m_9$ |
| $P_A$ | = | X | 0 | 0 | 1 | X | X |
| $P_B$ | = | 1 | X | 0 | 1 | | X |

**Table 17**

LIBRARY 743 IT

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

$\therefore$ Final reduction

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

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

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

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

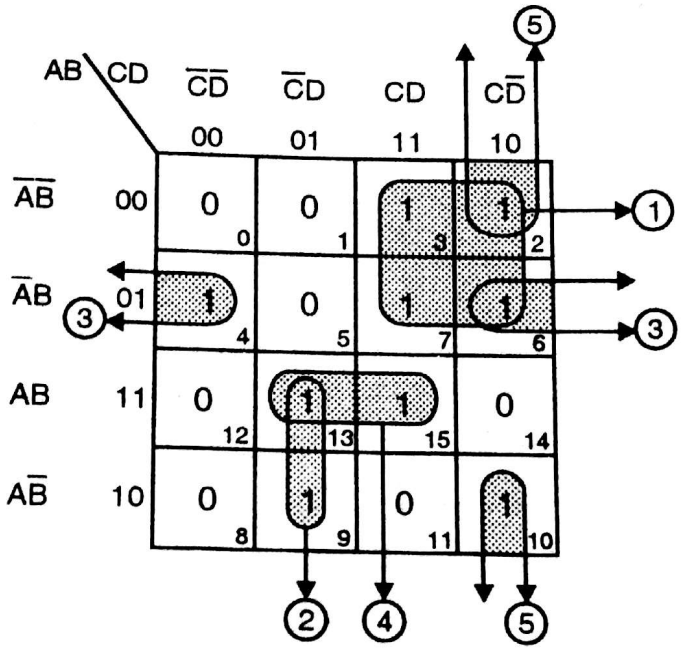
Scanned by CamScanner

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

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

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

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



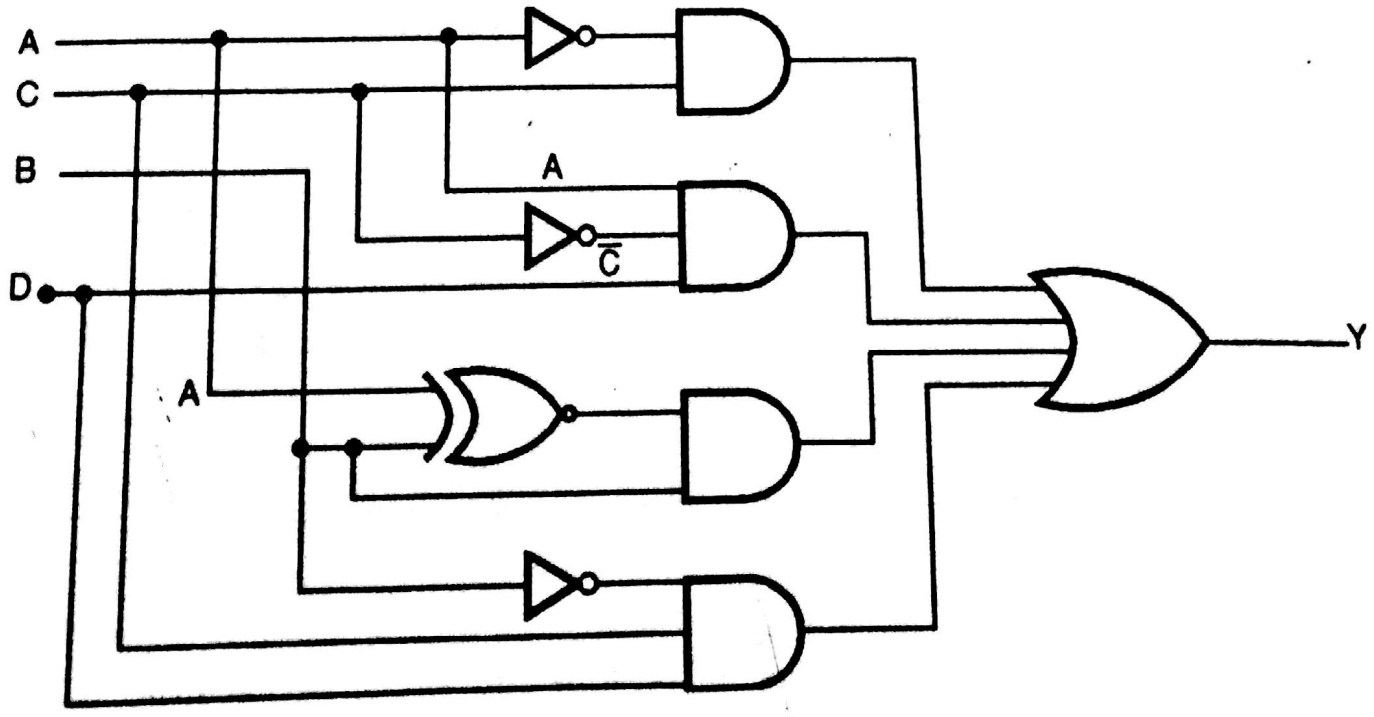

After making different groups of ones write down output equation.

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

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

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

**Implementing above using Equation (2)**




Scanned by CamScanner

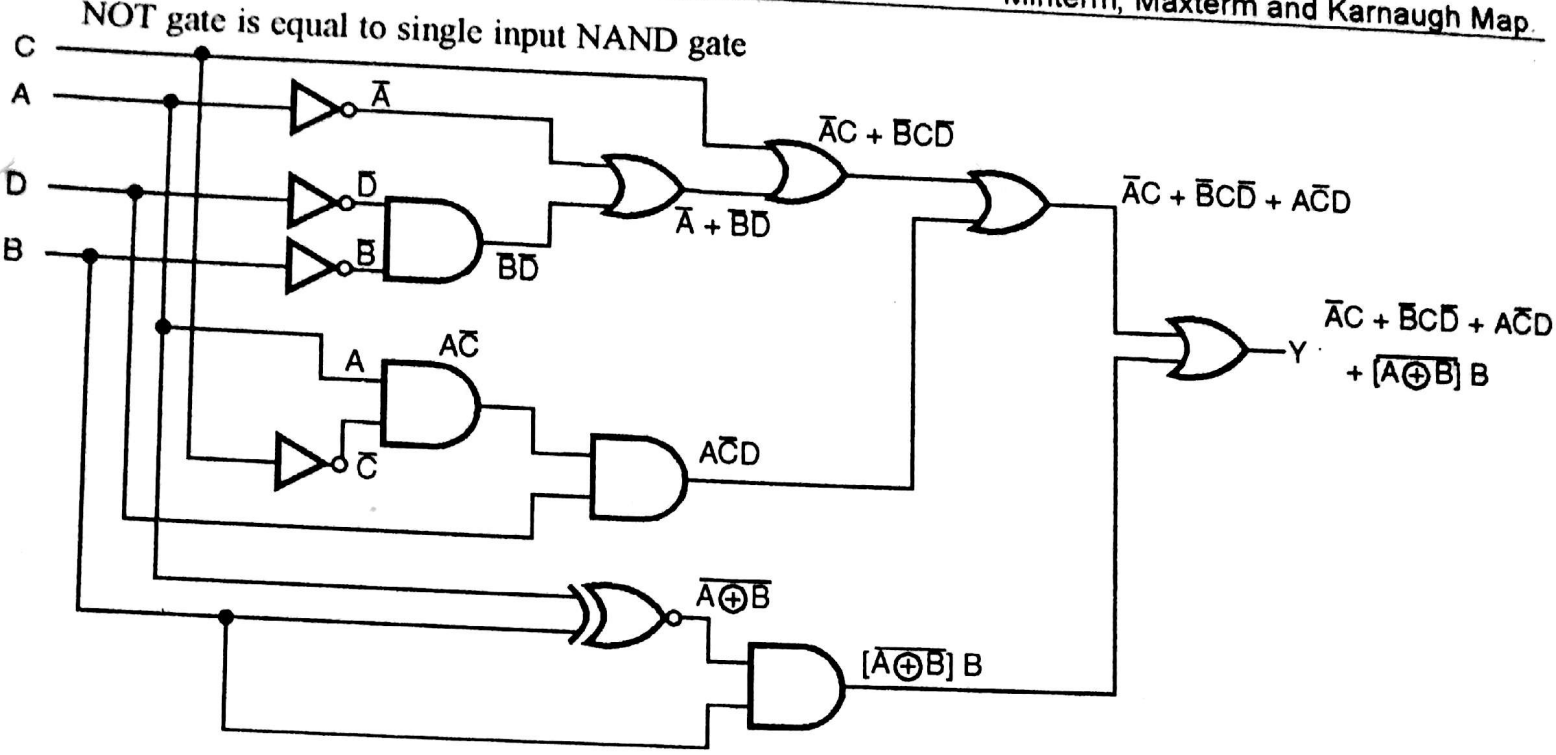



**Fig. Q. 4.10 (b)**

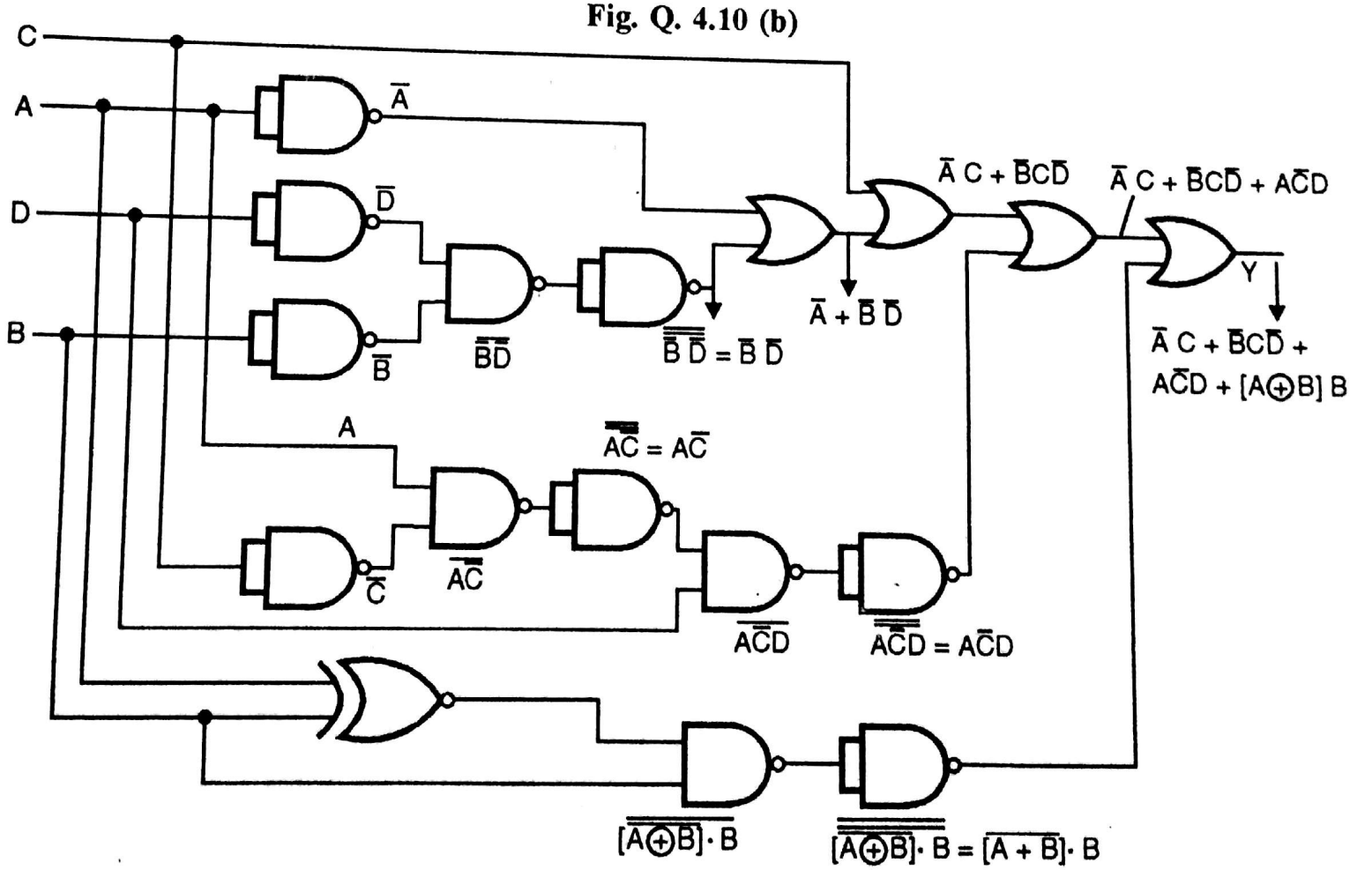



**Fig. Q. 4.10 (c)**

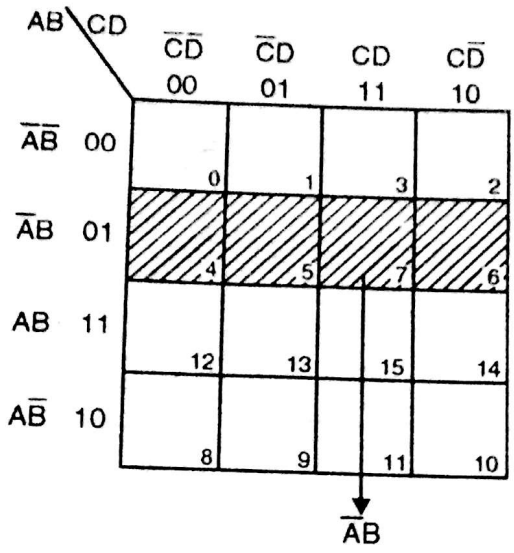
**11.** Obtain the K-map for the following function : **(Dec. 97, 6 Mark**

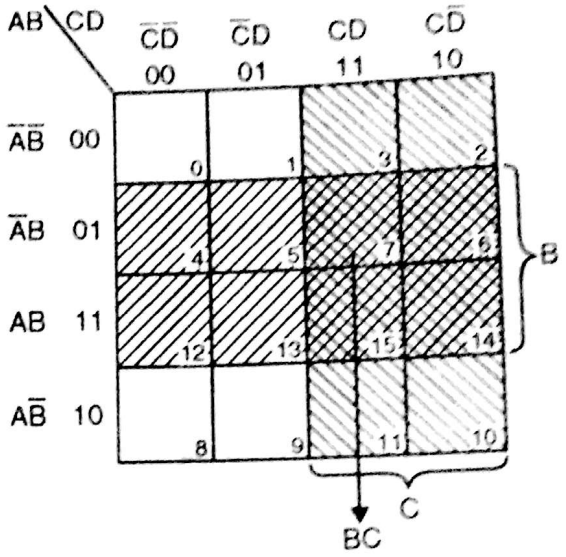
$$f(A, B, C, D) = \bar{A}B + BC + \bar{C}D$$

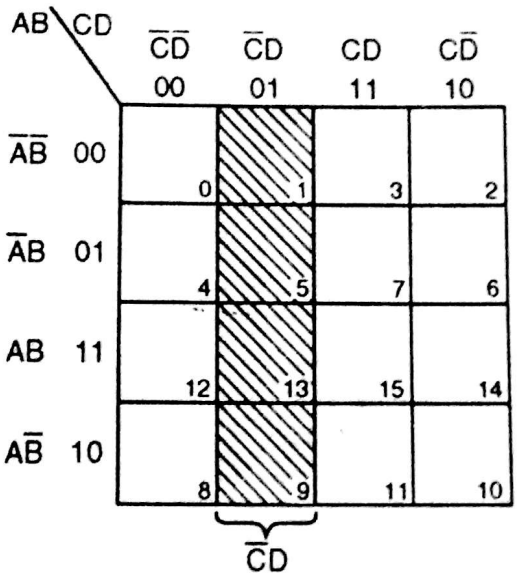
**Ans. :** Observing the expression we conclude that it has 4 variables,

**Draw basic skeleton for 4 variable K-map.**

Scanned by CamScanner

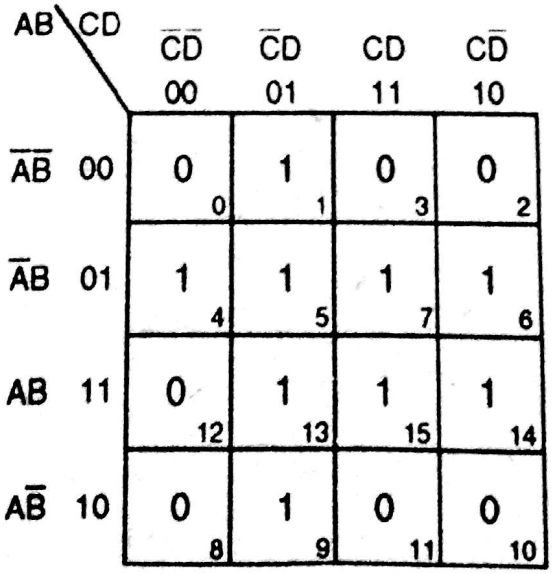



Now put 1 in the shaded blocks for every term in the equation and rest of the blocks are filled by zeros.

∴ K-map for the function will be




Scanned by CamScanner

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

$$f(A, B, C, D) = \sum m (1, 3, 7, 9, 10, 11, 13, 15)$$

**Ans. :** **(Dec. 97, 10 Marks)**

**Step I :** List all the minterms.

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

**Table 18**

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

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

**Table 19**

**Step III :** Using Table 19 prepare first reduction table.

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

**Table 20**

Scanned by CamScanner

**Step IV :** Using Table 20 list second reduction table

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

**Table 21**

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

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

**Table 22**

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

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

**Table 23**

All implicants are prime.

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

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

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

**(May 98, 10 Marks)**

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

Scanned by CamScanner

**Ans.:** The max-terms are,

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

**Step I :** List all the minterms

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

**Table 24**

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

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

**Table 25**

Scanned by CamScanner

**Step III :** Using Table 25 prepare first reduction table.

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

**Table 26**

**Step IV :** From Table 26 prepare second reduction table :

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

**Table 27**

Scanned by CamScanner

**Step V :** From Table 27 prepare final reduction table.

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

**Table 28**

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

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

**Table 29**

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

$\therefore$ Final reduction

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

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

**14. Consider following boolean expression :**

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

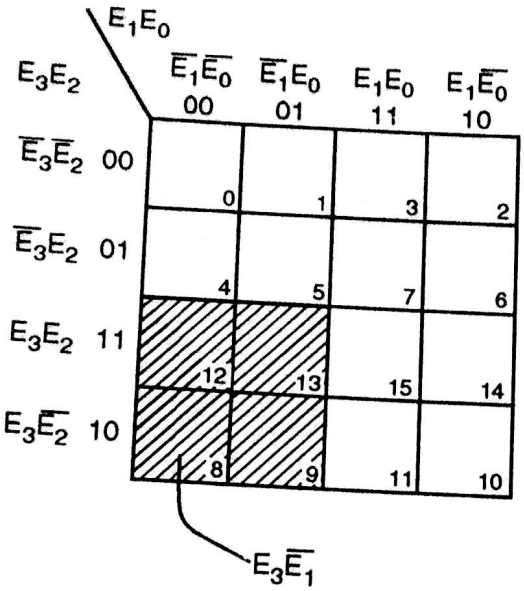
**(i) Show K–map for this function.**

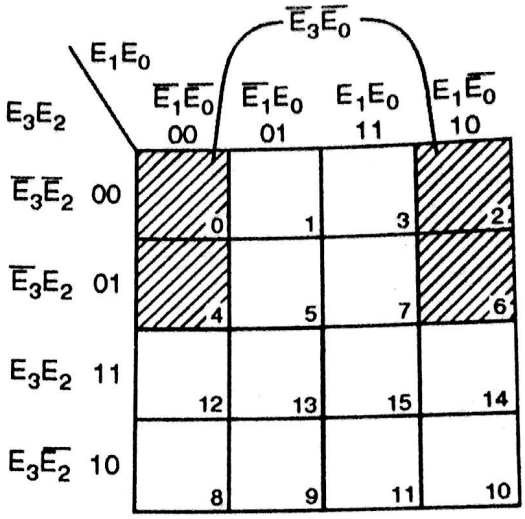
**(ii) Reduce it by using K–map.** **(May 98, 3 Marks)**

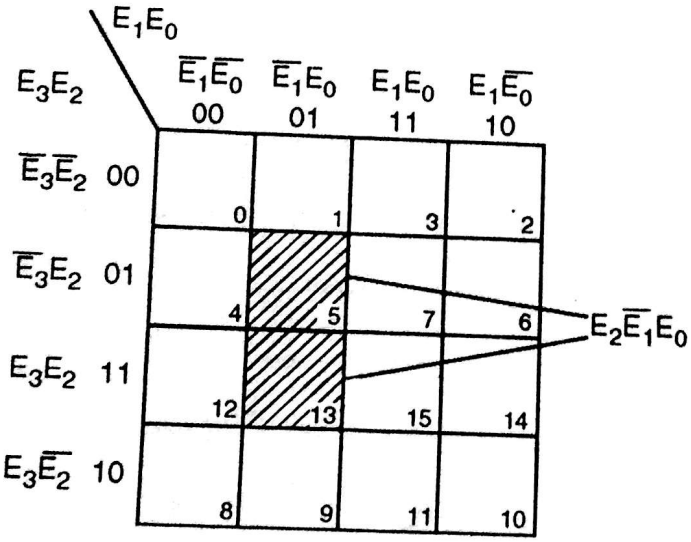
**Ans. :** (i) Observing the expression we conclude that it has 4 variables.

Draw basic skeleton of 4 variable K-map.

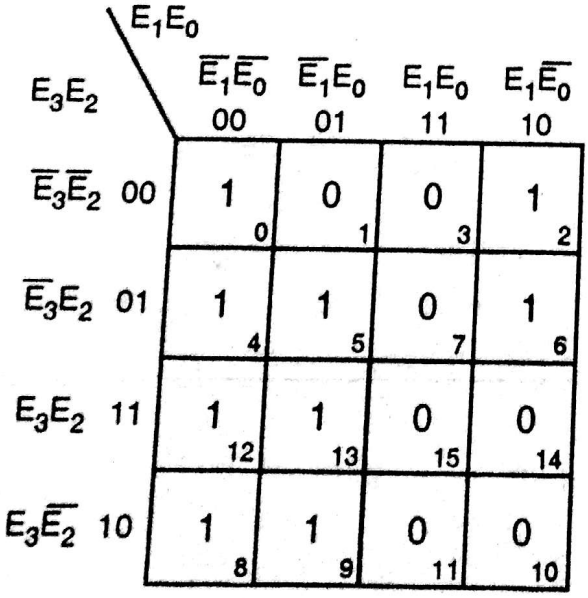
Scanned by CamScanner

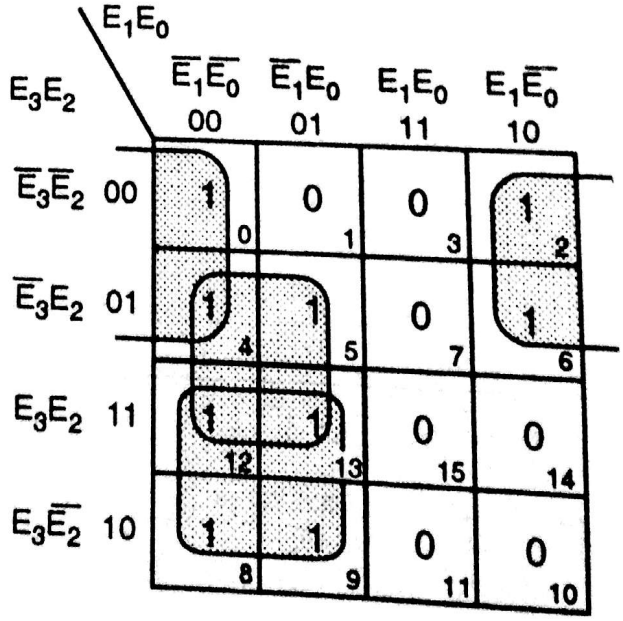



(ii) The K-map for the equation is as shown below :




$$\therefore \quad Y = \bar{E}_0\bar{E}_3 + \bar{E}_1E_3 + E_2\bar{E}_1$$

...Ans

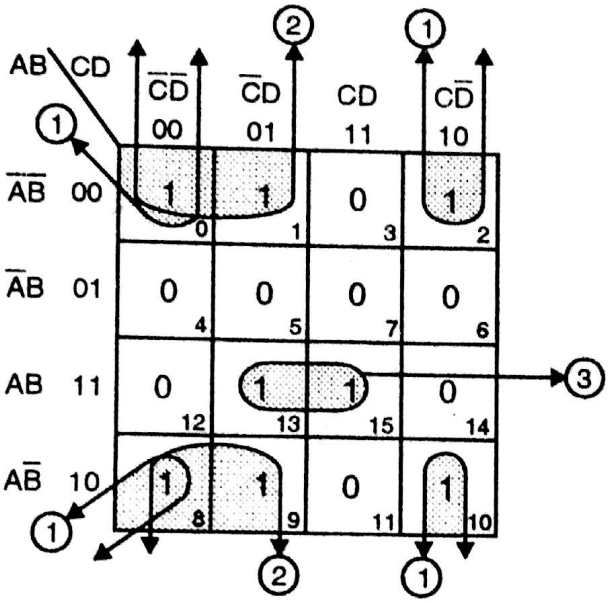
Scanned by CamScanner

**15. Draw Veitch diagram for the following function :**

$$f(A_3, A_2, A_1, A_0) = \Sigma\, m(0, 1, 2, 8, 9, 10, 13, 15)$$

**(May 98, 5 Marks)**

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




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

**...Ans.**

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

$$Y = A\overline{B} + A\overline{C} + C + AD + A\overline{B}C + ABC$$

**(May 98, 2 Marks)**

**Ans.:**

| AB \ CD | $\overline{C}\overline{D}$ 00 | $\overline{C}D$ 01 | CD 11 | $C\overline{D}$ 10 |
|---|---|---|---|---|
| $\overline{A}\overline{B}$ 00 | 0 (0) | 0 (1) | 1 (3) | 1 (2) |
| $\overline{A}B$ 01 | 0 (4) | 0 (5) | 1 (7) | 1 (6) |
| AB 11 | 1 (12) | 1 (13) | 1 (15) | 1 (14) |
| $A\overline{B}$ 10 | 1 (8) | 1 (9) | 1 (11) | 1 (10) |

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

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

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

Scanned by CamScanner

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

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

**...Ans.**

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

**Ans. :**

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

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

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

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

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

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

**...Ans.**

**Implement this using NAND gate**

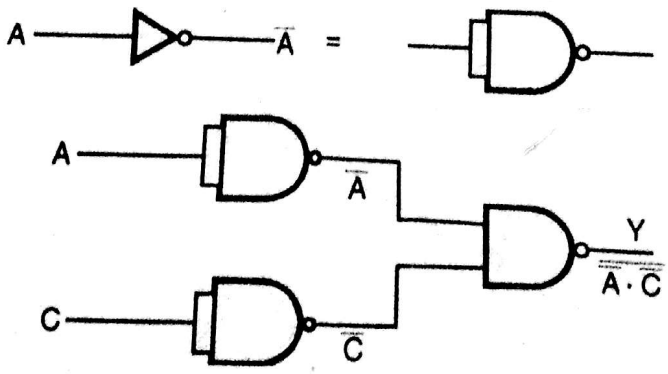



**Fig. Q.4.16**

**...Ans.**

Scanned by CamScanner

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

**Ans. :** The minimised expression from last answer is,

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

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

Simplify this, we get,

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

**...Ans.**

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

**Ans. :**

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

$$= (A + B + C)\,(D + \bar{D})\,(A + \bar{B} + C)\,(D + \bar{D}) \qquad \{\text{As } A + \bar{A} = 1\}$$

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

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

$$Y = A + C$$

Now using Boolean law

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

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

**...Ans.**

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

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

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

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

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

$$+ A_3\bar{A}_2\bar{A}_1A_0 + A_3\bar{A}_2A_1\bar{A}_0 + A_3\bar{A}_2A_1A_0$$

As $A + A = A$

So adding some terms to the equation we get

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

$$+ A_3\,\bar{A}_2A_1\,A_0 + A_3\,\bar{A}_2A_1\bar{A}_0 + A_3\,\bar{A}_2\bar{A}_1A_0 + A_3\bar{A}_2A_1A_0$$

$$= \bar{A}_3\bar{A}_2\bar{A}_0(\bar{A}_1 + A_1) + \bar{A}_3A_1A_0\,(\bar{A}_2 + A_2) + \bar{A}_3\bar{A}_2\,\bar{A}_1(A_0 + \bar{A}_0) + A_3\bar{A}_2A_0(\bar{A}_1 + A_1)$$

$$+ A_3\bar{A}_2A_1(\bar{A}_0 + A_0)$$

Scanned by CamScanner

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

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

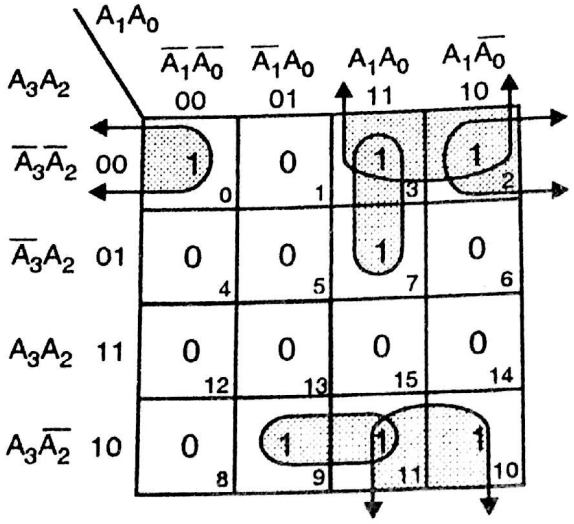
$$= \bar{A}_3\bar{A}_2\bar{A}_0 + \bar{A}_3A_1A_0 + \bar{A}_2A_1 + A_3\bar{A}_2A_0$$

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

**(Dec. 98, 4 Marks)**

**(ii) Find reduced expression using K-map.**

**Ans. :**




$$\boxed{Y = \bar{A}_3\,\bar{A}_2\,\bar{A}_0 + \bar{A}_3\,A_1\,A_0 + A_3\,\bar{A}_2\,A_0 + \bar{A}_2\,A_1}$$ ...Ans.

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

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

**ns. :**

**p I :** List all the minterms.

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

**Table 30**

Scanned by CamScanner

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

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

**Table 31**

**Step III :** Using Table 31 prepare first reduction Table :

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

**Table 32**

**Step IV :** Using Table 32 prepare second reduction table.

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

**Table 33**

Scanned by CamScanner

**Step V :**

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

**Table 34**

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

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

**Table 35**

$P_B$ $P_C$ and $P_D$ are essential implicants. If we cover $P_C$ and $P_D$, these terms will cover minterms 0, 1, 5, 13, 15, 16, 17, 29, 31. By taking these three terms, $P_A$ is covered.

∴ Final reduction

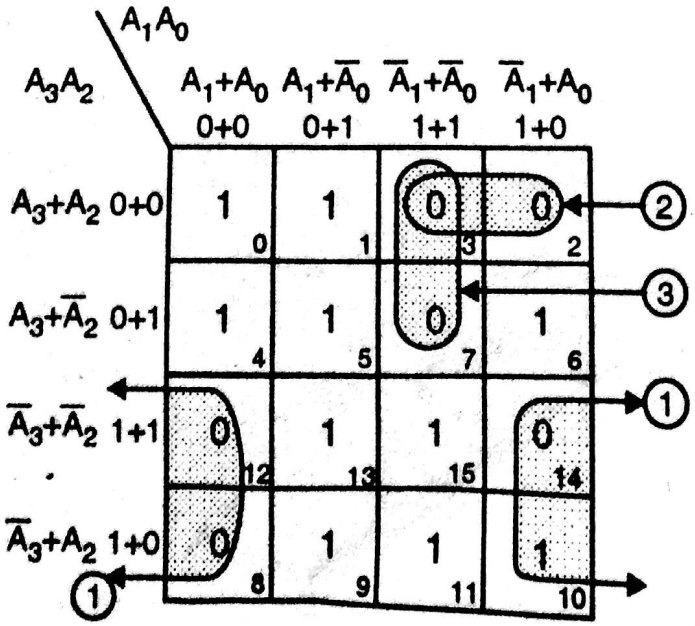
$$F(P_B, P_C, P_D) = \{ 0\ 0\ 1\ X\ 0, X\ 0\ 0\ 0\ X, X\ 1\ 1\ X\ 1 \}$$

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

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

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

**Ans. :**




Scanned by CamScanner

As it is product of maxterms make groups of zeros.

∴ Output equation is

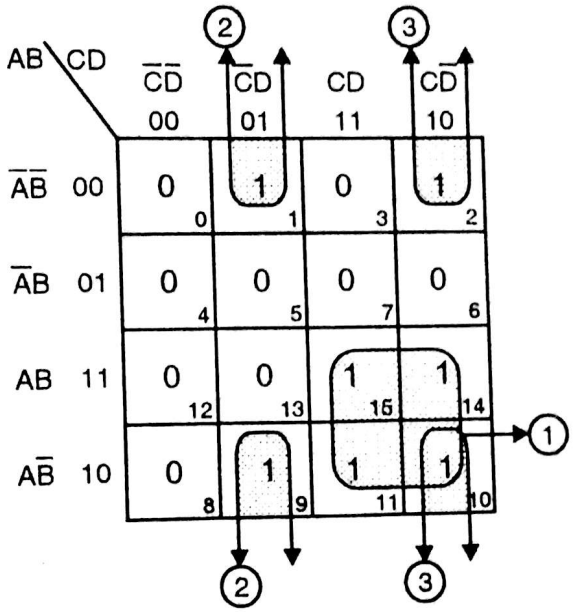
$$Y = \underset{(1)}{(A_0 + \bar{A}_3)}\ \underset{(2)}{(A_3 + A_2 + \bar{A}_1)}\ \underset{(3)}{(A_3 + \bar{A}_1 + \bar{A}_0)}$$

...Ans.

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

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

**Ans. :**



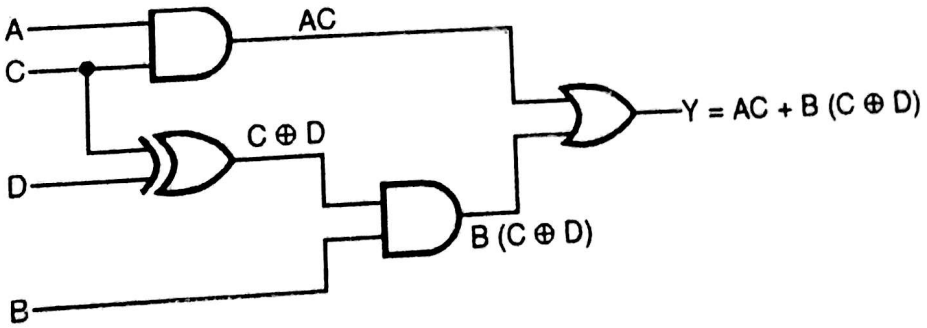

∴ Output equation, $Y = AC + \bar{B}\bar{C}D + \bar{B}C\bar{D}$

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

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

...Ans.

Implementation of above equation using any 2 input gates.




...Ans.

**Fig. Q. 4.20**

**(June 99, 6 Marks)**

**21. Draw K–map for the following expression :**

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

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

Scanned by CamScanner

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

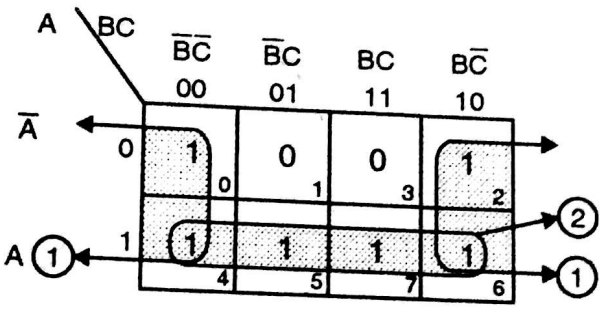
$$f(A, B, C, D) = \sum m\,(1, 3, 5, 10, 11, 12, 13, 14, 15)$$ **(June 99, 10 Marks)**

**Ans.:** Refer Section 4.9, Ex. 25.

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

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

**Ans. :**




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

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

**Ans. :**

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

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

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

As $(A + A = A)$

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

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

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

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

But Y in standard SOP form is,

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

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

**Ans. :** From K–map

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

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

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

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

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

**Now implement above equation using NAND gates only.**

Scanned by CamScanner

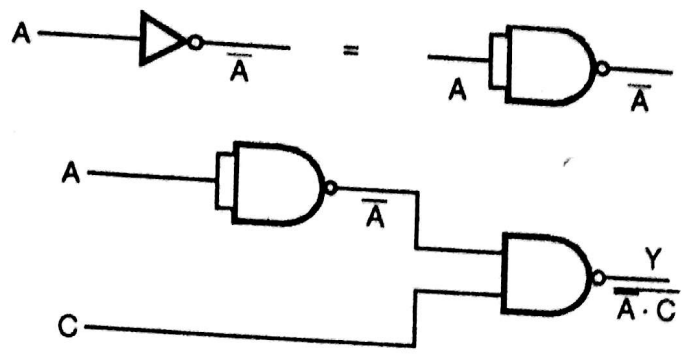
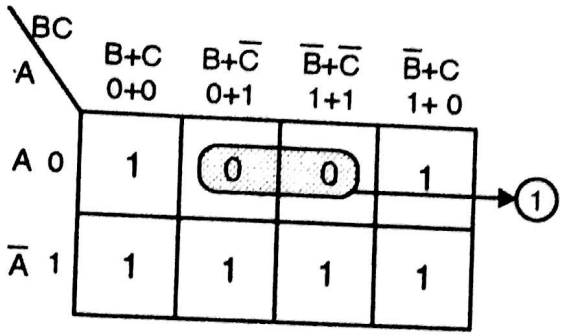



**Fig. Q.4.23 (a)**

**(d) Express Y in standard POS form.**

**Ans. :** The K-map for Y is, **(Dec. 99, 4 Marks)**




Making groups of zeros we get,

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

In standard POS form,

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

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

**...Ans.**

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

**Ans. :** From K–map, $Y = A + \overline{C}$

As $\overline{\overline{A}} = A$

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

**...Ans.**

**Implementation of above equation using NOR gates only.**

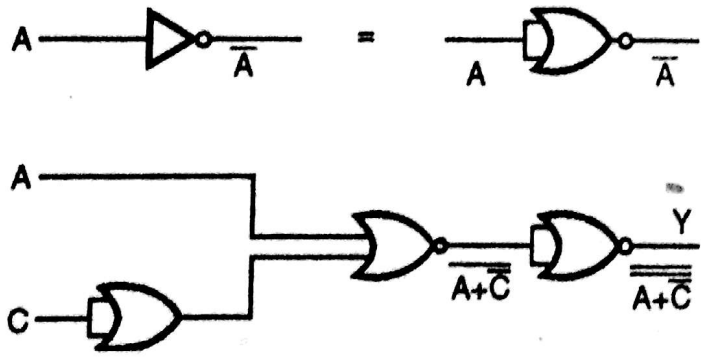



**Fig. Q.4.23 (b)**

Scanned by CamScanner

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

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

**Ans.:**

**Step I :** List all the minterms.

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

**Table 36**

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

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

**Table 37**

Scanned by CamScanner

**Step III :** Using Table 37 prepare first reduction table

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

**Table 38**

Scanned by CamScanner

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

**Table 39**

**Step V :** From Table 39 prepare third reduction table

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

**Table 40**

Scanned by CamScanner

Prime implicant table.

| Prime implicants | $m_8$ | $m_9$ | $m_{10}$ | $m_{11}$ | $m_{13}$ | $m_{15}$ | $m_{16}$ | $m_{18}$ | $m_{21}$ | $m_{24}$ | $m_{25}$ | $m_{26}$ | $m_{27}$ | $m_{30}$ | $m_{31}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P_A$ = 010XX | X | X | X | X | | | | | | X | X | | | | |
| $P_B$ = X10X0 | X | X | | | | | | | | X | X | | | | |
| $P_C$ = X1X10 | X | | X | | | | | | | X | | X | | | |
| $P_D$ = 1X0X0 | | | | | | | X | X | | X | | X | | | |
| $P_E$ = 01XX1 | | X | | X | X | X | | | | | | | | | |
| $P_F$ = X1X11 | | X | | X | | | | | | | X | | X | | |
| $P_G$ = X1011X | | | X | X | | | | | | | | X | X | | |
| $P_H$ = 11X1X | | | | | | | | | | X | X | X | X | | |
| $P_I$ = X1X01 | | | | X | | X | | | | | | | X | | X |
| $P_J$ = 11X1X | | | | | | | | | | | | X | X | X | X |

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

∴ Final reduction

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

$$f(A, B, C, D, E) = \bar{A}B\bar{C} + A\bar{C}\bar{E} + \bar{A}BE + ABD + AB\bar{C}\bar{D}E$$ **...Ans.**

❑❑❑

Scanned by CamScanner

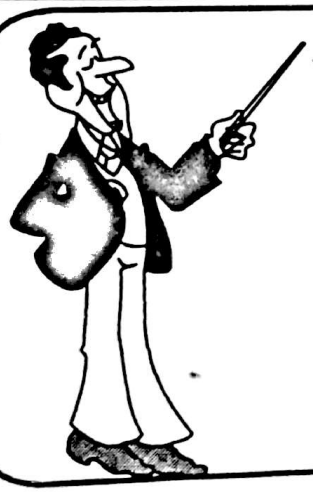
# Combinational Logic Circuits

5

**Syllabus :**

Mixed logic combinational circuits, Multiple output functions

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Introduction | 5.1 | ✓ | ✓ |
| • Multi–input multi–output combinational circuit | 5.2 | – | ✓ |
| • Code conversion | 5.3 | – | ✓ |



**Comments**

- Very important chapter.
- Don't skip any section.

Scanned by CamScanner

## 5.1 Introduction :

At this stage, you are aware with numbering system, codes, boolean expression, logic gates, reduction technique using K map, Quine Mc-cluskey method etc. Now in this chapter, we are going to use this knowledge for different applications in Digital Electronics. But before this, I would like to inform you that, in digital, we can configure any circuit under either sequential or combinational circuit.

Imp

**Digital Electronics**

| **Combinational Circuit** | **Sequential Circuit** |
|---|---|
| (1) Combinational circuit is one whose output at a specified time is function of the inputs at that time. | (1) Sequential circuit is a logic circuit whose output at specified time is a function of the inputs at that time as well as finite number of inputs of preceding time. |
| (2) Output of combinational circuit depends purely upon *PRESENT* input, therefore it doesn't require *Memory* to store past inputs (preceding inputs). | (2) Sequential circuit requires *Memory element*, because its output depends upon present as well as preceding inputs. |
| | (3) Basically sequential circuit is combination of *Memory element and Combinational circuit.* |

Presently we are going to concentrate upon ***Combinational logic circuits.***

### Combinational Logic Design :

In chapter 3 i.e. Boolean Algebra and Logic Gates, every time function was given to you (either specifying maxterm or minterm) and you were asked to minimise it. But the question is, *from where this function is derived ? Where is the origin of the function ?*

The answer for the question is, actually what you learned last time was just base plateform for designing combinational circuits. Let's see steps involved in designing.

(1) You will be given a problem.
(2) You analyse the problem and define some variables.
(3) After defining variables, you prepare truth table.
(4) From the truth table you get function interms of maxterm or minterm.
(5) Now apply K-map or any, minimisation technique and derive boolean expression, providing minimum hardware.

Now in this chapter you have to perform step 1 to 5, very correctly. For understanding let's solve one example.

**Ex. 1 :** As shown in Fig. 5.1, a combinational logic circuit is required for following specifications : **(Dec. 96, 10 Marks)**

Scanned by CamScanner

Inputs are four lines $A_3$, $A_2$, $A_1$, $A_0$ ($A_3$ = MSB), carry binary equivalent of decimal digit 0 – 9.
Output Y is '1' when input contains two or more bits '1', otherwise output Y = 0
Implement necessary logic using Nor gate.

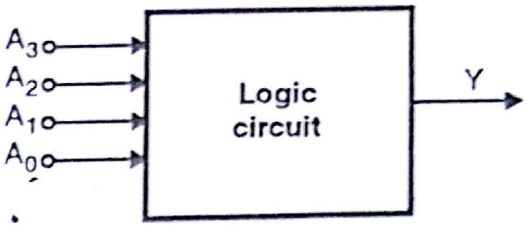



Fig. 5.1

**Soln. :** (1) Read statement carefully and conclude some points :

(a) No. of inputs lines are $A_3$, $A_2$, $A_1$, $A_0$ i.e. 4.

(b) Number of input combinations for 4 bits will be $2^n = 2^4 = 16$ (n = no. of bits)

(c) But in the problem it is mentioned that it carries binary equivalent of decimal digit 0–9. Therefore, remaining states i.e. 10 to 15 will be treated as ***Don't Care Condition***.

(d) Output Y depends upon number of 1's in nibble $A_3 - A_0$. i.e. Say, if $A_3\ A_2\ A_1\ A_0 = (0001)_2$. Number of 1'S are only one, therefore less than 2, therefore Output Y for $(0001)_2$ is equal to '0'. But if $A_3\ A_2\ A_1\ A_0 = (0111)_2$, then number of 1'S are 3 i.e. greater than 1, therefore Output Y = '1'.

(2) Now prepare truth table for this example. Truth table will contains two column Input(s) and Output(s). Under input(s) we have $A_3$, $A_2$, $A_1$, $A_0$ and output is only Y.

| Decimal | Input(s) | | | | Output |
|---|---|---|---|---|---|
| | $A_3$ | $A_2$ | $A_1$ | $A_0$ | Y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | X |
| 11 | 1 | 0 | 1 | 1 | X |
| 12 | 1 | 1 | 0 | 0 | X |
| 13 | 1 | 1 | 0 | 1 | X |
| 14 | 1 | 1 | 1 | 0 | X |
| 15 | 1 | 1 | 1 | 1 | X |

Don't care condition because input changes from '0' to '9' ONLY

**Table 1**

(3) Now draw K map, encircle '1's, therefore answer will be in SOP form.

Scanned by CamScanner

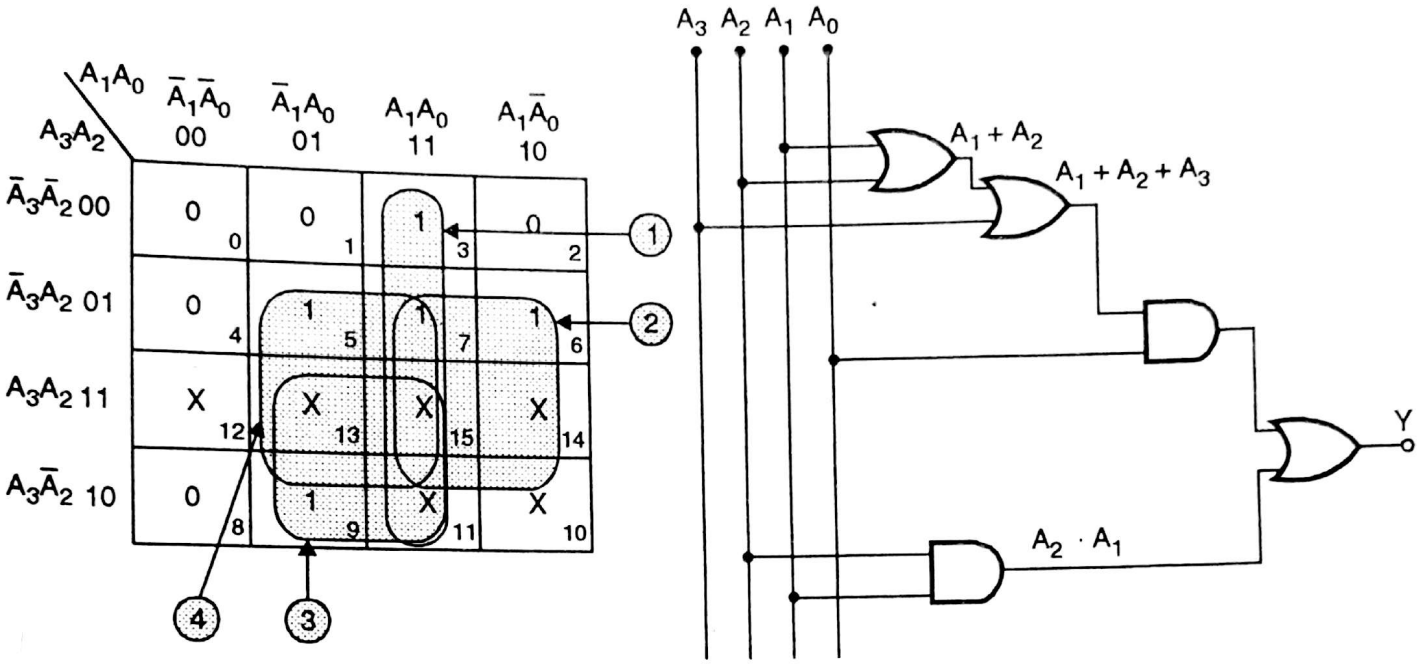



**Fig. 5.2**

Output Y = $A_1 A_0 + A_2 A_1 + A_3 A_0 + A_2 A_0$

(1) (2) (3) (4)

= $\boxed{A_0 (A_1 + A_2 + A_3) + A_2 A_1}$

We have to implement if using only NOR gate.

$$= \overline{\overline{A_0 (A_1 + A_2 + A_3) + A_2 A_1}} = \overline{\overline{A_0 . (A_1 + A_2 + A_3)} \; . \; \overline{A_2 A_1}}$$

$$= \overline{\bar{A}_0 + (\overline{A_1 + A_2 + A_3}) . (\bar{A}_2 + \bar{A}_1)} = \overline{\bar{A}_0 + (\overline{A_1 + A_2 + A_3}) . (\bar{A}_2 + \bar{A}_1)}$$

$$= \overline{\bar{A}_0 + \overline{A_1 + A_2 + A_3}} + \overline{\bar{A}_2 + \bar{A}_1}$$

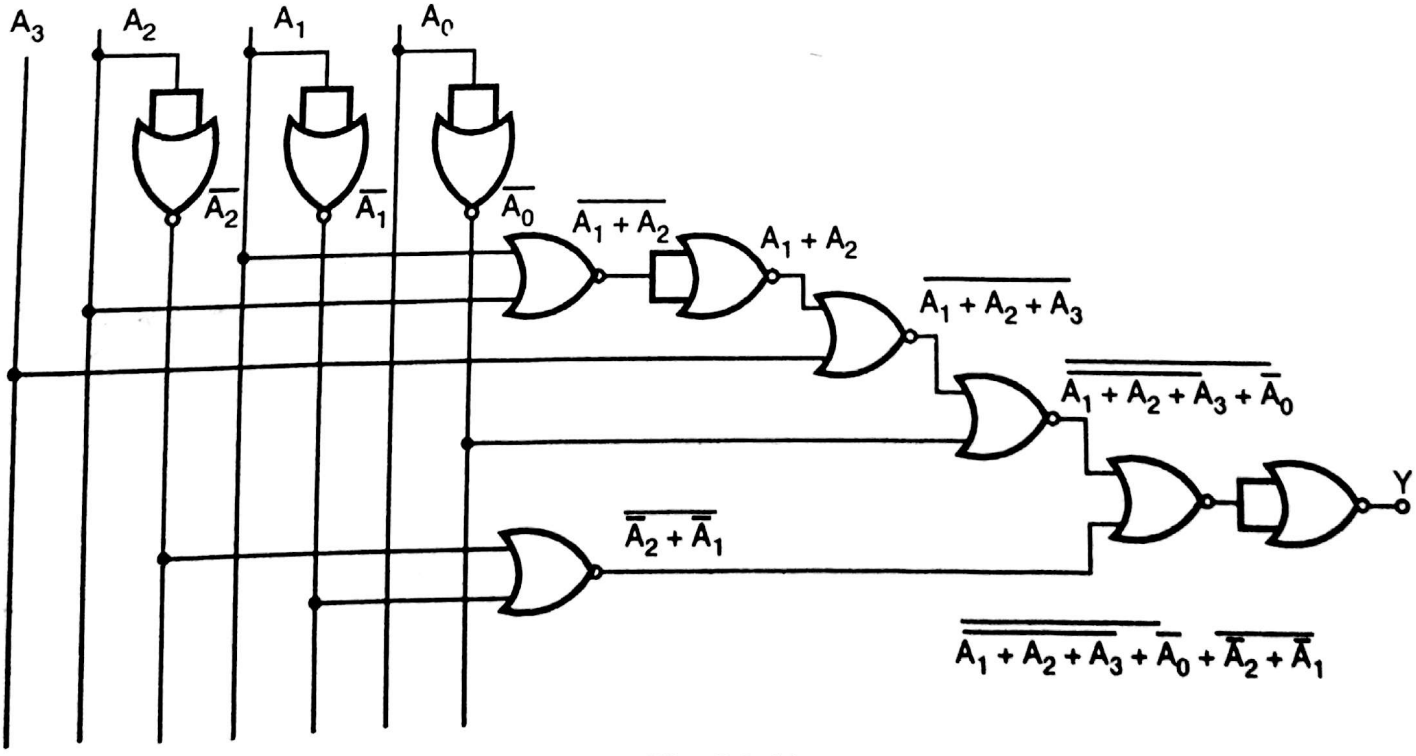



**Fig. 5.2 (c)**

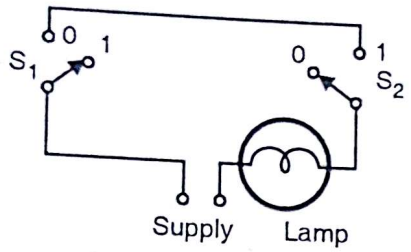
Scanned by CamScanner

**Ex. 2 :** A staircase light is controlled by two switches, one at the top and another at bottom of staircase as shown in Fig. 5.3.

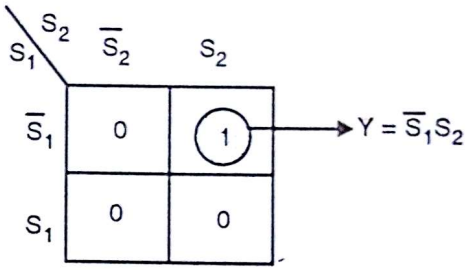
(i) Make a truth table

(ii) Write the logic equation in NAND NAND form

(iii) Realize the circuit using AND–OR gates.




(a)




(b)

**Fig. 5.3**

**Soln. :** Truth table is,

| Inputs | | Outputs |
|---|---|---|
| $S_1$ | $S_2$ | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**Table 2**

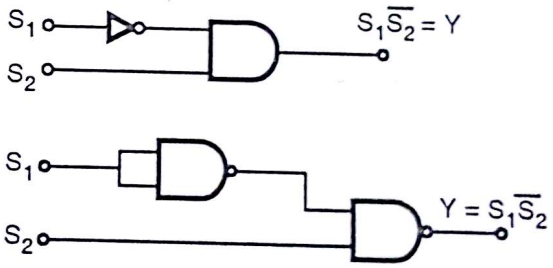



**Fig. 5.3 (c)**

To find NAND–NAND form, put double inversion.

$$\overline{\overline{Y}} = \overline{\overline{S_1 \bar{S}_2 + \bar{S}_1 S_2}} = \overline{\overline{S_1\bar{S}_2}\ \ \overline{\bar{S}_1 S_2}}$$

Implement using AND–OR gate is shown in Fig. 5.3 (c).

LIBRARY
743

**Ex. 3 :** A 4 bit binary number is represented as $A_3\ A_2\ A_1\ A_0$ where $A_3$, $A_2$, $A_1$ and $A_0$ represent the individual bits with $A_0$ equals to the LSB. Design a logic circuit that will produce a HIGH output whenever binary number is greater than $(0010)_2$ and less than $(1000)_2$.

**(June 99, 2 Marks)**

Scanned by CamScanner

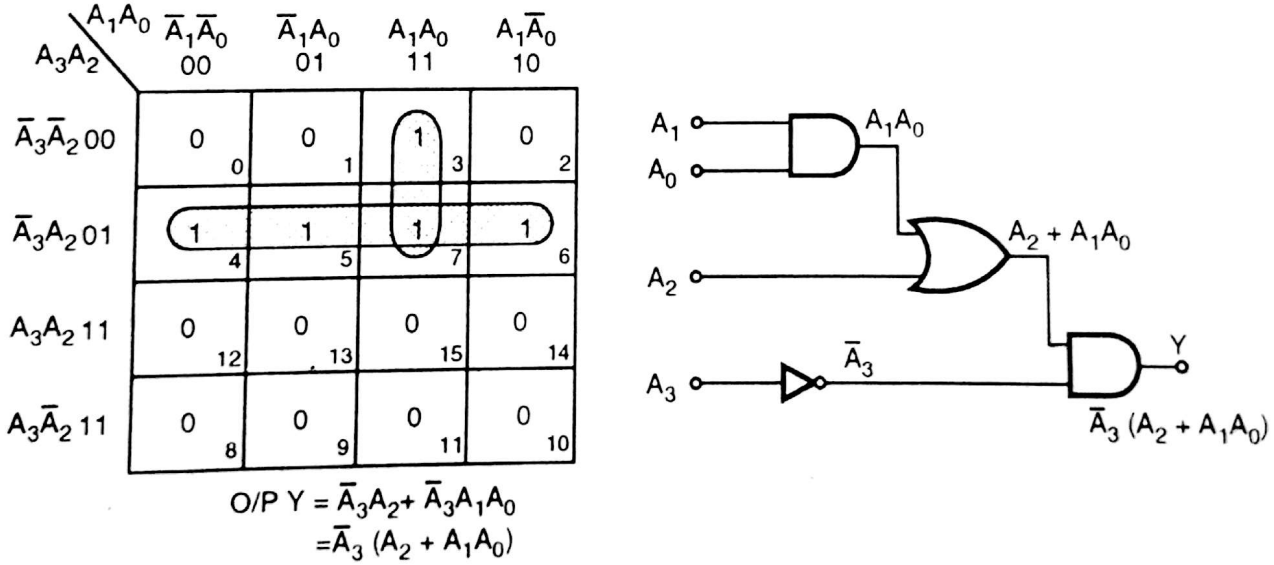
**Soln. :** The truth table is,

| Decimal | Inputs | | | | Output |
|---|---|---|---|---|---|
| | $A_3$ | $A_2$ | $A_1$ | $A_0$ | Y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 0 |
| 12 | 1 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 0 |

Number within $(0010)_2$ and $(1000)_2$

**Table 3**

Output is logic high when binary number is greater than (not equal to) $(0010)_2$ and less than (not equal to) $(1000)_2$.




**(a)** **(b)**

**Fig. 5.4**

**Ex. 4 :** Design a logic circuit whose output is HIGH only when a majority of inputs A, B, C are LOW.

**Soln. :** (1) There are three variables A, B and C.

(2) Number of states will be $2^n = 2^3 = 8$.

Scanned by CamScanner

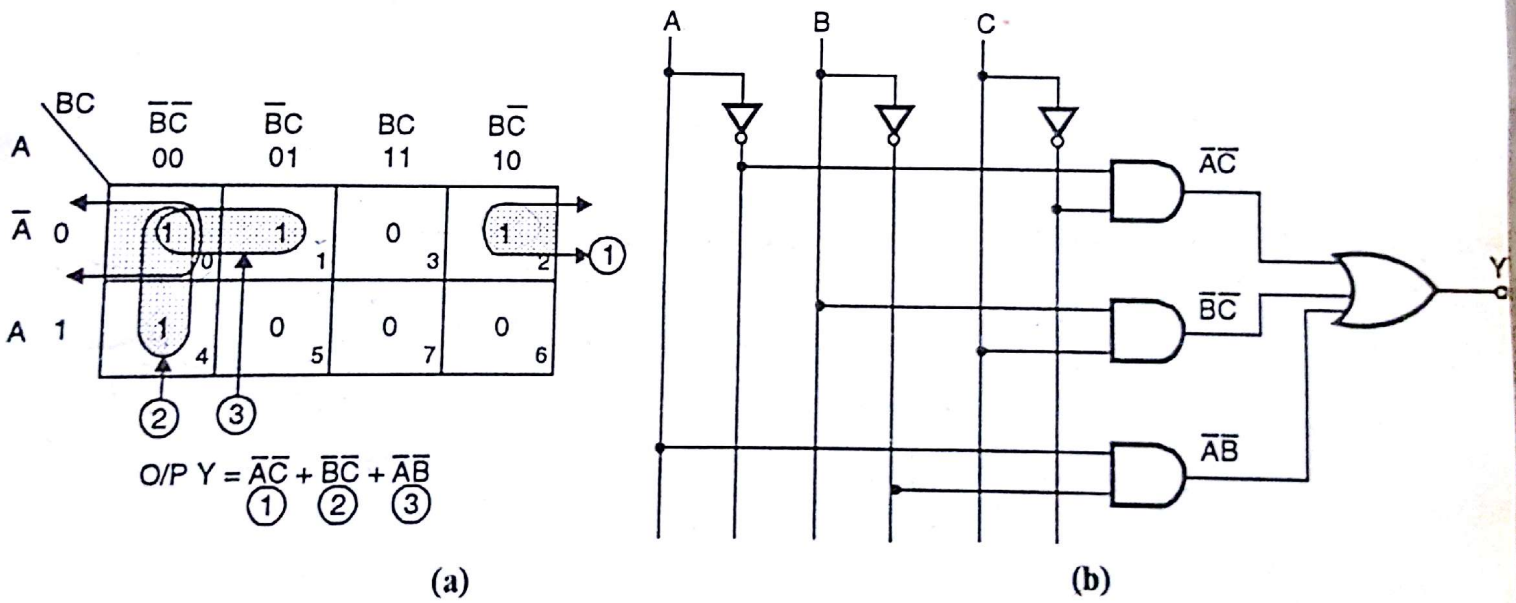
(3) We have to check for more number of logic '0' then logic 1.

(4) Truth table is,

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | C | Y | |
| 0 | 0 | 0 | 1 | → |
| 0 | 0 | 1 | 1 | → More '0"s are present |
| 0 | 1 | 0 | 1 | → (Majority) |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | → |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 0 | |

Table 4

(5) **K–map :**




(a) (b)

Fig. 5.5

**Ex. 5 :** Design a logic circuit whose output is HIGH when input is > 9. Assume that input to the circuit is 4 bit binary $A_3\ A_2\ A_1\ A_0$.

**Soln. :** (1) Variables are four $A_3$, $A_2$, $A_1$, $A_0$

(2) Therefore number of states = $2^n = 2^4 = 16$.

(3) We have to check for condition that input is greater than 9. At that time output will be HIGH. Here ***equal to*** condition is not mentioned, therefore Output Y will not be HIGH when input is equal to 9.

Scanned by CamScanner

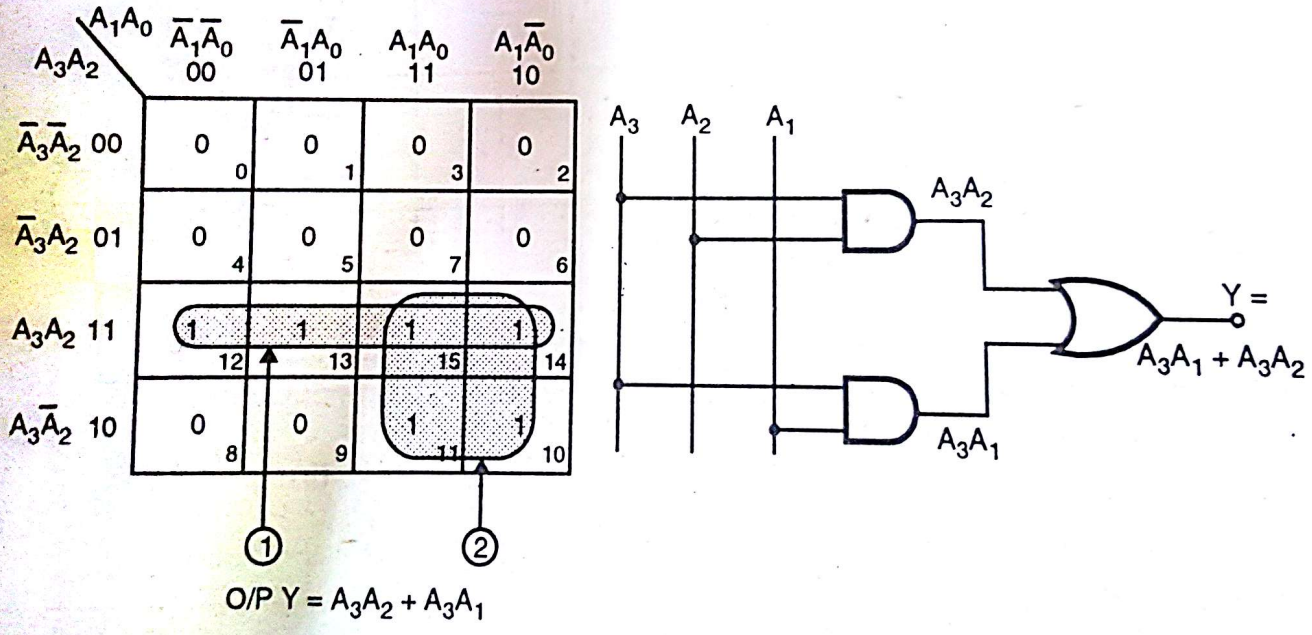
(4) This circuit is used for BCD addition, for checking, if the final answer is crossing boundary of 9.

(5) **Truth Table**

| Decimal | Inputs | | | | Output | |
|---|---|---|---|---|---|---|
| | $A_3$ | $A_2$ | $A_1$ | $A_0$ | Y | |
| 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 1 | 0 | |
| 2 | 0 | 0 | 1 | 0 | 0 | |
| 3 | 0 | 0 | 1 | 1 | 0 | |
| 4 | 0 | 1 | 0 | 0 | 0 | |
| 5 | 0 | 1 | 0 | 1 | 0 | |
| 6 | 0 | 1 | 1 | 0 | 0 | |
| 7 | 0 | 1 | 1 | 1 | 0 | |
| 8 | 1 | 0 | 0 | 0 | 0 | |
| 9 | 1 | 0 | 0 | 1 | 0 | |
| 10 | 1 | 0 | 1 | 0 | 1 | Number > 9 |
| 11 | 1 | 0 | 1 | 1 | 1 | |
| 12 | 1 | 1 | 0 | 0 | 1 | |
| 13 | 1 | 1 | 0 | 1 | 1 | |
| 14 | 1 | 1 | 1 | 0 | 1 | |
| 15 | 1 | 1 | 1 | 1 | 1 | |

**Table 5**

**(6) K–map :**




(a) (b)

**Fig. 5.6**

**Ex. 6 :** The input to a combinational logic circuit is a valid single digit BCD data. Design the logic circuit using minimum hardware to detect whenever a number greater than 5 appears at the input.

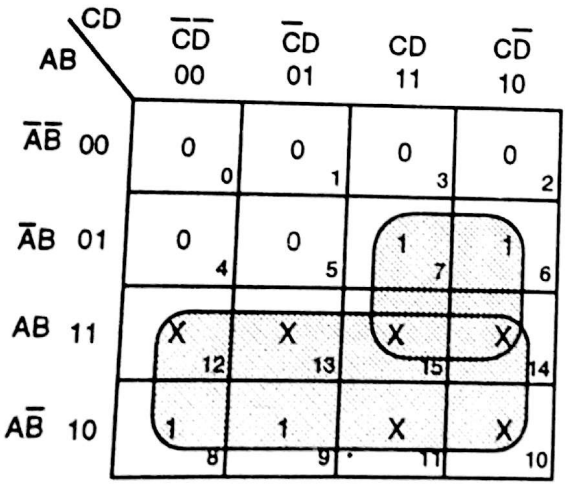
Scanned by CamScanner

**Soln. :** (1) Single digit BCD data requires 4 bits. Say ABCD.
(2) But as data input variables are four, number of states will be $2^4 = 16$.
(3) But for BCD input greater than 9 is invalid, therefore those states are taken as ***Don't Care*** (X) because we know that these states will not appear.
(4) We have to see that number great than 5 occurs.
(5) Truth Table,

| Decimal | Inputs | | | | Output | |
|---|---|---|---|---|---|---|
| | A | B | C | D | Y | |
| 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 1 | 0 | |
| 2 | 0 | 0 | 1 | 0 | 0 | |
| 3 | 0 | 0 | 1 | 1 | 0 | |
| 4 | 0 | 1 | 0 | 0 | 0 | |
| 5 | 0 | 1 | 0 | 1 | 0 | |
| 6 | 0 | 1 | 1 | 0 | 1 | |
| 7 | 0 | 1 | 1 | 1 | 1 | Numbers greater than 5 |
| 8 | 1 | 0 | 0 | 0 | 1 | |
| 9 | 1 | 0 | 0 | 1 | 1 | |
| – | 1 | 0 | 1 | 0 | x | Invalid BCD codes |
| – | 1 | 0 | 1 | 1 | x | |
| – | 1 | 1 | 0 | 0 | x | |
| – | 1 | 1 | 0 | 1 | x | |
| – | 1 | 1 | 1 | 0 | x | |
| – | 1 | 1 | 1 | 1 | x | |

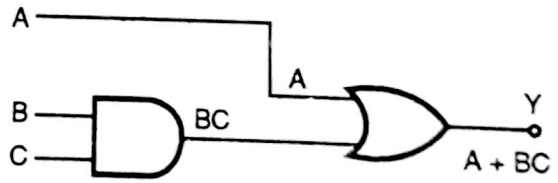
**Table 6**

**(6) K–map :**




$Y = A + BC$

**(a)**




**(b)**

**Fig. 5.7**

**Ex. 7 :** Fig. 5.8 shows diagram for an automobile alarm circuit used to detect certain undesirable conditions. The three switches are used to indicate the status of the door by the driver's seat, the ignition and the headlights respectively. Design the logic circuit with these switches as inputs so that the alarm will be activated whenever either of the following conditions exists:

Scanned by CamScanner

(a) The headlights are ON while ignition OFF.

(b) The door is open while ignition is ON.

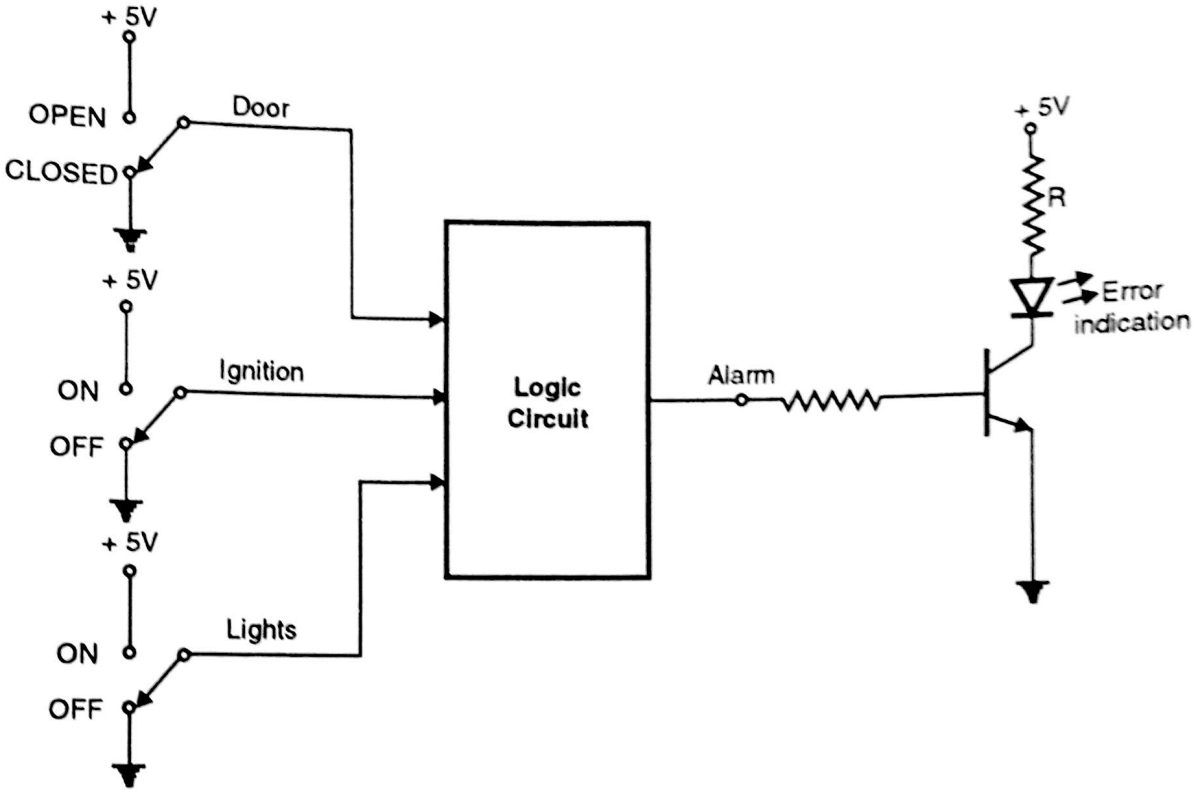



**Fig. 5.8**

**Soln. :** (1) Conclusion after reading problem carefully. There are mainly three variables.

(a) Door, (b) Ignition, (c) Head light.

Let's say Door → Variable A

Ignition → Variable B

Head light → Variable C

(2) Switch ON ⇒ Logic '1'

Switch OFF ⇒ Logic '0'

(3) Door open ⇒ Logic '1'

Door closed ⇒ Logic '0'.

(4) Basically user is interested in only mentioned two conditions in the problem. But if we write truth table. We are going to get total $2^n = 2^3 = 8$. So just go on checking for,

(a) Head light → ON → logic '1', while

Ignition → OFF → logic '0'

(Here you need not check Door open/closed)

(b) Door open → logic '1' while

Ignition ON → logic '1'.

(Here you need not check head lights)

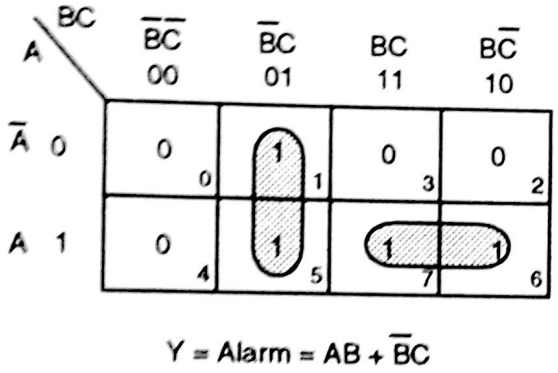
Scanned by CamScanner

(5) Truth table is,

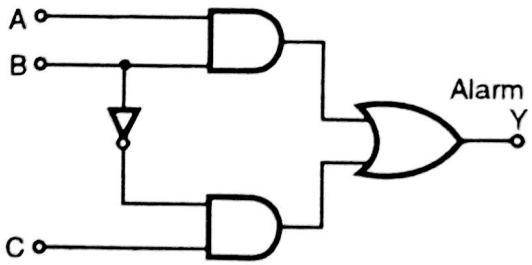
| Inputs | | | Output | |
|---|---|---|---|---|
| Door | Ignition | Headlight | Alarm | |
| A | B | C | Y | |
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 1 | → Ignition OFF, headlight ON |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | → Ignition OFF, headlight ON |
| 1 | 1 | 0 | 1 | → Door open |
| 1 | 1 | 1 | 1 | → Ignition ON |

**Table 7**

**(6) K–Map :**




(a)




(b)

**Fig. 5.9**

**Ex. 8 :** The input to a combinational logic circuit is a valid single digit 7421 BCD data. Design the logic circuit using minimum hardware to detect when an even number appears on input.

**Soln. :** Truth table is,

| Decimal | Input | | | | | Output |
|---|---|---|---|---|---|---|
| | $B_3$ | $B_2$ | $B_1$ | $B_0$ | | Y |
| | 7 | 4 | 2 | 1 | BCD | |
| 0 | 0 | 0 | 0 | 0 | | 1 |
| 1 | 0 | 0 | 0 | 1 | | 0 |
| 2 | 0 | 0 | 1 | 0 | | 1 |
| 3 | 0 | 0 | 1 | 1 | | 0 |
| 4 | 0 | 1 | 0 | 0 | | 1 |
| 5 | 0 | 1 | 0 | 1 | | 0 |
| 6 | 0 | 1 | 1 | 0 | | 1 |
| 7 | 1 | 0 | 0 | 0 | | 0 |
| 8 | 1 | 0 | 0 | 1 | | 1 |
| 9 | 1 | 0 | 1 | 0 | | 0 |

**Table 8**

Scanned by CamScanner

Since a valid BCD is only up to 9, the other remaining combinations can be treated as don't care conditions.

The K map is shown in Fig. 5.10

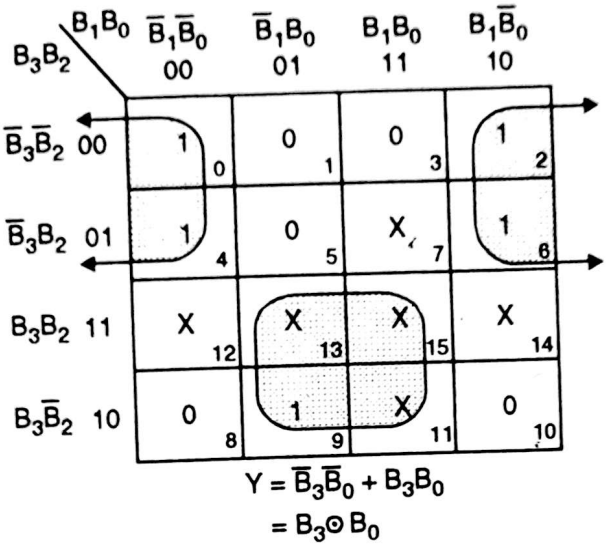



Fig. 5.10




Fig. 5.10(A)

**Ex. 9 :** A circuit is to be designed that has one control line and three data lines. When control line is low, the circuit should detect whether at least two of the data lines are simultaneously high. When control line is high, output lines should be low.

**Soln. :** Table 9 shows the truth table.

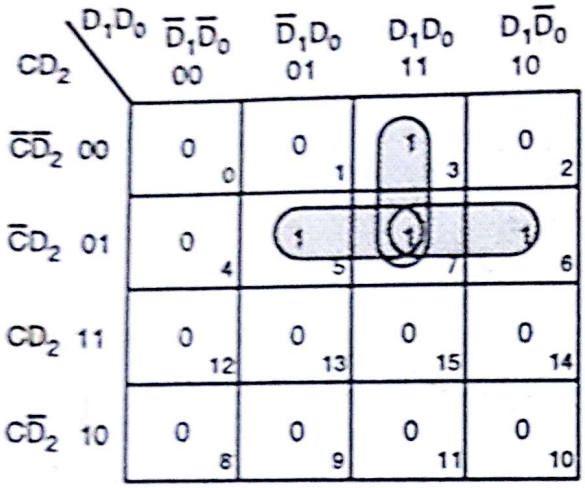
| Input | | | | Output |
|---|---|---|---|---|
| C | $D_2$ | $D_1$ | $D_0$ | Y |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

C = Control line
D = Data lines

**Table 9**

The K map is illustrated in Fig. 5.11 (a) . The circuit diagram is illustrated in Fig. 5.11(b)

Scanned by CamScanner




O/P $Y = \bar{C}D_2(D_0 + D_1) + \bar{C}D_1D_0$

Fig. 5.11(a)

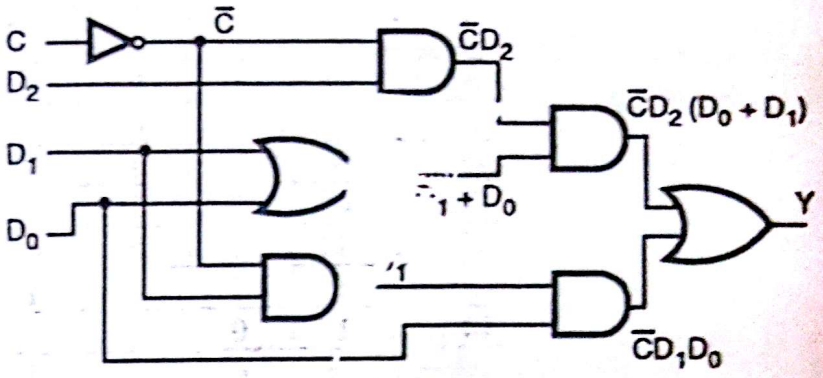



Fig. 5.11(b)

---

**Ex. 10 :** The conditions under which an insurance company will issue a policy are :

(i) A married female 25 years old or older, or
(ii) A female under 25 years or
(iii) A married male under 25 years with no accident record, or
(iv) A married male with no accident record, or
(v) A married male under 25 years with accident record.

Obtain a simplified logic expression stating to whom a policy can be issued.

**Soln. :** The input variables include :

(i) Male / female, (ii) Married / unmarried
(iii) Age ≥ 25 years, (iv) Accident record.

We assume the following conditions :

(a) $A = 0$ Male
$= 1$ female
(b) $B = 0$ Unmarried
$1$ Married
(c) $C = 0$ age < 25 years
$= 1$ age ≥ 25 years
(d) $D = 0$ No accident
$= 1$ Accident.

The truth table is illustrated in Table 10 and 11.

| | Input | | | | Output |
|---|---|---|---|---|---|
| | A | B | C | D | Y |
| (i) | 1 | 1 | 1 | X | 1 |
| (ii) | 1 | X | 0 | X | 1 |
| (iii) | 0 | 1 | 0 | 0 | 1 |
| (iv) | 0 | 1 | X | 0 | 1 |
| (v) | 0 | 1 | 1 | 1 | 1 |

Table 10

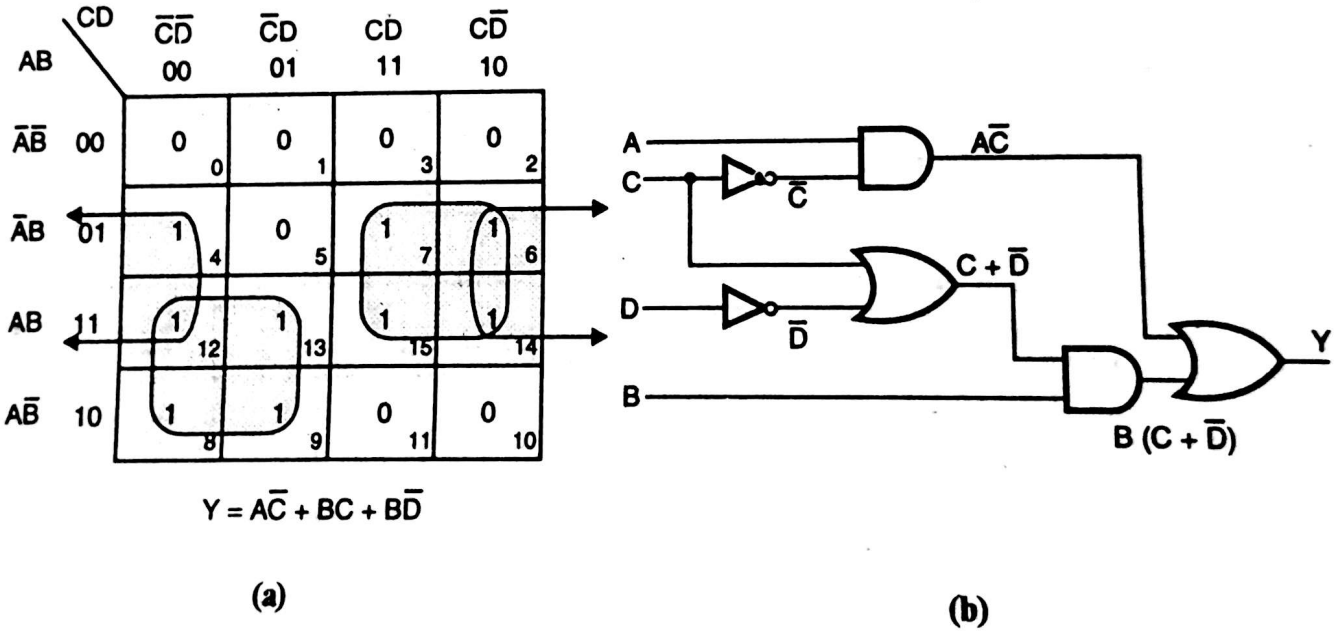
Scanned by CamScanner

Therefore we get,

| Decimal | A | B | C | D | Y | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 1 | 0 | |
| 2 | 0 | 0 | 1 | 0 | 0 | |
| 3 | 0 | 0 | 1 | 1 | 0 | |
| 4 | 0 | 1 | 0 | 0 | 1 | (iii) / (iv) |
| 5 | 0 | 1 | 0 | 1 | 0 | |
| 6 | 0 | 1 | 1 | 0 | 1 | (iv) |
| 7 | 0 | 1 | 1 | 1 | 1 | (v) |
| 8 | 1 | 0 | 0 | 0 | 1 | (ii) |
| 9 | 1 | 0 | 0 | 1 | 1 | |
| 10 | 1 | 0 | 1 | 0 | 0 | |
| 11 | 1 | 0 | 1 | 1 | 0 | |
| 12 | 1 | 1 | 0 | 0 | 1 | (ii) |
| 13 | 1 | 1 | 0 | 1 | 1 | |
| 14 | 1 | 1 | 1 | 0 | 1 | (i) |
| 15 | 1 | 1 | 1 | 1 | 1 | |

**Table 11**

The K–map is shown in Fig. 5.12(a)




(a) (b)

**Fig. 5.12**

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

**Ex. 11 :** Fig. 5.13 shows four switches that are part of the control circuitry in a copy machine. The switches are at various points along the path of the copy paper as the paper passed through the machine. Each switch is normally open and as the paper passes over a switch, the switch closes. It is impossible for switches $S_1$ and $S_4$ to be closed at the same time. Design the logic circuit to produce a HIGH output whenever two or more switches are closed at a time. Use K–map and take advantage of ***don't care*** condition.

Scanned by CamScanner

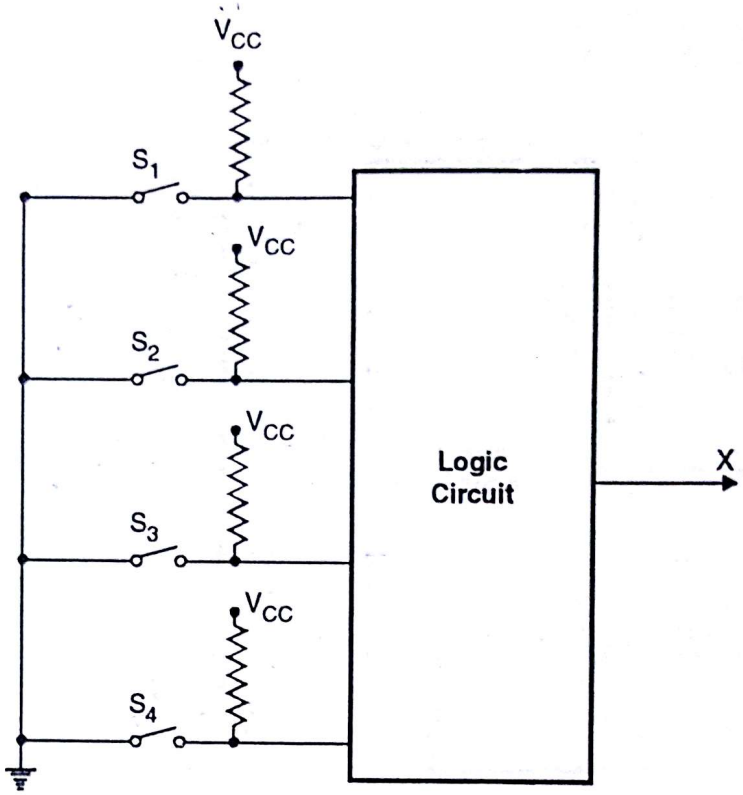



**Fig. 5.13**

**Soln. :** (1) As shown in Fig. 5.13.

Switch open → Input to logic circuit is logic '1'
Switch closed → Input to logic circuit is logic '0'
∴ Switch open → logic '1'
Switch closed → logic '0'

(2) There are four switches $S_1$, $S_2$, $S_3$, $S_4$ : four variables.

(3) Number of states will be $16 = 2^4$.

(4) First check for $S_1$ and $S_4$ closed (i.e. logic '0') and put don't care at the output column, because as mentioned in problem, $S_1$ and $S_4$ closed is impossible.

(5) Now check for two or more keys pressed, or in short check for two or more '0's, in a row.

(6) Truth Table.

| Inputs | | | | Output |
|---|---|---|---|---|
| $S_4$ | $S_3$ | $S_2$ | $S_1$ | Y |
| 0 | 0 | 0 | 0 | X |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | X |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | X |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | X |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

**Table 12**

Scanned by CamScanner

(7) **K–Map :** Refer Fig. 5.14.

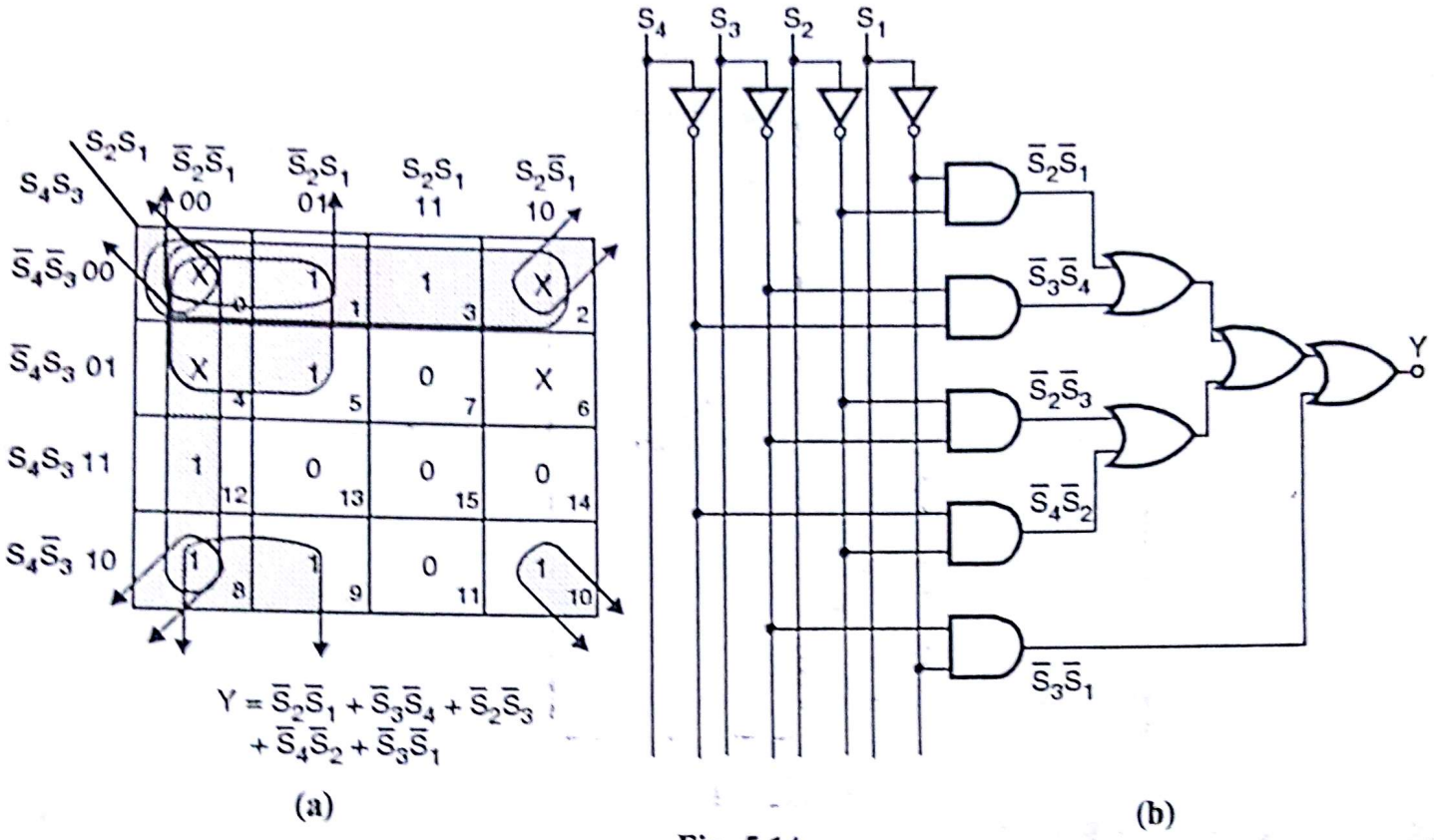



Fig. 5.14

**Ex. 12 :** Design a combinational circuit such that it has four inputs WXYZ and one output F. $D_1$ and $D_2$ represent two binary numbers each consisting of two bits WX and YZ respectively. The output F is to be HIGH only if $|D_1 - D_2| < 2$. Write the truth table.

Using K map reduction technique, find a minimum sum of product form and implement using ALL NAND logic.

**Soln. :** The truth table is illustrated in Table 13.

| | Input | | | | Output |
|---|---|---|---|---|---|
| Decimal | W | X | Y | Z | F |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 1 |

Table 13

Scanned by CamScanner

The K map is illustrated in Fig. 5.15.

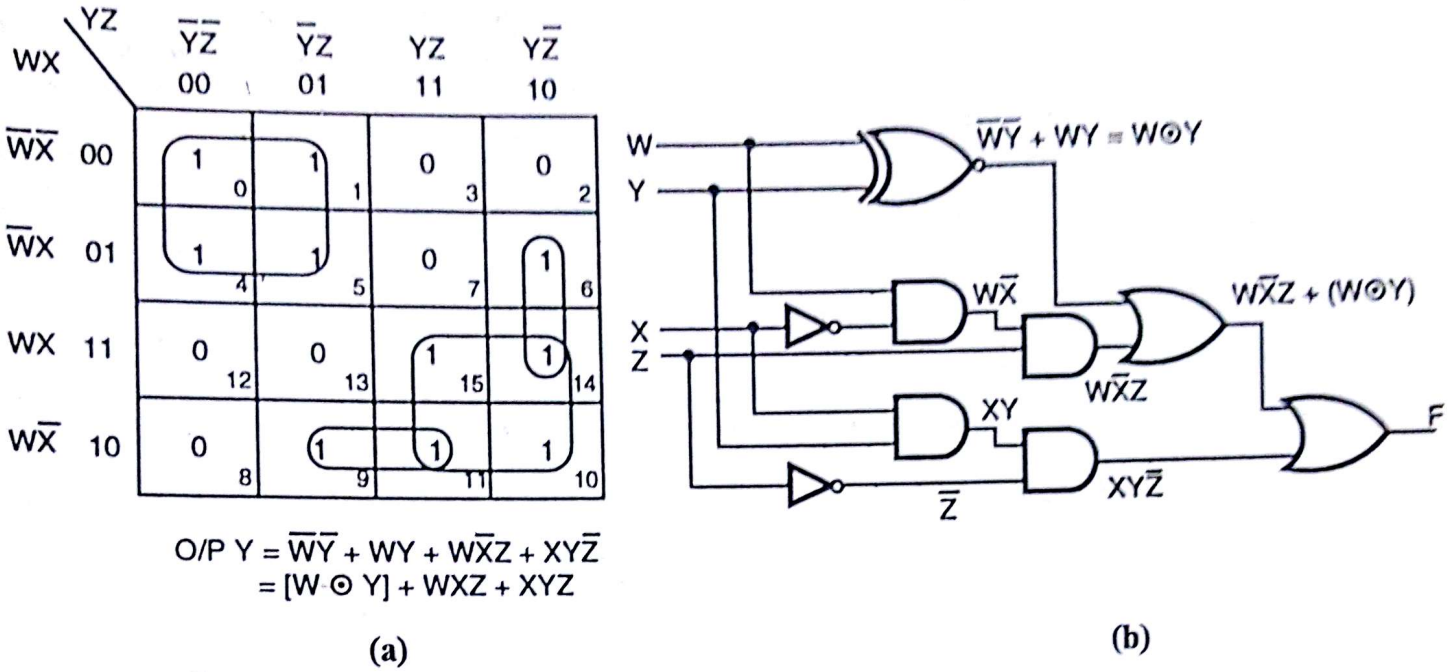



**Fig. 5.15**

To implement using only NAND, Output Y can be written as,

$$\text{Output } Y = \overline{\overline{W}\,\overline{Y} + WY + W\overline{X}Z + XY\overline{Z}}$$

$$= \overline{\overline{\overline{W}\,\overline{Y}} \cdot \overline{WY} \cdot \overline{W\overline{X}Z}\;\overline{XY\overline{Z}}}$$

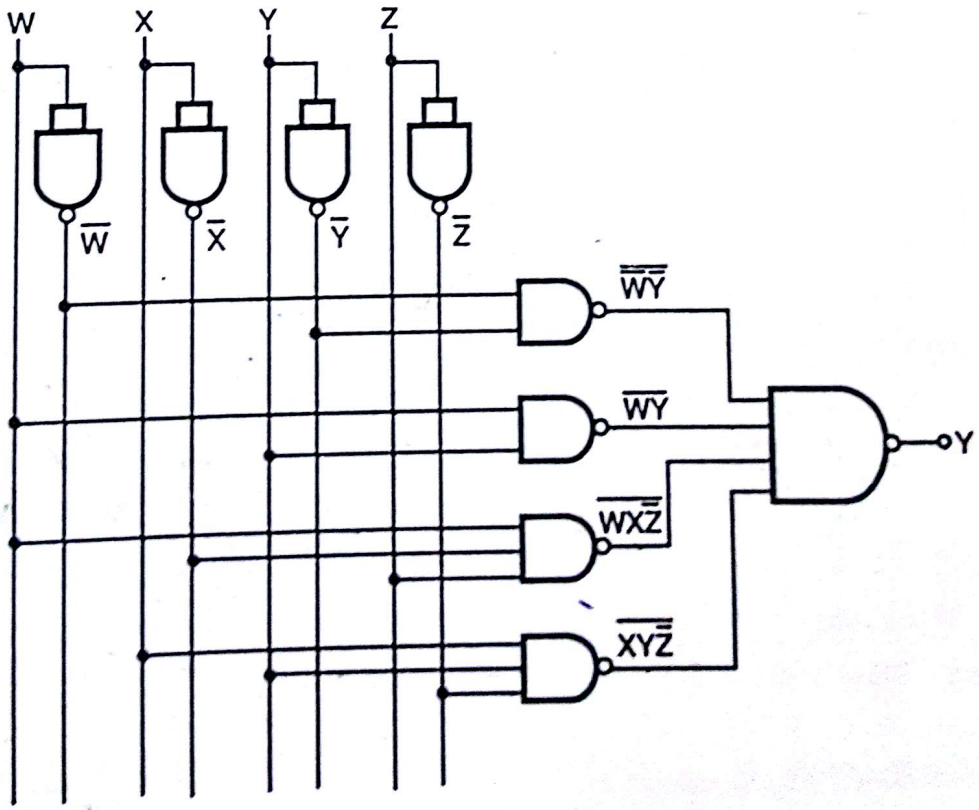



**Fig. 5.15 (c)**

Scanned by CamScanner

**Ex. 13 :** An aeroplane is to be equipped with a warning system that alerts the pilot under certain conditions of danger. The warning system monitors three instruments. These instruments indicate altitude, air speed and state of landing wheels. a landing warning lamp should glow if the air speed is less than the cruising speed and landing wheels are not down. If wheels are down and the pilot is at proper landing altitude, but not at the proper landing speed. Assume that only the logic level voltages are to be produced by the warning system.

Relationships between logic levels and different states are as below :

Altitude, A = 0 Landing altitude, = 1 Cruising altitude

Wheels, W = 0 Wheels up, = 1 Wheels down

Air speed, $S_1S_0$ = 0 0 for slow

0 1 Landing speed

1 0 Cruising speed

1 1 Too fast

Warning lamp, L = 0 Lamp off, = 1 Lamp on.

**Soln. :** Warning conditions

(i) Air speed < Cruising speed.
Wheels up.

(ii) Landing altitude proper
Landing speed not proper
Wheels down.

In both these cases, warning lamp must glow i.e., A,W, $S_1S_0$ are input variables and L is output variable. The truth table is shown in Table 14.

| Input | | | | Output |
|---|---|---|---|---|
| A | W | $S_1$ | $S_0$ | L |
| X | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |

**Table 14**

The output for the remaining conditions will be logic 0. The K map is drawn in Fig 16.

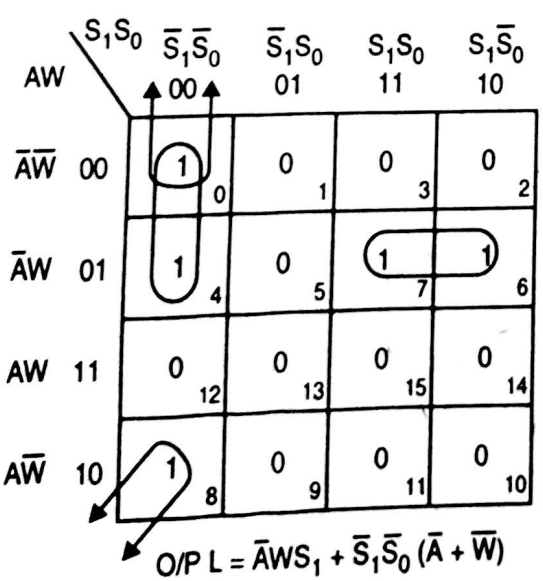



**Fig. 5.16 (a)**

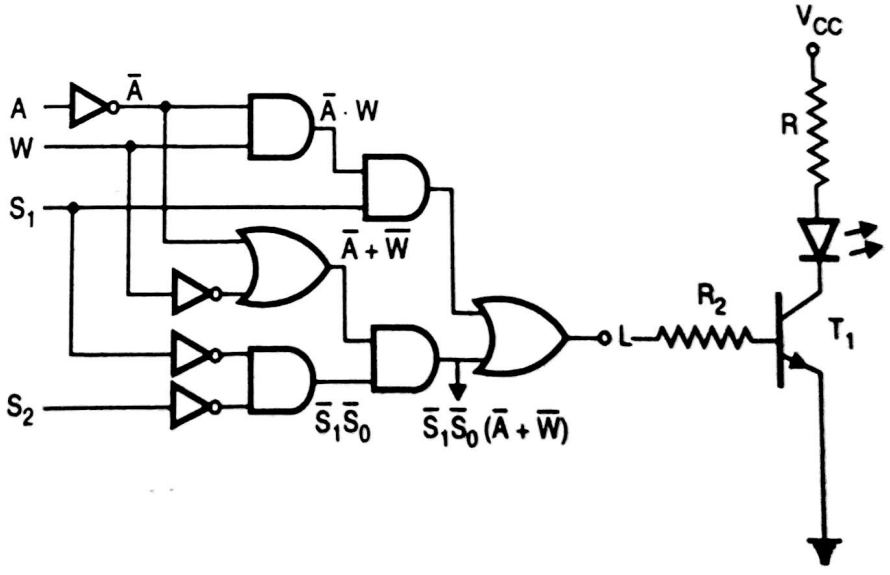



**Fig. 5.16 (b)**

Scanned by CamScanner

**Ex. 14 :** A majority function is a digital circuit whose output is logic 1 if and only if the majority of inputs is 1, the output is logic 0 otherwise.

(i) Write the truth table of a five input majority function.

(ii) Minimize the above function using K map.

**Soln. :** Given a give variable input.

Output Y = 1 if only if three or more bits are at logic 1 (majority).

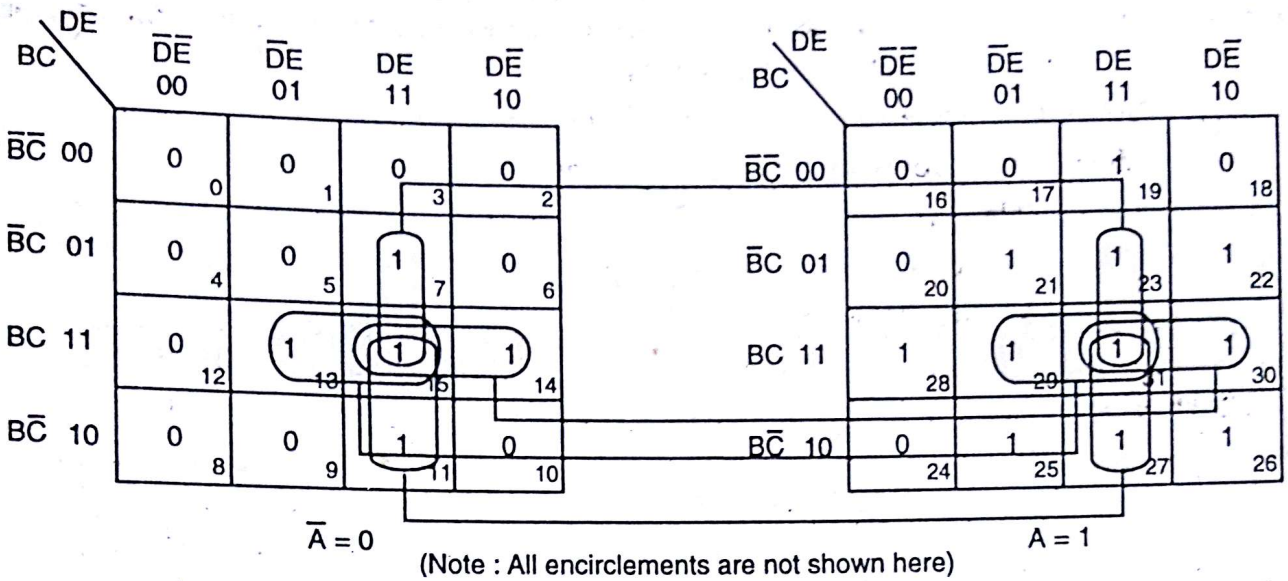
The truth table is illustrated in Table 15.

| | Inputs | | | | | Output |
|---|---|---|---|---|---|---|
| **Decimal** | **A** | **B** | **C** | **D** | **E** | **Y** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 1 | 1 | 1 |
| 8 | 0 | 1 | 0 | 0 | 0 | 0 |
| 9 | 0 | 1 | 0 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 | 0 | 0 |
| 11 | 0 | 1 | 0 | 1 | 1 | 1 |
| 12 | 0 | 1 | 1 | 0 | 0 | 0 |
| 13 | 0 | 1 | 1 | 0 | 1 | 1 |
| 14 | 0 | 1 | 1 | 1 | 0 | 1 |
| 15 | 0 | 1 | 1 | 1 | 1 | 1 |
| 16 | 1 | 0 | 0 | 0 | 0 | 0 |
| 17 | 1 | 0 | 0 | 0 | 1 | 0 |
| 18 | 1 | 0 | 0 | 1 | 0 | 0 |
| 19 | 1 | 0 | 0 | 1 | 1 | 1 |
| 20 | 1 | 0 | 1 | 0 | 0 | 0 |
| 21 | 1 | 0 | 1 | 0 | 1 | 1 |
| 22 | 1 | 0 | 1 | 1 | 0 | 1 |
| 23 | 1 | 0 | 1 | 1 | 1 | 1 |
| 24 | 1 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 1 | 0 | 0 | 1 | 1 |
| 26 | 1 | 1 | 0 | 1 | 0 | 1 |
| 27 | 1 | 1 | 0 | 1 | 1 | 1 |
| 28 | 1 | 1 | 1 | 0 | 0 | 1 |
| 29 | 1 | 1 | 1 | 0 | 1 | 1 |
| 30 | 1 | 1 | 1 | 1 | 0 | 1 |
| 31 | 1 | 1 | 1 | 1 | 1 | 1 |

**Table 15**

**The K map is illustrated in Fig. 5.17 (a).**

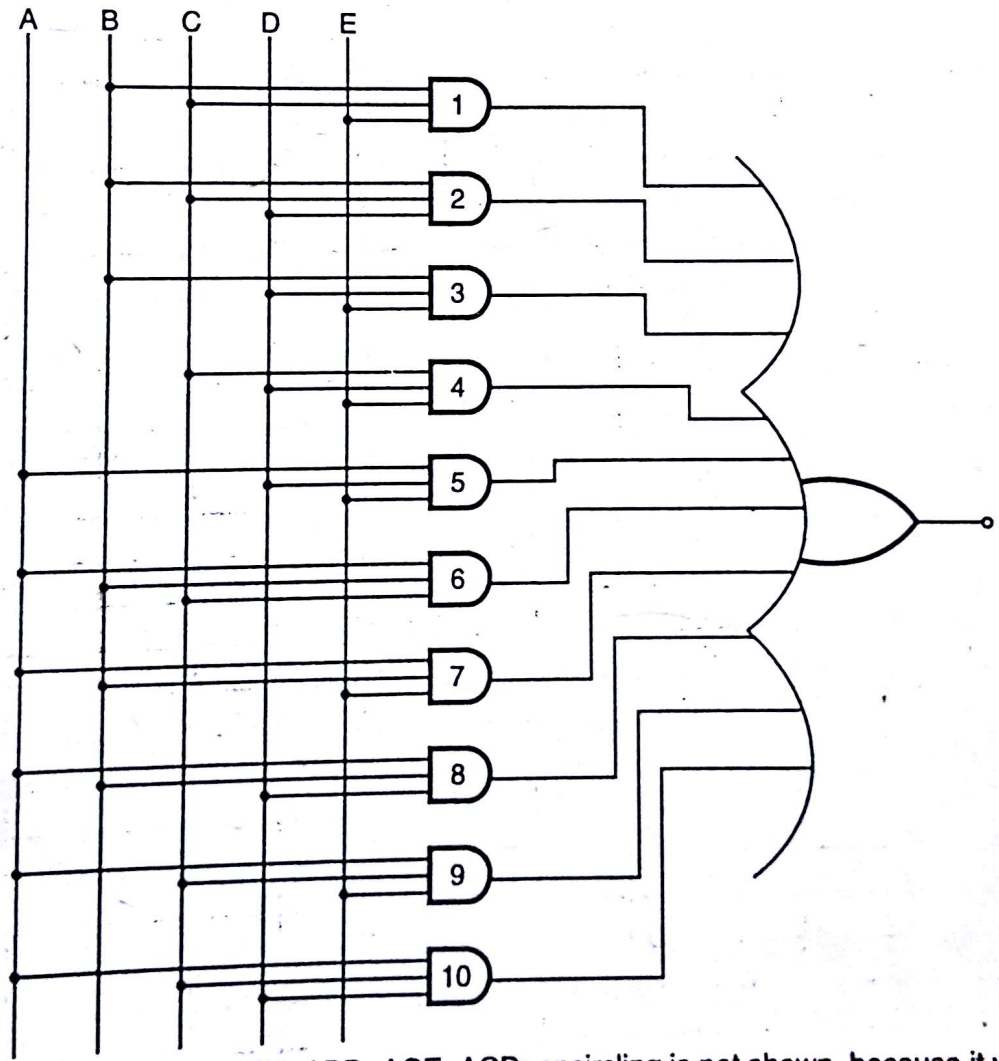
Scanned by CamScanner




(Note : All encirclements are not shown here)

**Fig. 5.17 (a)**

Output Y = BCE + BCD + BDE + CDE + ADE + ABC + ABE + ABD + ACE + ACD




**Note :** For terms ADE, ABC, ABE, ABD, ACE, ACD; encircling is not shown, because it will jumble up the K-map. So cooperate for the same.

**Fig. 5.17 (b)**

Scanned by CamScanner

## 5.2 Multi–input Multi-output Combinational Circuit :

Uptil now we have solved some examples. The common feature of all problems was, ***multi input and single output***. The circuit was accepting more than one inputs but providing single outputs.

It is also possible to design combinational logic circuit which accepts multiple input and provides multiple output. The generalised block schematic is should in Fig. 5.18.

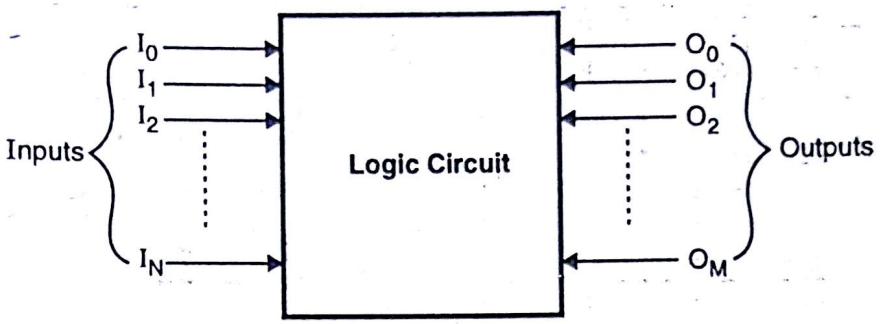



**Fig. 5.18**

As shown in Fig. 5.18, inputs to logic circuit are $I_0$ to $I_N$. Outputs are $O_0$ to $O_M$. It is not at all essential that N = M. It can be N > M, N = M or in some cases N < M.

***The procedure of designing the combinational circuit is going to remain same i.e. you have to write truth table (now truth table will contain more outputs), draw K-map and draw circuit. K–map for each output variable will be different. Circuits you can combine.***

Let's solve some example.

---

**Ex. 15 :** (a) The input to a combinational logic circuit is a 4–bit binary number. Design the logic circuit with minimum hardware for the following:

(i) Output Y1 = 1 if the input binary number is 5 or less than 5.

(ii) Output Y2 = 0 if the input binary number is 9 or more than 9.

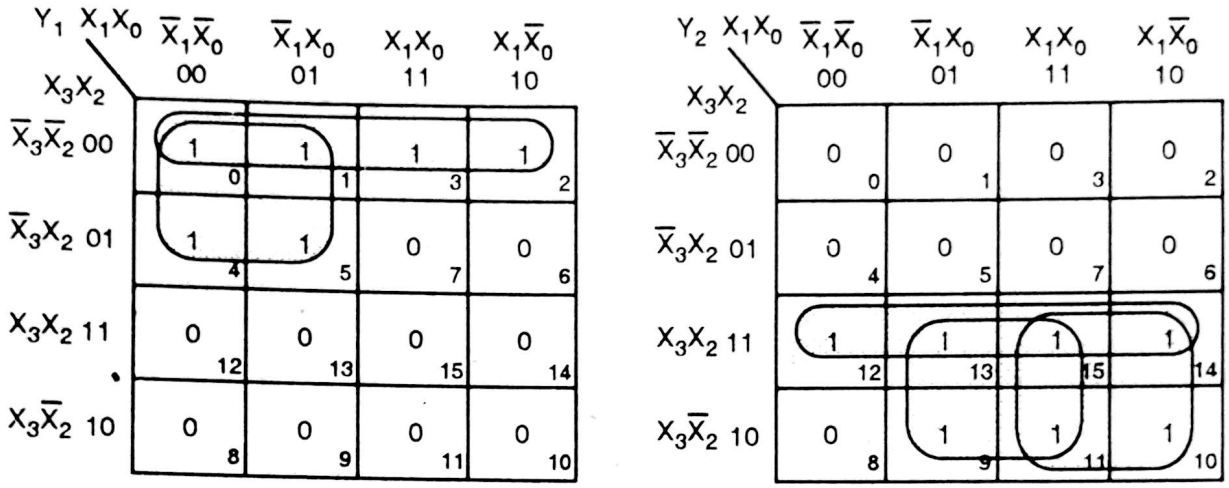
**Ans. :** Truth table for combinational logic circuit is,

| Decimal | INPUTS | | | | OUTPUTS | |
|---|---|---|---|---|---|---|
| | $X_3$ | $X_2$ | $X_1$ | $X_0$ | $Y_1$ | $Y_2$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | 0 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 |
| 13 | 1 | 1 | 0 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 0 | 1 |

**Table 16**

Scanned by CamScanner

**K Map :** Refer Fig. 5.19.




$$Y_1 = \overline{X}_3\overline{X}_2 + \overline{X}_3\,\overline{X}_1$$
$$= \overline{X}_3\,(\overline{X}_2+\overline{X}_1)$$
$$= \overline{X_3}\cdot\overline{(X_2X_1)}$$
$$= \overline{X_3 + X_2X_1}$$

$$Y_2 = X_3\cdot X_2 + X_3\,X_0 + X_3X_1$$
$$= X_3\cdot(X_0+X_1+X_2)$$

**Fig. 5.19 (a)**

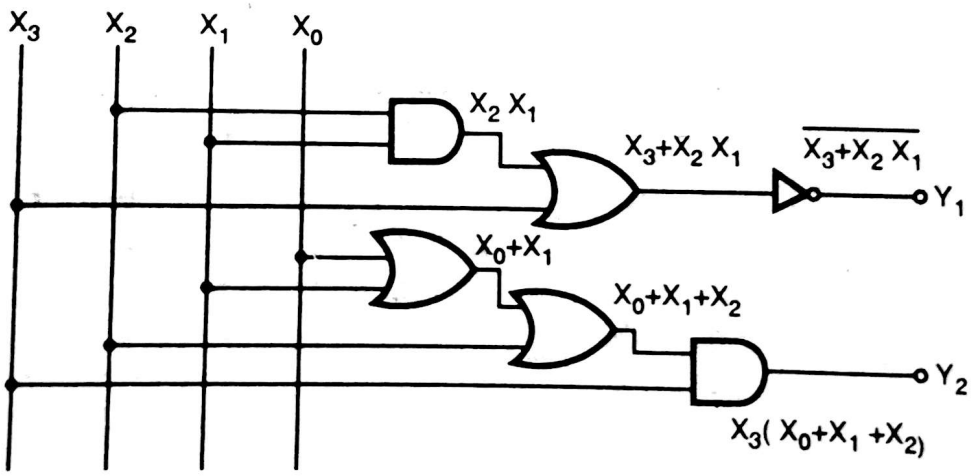



**Fig. 5.19 (b)**

**Ex. 16 :** Design a combinational logic circuit for the following description. The circuit has four inputs and two outputs. One of the outputs is to be true when the majority of inputs is true. The other output is to be true only when there is a tie between the inputs. Implement the above logic using NAND gates only. **(June 99, 10 Marks)**

**Soln. : The truth table is shown in Table 17.**

Scanned by CamScanner

| | Inputs | | | | Outputs | |
|---|---|---|---|---|---|---|
| Decimal | A | B | C | D | $Y_1$ | $Y_2$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 0 |
| 10 | 1 | 0 | 1 | 0 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 | 0 |
| 12 | 1 | 1 | 0 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 | 1 | 1 |

**Table 17**

The K maps are as shown in Fig. 5.20 (a).

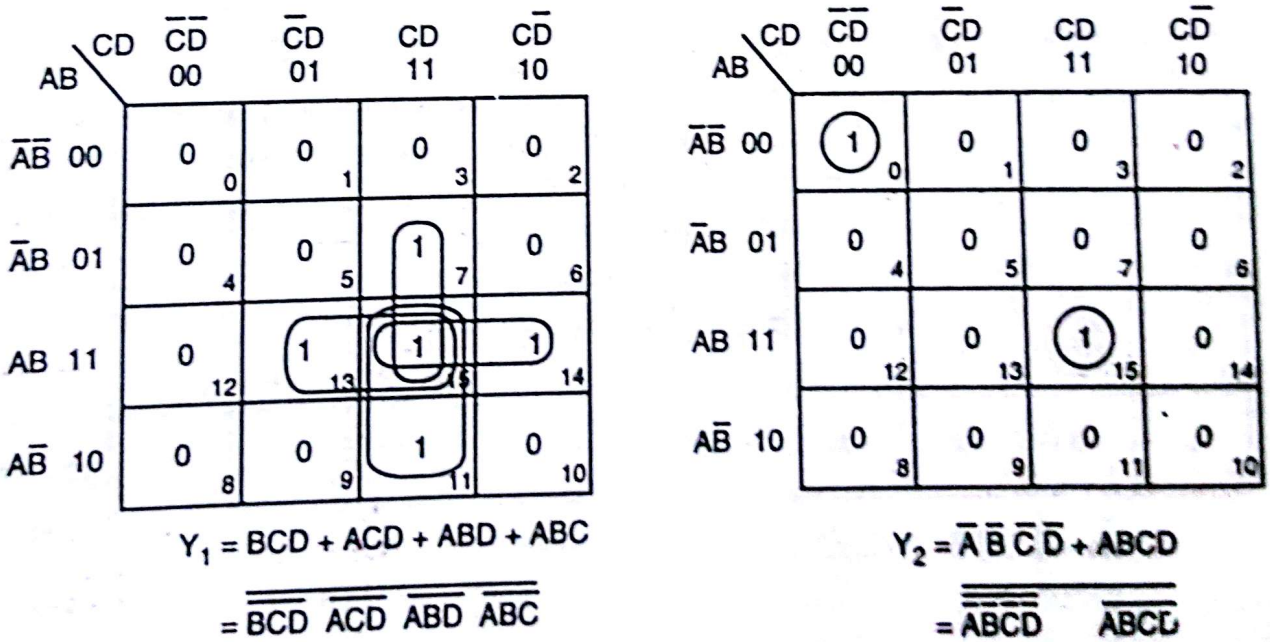



**Fig. 5.20 (a)**

In NAND form applying DeMorgans Law, the circuit diagram is illustrated in Fig. 5.20 (b).

Scanned by CamScanner

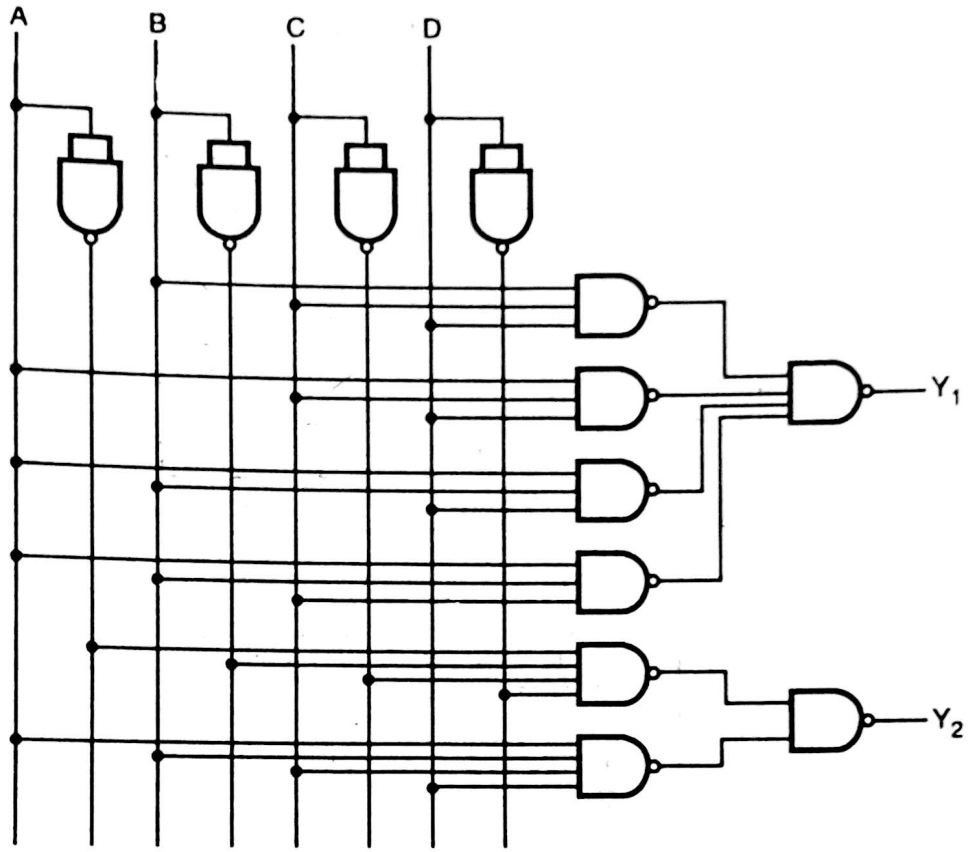



**Fig. 5.20 (b)**

**Ex. 17 :** A chemical reactor has three sensors indicating the following conditions :

(i) Pressure 'P' is low or high

(ii) Temperature 'T' is low or high

(iii) Liquid level 'L' is low or high

H has two controls : Heater (H) which is either ON or OFF and an inlet valve (V) which is either OPEN or CLOSED. The controls are operated as shown in Table 18.

(a) Using the convention HIGH = 1, LOW = 0, ON = 1, OFF = 0, OPEN = 1 and CLOSED = 0, draw the Karnaugh maps for H and V.

(b) Obtain the minimal SOP and POS expressions for H and V.

(c) Replace the logic H and V by using logic gates.

| INPUTS | | | OUTPUTS | |
|---|---|---|---|---|
| P | T | L | H | V |
| LOW | LOW | LOW | OFF | OPEN |
| LOW | LOW | HIGH | ON | CLOSED |
| LOW | HIGH | LOW | OFF | OPEN |
| LOW | HIGH | HIGH | OFF | CLOSED |
| HIGH | LOW | LOW | ON | OPEN |
| HIGH | LOW | HIGH | ON | CLOSED |
| HIGH | HIGH | LOW | OFF | CLOSED |
| HIGH | HIGH | HIGH | × | × |

× – system shut down

**Table 18**

Scanned by CamScanner

**Soln. :** The truth table is shown in Table 19. The K maps are illustrated in Fig. 5.21 (a)

| Decimal | P | T | L | H | V |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 1 | 0 | 0 |
| 4 | 1 | 0 | 0 | 1 | 1 |
| 5 | 1 | 0 | 1 | 1 | 0 |
| 6 | 1 | 1 | 0 | 0 | 0 |
| 7 | 1 | 1 | 1 | 0 | 0 |

**Table 19**

**K–map :**

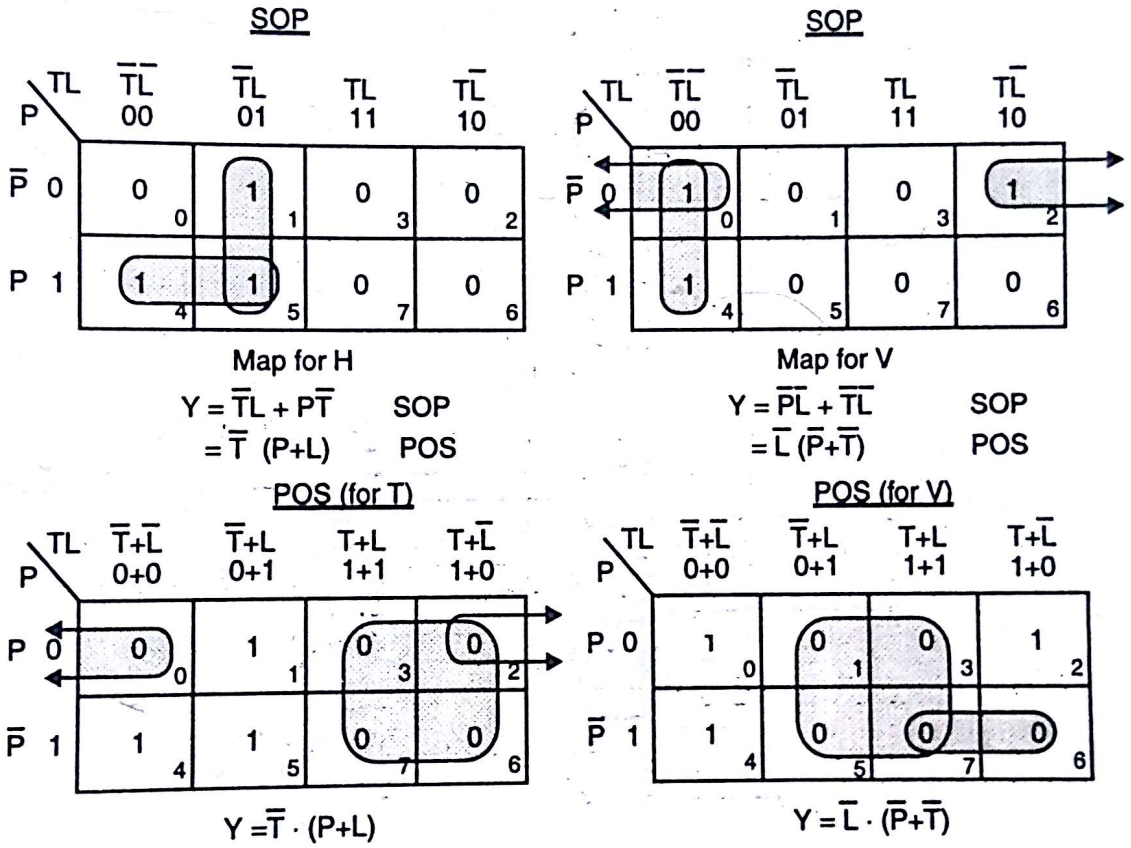
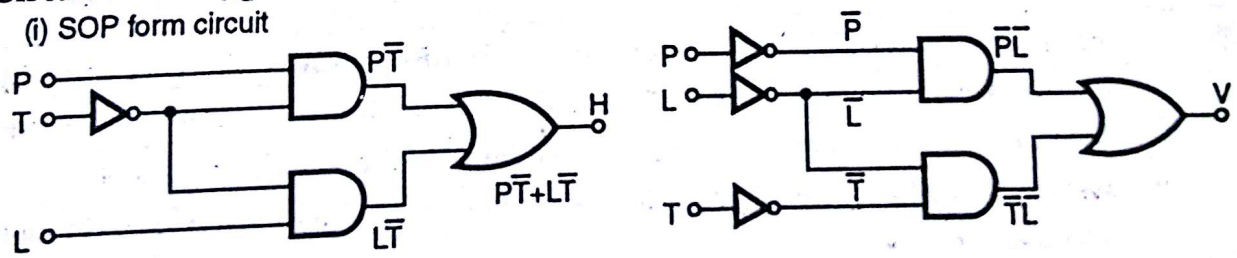



**Fig. 5.21 (a)**

**Circuit :** Refer Fig. 5.21 (b)

(i) SOP form circuit




(ii) POS form circuit

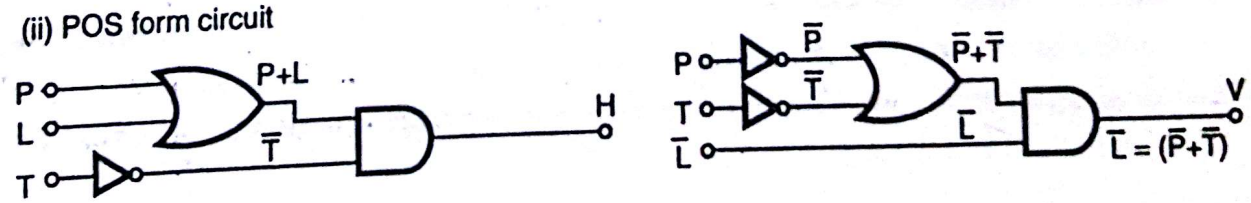



**Fig. 5.21 (b)**

Scanned by CamScanner

**Ex. 18 :** $A_3\ A_2\ A_1\ A_0$ and $B_3\ B_2\ B_1\ B_0$ represent two, four bit words. Design a circuit with control line M, such that when M = 1, the word given by $A_3\ A_2\ A_1 A_0$ is output and when M = 0, output gives word $B_3\ B_2 B_1\ B_0$. use gates.

**Soln. :** This problem involves either passing $A_3\ A_2\ A_1\ A_0$ or $B_3\ B_2\ B_1\ B_0$. This can be implemented using-AND gates. We have already discussed its use as a control gate.

Since output is either $A_3A_2\ A_1A_0$ or $B_3\ B_2\ B_1B_0$ we logically OR the corresponding A and B bits as shown in Fig. 5.22.

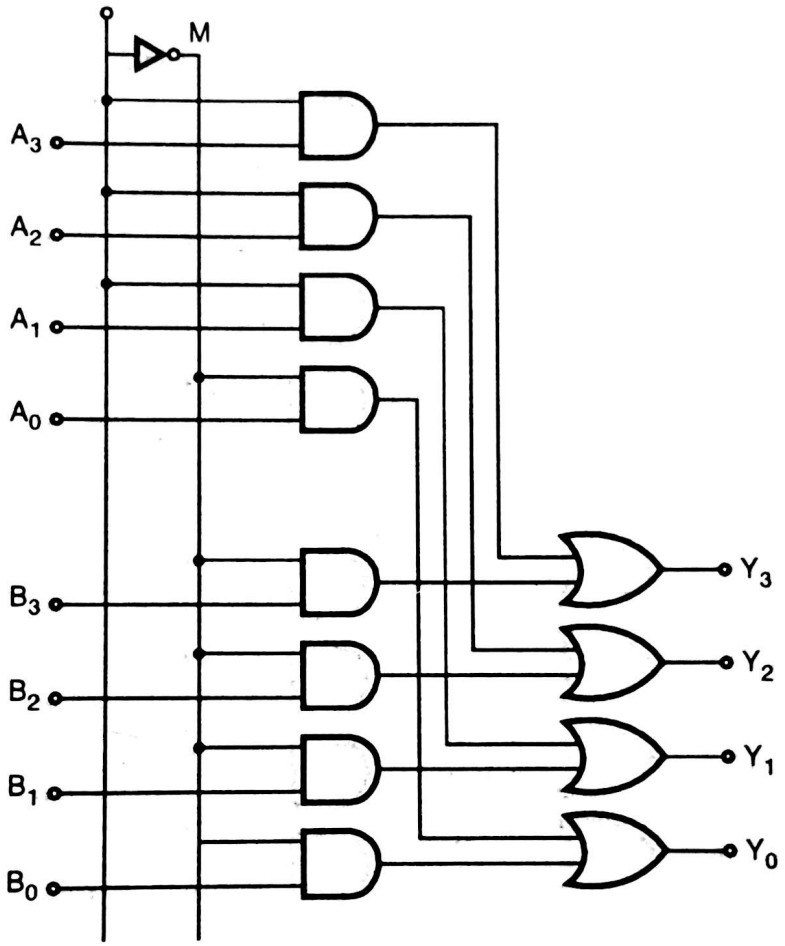



**Fig. 5.22**

**Ex. 19 :** A step in space vehicle checkout depends on FOUR sensors $S_1$, $S_2$, $S_3$ and $S_4$. Every circuit is working properly if sensor $S_1$ and atleast two of the other three sensors are at logic 1. (Assuming a sensor produces logic 1 when the circuit is functioning properly). Implement the system using NAND gates only, after finding the minimal SOP expression for output if the circuit works properly.

Find the minimal SOP expression for output if the circuit is not working properly. The output is connected to a red LED which must glow if the circuit is not working properly.

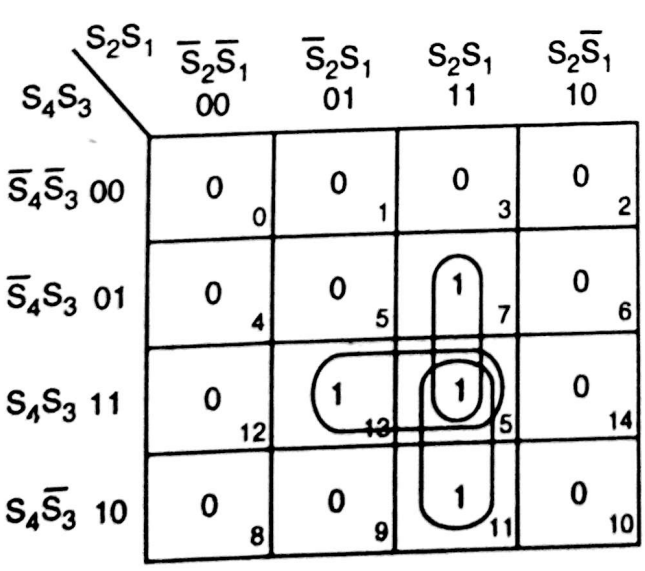
**Soln. :** Given four sensors i.e., we have four variables. The truth table is illustrated in Table 10.

Scanned by CamScanner

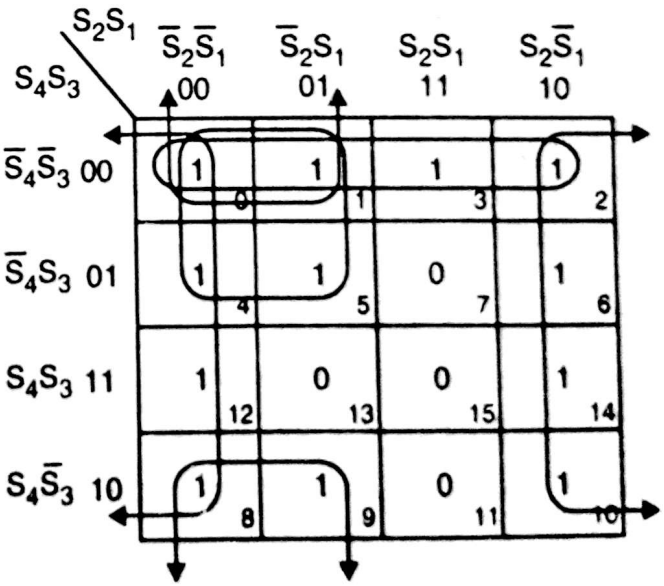
| Decimal | $S_4$ | $S_3$ | $S_2$ | $S_1$ | $Y_1$ | $Y_2$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | 1 | 0 |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 |
| 13 | 1 | 1 | 0 | 1 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 1 | 0 |
| | | | | | (Working Properly) | (Not working Properly) |

**Table 20**

The K–maps are shown in Fig. 5.23 (a)




$$Y_1 = S_2S_1 (S_4 + S_3) + S_4S_3S_1$$
$$= S_1 (S_2S_4 + S_2S_3 + S_3S_4)$$




$$Y_2 = \bar{S}_1 + \bar{S}_4\bar{S}_3 + \bar{S}_4\bar{S}_2 + \bar{S}_3\bar{S}_2$$

**Fig. 5.23 (a)**

Scanned by CamScanner

**Circuit :** Refer Fig. 5.23 (b)

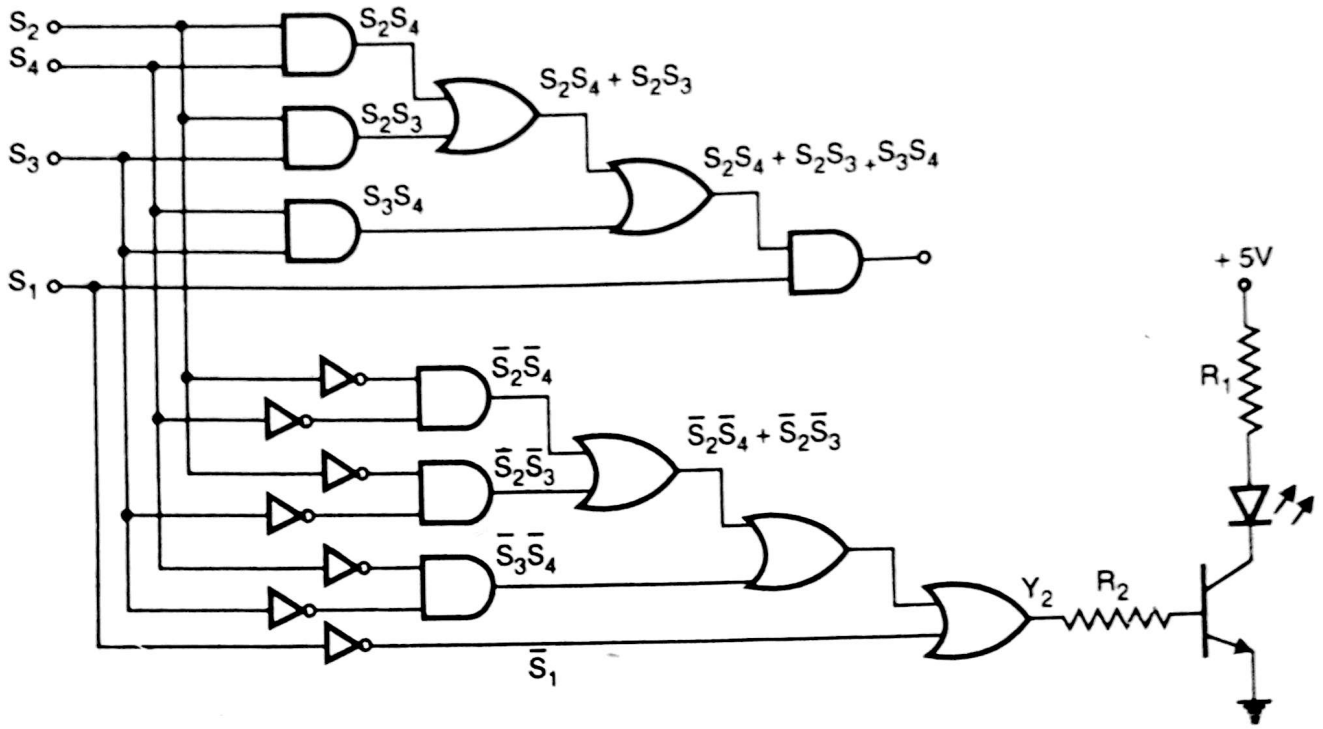



**Fig. 5.23 (b)**

$Y_2$ could also have been obtained by connecting an inverter between $Y_1$ and $Y_2$.

L. E. D. design

Red LED : $V_D = 1.2$ V; $I_D = 20$ mA

Applying KVL at input and output,

$5 - 20 \text{ mA } R_1 - 1.2 - 0.2 = 0$

$\therefore R_1 = 180\ \Omega$

$2.4 - \frac{20\text{mA}}{100} R_2 - 0.7 = 0$ (Assuming $\beta = 100$ for switching transistor)

$\therefore R_2 = 8.5\ \text{k}\Omega$

---

**Ex. 20 :** The input to a combinational logic circuit is a 4 bit binary number. Design the circuit with minimum hard ware for the following :

(i) Output $0_1 = 1$ if the number is prime.

(ii) Output $0_2 = 1$ if the number is divisible by 3.

**Soln. :** Table 21 gives the truth table.

Scanned by CamScanner

| Decimal | Input | | | | Output | |
|---|---|---|---|---|---|---|
| | $B_3$ | $B_2$ | $B_1$ | $B_0$ | $0_1$ | $0_2$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 | 0 |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 |
| 13 | 1 | 1 | 0 | 1 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 0 | 1 |

**Table 21**

The K maps are as shown in Fig. 5.24 (a).

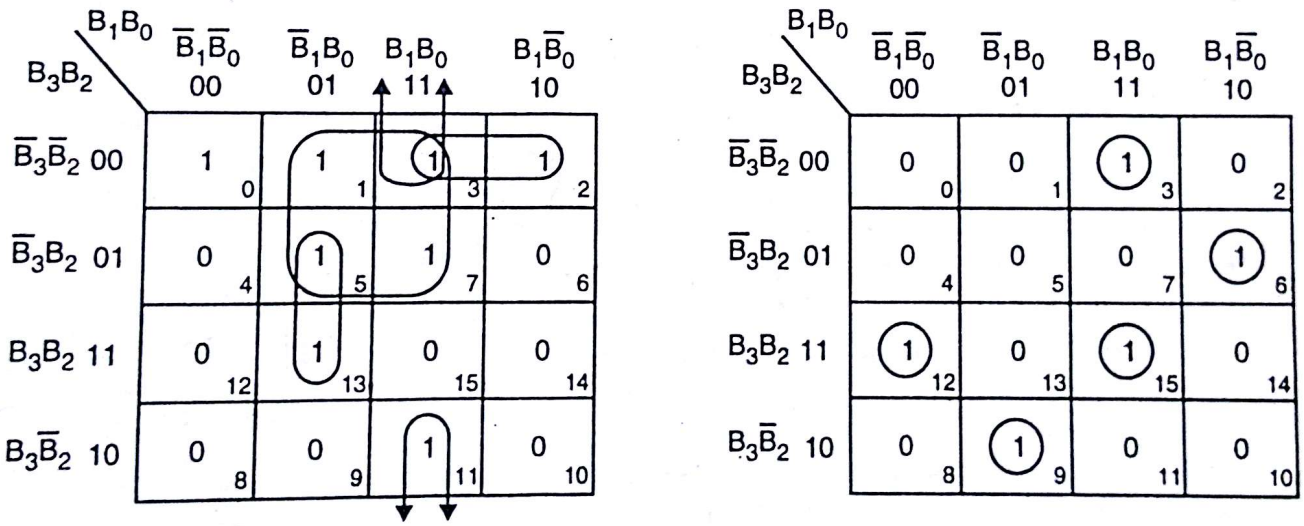



**Fig. 5.24 (a)**

$$O_1 = \bar{B}_3 B_0 + B_2 \bar{B}_1 B_0 + \bar{B}_2 B_1 B_0 + \bar{B}_3 \bar{B}_2 B_1$$

$$= \bar{B}_3 (B_0 + \bar{B}_2 B_1) + B_0 (B_1 \bar{B}_2 + \bar{B}_1 B_2)$$

$$= \bar{B}_3 (B_0 + \bar{B}_2 B_1) + B_0 (B_1 \oplus B_2)$$

$$O_2 = B_3 B_2 \bar{B}_1 \bar{B}_0 + B_3 \bar{B}_2 \bar{B}_1 B_0 + B_1 B_0 (B_3 B_2 + \bar{B}_3 \bar{B}_2) + \bar{B}_3 B_2 B_1 \bar{B}_0$$

$$= B_3 \bar{B}_1 (B_2 \bar{B}_0 + \bar{B}_2 B_0) + B_1 B_0 (B_3 B_2 + \bar{B}_3 + \bar{B}_2) + \bar{B}_3 B_2 B_1 \bar{B}_0$$

$$= B_3 \bar{B}_1 (B_2 \oplus B_0) + B_1 B_0 (B_3 \odot B_2) + \bar{B}_3 B_2 B_1 \bar{B}_0$$

Scanned by CamScanner

The circuit diagram is as shown in Fig. 5.24 (b)

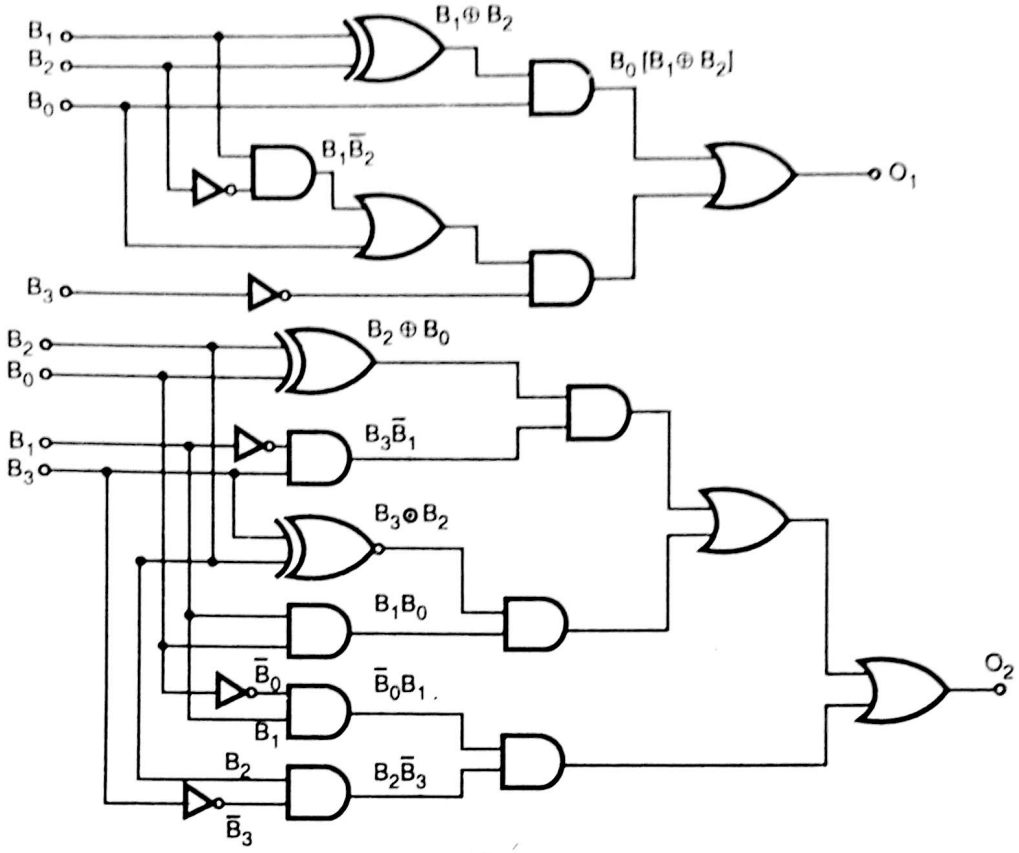



Fig. 5.24 (b)

**Ex. 21 :** Fig. 5.25 shows the intersection of main highway with a second acess road. Vehicle detection sensors are placed along lane C and D (Main Road) and lanes A and B (access road). These sensor outputs are ***LOW*** when no hehicle is present and ***HIGH*** when vehicle is present. The taffic lights should be controlled according to the following logic :

(i) The EW traffic light will be green whenever both C and D lanes are occupied.

(ii) The EW light will be green whenever C or D is occupied but lanes A and B are not both occupied.

(iii) The NS light will be green whenever both lanes A and B are occupied but C and D are not occupied

(iv) The NS light will also be green when either A or B is occupied while C and D are both vacant.

(v) The EW light will be green when no vehicle is present.

Using A, B, C and D sensor outputs as inputs, design a logic circuit to control traffic lights. There are two outputs, N-S and E-W which goes high when corresponding light is green.

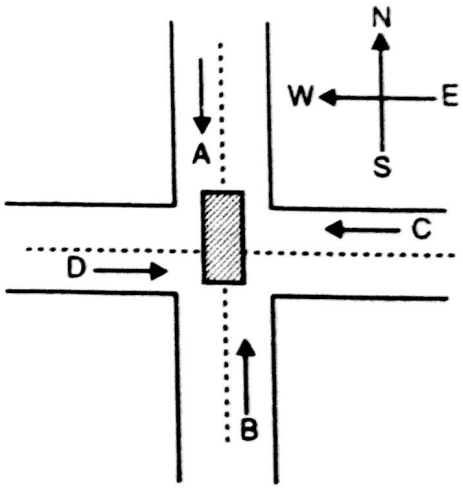



Fig. 5.25

**Soln. :** (1) First read all statements carefully, i.e. (i), (ii), (iii), (iv) and (v).

(2) Remember one thing when four roads are meeting at a junction, E–W lighting will be complement of N–S light or V.V. This is normal trend.

(3) As we know, variables are 4 i.e. A, B, C, D. Therefore there will be total 16 states.

(4) We will decide that there will be two outputs namely E–W and N–S.

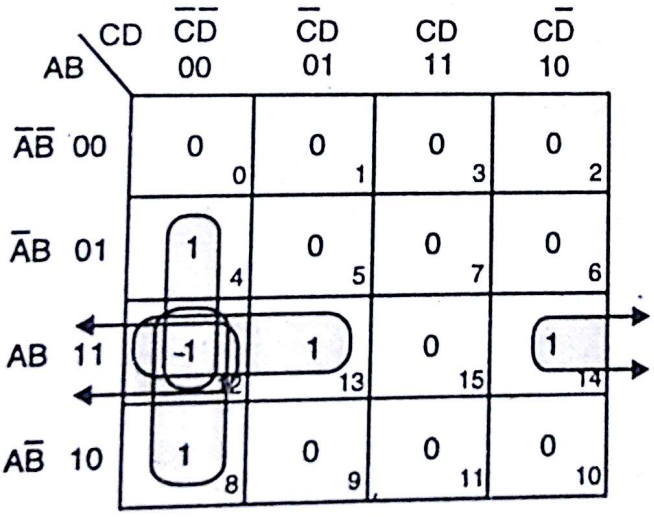
Scanned by CamScanner

(5) First write down truth table for the same.
Remember,
LOW → Logic '0' → No vehicle
HIGH → Logic '1' → Vehicle present.

| Inputs | | | | Outputs | | |
|---|---|---|---|---|---|---|
| A | B | C | D | N – S | E – W | |
| 0 | 0 | 0 | 0 | 0 | 1 | → condition (v) |
| 0 | 0 | 0 | 1 | 0 | 1 | |
| 0 | 0 | 1 | 0 | 0 | 1 | → condition (ii) |
| 0 | 0 | 1 | 1 | 0 | 1 | → condition (i) |
| 0 | 1 | 0 | 0 | 1 | 0 | → condition (iv) |
| 0 | 1 | 0 | 1 | 0 | 1 | → condition (ii) |
| 0 | 1 | 1 | 0 | 0 | 1 | |
| 0 | 1 | 1 | 1 | 0 | 1 | → condition (i) |
| 1 | 0 | 0 | 0 | 1 | 0 | → condition (iv) |
| 1 | 0 | 0 | 1 | 0 | 1 | → condition (ii) |
| 1 | 0 | 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 1 | 0 | 1 | → condition (i) |
| 1 | 1 | 0 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 1 | 1 | 0 | → condition (ii), (iii) |
| 1 | 1 | 1 | 0 | 1 | 0 | |
| 1 | 1 | 1 | 1 | 0 | 1 | → condition (i) |

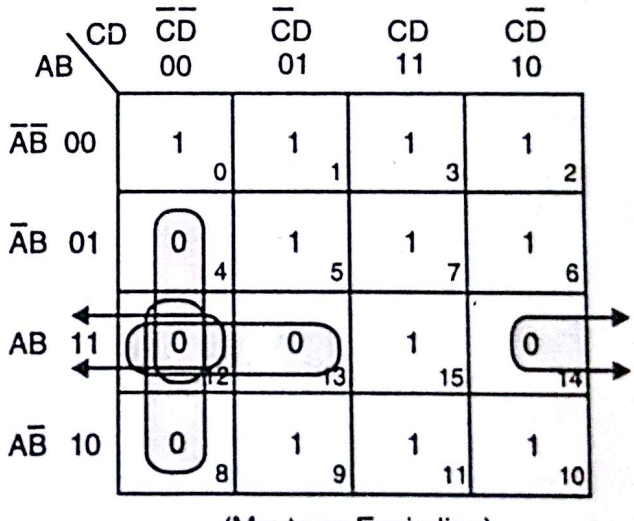
**Table 22**

**K–Map :**




(Minterm Encircling)

$N\text{-}S = AB(\overline{C}+\overline{D}) + \overline{C}\overline{D}(A+B)$




(Maxterm Encircling)

$$E\text{-}W = (\overline{A}+\overline{B}+C)(\overline{A}+\overline{B}+D)(C+D+\overline{A})(C+D+\overline{B})$$

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

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

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

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

$$= \overline{N.S}$$

**Fig. 5.26 (a)**

Scanned by CamScanner

**Circuit :**

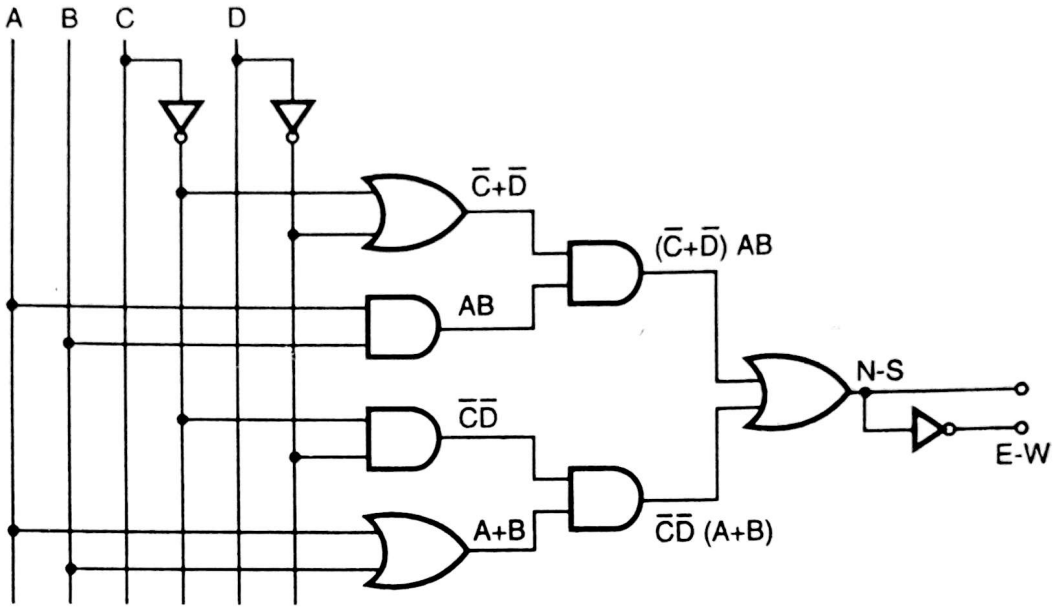



**Fig. 5.26 (b)**

**Ex. 22 :** Design a combinational logic circuit that will accomplish multiplication of a two bit number $X_1X_0$ by a two bit binary number $Y_1Y_0$. The circuit will have four outputs representing the product $p_3p_2p_1p_0$.

**Soln. :** Truth table is illustrated in Table 23. K maps are drawn in Fig. 5.27.

| Input | | | | | | Output | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Decimal** | $X_1$ | $X_0$ | $Y_1$ | $Y_0$ | **Decimal** | $p_3$ | $p_2$ | $p_1$ | $p_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 3 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 2 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 3 | 0 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 2 | 1 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 3 | 0 | 1 | 1 | 0 |
| 3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 3 | 1 | 1 | 1 | 0 | 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 3 | 1 | 0 | 0 | 1 |

**Table 23**

Scanned by CamScanner

**K–Map :**

| $Y_1Y_0$ \ $X_1X_0$ | $\overline{X}_1\overline{X}_0$ 00 | $\overline{X}_1X_0$ 01 | $X_1X_0$ 11 | $X_1\overline{X}_0$ 10 |
|---|---|---|---|---|
| $\overline{Y}_1\overline{Y}_0$ 00 | 0 (0) | 0 (4) | 0 (12) | 0 (8) |
| $\overline{Y}_1Y_0$ 01 | 0 (1) | 0 (5) | 0 (13) | 0 (9) |
| $Y_1Y_0$ 11 | 0 (3) | 0 (7) | 1 (15) | 0 (11) |
| $Y_1\overline{Y}_0$ 10 | 0 (2) | 0 (6) | 0 (14) | 0 (10) |

$P_3 = X_1X_0Y_1Y_0$

| $Y_1Y_0$ \ $X_1X_0$ | $\overline{X}_1\overline{X}_0$ 00 | $\overline{X}_1X_0$ 01 | $X_1X_0$ 11 | $X_1\overline{X}_0$ 10 |
|---|---|---|---|---|
| $\overline{Y}_1\overline{Y}_0$ 00 | 0 (0) | 0 (4) | 0 (12) | 0 (8) |
| $\overline{Y}_1Y_0$ 01 | 0 (1) | 0 (5) | 0 (13) | 0 (9) |
| $Y_1Y_0$ 11 | 0 (3) | 0 (7) | 0 (15) | 1 (11) |
| $Y_1\overline{Y}_0$ 10 | 0 (2) | 0 (6) | 1 (14) | 1 (10) |

$P_3 = X_1\overline{X}_0Y_1 + X_1Y_1\overline{Y}_0$

$= X_1Y_1(\overline{X}_0 + \overline{Y}_0)$

| $Y_1Y_0$ \ $X_1X_0$ | $\overline{X}_1\overline{X}_0$ 00 | $\overline{X}_1X_0$ 01 | $X_1X_0$ 11 | $X_1\overline{X}_0$ 10 |
|---|---|---|---|---|
| $\overline{Y}_1\overline{Y}_0$ 00 | 0 (0) | 0 (4) | 0 (12) | 0 (8) |
| $\overline{Y}_1Y_0$ 01 | 0 (1) | 0 (5) | 1 (13) | 1 (9) |
| $Y_1Y_0$ 11 | 0 (3) | 1 (7) | 0 (15) | 1 (11) |
| $Y_1\overline{Y}_0$ 10 | 0 (2) | 1 (6) | 1 (14) | 0 (10) |

$P_1 = X_1\overline{Y}_1Y_0 + X_1\overline{X}_0Y_0 + \overline{X}_1X_0Y_1 + X_0Y_1\overline{Y}_0$

$= X_1Y_0(\overline{Y}_1 + \overline{X}_0) + X_0Y_1(\overline{X}_1 + \overline{Y}_0)$

$= X_1Y_0\overline{Y_1X_0} + X_0Y_1\overline{X_1Y_0} = X_1Y_0 \oplus X_0Y_1$

| $Y_1Y_0$ \ $X_1X_0$ | $\overline{X}_1\overline{X}_0$ 00 | $\overline{X}_1X_0$ 01 | $X_1X_0$ 11 | $X_1\overline{X}_0$ 10 |
|---|---|---|---|---|
| $\overline{Y}_1\overline{Y}_0$ 00 | 0 (0) | 0 (4) | 0 (12) | 0 (8) |
| $\overline{Y}_1Y_0$ 01 | 0 (1) | 1 (5) | 1 (13) | 0 (9) |
| $Y_1Y_0$ 11 | 0 (3) | 1 (7) | 1 (15) | 0 (11) |
| $Y_1\overline{Y}_0$ 10 | 0 (2) | 0 (6) | 0 (14) | 0 (10) |

$P_0 = X_0Y_0$

**Fig. 5.27 (a)**

Scanned by CamScanner

**Circuit :**

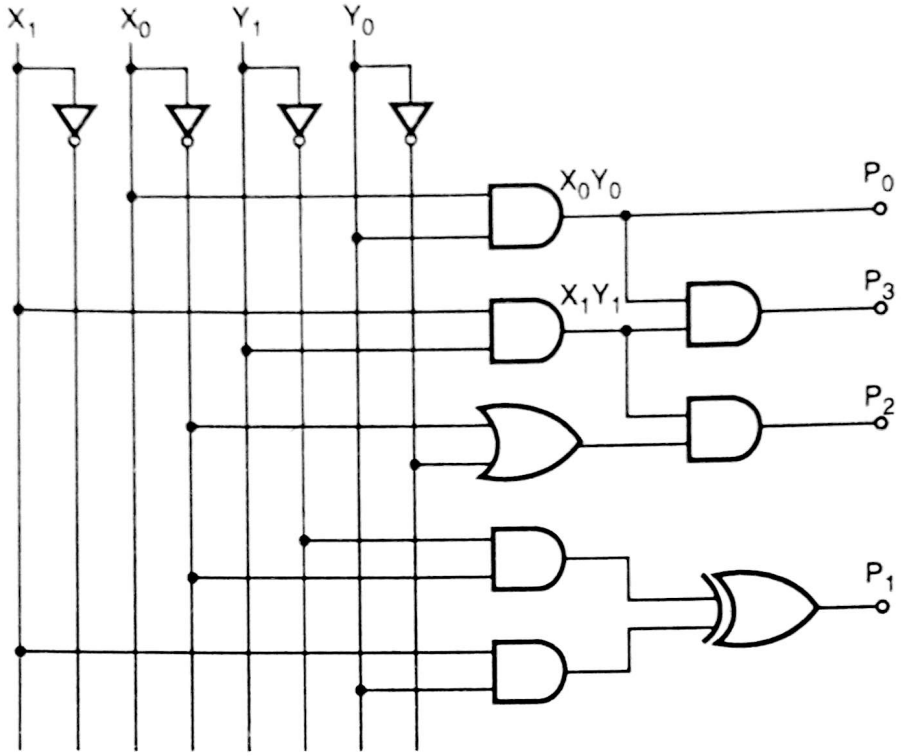



**Fig. 5.27 (b)**

## 5.3 Code Conversion :

Under this section we will convert one type of code to another; for example binary to gray, gray to binary, BCD to XS–3, XS–3 to BCD, 5421 code to 5421 gray code and vice versa etc. To convert from one code to other we are going to use K–map only. Now in this case ***Normally*** we will deal with multiple inputs and multiple outputs.

We are using 4 bit input and answer is normally 4 bit. So you have to draw K–map for number of output lines (bits). Let's solve some examples.

---

**Ex. 23 :** Convert 4 bit binary number to 4 bit gray.

**Soln. :** (1) Firstly we have to prepare a table where we have binary (4 bits, $B_3$ $B_2$ $B_1$ $B_0$) inputs and Gray (4 bits, $G_3$ $G_2$ $G_1$ $G_0$) outputs (Refer chapter 1 for the table).

(2) Truth table is,

| Decimal | $B_3$ | $B_2$ | $B_1$ | $B_0$ | $G_3$ | $G_2$ | $G_1$ | $G_0$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 11 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 12 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

**Table 24**

Scanned by CamScanner

(3) Now the next step is very simple. You draw K–Maps for $G_3$, $G_2$, $G_1$ and $G_0$ SEPARATELY

**(4) K–Maps :**

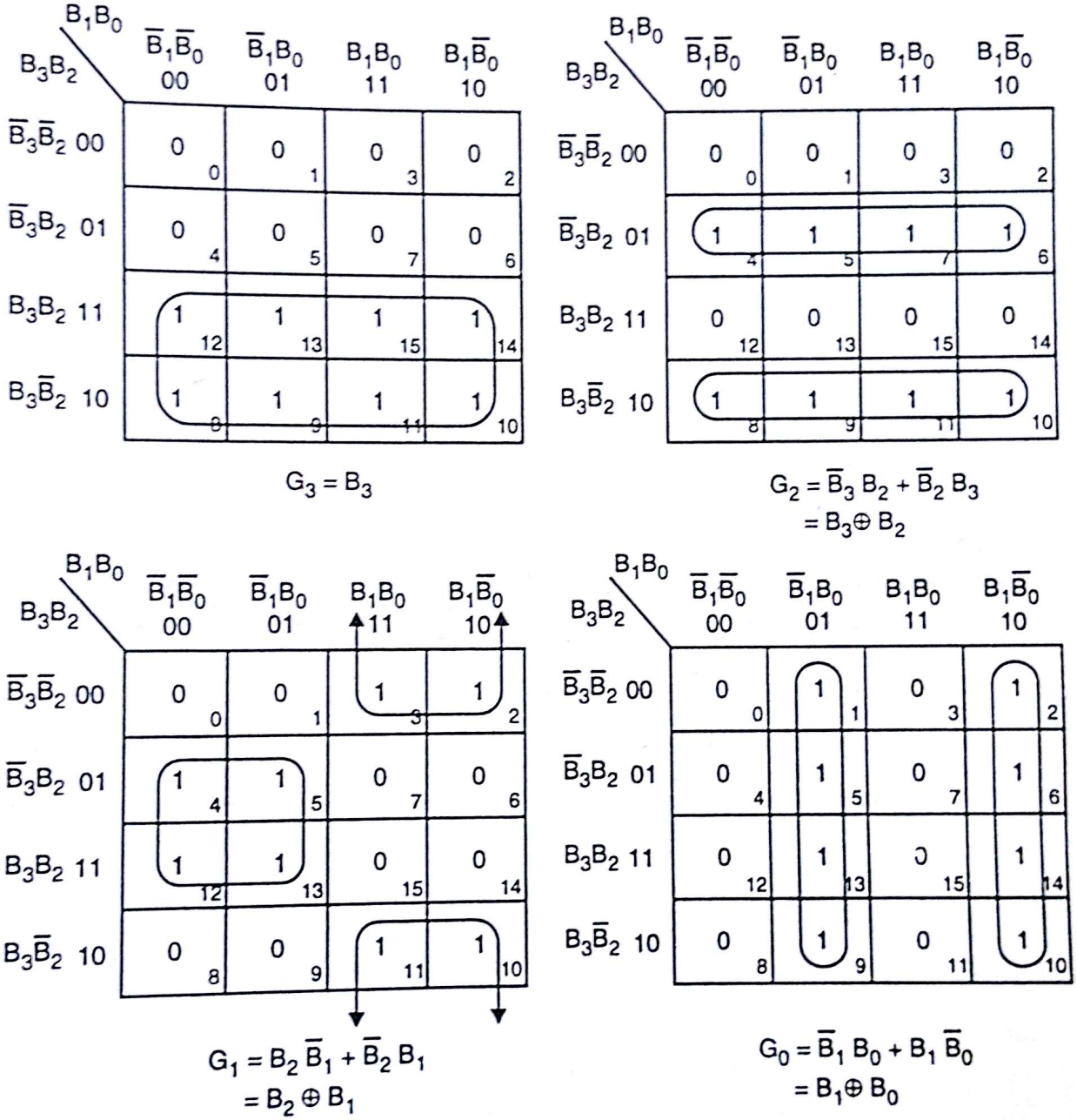



**Fig. 5.28 (a)**

Scanned by CamScanner

**(5) Circuit :**

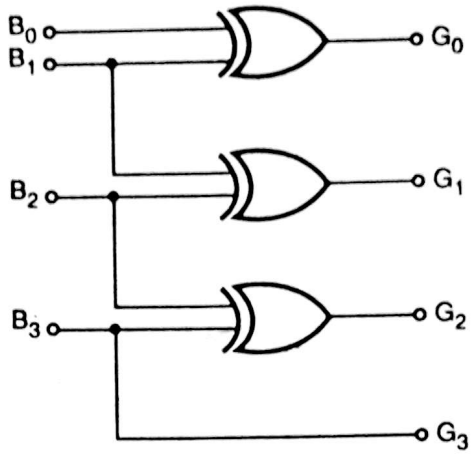



**Fig. 5.28 (b)**

**Ex. 24 :** Convert given 4 bit gray codes to binary (4 bit).

**Soln. :** Now in this case we have to do reverse i.e. convert gray codes to binary. Procedure is same as that we followed in Ex. 23.

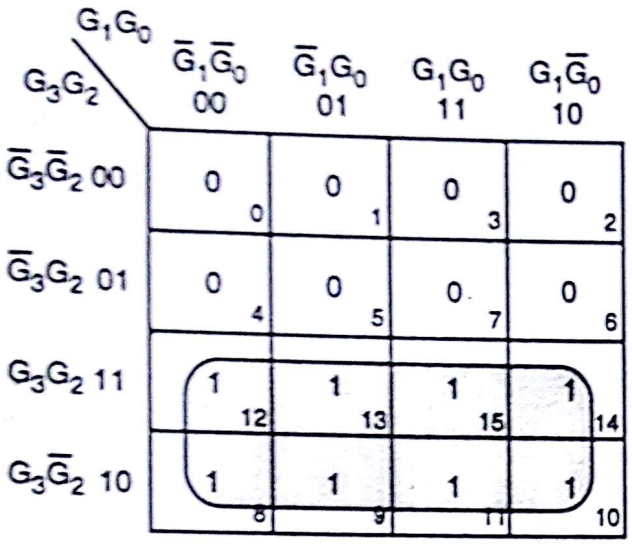
(1) Truth table.

| Decimal | $G_3$ | $G_2$ | $G_1$ | $G_0$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

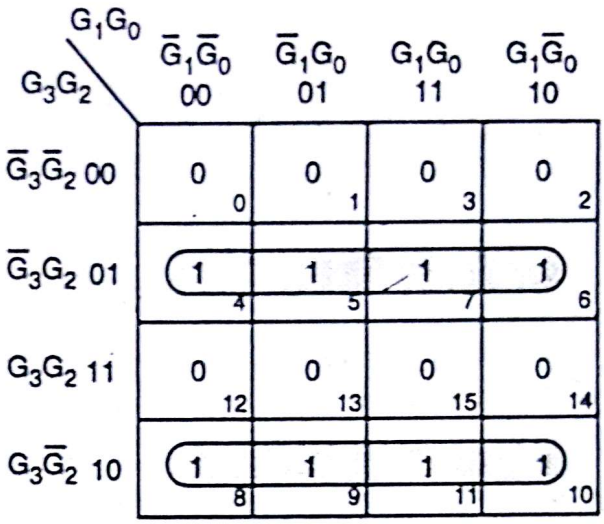
**Table 25**

Scanned by CamScanner

(2) K–map :

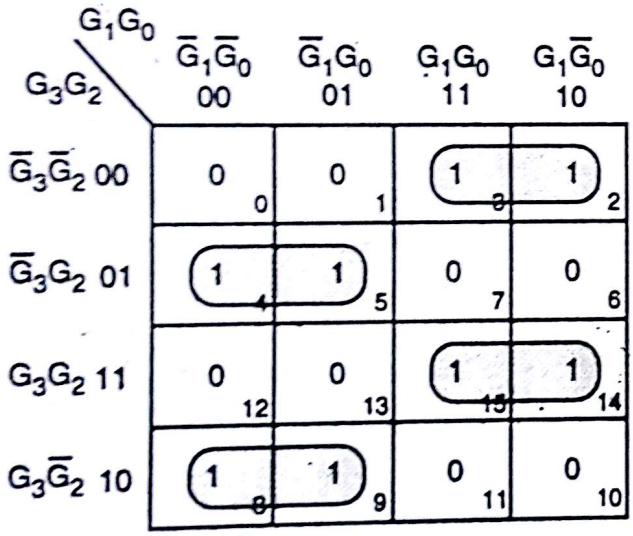



$B_3 = G_3$




$B_3 = G_3 \bar{G}_2 + \bar{G}_3 G_2$

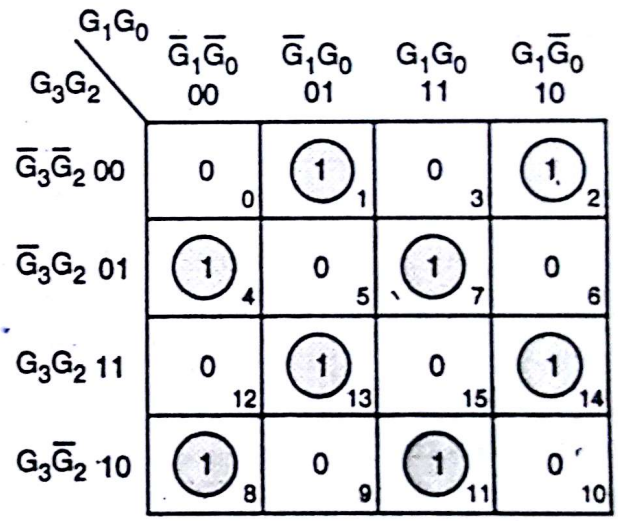
$= G_2 \oplus G_3$




$B_1 = \bar{G}_1 (\bar{G}_3 G_2 + G_3 \bar{G}_2) + G_1 (\bar{G}_3 \bar{G}_2 + G_3 G_2)$

$= \bar{G}_1 (G_2 \oplus G_3) + G_1 \overline{(G_2 \oplus G_3)}$

$= G_1 \oplus G_2 \oplus G_3$




$B_0 = \bar{G}_1 \bar{G}_0 (G_3 \bar{G}_2 + \bar{G}_3 G_2) + \bar{G}_1 G_0 (\bar{G}_3 \bar{G}_2 + G_3 G_2) + G_1 G_0 (G_3 \bar{G}_2 + \bar{G}_3 G_2) + G_1 \bar{G}_0 (\bar{G}_3 \bar{G}_2 + G_3 G_2)$

$= \bar{G}_1 \bar{G}_0 (G_3 \oplus G_2) + \bar{G}_1 G_0 \overline{(G_3 \oplus G_2)} + G_1 G_0 (G_3 \oplus G_2) + G_1 \bar{G}_0 \overline{(G_3 \oplus G_2)}$

$= (G_3 \oplus G_2) + (\bar{G}_1 \bar{G}_0 + G_1 G_0) + \overline{(G_3 \oplus G_2)} (\bar{G}_1 G_0 + G_1 \bar{G}_0)$

$= (G_3 \oplus G_2) \overline{(G_1 \oplus G_0)} + \overline{(G_3 \oplus G_2)} (G_1 \oplus G_0)$

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

$= G_0 \oplus G_1 \oplus G_2 \oplus G_3$

**Fig. 5.29 (a)**

Scanned by CamScanner

**(3) Circuit :**

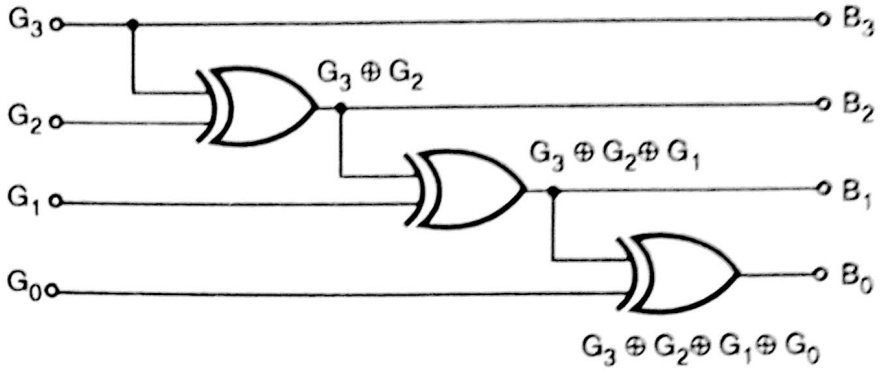



**Fig. 5.29 (b)**

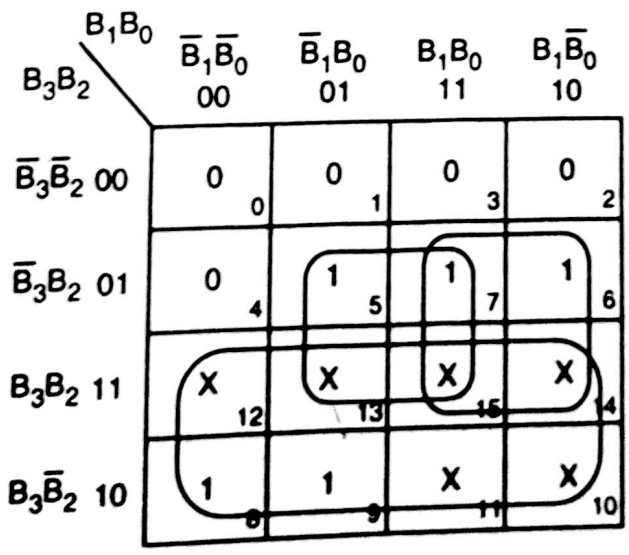
**Ex. 25 :** Convert BCD (binary coded decimal–4 bit) to excess – 3 (XS–3).

**Soln. :** BCD is valid upto $(9)_{10}$ only. But as we are taking 4 bits, states will be 0 to 15.Therefore states from $(10)_{10}$ to $(15)_{10}$ the output will be 'X' (don't care) because we know that for valid BCD, these codes (10 – 15) will not occur. Table 26 shows the truth table.

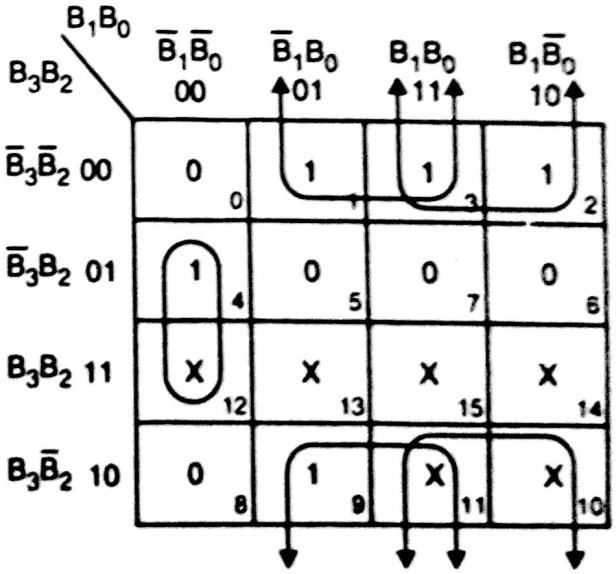
| Decimal | $B_3$ | $B_2$ | $B_1$ | $B_0$ | $E_3$ | $E_2$ | $E_1$ | $E_0$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 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 |

**Table 26**

**Now draw K-maps :**




$$E_3 = B_3 + B_2 (B_0 + B_1)$$




$$E_2 = B_2\bar{B}_1\bar{B}_0 + \bar{B}_2B_0 + \bar{B}_2 B_1$$
$$= B_2\bar{B}_1\bar{B}_0 + \bar{B}_2 (B_0+B_1)$$

Scanned by CamScanner

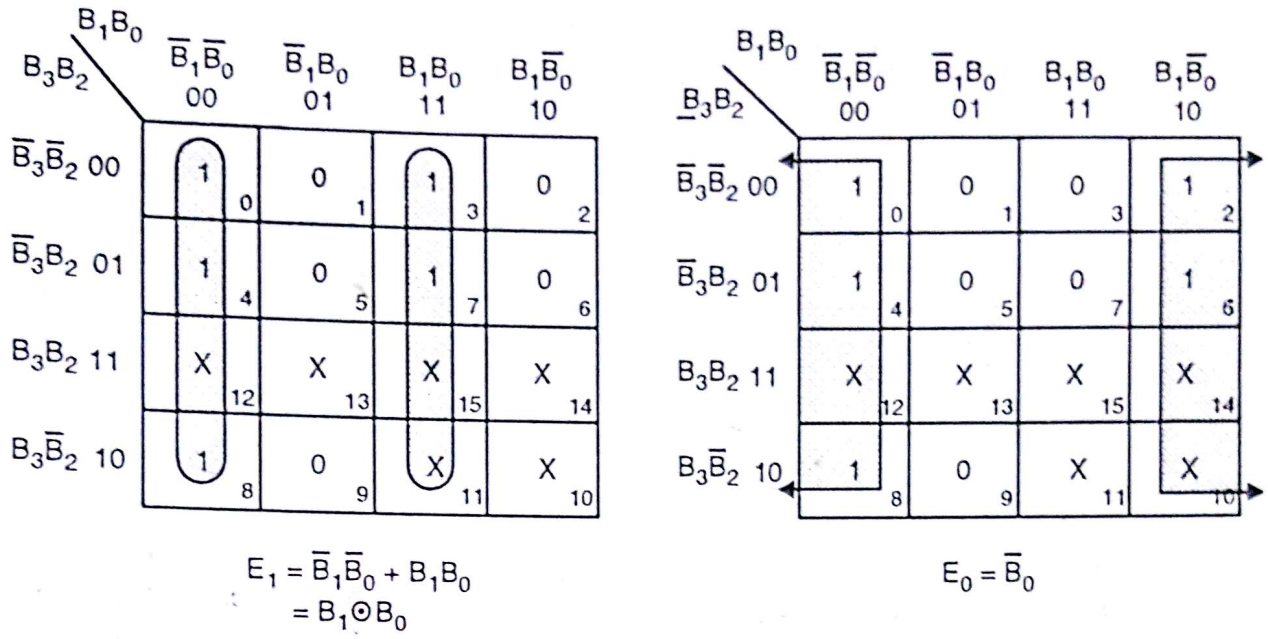



Fig. 5.30 (a)

**Circuit :**

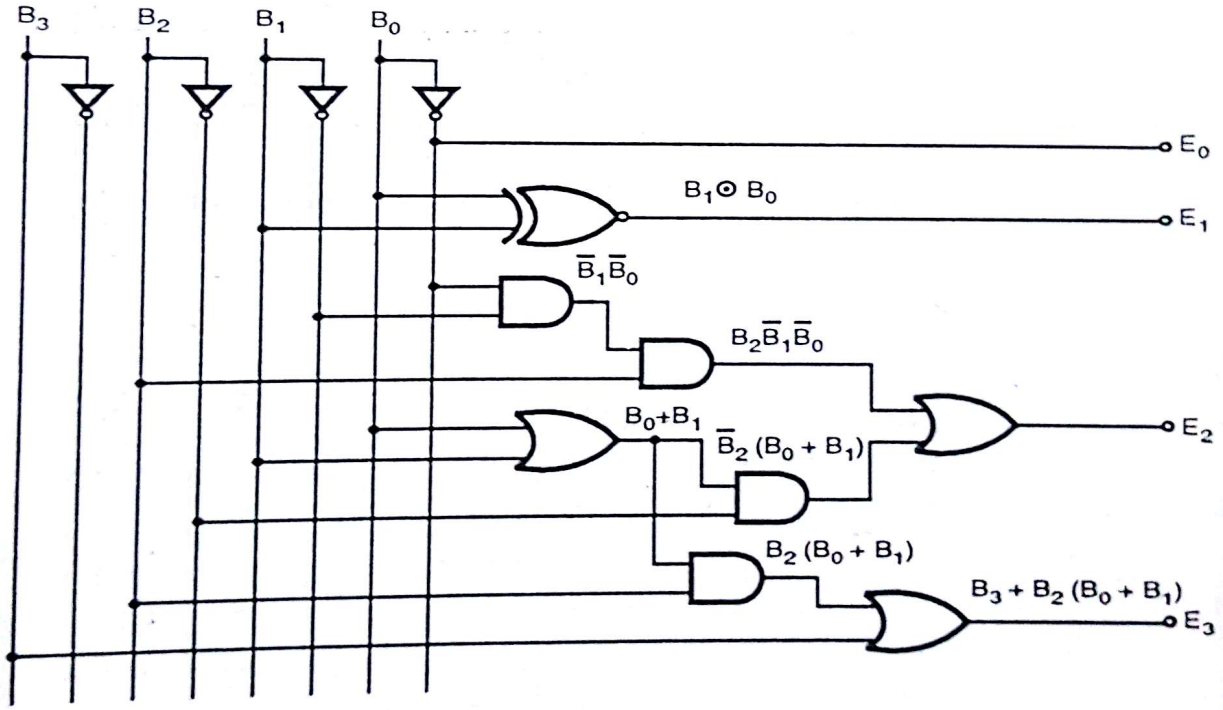



Fig. 5.30 (b)

**Ex. 26 :** Convert BCD (single digit – 4 bit) to gray. Design logic circuit for the same.

**Soln. :** Valid BCD is from 0 to 9. AH to FH are invalid codes, therefore Output will be 'X' (don't care).

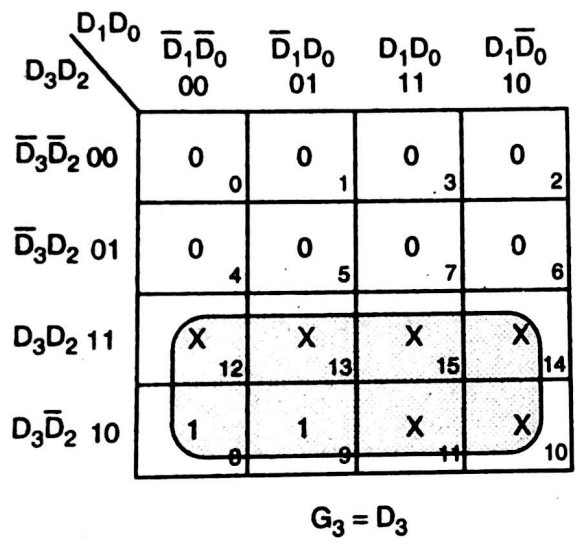
Table 27 shows BCD to gray truth table.

Scanned by CamScanner

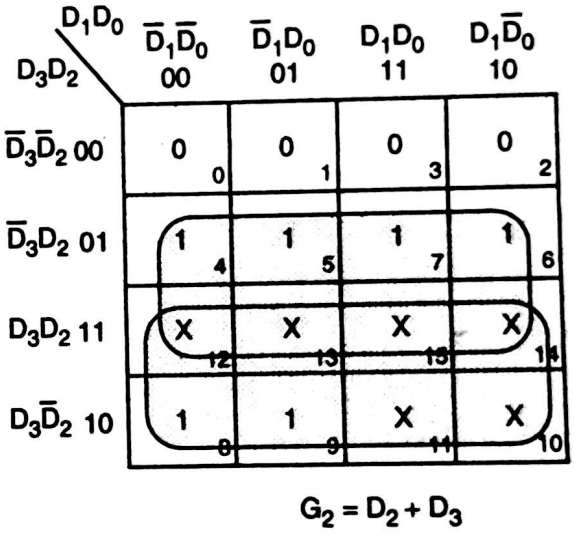
| Decimal | $D_3$ | $D_2$ | $D_1$ | $D_0$ | $G_3$ | $G_2$ | $G_1$ | $G_0$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

**Table 27**

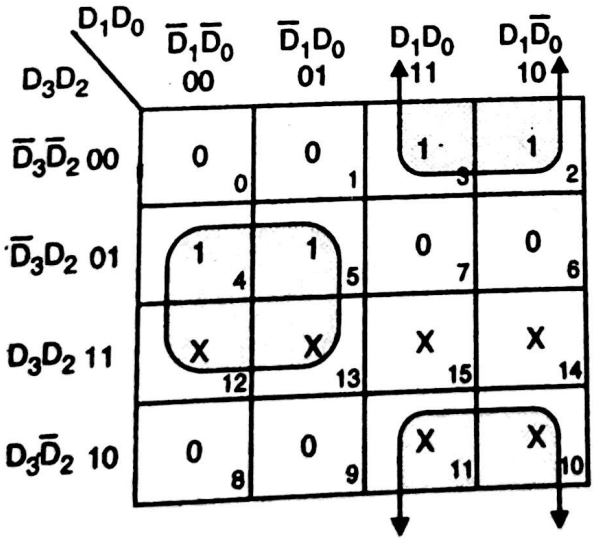
**Now draw K-maps :**



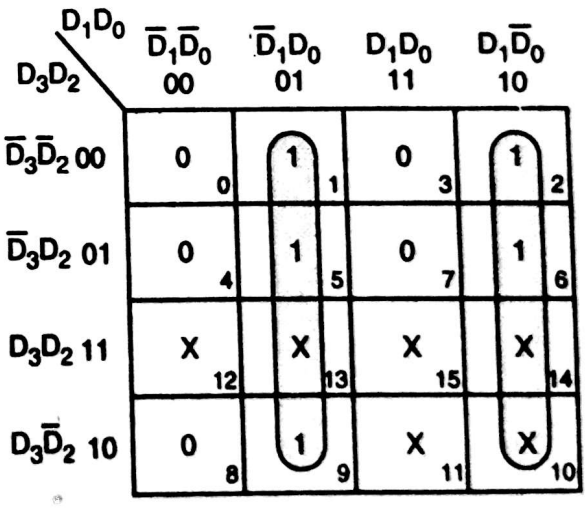

$G_3 = D_3$




$G_2 = D_2 + D_3$




$G_1 = D_2\bar{D}_1 + \bar{D}_2 D_1$
$= D_2 \oplus D_1$




$G_0 = \bar{D}_1 D_0 + D_1 \bar{D}_0$
$= D_1 \oplus D_0$

**Fig. 5.31 (a)**

Scanned by CamScanner

**Circuit :**

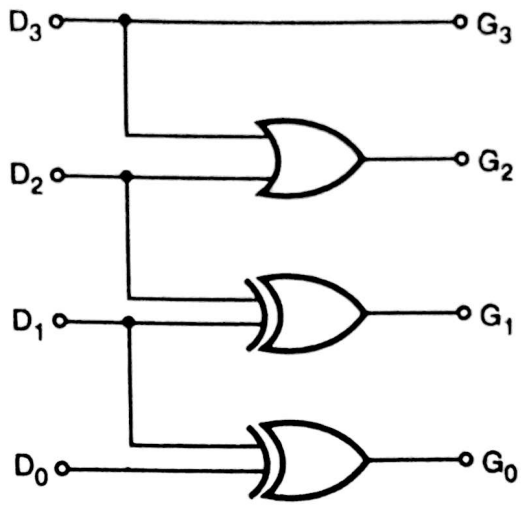



**Fig. 5.31 (b)**

**Ex. 27 :** Convert binary to BCD. Design circuit which will accept 4 bit binary and will provide 5 bit BCD code.

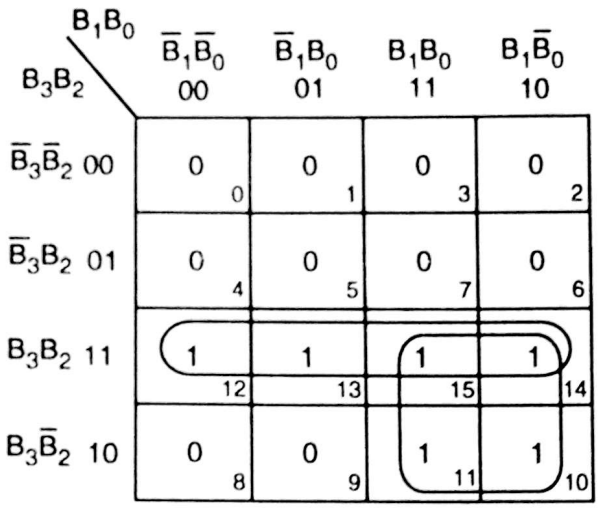
**Soln. :** Table 28 shows truth table for binary to BCD conversion.

| Decimal | $B_3$ $B_2$ $B_1$ $B_0$ | $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ |
|---|---|---|
| 0 | 0 0 0 0 | 0 0 0 0 0 |
| 1 | 0 0 0 1 | 0 0 0 0 1 |
| 2 | 0 0 1 0 | 0 0 0 1 0 |
| 3 | 0 0 1 1 | 0 0 0 1 1 |
| 4 | 0 1 0 0 | 0 0 1 0 0 |
| 5 | 0 1 0 1 | 0 0 1 0 1 |
| 6 | 0 1 1 0 | 0 0 1 1 0 |
| 7 | 0 1 1 1 | 0 0 1 1 1 |
| 8 | 1 0 0 0 | 0 1 0 0 0 |
| 9 | 1 0 0 1 | 0 1 0 0 1 |
| 10 | 1 0 1 0 | 1 0 0 0 0 |
| 11 | 1 0 1 1 | 1 0 0 0 1 |
| 12 | 1 1 0 0 | 1 0 0 1 0 |
| 13 | 1 1 0 1 | 1 0 0 1 1 |
| 14 | 1 1 1 0 | 1 0 1 0 0 |
| 15 | 1 1 1 1 | 1 0 1 0 1 |

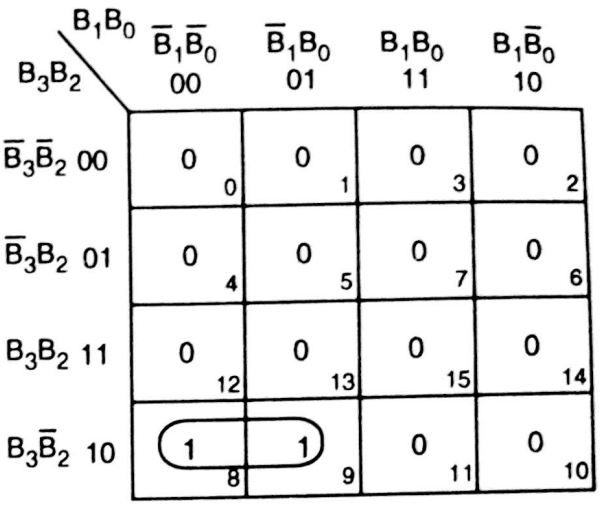
**Table 28**

Scanned by CamScanner

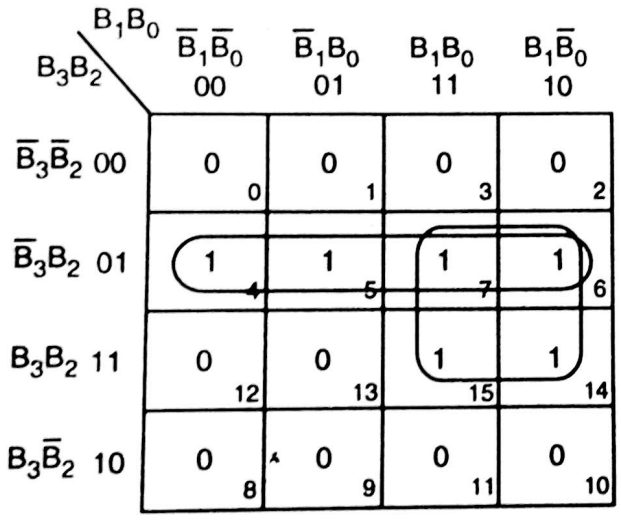
**K–map :**



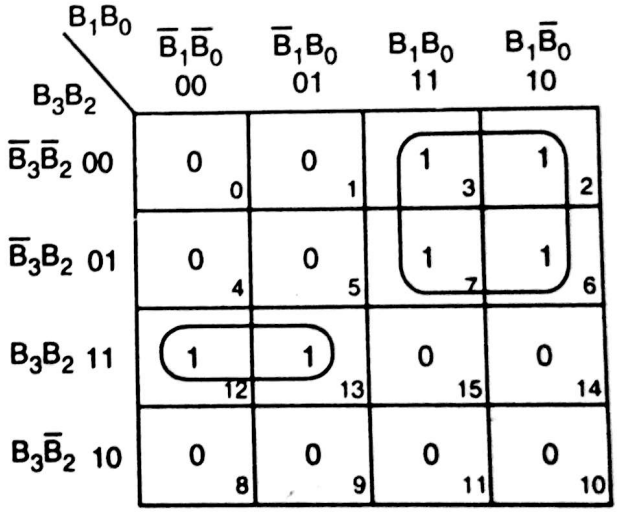

$D_4 = B_3 B_2 + B_3 B_1$
$= B_3 (B_2 + B_1)$



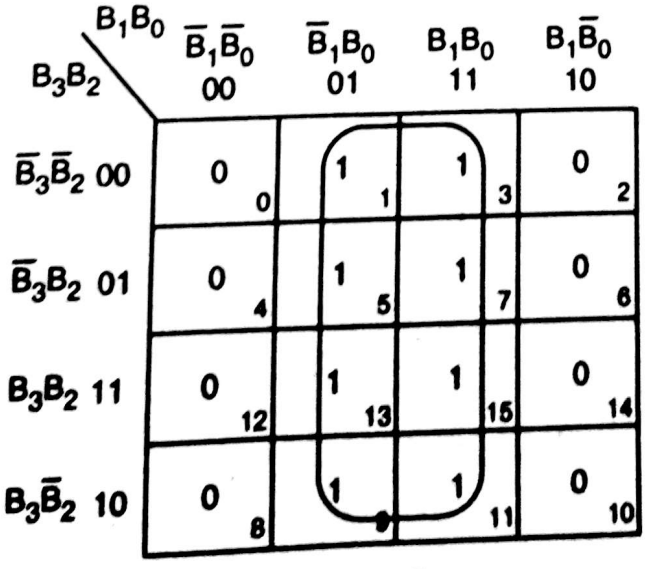

$D_3 = B_3 \bar{B}_2 \bar{B}_1$




$D_2 = \bar{B}_3 B_2 + B_2 B_1$
$= B_2 (\bar{B}_3 + B_1)$




$D_1 = B_3 B_2 \bar{B}_1 + \bar{B}_3 B_1$




$D_0 = B_0$

**Fig. 5.32 (a)**

Scanned by CamScanner

Circuit :

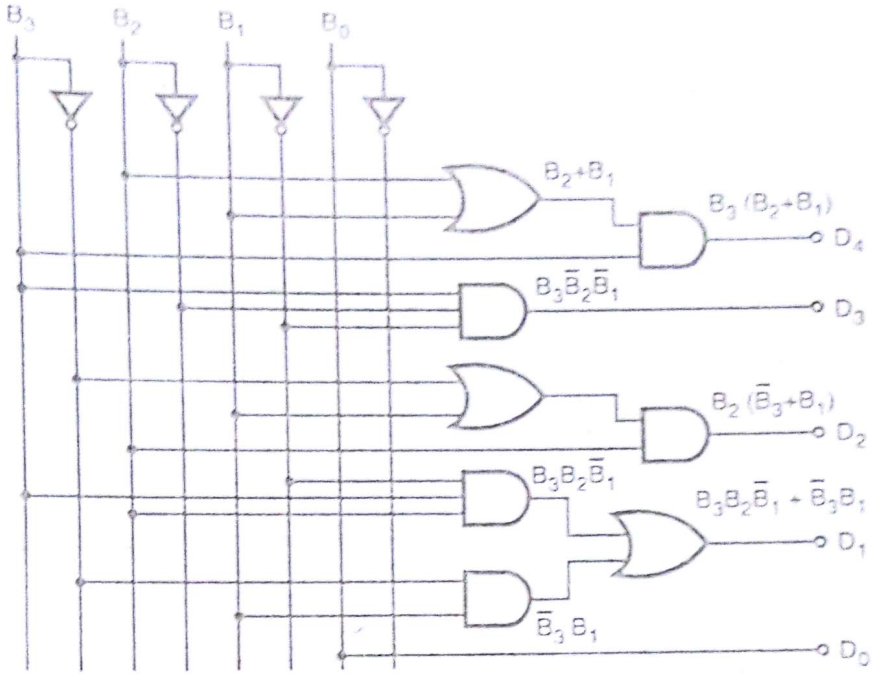



Fig. 5.32 (b)

**Ex. 28 :** Design a combinational circuit whose input is a four bit number and output is the two's complement of the input number.

**Soln. :** The truth table is illustrated in Table 29.

| | Input | | | | Output | | | |
|---|---|---|---|---|---|---|---|---|
| Decimal | $A_3$ | $A_2$ | $A_1$ | $A_0$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Table 29

Scanned by CamScanner

The K map is illustrated in Fig. 5.33 (a). The circuit diagram is shown in Fig. 5.33 (b).

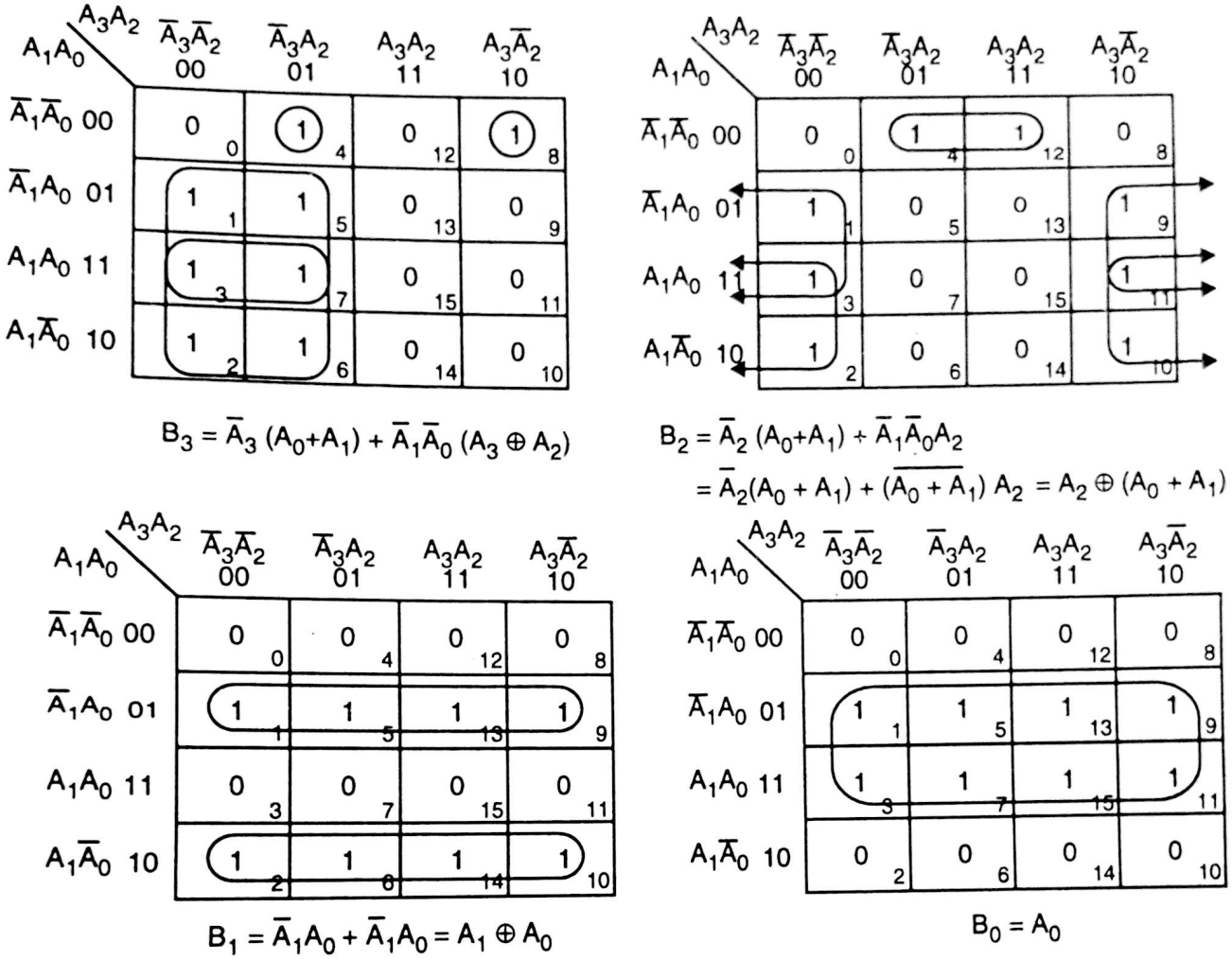



**Fig. 5.33 (a)**

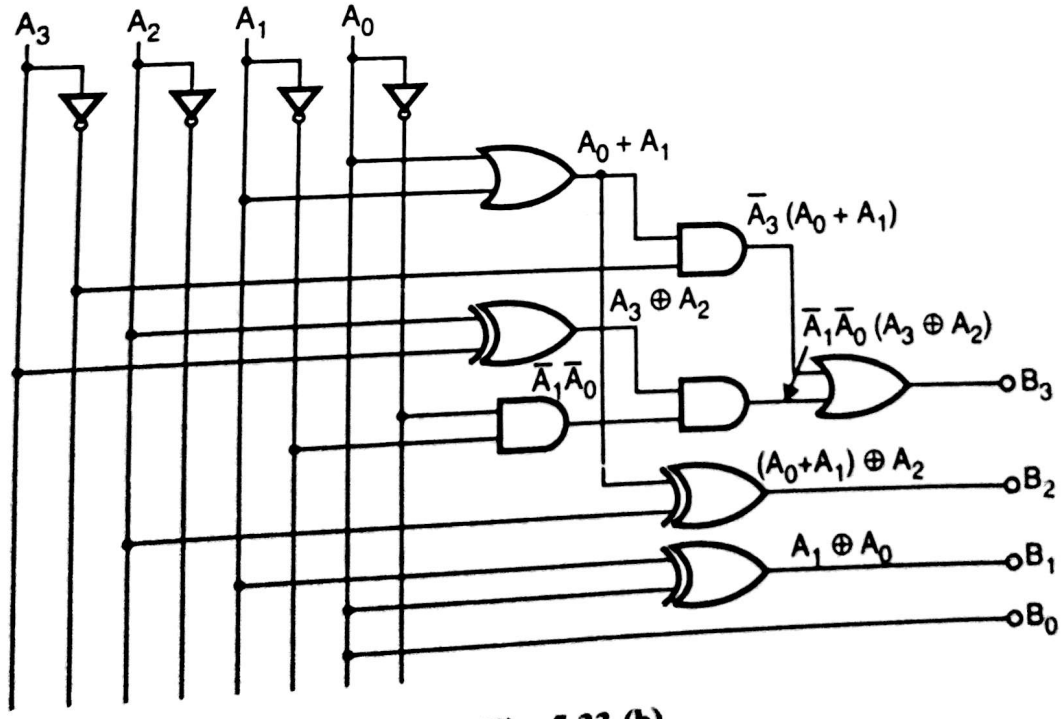



**Fig. 5.33 (b)**

Scanned by CamScanner

**Ex. 29 :** A 2-bit digital comparator accepts two words A = $A_2A_1$ and B = $B_2B_1$ and gives three outputs G, E and L.

(i) The output G is HIGH when A > B

(ii) The output E is HIGH when A = B

(iii) The output L is HIGH when A < B

(a) Write the truth table for this comparator.

(b) Draw the Karnaugh maps for G, L and E outputs and write the SOP expressions for each.

(c) Draw logic diagram of this comparactor.

**Soln. :** G : One quad two pairs

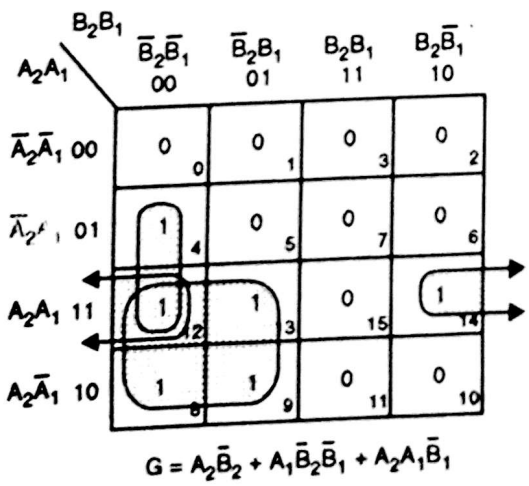
L : One quad two pairs

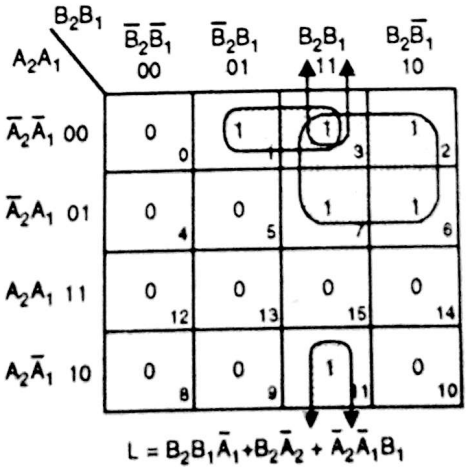
E : Four minterms.

| Inputs | | | | Output | | |
|---|---|---|---|---|---|---|
| $A_2$ | $A_1$ | $B_2$ | $B_1$ | G | L | E |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |

**Table 30**

**K–map :**



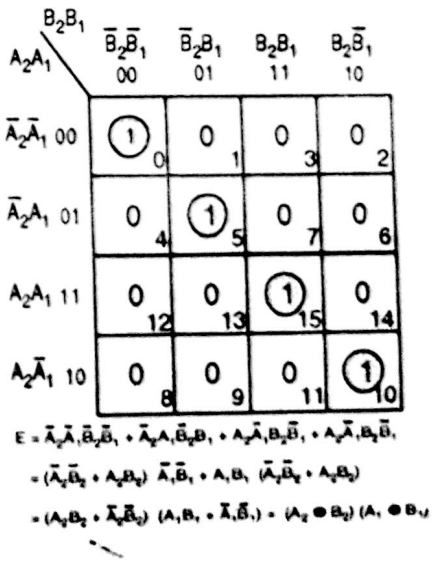




**Fig. 5.34 (a)**

Scanned by CamScanner

**Circuit :**

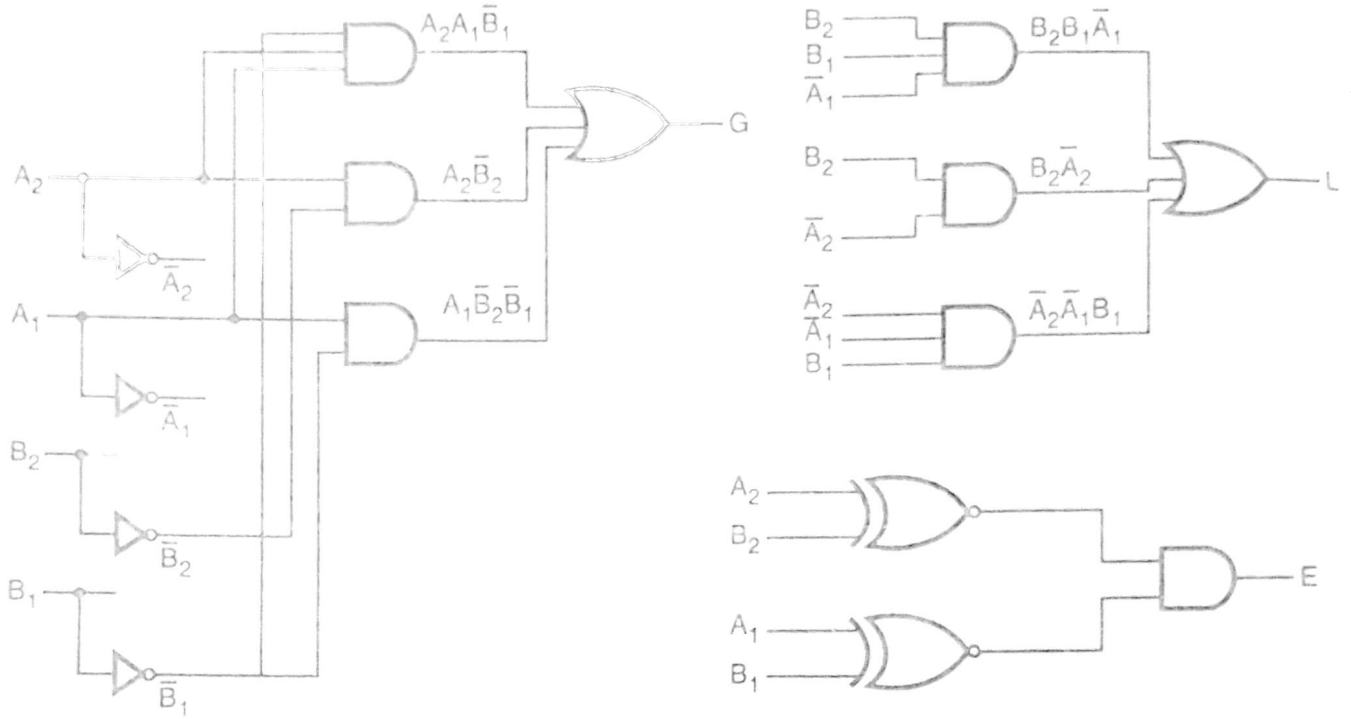



Fig. 5.34 (b)

**Ex. 30 :** Convert XS–3 (4 bit) to BCD (4 bit).

**Soln. :** The table 31 shows XS–3 code to BCD.

| Inputs | | | | Outputs | | | |
|---|---|---|---|---|---|---|---|
| $E_3$ | $E_2$ | $E_1$ | $E_0$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Table 31

Scanned by CamScanner

**K–map :**

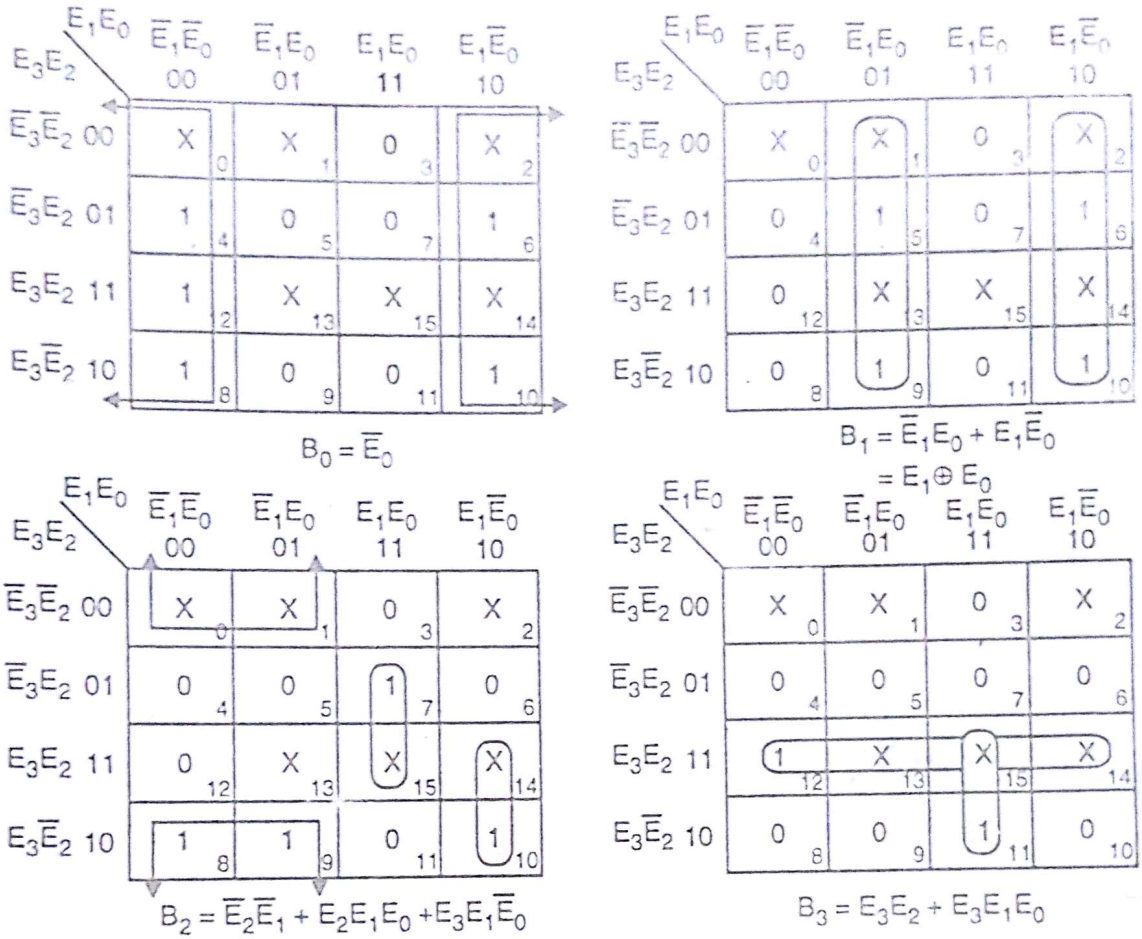



Fig. 5.35 (a)

**Circuit :**

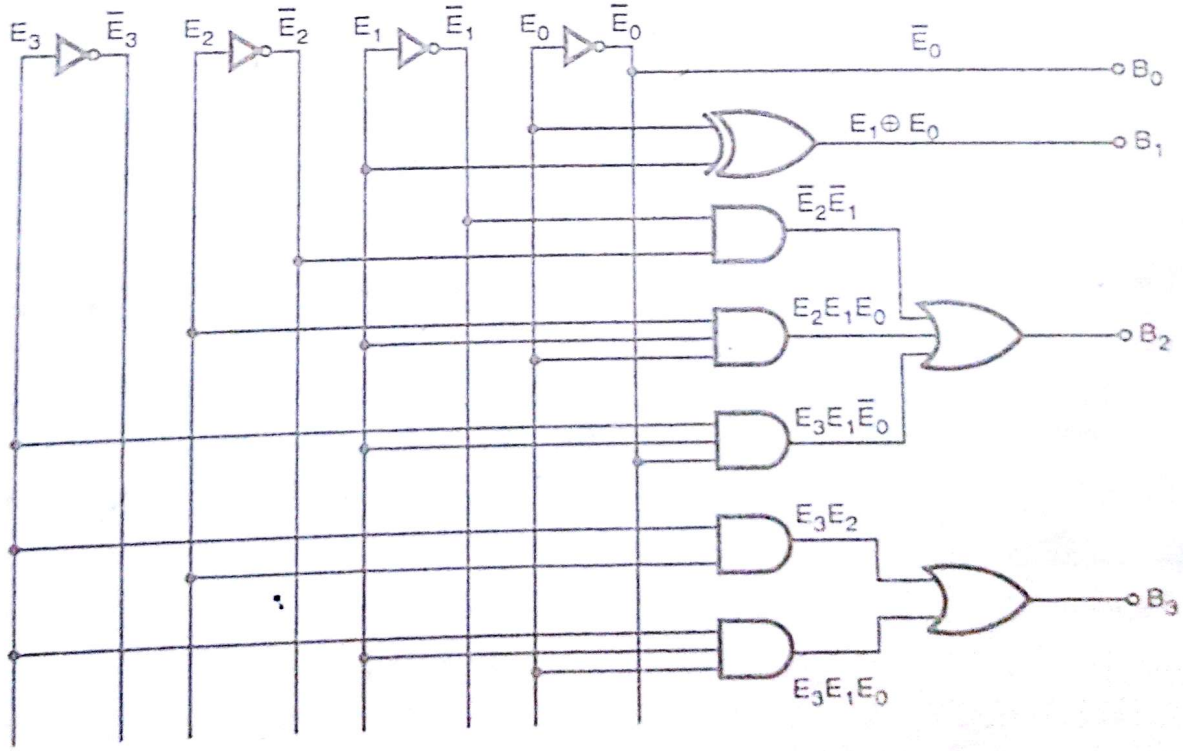



Fig. 5.35 (b)

Scanned by CamScanner

**Ex. 31 :** Design a BCD to 7 segment code converter circuit. Assume 7 segment displays available are of common anode type. Draw a complete, self explanatory logic diagram. Consider the configuration of segments and display of digits as shown in Fig. 5.36.

**Soln. : Description of 7 segment display :** Before we start solving the example, I would like to give brief about 7 segment display. At this stage I hope you are aware with LED. (Light emitting diode). In 7 segment, as name indicates, we have 7 LED segment with one decimal point LED. The arrangement is shown in Fig. 5.37. The arrangement is like English eight (8). The reason for this is, with this arrangement you can display digits (0 – 9) and some alphanumeric characters like A, b, C, d, e, F, G, H and so on. (Refer Fig. 5.36. It shows the LED's supposed to be ON to display 0 to 9). Here total we have eight LED (Including dp → decimal point). Therefore we have 16 terminals (8 anode, 8 cathode). So here to make circuit simple we can make one the terminal i.e. A or K common. Presently in example they have asked for common anode. Fig. 5.37 (b) will show you common anode connection. In this anodes of all LED's are tied together and given to $V_{CC}$. Whenever you want to lit LED simply ground cathode (k) through resistance, that particular LED will be ON. The LED segments are normally referred as a, b, c, d, e, f, g, dp.



**Fig. 5.36**

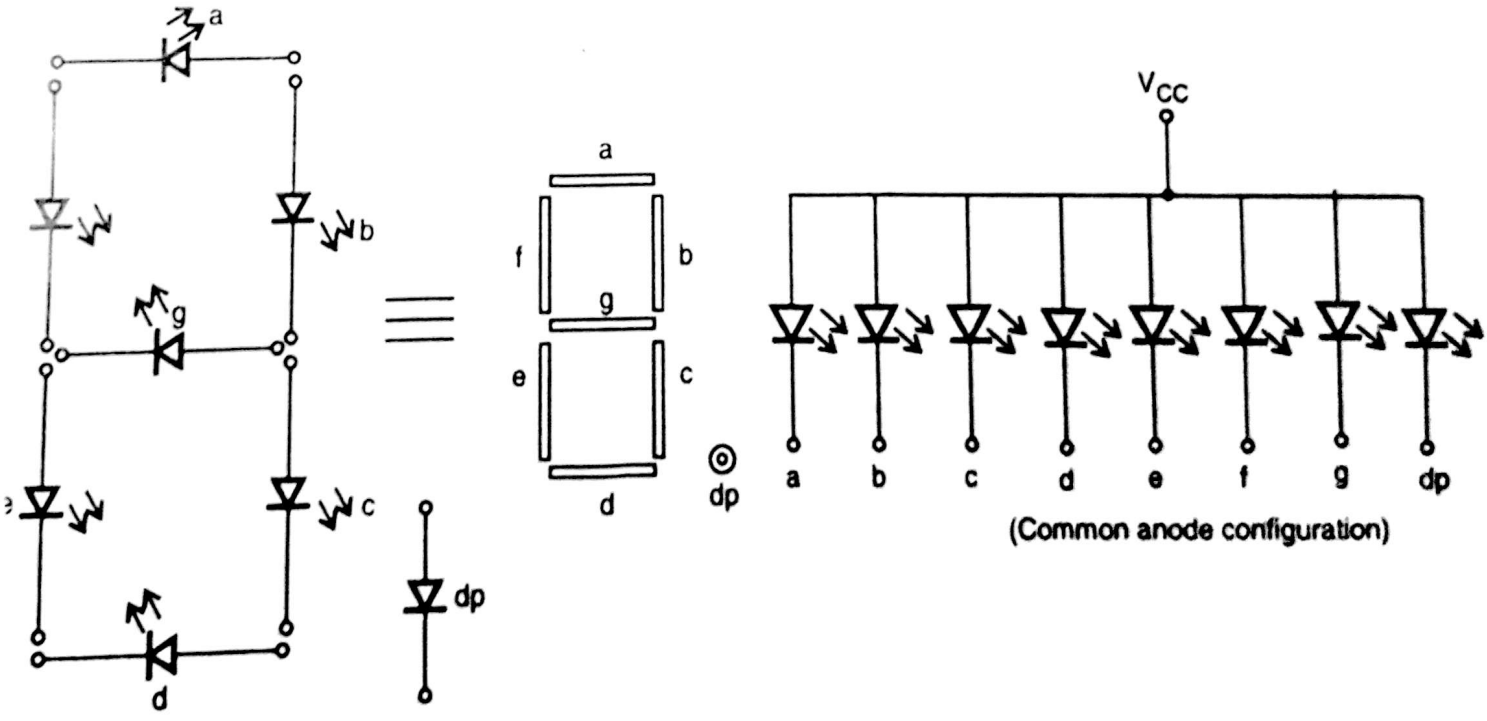



**Fig. 5.37**

Scanned by CamScanner

**Aim :** Our aim is to design a circuit which will convert given BCD i.e. $(0000)_2$ to $(1001)_2$, to its equivalent 7 segment code. I will give you one example. Find 7 segment code for BCD $(2)_{10}$ i.e. $(0010)_2$. Refer Fig. 5.38 (a).

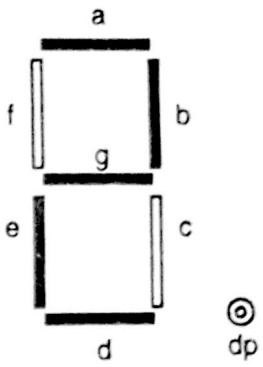



**Fig. 5.38 (a)**

As shown in Fig. 5.38(a), we have to 'lit' a, b, g, e and d LEDS. All others will be OFF. You know that to lit LED you have to force segment to logic '0'.

∴ Code of 7 segment for BCD 2 is

| a | b | c | d | e | f | g | dp |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| ↑ | ↑ | | ↑ | ↑ | | ↑ | |

LED ON ∴ display 2

Therefore, now we have to prepare truth table where BCD inputs are $B_3$ $B_2$ $B_1$ $B_0$ and outputs are a, b, c, d, e, f, g, dp.

Table 32 shows truth table.

| Decimal | Inputs | | | | Outputs | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $B_3$ | $B_2$ | $B_1$ | $B_0$ | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Table 32**

**Now next step is to drawn K–map we have to draw 7 K-maps.**

Scanned by CamScanner

| $B_3B_2$ \ $B_1B_0$ | $\bar{B}_1\bar{B}_0$ 00 | $\bar{B}_1B_0$ 01 | $B_1B_0$ 11 | $B_1\bar{B}_0$ 10 |
|---|---|---|---|---|
| $\bar{B}_3\bar{B}_2$ 00 | 0 (0) | 1 (1) | 0 (3) | 0 (2) |
| $\bar{B}_3B_2$ 01 | 1 (4) | 0 (5) | 0 (7) | 0 (6) |
| $B_3B_2$ 11 | X (12) | X (13) | X (15) | X (14) |
| $B_3\bar{B}_2$ 10 | 0 (8) | 0 (9) | X (11) | X (10) |

$$a = B_2\bar{B}_1\bar{B}_0 + \bar{B}_3\bar{B}_2\bar{B}_1B_0$$

| $B_3B_2$ \ $B_1B_0$ | $\bar{B}_1\bar{B}_0$ 00 | $\bar{B}_1B_0$ 01 | $B_1B_0$ 11 | $B_1\bar{B}_0$ 10 |
|---|---|---|---|---|
| $\bar{B}_3\bar{B}_2$ 00 | 0 (0) | 0 (1) | 0 (3) | 0 (2) |
| $\bar{B}_3B_2$ 01 | 0 (4) | 1 (5) | 0 (7) | 1 (6) |
| $B_3B_2$ 11 | X (12) | X (13) | X (15) | X (14) |
| $B_3\bar{B}_2$ 10 | 0 (8) | 0 (9) | X (11) | X (10) |

$$b = B_2\bar{B}_1B_0 + B_2B_1\bar{B}_0$$

| $B_3B_2$ \ $B_1B_0$ | $\bar{B}_1\bar{B}_0$ 00 | $\bar{B}_1B_0$ 01 | $B_1B_0$ 11 | $B_1\bar{B}_0$ 10 |
|---|---|---|---|---|
| $\bar{B}_3\bar{B}_2$ 00 | 0 (0) | 0 (1) | 0 (3) | 1 (2) |
| $\bar{B}_3B_2$ 01 | 0 (4) | 0 (5) | 0 (7) | 0 (6) |
| $B_3B_2$ 11 | X (12) | X (13) | X (15) | X (14) |
| $B_3\bar{B}_2$ 10 | 0 (8) | 0 (9) | X (11) | X (10) |

$$c = \bar{B}_2B_1\bar{B}_0$$

| $B_3B_2$ \ $B_1B_0$ | $\bar{B}_1\bar{B}_0$ 00 | $\bar{B}_1B_0$ 01 | $B_1B_0$ 11 | $B_1\bar{B}_0$ 10 |
|---|---|---|---|---|
| $\bar{B}_3\bar{B}_2$ 00 | 0 (0) | 1 (1) | 0 (3) | 0 (2) |
| $\bar{B}_3B_2$ 01 | 1 (4) | 0 (5) | 1 (7) | 0 (6) |
| $B_3B_2$ 11 | X (12) | X (13) | X (15) | X (14) |
| $B_3\bar{B}_2$ 10 | 0 (8) | 0 (9) | X (11) | X (10) |

$$d = \bar{B}_3\bar{B}_2\bar{B}_1B_0 + B_2\bar{B}_1\bar{B}_0 + B_2(B_1B_0)$$

| $B_3B_2$ \ $B_1B_0$ | $\bar{B}_1\bar{B}_0$ 00 | $\bar{B}_1B_0$ 01 | $B_1B_0$ 11 | $B_1\bar{B}_0$ 10 |
|---|---|---|---|---|
| $\bar{B}_3\bar{B}_2$ 00 | 0 (0) | 1 (1) | 1 (3) | 0 (2) |
| $\bar{B}_3B_2$ 01 | 1 (4) | 1 (5) | 1 (7) | 0 (6) |
| $B_3B_2$ 11 | X (12) | X (13) | X (15) | X (14) |
| $B_3\bar{B}_2$ 10 | 0 (8) | 1 (9) | X (11) | X (10) |

$$e = B_0 + B_2\bar{B}_1$$

Scanned by CamScanner

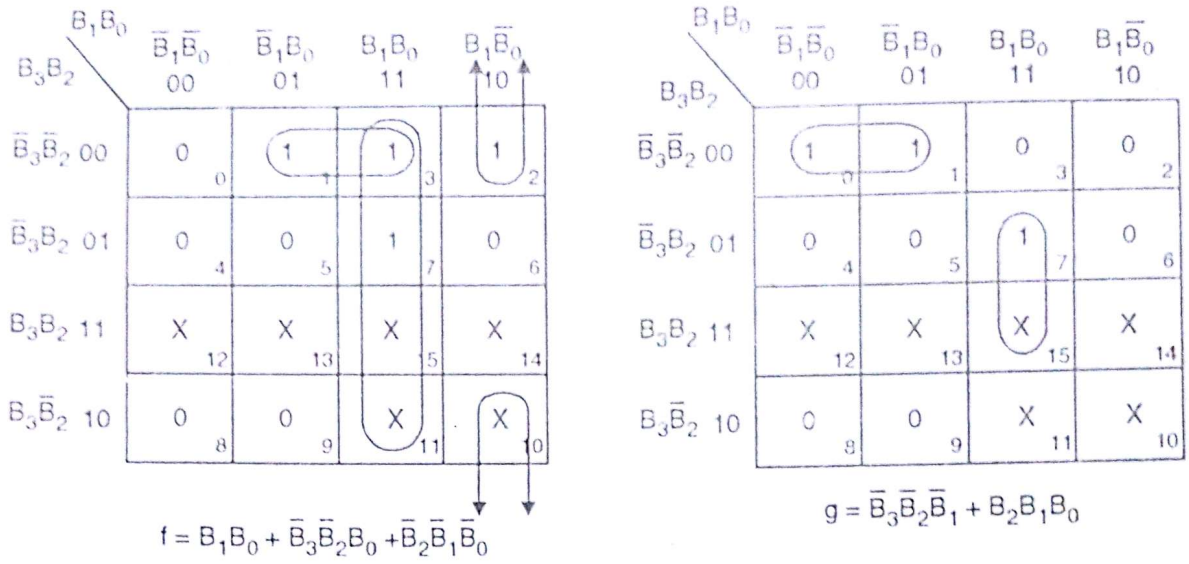



Fig. 5.39 (a)

Circuit :

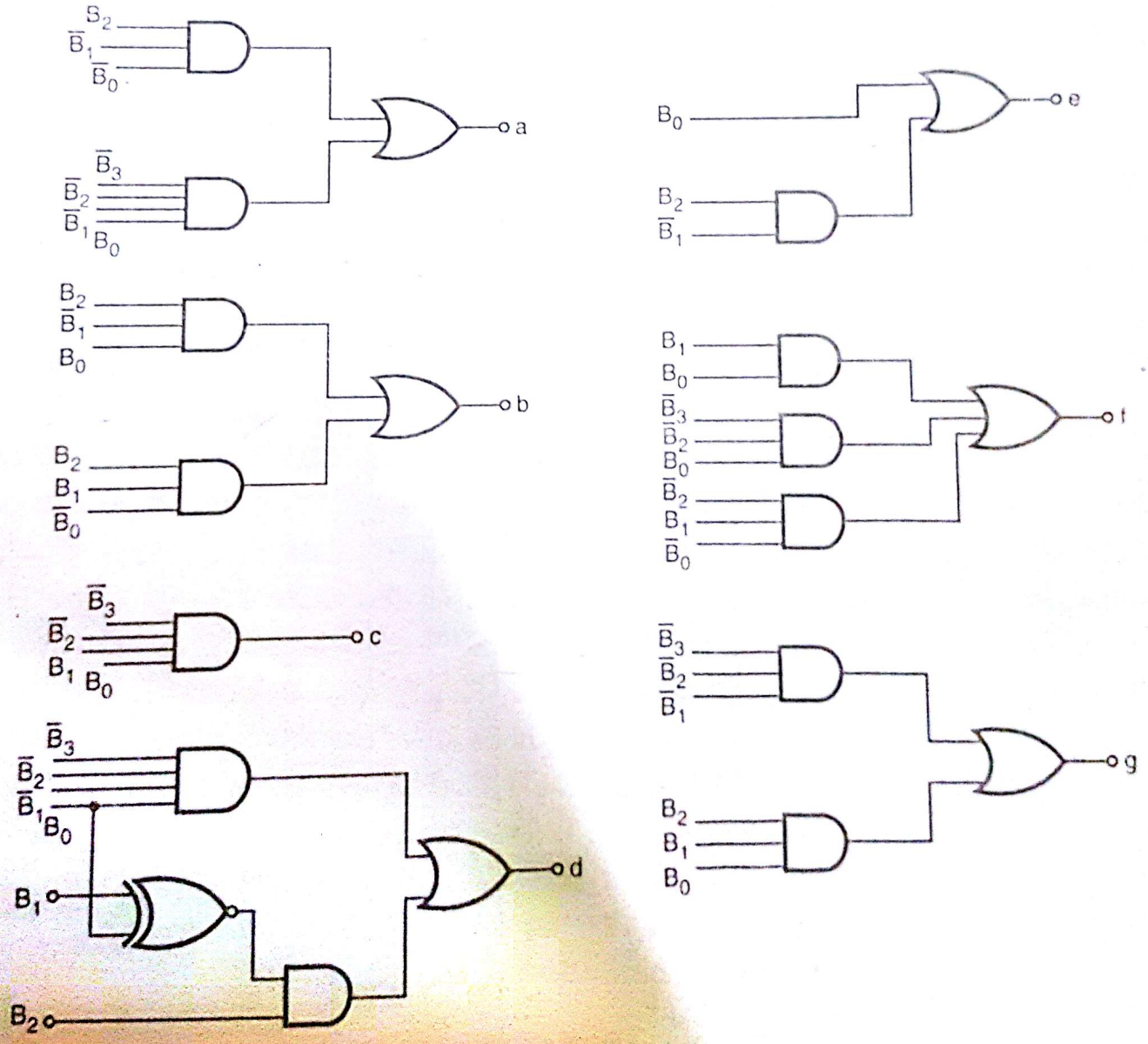



Fig. 5.39 (b)

Scanned by CamScanner

## 5.4 Examples from University Papers :

**1. As shown in Fig. Q.5.1 a combinational logic circuit is required for specifications given below :**

**Input four lines $A_3$, $A_2$, $A_1$, $A_0$ ($A_3$ being most significant) carry binary equivalent of decimal digit 0–9. Output Y is '1' when input contains two or more bits '1'. Otherwise output Y = '0'. Implement the necessary logic using NOR gates.**

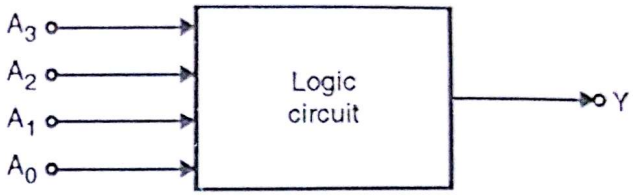



Fig. Q.5.1

**(Dec. 96, 10 Marks)**

**Ans.:** Refer Section 5.1, Ex. 1.

**2. As shown in Fig. Q.5.2 (a) a combinational logic circuit is required to satisfy following requirements. Input lines $A_3$, $A_2$, $A_1$, $A_0$ ($A_3$ being most significant) carry binary equivalent of decimal digit 0 - 9. Output Y is low when at the input only 2 bits are high otherwise Y = '1'. Implement the necessary logic using NAND gates.**

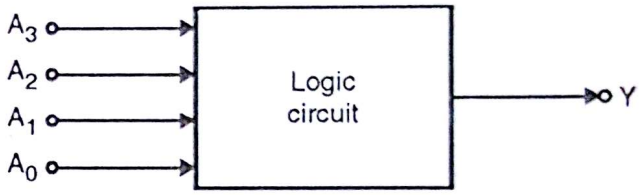



Fig. Q. 5.2 (a)

**(Dec. 97, 10 Marks)**

**Ans. :** (a) Number of input lines are $A_3$, $A_2$, $A_1$, $A_0$.

(b) Number of input combinations for 4 bits will be $2^n = 2^4 = 16$

(c) But in the problem it is mentioned that it carries binary equivalent of decimal digit $0 - 9$.

Therefore, remaining states i.e. 10 to 15 will be treated as Don't care condition.

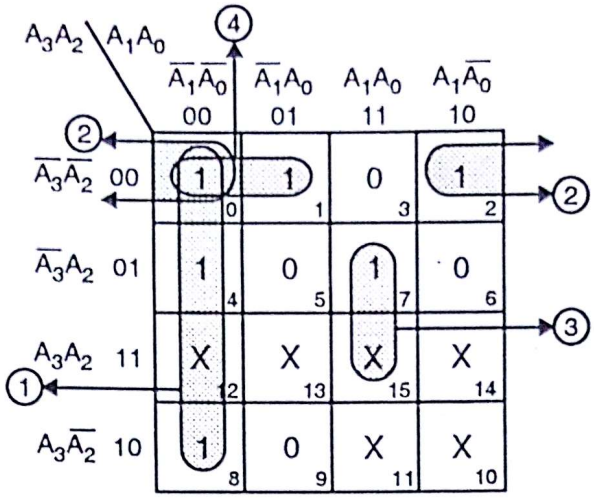
Now prepare truth table for this example.

**Truth Table 1**

| Decimal | Inputs | | | | Output |
|---|---|---|---|---|---|
| | $A_3$ | $A_2$ | $A_1$ | $A_0$ | Y |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | X |
| 11 | 1 | 0 | 1 | 1 | X |
| 12 | 1 | 1 | 0 | 0 | X |
| 13 | 1 | 1 | 0 | 1 | X |
| 14 | 1 | 1 | 1 | 0 | X |
| 15 | 1 | 1 | 1 | 1 | X |

Scanned by CamScanner

**(d) Now draw K–map, encircle 1's therefore answer will be in SOP form.**




∴ From K-map output equation,

$$Y = \bar{A}_1\bar{A}_0 + \bar{A}_3\bar{A}_2\bar{A}_0 + A_2A_1A_0$$ **...Ans.**

We have to implement above equation using NAND gates.

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

∴ $Y = \overline{\bar{A}_1\bar{A}_0 + \bar{A}_3\bar{A}_2\bar{A}_0 + A_2A_1A_0}$

$Y = \overline{\bar{A}_1\bar{A}_0} \cdot \overline{\bar{A}_3\bar{A}_2\bar{A}_0} \cdot \overline{A_2A_1A_0}$

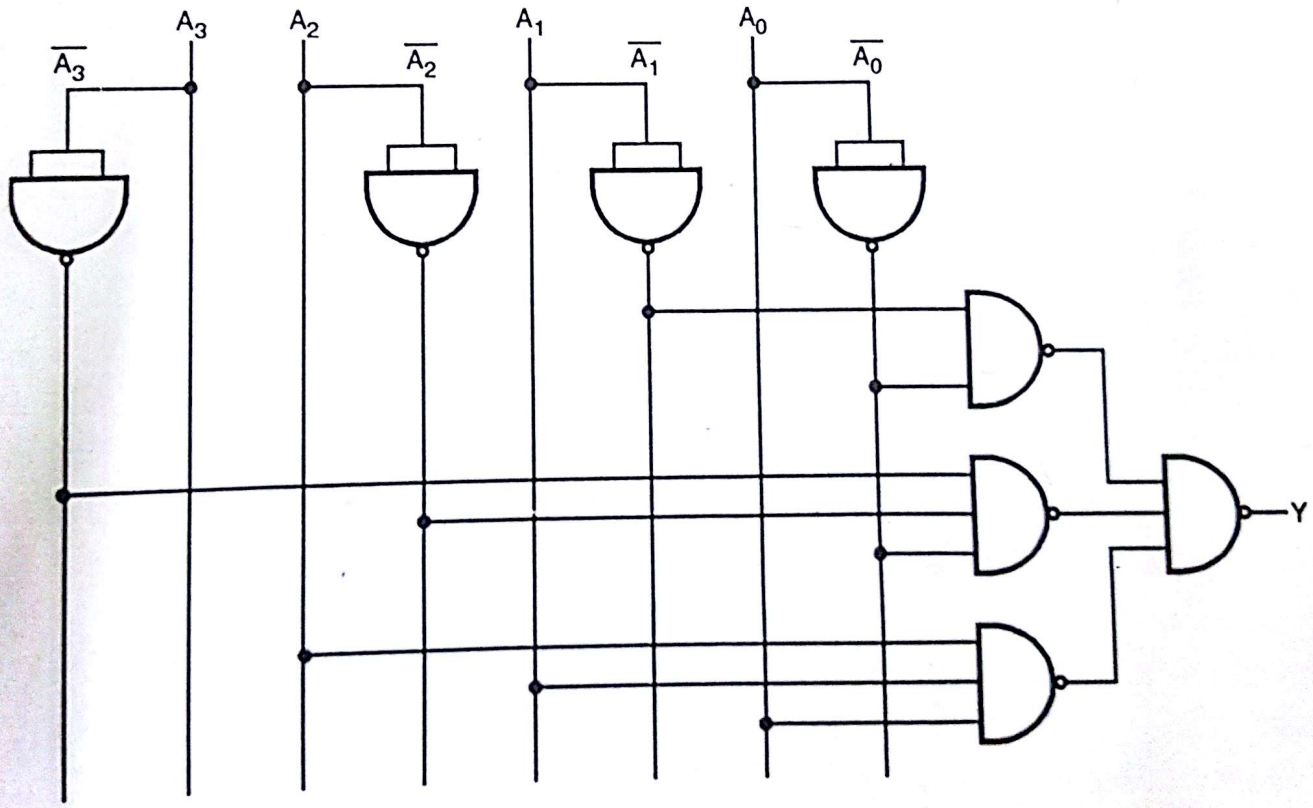



**Fig. Q. 5.2 (b)**

Scanned by CamScanner

3. Make a truth table of function that is true when two or more of four variables are true.

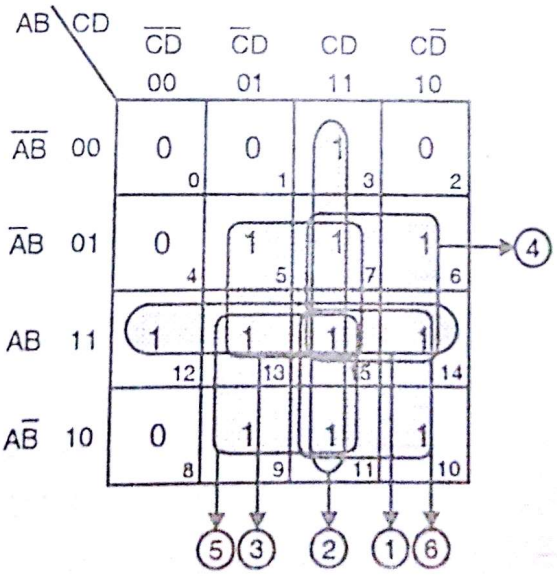
(Dec. 98, 8 Marks)

Ans.: As it is given that the function contains 4 variable. Therefore the circuit must have 4 input lines and one output. Number of input combinations for 4 bits will be $2^n = 2^4 = 16$.

Truth Table 2

| Decimal | Inputs | | | | Output |
|---|---|---|---|---|---|
| | A | B | C | D | Y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 1 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 1 |

From Truth Table 2, make a K–map for the above function




∴ Output equation from K-map

Scanned by CamScanner

$$Y = \underset{(1)}{AB} + \underset{(2)}{CD} + \underset{(3)}{BD} + \underset{(4)}{BC} + \underset{(5)}{AD} + \underset{(6)}{AC}$$

...Ans.

4. A 4–bit binary number is represented as $A_3, A_2, A_1, A_0$ where $A_3, A_2, A_1$ and $A_0$ represent the individual bits with $A_0$ equals to the LSB. Design a logic circuit that will produce a HIGH output whenever binary number is greater than $(0010)_2$ and less than $(1000)_2$.

(June 99, 2 Marks)

Ans.: Refer Section 5.1, Ex. 3.

5. Design a combinational logic circuit for the following description :

The circuit has four inputs and two outputs. One of the outputs is to be true only when the majority of inputs is true. The other output is to be true only when there is a tie between the inputs is true. Implement the above logic using NAND gates only. (June 99, 10 Marks)

Ans.: Refer Section 5.2, Ex. 16.

6. Design and implement a combinational logic circuit for the following description, using NAND gates only. The circuit has four inputs and two outputs. One of the outputs is to be true when the majority of inputs are false. The other output is true only when there are equal number of true and false in the inputs. (Dec. 99, 10 Marks)

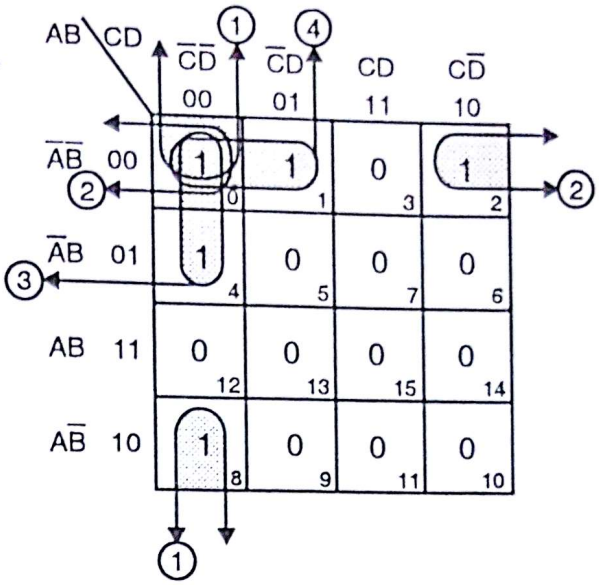
Ans.: As there are four inputs so there will be $2^n = 2^4 = 16$ input lines.

Truth Table 3

| Decimal | Inputs | | | | Outputs | |
|---|---|---|---|---|---|---|
| | A | B | C | D | $Y_1$ | $Y_2$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | 0 | 0 |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 |
| 13 | 1 | 1 | 0 | 1 | 0 | 0 |
| 14 | 1 | 1 | 1 | 0 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 0 | 0 |

Scanned by CamScanner

Now prepare K–map for the Truth Table 3



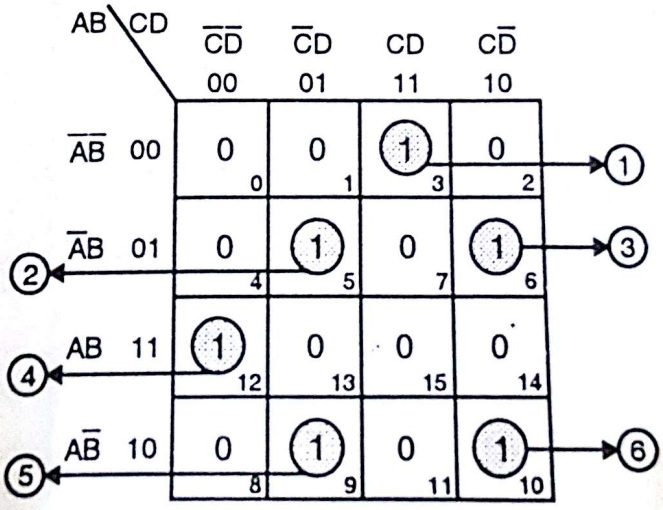

Output equation for above K–map,

$$Y_1 = \underset{(1)}{\overline{B}\overline{C}\overline{D}} + \underset{(2)}{\overline{A}\overline{B}\overline{D}} + \underset{(3)}{\overline{A}\overline{C}\overline{D}} + \underset{(4)}{\overline{A}\overline{B}\overline{C}}$$

$$\therefore \quad Y_1 = \overline{\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{D} + \overline{A}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}}$$

$$Y_1 = \overline{\overline{B}\overline{C}\overline{D}} \cdot \overline{\overline{A}\overline{B}\overline{D}} \cdot \overline{\overline{A}\overline{C}\overline{D}} \cdot \overline{\overline{A}\overline{B}\overline{C}} \quad \text{...Ans.}$$

Now prepare K–map for output $Y_2$




∴ **Output equation for $Y_2$**

$$Y_2 = \underset{(1)}{\overline{A}\,\overline{B}\,C\,D} + \underset{(2)}{\overline{A}\,B\,\overline{C}\,D} + \underset{(3)}{\overline{A}\,B\,C\,\overline{D}} + \underset{(4)}{A\,B\,\overline{C}\,\overline{D}} + \underset{(5)}{A\,\overline{B}\,\overline{C}\,D} + \underset{(6)}{A\,\overline{B}\,C\,\overline{D}}$$

**Now using Boolean laws $\overline{\overline{A}} = A$ and Demorgan theorem $\overline{A + B} = \overline{A} \cdot \overline{B}$**

$$\therefore \quad Y_2 = \overline{\overline{A}\,\overline{B}\,CD + \overline{A}\,B\,\overline{C}\,D + \overline{A}\,B\,C\overline{D} + AB\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,D + A\,\overline{B}\,C\overline{D}}$$

Scanned by CamScanner

$$\therefore \quad Y_2 = \overline{\bar{A}\bar{B}CD} \cdot \overline{\bar{A}B\bar{C}D} \cdot \overline{\bar{A}BC\bar{D}} \cdot \overline{AB\bar{C}\bar{D}} \cdot \overline{A\bar{B}\bar{C}D} \cdot \overline{A\bar{B}C\bar{D}}$$

Implementation of $Y_1$ and $Y_2$ using NAND gates only :

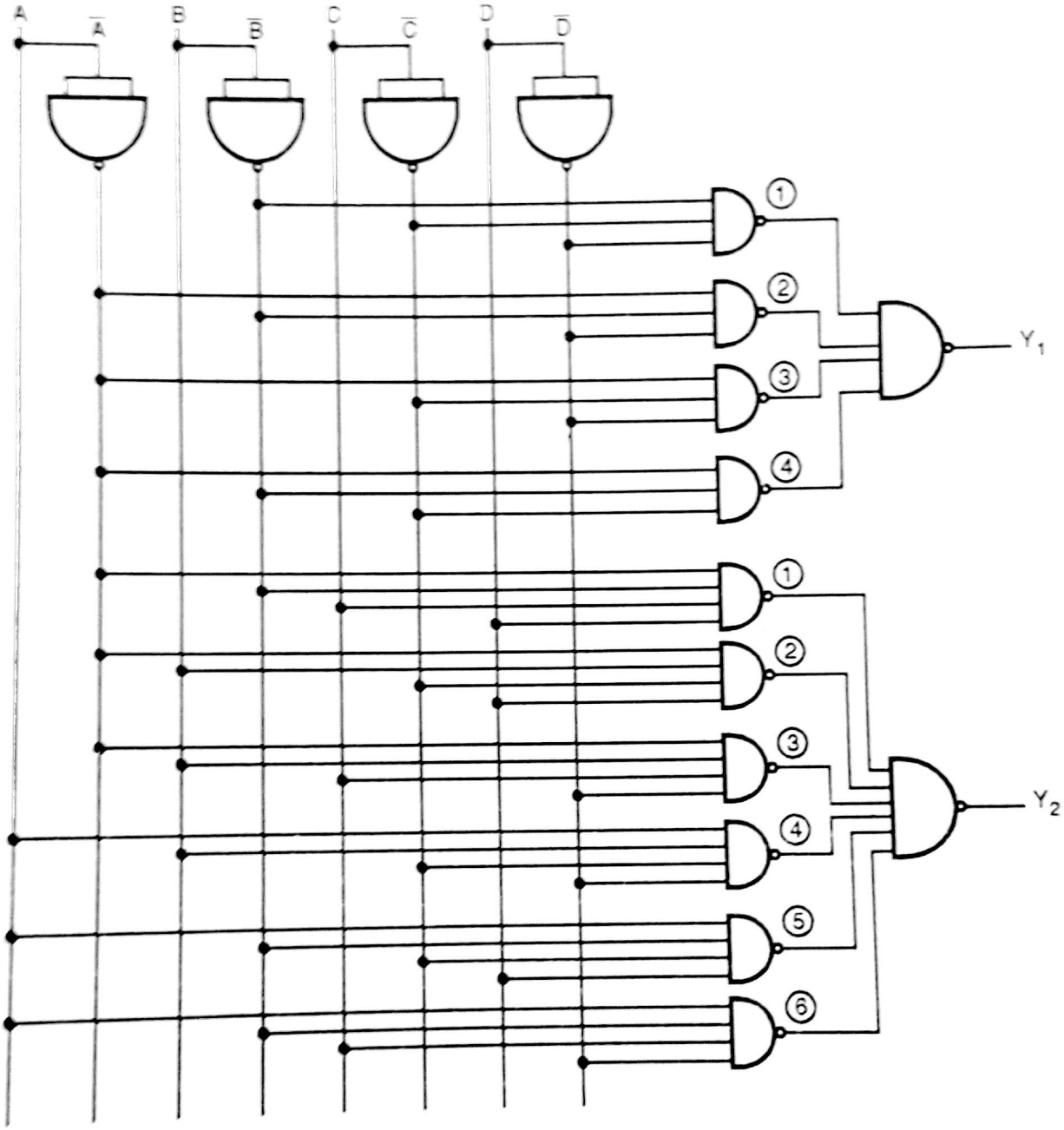



**Fig. Q. 5.6(a)**

Scanned by CamScanner

# Arithmetic Circuit

6

**Syllabus :**

Binary adders, subtractor, BCD adder, Binary comparator, Arithmetic and logic units.

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Adder | 6.2 | ✓ | ✓ |
| • BCD adder | 6.3 | ✓ | ✓ |
| • Excess-3 adder | 6.4 | ✓ | ✓ |
| • Binary subtractor | 6.5 | ✓ | ✓ |
| • BCD subtractor | 6.6 | ✓ | ✓ |
| • XS–3 subtractor | 6.7 | ✓ | ✓ |
| • Multiplier | 6.8 | ✓ | – |
| • Comparator | 6.9 | ✓ | ✓ |



**Comments**

- Full chapter is very important.
- Minimum 1 question worth of 10 to 12 marks, one can expect.

Scanned by CamScanner

## 6.1 Introduction :

In this chapter we are going to design arithmetic circuit based on our knowledge of boolean algebra, logic gates, K-map. Arithmetic circuits are basically designed to perform mathematical task such as addition, subtraction, multiplication etc. We want these circuits to perform complicated mathematical function as fast and efficiently as possible.

## 6.2 Adder :

Function of adder circuit is to add given number. We have different codes, adder changes depending upon the code. Basically we are going to learn presently binary adder.

### 6.2.1 Half Adder :

This is the basic building block for addition of two, **'single'** bit numbers. The truth Table is shown in Table 6.1 where $A_0$ and $B_0$ are the two one bit inputs and which produces sum and carry.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| $A_0$ | + | $B_0$ | Sum (S) | Carry (C) |
| 0 | | 0 | 0 | 0 |
| 0 | | 1 | 1 | 0 |
| 1 | | 0 | 1 | 0 |
| 1 | | 1 | 0 | 1 |

**Table 6.1**

$$\begin{array}{r} A_0 \\ +\ B_0 \\ \hline \boxed{\text{Carry (C)}}\quad \text{Sum (S)} \end{array}$$

This table we have already studied in ***Binary Arithmetic***. The K-map and the circuit is shown in Fig. 6.1.

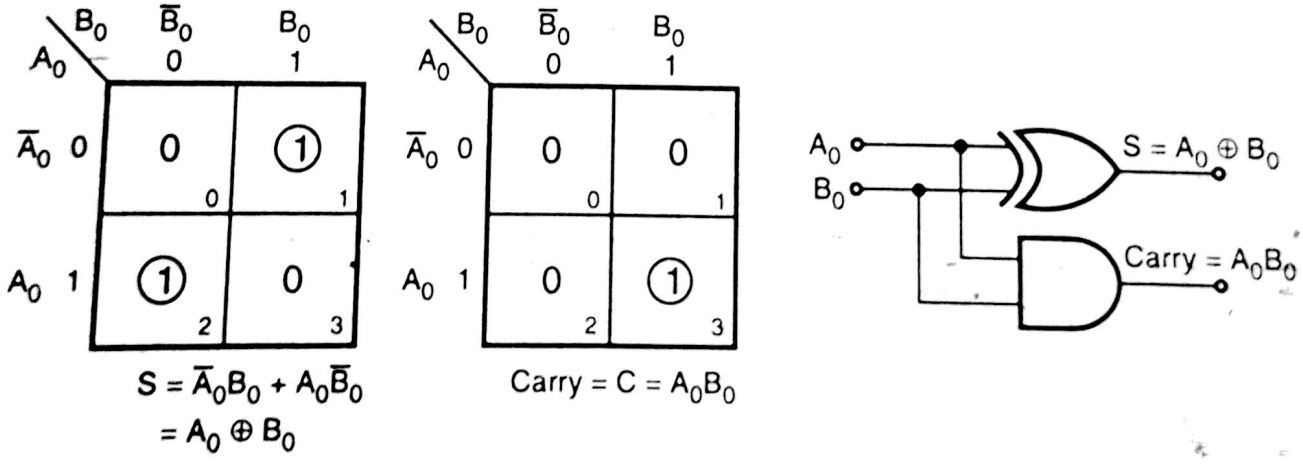



**Fig. 6.1**

### Drawback :

**Drawback of the circuit is in multibit addition, it doesn't cater to carry. Let's say we want to add $A_1$ $A_0$ and $B_1$ $B_0$.**

Scanned by CamScanner

$A_1$ $A_0$

\+

$B_1$ $B_0$

$C_1$ → carry generated from $A_0$, $B_0$ bit

$S_0$

Half adder will add $A_0$ and $B_0$, but in next stage we have to add three bits i.e. $A_1$, $B_1$ and $C_1$ which will not be done by the circuit, therefore we have to design 3 (three) single bit adder circuit which is named ***FULL ADDER***.

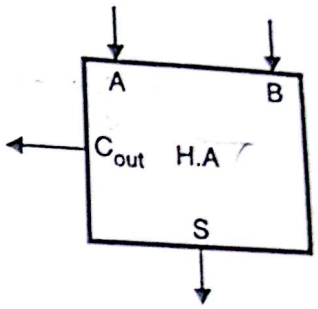



**Fig. 6.2 : Block schematic of half adder**

## 6.2.2 Full Adder :

To overcome the drawback of half adder, we will design three, single bit adder which will consider *Carry* also. The truth table is shown in Table 6.2.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| $A_n$ | $B_n$ | $C_n$ | $S_n$ | $C_{n+1}$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

**Table 6.2**

$$\begin{array}{r} A_n \\ + \; B_n \\ + \; C_n \\ \hline S_n \end{array}$$

$C_{n+1}$ ↑ Carry

Here $C_{n+1}$ represents carry for next stage . Solve K maps for $S_n$ and $C_{n+1}$ we get,

| $C_n$ \ $A_nB_n$ | $\overline{A}_n\overline{B}_n$ 00 | $\overline{A}_nB_n$ 01 | $A_nB_n$ 11 | $A_n\overline{B}_n$ 10 |
|---|---|---|---|---|
| $\overline{C}_n$ 0 | 0 (0) | 1 (2) | 0 (6) | 1 (4) |
| $C_n$ 1 | 1 (1) | 0 (3) | 1 (7) | 0 (5) |

$$S_n = \overline{C}_n(\overline{A}_nB_n + A_n\overline{B}_n) + C_n(\overline{A}_n\overline{B}_n + A_nB_n)$$
$$= \overline{C}_n(\overline{A}_nB_n + A_n\overline{B}_n) + C_n(\overline{\overline{A}_nB_n + A_n\overline{B}_n})$$
$$= C_n \oplus (\overline{A}_nB_n + A_n\overline{B}_n)$$
$$= C_n \oplus A_n \oplus B_n$$

| $C_n$ \ $A_nB_n$ | $\overline{A}_n\overline{B}_n$ 00 | $\overline{A}_nB_n$ 01 | $A_nB_n$ 11 | $A_n\overline{B}_n$ 10 |
|---|---|---|---|---|
| $\overline{C}_n$ 0 | 0 (0) | 0 (2) | 1 (6) | 0 (4) |
| $C_n$ 1 | 0 (1) | 1 (3) | 1 (7) | 1 (5) |

$$C_{n+1} = A_nB_n + B_nC_n + C_nA_n$$
$$= A_nB_n + B_nC_n + \overline{B}_nC_nA_n$$
$$= A_nB_n + \overline{A}_nB_nC_n + \overline{B}_nC_nA_n$$
$$= A_nB_n + C_n(\overline{A}_nB_n + \overline{B}_nC_n)$$
$$= A_nB_n + C_n(A_n \oplus B_n)$$

**Fig. 6.3**

Scanned by CamScanner

Circuit diagram is as shown in Fig. 6.4.

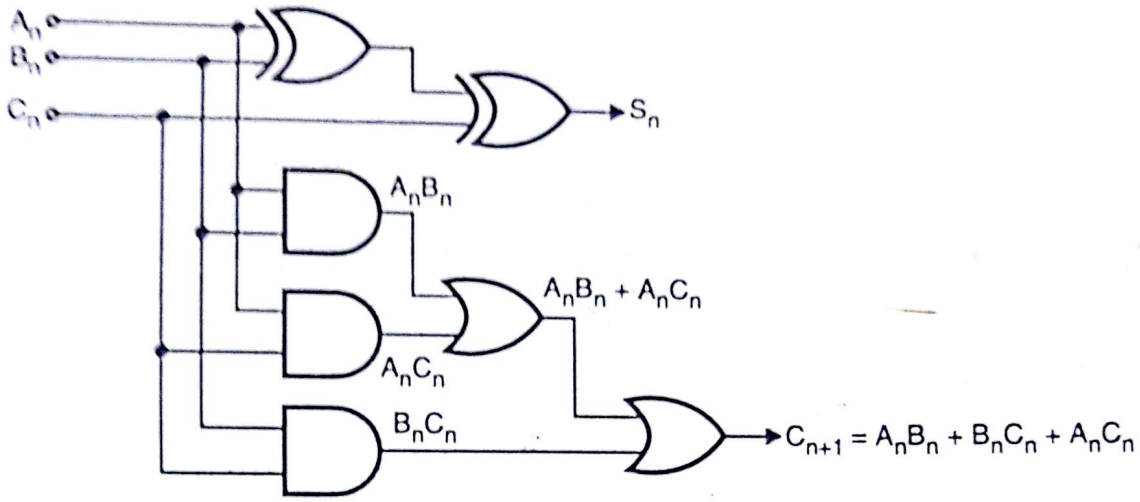



**Fig. 6.4**

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

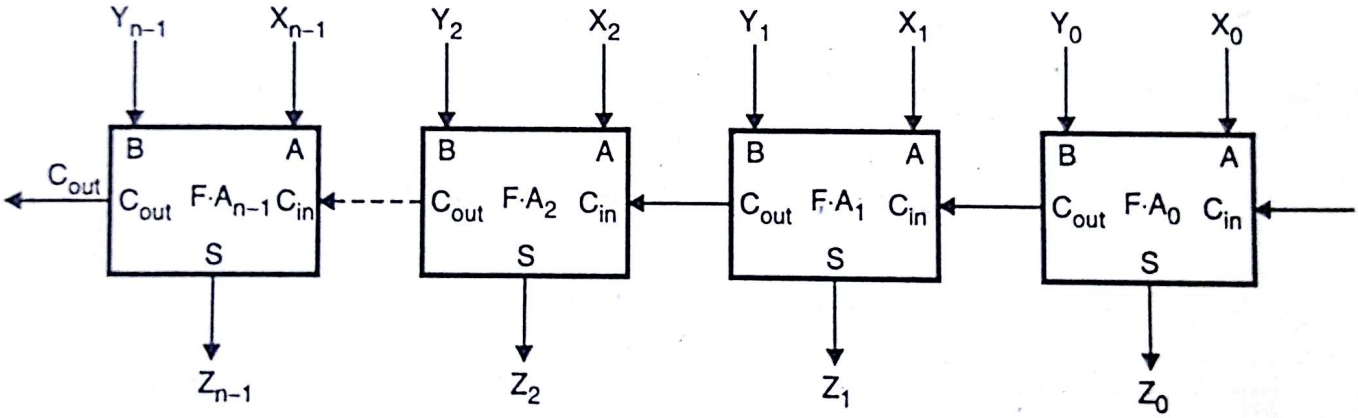



**Fig. 6.5**

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

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

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

A – B can be written as A + (– B)

Scanned by CamScanner

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

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

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

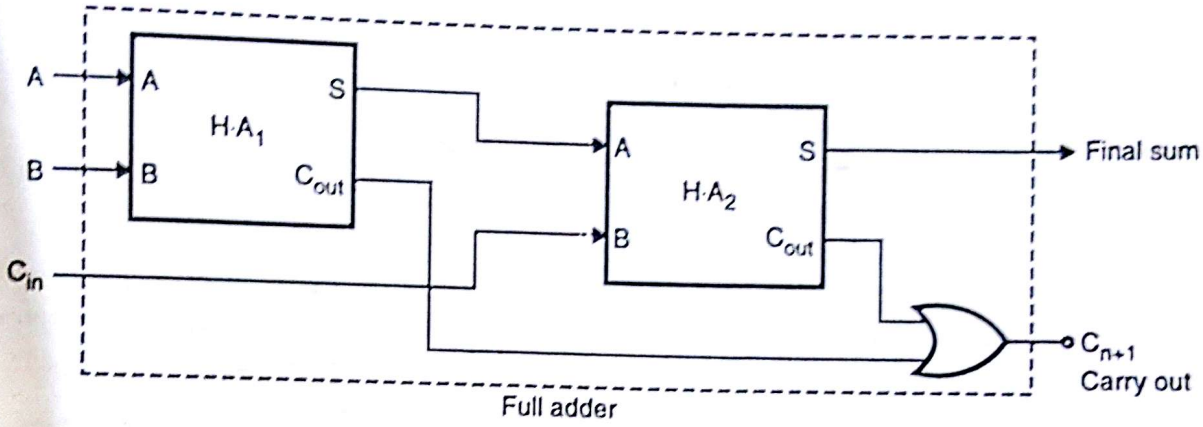



**Fig. 6.6**

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

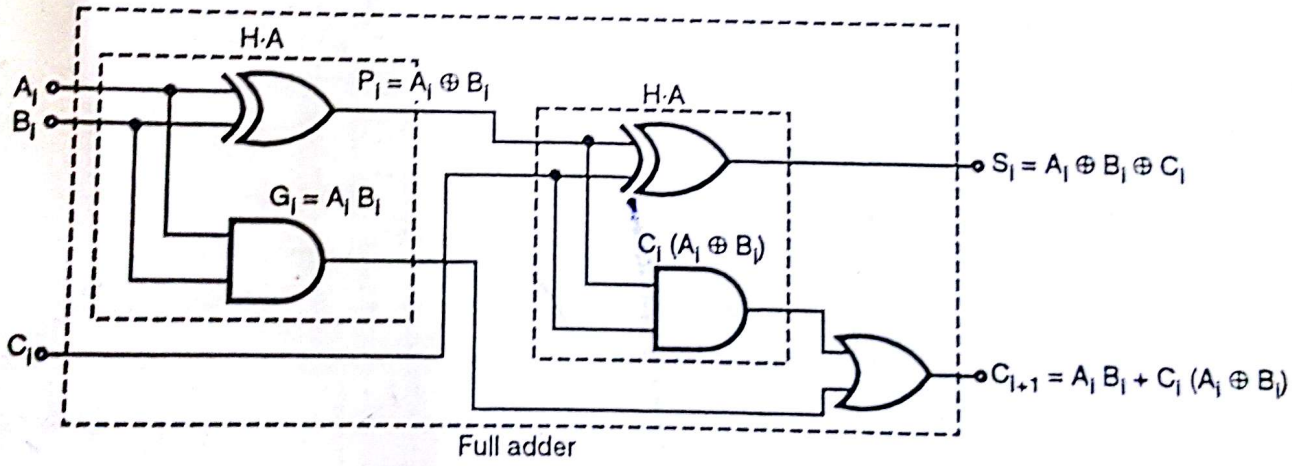



**Fig. 6.7**

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

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

This can be written as,

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

where, $G_i = A_i B_i$

and $P_i = A_i \oplus B_i$

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

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

Scanned by CamScanner

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

$\therefore \quad C_1 = G_o = A_0B_0$

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

i = 1 $\quad \therefore \quad C_2 = G_1 + P_1C_1$ ...(3)

i = 2 $\quad \therefore \quad C_3 = G_2 + P_2C_2$ Place Equation (3)

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

$\therefore \quad C_3 = G_2 + P_2G_1 + P_1P_2C_1$ ...(4)

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

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

$= G_3 + G_2P_3 + G_1P_2P_3 + P_1P_2P_3C_1$ ...(5)

i = 4 $\quad \therefore \quad C_5 = G_4 + P_4C_4$ Place Equation (3)

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

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

Now, draw circuit for $C_2$, $C_3$ and $C_4$

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

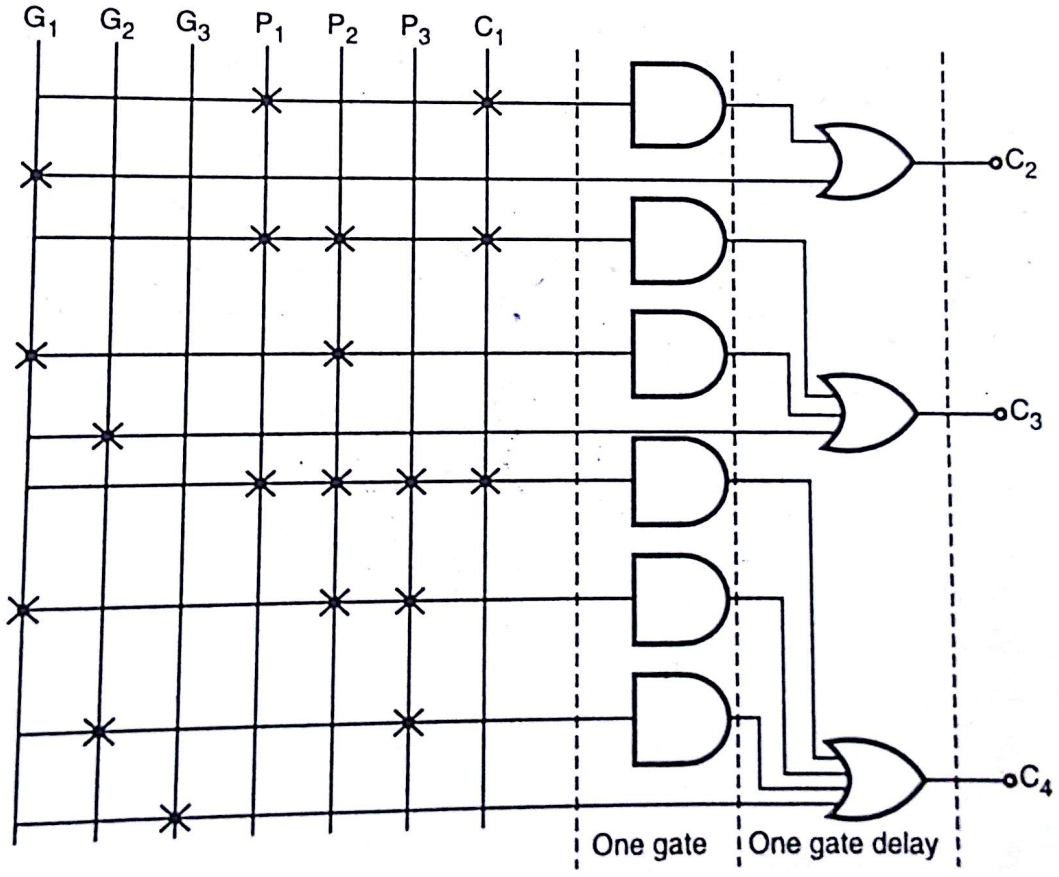



**Fig. 6.8**

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

$S_i = A_i \oplus B_i \oplus C_i$.

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

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

Scanned by CamScanner

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

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

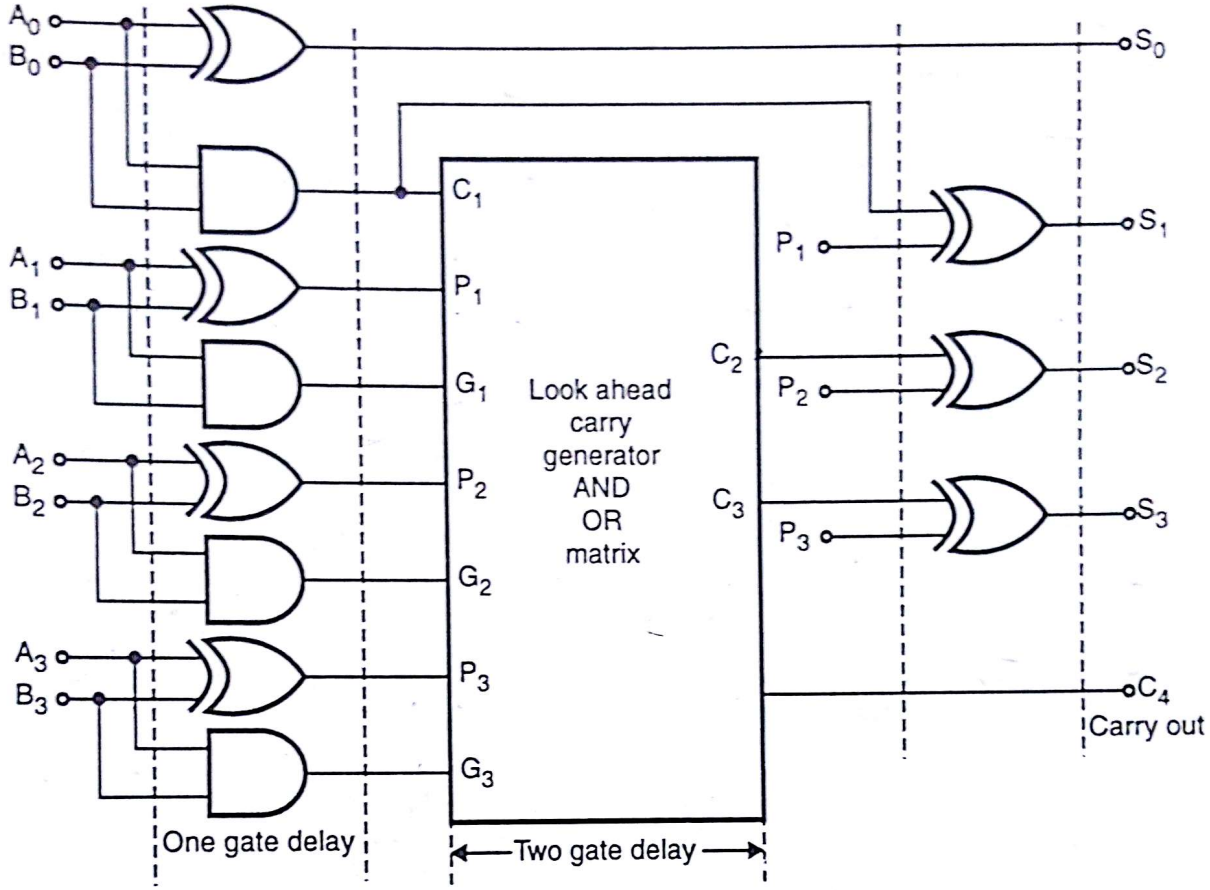



**Fig. 6.9**

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

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

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

(b) Output will be 4 bit.

(c) We have to cascade all the stage.

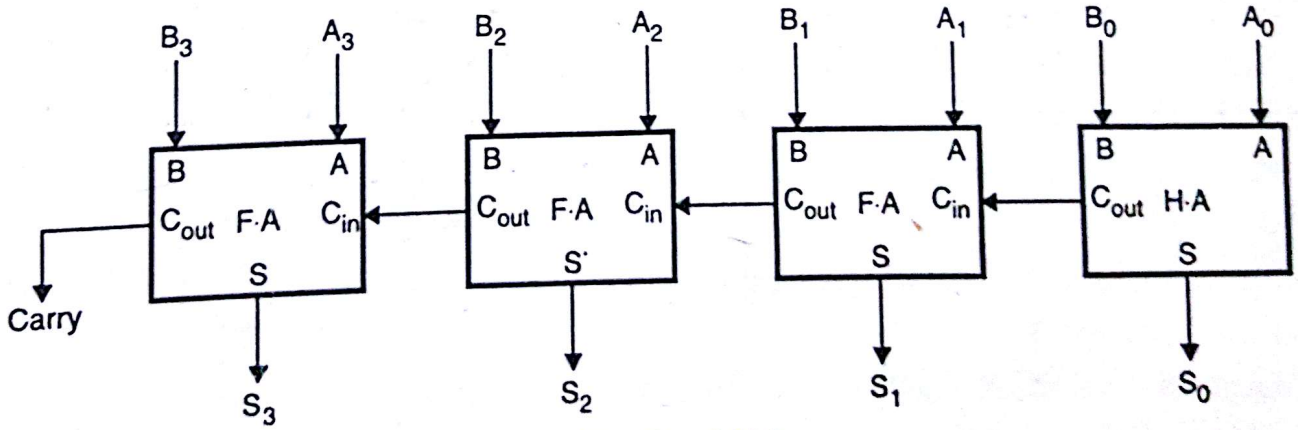



**Fig. Ex. 6.1(a)**

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

Scanned by CamScanner

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

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

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

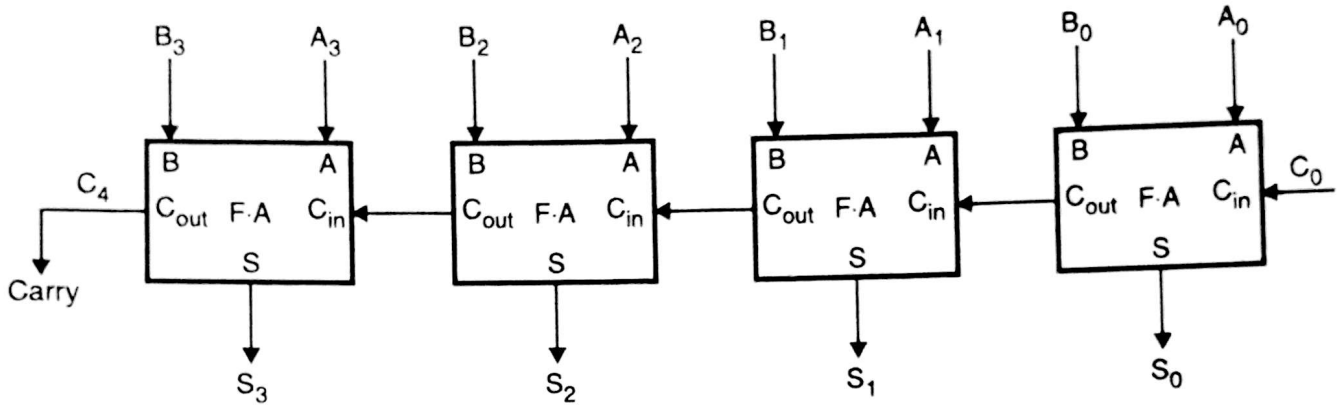



**Fig. Ex. 6.2(a)**

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

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

Final carry

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

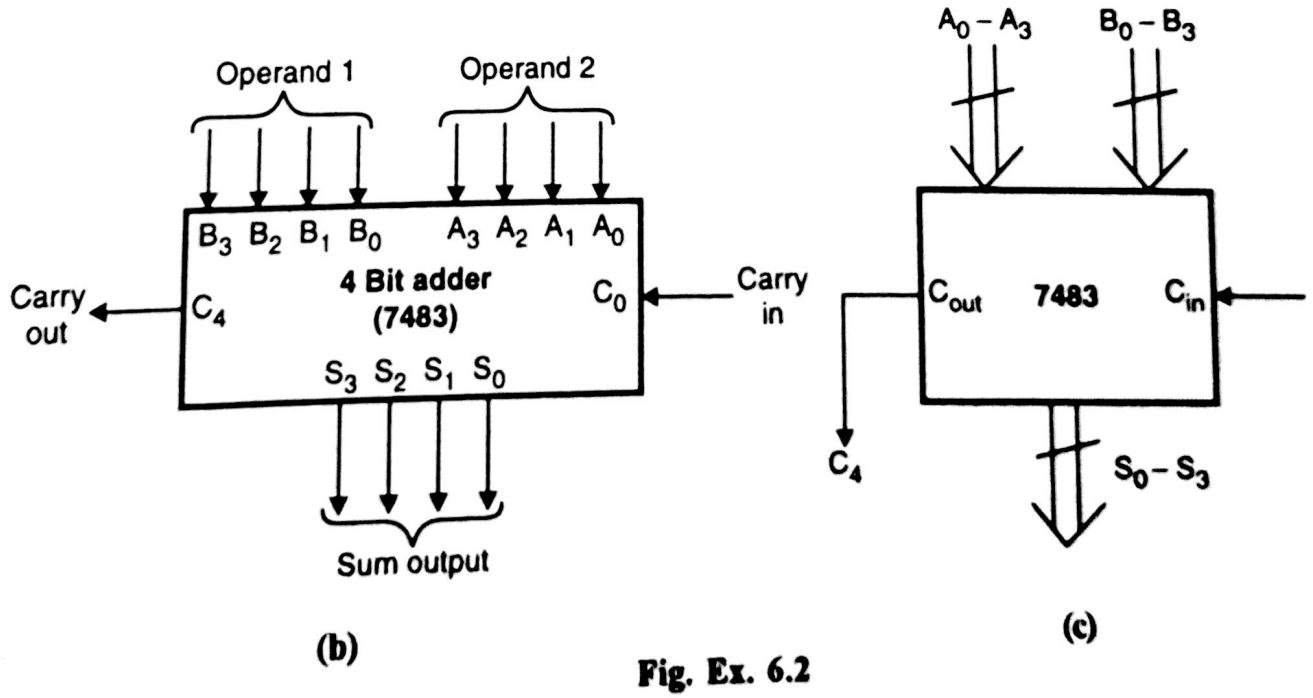



**Fig. Ex. 6.2**

Scanned by CamScanner

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

This circuit is also called as ***Binary Parallel Adder***

**Pin Description :**

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

**Table 1**

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

**Soln. :**

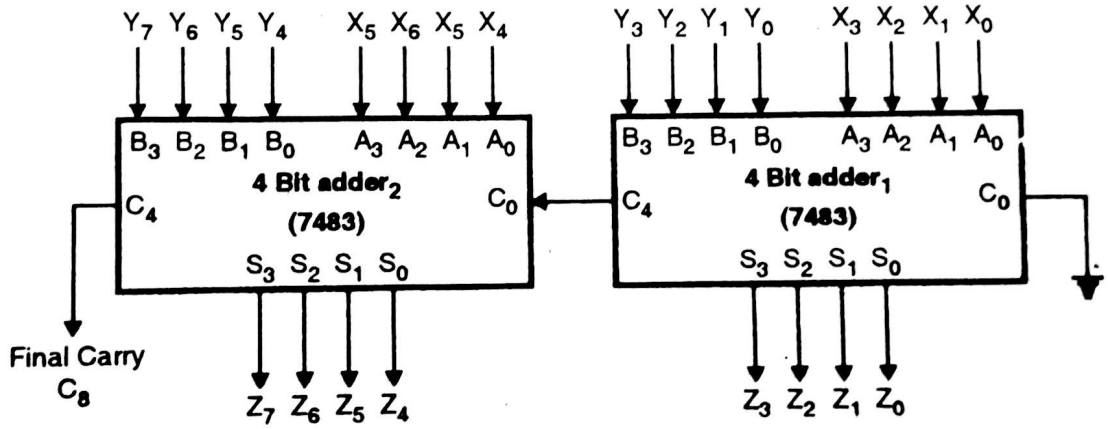



**Fig. Ex. 6.3(a)**

For 4 bit adder 1

(1) $A_3$ $A_2$ $A_1$ $A_0$ = $X_3$ $X_2$ $X_1$ $X_0$
$B_3$ $B_2$ $B_1$ $B_0$ = $Y_3$ $Y_2$ $Y_1$ $Y_0$
$S_3$ $S_2$ $S_1$ $S_0$ = $Z_3$ $Z_2$ $Z_1$ $Z_0$

$C_0 = 0$ (Because this stage is the starting stage and assumes NO CARRY GENERATED FROM PREVIOUS STAGE.)

$C_4 \Rightarrow$ Carry from 3rd bit ***propagated*** or ***Passed on to*** next stage.

(2) For 4 bit adder 2

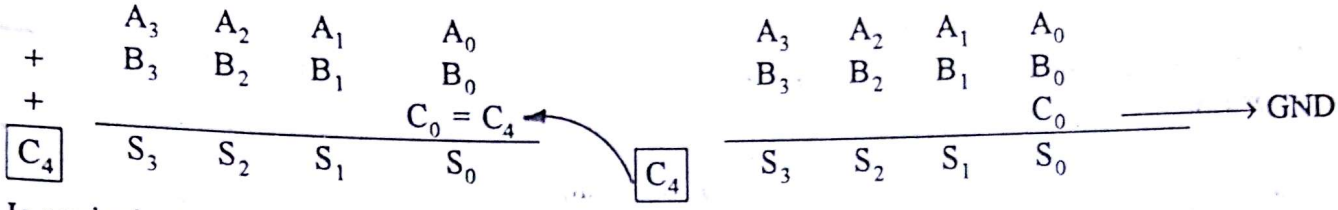
$A_3$ $A_2$ $A_1$ $A_0$ = $X_7$ $X_6$ $X_5$ $X_4$
$B_3$ $B_2$ $B_1$ $B_0$ = $Y_7$ $Y_6$ $Y_5$ $Y_4$
$S_3$ $S_2$ $S_1$ $S_0$ = $Z_7$ $Z_6$ $Z_5$ $Z_4$

**$C_0$ = Connected to output $C_4$ of 1st stage**

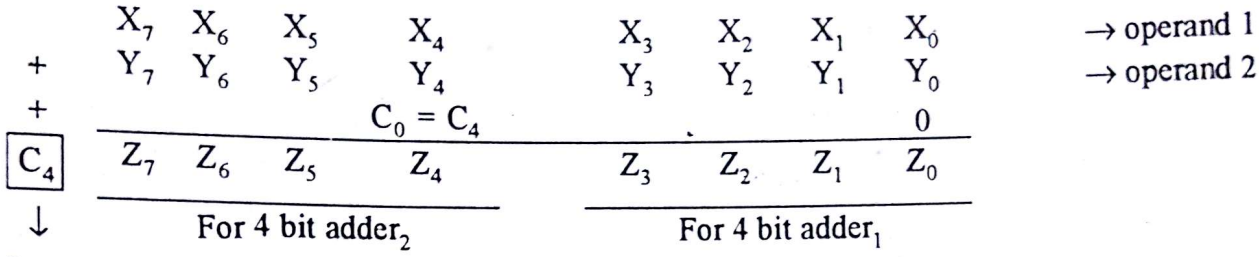
**$C_4 \Rightarrow$ final carry**

Scanned by CamScanner

Operation performed is,




Is equivalent to,




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

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

## 6.3 BCD Adder :

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

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

The above steps are performed to get VALID BCD Number

Reference for the same is Chapter 2.

Let's refresh the concept

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

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

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

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

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

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

Take one more example,

Add BCD 9 and 8

(9) ⇒ 1 0 0 1
(8) ⇒ 1 0 0 0
Carry 1 | 0 0 0 1
0 1 1 0
1 | 0 1 1 1
1 7

First level of addition

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

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

∴ Answer is 17 (valid answer)

Scanned by CamScanner

**Conclusion from above two examples :**

(1) We require digital circuit which senses that answer is > 9 i.e. invalid BCD. After sensing, it should add 6 to the answer.

(2) We have to also check 'Carry' generated because when carry generates answer is normally valid BCD. Therefore, circuit designed in point 1 will not help us to add 6, therefore sense 'Carry' and add six.

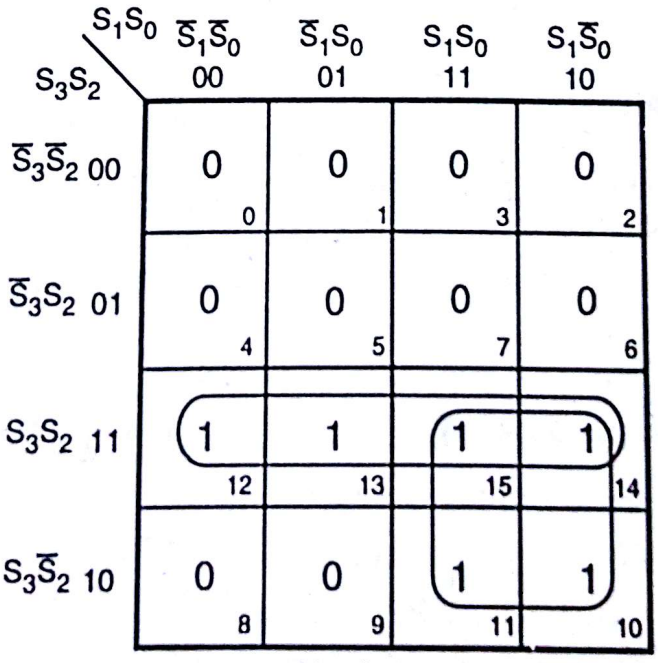
(3) To achieve correct BCD answer we require two level of adder circuit. First adder will add BCD inputs. Second will add 6 to the answer only and only if numebr is invalid BCD **OR** carry generated from FIRST STAGE.

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

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

**Table 6.3**

**K-Map :**




$\therefore Y = S_3S_2 + S_3S_1$

**Fig. 6.10**

Scanned by CamScanner

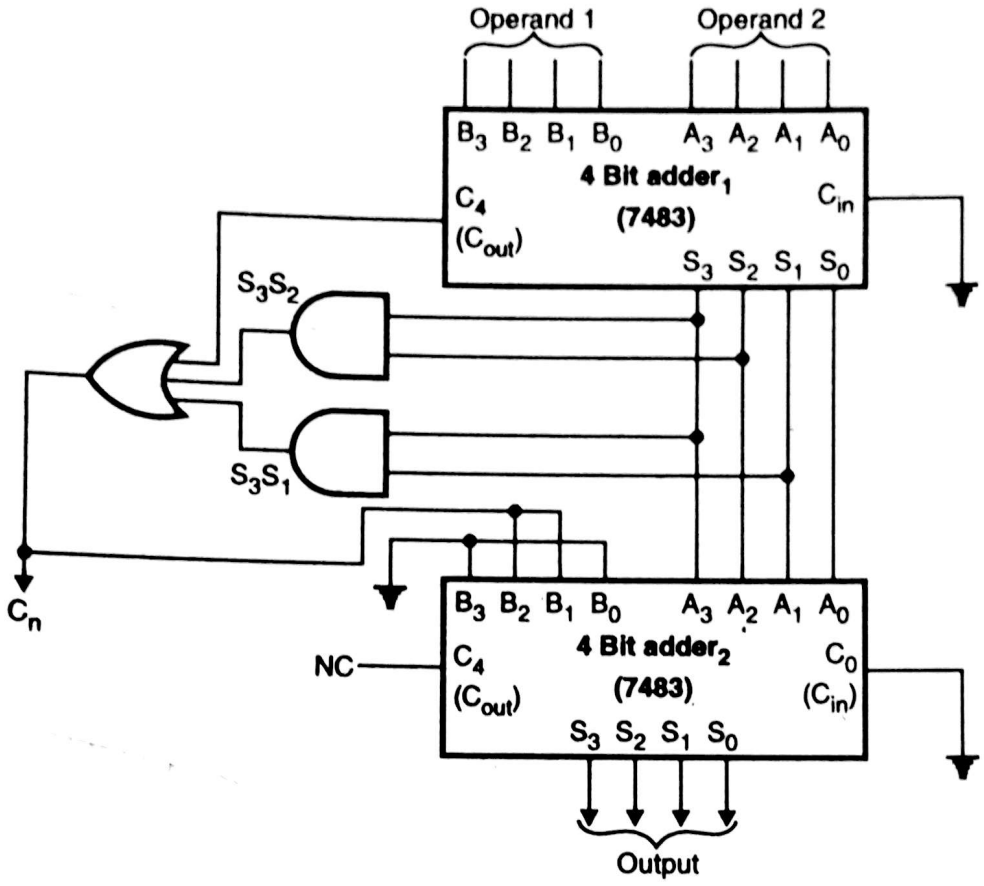



**Fig. 6.11**

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

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

Refer Fig. 6.11.

### 6.3.1 Packed BCD Addition :

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

**Consider example :**

(i)

| | | | | |
|---|---|---|---|---|
| | 19 | 0 0 0 1 | 1 0 0 1 | $[X_7 - X_0]$ |
| + | 16 | 0 0 0 1 | 0 1 1 0 | $[Y_7 - Y_0]$ |
| | 35 | 0 0 1 0 | 1 1 1 1 | |
| | | | +0 1 1 0 | |
| | | 0 0 1 1 | 0 1 0 1 | |
| | | 3 | 5 | |

| | | | |
|---|---|---|---|
| | 90 | 1 0 0 1 | 0 0 0 0 $[X_7 - X_0]$ |
| + | 91 | 1 0 0 1 | 0 0 0 1 $[Y_7 - Y_0]$ |
| | 181 | 1 | 0 0 1 0 0 0 0 1 |
| | | + | 0 1 1 0 0 0 0 0 |
| | | 1 | 1 0 0 0 0 0 0 1 |
| | | 1 | 8 1 |

Scanned by CamScanner

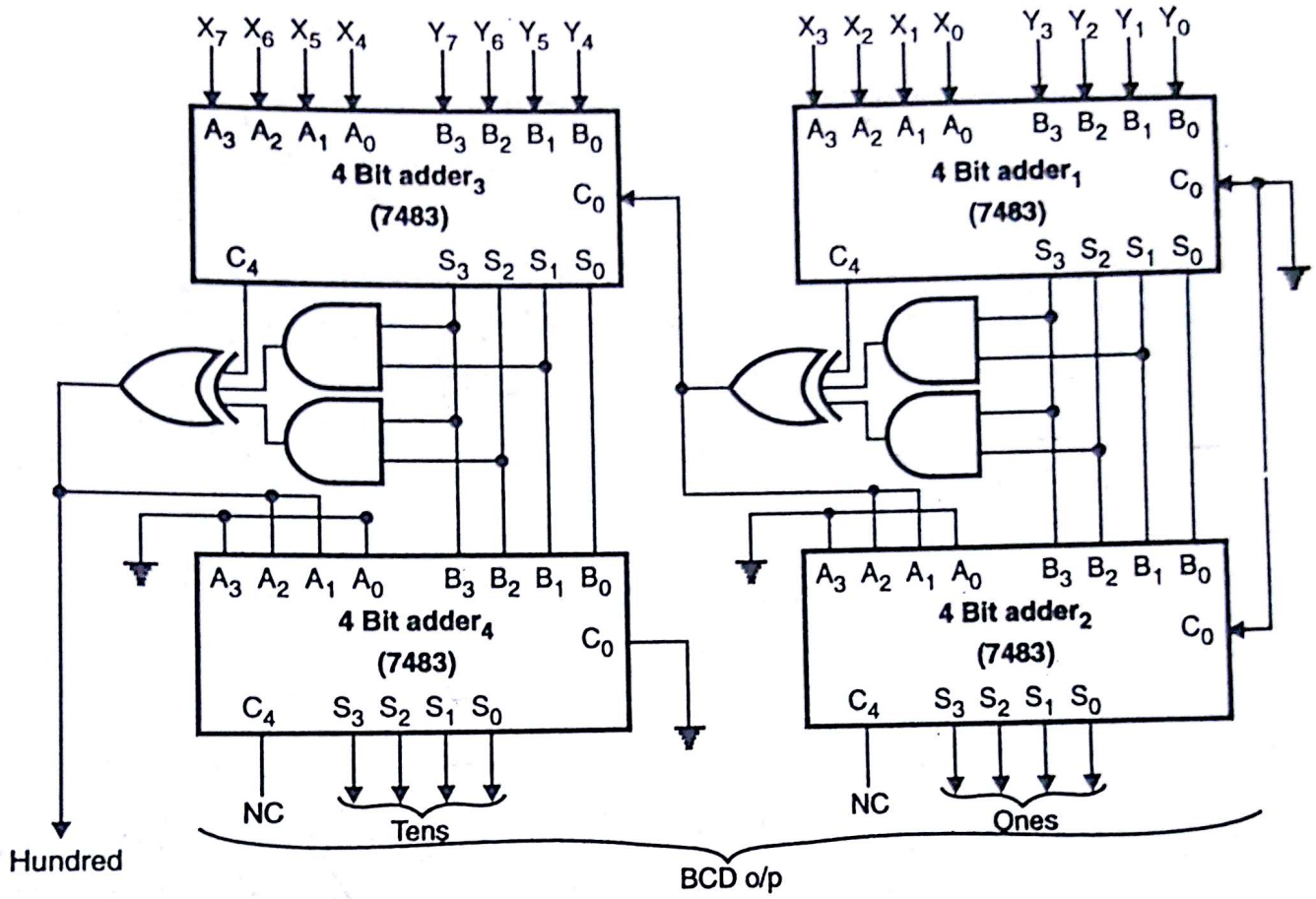



**Fig. 6.12**

## 6.4 Excess – 3 Adder :

**Basic for excess – 3 addition :**

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

(2) Now add both the XS–3 numbers.

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

else ⟶ subtract 3 from sum of two digits.

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

$\therefore \quad (3)_{10} = 0\ 0\ 1\ 1$

| | | |
|---|---|---|
| 1's complement | = | 1 1 0 0 |
| + 1 | | + 1 |
| 2's complement | | 1 1 0 1 |

$\therefore$ add (1101) to the answer and neglect carry.

Take an example, add 8 and 4 in XS–3.

$(8)_{10} \Rightarrow (1000)_2 \quad \therefore$ XS–3 = 1 0 0 0 + 0 0 1 1 ; = 1 0 1 1

$(4)_{10} \Rightarrow (0100)_2 \quad \therefore$ XS–3 = 0 1 0 0 + 0 0 1 1 = 0 1 1 1

Scanned by CamScanner

Now add both :

$$\begin{array}{llll} & 1 & 0 & 1 & 1 \longrightarrow \text{XS} - 3\ (8) \\ & 0 & 1 & 1 & 1 \longrightarrow \text{XS} - 3\ (4) \\ \hline \text{carry} \rightarrow \boxed{1} & 0 & 0 & 1 & 0 \quad \text{As carry is 1, add 3} \end{array}$$

$$\begin{array}{llll} + & & & + & \\ 0 & 0 & 1 & 1 & 0\ 0\ 1\ 1 \\ \hline 0 & 1 & 0 & 0 & 0\ 1\ 0\ 1 \\ \underbrace{\quad\quad}_{1} & & & \underbrace{\quad\quad}_{2} & \longrightarrow \text{in XS–3} \end{array}$$

Take one more example, add 4 and 3

4 in XS – 3 is 0111

3 in XS – 3 is 0110

$$\begin{array}{lll} \therefore\ 4 & 0\ 1\ 1\ 1 & \\ + \quad 3 & 0\ 1\ 1\ 0 & \\ \hline 7 \quad \boxed{0} & 1\ 1\ 0\ 1 & \text{As carry is not generated subtract 3. Subtract 3 is same as} \\ \quad + & 1\ 1\ 0\ 1 & \text{adding 1101.} \\ \hline \boxed{1} & \underbrace{1\ 0\ 1\ 0}_{7 \text{ in XS} - 3} & \end{array}$$

Neglect carry

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

| $B_3$ | $B_2$ | $B_1$ | $B_0$ | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | ⟶ | when carry = 1 |
| 1 | 1 | 0 | 1 | ⟶ | when carry = 0 |

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

$\therefore$ when Carry = 1 $B_1 = 1$ $B_2 = B_3 = 0$

Carry = 0 $B_1 = 0$ $B_2 = B_3 = 1$

follows carry complement of carry

Circuit is,

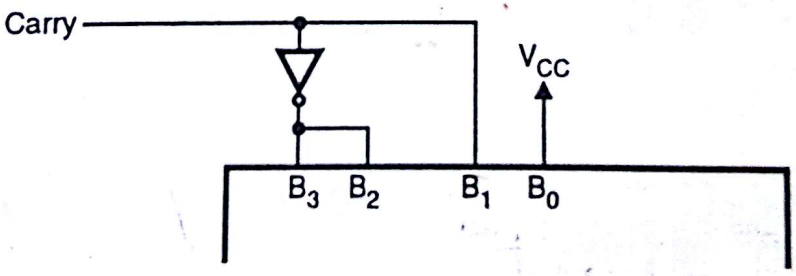



**Fig. 6.13 (a)**

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

Scanned by CamScanner

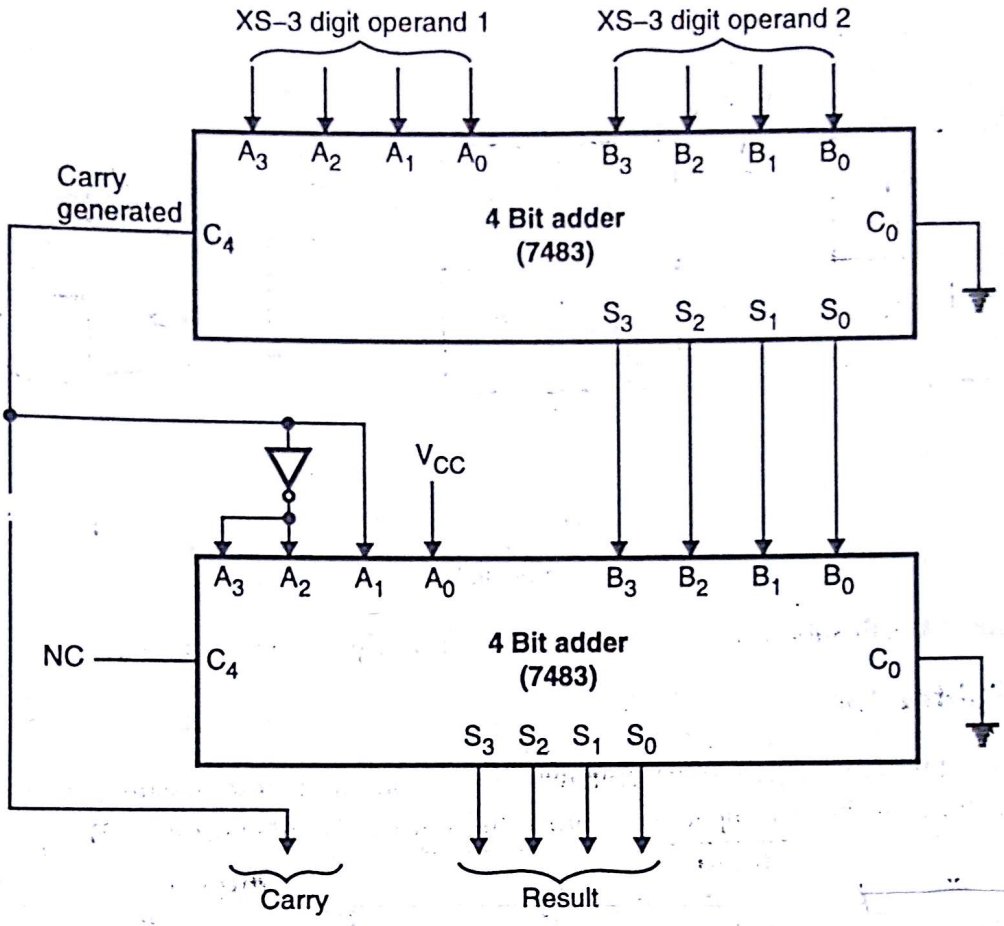



Fig. 6.13 (b)

## 6.5 Binary Subtractor :

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

### 6.5.1 Half Subtractor :

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

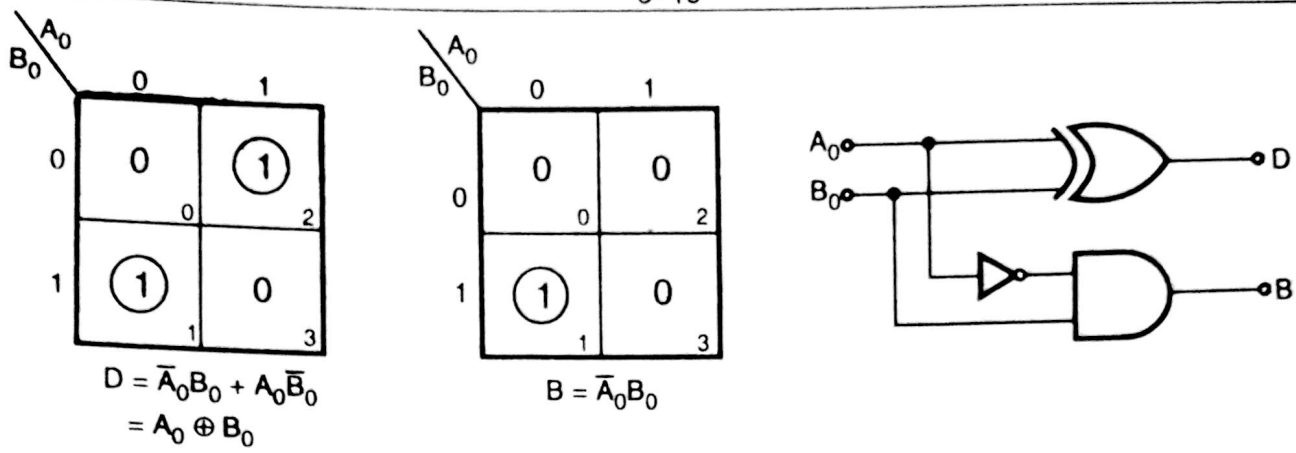
| Inputs | | Outputs | |
|---|---|---|---|
| $A_0$ | $B_0$ | Difference (D) | Borrow (B) |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

Table 6.4

The rules for subtraction has been already covered in chapter 2.

**K-Map :** The circuit is as shown in Fig. 6.14.

Scanned by CamScanner




(a) (b) (c)

**Fig. 6.14**

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

## 6.5.2 Full Subtractor :

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

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

**Table 6.5**

**K-Map :**

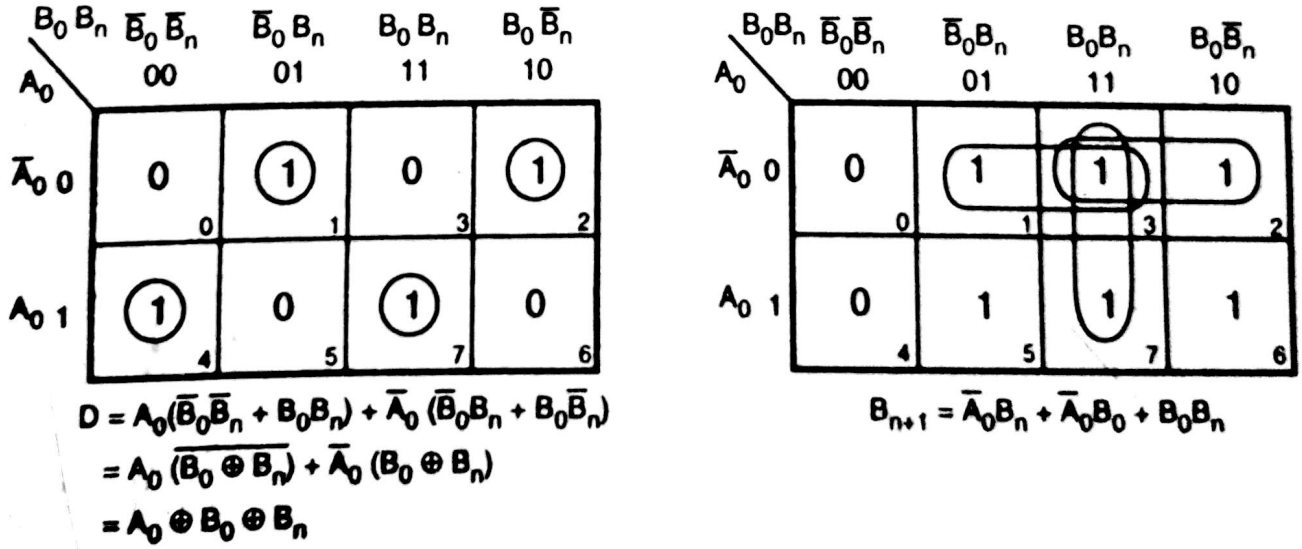



**Fig. 6.15 (a)**

Scanned by CamScanner

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

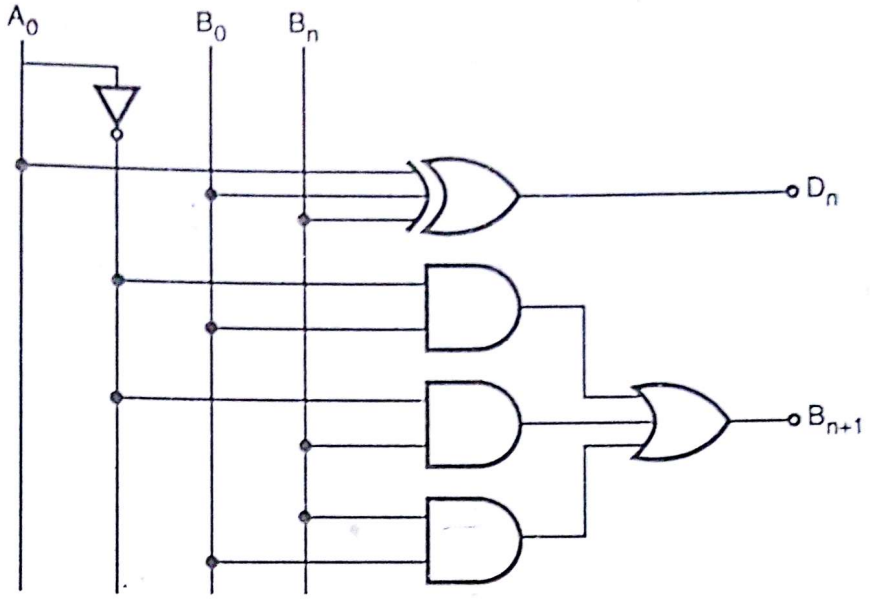



Fig. 6.15 (b)

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

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

### 6.5.3 Binary Parallel Subtractor using 1's Complement Method :

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

The circuit is as shown in Fig. 6.16.

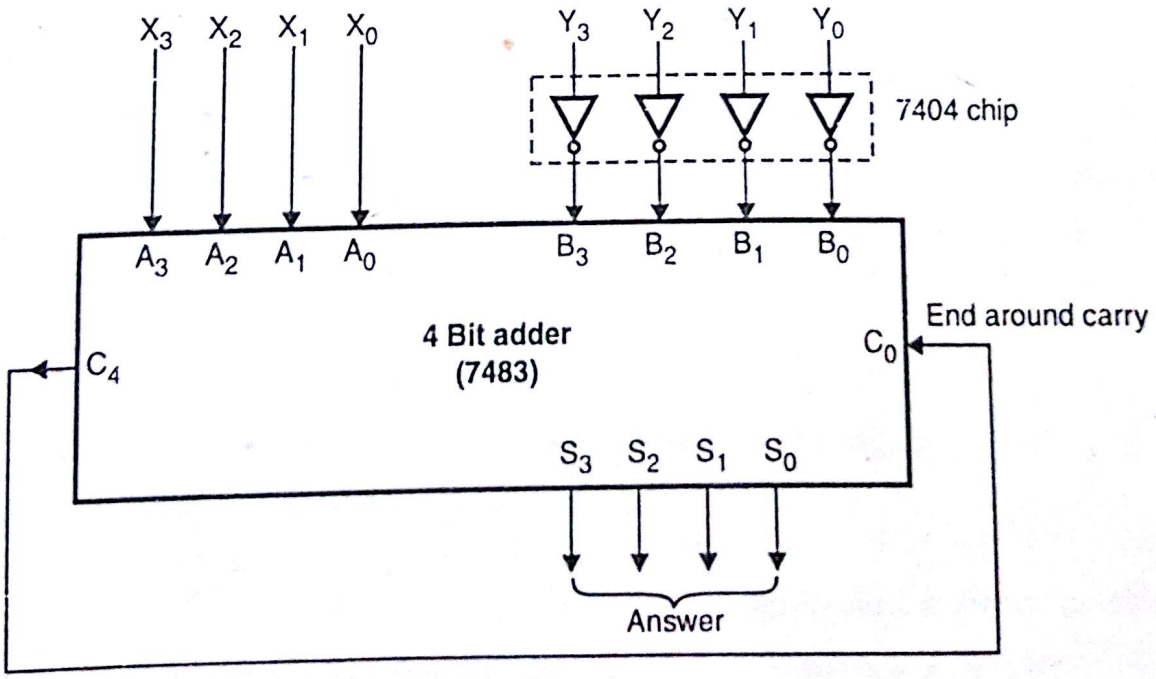



Fig. 6.16

Scanned by CamScanner

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

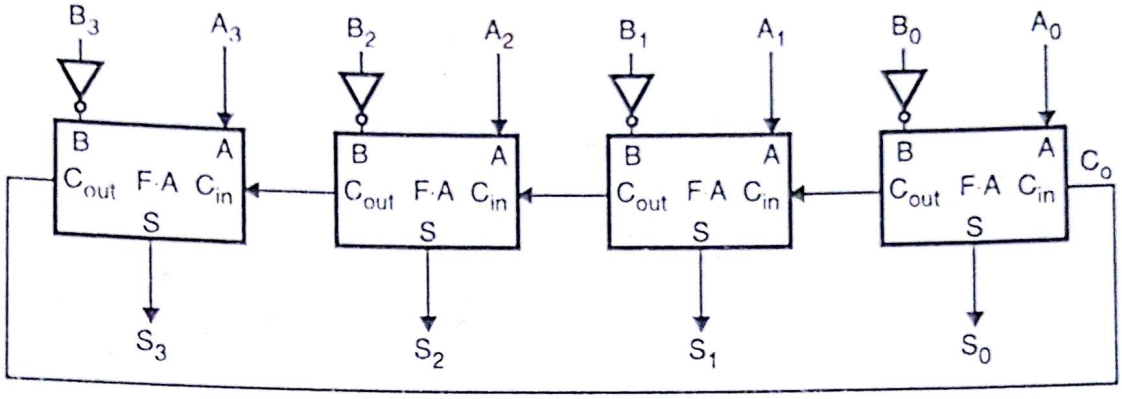



**Fig. 6.17**

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

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

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

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

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

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ∴ | $A_3$ | $A_2$ | $A_1$ | $A_0$ | | 0 | 1 | 0 | 0 |
| + | $B_3$ | $B_2$ | $B_1$ | $B_0$ | + | 1 | 1 | 0 | 1 |
| | | | | | 1 | 0 | 0 | 0 | 1 |
| | | | | | | | | | 1 |
| | | | | $C_4$ | | 0 | 0 | 1 | 0 |

$\longrightarrow S_3\ S_2\ S_1\ S_0$

Carry (Indicates answer in true form)

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

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

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

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

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $A_3$ | $A_2$ | $A_1$ | $A_0$ | | 0 | 0 | 1 | 0 |
| ∴ + | $B_3$ | $B_2$ | $B_1$ | $B_0$ | + | 1 | 0 | 1 | 1 |
| | | | | | 0 | 1 | 1 | 0 | 1 |
| | | | | | | | | | 0 |
| | | | | $C_4$ | | 1 | 1 | 0 | 1 |

$\longrightarrow S_3\ S_2\ S_1\ S_0$

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

So if we want in true form again invert it

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

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

Scanned by CamScanner

### 6.5.4 Binary Parallel Subtractor using 2's Complement Method :

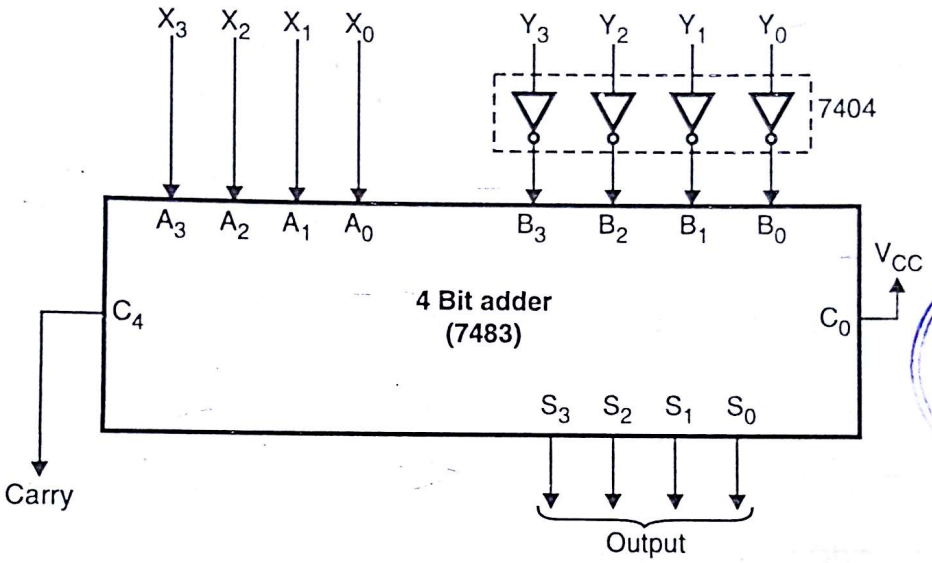
Normally in practical condition we use 2's complement method of subtraction :

2's complement = 1's complement + 1

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

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

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




COLLEGE OF CO... LIBRARY 743

**Fig. 6.18**

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

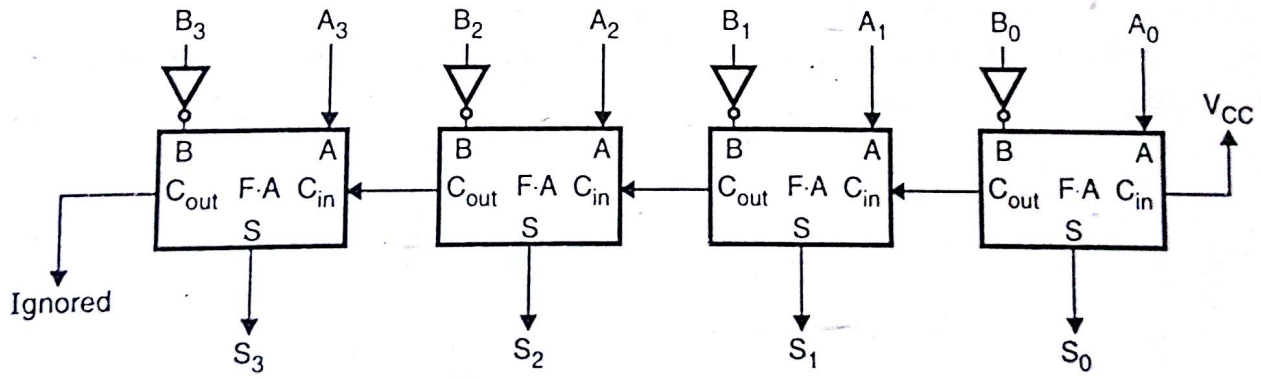



**Fig. 6.19**

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

**Working of circuit :**

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

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

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

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

Scanned by CamScanner

$\therefore$

$$\begin{array}{llcccc} & \overline{A_3} & \overline{A_2} & \overline{A_1} & \overline{A_0} & 0\ 1\ 0\ 0 \\ + & \overline{B_3} & \overline{B_2} & \overline{B_1} & \overline{B_0} & + \quad 1\ 1\ 0\ 1 \\ + & & & & C_0 & \quad\quad\ \ 1 \\ \hline & & & & & \boxed{1}\ \ 0\ 0\ 1\ 0 \end{array} \longrightarrow S_2\ S_2\ S_1\ S_0$$

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

carry $C_4$

Answer

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

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

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

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

$\therefore$ Operation :

$$\begin{array}{llcccc} & \overline{A_3} & \overline{A_2} & \overline{A_1} & \overline{A_0} & 0\ 0\ 1\ 0 \\ + & \overline{B_3} & \overline{B_2} & \overline{B_1} & \overline{B_0} & + \quad 1\ 0\ 1\ 1 \\ + & & & & C_0 & + \quad\quad\ \ 1 \\ \hline & & & & & \boxed{0}\ \ 1\ 1\ 1\ 0 \end{array} \longrightarrow S_3\ S_2\ S_1\ S_0$$

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

answer

$C_4$ (Carry)

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

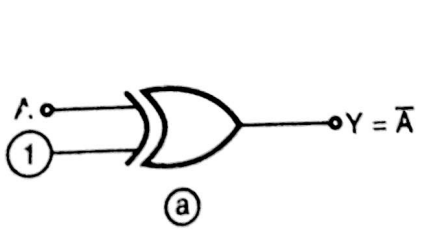
$\therefore$ $S_3\ S_2\ S_1\ S_0$ $\quad 1\ 1\ 1\ 0 \longrightarrow (-2)_{10}$

$$\begin{array}{lcccc} \text{1's complement} & 0 & 0 & 0 & 1 \\ + \quad\quad 1 & & & & 1 \\ \hline \text{2's complement} & 0 & 0 & 1 & 0 \end{array} \longrightarrow (2)_{10}$$

## 6.5.5 Controlled Addition / Subtraction :

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

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




(a)

| Inputs | | Output |
|---|---|---|
| A | 1 | Y |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

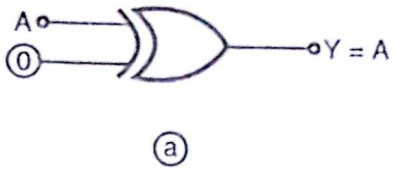
(b) Truth Table

**Fig. 6.20**

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

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

Scanned by CamScanner




(a)

| Inputs | | Output |
|---|---|---|
| A | 0 | Y |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

(b) Truth Table

Fig. 6.21

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

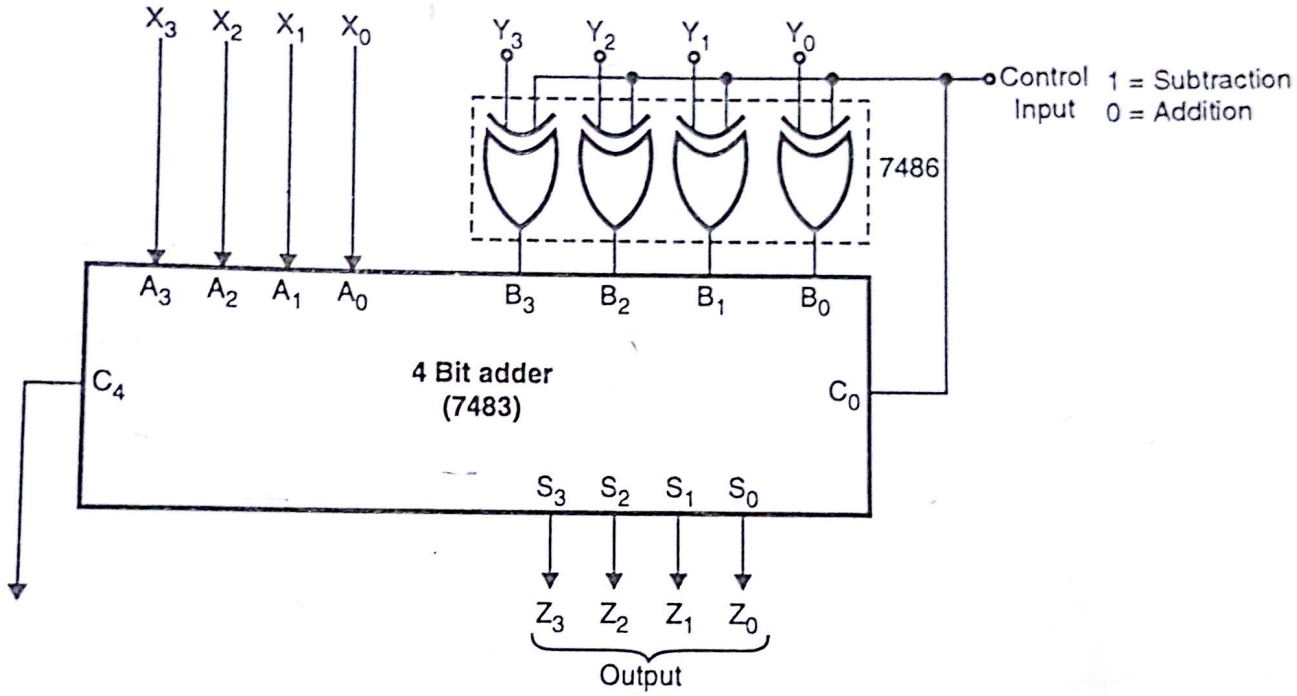



Fig. 6.22

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

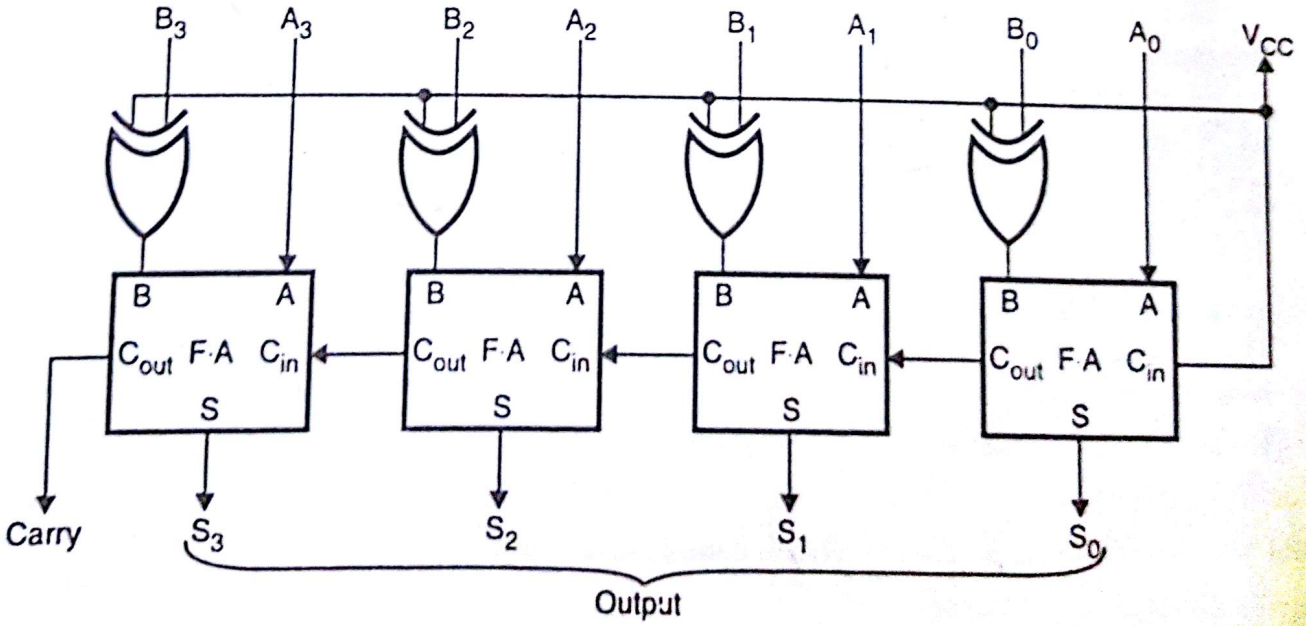



Fig. 6.23

Scanned by CamScanner

**Operation :**

**Case – I :** Control Input = 0

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

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

$\therefore$ 7483 performs,

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

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

**Case – II :** Control Input = 1

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

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

Here

$$\begin{array}{llllll} & B_3 & B_2 & B_1 & B_0 & \text{1's complement} \\ + & & & & C_0 & \equiv + \quad 1 \\ \hline & & & & & \text{2's complement} \end{array} \equiv + \begin{array}{l} \bar{Y}_3\ \bar{Y}_2\ \bar{Y}_1\ \bar{Y}_0 \\ C_0 \\ \hline \text{2's complement of given number } Y_3\ Y_2\ Y_1\ Y_0 \end{array}$$

Final operation can be represented mathematically,

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

## 6.6 BCD Subtractor :

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

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

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

$$\begin{array}{ll} \therefore (9)_{10} & 1\ 0\ 0\ 1 \\ - (2)_{10} & 0\ 0\ 1\ 0 \\ \hline & 0\ 1\ 1\ 1 \end{array} \longrightarrow \text{Nine's complement of } (2)_{10}$$

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

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

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

$$\begin{array}{ll} (10)_{10} & \longrightarrow 1\ 0\ 1\ 0 \\ + (-2)_{10} & \longrightarrow 1\ 1\ 0\ 1 \\ \hline \text{carry} \rightarrow & \boxed{1}\ 0\ 1\ 1\ 1 \end{array} \longrightarrow \text{Nine's complement of } (2)_{10}$$

(don't take carry into account)

Scanned by CamScanner

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

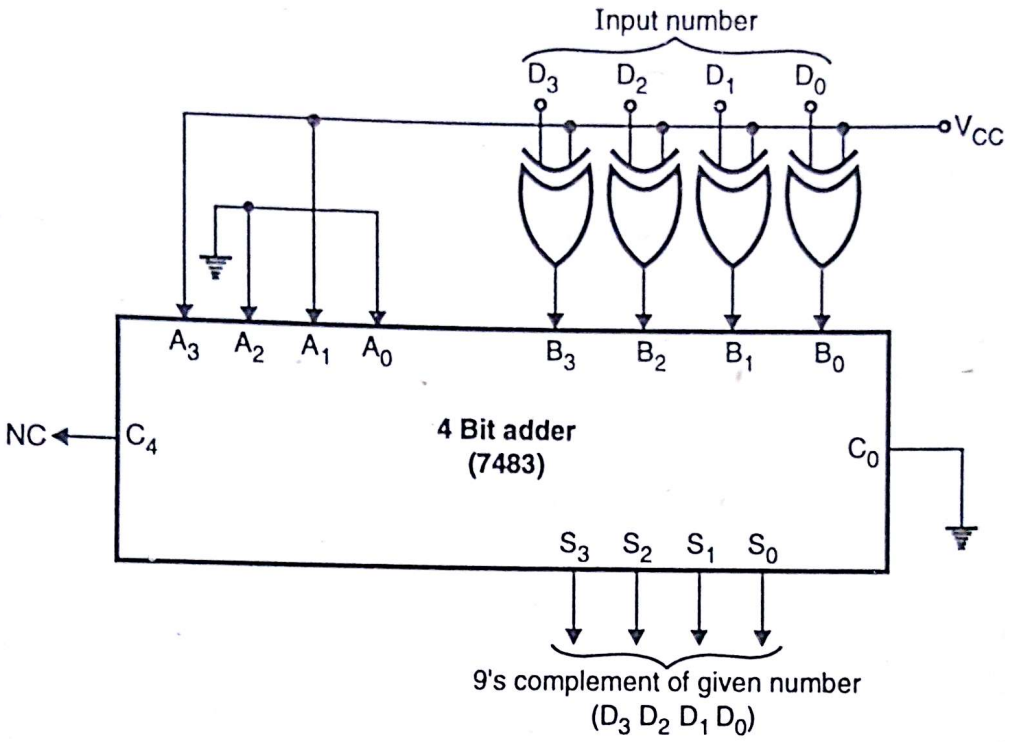



**Fig. 6.24**

### 6.6.1 Working of Circuit :

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

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

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

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

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

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

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

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

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

Let's study two examples for the same :

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

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

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

Scanned by CamScanner

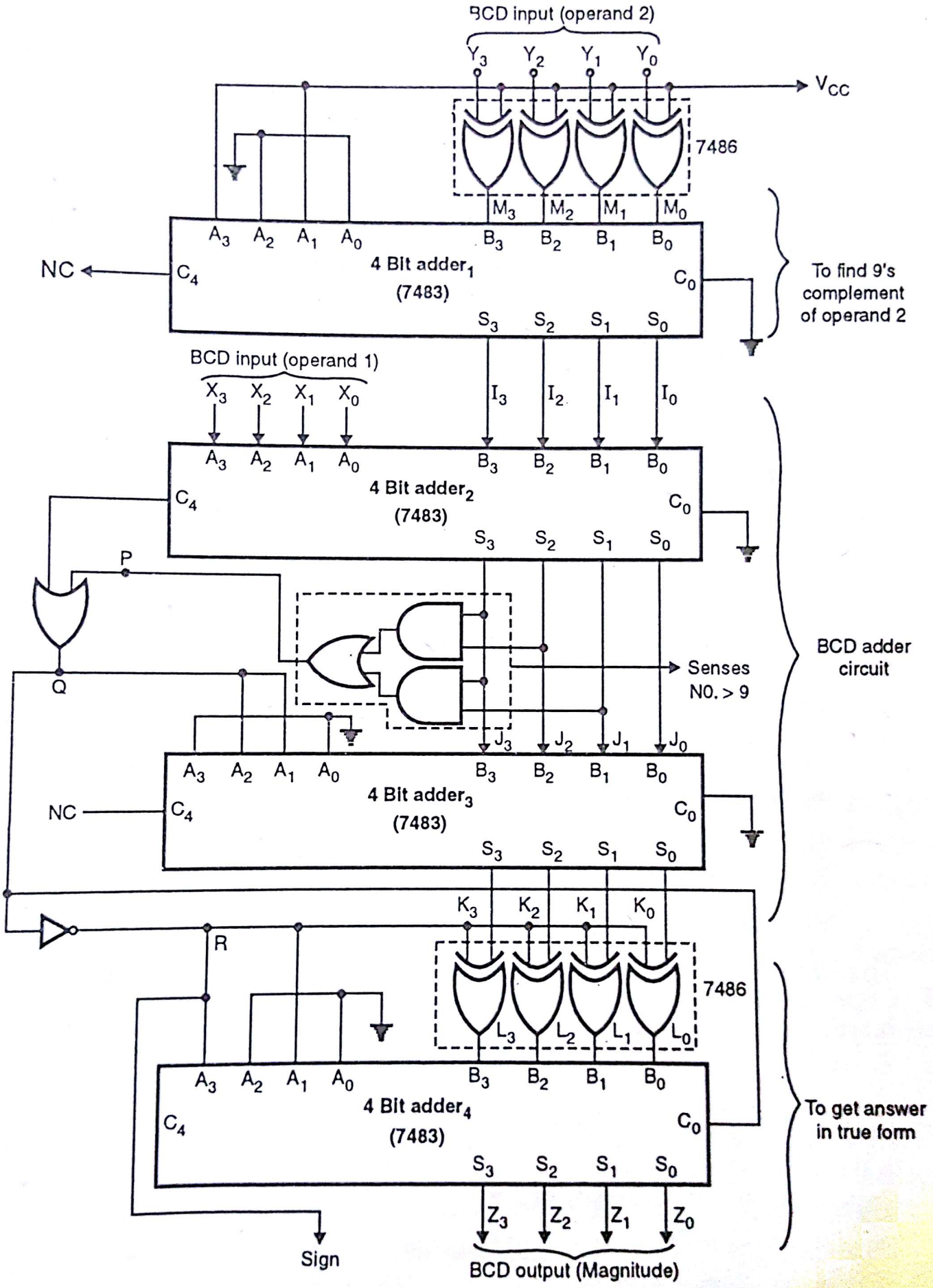



**Fig. 6.25**

Scanned by CamScanner

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

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

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

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

↑ carry ignored

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

**Operation performed by 4 bit adder$_2$ :**

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

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

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

↑ carry

**Operation performed by 4 bit adder$_3$ :**

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

$\therefore$ $7483_3$ will add $(6)_{10} = (0110)_2$ to $J_3\ J_2\ J_1\ J_0$.

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

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

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

↑ carry

**Operation performed by 4 bit adder$_4$ :**

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

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

Finally operation is :

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

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

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

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

and sign = 0

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

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

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

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

**Operation performed by 4 bit adder$_1$ :**

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

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

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

↑ ignored

Scanned by CamScanner

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

**Operation performed by 4 bit adder$_2$ :**

$$\begin{array}{llllllllllll} & A_3 & A_2 & A_1 & A_0 & \rightarrow & X_3 & X_2 & X_1 & X_0 & \rightarrow & 0\ 0\ 1\ 1 \\ + & B_3 & B_2 & B_1 & B_0 & \rightarrow & I_3 & I_2 & I_1 & I_0 & \rightarrow & 0\ 0\ 0\ 1 \\ \hline & S_3 & S_2 & S_1 & S_0 & & J_3 & J_2 & J_1 & J_0 & \boxed{0} & 0\ 1\ 0\ 0 \end{array}$$

$\uparrow$ carry

**Operation performed by 4 bit adder$_3$ :**

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

$$\therefore \quad \begin{array}{llllllllllll} & A_3 & A_2 & A_1 & A_0 & \longrightarrow & 0 & 0 & 0 & 0 & \longrightarrow & 0\ 0\ 0\ 0 \\ & B_3 & B_2 & B_1 & B_0 & \longrightarrow & J_3 & J_2 & J_1 & J_0 & \longrightarrow & 0\ 1\ 0\ 0 \\ \hline & & & & & & K_3 & K_2 & K_1 & K_0 & & 0\ 1\ 0\ 0 \end{array}$$

**Operation performed by 4 bit adder$_4$ :**

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

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

$$\begin{array}{llllllllllll} & A_3 & A_2 & A_1 & A_0 & \longrightarrow & 1 & 0 & 1 & 0 & \rightarrow & 1\ 0\ 1\ 0 \\ + & B_3 & B_2 & B_1 & B_0 & + & L_3 & L_2 & L_1 & L_0 & + & 1\ 0\ 1\ 1 \\ + & & & & C_0 & + & & & & C_0 & + & 0 \\ \hline & S_3 & S_2 & S_1 & S_0 & & Z_3 & Z_2 & Z_1 & Z_0 & \boxed{1} & 0\ 1\ 0\ 1 \rightarrow (5)_{10} \end{array}$$

ignore

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

So answer is – 5 as sign = 1

## 6.7 XS – 3 Subtractor :

Steps involved in XS – 3 subtraction are as follows :

**Step I** : Convert both number in their XS–3 format

**Step II** : Complement the subtrahend

**Step III** : Add complemented subtrahend to minuend

**Step IV** : After addition of these numbers

If carry generated then

(i) Result is positive

(ii) Add carry to the result therefore called ***end around carry***.

else

(i) Result is negative

(ii) add $(13)_{10}$

(iii) Take 1's complement of the result.

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

Scanned by CamScanner

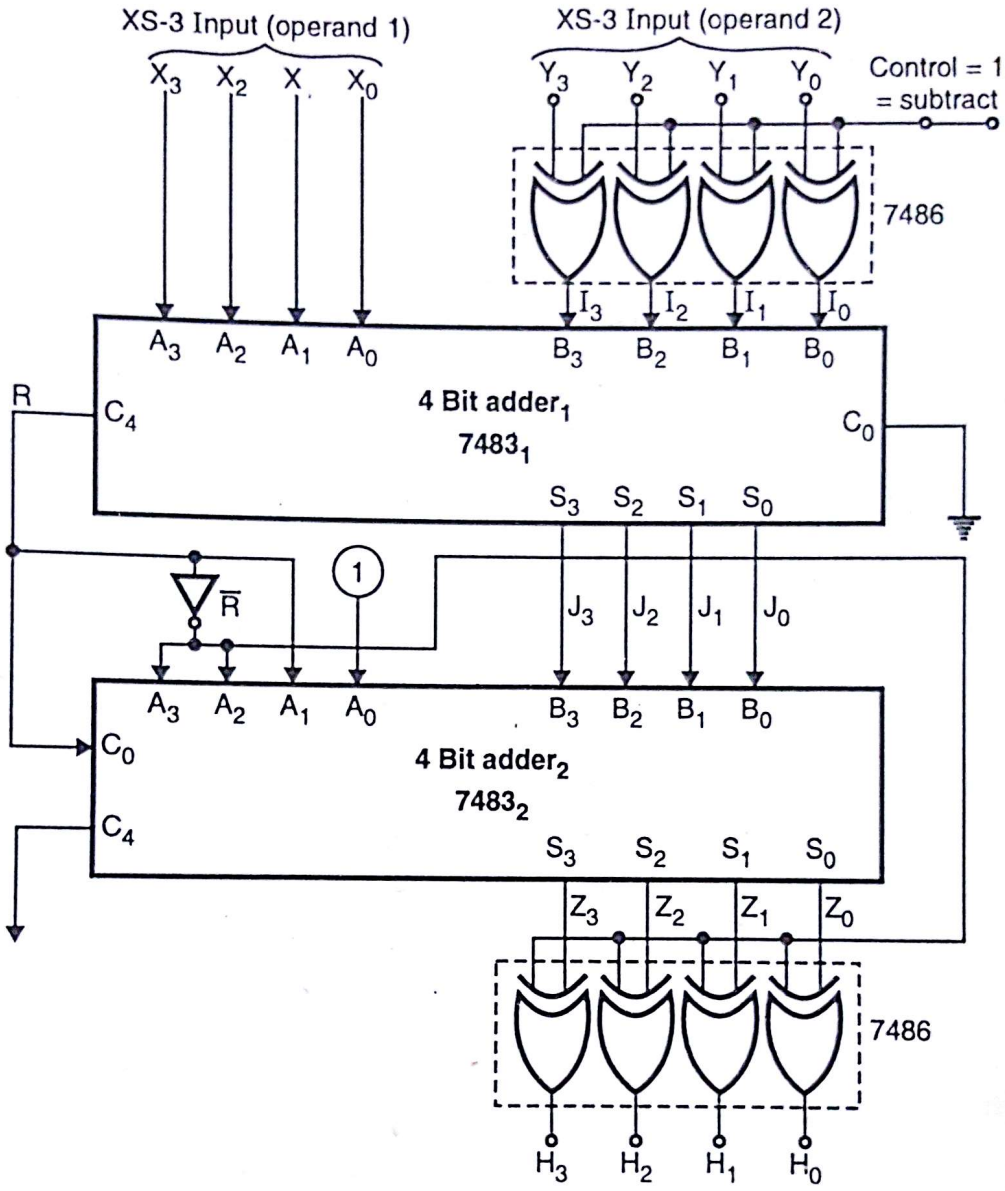



Fig. 6.26

**Operation of the circuit :**

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

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

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

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

**Operation performed by 4 bit adder$_1$ is :**

$\therefore$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $\rightarrow$ | 1 | 0 | 1 | 1 | $\rightarrow$ | | 1 | 0 | 1 | 1 |
| $B_3$ | $B_2$ | $B_1$ | $B_0$ | $\rightarrow$ | $I_3$ | $I_2$ | $I_1$ | $I_0$ | $\rightarrow$ | | 1 | 0 | 0 | 1 |
| $S_3$ | $S_2$ | $S_1$ | $S_0$ | | $J_3$ | $J_2$ | $J_1$ | $J_0$ | | [1] | 0 | 1 | 0 | 0 |
| | | | | | | | | | | $C_4$ | | | | |

Scanned by CamScanner

**Operation performed by 4 bit adder$_2$ is :**

$$\begin{array}{llllllllllllllll} & A_3 & A_2 & A_1 & A_0 & \rightarrow & 0 & 0 & 1 & 1 & \rightarrow & 0 & 0 & 1 & 1 \\ + & B_3 & B_2 & B_1 & B_0 & + & J_3 & J_2 & J_1 & J_0 & + & 0 & 1 & 0 & 0 \\ + & & & & C_0 & + & & & & 1 & + & 0 & 0 & 0 & 1 \\ \hline & S_3 & S_2 & S_1 & S_0 & & Z_3 & Z_2 & Z_1 & Z_0 & \boxed{0} & 1 & 0 & 0 & 0 \end{array} \rightarrow Z_3\ Z_2\ Z_1\ Z_0$$

Ignore

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

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

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

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

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

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

**Operation performed by 4 bit adder$_1$ :**

$$\therefore \begin{array}{llllllllllllllll} & A_3 & A_2 & A_1 & A_0 & \rightarrow & X_3 & X_2 & X_1 & X_0 & \rightarrow & 0 & 1 & 1 & 0 \\ & B_3 & B_2 & B_1 & B_0 & \rightarrow & I_3 & I_2 & I_1 & I_0 & \rightarrow & 0 & 1 & 0 & 0 \\ \hline & S_3 & S_2 & S_1 & S_0 & & J_3 & J_2 & J_1 & J_0 & \boxed{0} & 1 & 0 & 1 & 0 \end{array}$$

$\uparrow$ $C_4$ of 4 bit adder

As $C_4 = 0$, R = 0 $\therefore \bar{R} = 1$

**Operation performed by 4 bit adder$_2$ :**

$$\begin{array}{llllllllllllllll} & A_3 & A_2 & A_1 & A_0 & \rightarrow & 0 & 0 & 1 & 1 & \rightarrow & 1 & 1 & 0 & 1 \\ + & B_3 & B_2 & B_1 & B_0 & + & J_3 & J_2 & J_1 & J_0 & + & 1 & 0 & 1 & 0 \\ + & & & & C_0 & + & & & & R & + & & & & 0 \\ \hline & S_3 & S_2 & S_1 & S_0 & & Z_3 & Z_2 & Z_1 & Z_0 & \boxed{1} & 0 & 1 & 1 & 1 \end{array}$$

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

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

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

So R will tell you about positive or negative sign.

---

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

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

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

(iii) $PQ\bar{R}$

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

Scanned by CamScanner

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

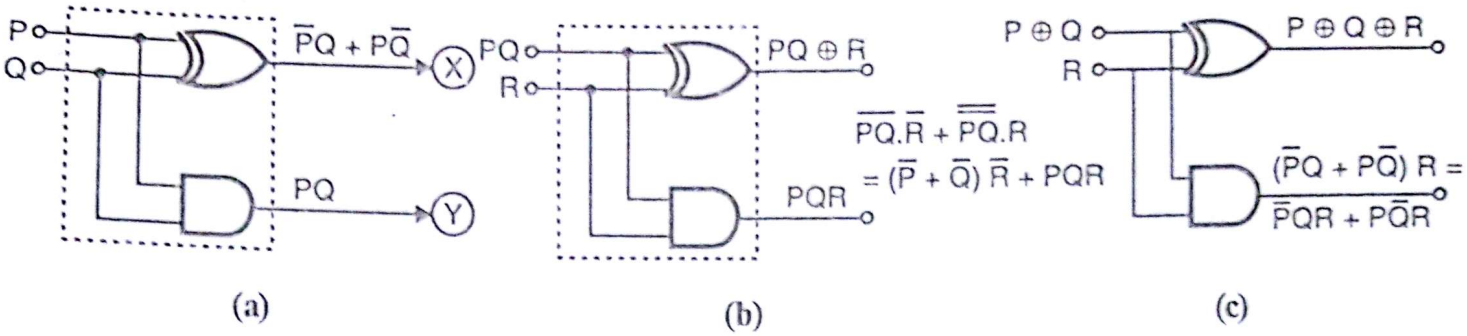



Fig. Ex. 6.4

---

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

**Soln. :** For a full adder we have,

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

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

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

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

---

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

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

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

**Table 2**

Scanned by CamScanner

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

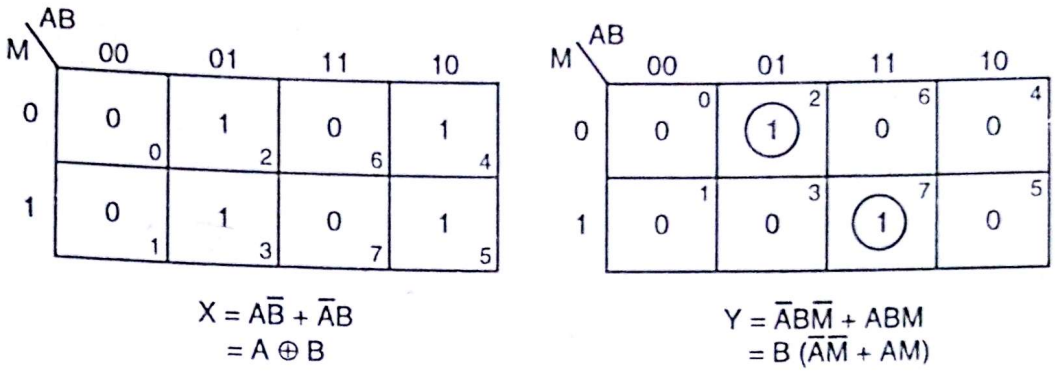



**Fig. Ex. 6.6(a)**

Circuit : Refer Fig. Ex. 6.6(b)

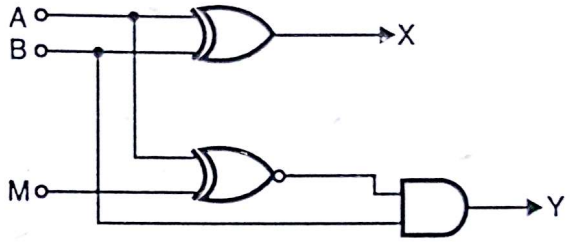



**Fig. Ex. 6.6(b)**

---

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

**Soln. :** We know that,

| For half subtractor | For full subtractor |
|---|---|
| $D = A_0 \oplus B_0$ | $D_n = A_0 \oplus B_0 \oplus B_n$ |
| $B = \bar{A}_0 B_0$ | $B_{n+1} = \bar{A}_0 B_n + \bar{A}_0 B_0 + B_0 B_n$ |

Let's say $A_0 = X$

$B_0 = Y$

$\therefore$ For half subtractor

| | |
|---|---|
| $D = x \oplus y$ | $D_n = x + y + B_n$ |
| $B = \bar{x}\, y$ | $B_{n+1} = \bar{x}\, B_n + \bar{x}\, y + y\, B_n$ |

The circuit is as shown in Fig. Ex. 6.7.

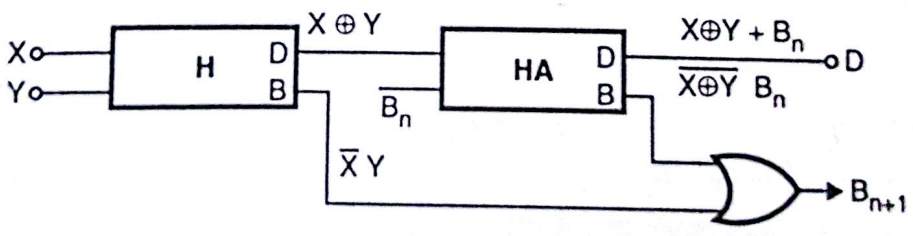



**Fig. Ex. 6.7**

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

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

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

Scanned by CamScanner

$$= \bar{x}\,(y + \bar{y}\,B_n) + x\,y\,B_n \qquad \because A + \bar{A}B = A + B$$
$$= \bar{x}\,(y + B_n) + xy\,B_n$$
$$= \bar{x}y + \bar{x}B_n + xyB_n$$
$$= y\,(\bar{x} + xB_n) + \bar{x}B_n$$
$$= y\,(\bar{x} + B_n) + \bar{x}\,B_n \qquad \because \bar{A} + AB = \bar{A} + B$$
$$= \bar{x}\,y + y\,B_n + \bar{x}\,B_n$$

## 6.8 Multiplier :

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

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

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

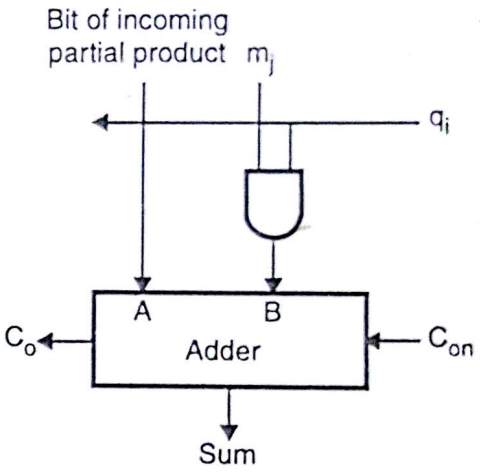



**Fig. 6.27**

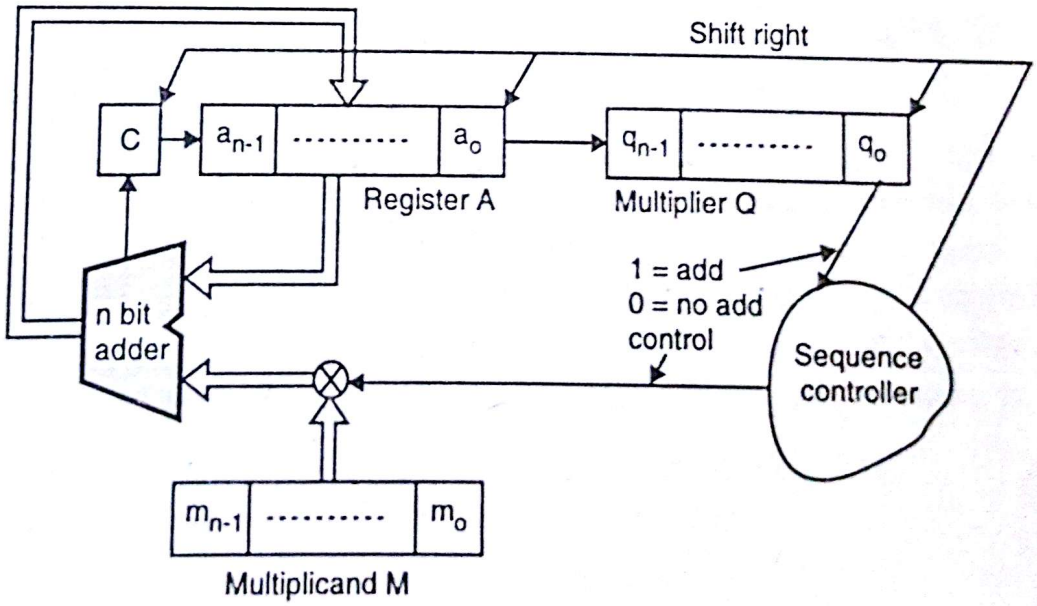



**Fig. 6.28**

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

Scanned by CamScanner

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

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

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

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

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

### 6.8.1 Fast Multiplier :

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

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

M = 6 = 0110

Q = 5 = 0101

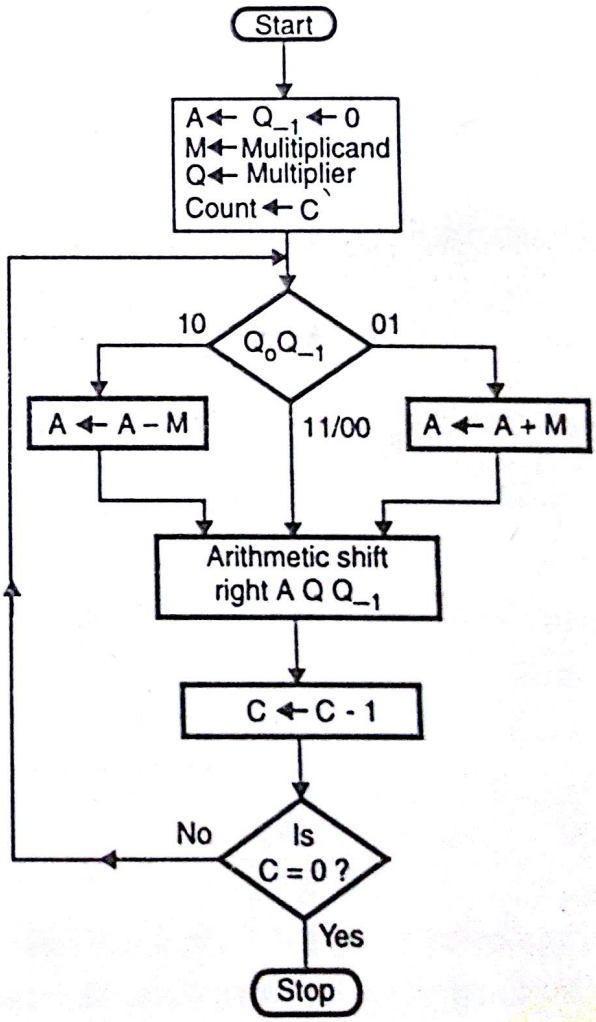



Fig. 6.29

Scanned by CamScanner

i.e.,

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

Four shifts as number of bits in multiplier are four.

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

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

| Multiplier bit | | Multiplicand selected by |
|---|---|---|
| i | i – 1 | bit i |
| 0 | 0 | 0 X M |
| 0 | 1 | +1 X M |
| 1 | 0 | –1 X M |
| 1 | 1 | 0 X M |

(i) Multiplicand and multiplier both positive $7 \times 3 = 21$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 1 | 1 | 1 | |
| | | | × | 0 | 0 | 1 | 1 | (0) |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 – 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 1 – 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | X | X | 0 – 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | = 21 |

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

(ii) Multiplicand positive and multiplier negative $7 \times -3 = 21$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 1 | 1 | 1 | |
| | | | × | 1 | 1 | 0 | 1 | (0) |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 – 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | X | 0 – 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | X | X | 1 – 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | = Two's complement of 21 |

(iii) Multiplicand negative and multiplier positive $-7 \times 3 = -21$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 0 | 0 | 1 | |
| | | | × | 0 | 0 | 1 | 1 | (0) |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 – 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 1 – 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | X | X | 0 – 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | = Two's complement of 21 |

← sign extend to 0 as the original number is negative

Scanned by CamScanner

(iv) Both, multiplicand and multiplier are negative $-7 \times -3 = +21$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 0 | 0 | 1 | |
| | | | × | 1 | 1 | 0 | 1 | (0) |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 – 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | X | 0 – 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | X | X | 1 – 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | = + 21 |

## 6.9 Comparator :

In many of the applications, we would like to compare two numbers, when we compare two numbers say A and B, there are only three possibilities :

(i) $A = B$ (ii) $A > B$ (iii) $A < B$.

In digital we have one device which compare two numbers, named ***Magnitude Comparator***. Why it is called as magnitude comparator because it basically compares value of two numbers. After comparing it produces three outputs $A = B$, $A > B$, $A < B$, we will just refresh concept by learning ***single bit magnitude comparator***.

### 6.9.1 Single Bit Magnitude Comparator :

Consider two, one bit numbers A and B. The truth Table is as shown in Table 6.6.

| Inputs | | Outputs | | |
|---|---|---|---|---|
| **A** | **B** | $Y_{A=B}$ | $Y_{A>B}$ | $Y_{A<B}$ |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |

**Table 6.6**

**K-map :**

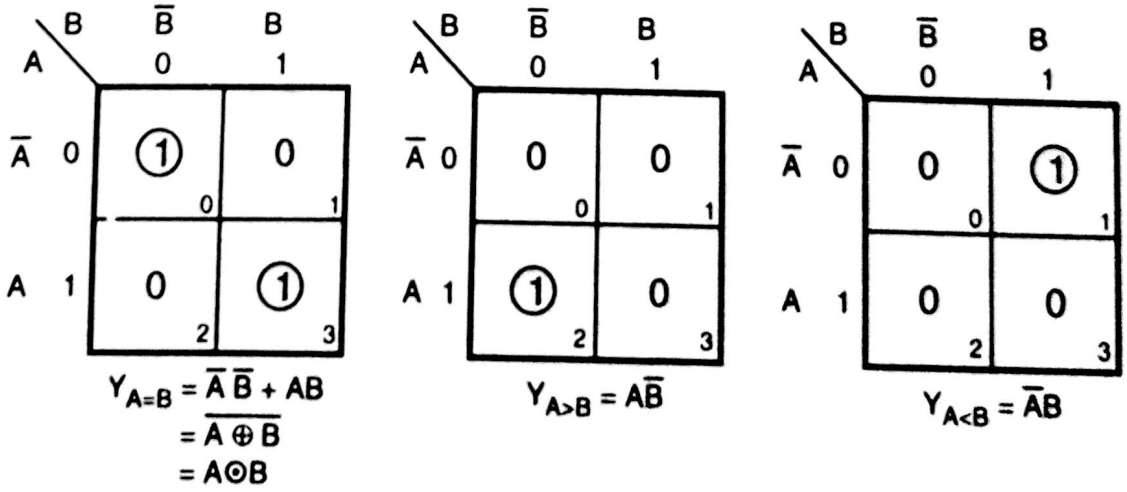



**Fig. 6.30**

Scanned by CamScanner

Same way if we proceed further for two, 2 bit magnitude comparator we have to draw 4 variable K map. For two, 3 bit magnitude comparator, we have to draw 6 variable K map. So as you go on increasing bit one has to face difficulty of drawing K map diagram.

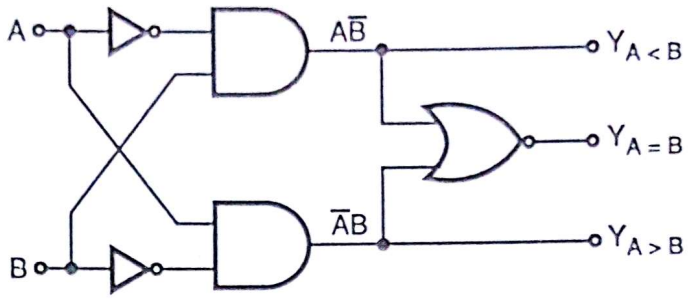



**Fig. 6.31**

Therefore we have to go for simple logic. Recall basic fundamental of comparing two numbers.

**Case I :** Let's say $A = (43)_{10}$ and $B = (26)_{10}$. Just by looking to most significant digit of A and B i.e. 4 and 2 respectively, one can conclude that A > B.

**Case II :** Let's say $A = (26)_{10}$ and $B = (43)_{10}$. Here by checking MSB digit you can conclude A < B.

**Case III :** Let's say $A = (43)_{10}$ and $B = (46)_{H}$. Here as MSB digits are same you switch over to next digit and will start comparing and from that you conclude something. But by chance next digits same, you declare A = B.

**Conclusion :**
(1) MSB is decision maker if MSB of two numbers is not same.
(2) But if MSB is equal, you switch over to next digit, and will again start.
(3) The step 2 will continue till you reach final LSB.

**Note :** Remember basic equation for $Y_{A=B}$, $Y_{A>B}$, and $Y_{A<B}$.

**Ex. 8 :** Design two bit magnitude comparator.

**Soln. :** Say $Y_{A_1=B_1} = x_1$

$Y_{A_0=B_0} = x_0$

One can write final equation of $Y_{A=B}$

$Y_{A=B} = x_1.x_0$ [means $A_1 = B_1$ AND $A_0 = B_0$]

$Y_{A<B} = \bar{A}_1B_1 + x_1.\bar{A}_0B_0$

$\bar{A}_1B_1 \rightarrow$ checking most significant bit and derived from basic equation.

$x_1 \cdot \bar{A}_0B_0 \rightarrow A_1 = B_1$ condition satisfied AND $\bar{A}_0\ B_0$ condition satisfied. So either of the condition satisfies output = 1.

$$Y_{A>B} = A_1\bar{B}_1 + x_1 \cdot A_0\ \bar{B}_0$$

We will verify the result with previous example

$$Y_{A=B} = x_1.x_0 = (A_1 \cdot B_1).(A_0.B_0)$$

$$Y_{A<B} = \bar{A}_1B_1 + (A_1.B_1)\ \bar{A}_0B_0 = \bar{A}_1B_1 + [A_1B_1 + \bar{A}_1\bar{B}_1]\ \bar{A}_0B_0$$

Scanned by CamScanner

$$= \bar{A}_1 B_1 + A_1 B_1 \bar{A}_0 B_0 + \bar{A}_1 \bar{B}_1 \bar{A}_0 B_0$$

$$= B_1 [\bar{A}_1 + A_1 \bar{A}_0 B_0] + \bar{A}_1 \bar{B}_1 \bar{A}_0 B_0 \quad [\text{Law } \bar{A} + AB = \bar{A} + B]$$

$$= B_1 [\bar{A}_1 + \bar{A}_0 B_0] + \bar{A}_1 \bar{B}_1 \bar{A}_0 B_0 = \bar{A}_1 B_1 + \bar{A}_0 B_0 B_1 + \bar{A}_1 \bar{B}_1 \bar{A}_0 B_0$$

$$= \bar{A}_1 [B_1 + \bar{B}_1 \bar{A}_0 B_0] + \bar{A}_0 B_0 B_1 \quad [\text{Law } A + \bar{A}B = A + B]$$

$$= \bar{A}_1 [B_1 + \bar{A}_0 B_0] + \bar{A}_0 B_0 B_1$$

$$\boxed{Y_{A<B} = \bar{A}_1 B_1 + \bar{A}_0 \bar{A}_1 B_0 + \bar{A}_0 B_1 B_0}$$

**Ex. 9 :** Write Boolean expression for 4 bit magnitude comparator circuit.

**Soln. :** $Y_{A=B}\, x_3.x_2.x_1.x_0$ where $x_3 = Y_{A_3=B_3}$; $x_2 = Y_{A_2=B_2}$

$$Y_{A<B} = \bar{A}_3 B_3 + x_3 \bar{A}_2 B_2 + x_3.x_2.\bar{A}_1 B_1 + x_3.x_2.x_1 \bar{A}_0 B_0$$

$$Y_{A>B} = A_3 \bar{B}_3 + x_3 A_2 \bar{B}_2 + x_3.x_2 A_1 \bar{B}_1 + x_3.x_2.x_1.A_0 \bar{B}_0$$

To make our task easy for composition, we have MSI chip 7485.

## 6.9.2 IC 7485 – 4 bit Magnitude Comparator :

It is feasible to implement two or more variable magnitude comparator using logic gates. We therefore use MSI chip 7485 (4 bit magnitude comparator).

**Feature :**

(1) Compares straight binary or BCD codes (4 bit).
(2) Outputs three lines, which indicate A = B, A > B or A < B.
(3) Typical power dissipation 275 mω.
(4) Typical delay (4 bit words) 23 η sec.

Pin configuration of the chip is as shown in Fig. 6.32.

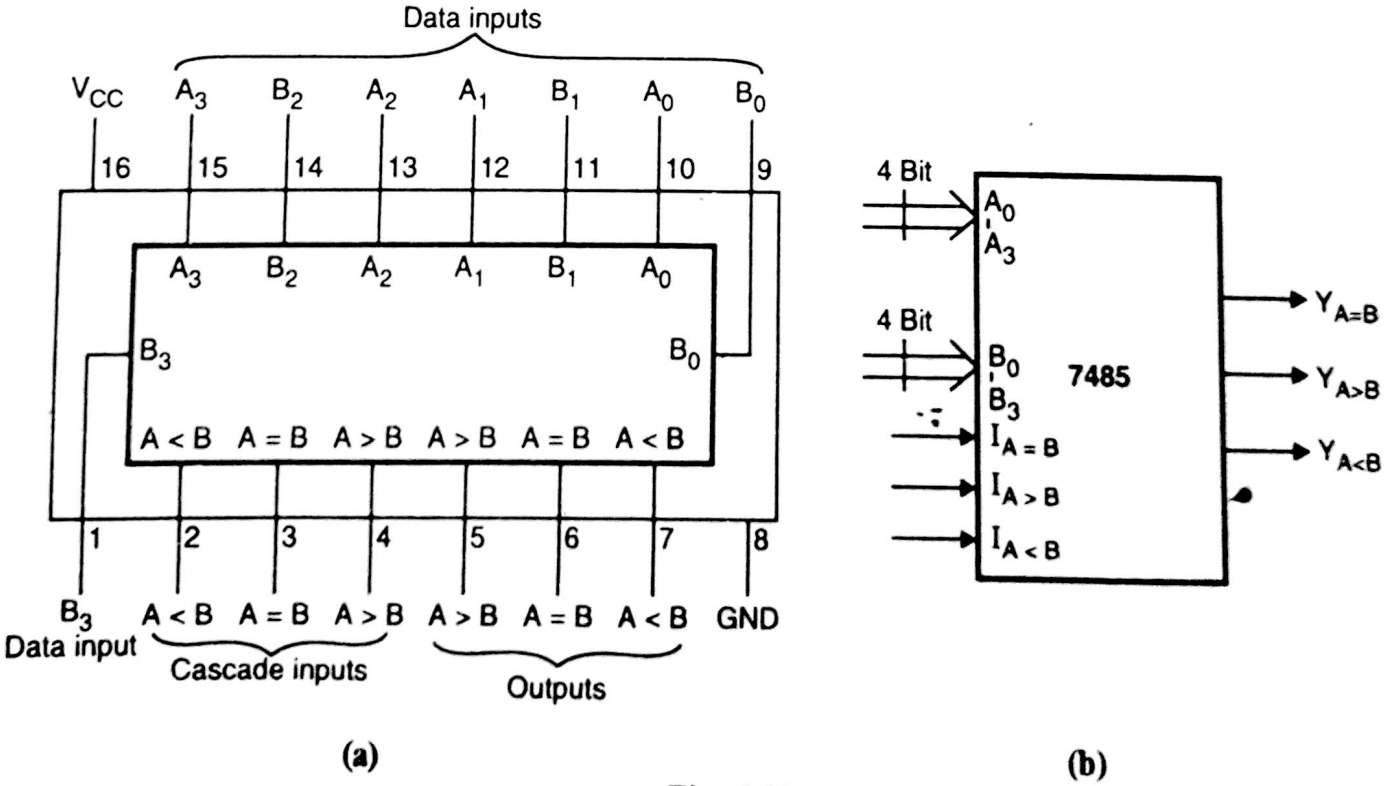



**Fig. 6.32**

Scanned by CamScanner

**Pin Names and Functions :**

| Pin Names | Function |
|---|---|
| $A_0 - A_3$ | Binary Input (operand 1) Active HIGH |
| $B_0 - B_3$ | Binary Input (operand 2) Active HIGH |
| Cascade A = B<br>Cascade A < B<br>Cascade A > B | These three lines are used for cascading 'N' number of 7485. Lines are input to the chip Output of previous stage is fed as input to this stage. |
| Output A = B<br>Output A > B<br>Output A < B | These three are final output from the chip When 7485 are cascaded, these outputs are given to cascading input of next stage (Active HIGH) |

The truth table is as follows :

| Comparing Inputs | | | | Cascading Inputs | | | Outputs | | |
|---|---|---|---|---|---|---|---|---|---|
| $A_3, B_3$ | $A_2, B_2$ | $A_1, B_1$ | $A_0, B_0$ | $I_{A>B}$ | $I_{A<B}$ | $I_{A=B}$ | $Y_{A>B}$ | $Y_{A<B}$ | $Y_{A=B}$ |
| $A_3 > B_3$ | X | X | X | X | X | X | H | L | L |
| $A_3 < B_3$ | X | X | X | X | X | X | L | H | L |
| $A_3 = B_3$ | $A_2 > B_2$ | X | X | X | X | X | H | L | L |
| $A_3 = B_3$ | $A_2 < B_2$ | X | X | X | X | X | L | H | L |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 > B_1$ | X | X | X | X | H | L | L |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 < B_1$ | X | X | X | X | L | H | L |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 > B_0$ | X | X | X | H | L | L |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 < B_0$ | X | X | X | L | H | L |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | H | L | L | H | L | L |
| $A_1 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | L | H | L | L | H | L |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | L | L | L | H | H | L |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | X | X | H | L | L | H |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | H | H | L | L | L | L |

**Table 6.7**

The function table is as shown in Table 6.8.

| Comparing inputs | Cascading inputs | | | Outputs | | |
|---|---|---|---|---|---|---|
| | $I_{A<B}$ | $I_{A=B}$ | $I_{A>B}$ | $Y_{A<B}$ | $Y_{A=B}$ | $Y_{A>B}$ |
| A < B | X | X | X | 1 | 0 | 0 |
| A = B | 0 | 0 | 0 | 1 | 0 | 1 |
| | 0 | 0 | 1 | 0 | 0 | 1 |
| | 1 | 0 | 0 | 1 | 0 | 0 |
| | 1 | 0 | 1 | 0 | 0 | 0 |
| | X | 1 | X | 0 | 1 | 0 |
| A > B | X | X | X | 0 | 0 | 1 |

**Table 6.8**

Scanned by CamScanner

The truth table states that if $I_{A<B}$ or $I_{A>B}$ is at logic 1 and other two cascading inputs at logic 0, then corresponding output will be active. In case A = B, then output will be true only is $I_{A=B}$ is at logic 1 irrespective of the other two inputs.

**General Description :**

7485 4 bit magnitude comparator perform comparison of straight binary or BCD codes. Three full decoded decisions about 4 bit words ($A_0 - A_3$, $B_0 - B_3$) are made and are externally available at three outputs ($Y_{A=B}$, $Y_{A>B}$, $Y_{A<B}$). These devices are fully expanded to any number of bits without external gates. Words of greater length may be compared by connecting comparator in cascade. For cascading, input lines $I_{A=B}$, $I_{A>B}$ and $I_{A<B}$ are provided. $Y_{A=B}$, $Y_{A>B}$ and $Y_{A<B}$ of previous stage should be connected to cascading inputs $I_{A=B}$, $I_{A>B}$ and $I_{A<B}$ respectively. Normally previous stage handles lower significant bits and next stage handles more significant bits.

Before we proceed for different configuration of IC 7485, let's study truth Table and functional Table. Refer Table 6.7 / 6.8. As shown, MSB is compared first i.e. $A_3$ and $B_3$. Depending upon $A_3 > B_3$ or $A_3 < B_3$, output $Y_{A>B}$ or $Y_{A<B}$ are active. But if $A_3 = B_3$, chip will compare $A_2$ and $B_2$. If $A_2 = B_2$, it will compare $A_1$ and $B_1$. If $A_1 = B_1$, it will compare $A_0$ and $B_0$. Now if $A_0 = B_0$, then $A_3\ A_2\ A_1\ A_3 = B_3\ B_2\ B_1\ B_0$, so now chip checks cascading inputs i.e. $I_{A>B}$, $I_{A=B}$ and $I_{A<B}$.

Here there are two cases, i.e. if cascading inputs are driven by previous stages, then no problem. But if somebody (user) connects switches to cascading input and try to 'FOOL' chip then response given, will be described as follows. If $I_{A<B} = 1$ and $I_{A=B}$, $I_{A>B} = 0$, then case is ***previous stage*** indicates that in least significant bit A is less than B.

If $I_{A>B} = 1$ and $I_{A=B}$ , $I_{A<B} = 0$ then case is ***previous stage*** indicates that in least significant bit A is greater than B. If $I_{A=B} = 1$, then chip will not check for $I_{A>B}$ and $I_{A<B}$ and concludes that least significant bits in previous stage are equal.

But if $I_{A>B}$ , $I_{A<B} = 1$ and $I_{A=B} = 0$, then you are informing chip that ***A is greater than B*** as well as ***A is less than B.*** Now one cannot think of such situation. Secondly if $I_{A>B} = I_{A<B} = I_{A=B} = 0$, means you are informing chip that neither of the condition is true. This is again ***impossible.***

Then here chip will also fool you i.e. if you apply $I_{A>B} = I_{A<B} = 1$ and $I_{A=B} = 0$, then chip given you output $I_{A>B} = I_{A<B} = I_{A=B} = 0$. And if you apply $I_{A>B} = I_{A<B} = I_{A=B} = 0$, chip gives you output $I_{A>B}$ , $I_{A<B} = 1$ and $I_{A=B} = 0$. So if you observe the situation $I_{A=B}$ remains at '0' logic and $I_{A>B}$ and $I_{A<B}$ are 'complemented' and given as an output.

**Note :** This output is also important i.e. $I_{A>B} = I_{A<B}$ = '0' or '1'. This will help us out to implement 5 bit comparator using single chip.

**Ex. 10 :** Design 8 bit comparator using IC 7485.

**Soln. :** The circuit for the 8 bit comparator is shown in Fig. Ex. 6.10.

(1) As shown in Fig. Ex. 6.10 operand 1 ($X_0 - X_7$) and operand ($Y_0 - Y_7$) are applied to magnitude comparator for comparison.

(2) Cascading outputs of $7485_1$ is given to cascading inputs of $7485_2$ i.e. $Y_{A>B}$, $Y_{A=B}$ and $Y_{A<B}$ of $7485_1$, is interfaced to $I_{A>B}$, $I_{A=B}$ and $I_{A<B}$.

(3) Final output is taken from $7485_2$.

Scanned by CamScanner

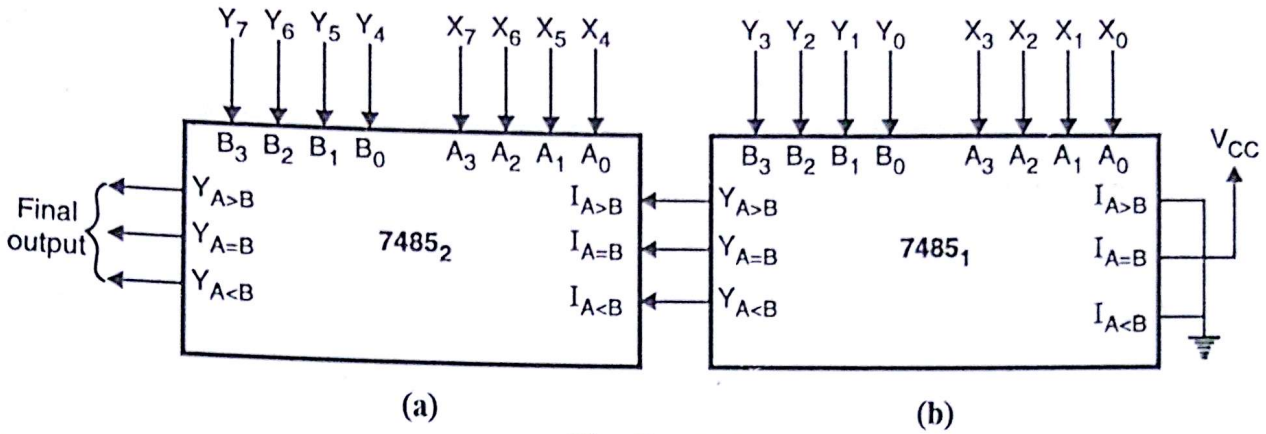



**Fig. Ex. 6.10**

**Operation :** When operand$_1$ and operand$_2$ is applied.

(1) $7485_2$ will start comparison of $A_3$ and $B_3$ i.e. $X_7$ and $Y_7$ respectively. $X_7$ and $Y_7$ are MSB bit of operand.

(2) As we know if $A_3 = B_3$, $7485_2$ will switch over to $A_2$, $B_2$. If $A_2 = B_2$, it will switch over to $A_1$ $B_1$ and so on.

(3) If $A_3 - A_0 = B_3 - B_0$ (i.e. $X_7 - X_4 = Y_7 - Y_4$) of $7485_2$, then finally, chip will check cascading inputs. Cascading input to this chip is nothing but output of $7485_1$.

(4) Now $7485_1$ will execute the step 1 and 2 same as that followed by $7485_2$. But here comparison is made between $X_3 - X_0$ and $Y_3$ to $Y_0$.

(5) If $A_3 - A_0 = B_3 - B_0$ (i.e. $X_3 - X_0 = Y_3$ to $Y_0$) of $7485_1$, then $7485_1$ will check cascading inputs. Now here important point is ***As $7485_1$ is LSB chip, we have to adjust logic level at cascading input is such a way that it should declare A = B.***

This statement is valid only when chip has LSB input. Referring to function (or truth Table) we conclude that if $I_{A=B} = 1$ and $I_{A>B} = I_{A<B} = X$, chip will give output $Y_{A=B} = 1$, $I_{A>B} = I_{A<B} = 0$.

$\therefore$ In circuit diagram, $I_{A=B} = 1 = V_{CC}$ and $I_{A>B} = I_{A<B} = 0 = GND$.

---

**Ex. 11 :** Design 5 bit magnitude comparator using magnitude comparator chip (7485).

**Soln. :** We know by now that output of a comparator is based on the data inputs as well as cascading inputs. Also, comparison always begins from MSB bit. As only four bits are available per variable, we accommodate the fifth bit for $A_0$ and $B_0$ i.e. LSB's on $I_{A>B}$ and $I_{A<B}$ inputs. This leaves with $I_{A=B}$ whose connection can be found out based on function Table.

| Comparing inputs | | Cascading Inputs | | | Outputs | | |
|---|---|---|---|---|---|---|---|
| | | $I_{A<B}$ | $I_{A=B}$ | $I_{A>B}$ | $Y_{A<B}$ | $Y_{A=B}$ | $Y_{A>B}$ |
| A = B | (a) | 0 | 0 | 0 | 1 | 0 | 1 |
| | (b) | 0 | 0 | 1 | 0 | 0 | 1 |
| | (c) | 1 | 0 | 0 | 1 | 0 | 0 |
| | (d) | 1 | 0 | 1 | 0 | 0 | 0 |
| | (e) | X | 1 | X | 0 | 1 | 0 |

**Table 3**

Scanned by CamScanner

Condition (c) is ruled out as it always indicate $Y_{A=B} = 1$.

The other cases imposes a condition that $I_{A=B}$ must be ground.

Cases (b) and (c) gives true output. However for case (a) and (d), output $Y_{A=B}$ must be at logic 1.

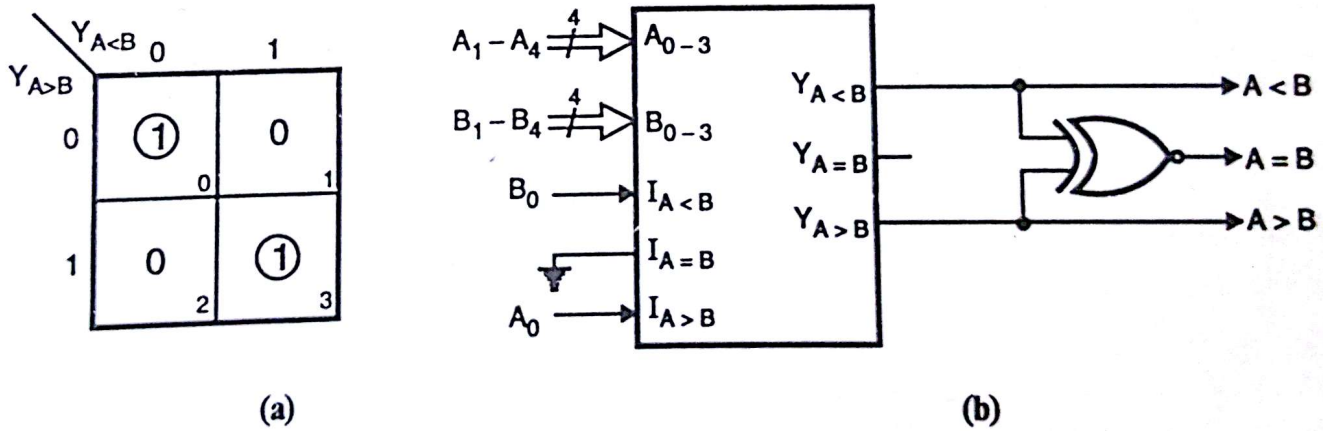
This can be implemented by considering that,




Since output $Y_{A=B}$ cannot be used, we design a new output for A = B as a function of $Y_{A<B}$ and $Y_{A>B}$ i.e. when $Y_{A>B} = Y_{A<B}$ = '0' or '1', output $Y_{A=B} = 1$.

| Input | | Output |
|---|---|---|
| $Y_{A>B}$ | $Y_{A<B}$ | $Y_{A=B}$ (Output) |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**K map :**




(a) (b)

**Fig. Ex. 6.11**

$$Y_{A=B} = \overline{Y_{A>B}}\ \overline{Y_{A<B}} + Y_{A<B}\, Y_{A>B} = Y_{A>B} \odot Y_{A<B}.$$

$\therefore$ Final circuit diagram is as shown in Fig. Ex. 6.11(b).

**Ex. 12 :** Design 24 bit magnitude comparator using minimum magnitude comparator chips (7485).

**Soln. :** Refer Fig. Ex. 6.12

Scanned by CamScanner

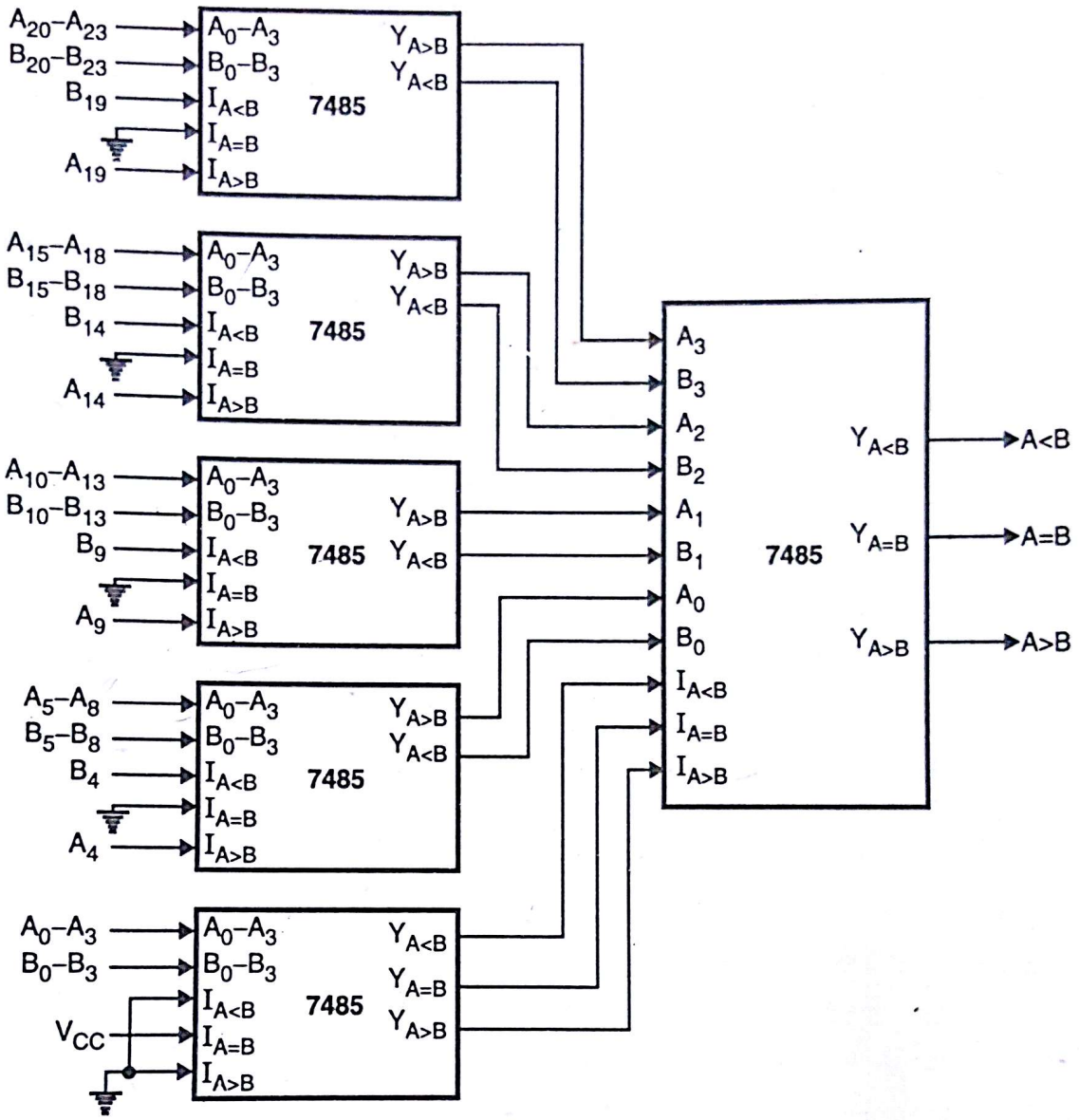



**Fig. Ex. 6.12**

Here we can cascade six chips. But the problem is of propagation delay, therefore circuit is bit slower.

The 24 bit input is fed to five comparators, since there are only 20 inputs per variable the remaining four are accommodated as explained in previous problem. This connection is not done for LSB nibble comparator.

Scanned by CamScanner

## 6.10 Examples from University Papers :

**1. Explain working of carry look ahead adder and give its advantage.**

**(May 96, Dec. 96, Dec. 97, 6 Marks)**

**Ans.:** Refer Section 6.2.3, Page 6-5 to 6.7, till Fig. 6.9 only.

**2. Implement 8–bit adder using 4–bit full adder.** **(June 99, 6 Marks)**

**Ans.:** Refer Ex. 3 on Page 6-9.

**3. Implement a 2 bit adder to add two 2– bit numbers, $A = A_1A_0$ and $B = B_1B_0$, using minimum number of half adder blocks.** **(June 99, 6 Marks)**

**Ans. :** As the numbers are $A = A_1A_0$ and $B = B_1B_0$ therefore output will be sum 1 ($S_1$), sum 0 ($S_0$) and Carry (C).

$$\begin{array}{lll} \therefore & A_1 & A_0 \\ + & B_1 & B_0 \\ \text{Carry} \rightarrow \boxed{C} & \boxed{C_0} & \\ \hline & S_1 & S_0 \end{array}$$

As half adder has 2 inputs A, B and 2 outputs sum (S) and carry (C).

Its truth table is

| Input | | Output | |
|---|---|---|---|
| A | B | S | C |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$S = A \oplus B$

$C = A \cdot B$

**Truth Table 1**

∴ Using above equation we can design half adder as follows :

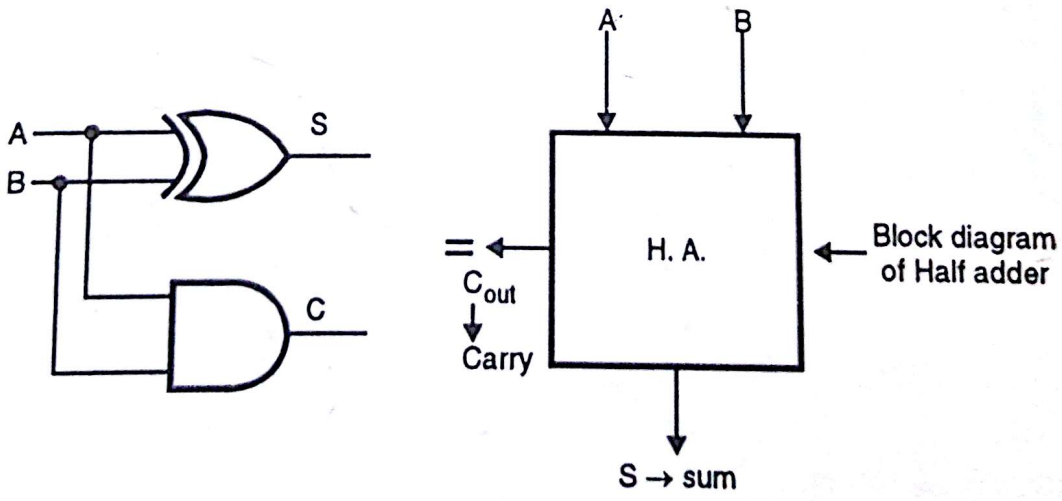



**Fig. Q. 6.3 (a)**

Scanned by CamScanner

**Block diagram of 2–bit adder using half adders :**

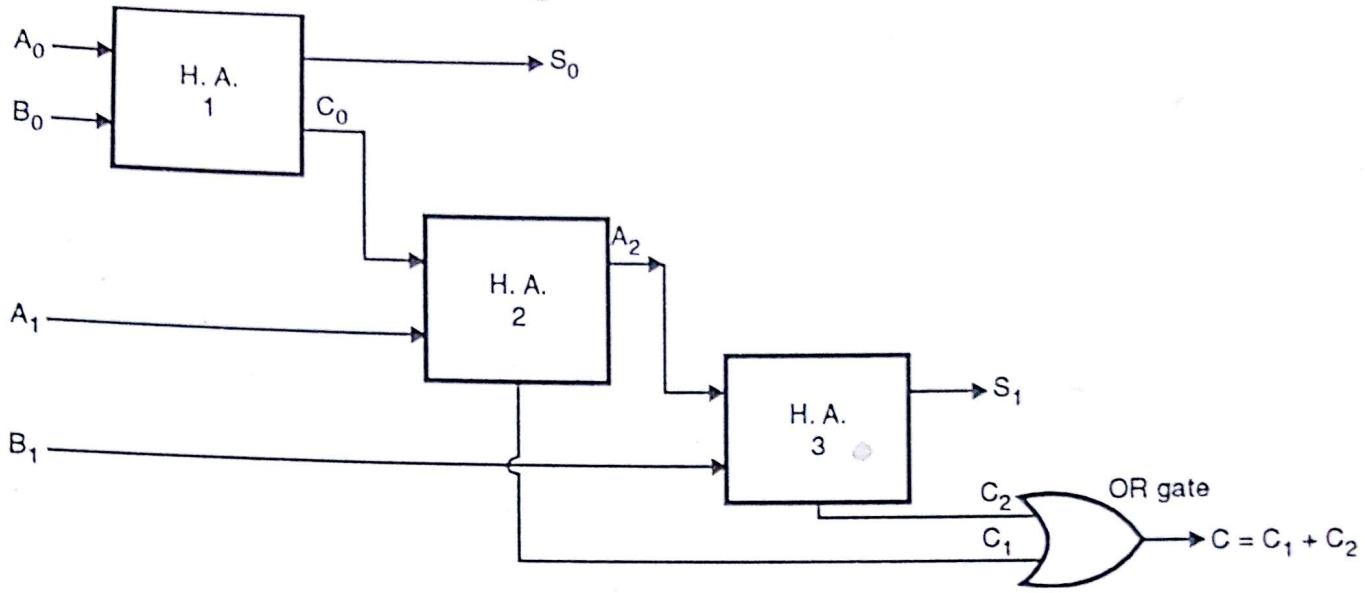



**Fig. Q. 6.3(b)**

**Step 1 :** We can say $A_0 + B_0$ gives $S_0$ and carry $C_0$.

$$S_0 = A_0 \oplus B_0 \quad \ldots(1)$$
$$C_0 = A_0 \cdot B_0$$

**Step 2 :** $C_0$ is added to $A_1$.

$$\therefore \quad \left.\begin{aligned} C_0 \oplus A_1 &= A_2 \rightarrow \text{Sum} \\ C_0 \cdot A_1 &= C_1 \rightarrow \text{Carry} \end{aligned}\right\} \text{ Outputs of Second half adder}$$

**Step 3 :** Now sum of second half adder $A_2$ is added to bit $B_1$ and carry $C_2$ is

$$\therefore \quad \left.\begin{aligned} A_2 \oplus B_1 &= S_1 \longrightarrow \text{Sum 1} \\ A_2 \cdot B_1 &= C_2 \longrightarrow \text{Carry} \end{aligned}\right\} \text{ Third half adder} \quad \ldots(2)$$

**Step 4 :** Now carry of third half adder $C_2$ is added to carry of second half adder $C_1$ which gives final carry C.

$$\therefore \quad C_2 + C_1 = C \longrightarrow \text{Carry} \quad \ldots(3)$$

**Diagrammatic representation of 2–bit adder using half adders and gates is as follows :**

Scanned by CamScanner

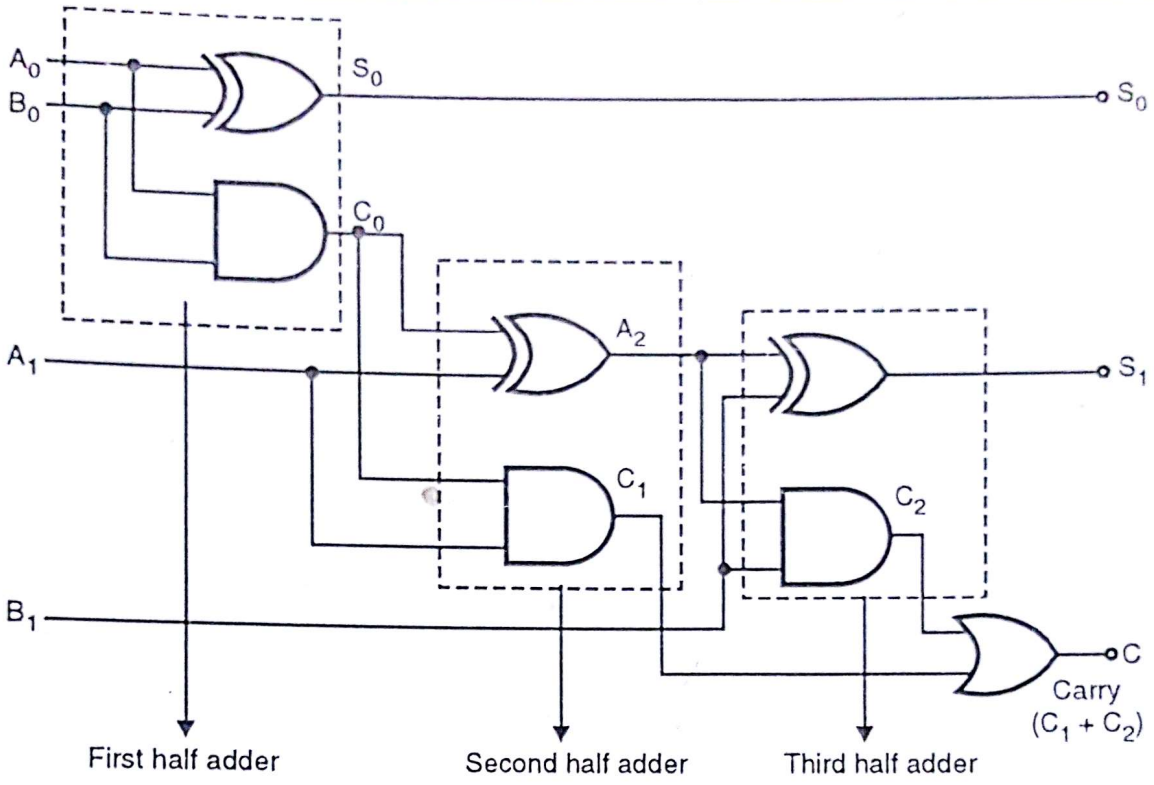



**Fig. Q. 6.3(c)**

❑❑❑

Scanned by CamScanner

# Multiplexer, Demultiplexer, ALU Encoder, Decoder

7

**Syllabus :**

Multiplexer, Demultiplexer, Decoder, Encoder, Priority encoder.

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Multiplexer | 7.2 | ✓ | ✓ |
| • Demultiplexer | 7.3 | ✓ | ✓ |
| • Decoder | 7.4 | ✓ | ✓ |
| • ALU | 7.5 | ✓ | ✓ |
| • Encoder | 7.6 | ✓ | ✓ |



**Comments**

- Full chapter is important.
- Concentrate on decoder because most widely used in microprocessor.
- MUX / DEMUX pair normally used in communication system.
- ALU used in CPU (Central processing unit)

Scanned by CamScanner

## 7.1 Introduction :

In this chapter we are going to learn about multiplexer, demultiplexer, encoder, decoder and ALU. The description is as follows in next section.

## 7.2 Multiplexer (Data Selectors) :

### Introduction

Multiplexer or data selector is very popular and most widely used combination circuit. It has multiple inputs and one output. i.e. it accepts several data inputs and allows only one of them AT A TIME to get through to the output. The routing of desired data input to output is controlled by SELECT LINES. The Fig. 7.1 shows generalized block schematic of multiplexer.

The multiplexer acts like a digitally controlled multiposition switch. In this case SELECT LINES will decide which data inputs ($D_{N-1} - D_0$) will appear over output. In short, multiplexer selects 1 out of N input data sources and transmits selected data to single output channel. This is called **multiplexing**.

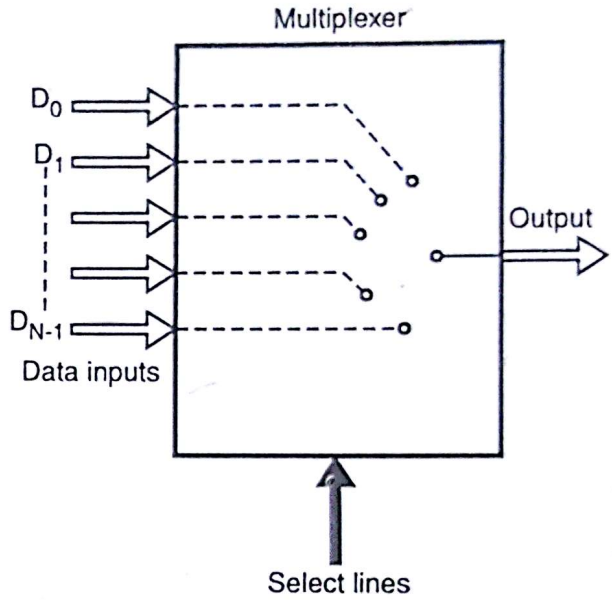



Fig. 7.1

### Basic 2 Input Multiplexer

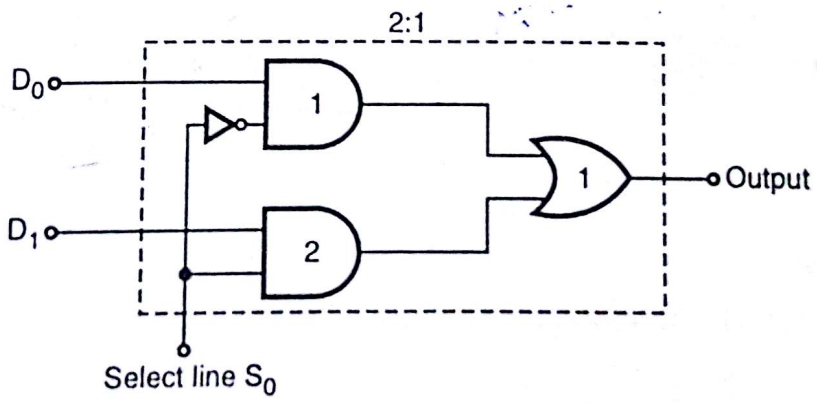



Fig. 7.2

Here basic concept is, we are going to use AND gate as 'GATING' device. AND gate OPENS gate when its one of the input is at logic '1', AND gate CLOSES gate when its one of the input is at logic '0'. Therefore one line to AND gate is data inputs i.e. $D_0 - D_1$. Now depending upon 'Address' of select lines or status of select lines, one has to generate logic '1' so that only that particular gates allows particular bit to propagate through it.

Scanned by CamScanner

Refer Fig. 7.2 when we want to pass $D_0$ bit, put $S_0 = 0$, therefore, AND1 will get enabled and AND2 is disabled giving AND2 output = 0. But AND1 gate is open. Therefore if $D_0 = 0$ OR1 gives final output = 0, if $D_0 = 1$, OR1 gives final output = 1. Thus AND1 acts as transparent gate.

But if select line $S_0 = 1$, AND1 is disabled, AND2 is enabled. Thus $D_1$ bit status is passed on to final output through OR1. This multiplexer can be named as 2 : 1 mux, where 2 is input lines, 1 is output line and mux is multiplexer. Let's see 4 : 1 circuit.

In the circuit we have introduced one more terminal i.e. E. E stands for 'enable.' This pin is master control to enable/disable multiplexer., E = 1 → enabled. E = 0 → disabled.

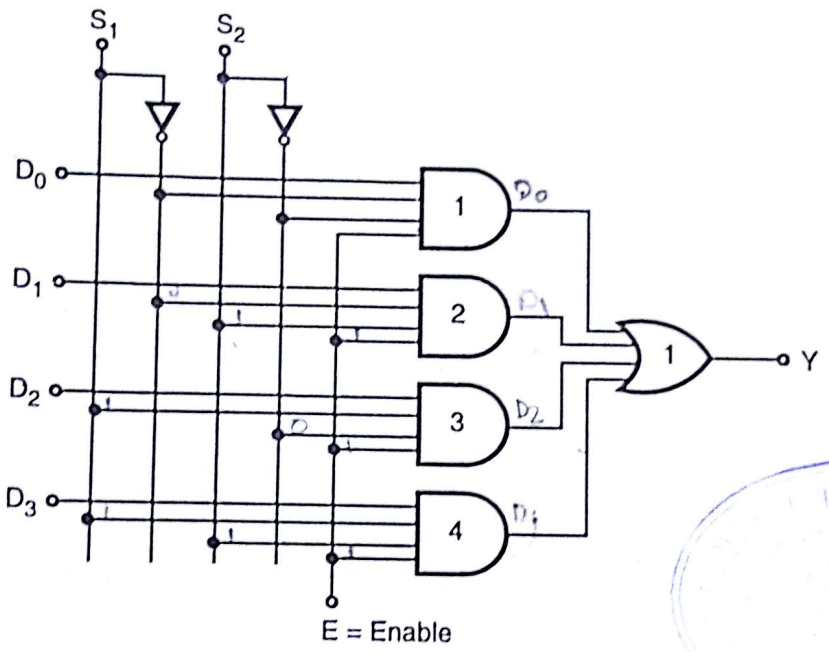



**Fig. 7.3**

Let's see the truth table. 4:1 Mux

| Inputs | | | Output |
|---|---|---|---|
| Enable | Select inputs | | |
| E | $S_1$ | $S_0$ | |
| 0 | X | X | 0 |
| 1 | 0 | 0 | $D_0$ |
| 1 | 0 | 1 | $D_1$ |
| 1 | 1 | 0 | $D_2$ |
| 1 | 1 | 1 | $D_3$ |

Same way you can also have 8 : 1, 16 : 1 multiplexer. The only change will be no. of select lines will increase and data lines will increase. The relationship between select lines and inputs is given by $n = 2^m$ where, m = number of select lines; n = Number of data inputs.

Scanned by CamScanner

MSI chips available are,

| Chips | Function | |
|---|---|---|
| 74150 | 16 : 1 MUX | Inverted output |
| 74152 | 8 : 1 MUX | Inverted output |
| 74157 | Quad 2 : 1 MUX | O/P same as input |
| 74158 | Quad 2 : 1 MUX | Inverted output |

**Ex. 1 :** Draw the logic diagram of 4 : 1 MUX using NAND gate only (input Active Low).

**Soln. :** To design a circuit using NAND gates we replace all AND gates in Fig. 7.3 by NAND gates. This will make the OR gate an input bubbled OR which is equivalent to a NAND gate. The logic diagram is shown in Fig. Ex. 7.1 (a).

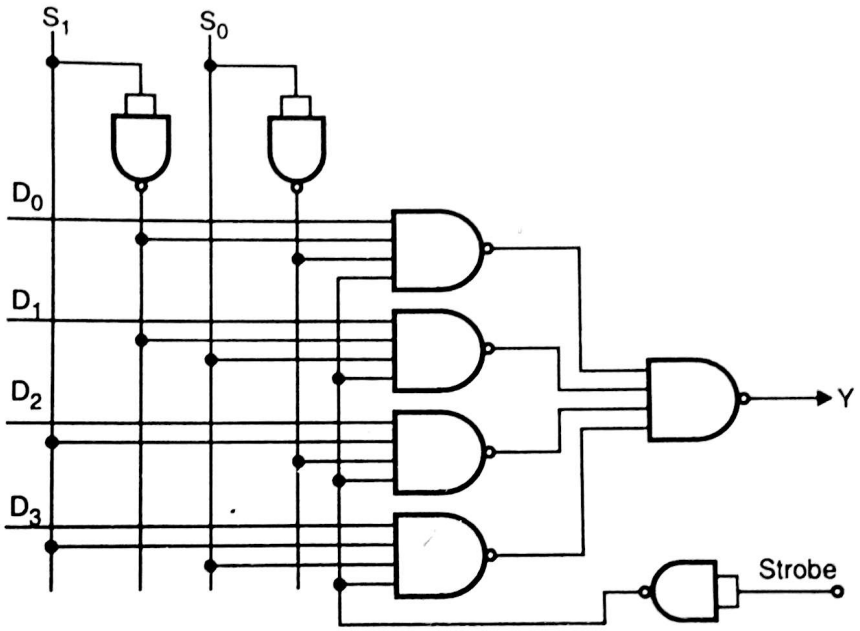



**Fig. Ex. 7.1 (a)**

**Ex. 2 :** Draw 8 : 1 Multiplexer using logic gates along with its truth table.

**Soln. :** Fig. Ex. 7.2 (a) shows 8 : 1 multiplexer using logic gates.

| Input | | | Output |
|---|---|---|---|
| $S_2$ | $S_1$ | $S_0$ | Y |
| 0 | 0 | 0 | $D_0$ |
| 0 | 0 | 1 | $D_1$ |
| 0 | 1 | 0 | $D_2$ |
| 0 | 1 | 1 | $D_3$ |
| 1 | 0 | 0 | $D_4$ |
| 1 | 0 | 1 | $D_5$ |
| 1 | 1 | 0 | $D_6$ |
| 1 | 1 | 1 | $D_7$ |

**Truth Table 1**

Scanned by CamScanner

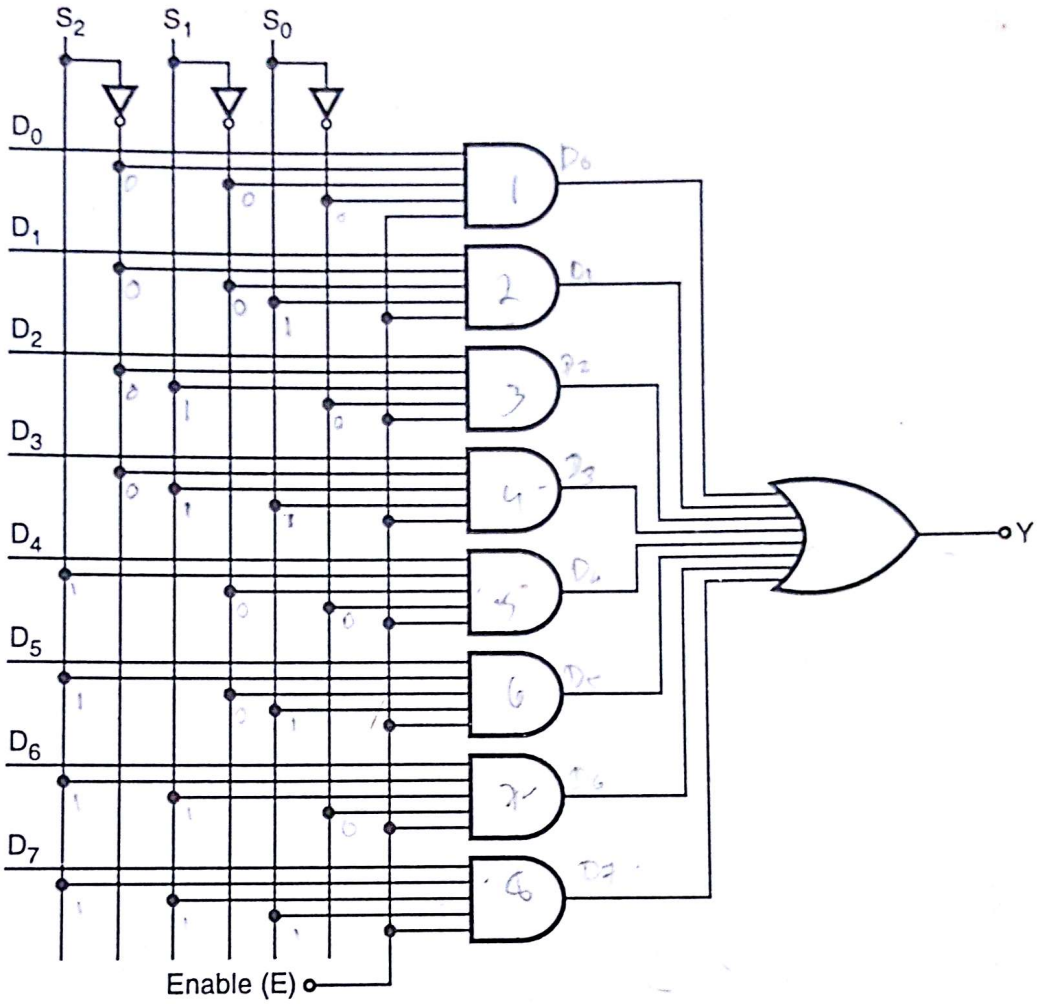



**Fig. Ex. 7.2 (a)**

Multiplexer can be also used to implement ***Combinational logic circuit (SOP FORM)***. This leads to minimum and organised hardware. Let's solve some examples.

**Ex. 3 :** Implement following function using 8 : 1 multiplexer.

$f = \Sigma\, m\,(0, 2, 3, 5, 7)$.

**Soln. :**

(1) Function specifies minterms

(2) There are total three variable A, B, C because maximum minterms given is 7, which requires three variables to represent in the binary.

Also being SOP form, output should be at logic 1 for the given minterms. The circuit diagram is shown in Fig. Ex. 7.3 (a).

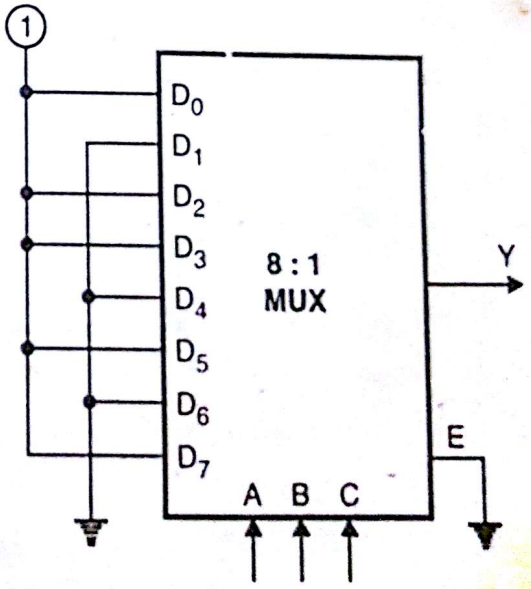



**Fig. Ex. 7.3 (a)**

Scanned by CamScanner

**Ex. 4 :** Implement full adder circuit using 8 : 1 MUX.

**Soln. :** Truth table for full adder is as follows :

| I / P | | | O / P | |
|---|---|---|---|---|
| A | B | C | Sum (S) | Carry |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

**Table 2**

The circuit is shown in Fig. Ex. 7.4..

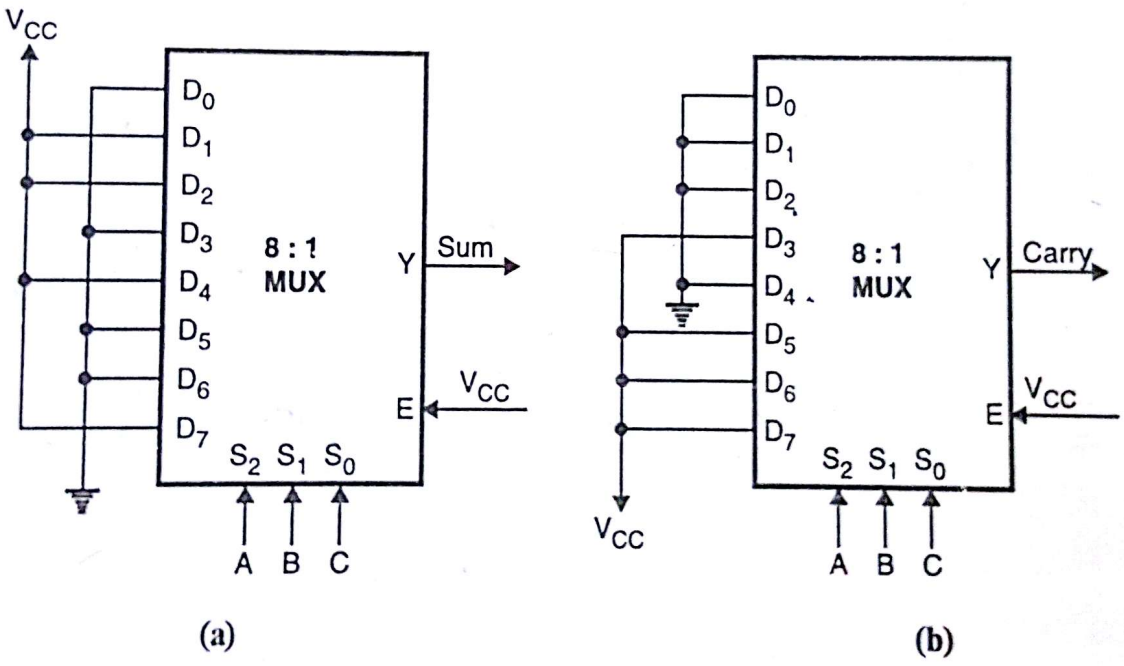



**Fig. Ex. 7.4**

**Ex. 5 :** Implement a 16 : 1 MUX using 8 : 1 MUX.

**Soln. :** This problem asks for cascading the multiplexer. Cascading takes care of the fact that MSB bit is always zero for first half of the minterms, and logic 1 for the remaining half. The MSB bit is therefore connected to enable input of MUX. The remaining lines are connected to SELECT lines. Refer Fig. Ex. 7.5 (a) .

In Fig. Ex. 7.5 (a) A, B, C, D are used to select any input line. A = MSB and D = LSB. A is going to enable / disable multiplexer. When A = 0, upper multiplexer will be enabled, therefore inputs of upper

Scanned by CamScanner

MUX are labelled from $D_0$ to $D_7$. When A = 1, lower multiplexer is enabled and therefore inputs of lower MUX are labelled as $D_8$ to $D_{15}$. The total combined gives us $D_0$ to $D_{15}$. The output of two MUX is logically ORed.

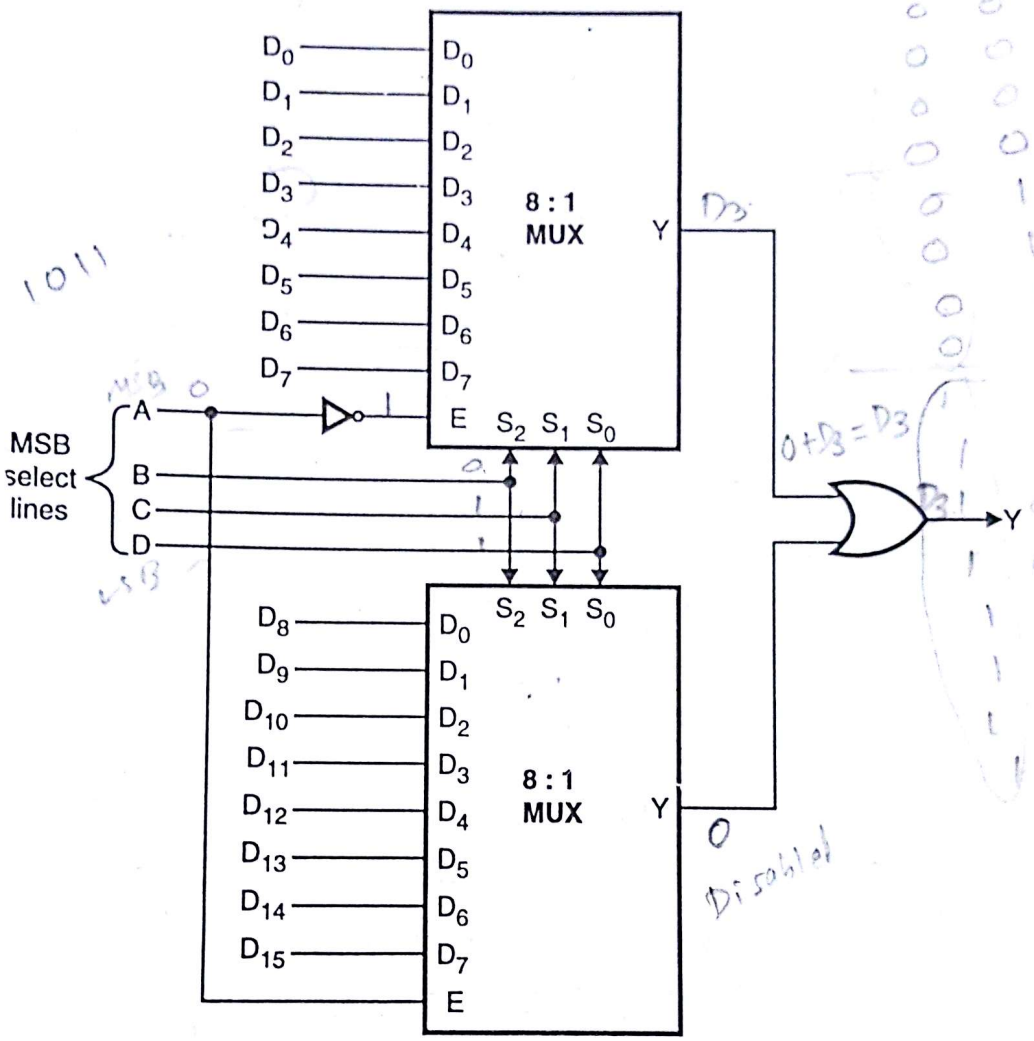



**Fig. Ex. 7.5 (a) : Cascading multiplexer**

**Ex. 6 :** Construct 16 : 1 MUX using 4 : 1 MUX (s).

**Soln. :** As we want to implement 16 : 1 MUX, we require 4 select lines. 4 : 1 MUX has only two select lines. The configuration or implementing circuit is shown in Fig. Ex. 7.6 (a).

As shown $S_1$ $S_0$ lines of $MUX_{1, 2, 3, 4}$ are tied together and given C and D inputs. Here we are using ABCD i.e. four variables, where A = MSB and D = LSB.

When A = 0, B = 0. $D_0$ line of $MUX_5$ will be selected. Now toggle CD from $(00)_2$ to $(11)_2$, this will select $D_0$, $D_1$, $D_2$, $D_3$ of $MUX_1$. As A = 0, B = 1, $D_1$ line of $MUX_5$ will be selected.

Now toggle CD from $(00)_2$ to $(11)_2$, this will select $D_4$, $D_5$, $D_6$, $D_7$ of $MUX_2$.

Now change A = 1, B = 0 and finally A = 1, B = 1, this will select $MUX_3$ and $MUX_4$ respectively.

Table 3 shows O/P selected.

Scanned by CamScanner

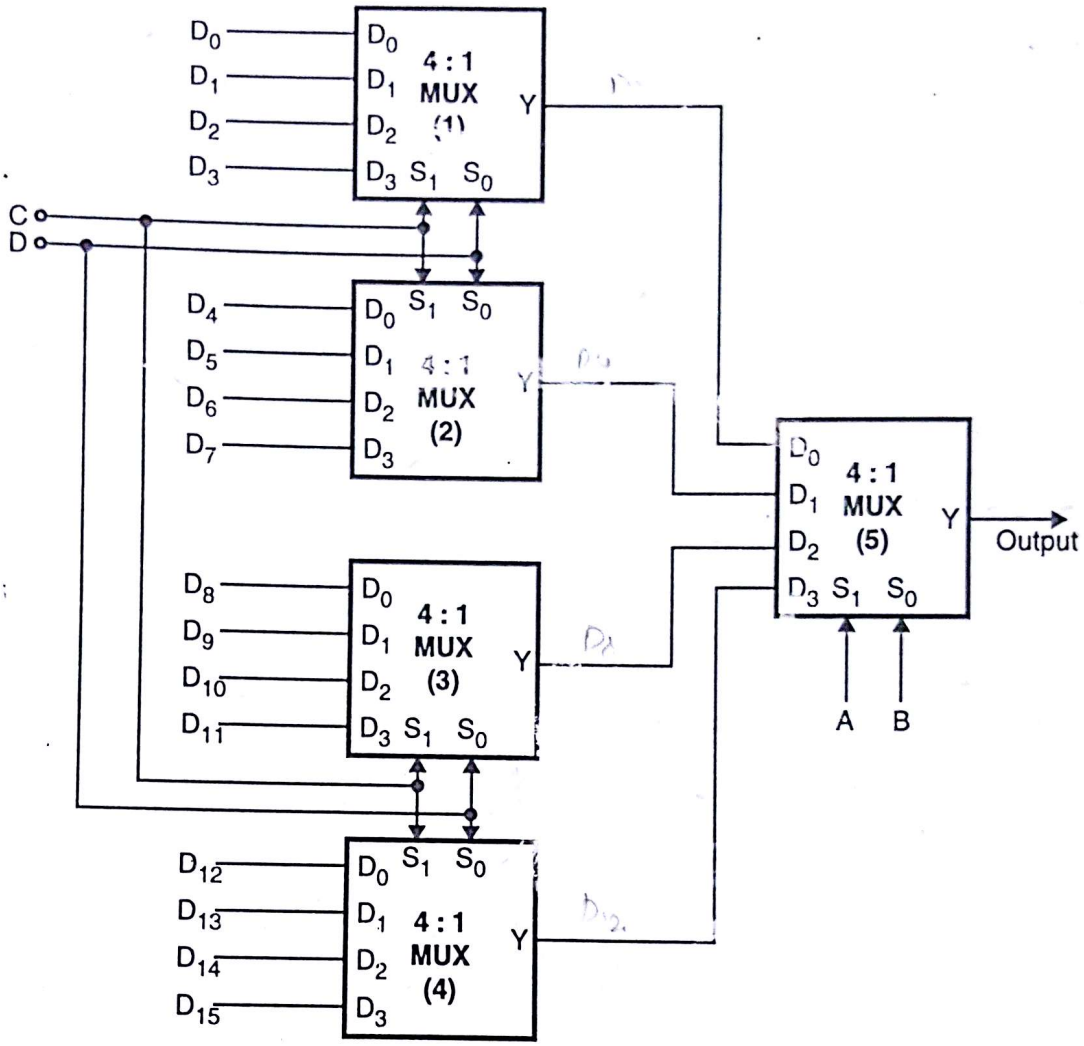



Fig. Ex. 7.6 (a) : MUX using 4 : 1

| Decimal | Inputs | | | | Output | |
|---|---|---|---|---|---|---|
| | A | B | C | D | Y | |
| 0 | 0 | 0 | 0 | 0 | $D_0$ | $MUX_1$ selected |
| 1 | 0 | 0 | 0 | 1 | $D_1$ | |
| 2 | 0 | 0 | 1 | 0 | $D_2$ | |
| 3 | 0 | 0 | 1 | 1 | $D_3$ | |
| 4 | 0 | 1 | 0 | 0 | $D_4$ | $MUX_2$ selected |
| 5 | 0 | 1 | 0 | 1 | $D_5$ | |
| 6 | 0 | 1 | 1 | 0 | $D_6$ | |
| 7 | 0 | 1 | 1 | 1 | $D_7$ | |
| 8 | 1 | 0 | 0 | 0 | $D_8$ | $MUX_3$ selected |
| 9 | 1 | 0 | 0 | 1 | $D_9$ | |
| 10 | 1 | 0 | 1 | 0 | $D_{10}$ | |
| 11 | 1 | 0 | 1 | 1 | $D_{11}$ | |
| 12 | 1 | 1 | 0 | 0 | $D_{12}$ | $MUX_4$ selected |
| 13 | 1 | 1 | 0 | 1 | $D_{13}$ | |
| 14 | 1 | 1 | 1 | 0 | $D_{14}$ | |
| 15 | 1 | 1 | 1 | 1 | $D_{15}$ | |

Table 3

Scanned by CamScanner

We have solved examples and in that we have taken very simple criteria into account i.e. number of select lines = number of variables. But it is also possible to implement a function having 'One variable more than select lines, i.e. we have 8 : 1 MUX having 3 select lines, but we can implement minterm function having 16 states containing four variables. Let's solve example to understand the concept.

**Ex. 7 :** Implement following functions using 8 : 1 MUX :

(a) $F = f(A, B, C, D) = \Sigma m (2, 4, 5, 7, 10, 14)$

(b) $F = f(A, B, C, D) = \Sigma m (0, 2, 4, 6, 8, 10, 12, 14)$.

**Soln. :** (a) (1) Given function has four variables A, B, C, D.

(2) Minterm numbers where O/P required is logical '1', are 2, 4, 5, 7, 10, 14.

(3) We have 8 : 1 MUX, which has three select lines $S_2$, $S_1$, $S_0$.

(4) Therefore $S_2$, $S_1$, $S_0$ will be connected to B, C, D input variables.

(5) To introduce fourth variable (select line) we use ***design table***.

(6) Since there are only eight inputs lines, we have to accommodate the minterms in two rows of eight each (total 16 minterms for 4 variables), in design table. Design table is as shown in Table 4.

| Inputs | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|---|
| $\bar{A}$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| A | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Final I/P to MUX | | | | | | | | |

**Table 4**

Table 5 shows generalized structure.

(b) Now you have to encircle the terms specified by the function i.e. 2, 4, 5, 7, 10, 14, in the design table.

| Inputs | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|---|
| $\bar{A}$ | 0 | 1 | (2) | 3 | (4) | (5) | 6 | (7) |
| A | 8 | 9 | (10) | 11 | 12 | 13 | (14) | 15 |
| Final I/P to MUX | | | | | | | | |

**Table 5**

After encircling the required minterm you would like to fill the row having heading ***Final I/P to MUX***. For that you have to follow some standard rule which are as follows :

(1) If all the minterms in column are circled, put a logic 1 below i.e. at ***Final I/P to MUX*** row. It means that output must be at logic 1, if either of the minterm is selected.

(2) If in column, only top row minterm is circled, then being amongst lower 16 minterms, its MSB bit must be at logic '0' i.e. variable A must be ground. Therefore $\bar{A}$ is selected.

(3) If in a column, only bottom row minterm is selected or circled, then being amongst upper 16 minterms, its MSB must be at logic 1 i.e. variable A must be Vcc. Therefore A is selected instead of $\bar{A}$

Scanned by CamScanner

(4) If none of the minterms in a particular column are selected, put logic '0' below i.e. output must be logical '0' as there minterms are not a part of SOP expression.

∴ Finally the table will be as follows :

| Inputs | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|---|
| $\bar{A}$ | 0 | 1 | (2) | 3 | (4) | (5) | 6 | (7) |
| A | 8 | 9 | (10) | 11 | 12 | 13 | (14) | 15 |
| Final I/P to MUX | 0 | 0 | 1 | 0 | $\bar{A}$ | A | A | $\bar{A}$ |

**Table 6**

The circuit diagram is shown in Fig. Ex. 7.7 (a).

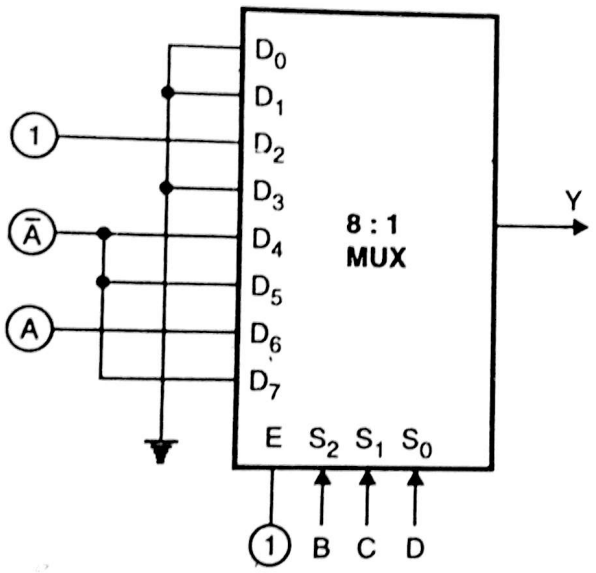



**Fig. Ex. 7.7 (a) : Circuit diagram**

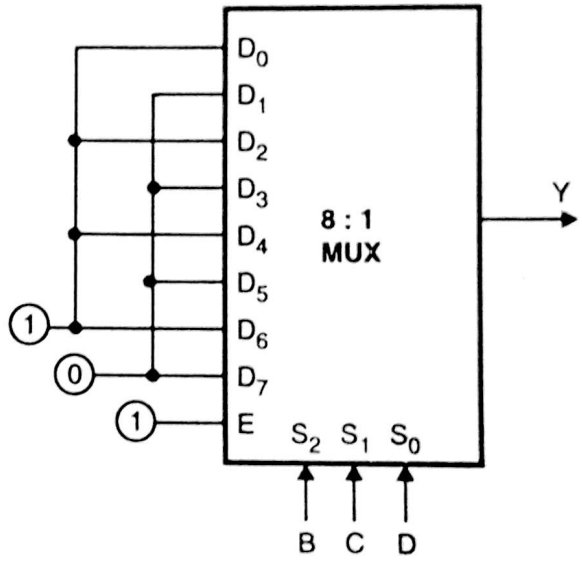



**Fig. Ex. 7.7 (b) : Circuit diagram**

(b) The design table for this problem is shown in Table 7.

| Inputs | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|---|
| $\bar{A}$ | (0) | 1 | (2) | 3 | (4) | 5 | (6) | 7 |
| A | (8) | 9 | (10) | 11 | (12) | 13 | (14) | 15 |
| Final I/P to MUX | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

**Table 7**

The circuit diagram is shown in Fig. Ex. 7.7 (b).

---

**Ex. 8 :** Implement following function using 16 : 1 MUX.

$F(A, B, C, D, E) = \Sigma m (2, 4, 5, 7, 10, 14, 15, 16, 17, 25, 26, 30, 31)$

**Soln. :** Given five variable function is to be implemented using 16 : 1 MUX 74150, 74150 has only four select lines. Therefore we have to introduce one more variable (or select line). To achieve this we design Table 8.

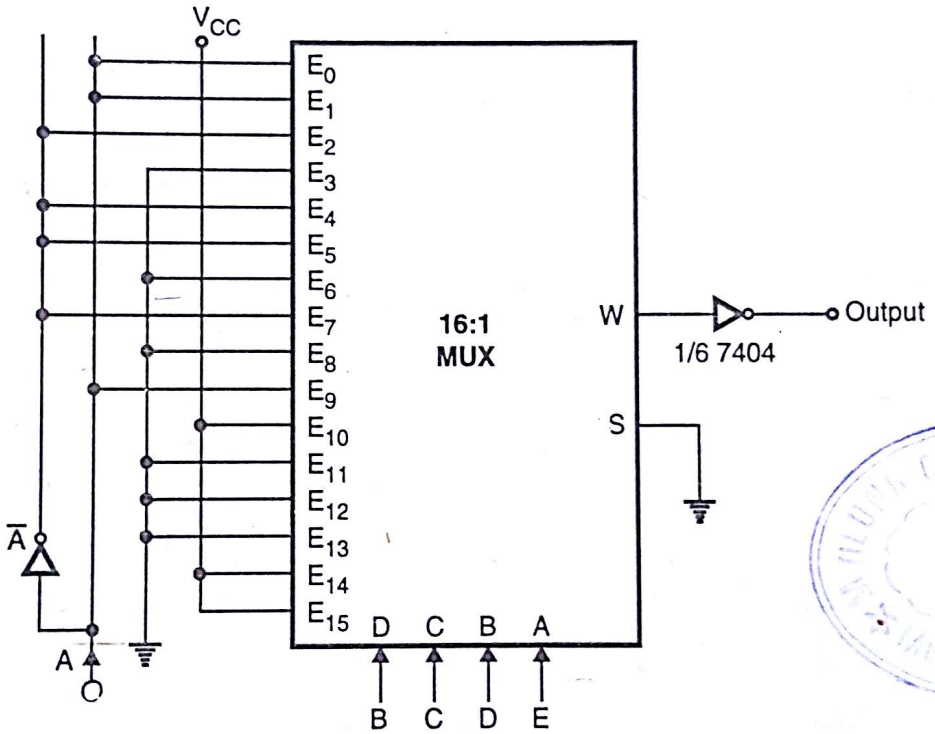
Scanned by CamScanner

| Inputs | $D_0$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bar{A}$ | 0 | 1 | (2) | 3 | (4) | (5) | 6 | (7) | 8 | 9 | (10) | 11 | 12 | 13 | (14) | (15) |
| A | (16) | (17) | 18 | 19 | 20 | 21 | 22 | 23 | 24 | (25) | (26) | 27 | 28 | 29 | (30) | (31) |
| Final I/P to MUX | A | A | $\bar{A}$ | 0 | $\bar{A}$ | $\bar{A}$ | 0 | $\bar{A}$ | 0 | A | 1 | 0 | 0 | 0 | 1 | 1 |

**Table 8**

Since there are only 16 inputs, to accommodate minterms from 0 to 31, we made two rows of 16 each. Here A = MSB and D = LSB. Now, after writing states from 0 to 31, circle the variables (or state) given in expressions in the design table. Follow the rules for deciding final I/P to MUX.

**Circuit diagram :**




LIBRARY 743

**Fig. Ex. 7.8 (a)**

**Ex. 9 :** Realise the following expression using 16 : 1 MUX.
$F = f(A, B, C, D, E) = \Sigma m (1, 9, 10, 12, 18)$.

**Soln. :** Form design table first.

| Inputs | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $D_8$ | $D_9$ | $D_{10}$ | $D_{11}$ | $D_{12}$ | $D_{13}$ | $D_{14}$ | $D_{15}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bar{A}$ | (0) | (1) | 2 | 3 | 4 | 5 | 6 | 7 | 8 | (9) | (10) | 11 | (12) | 13 | 14 | 15 |
| A | 16 | 17 | (18) | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Final I/P to MUX | 0 | $\bar{A}$ | A | 0 | 0 | 0 | 0 | 0 | 0 | $\bar{A}$ | $\bar{A}$ | 0 | $\bar{A}$ | 0 | 0 | 0 |

**Table 9**

Given a 16 : 1 MUX whereas minterm 18 requires five variables. Now draw circuit diagram.

Scanned by CamScanner

D

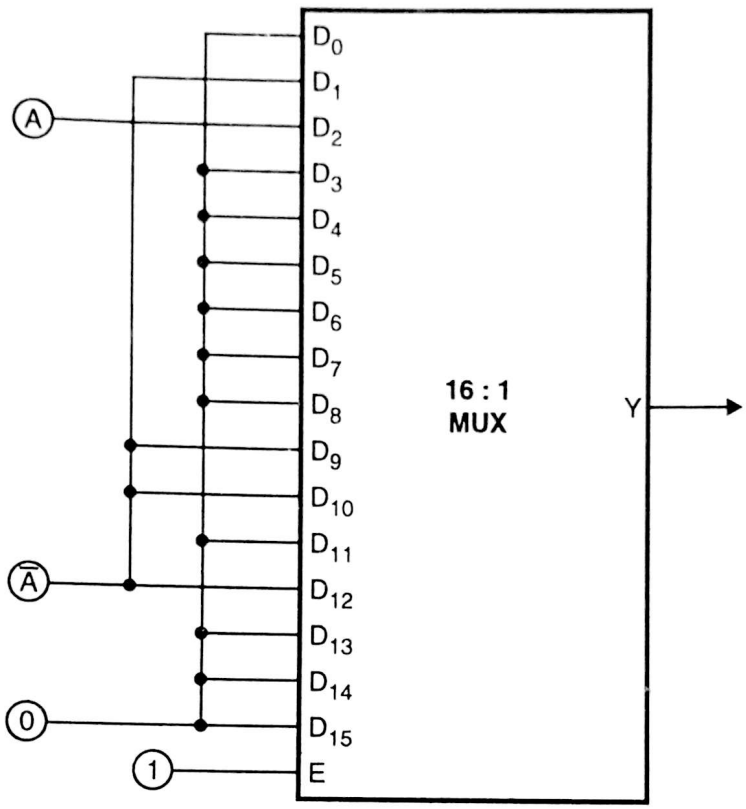



**Fig. Ex. 7.9 (a)**

**Ex. 10 :** Implement the following function :

f (A, B, C, D, E) = Σ m (0, 1, 2, 3, 6, 8, 9, 10, 13, 15, 17, 20, 24, 30) using,

(i) all 4 : 1 MUX

(ii) all 8 : 1 MUX

(iii) all 16 : 1 MUX

**Soln. :** Design table will be as shown in Table 10 :

To incorporate variable A we design the circuit as shown in Fig. Ex. 7.10(a).

| Inputs | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $D_8$ | $D_9$ | $D_{10}$ | $D_{11}$ | $D_{12}$ | $D_{13}$ | $D_{14}$ | $D_{15}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bar{A}$ | (0) | (1) | (2) | (3) | 4 | 5 | (6) | 7 | (8) | (9) | (10) | 11 | 12 | (13) | 14 | (15) |
| A | 16 | (17) | 18 | 19 | (20) | 21 | 22 | 23 | (24) | 25 | 26 | 27 | 28 | 29 | (30) | 31 |
| Final I/P to MUX | $\bar{A}$ | 1 | $\bar{A}$ | $\bar{A}$ | A | 0 | $\bar{A}$ | 0 | 1 | $\bar{A}$ | $\bar{A}$ | 0 | 0 | $\bar{A}$ | A | $\bar{A}$ |

**Table 10**

(i) Use all 4 : 1 MUX :

We have already constructed 16 : 1 MUX using, 4 : 1 MUX (Ex. 6). Using 16 : 1 MUX you can implement function having 32 states (0 to 31) . The circuit is shown in Fig. Ex. 7.10 (a).

Scanned by CamScanner

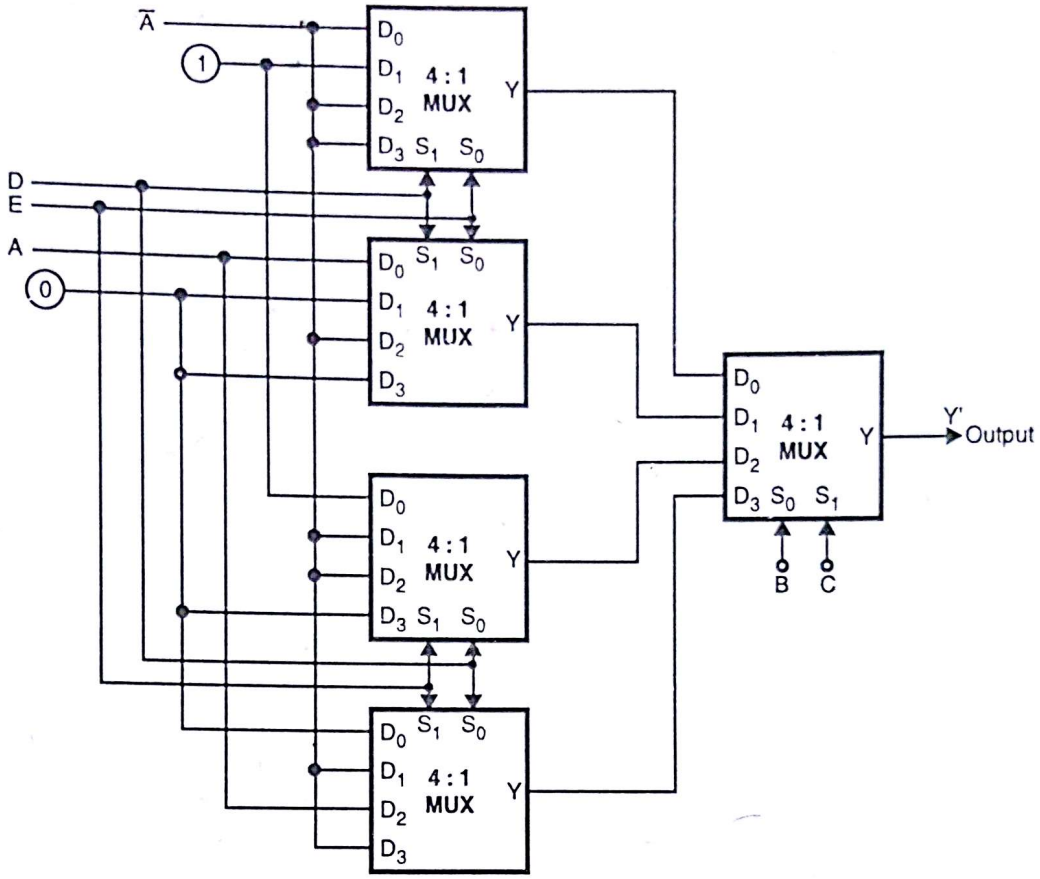



**Fig. Ex. 7.10 (a)**

(ii) Using 8 : 1 MUX :

We will first construct 16 : 1 using 8 : 1 MUX, and then apply the design table of Table 10.

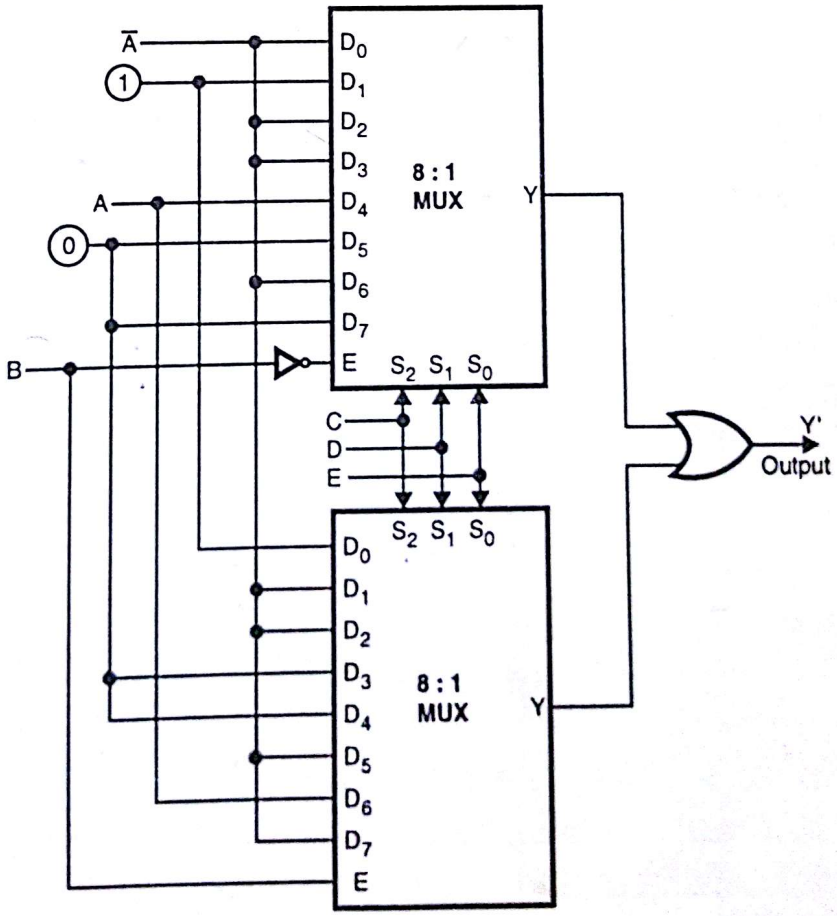



**Fig. Ex. 7.10 (b)**

Scanned by CamScanner

(iii) Using 16 : 1 MUX :
Since, it already has 16 inputs (4 variables) we introduce fifth MSB variable A at input as shown in Fig. Ex. 7.10 (c).

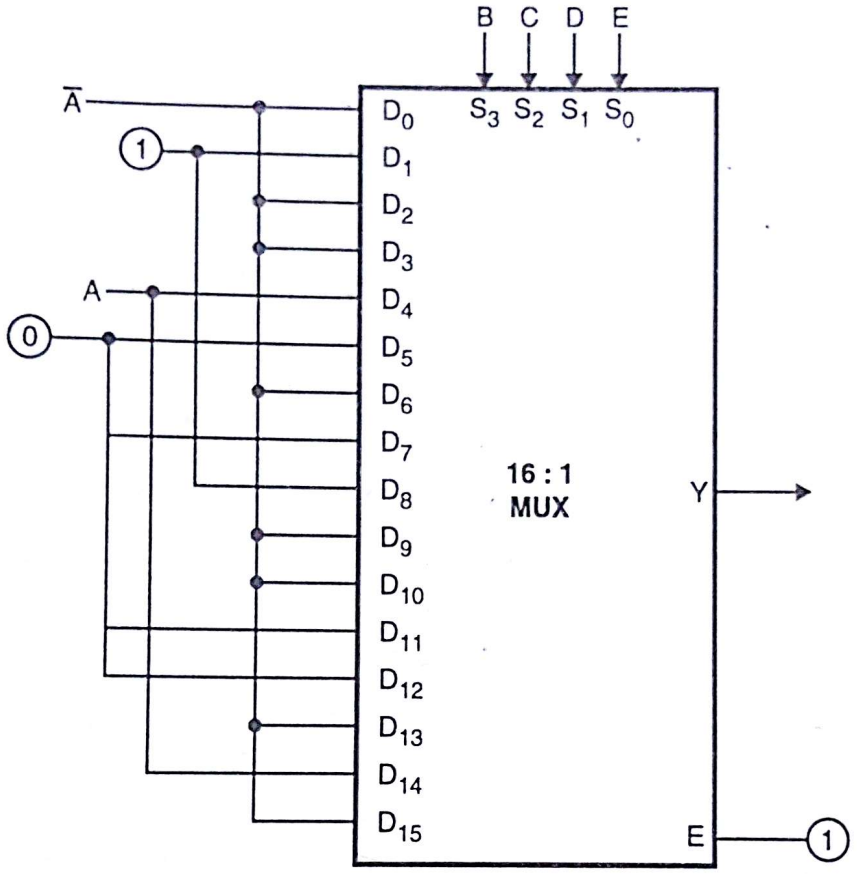



**Fig. Ex. 7.10 (c)**

**Ex. 11 :** A chemical reactor has three sensors indicating the following conditions :

(i) Pressure (P) is 'low' or 'high'. (ii) Temperature (T) is 'low' or 'high'

(iii) Liquid level (L) is 'low' or 'high

H has two controls : Heater (H) which is either ON or OFF and an inlet value (V) which is either OPEN or CLOSED. The controls are operated as per table shown.

(a) Using the convention HIGH = 1, LOW = 0, ON = 1.
OFF = 0, OPEN = 1 and CLOSED = 0, draw the Karnaugh maps for H and V.

(b) Obtain the minimal SOP and POS expressions for H and V.

(c) Realize the logic H and V using two 4 - input multiplexers with T and L as control inputs. Use T as MSB.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| P | T | L | H | V |
| LOW | LOW | LOW | OFF | OPEN |
| LOW | LOW | HIGH | ON | CLOSED |
| LOW | HIGH | LOW | OFF | OPEN |
| LOW | HIGH | HIGH | OFF | CLOSED |
| HIGH | LOW | LOW | ON | OPEN |
| HIGH | LOW | HIGH | ON | CLOSED |
| HIGH | HIGH | LOW | OFF | CLOSED |
| HIGH | HIGH | HIGH | X | X |

X system shut down.

Scanned by CamScanner

**Soln. :** Consider the truth table as shown in Table 11.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| P | T | L | H | V |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | X | X |

**Table 11**

Since there are eight input conditions and a 4 : 1 MUX, we make the inputs a function of MSB bit P. The design table is shown in Table 12.

| Inputs → | $D_0$ | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|---|
| $\bar{P}$ | 0 | ① | 2 | 3 |
| P | ④ | ⑤ | 6 | 7 |
| Final I/P to MUX | P | 1 | 0 | 0 |

**Table 12 : Table for H output**

The circuit diagram is shown in Fig. Ex. 7.11(a). Similarly for output V we have,

| Inputs → | $D_0$ | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|---|
| $\bar{P}$ | ⓪ | 1 | ② | 3 |
| P | ④ | 5 | 6 | 7 |
| Final I/P to MUX | P | 0 | $\bar{P}$ | 0 |

**Table 13 : For O/P V**

The circuit diagram is shown in Fig. Ex. 7.11 (b).

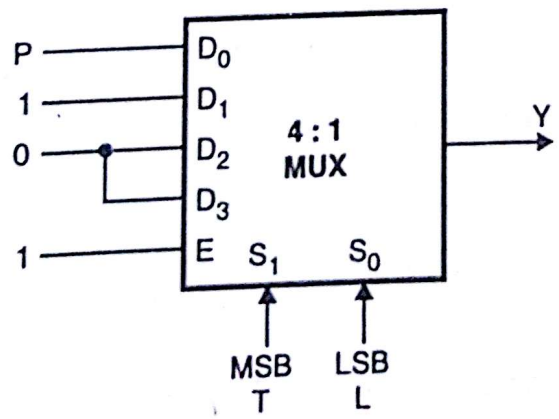



**Fig. Ex. 7.11 (a)**

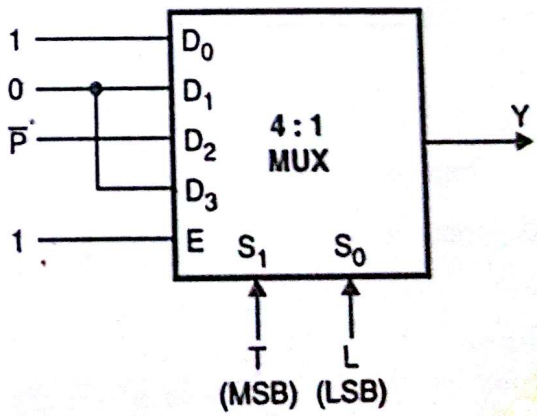



**Fig. Ex. 7.11 (b)**

Scanned by CamScanner

**Ex. 12 :** Design a circuit using 2 : 1 MUX to implement each of the following functions. Use gates and inverters if necessary.

(i) $Y = \bar{A}\,B$, use B as a select input (ii) $Y = A + \bar{B}$, use A as a select input

(iii) $Y = A \oplus B$, use B as a select input (iv) $Y = A \odot B$, use A as a select input.

**Soln. :** (i) $Y = \bar{A}B$ means O/P is logic 1 only when A = 0, B = 1, $(1)_{10}$, for all other combination O/P = 0.

Therefore, design Table 14,

| Input | $D_0$ | $D_1$ |
|---|---|---|
| $\bar{A}$ | 0 | ① |
| A | 2 | 3 |
| Final I/P to MUX | 0 | $\bar{A}$ |

**Table 14**

Circuit as shown in Fig. Ex. 7.12 (a).

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

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

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

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

∴ O/P is logic '1' when,

$A = 0 = B \rightarrow (0)_{10}$

$A = 1, B = 0 \rightarrow (2)_{10}$

$A = 1, B = 1 \rightarrow (3)_{10}$

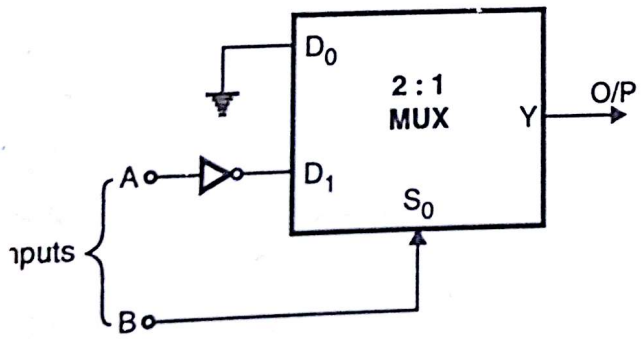



**Fig. Ex. 7.12 (a)**

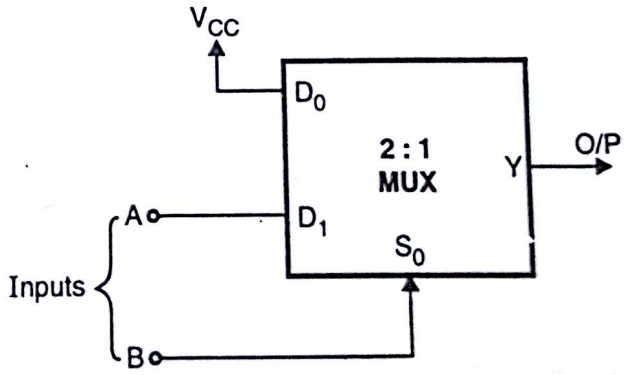



**Fig. Ex. 7.12 (b)**

∴ Design table is,

| Input | $D_0$ | $D_1$ |
|---|---|---|
| $\bar{A}$ | ⓪ | 1 |
| A | ② | ③ |
| Final I/P to MUX | $V_{CC}$ | A |

**Table 15**

Scanned by CamScanner

(iii) $Y = A \oplus B$

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

$\therefore$ O/P is logical when A = 1 B = 0 i.e. $(2)_{10}$

A = 0 B = 1 i.e. $(1)_{10}$

Design table,

| Input | $D_0$ | $D_1$ |
|---|---|---|
| $\bar{A}$ | 0 | (1) |
| A | (2) | 3 |
| Final I/P to MUX | A | $\bar{A}$ |

Table 16

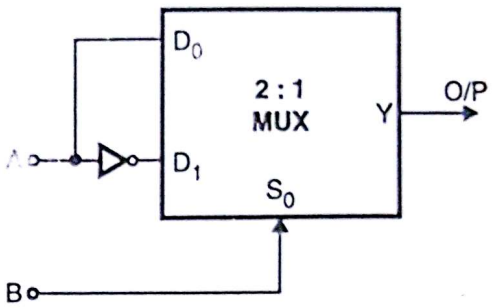



Fig. Ex. 7.12 (c)

(iv) $Y = A$ (XNOR) $B = \bar{A}\,\bar{B} + AB$

$\therefore$ O/P is logical '1' when A = B = 0 and A = B = 1. i.e. $(0)_{10}$ and $(3)_{10}$ respectively.

Design table

| Input | $D_0$ | $D_1$ |
|---|---|---|
| $\bar{A}$ | (0) | 1 |
| A | 2 | (3) |
| Final I/P to MUX | $\bar{A}$ | A |

Table 17

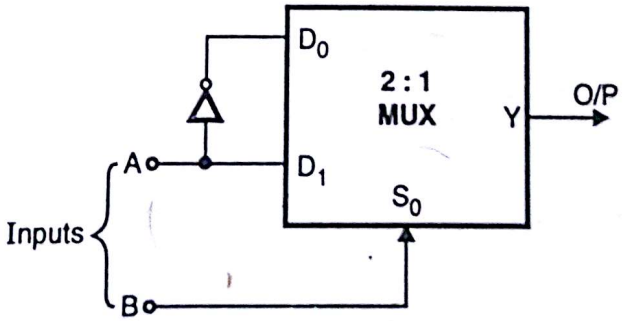



Fig. Ex. 7.12 (d)

We have solved different types of problems. Refer Ex. 7, in that we have seen that four variables function implemented using 8 : 1 MUX. In that we have implemented method where we have taken MSB as additional bit. Now we are going to learn one more method where we will select LSB line as additional line (bit).

Scanned by CamScanner

**Ex. 13 :** Design circuit using 8 : 1 MUX.

$F(A, B, C, D) = \Sigma m (2, 4, 5, 7, 10, 14)$.

**Soln. :** (1) A, B, C, D → four variables are present.

(2) Therefore no. of states will be $2^n = 2^4 = 16$.

(3) Make a table which contains binary $(0000)_2$ to $(1111)_2$.

(4) After that write down O/P Y i.e. you want 0/1. Normally you should write down logic 1 in O/P Y column, in the row specified by function i.e. 2, 4, 5, 7, 10, 14.

(5) Now go on grouping 2 states.

(6) Finally, keenly observe bit D (i.e. LSB) and O/P Y.

If Y contains '0' or '1' irrespective of D changes, then simply write down logic '0' or logic '1' respectively.

If O/P Y is same as that of D bit, write D in front of function. If O/P Y is complement of D bit, write $\bar{D}$.

Truth table is as shown in Table 18.

| Inputs | | | | Output | Function | I/P to MUX |
|---|---|---|---|---|---|---|
| A | B | C | D | Y | | |
| 0 | 0 | 0 | 0 | 0 | GND (logic '0') | $D_0$ |
| 0 | 0 | 0 | 1 | 0 | | |
| 0 | 0 | 1 | 0 | 1 | $\bar{D}$ (when D = 0, Y = 1; D = 1, Y = 0) | $D_1$ |
| 0 | 0 | 1 | 1 | 0 | | |
| 0 | 1 | 0 | 0 | 1 | $V_{CC}$ (logic 1) | $D_2$ |
| 0 | 1 | 0 | 1 | 1 | | |
| 0 | 1 | 1 | 0 | 0 | D (when D = 0, Y = 0; D = 1, Y = 1) | $D_3$ |
| 0 | 1 | 1 | 1 | 1 | | |
| 1 | 0 | 0 | 0 | 0 | logic '0', GND | $D_4$ |
| 1 | 0 | 0 | 1 | 0 | | |
| 1 | 0 | 1 | 0 | 1 | $\bar{D}$ | $D_5$ |
| 1 | 0 | 1 | 1 | 0 | | |
| 1 | 1 | 0 | 0 | 0 | GND logic '0' | $D_6$ |
| 1 | 1 | 0 | 1 | 0 | | |
| 1 | 1 | 1 | 0 | 1 | $\bar{D}$ | $D_7$ |
| 1 | 1 | 1 | 1 | 0 | | |

**Table 18**

Scanned by CamScanner

Circuit diagram is shown in Fig. Ex. 7.13 (a).

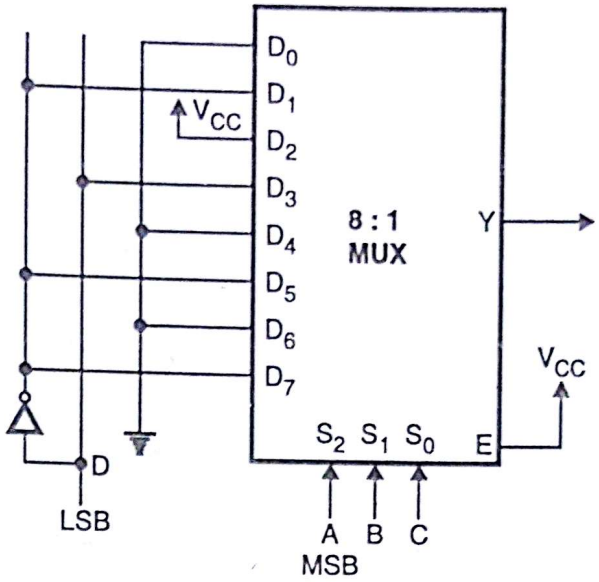



**Fig. Ex. 7.13 (a)**

**Ex. 14 :** Implement the following function by using a 8 : 1 MUX. Draw the logic diagram and explain the connection scheme.

$F = f(A, B, C, D) = \Sigma m (0, 1, 4, 8, 10, 14, 15)$.

**Soln. :**

| A | B | C | D | Y | Function | 8 : 1 MUX |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | $D_0$ |
| 0 | 0 | 0 | 1 | 1 | | |
| 0 | 0 | 1 | 0 | 0 | 0 | $D_1$ |
| 0 | 0 | 1 | 1 | 0 | | |
| 0 | 1 | 0 | 0 | 1 | $\bar{D}$ | $D_2$ |
| 0 | 1 | 0 | 1 | 0 | | |
| 0 | 1 | 1 | 0 | 0 | 0 | $D_3$ |
| 0 | 1 | 1 | 1 | 0 | | |
| 1 | 0 | 0 | 0 | 1 | $\bar{D}$ | $D_4$ |
| 1 | 0 | 0 | 1 | 0 | | |
| 1 | 0 | 1 | 0 | 1 | $\bar{D}$ | $D_5$ |
| 1 | 0 | 1 | 1 | 0 | | |
| 1 | 1 | 0 | 0 | 0 | 0 | $D_6$ |
| 1 | 1 | 0 | 1 | 0 | | |
| 1 | 1 | 1 | 0 | 1 | 1 | $D_7$ |
| 1 | 1 | 1 | 1 | 1 | | |

**Table 19**

Scanned by CamScanner

**Circuit Diagram :** Refer Fig. Ex. 7.14 (a).

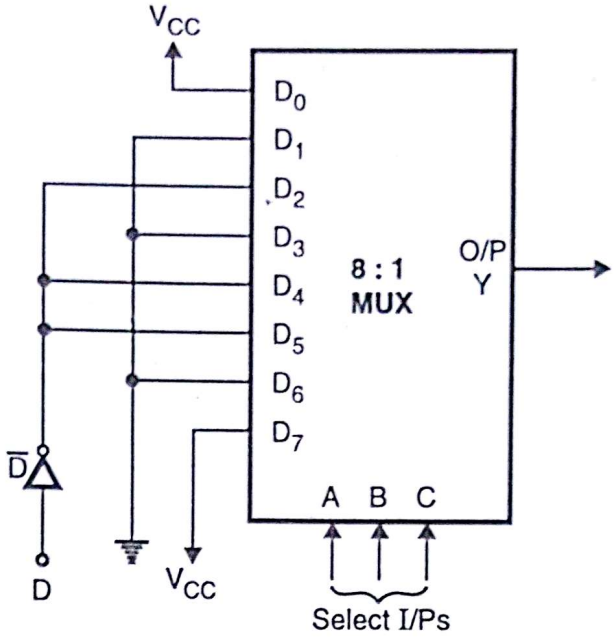



**Fig. Ex. 7.14 (a)**

**Ex. 15 :** Implement the following functions using a 8 : 1 multiplexer. Draw the logic diagram and explain the connection scheme.

$F = f(A, B, C, D) = \Sigma m (0, 1, 3, 4, 6, 7, 10, 12, 14, 15) + d (2, 5)$.

**Soln. :** In the function ***Don't Care*** conditions are introduced. As you are aware, Don't care can be treated logic '0', '1' depending upon your convenience.

$F = f(A, B, C, D) = \Sigma m (0, 1, 3, 4, 6, 7, 10, 12, 14, 15) + d (2, 5)$

| Minterm | A | B | C | D | Output | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | |
| 2 | 0 | 0 | 1 | 0 | X | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | |
| 4 | 0 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | X | |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | |
| 10 | 1 | 0 | 1 | 0 | 1 | $\overline{D}$ |
| 11 | 1 | 0 | 1 | 1 | 0 | |
| 12 | 1 | 1 | 0 | 0 | 1 | $\overline{D}$ |
| 13 | 1 | 1 | 0 | 1 | 0 | |
| 14 | 1 | 1 | 1 | 0 | 1 | 1 |
| 15 | 1 | 1 | 1 | 1 | 1 | |

**Table 20**

**Truth Table**

| A | B | C | O/P F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | $\overline{D}$ |
| 1 | 1 | 0 | $\overline{D}$ |
| 1 | 1 | 1 | 1 |

**Both 'X' are treated as logic '1'**

Scanned by CamScanner

Both 'X' are treated as logic '1'.

**Circuit diagram :** Refer Fig. Ex. 7.15 (a).

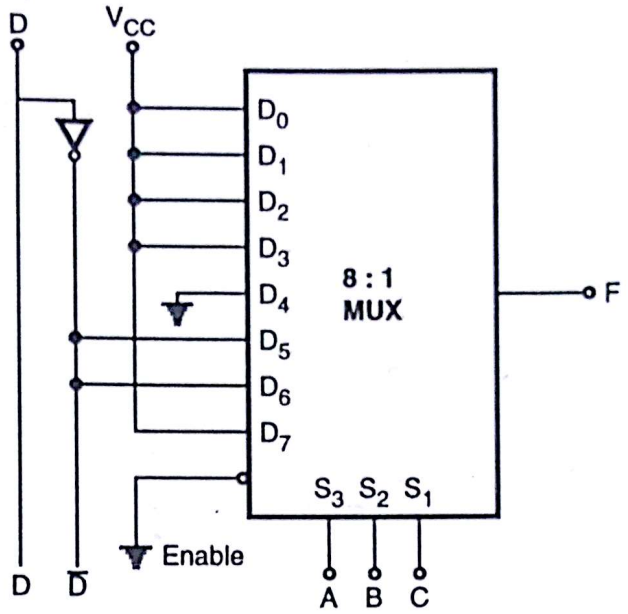



**Fig. Ex. 7.15 (a)**

**Ex. 16 :** Implement the following function by using 8 : 1 MUX,

$f = \pi M (1, 4, 5, 8, 10, 12, 15) + d (2, 7)$.

**Soln. :** Here instead of minterm, maxterm is specified. The same function can be written as,

$\Sigma m (0, 3, 6, 9, 11, 13, 14) + d (2, 7)$.

*Don't Care* can be treated logic '0' or '1'.

**Truth Table**

| Decimal | A | B | C | D | Y | | MUX I/Ps |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | $\bar{D}$ | $D_0$ |
| 1 | 0 | 0 | 0 | 1 | 0 | | |
| 2 | 0 | 0 | 1 | 0 | X | 1 | $D_1$ |
| 3 | 0 | 0 | 1 | 1 | 1 | | |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | $D_2$ |
| 5 | 0 | 1 | 0 | 1 | 0 | | |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 | $D_3$ |
| 7 | 0 | 1 | 1 | 1 | X | | |
| 8 | 1 | 0 | 0 | 0 | 0 | D | $D_4$ |
| 9 | 1 | 0 | 0 | 1 | 1 | | |
| 10 | 1 | 0 | 1 | 0 | 0 | D | $D_4$ |
| 11 | 1 | 0 | 1 | 1 | 1 | | |
| 12 | 1 | 1 | 0 | 0 | 0 | D | $D_6$ |
| 13 | 1 | 1 | 0 | 1 | 1 | | |
| 14 | 1 | 1 | 1 | 0 | 1 | $\bar{D}$ | $D_7$ |
| 15 | 1 | 1 | 1 | 1 | 0 | | |

**Table 21**

Scanned by CamScanner

Here $(2)_{10}$ minterm is treated as logic '1',
$(7)_{10}$ minterm is treated as logic '1'.

**Circuit diagram :**

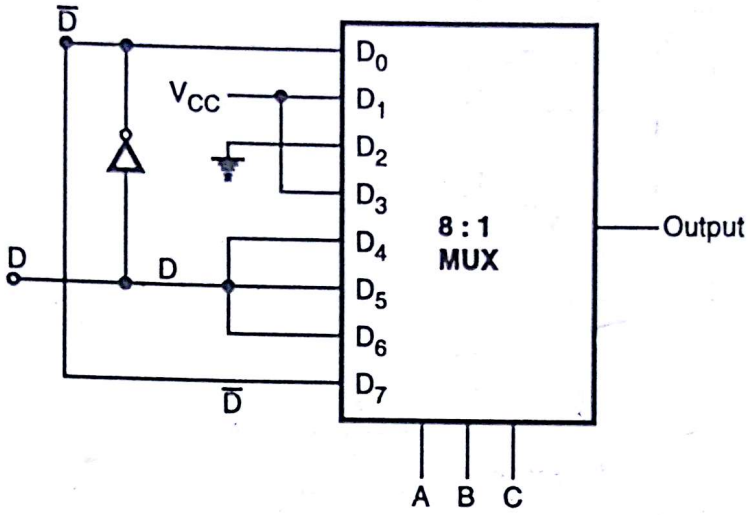



**Fig. Ex. 7.16(a)**

**Ex. 17 :** Implement function F = π M (1, 3, 5, 10, 12, 14) + d (0, 2) using 16 : 1 MUX.

**Soln. :** The same function can be written as,

$F = \Sigma m\ (4, 6, 7, 8, 9, 11, 13, 15) + d\ (0, 2)$.

| Decimal | Inputs | | | | Output |
|---|---|---|---|---|---|
| | A | B | C | D | Y |
| 0 | 0 | 0 | 0 | 0 | X |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | X |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 1 |

**Truth Table 22**

Scanned by CamScanner

**Circuit diagram :**

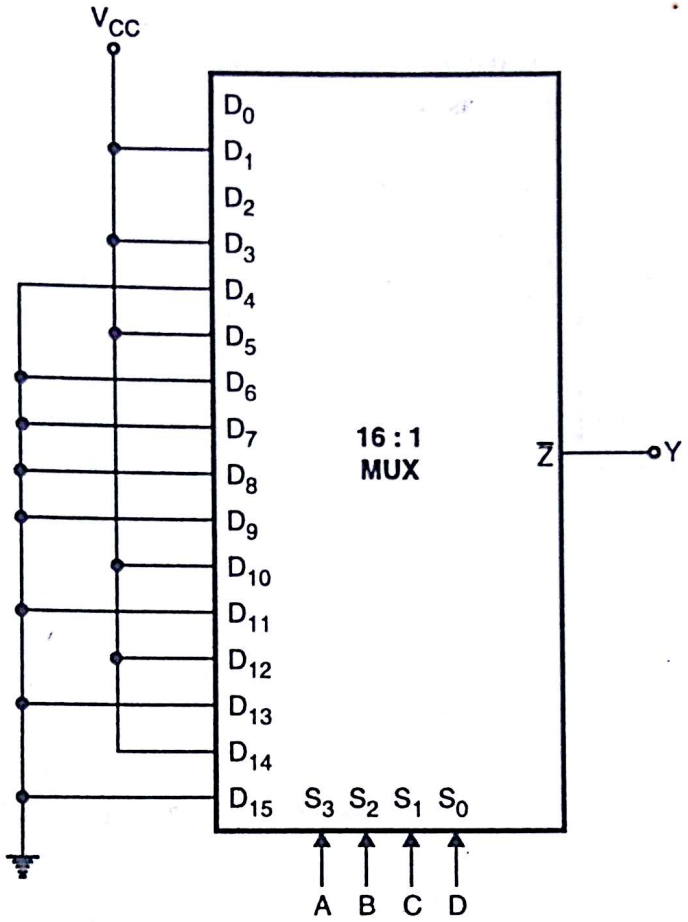



**Fig. Ex. 7.17 (a)**

As shown, 2 and 0 inputs are not connected as those terminals are ***Don't Care***.

## 7.3 Demultiplexer (Data Distributor) :

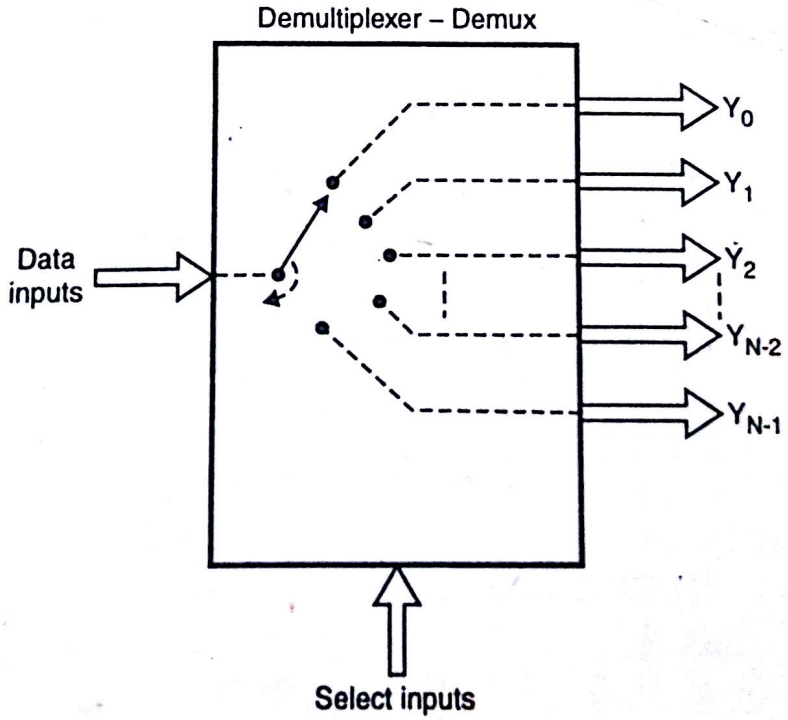



**Fig. 7.4**

Scanned by CamScanner

Demultiplexer is exactly reverse that of multiplexer i.e. it accepts single input and distributes it over several outputs. The single input should appear over which output line is decided by SELECT LINES (same as that of multiplexer). In short, the demultiplexer takes one input data source and selectively distributes it to 1 of N output channels just as that of multiposition switch. The functional representation is shown in Fig. 7.4.

The data at the input can be routed to a specific output based on select line combination. The relationship between select lines and output lines can be given by, $2^m = n$.

where m = Number of select lines, n = Number of output lines.

## Basic 2 Output Demultiplexer

Here again the basic concept is we are going to use AND gate to pass input data to ***required*** output line, when proper address (select) line status is available i.e. If we want data to appear at $Y_2$ line select line status should be $(010)_2$. Lets see Fig. 7.5.

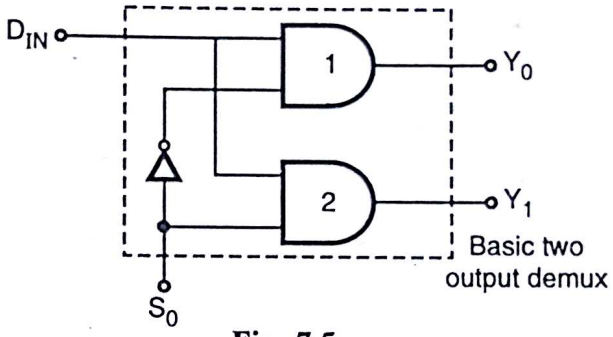



**Fig. 7.5**

when, $S_0 = 0$, AND1 is enabled and $Y_0 = D_{IN}$, $Y_1 = 0$

$S_0 = 1$, AND2 is enabled and $Y_0 = 0$, $Y_1 = D_{IN}$.

## 1 : 4 Demultiplexer

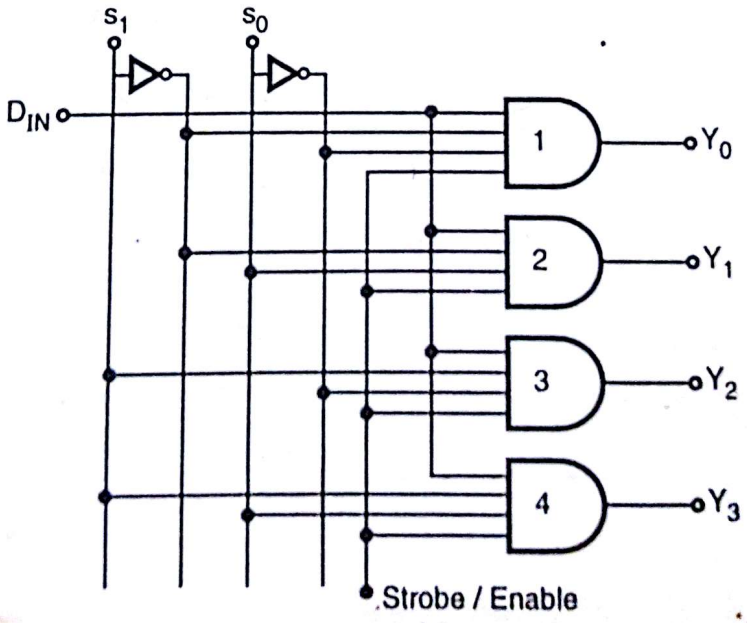



**Fig. 7.6**

Scanned by CamScanner

Fig. 7.6 shows 1 : 4 demux, $S_1$ $S_0$ = 00, AND1 is enabled. $S_1$ $S_0$ = 01 AND 2 is enabled and so on. Depending upon select line, O/P will appear at $Y_0$, $Y_1$ and so on respectively. In the circuit we have introduced 'Strobe/Enable' terminal. This is master control. If strobe = 0, $Y_0$ to $Y_3$ all will provide 'LOW' output, irrespective of $D_{IN}$ and $S_1S_0$.

| Input | | | Output |
|---|---|---|---|
| Enable | $S_1$ | $S_0$ | |
| 0 | X | X | $Y_0 = Y_1 = Y_2 = Y_3 = 0$ |
| 1 | 0 | 0 | $Y_0 = D_{IN}$ |
| 1 | 0 | 1 | $Y_1 = D_{IN}$ |
| 1 | 1 | 0 | $Y_2 = D_{IN}$ |
| 1 | 1 | 1 | $Y_3 = D_{IN}$ |

**Truth Table 1**

Typical Demux chips are :

| Chips | Function |
|---|---|
| 74154 | 1 : 16 Demux |
| 74155 | Dual 2 : 4 Demux |
| 74156 | Dual 2 : 4 Demux |

**Ex. 18 :** Cascade Demultiplexer. Built 1 : 8 demux using 1 : 4 demux chips.

**Soln. :** We are going to use 1 : 4 demux with 'enable' terminal. Refer Fig. Ex. 7.18 (a).

A, B, C, three variables are used, where A = MSB and C = LSB. B and C variables are given to select lines $S_1$ and $S_0$ respectively. A is directly given to lower demux and $\bar{A}$ is given to upper demux. '$D_{IN}$' of both the demux is tide together.

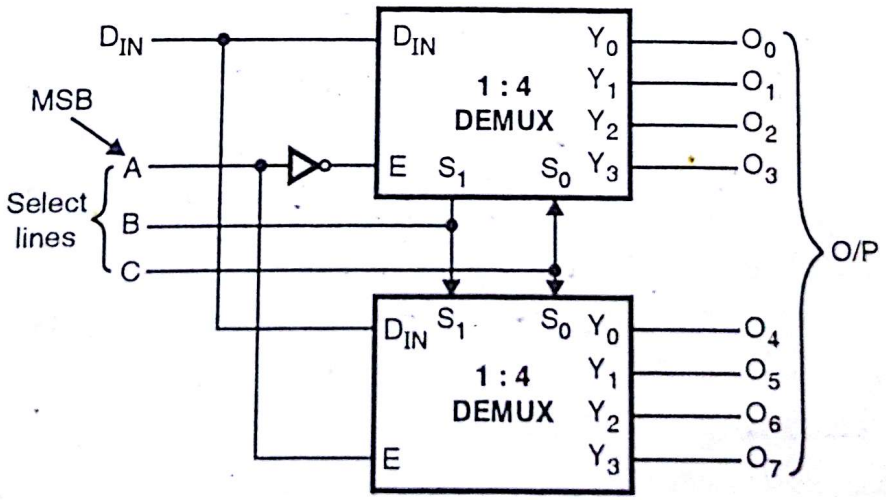



**Fig. Ex. 7.18 (a)**

**Ex. 19 :** Implement full adder using demultiplexer. **(Dec. 98, 6 Marks)**

**Soln. :** Refer Table 2 for truth table of full adder circuit.

Scanned by CamScanner

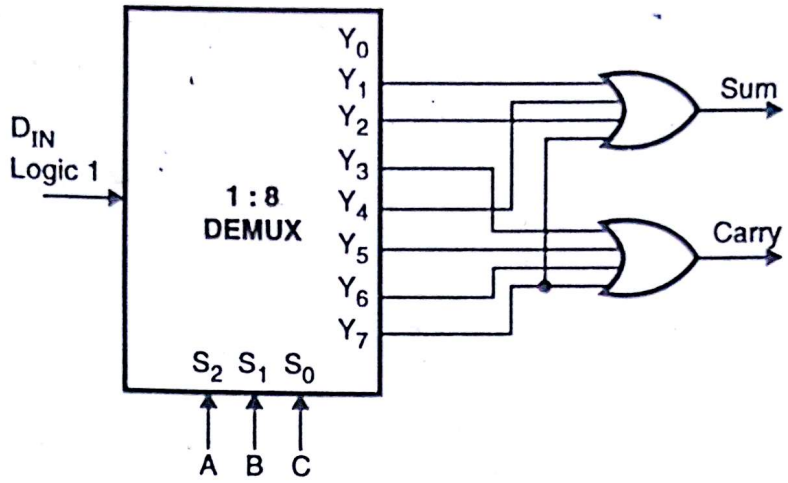



**Fig. Ex. 7.19 (a)**

Sum function can be written as,

$$S = f(A, B, C) = \Sigma m (1, 2, 4, 7)$$

$\therefore$ $Y_1$, $Y_2$, $Y_4$ and $Y_7$ is selected.

For carry

$$C = f(A, B, C) = \Sigma m (3, 5, 6, 7)$$

$\therefore$ $Y_3$, $Y_5$, $Y_6$ and $Y_7$ are selected.

---

**Ex. 20 :** Implement the following functions using suitable demux and logic gates.

$Y_1 = f_1 (A, B, C) = \Sigma m (0, 1, 3, 5, 6)$, $Y_2 = f_2 (A, B, C) = \Sigma m (0, 1, 2, 4, 6)$

$Y_3 = f_3 (A, B, C) = \Sigma m (1, 2, 3, 6)$. Draw the logic diagram.

**Soln. :** (1) We will use 1 : 8 demultiplexer.

**(2) Circuit diagram :**

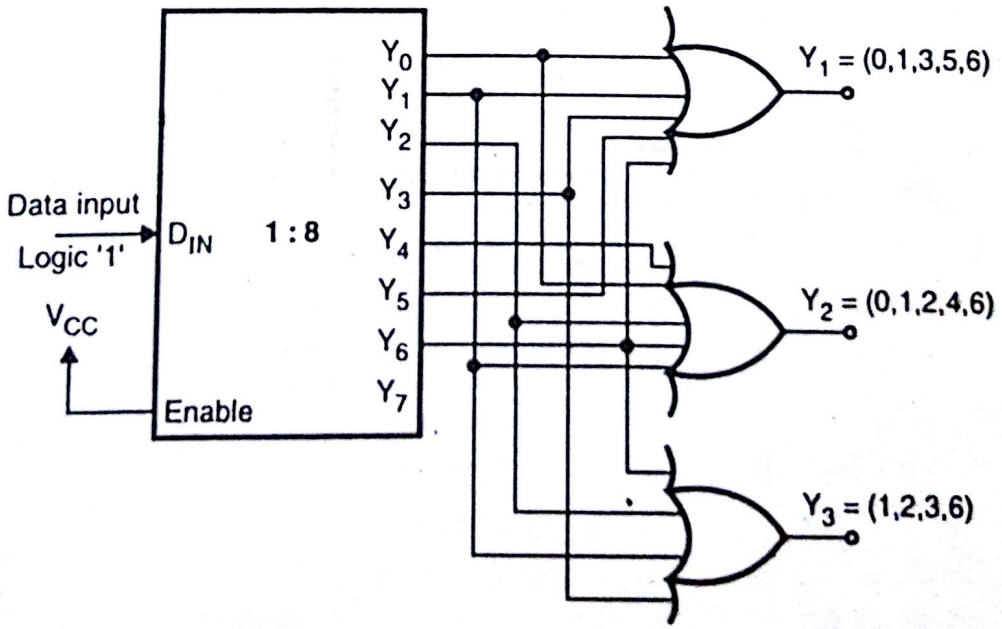



**Fig. Ex. 7.20 (a)**

Scanned by CamScanner

$Y_1$ will be logic '1' whenever minterm 0, 1, 3, 5, 6 is selected. For remaining minterm O/P will be zero. Same way $Y_2$ will be logical '1' when minterm 0, 1, 2, 4, 6, is selected. $Y_3$ will be logic '1' when 1, 2, 3, 6 is selected.

**Ex. 21 :** Implement the following multiple output function using suitable Demux.

$f_1 (A, B, C) = \Sigma (0, 1, 3, 7) + d (2, 5)$,

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

$f_3 (A, B, C) = \Sigma m (0, 2, 4, 6)$.

**Soln. :**

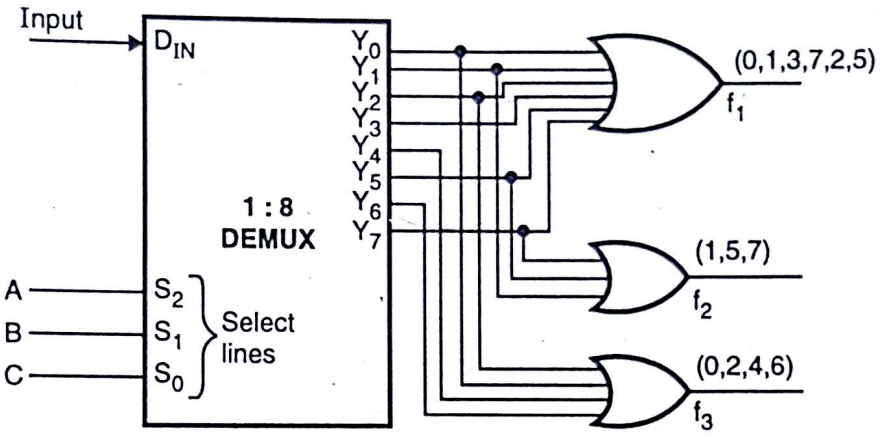



**Fig. Ex. 7.21 (a)**

## 7.4 Decoder :

A decoder is a digital logic circuit that converts N-bit binary input code into M output lines. Here each output line will be activated for only one of the possible combination of inputs. Fig. 7.7 shows generalised block schematic of decoder logic.

The number of input lines are N and O/P lines are M. In decoder N can be any integer and M is an integer that is less than or equal to $2^N$. $M \leq 2^N$ because $2^N$ possible codes are there, one of the M lines will be activated depending upon input combination or codes. All other lines will be deactivated. Active logic can be LOW or HIGH, depends upon chip.

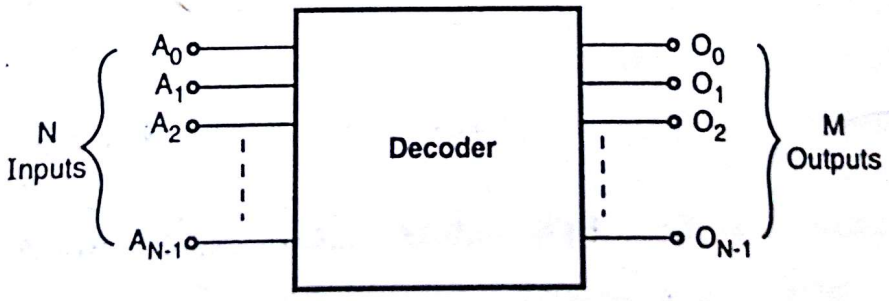



**Fig. 7.7**

In some of the decoders all $2^N$ possible combinations are not used. But only certain are used. If unused codes are applied, none of the output will be activated. We will study this type of decoder i.e. BCD to decimal decoder.

Decoder can be referred in several ways :

Scanned by CamScanner

(1) Say for example, 3 line to 8 line decoder, means it accepts three lines and gives out eight output line.

(2) It is also called as ***1 of 8 decoder*** (or 1 : 8 demux) because only 1 O/P will be activated out of 8 outputs at a time.

(3) It is also called as binary to octal decoder because it takes 3 bit binary input code and activates one of the eight (octal) outputs corresponding to the code.

**Comparison between Demux and Decoder :**

If we compare demux and decoder then decoder doesn't have $D_{IN}$ (data input) line, where demux has. Let's see 2 line to 4 line decoder.

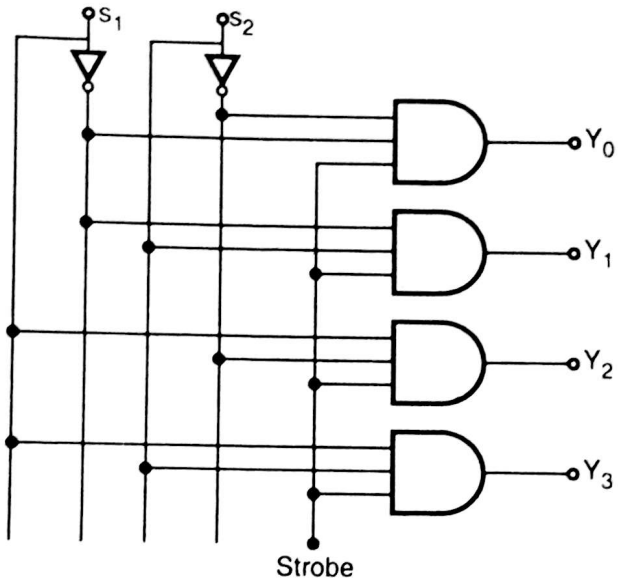



**Fig. 7.8**

Compare Figs. 7.6 and 7.8, $D_{IN}$ is not present. Remaining circuit is as it is.

**Ex. 22 :** Implement the following using 4 : 16 decoder

(i) $F_1 = \Sigma\, m\,(1, 2, 4, 7, 8, 11, 12, 13)$

(ii) $F_2 = \pi m\,(10, 12, 13, 14)$

(iii) $F_3 = \Sigma m\,(1, 5, 8, 10)$.

**Soln. :** A 4 : 16 decoder has four select lines i.e. four variables C and D. We may rewrite all the function as follows :

$$F_1 = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + AB\bar{C}\bar{D} + AB\bar{C}D$$

$$F_2 = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + A\bar{B}\bar{C}D + A\bar{B}CD$$

$$F_3 = A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D + ABC\bar{D}$$

$$F_4 = \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D}$$

**The circuit diagram is shown in Fig. Ex. 7.22(a).**

Scanned by CamScanner

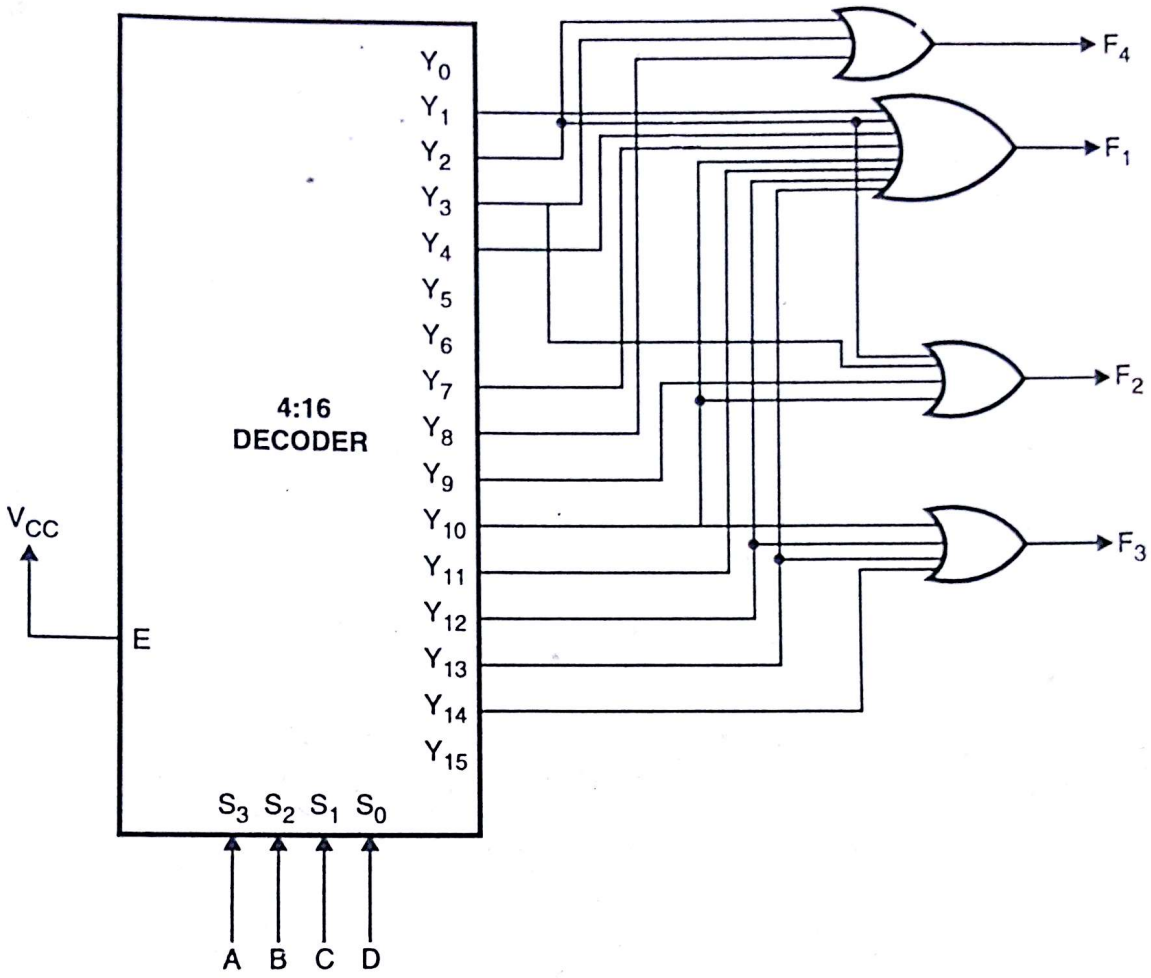



**Fig. Ex. 7.22(a)**

**Ex. 23 :** Draw a logic diagram of a 2 : 4 line decoder / demux using NOR gates only.

**Soln. :** Fig. Ex. 7.23(a) shows the required circuit diagram.

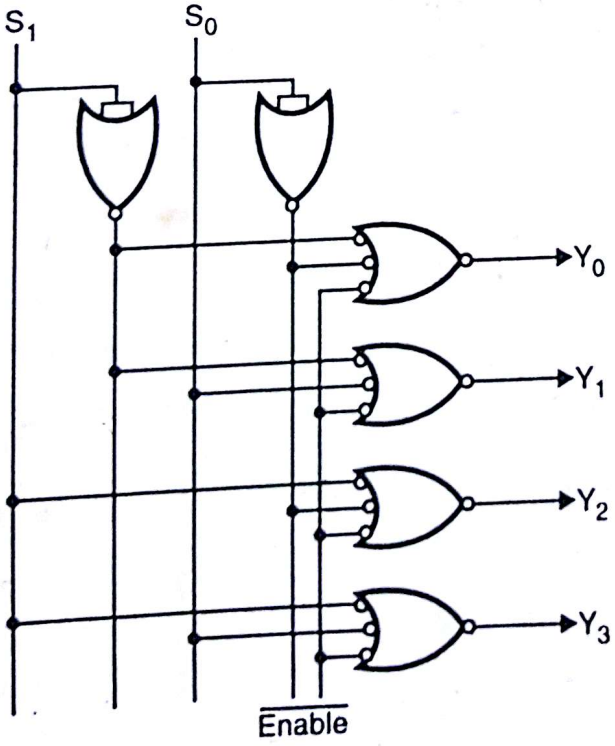



**Fig. Ex. 7.23(a)**

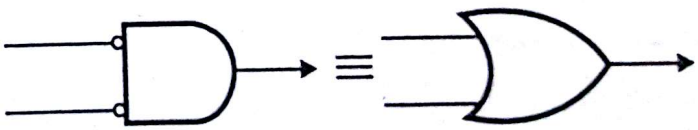


**Fig. Ex. 7.23(b)**

Scanned by CamScanner

Now we will study IC 74138.

**74138**

3 : 8 line decoder OR 1 of 8 Decoder/Demultiplexer.

74138 is most widely used 3 : 8 line decoder in microprocessor or microcomputer based system. 74138 is high speed 1 of 8 decoder/demultiplexer. Let's see pin configuration, and logic symbol in Fig. 7.9.

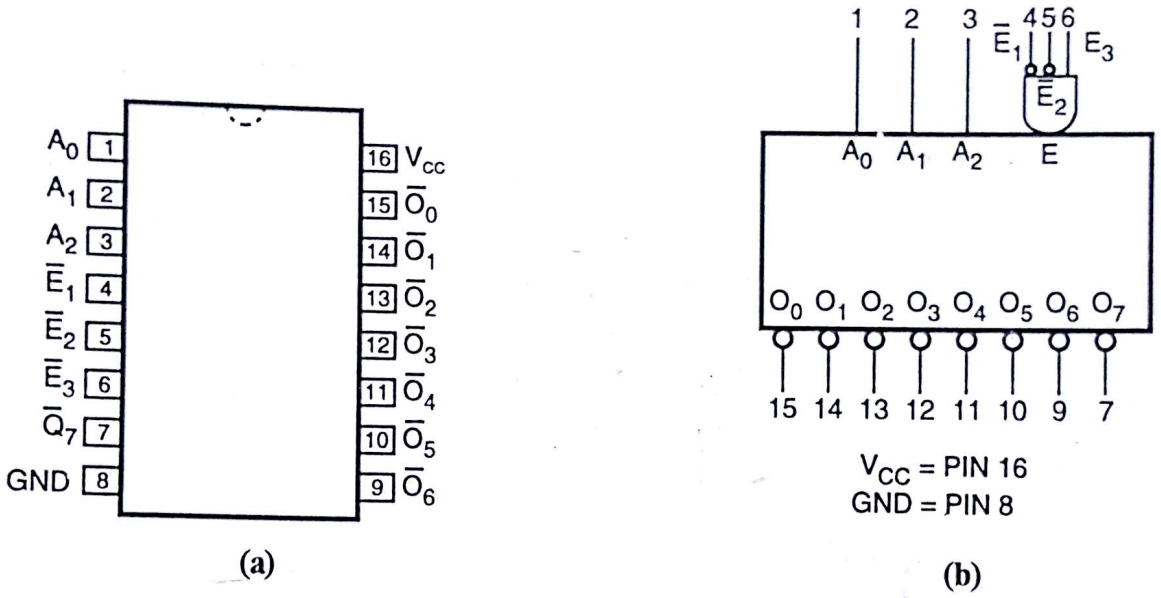



**Fig. 7.9**

**Pin Names and Description**

| Pin Names | Description |
|---|---|
| $A_0 - A_2$ | Address inputs (Select lines) |
| $\bar{E}_1 - \bar{E}_2$ | Enable inputs (Active Low) |
| $E_3$ | Enable Input (Active HIGH) |
| $\bar{O}_0 - \bar{O}_7$ | Outputs (Active Low) |

**Table 7.2**

74138 chip is 16 pin chip with $A_0 - A_2$ select lines to select required output line to be active. $\bar{E}_1$ to $\bar{E}_2$, two inputs are active low enable inputs. Make $\bar{E}_1 = \bar{E}_2 = 0$ to enable the chip. $E_3$ is Active HIGH enable input. Make $E_3 = 1$ to enable the chip. Finally $\bar{O}_0 = \bar{O}_7$ are, eight active low outputs present.

**Features :**

(1) Schottky process for high speed.
(2) Demultiplexing capability.
(3) Multiple input enable for easy expansion.
(4) Active LOW mutually exclusive outputs.

To understand the working of functional diagram, let's see truth table of the chip.

Scanned by CamScanner

**Truth Table**

| Inputs | | | | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bar{E}_1$ | $\bar{E}_2$ | $E_3$ | $A_0$ | $A_1$ | $A_2$ | $\bar{O}_0$ | $\bar{O}_1$ | $\bar{O}_2$ | $\bar{O}_3$ | $\bar{O}_4$ | $\bar{O}_5$ | $\bar{O}_6$ | $\bar{O}_7$ |
| H | X | X | X | X | X | H | H | H | H | H | H | H | H |
| X | H | X | X | X | X | H | H | H | H | H | H | H | H |
| X | X | L | X | X | X | H | H | H | H | H | H | H | H |
| L | L | H | L | L | L | L | H | H | H | H | H | H | H |
| L | L | H | H | L | L | H | L | H | H | H | H | H | H |
| L | L | H | L | H | L | H | H | L | H | H | H | H | H |
| L | L | H | H | H | L | H | H | H | L | H | H | H | H |
| L | L | H | L | L | H | H | H | H | H | L | H | H | H |
| L | L | H | H | L | H | H | H | H | H | H | L | H | H |
| L | L | H | L | H | H | H | H | H | H | H | H | L | H |
| L | L | H | H | H | H | H | H | H | H | H | H | H | L |

**Table 7.3**

$A_0$ = LSB, $A_2$ = MSB, H = HIGH Voltage Level, L = LOW Voltage Level

X = Immaterial

**Ex. 24 :** Design full adder using decoder logic.

**Soln. :** We have already studied full adder truth table.

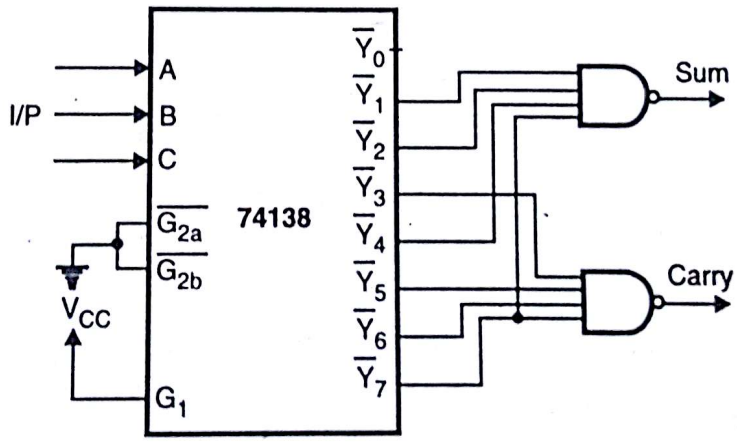



**Fig. Ex. 7.24 (a)**

## 7.5 ALU (Arithmetic Logic Unit) : [ASKED IN EXAM : Dec 96, May 97]

Arithmetic logic unit is very powerful device, able to perform different arithmetic and logical functions. Basically ALU is the heart of central processing unit (CPU) of the computer. It is capability of ALU which allows you to perfom different functions.

We are going to first study popular and most widely used IC 74181. The Fig. 7.10 shows pin configuration of IC 74181.

Scanned by CamScanner

Let's see description of pin configuration.

| Pin Names | Description |
|---|---|
| $\overline{A}_0 - \overline{A}_3$ | Operand Inputs (Active LOW) |
| $\overline{B}_0 - \overline{B}_3$ | Operand Inputs (Active LOW) |
| $S_0 - S_3$ | Function Select Inputs |
| M | Mode Control Input |
| $C_n$ | Carry Input |
| $\overline{F}_0 - \overline{F}_3$ | Function Outputs (Active LOW) |
| A = B | Comparator Output |
| $\overline{G}$ | Carry Generate Output (Active LOW) |
| $\overline{P}$ | Carry Propagate Output (Active LOW) |
| $C_{n+4}$ | Carry Output |

**Table 7.4**

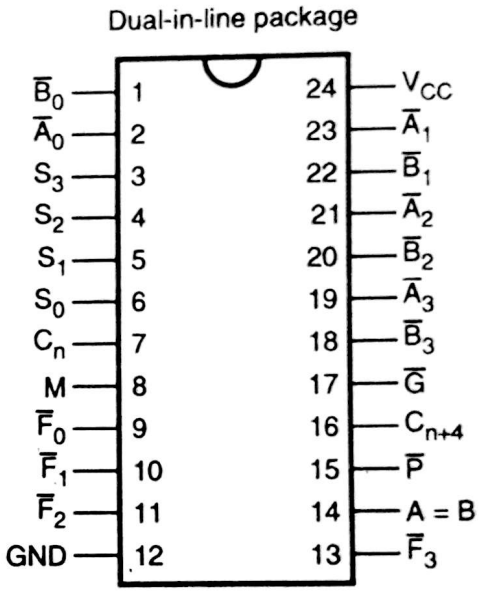



**Fig. 7.10**

Basically 74181 is 4 bit ALU which can perform all the possible 16 logic operations on two varibales and a variety of arithmetic operations.

**Features of the chip are :**

- Provides 16 arithmetic operations, add, subtract, compare, double, plus twelve other arithmetic functions.
- Provides all 16 logic operations of two variables. Exclusive – OR, compare, AND, NAND, OR, NOR, plus ten other logic operations.
- Full lookahead for high speed arithmetic operation on long word.

## Functional Description :

The 74181 is a 4 bit high speed parallel arithmetic logic unit (ALU). Controlled by the four function select inputs ($S_0 - S_3$) and mode control input M, it can perform all the 10 possible logic operations or 16 different arithmetic operations on active HIGH or active LOW operands. The Fig. 7.11 shows logic symbols for active high and active low operands.

If you compare Figs. 7.11 (a) and (b), you will find that bubbles (shows inversion) have been introduced of input operand $A_0 - A_3$ and $B_0 - B_3$ as well as at G and P. The bubble has been removed from $C_n$ and $C_{n+1}$. The function table shows all the operations :

Scanned by CamScanner

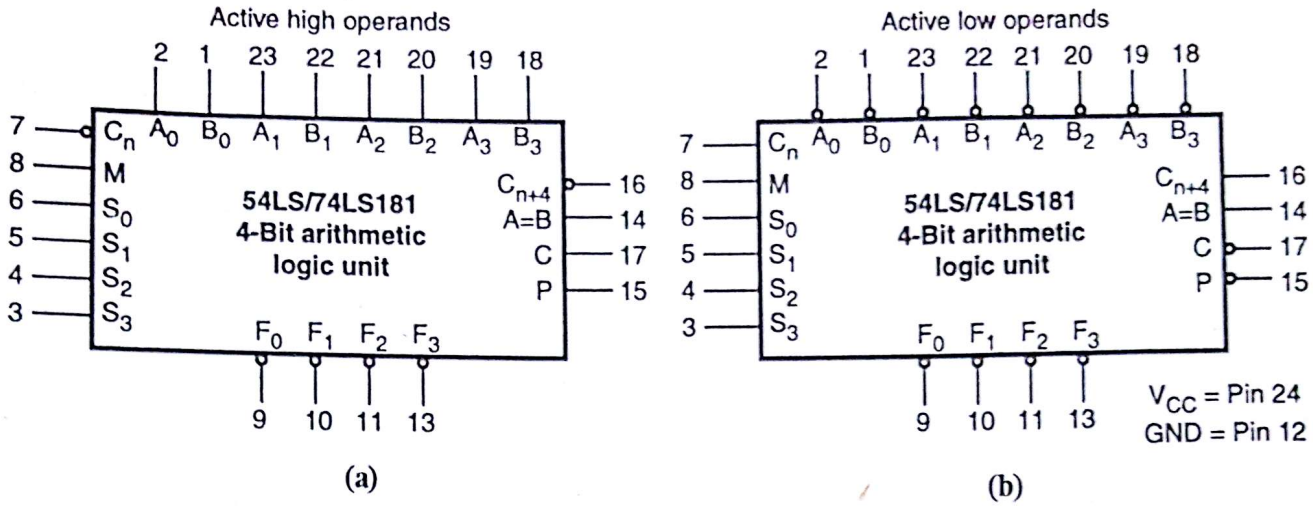



**Fig. 7.11**

**Function Table**

| Mode Select Inputs | | | | Active LOW Operands and $F_n$ Outputs | | Active HIGH Operands and $F_n$ Outputs | |
|---|---|---|---|---|---|---|---|
| $S_3$ | $S_2$ | $S_1$ | $S_0$ | Logic (M = H) | Arithmetic (M = L) ($C_n$ = L) | Logic (M = H) | Arithmetic (M = L) ($C_n$ = H) |
| L | L | L | L | $\overline{A}$ | A minus 1 | $\overline{A}$ | A |
| L | L | L | H | $\overline{AB}$ | AB minus 1 | $\overline{A+B}$ | A + B |
| L | L | H | L | $\overline{A+B}$ | $A\overline{B}$ minus 1 | $\overline{A}B$ | $A+\overline{B}$ |
| L | L | H | H | Logic 1 | minus 1 | Logic 0 | minus 1 |
| L | H | L | L | $\overline{A+B}$ | A plus $(A+\overline{B})$ | $\overline{AB}$ | A plus $A\overline{B}$ |
| L | H | L | H | $\overline{B}$ | AB plus $(A+\overline{B})$ | $\overline{B}$ | $(A+B)$ plus $A\overline{B}$ |
| L | H | H | L | $\overline{A \oplus B}$ | A minus B – 1 | $A \oplus B$ | minus 1 |
| L | H | H | H | $A+\overline{B}$ | $A+\overline{B}$ | $A\overline{B}$ | AB minus 1 |
| H | L | L | L | $\overline{A}B$ | A plus (A + B) | $\overline{A+B}$ | A plus AB |
| H | L | L | H | $A \oplus B$ | A plus B | $\overline{A \oplus B}$ | A plus B |
| H | L | H | L | B | $A\overline{B}$ plus (A + B) | B | (A + B) plus AB |
| H | L | H | H | A + B | A + B | AB | AB minus 1 |
| H | H | L | L | Logic 0 | A plus A* | Logic 1 | A plus A* |
| H | H | L | H | $A\overline{B}$ | AB plus A | $A+\overline{B}$ | (A + B) plus A |
| H | H | H | L | AB | $A\overline{B}$ minus A | A + B | $(A+\overline{B})$ plus A |
| H | H | H | H | A | A | A | A minus 1 |

*Each bit is shifted to the next move significant position.

**Table 7.5**

Scanned by CamScanner

As shown in Table 7.5, when mode control input (M) is HIGH, all internal carries are inhibited and device performs logical operations on individual bits as listed.

When mode control input is LOW, the carries are enabled and the device performs arithmetic operations on two, 4 bit number. The Table 7.5 shows $C_n = 0$ when $M = 0$ (active low operands) and $M = 0$ $C_n = 0$ (Active high operands). Now we will see what will be the change when $C_n = 0$ when $H = 0$ (active HIGH). The table is shown in Fig. 7.11(A).

| Section $S_3$ $S_2$ $S_1$ $S_0$ | Active High Data | | |
|---|---|---|---|
| | M = 1 Logic | M = 0 Arithmetic Operations | |
| | Functions | $\bar{C}_n = 1$ | $\bar{C}_n = 0$ |
| 0 0 0 0 | $F = \bar{A}$ | F = A | F = A PLUS 1 |
| 0 0 0 1 | $F = \overline{A+B}$ | F = A + B | F = (A + B) PLUS 1 |
| 0 0 1 0 | $F = \bar{A} \cdot B$ | F = $A+\bar{B}$ | F = $(A+\bar{B})$ PLUS 1 |
| 0 0 1 1 | F = 0 | F = MINUS 1 | F = ZERO |
| 0 1 0 1 | $F = \bar{B}$ | F = (A + B) PLUS $A\bar{B}$ | F = (A+B) PLUS $A\bar{B}$ PLUS 1 |
| 0 1 0 0 | $F = \overline{AB}$ | F = A PLUS $A\bar{B}$ | F = A PLUS $A\bar{B}$ PLUS 1 |
| 0 1 1 0 | $F = A \oplus B$ | F = A MINUS B MINUS 1 | F = A MINUS B |
| 0 1 1 1 | $F = A \cdot \bar{B}$ | F = $A \cdot \bar{B}$ MINUS 1 | F = $A\bar{B}$ |
| 1 0 0 0 | $F = \bar{A}+B$ | F = A PLUS AB | F = A PLUS AB PLUS 1 |
| 1 0 0 1 | $F = \overline{A \oplus B}$ | F = A PLUS B | F = A PLUS B PLUS 1 |
| 1 0 1 0 | F = B | F = $(A+\bar{B})$ PLUS AB | F = $(A+\bar{B})$ PLUS AB PLUS 1 |
| 1 0 1 1 | F = A · B | F = AB MINUS 1 | F = AB |
| 1 1 0 0 | F = 1 | F = A PLUS A | F = A PLUS A PLUS 1 |
| 1 1 0 1 | F = $A+\bar{B}$ | F = (A + B) PLUS A | F = (A + B) PLUS A PLUS 1 |
| 1 1 1 0 | F = A + B | F = $(A+\bar{B})$ PLUS A | F = $(A+\bar{B})$ PLUS A PLUS 1 |
| 1 1 1 1 | F = A | F = A MINUS 1 | F = A |

**Fig. 7.11(A)**

Basically carry in and carry out lines are provided for cascading 'N' number of 74181. The device provides full internal carry lookahead and provides for either ripple carry between devices using $C_{n+4}$ output, or for carry lookahead between packages using signal $\bar{P}$ (carry propagate) and $\bar{G}$ (carry generate).

When speed requirements are not stringent, it can be used in simple ripple carry mode by connecting carry out ($C_{n+4}$) signal to carry input ($C_n$) of the next unit. To understand this we will see one example.

Scanned by CamScanner

**Ex. 25 :** Design 8 bit ALU circuit using 74181.

**Soln. :** The example asks for cascading IC 74181. The Fig. Ex. 7.25 (a) is as follows.

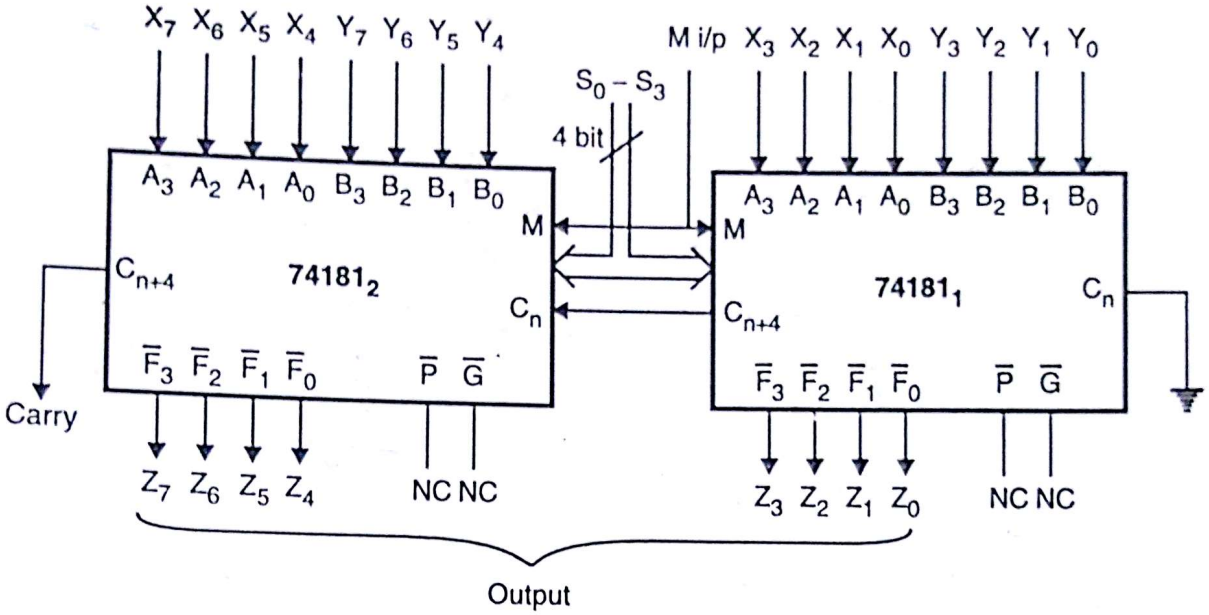



**Fig. Ex. 7.25(a)**

As shown mode, control (M) and select inputs ($S_3 - S_0$) lines are tied together commonly, so both IC's operates in same mode. Input operand 1 = $X_7 - X_0$ and operand 2 = $Y_7 - Y_0$ are connected to $A_3$ $A_2$ $A_1$ $A_0$ and $B_3$ $B_2$ $B_1$ $B_0$ of two 74181, respectively $C_n$ of $74181_1$ is ground. $C_{n+4}$ of $74181_1$ is connected to $74181_2$ i.e. carry of one stage propagated to next stage. The final carry output $C_{n+4}$ is generated from $74181_2$. Now apply operand 1 and 2. Select required function and take output.

**$\bar{P}$ (carry propagate) and $\bar{G}$ (carry generate) outputs :** In the ADD mode, $\bar{P}$ indicates that output ($\bar{F}_3 - \bar{F}_0$) is 15 or more, while $\bar{G}$ indicates that $\bar{F}_3 - \bar{F}_0$ output is 16 or more. In the subtract mode, $\bar{P}$ indicates $\bar{F}$ is zero or less, while $\bar{G}$ indicates that $\bar{F}$ is less than zero.

$\bar{P}$ and $\bar{Q}$ are not affected by carry. When high speed operation is required 74181 is used with 9342 (carry lookahead generator). Four 74181 device can be interfaced to single 9342.

**(A = B) output line :**

**Case I :** (A = B) output goes HIGH when all output $F_3 - F_0$ are HIGH.

**Case II :** (A = B) output is HIGH to indicate logic equivalence over four bits when the circuit is in subtract mode. (A = B) output is open collector and can be WIRED AND with other A = B output of 74181 to give comparision for more than 4 bits. The A = B signal can be used with $C_{n+4}$ signal to indicate A>B or A<B.

---

**Ex. 26 :** Find the expression for $F_i$ and $C_{i+1}$ in terms of $A_i$, $B_i$ and $C_i$ for the circuit of simple ALU shown in Fig. Ex. 7.26. M = mode control and $S_1$ and $S_2$ are function select bits.

Scanned by CamScanner

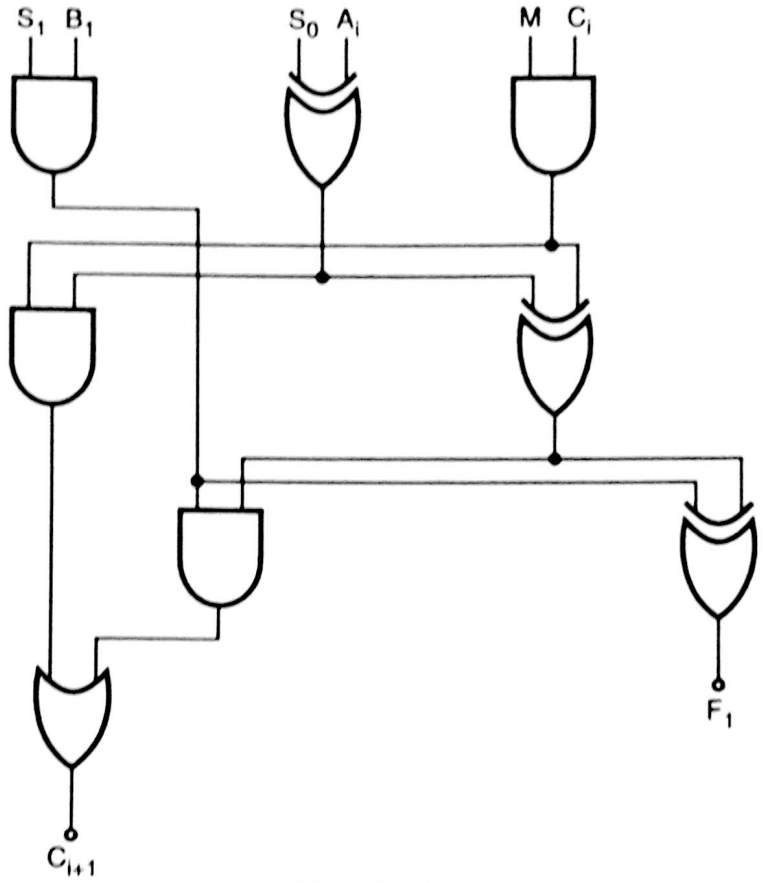



**Fig. Ex. 7.26(a)**

**Soln. :**

$$F_i = S_i B_i \oplus \left[S_0 \oplus A_i + M C_i\right]$$

$$C_{i+1} = MC_i (S_0 \oplus A_i) + S_i B_i \left[(MC_i) \oplus S_i \oplus A_i\right]$$

## 7.6 Encoders :

In decoder N-bit input code produces HIGH (or LOW) at the output at only one output line. In short, decoder recognizes an input code and accordingly responds.

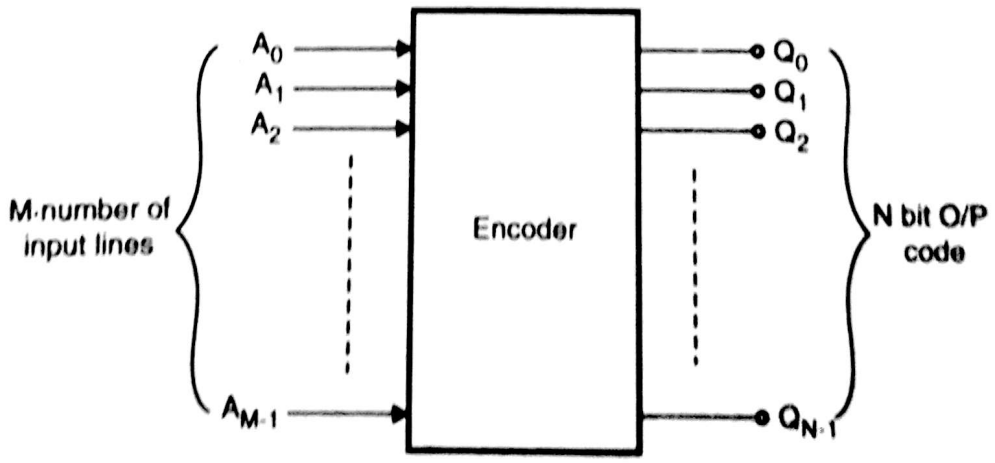



**Fig. 7.12**

**The encoder performs exactly reverse operation, the operation is called as *encoding* and digital circuit built to perform this operation is called as on *encoder*. An encoder has M input lines and N output lines**

Scanned by CamScanner

Out of M input lines only one is activated at a time and it produces equivalent code on output N lines. The block schematic is as shown in Fig. 7.12.

For example, suppose we have octal to binary encoder, then input lines will be 8 and 3 O/P lines. Out of eight line only one will be activated at a time and we get 3 bit output code corresponding to activated input. Say for e.g. input lines are $I_0 - I_7$ and output lines are $Q_2 - Q_0$. If $I_1 = 1$, $I_0 = I_2 = I_3 = I_4 = I_5 = I_6 = I_7 = 0$ then $Q_2\ Q_1\ Q_0\ (001)_2$ , where $(001)_2 = (1)_{10}$.

Same way if $I_6 = 1$ then $Q_2\ Q_1\ Q_0 = (110)_2$.

The truth table can be shown as in Table 7.6 :

| Input | | | | | | | | Output | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $I_0$ | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ | $I_7$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**Table 7.6 : Truth Table**

Now here you may ask question that if two or more inputs are held HIGH at the sametime, how the circuit will behave ? Here answer is if two or more lines are held HIGH at a time, depending upon line, combination, O/P will appear.

For example, suppose we make $I_2$ and $I_4$ HIGH. Because of $I_2$ , $Q_2\ Q_1\ Q_0 = 010$ and because of $I_4$, $Q_2\ Q_1\ Q_0 = 100$

$\therefore$ Final O/P is $010 + 000 \rightarrow 110$

$\downarrow$

logical OR

But, this output is not expected. The remedy over this problem is priority encoders.

## 7.6.1 Priority Encoders :

In previous case we saw that when two or more lines are activated, the output code you get is invalid. Therefore we have to modify the circuit. The modification is such that we have to define the ***priority*** of the given number. It means that whenever two ore more inputs are applied at a time, internal hardware will check this condition and if the 'priority' is set such that higher-numbered input should be taken into account and remaining are considered as don't care then output code which will appear will be of ***higher numbered input***. For example, say $I_3$ to $I_6$ both are activated at a time, then $I_6$ code will appear at $Q_2\ Q_1\ Q_0$ because 6 is greater than 2.

Available encoders in chip form are.

| | |
|---|---|
| 74147 | 10 : 4 priority encoder |
| 74148 | Octal to binary priority encoder |

Scanned by CamScanner

In 74147 is basically decimal to BCD priority encoder. The logic symbol and truth table for 74147 is shown in Fig. 7.13.

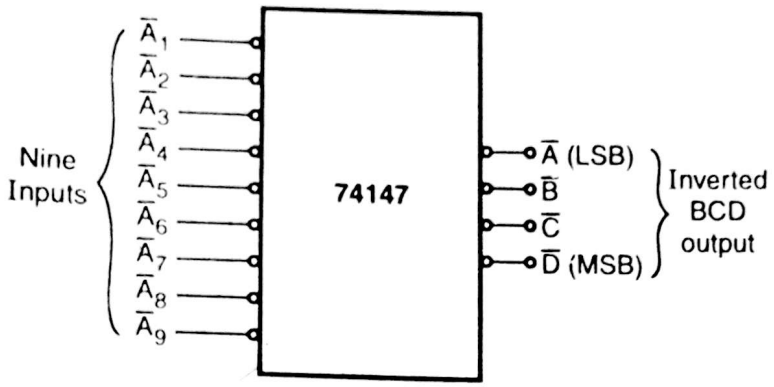



Fig. 7.13(a)

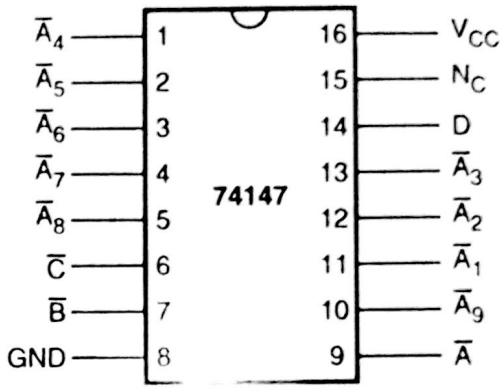



Fig. 7.13 (b)

| Inputs | | | | | | | | | Outputs (Inverted BCD) | | | | Normal BCD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bar{A}_1$ | $\bar{A}_2$ | $\bar{A}_3$ | $\bar{A}_4$ | $\bar{A}_5$ | $\bar{A}_6$ | $\bar{A}_7$ | $\bar{A}_8$ | $\bar{A}_9$ | D | C | B | A | D | C | B | A |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| X | X | X | X | X | X | X | X | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| X | X | X | X | X | X | X | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| X | X | X | X | X | X | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| X | X | X | X | X | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| X | X | X | X | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| X | X | X | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| X | X | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| X | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

Fig. 7.13(c)

From truth table we conclude that all nine inputs are ACTIVE LOW representing decimal digit from 1 to 9. In response to input, chip produces inverted BCD code corresponding to highest numbered ACTIVE INPUT.

Here you may ask what about '0' decimal ? The answer is when all inputs are held high, output $\bar{D}\,\bar{C}\,\bar{B}\,\bar{A}$ = 1 1 1 1 i.e. DCBA = $(0000)_2$ = $(0)_{10}$.

The table also shows normal BCD count, which is inversion of output of the chip. Let's say $\bar{A}_9 = 0$, then remaining all lines are treated as 'don't care' and chip produces $\bar{D}\,\bar{C}\,\bar{B}\,\bar{A}$ = 0110 i.e. DCBA = $(1001)_2$ = $(9)_{10}$. Same way you go on making $\bar{A}_8 - \bar{A}_1$ line low and check the output.

The 74147 produces NORMALLY HIGH when none of the inputs are activated.

Scanned by CamScanner

## 7.7 Examples from University Papers :

**1. Realise 4 : 1 multiplexer using NAND gates.** **(May 96, May 97, Dec. 97, 6 Marks)**

**Ans. :** As it is 4 : 1 multiplexer it has 4 input lines and one output.

As 4 input lines are there, it will have two select lines ($S_1$, $S_0$) = m

$\therefore$ No. of input lines $n = 2^m$

$= 2^2$

$= 4$

In this circuit we have introduced one more terminal i.e. E. E stands for 'enable'. This pin is master control to enable/disable multiplexer.

$E = 1 \rightarrow$ enabled ; $E = 0 \rightarrow$ disabled.

**The truth table for 4 : 1 MUX is as follows :**

| Inputs | | | Output |
|---|---|---|---|
| Enable | Select | Inputs | |
| E | $S_1$ | $S_0$ | |
| 0 | X | X | 0 |
| 1 | 0 | 0 | $D_0$ |
| 1 | 0 | 1 | $D_1$ |
| 1 | 1 | 0 | $D_2$ |
| 1 | 1 | 1 | $D_3$ |

**Table 1**

Output equation Y is given as,

$$Y = E D_0 \bar{S}_1 \bar{S}_0 + E D_1 \bar{S}_1 S_0 + E D_2 S_1 \bar{S}_0 + E D_3 S_1 S_0 \quad \ldots(1)$$

From above equation we have to design the circuit using NAND gates.

Simplifying output equation

$$Y = \overline{\overline{E D_0 \bar{S}_1 \bar{S}_0 + E D_1 \bar{S}_1 S_0 + E D_2 S_1 \bar{S}_0 + E D_3 S_1 S_0}}$$

As $\bar{\bar{A}} = A$

Using Demorgan theorem,

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

$$Y = \overline{\overline{E D_0 \bar{S}_1 \bar{S}_0} \cdot \overline{E D_1 \bar{S}_1 S_0} \cdot \overline{E D_2 S_1 \bar{S}_0} \cdot \overline{E D_3 S_1 S_0}} \quad \ldots(2$$

Scanned by CamScanner

**Diagrammatic representation of 4 : 1 MUX using NAND gates :**

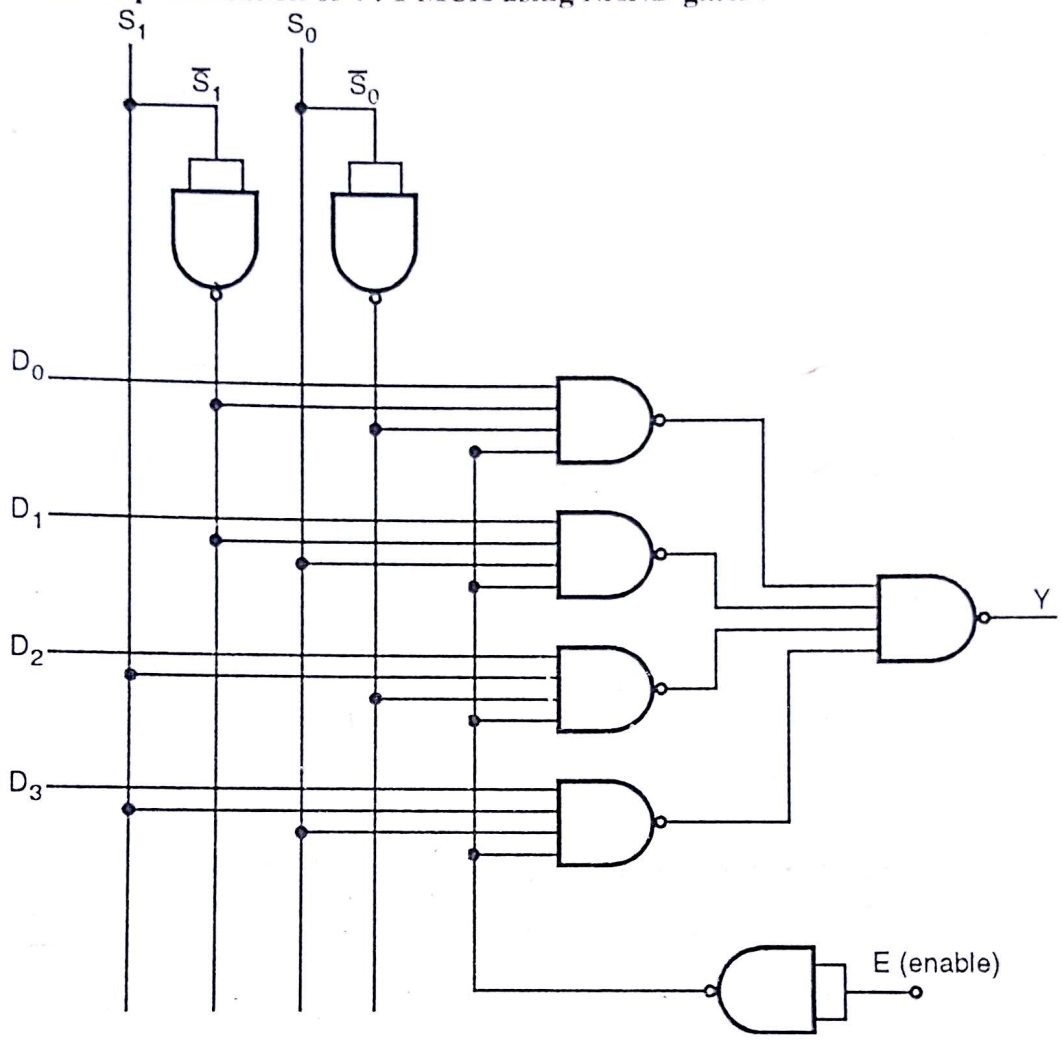



**Fig. Q. 7.1**

**2. Realise 1 : 4 demultiplexer using NAND gates.** **(Dec. 96, 6 Marks)**

**Ans. :** As it is 1 : 4 Demux, it has one input and four output lines.

As it has four output lines, so there will be two select lines ($S_1$, $S_0$).

Depending upon select lines, output will appear at $Y_0$, $Y_1$, and so on respectively. In the circuit we have introduced strobe/enable terminal. This is master control. If strobe = 0, $Y_0$ to $Y_3$ all will provide low output, irrespective of $D_{IN}$ and $S_1$, $S_0$.

**Truth table for 1 : 4 Demux is as follows :**

| Input | | | Output |
|---|---|---|---|
| Enable | $S_1$ | $S_0$ | |
| 0 | X | X | $Y_0 = Y_1 = Y_2 = Y_3 = 0$ |
| 1 | 0 | 0 | $Y_0 = \bar{S}_1\bar{S}_0D_{IN}$ |
| 1 | 0 | 1 | $Y_1 = \bar{S}_1S_0D_{IN}$ |
| 1 | 1 | 0 | $Y_2 = S_1\bar{S}_0D_{IN}$ |
| 1 | 1 | 1 | $Y_3 = S_1S_0D_{IN}$ |

**Table 2**

Scanned by CamScanner

$$Y_0 = \overline{S}_1 \overline{S}_0 D_{IN} E \quad \ldots(1)$$

$$Y_1 = \overline{S}_1 S_0 D_{IN} E \quad \ldots(2)$$

$$Y_2 = S_1 \overline{S}_0 D_{IN} E \quad \ldots(3)$$

$$Y_3 = S_1 S_0 D_{IN} E \quad \ldots(4)$$

Now Equation (1) can be written as,

$$Y_0 = \overline{\overline{S}_1 \overline{S}_0 D_{IN} E} \quad (\text{As } \overline{\overline{A}} = A) \quad \ldots(a)$$

Similarly Equations (2), (3) and (4) can be written as,

$$Y_1 = \overline{\overline{S}_1 S_0 D_{IN} E} \quad \ldots(b)$$

$$Y_2 = \overline{S_1 \overline{S}_0 D_{IN} E} \quad \ldots(c)$$

$$Y_3 = \overline{S_1 S_0 D_{IN} E} \quad \ldots(d)$$

Fig. Q.7.2(a) shows NOT gate is equivalent to single input NAND gate.

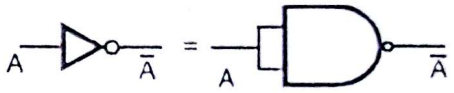



**Fig. Q.7.2(a)**

Now using Equations (a), (b), (c), (d) we can implement 1 : 4 Demux using NAND gates.

**Diagrammatic representation of 1 : 4 Demux :**

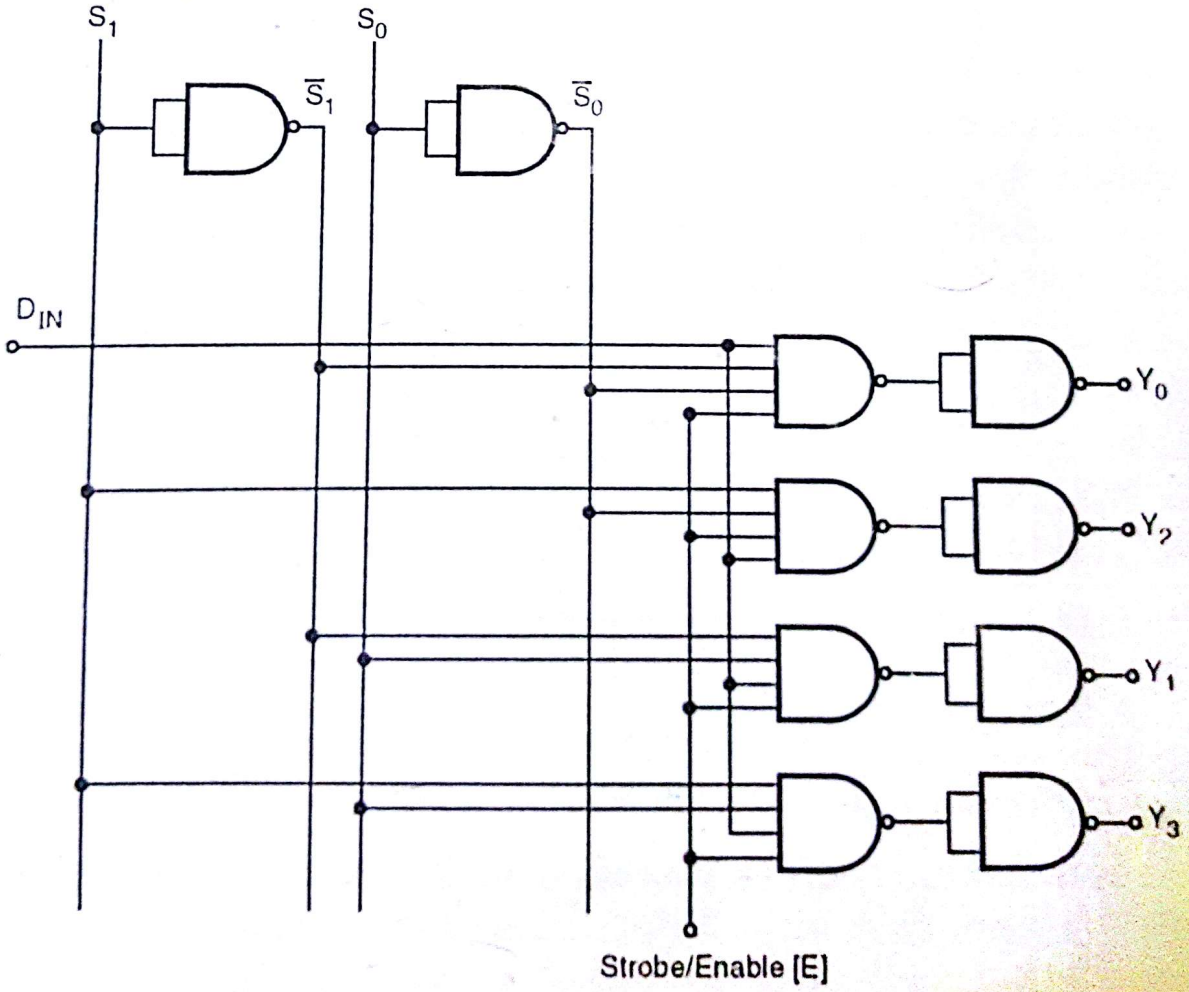



**Fig. Q. 7.2 (b)**

Scanned by CamScanner

**3. Write short note on : Arithmetic logic unit.** **(Dec. 96, May 97, 6 Marks)**

Ans.: Refer Section 7.5.

**4. What is the functional difference between 'demultiplexer' and a 'decoder' ?**

**(May 98, 3 Marks)**

Ans.: Refer Section 7.4.

**5. The following two functions are to be realised using 1 : 8 demultiplexer having 'active low' outputs.**

$$f_1 = \pi M(1, 3, 7)$$

$$f_2 = \pi M(3, 4, 5)$$ **(May 98, 8 Marks)**

**Which gate you will use if conversion SOP form is not allowed? Show realisation diagram and explain your logic.**

Ans.: As the two functions $f_1$ and $f_2$ gives the product of maxterms, so the output will be in POS form.

First write down truth table for the above functions. The functions are to be realised using 1 : 8 Demux having 'active low' outputs, so output is zero for given maxterms.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | C | $f_1$ | $f_2$ |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |

**Table 3**

As conversion of POS to SOP form is not allowed. So we will use AND gates for the outputs of both the functions.

**For function $f_1$ :** The product of maxterms 1, 3, 7 gives active low output ('0' output). As output of AND gate is zero if any one of the inputs is zero. So we are using AND gates to get active low outputs.

Now we have to implement above two functions using 1 : 8 Demux. A, B, C are select lines for 1 : 8 Demux, so outputs will be $Y_0$ to $Y_7$.

Scanned by CamScanner

**Diagrammatic Representation of $f_1$ and $f_2$ :**

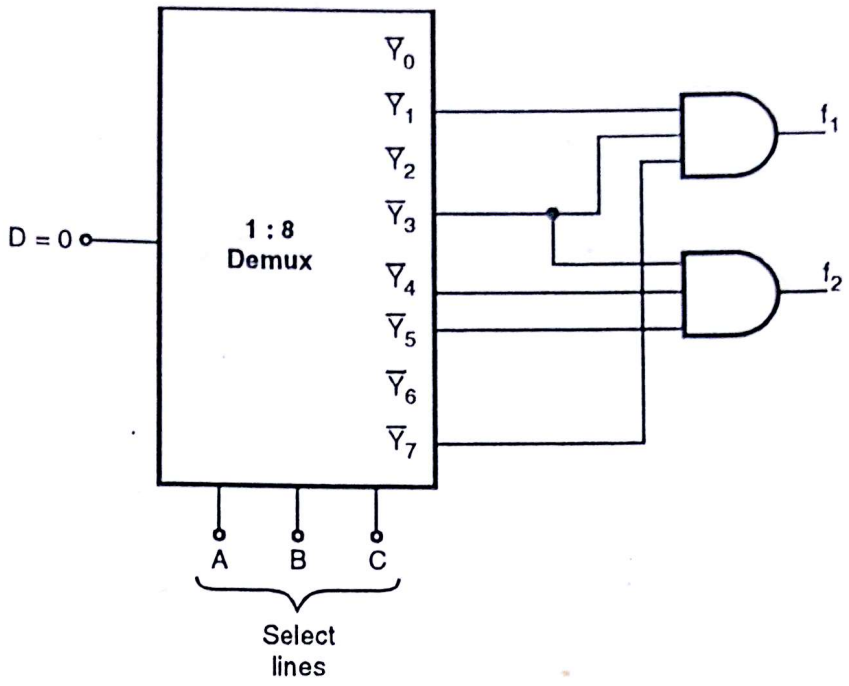



**Fig. Q. 7.5**

**6. $Y = A + B + \overline{C}$. Realise using a multiplexer.** **(Dec. 98, 6 Marks)**

**Ans. :** First we will draw K-map for the above equation,

| A \ BC | $\overline{B}\overline{C}$ 00 | $\overline{B}C$ 01 | BC 11 | $B\overline{C}$ 10 |
|---|---|---|---|---|
| $\overline{A}$ 0 | 1 (0) | 0 (1) | 1 (3) | 1 (2) |
| A 1 | 1 (4) | 1 (5) | 1 (7) | 1 (6) |

From K-map we can write the truth table for above function.

| A | B | C | Output Y | Input to MUX |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | $D_0$ |
| 0 | 0 | 1 | 0 | $D_1$ |
| 0 | 1 | 0 | 1 | $D_2$ |
| 0 | 1 | 1 | 1 | $D_3$ |
| 1 | 0 | 0 | 1 | $D_4$ |
| 1 | 0 | 1 | 1 | $D_5$ |
| 1 | 1 | 0 | 1 | $D_6$ |
| 1 | 1 | 1 | 1 | $D_7$ |

**Table 4**

Scanned by CamScanner

Here there are three select lines A, B, C therefore inputs to a multiplexer will be $2^3 = 8$ So we will use 8 : 1 multiplexer for the function

$$Y = A + B + \bar{C}$$

**Diagrammatic representation of equation using 8 : 1 MUX :**

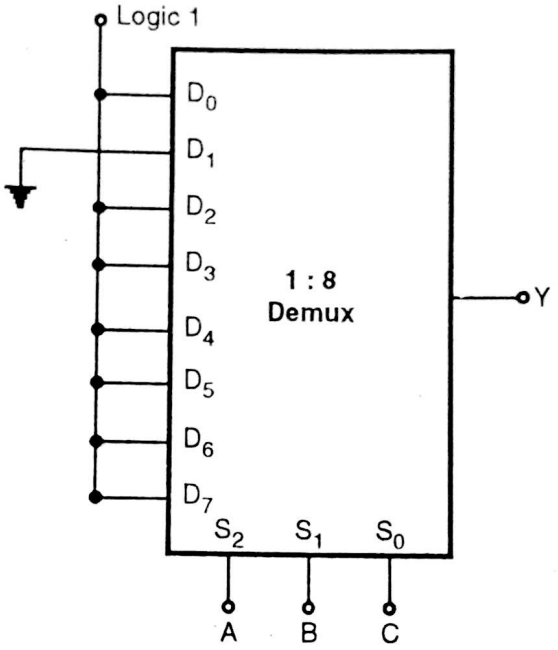



**Fig. Q. 7.6**

7. **Realise following function using 2 : 1 Mux :**

**Y = AB, with B as the select input.**

**Ans.:** Y = AB means output is logic 1 only when A = 1, B = 1, $(3)_{10}$, for all other combinations output is zero.

Therefore, design Table 5.

| Input | $D_0$ | $D_1$ |
|---|---|---|
| $\bar{A}$ | 0 | 1 |
| A | 2 | ③ |
| Final input to MUX | 0 | A |

**Table 5**

Here as B is the select input therefore inputs to multiplexer are $2^1 = 2$

Scanned by CamScanner

**Diagrammatic representation of Y = AB using 2 : 1 MUX :**

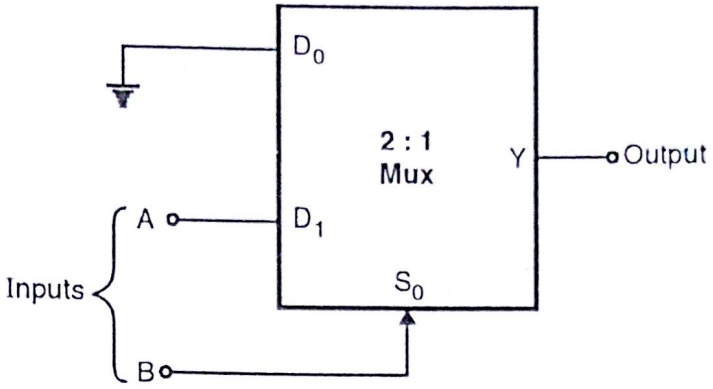



**Fig. Q. 7.7**

**8. Realize a 2–bit comparator to compare two 2–bit numbers, using appropriate demultiplexer with active high outputs and additional gates.** **(June 99, 8 Marks)**

**Ans. :** We have to design 2–bit comparator which compares two 2–bit numbers.

Let one number is $A_1A_0$ and another is $B_1B_0$. Therefore the truth table contains 4 variables so there will be 16 output lines.

The output of 2–bit comparator will be of 3 types.

If $A = A_1A_0$ and $B = B_1B_0$

Therefore outputs will be (i) $A > B$ (ii) $A = B$ (iii) $A < B$.

Truth table for 2–bit comparator is as follows :

| Inputs | | | | Outputs | | |
|---|---|---|---|---|---|---|
| $A_1$ | $A_0$ | $B_1$ | $B_0$ | A > B | A = B | A < B |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

**Truth Table 6**

Scanned by CamScanner

Now we have to design 2–bit comparator using Demultiplexer with active high outputs. As there are four select lines $A_1A_0B_1B_0$, output lines are $2^m = n$.

$$2^4 = 16$$

Therefore demultiplexer we are using is 1 : 16 Demultiplexer.

As this Demultiplexer uses active high outputs so we will consider logic 1's only.

**Diagrammatic representation of 2–bit comparator**

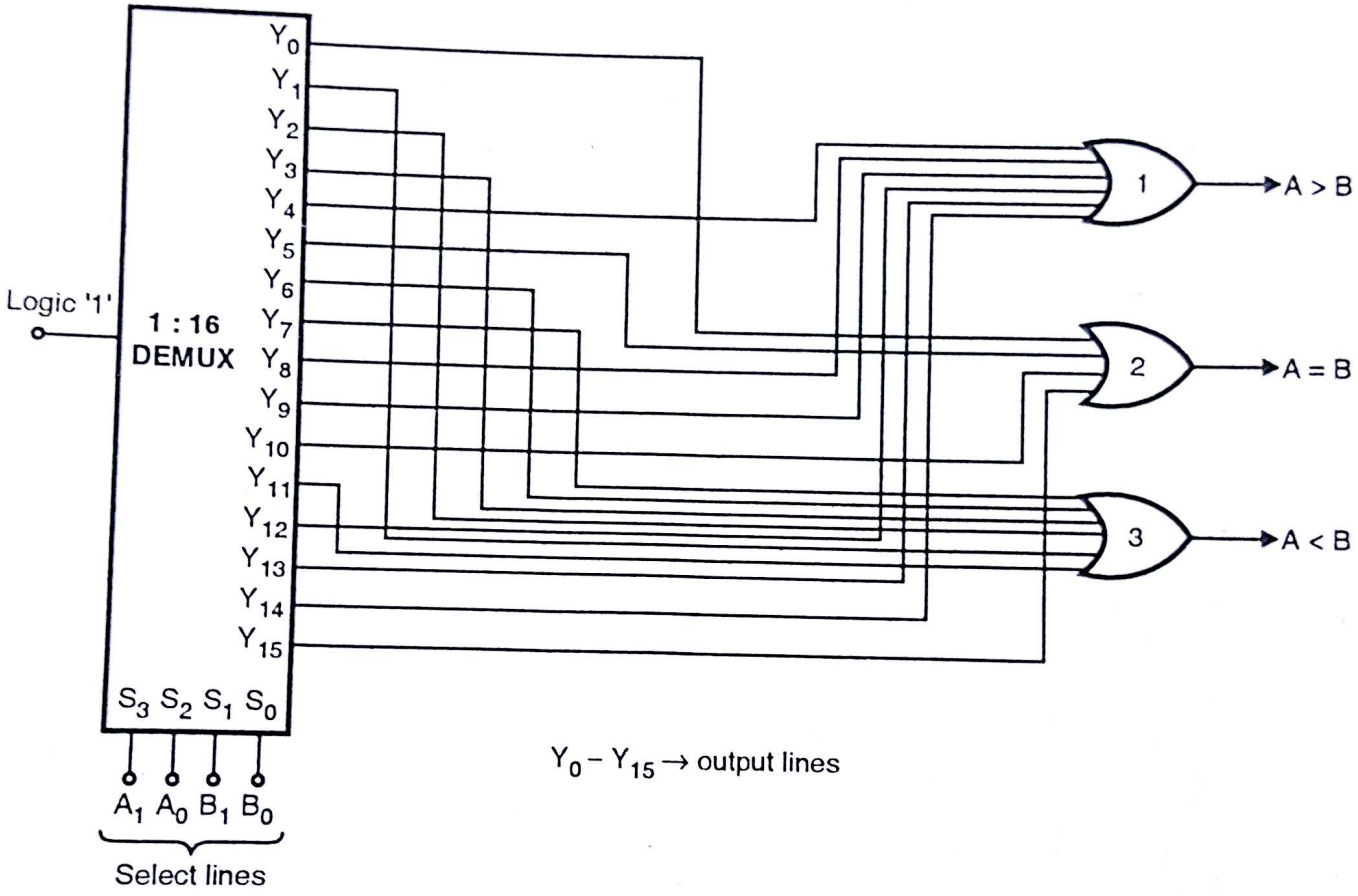



**Fig. Q.7.8**

**9. Implement an even parity checker for a 4–bit data, using 8 : 1 MUX and inverters.**

**(June 99, 8 Marks)**

**Ans. :** The parity bit can be either 0 or 1 depending upon number of 1's contained in the code group. An even parity checker checks the number of 1's in the data and gives the output of number of 1's are even then parity bit gives zero as the output.

The truth table for 4 bit even parity checker is as follows (Refer Table 8) :

As there are eight input pins therefore we will use 8 : 1 MUX.

Diagrammatic representation of 4–bit even parity checker is as shown in Fig. Q.7.9.

A, B, C are select lines.

Bit D is LSB.

Observing bit D and output $P_e$ we have written function.

Scanned by CamScanner

| A | B | C | D | Output $P_e$(Parity bit) | I/P to multiplexer | Function |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | |
| 0 | 0 | 0 | 1 | 1 | $D_0$ | $D_0 = D$ |
| 0 | 0 | 1 | 0 | 1 | | |
| 0 | 0 | 1 | 1 | 0 | $D_1$ | $D_1 = \bar{D}$ |
| 0 | 1 | 0 | 0 | 1 | | |
| 0 | 1 | 0 | 1 | 0 | $D_2$ | $D_2 = \bar{D}$ |
| 0 | 1 | 1 | 0 | 0 | | |
| 0 | 1 | 1 | 1 | 1 | $D_3$ | $D_3 = D$ |
| 1 | 0 | 0 | 0 | 1 | | |
| 1 | 0 | 0 | 1 | 0 | $D_4$ | $D_4 = \bar{D}$ |
| 1 | 0 | 1 | 0 | 0 | | |
| 1 | 0 | 1 | 1 | 1 | $D_5$ | $D_5$ D |
| 1 | 1 | 0 | 0 | 0 | | |
| 1 | 1 | 0 | 1 | 1 | $D_6$ | $D_6 = D$ |
| 1 | 1 | 1 | 0 | 1 | | |
| 1 | 1 | 1 | 1 | 0 | $D_7$ | $D_7 = \bar{D}$ |

**Truth Table 7**

Connect $D_0$, $D_3$, $D_5$ and $D_6$ are connected to LSB bit D and $D_1$ , $D_2$, $D_4$ and $D_7$ are connected to $\bar{D}$ using inverter.

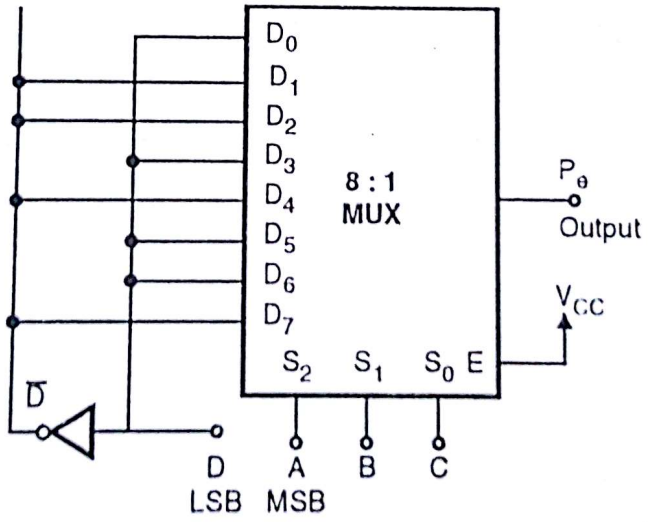



**Fig. Q.7.9**

**10. Realize a full adder using appropriate demultiplexer and additional gates. Explain.**

**(Dec.98, 6 Marks)**

**Ans.:** Refer Ex. 19, Page. 7.26.

❑❑❑

Scanned by CamScanner

# Flip–Flop

8

**Syllabus :**

Flip-flops, SR, clocked SR, JK, D, T, MS-JK,

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Terminologies used in sequential system | 8.2 | ✓ | – |
| • NOR gate latch | 8.3 | ✓ | ✓ |
| • DFF (Delay Flip-Flop) | 8.4 | ✓ | ✓ |
| • Flip-Flop timings | 8.5 | ✓ | – |
| • JK Flip-Flop | 8.6 | ✓ | – |
| • Master Slave JKFF | 8.7 | ✓ | – |
| • T Flip-Flop | 8.8 | ✓ | – |
| • Conversion from one type of FF to another. | 8.9 | – | ✓ |
| • Applications of Flip-Flops | 8.10 | ✓ | ✓ |



**Comment**

- Very very very important chapter.
- Base chapter for counter design.
- Base chapter for Analysis and synthesis of sequential circuits.
- Concentrate on full chapter very corectly.

Scanned by CamScanner

## 8.1 Introduction :

➥ [ASKED IN EXAM - MAY 1996 !!! ]

In previous chapters, we have directed our studies towards analysis and design of combinational digital circuits. The output of combinational circuit at any instant of time, depends upon levels present at the input terminal. Secondly, any prior input level condition have no effect on present outputs because combinational circuit is memoryless. [It cannot memorize previous states].

But in sequential circuit, ***timing*** parameter is coming into picture and output levels are sequential. To understand in better manner let's analyse it taking example of normal life, i.e. you wake up in the morning, brush your teeth, take breakfast, attend college and so on. Your each and every activity is performed sequentially, plus it should follow a ***particular defined sequence***. Why I say defined sequence, because you are not going to brush the teeth first and then going to wake up. In short, next activity depends upon previous activity. Being human you can ***remember*** the sequence. But what about digital circuit ?. So now we are going to study different memory element in this chapter.

Fig. 8.1 shows, combination of memory element and combinational logic circuit. The combinational portion accepts logic signals from external inputs and from the present outputs of the memory elements. (Present outputs of memory element is referred as *'present state'*). The combinational circuit operates on these inputs to produce various new outputs. Some of the new outputs are stored in memory element, referred to as ***next state***.

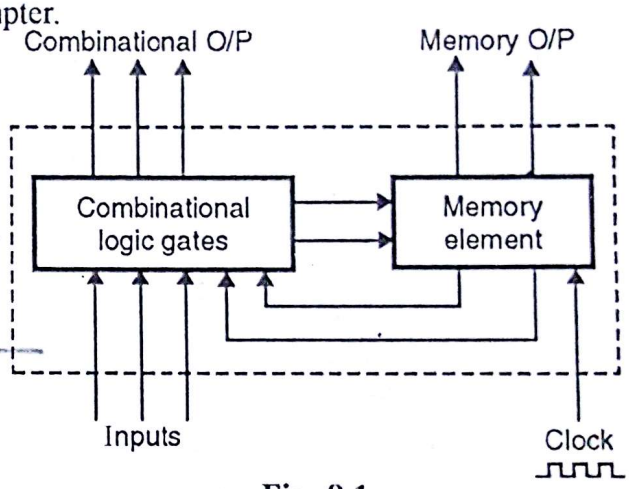



**Fig. 8.1**

In sequential logic circuit most important part seems to be ***memory element***. The memory element is called ***Flip-Flop*** (abbreviated as FF). In this chapter we are going to study following flip-flops :

| | |
|---|---|
| 1. SR FF (Set Reset Flip - Flop) | 2. D FF (Delay Flip - Flop) |
| 3. JK FF | 4. T FF (Toggle Flip - Flop) |
| 5. Master Slave JK FF (M/S JK Flip - Flop) | |

FF is also called as ***latch*** because it holds or latches contents of previous level of signal.

## 8.2 Terminologies used in Sequential System :

Before we start study of sequential system, it is necessary to introduce few new terminologies, to describe sequential systems more easily.

### 8.2.1 CLOCK (CLK) - A Timing Signal :

**Timing Signal :**

The main key control in synchronous sequential circuit is clock signal.

**Characteristic of Clock :**

- CLK has a regular waveform that repeats once every T seconds, where T is the clock period or cycle.
- Normally CLK has 50% duty cycle. Therefore, ratio $T_1/T$ is around 0.5.

Scanned by CamScanner

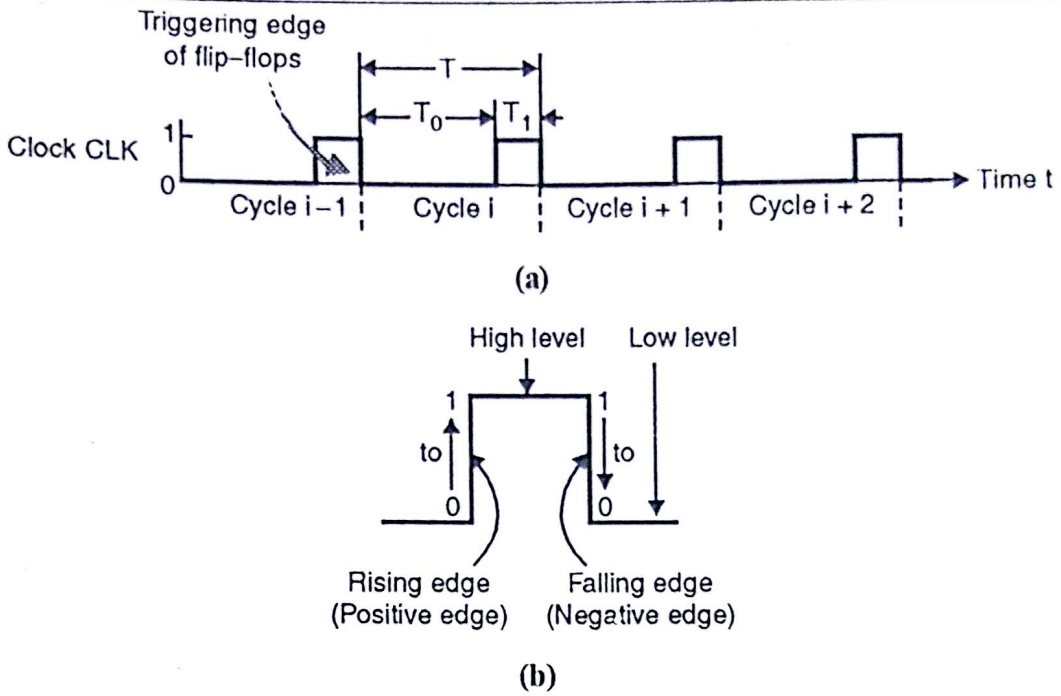



**Fig. 8.2 : CLOCK signal and CLOCK edges**

- During each CLK period, CLK exhibits positive edge and a negative edge. Either of the edge changes state of the circuit.
- Normally it is convenient to design CLK such that it begins at one triggering edge and ending at the next. In Fig. 8.2, it is shown that clock cycle begins and ends at negative (falling) edge of the CLK.

Here question comes, ***why we name it, as positive and negative edge*** ?

Friends, recall ***differentiator circuit*** and try to answer what will be the output of the differentiator when CLK input is given. Answer is graphically represented in Fig. 8.3.

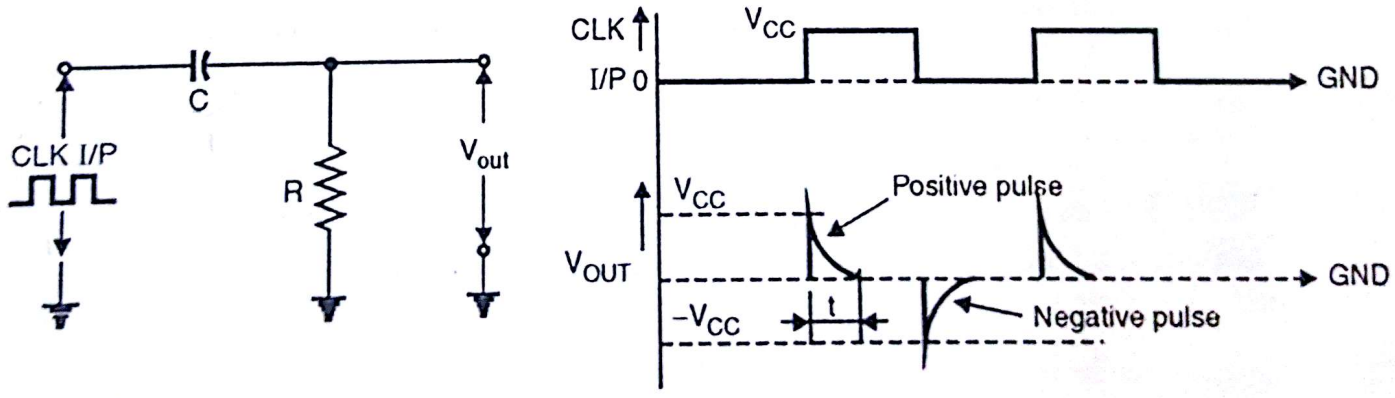



**Fig. 8.3**

## 8.2.2 State :

State represents ***what is stored in memory.*** It is stored in binary devices.

Scanned by CamScanner

### 8.2.3 State Table :

Behaviour of sequential circuit is defined by the relationship between its input and output signals. In combinational circuit, we use to prepare truth table. But in sequential circuit, if we describe behaviour of it, we have to list all possible input/output sequences that can be applied to the circuit. Because there is an infinite number of such sequences and sequence can be of arbitrary length. Such an explicit truth table like listing is impractical, even for simple circuit.

To deal with this infinity of input/output sequences, we observe that number of combinations of primary input and internal state values is finite.

Let's say if we have, $n \longrightarrow$ Input variables X

$p \longrightarrow$ State variables Y

then $2^{n+p} \longrightarrow$ Number of different X, Y combinations.

The time sequence of input, output and flip flop states may be enumerated in a state table. (Actually switching circuit theory books call this table a transition table).

The generalised format for state table is shown in Fig. 8.4.

| Previous state | | Next state | | | | Output | |
|---|---|---|---|---|---|---|---|
| | | x = 0 | | x = 1 | | x = 0 | x = 1 |
| A | B | A | B | A | B | y | y |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |

**Fig. 8.4 : Generalised format for state table**

Note : (1) A and B, are state variables.
(2) x is input variable
(3) y is single output.

State table consists of :

(1) Present State
(2) Next State
(3) Output.

**Present state** : It designates the states of flip flop before the occurrence of a clock pulse.
**Next state** : It designates the states of flip flop after occurence of clock pulse.
**Output** : This section lists out values of output variables during the present state.

Both the next state and output sections have two columns, one for x = 0 and the other for x = 1. The derivation of the state table starts from an assumed initial state. The initial state for most practical sequential circuits assumed to be 0's, because of power ON reset. It is also possible to have state other than '0'. But it depends upon sequential circuit. Normally, state table contains all the circuit's input/output sequences in implicit and finite form.

Scanned by CamScanner

### 8.2.4 State Diagrams :

The information available in state table may be represented graphically in state diagram.

- Internal state is represented by a node or a circle.
- Arrows between the nodes define state transition.
- Each state transition arrow is labeled with primary input and output signals associated with the transition. Normal way of writing the same is X/y.

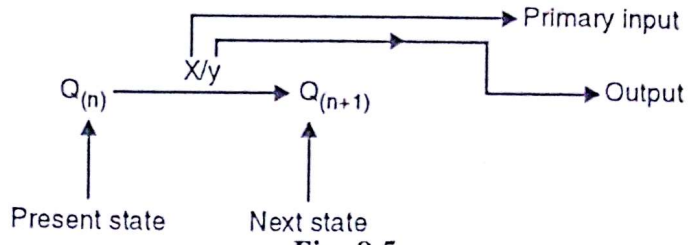
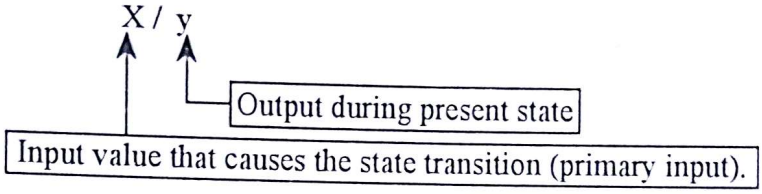
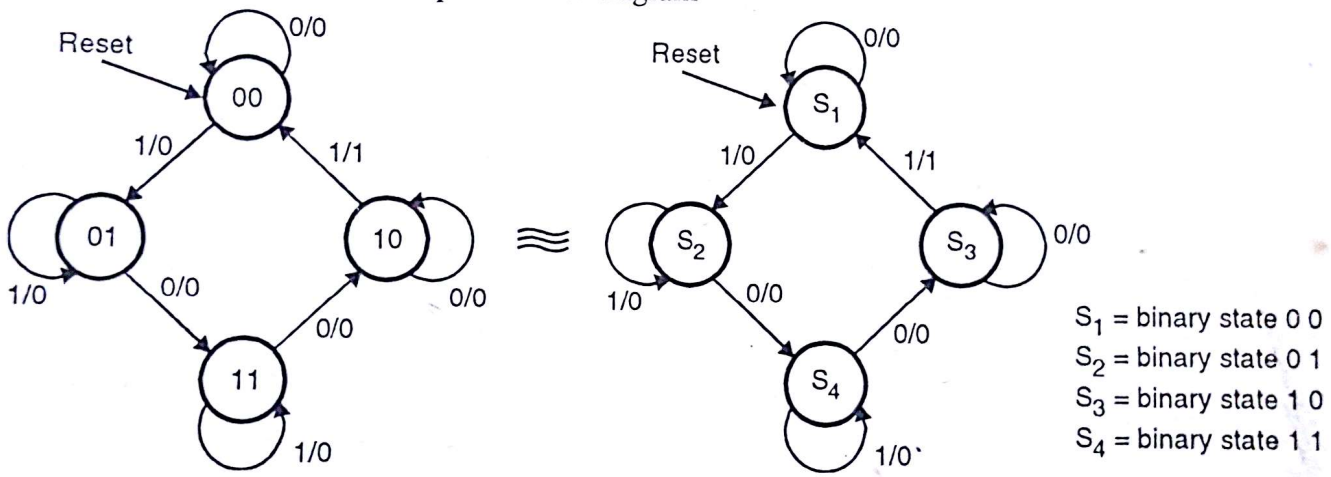



Fig. 8.5




Slash serve to separate input from output.
Let's refer Fig. 8.6 as an example for state diagram




**(a) State diagram (An example)** **(b)**

**Fig. 8.6**

**Important Points :**

(a) Directed line from state 00 ($S_1$) to state 01 ($S_2$) is labeled 1/0. It means that sequential circuit is in a present state 00 ($S_1$). While input is '1' and output state is '0', on the termination of next clock pulse, circuit goes to next state 01 ($S_2$).

(b) A directed line connecting to a circle itself, indicates no change in the state of the sequential circuit, for that particular input.

(c) The arrow marked RESET indicates that the state it enters is the circuit's initial or reset state. Because it is an asynchronous signal and may be applied in any state, RESET is drawn coming from nowhere in particular.

(d) One can have multiple inputs and multiple outputs.

Scanned by CamScanner

(e) There must be one path from each state for each possible input combination.

(f) Sometimes, the same next state is reached for both input combinations. It means that irrespective of input, for next clock, simply change the state. In such situation input will be ***don't care*** Fig. 8.7 depicts the same.

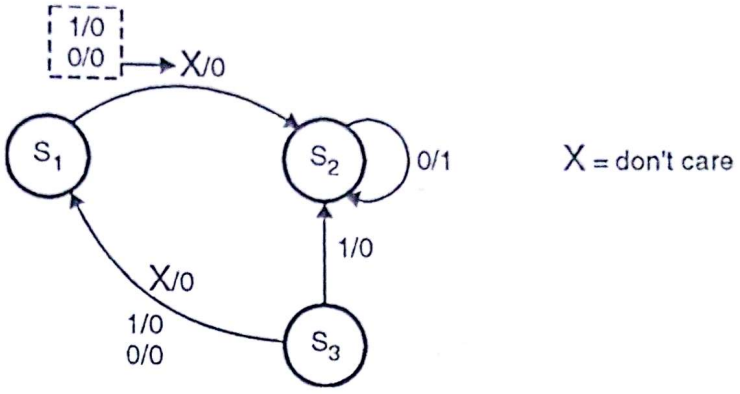



**Fig. 8.7 : Don't care condition in state diagram**

(g) Sometimes, user doesn't require any output. In such case, output will be don't care.

(h) If input variable(s) and output(s) ; both are don't care ( X ), circuit will simply switch over from one state to another, after each clock transition.

| Note : | This type of sequential circuit, which simply switches over to next state irrespective of input(s), is nothing but a **counter** circuit. The same we will study in next chapter. |
|---|---|

## 8.2.5 State Equation :

| Note : | Basically there is no difference between state diagram and state table, except, manner of representation. The state table is easier to derive from given logic diagram and the state diagram follows directly from a state table. State diagram is sometime used as ***initial design specification*** of a sequential circuit. As state diagram is graphical presentation, it provides suitability for human interpretation of the circuit operation. |
|---|---|

State equation describes conditions for flip flop state transition in algebraic form. The left side of the equation denotes next state of the flip flop. The R.H.S. part of equation defines boolean function that specifies the present state conditions that make the next state equal to 1. For example,

$$\underbrace{A(n+1)}_{\text{Next state}} = \underbrace{(\bar{A}B + A\bar{B} + AB)\,\bar{x} + ABx}_{\text{Present state and inputs}} \qquad \ldots(1)$$

State equation is also referred as ***application equation***.

Normally state equation is derived from state table. If you observe Equation (1), you can conclude that state equation is similar in form to flip flop characteristic equation, only difference is state expression defines next state in terms of external input variables and other flip flop values. Secondly, state equation is boolean function with time included. Next state depends upon R.H.S. part of state equation.

If RHS part is equal to '0' and clock is given, next state = 0.

Scanned by CamScanner

If RHS part is equal to '1' and clock is given, next state = 1.

There are always two phases of operation i.e. Analysis and Synthesis. In analysis, we find state table, state diagram for a given sequential circuit ; whereas in synthesis, for a given statement or state diagram, we design sequential circuit.

## 8.3 NOR Gate Latch : [S – R Flip–Flop ] :

Now we will analyse latch using NOR gate.

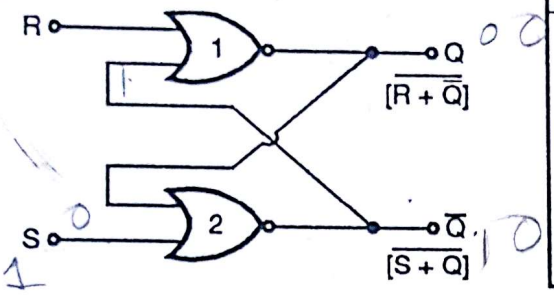



Fig. 8.8(a) : NOR gate latch

| S | R | $Q_{n+1}$ | $\bar{Q}_{n+1}$ | |
|---|---|---|---|---|
| 0 | 0 | NC ($Q_n$) | NC ($\bar{Q}_n$) | |
| 0 | 1 | 0 | 1 | → Reset |
| 1 | 0 | 1 | 0 | → Set |
| 1 | 1 | RACE | RACE | 1 |

Fig. 8.8(b) : Truth Table

**Note :** $Q_n$ = Previous state ; $Q_{n+1}$ = Next state.

Refer Truth table of NOR gate as well as Fig. 8.8(a).

**Working :**

**Case I :** (i) S = R = 0. Previous state of Q = 0 and $\bar{Q}$ = 1.

∴ Input to NOR1 is R = 0, $\bar{Q}$ = 1. ∴ Q = 0.

∴ Input to NOR2 is S = 0, Q = 0. ∴ $\bar{Q}$ = 1.

So we conclude that output does not change.

(ii) S = R = 0. Previous state of Q = 1, $\bar{Q}$ = 0

∴ Input to NOR1 is R = 0, $\bar{Q}$ = 0. ∴ Q = 1

∴ Input to NOR2 is S = 0, Q = 1. ∴ $\bar{Q}$ = 0. Again no change in output

**Conclusion :** When S = R = 0, Next state ($Q_{n+1}$) = Previous state ($Q_n$).

**Case II :** (i) S = 0, R = 1, $Q_n$ = 1, $\bar{Q}_n$ = 0

∴ Input to NOR1 is R = 1, $\bar{Q}_n$ = 0, means Q will change from 1 to 0. ∴ $Q_{n+1}$ = 0.

Now input to NOR2 is S = 0, $Q_{n+1}$ = 0, ∴ $\bar{Q}_n$ will change from 0 to 1, ∴ $\bar{Q}_{n+1}$ = 1.

Again go through the chain.

Now, Input to NOR1 is R = 1, $\bar{Q}_{n+1}$ = 1, ∴ $Q_{n+1}$ = 0.

Input to NOR2 is S = 0, $Q_{n+1}$ = 0, ∴ $\bar{Q}_{n+1}$ = 1. means circuit reached to stable state.

(ii) S = 0, R = 1, $Q_n$ = 0, $\bar{Q}_n$ = 1.

As seen in above case $Q_n$ = 0, $\bar{Q}_n$ = 1 is stable state when S = 0, R = 1.

Scanned by CamScanner

$\therefore \quad Q_{n+1} = 0, \bar{Q}_{n+1} = 1$

**Conclusion :** When S = 0, R = 1, irrespective of previous state, $Q_{n+1} = 0$, $\bar{Q}_{n+1} = 1$. This state is ***'RESET'*** state of circuit.

**Case III :**

(i) S = 1, R = 0, $Q_n = 0$, $\bar{Q}_n = 1$

$\therefore$ Input to NOR1 is R = 0, $\bar{Q} = 1$ ; produces Q = 0.

$\therefore$ Input to NOR2 is S = 1, Q = 0.

$\therefore$ Q will change from 1 to 0, $\therefore \bar{Q}_{n+1} = 0 = \bar{Q}$.

$\therefore$ Input to NOR1 is R = 0, $\bar{Q} = 0$.

$\therefore$ Q will change from 0 to 1, $\therefore Q_{n+1} = 1 = Q$.

Now input to NOR2 is S = 1 and Q = 1. $\therefore \bar{Q} = 0$.

$\therefore$ Input to NOR1 is R = 0, $\bar{Q} = 0$. $\therefore$ Q = 1.

Means circuit reached to stable state, $Q_{n+1} = 1$, $\bar{Q}_{n+1} = 0$.

(ii) S = 1, R = 0, $Q_n = 1$, $\bar{Q}_n = 0$.

As seen in above case $Q_n = 1$, $\bar{Q}_n = 0$ is stable state when S = 1, R = 0.

**Conclusion :** When S = 1, R = 0, irrespective of previous condition $Q_{n+1} = 1$, $\bar{Q}_{n+1} = 0$. This state is ***'SET'*** state of the circuit.

**Case IV :** When S = 1, R = 1, we know the fact that when either of the input to NOR gate is 1, output of gate is 0. This condition leads us to 'RACE' condition of the circuit. It means both the outputs will try to reach logic1; and therefore, this state is ambiguous or undefined. Normally referred as ***RACE***.

**State table**

For SR FF, we have two inputs, i.e. S and R. Q output represents, state of FF. As Q is single output, there will be two states i.e. 0 and 1. Note that $\bar{Q}$ is simply inverted output of Q. Therefore not treated as separate output. Thus state table will be as follows :

| Present State Q(n) | Next state Q (n + 1) | | | |
|---|---|---|---|---|
| | SR = 00 | SR = 01 | SR = 10 | SR = 11 |
| 0 | 0 | 0 | 1 | Race |
| 1 | 1 | 0 | 1 | Race |

**Table 8.1 : State table for SR FF**

Race condition cannot be shown or predicted.

Scanned by CamScanner

**State Diagram**

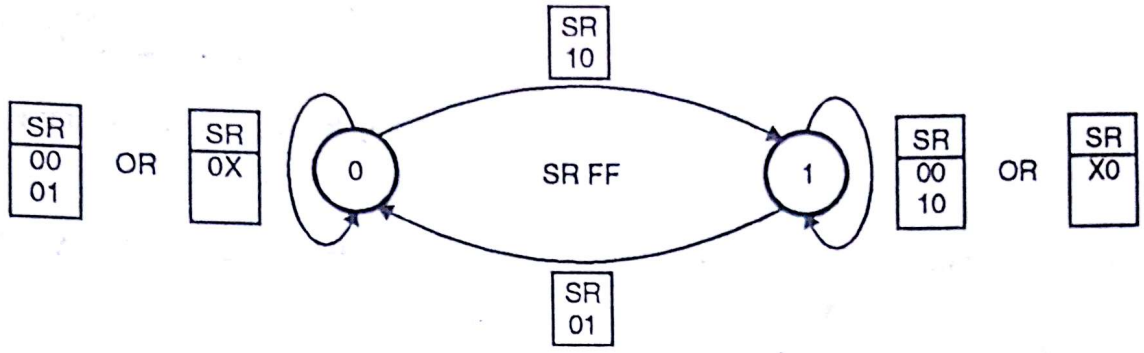



**Fig. 8.9 : State diagram of SR FF**

Thus if we observe we have following state transitions, with different combinations of SR input.

| Previous state (Q(n)) | to | Next state (Q(n + 1) | If | S | R |
|---|---|---|---|---|---|
| 0 | | 0 | | 0 | X |
| 0 | | 1 | | 1 | 0 |
| 1 | | 0 | | 0 | 1 |
| 1 | | 1 | | X | 0 |

**Table 8.2 : Excitation table**

Read the table as given below,

***If previous state is Q(n) then to switch over to next state i.e. Q (n + 1) input S and R should be Q(n + 1).***

***For example,*** if previous state is 0, then to switch over to next state 1, input S and R should be 10 [S = 1, R = 0].

The above Table 8.2 is referred as excitation table.

**State Equation**

For SR FF, the state equation is,

$$Q(n+1) = S + \bar{R}\,Q(n)$$

$\overline{RQ_n}$

$S\bar{Q}$

## 8.3.1 Nand Latch :

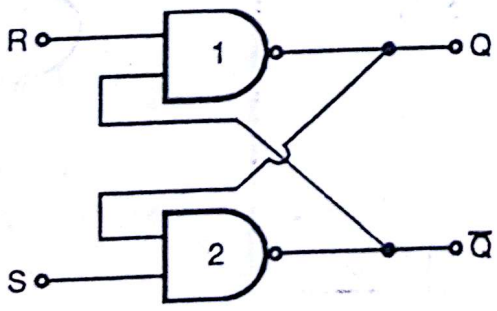



**Fig. 8.10(a) : Nand latch**

| S | R | $Q_{n+1}$ | $\bar{Q}_{n+1}$ | |
|---|---|---|---|---|
| 0 | 0 | RACE | RACE | |
| 0 | 1 | 0 | 1 | → Reset |
| 1 | 0 | 1 | 0 | → Set |
| 1 | 1 | NC ($Q_n$) | NC ($\bar{Q}_n$) | → Inactive |

**Fig. 8.10(b) : Truth Table**

Scanned by CamScanner

The way we have analysed NOR latch, you can analyse NAND latch and prove the truth table shown in Fig. 8.10(b).

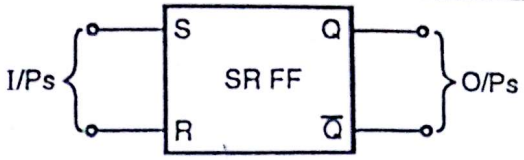



**Fig. 8.11 : Symbol of simple SR FF**

## 8.3.2 Level Clocking :

[ASKED IN EXAM - DEC. 97 !!! ]

As mentioned, sequential circuit requires timing pulses. i.e. clock. Clock is nothing but pulses generated from oscillator. Logic level toggles from 1/0 and 0/1. Normally, 50% duty cycle is preferred. Refer Fig. 8.12.

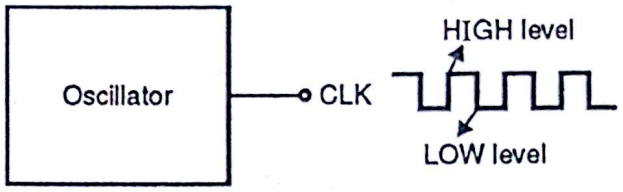
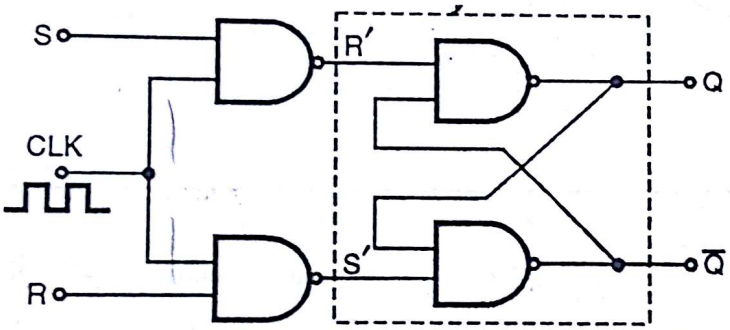


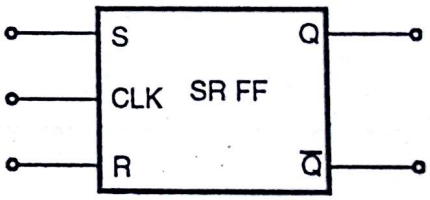

**Fig. 8.12**

Level clocking means FF responds either on *LOW* level of CLK or *HIGH* level of CLK. HIGH level clocked is termed as positive level clocked. LOW level clocked is termed as negative level clocked.

## 8.3.3 Clocked Nand Latch :




**(a) Clocked SR FF**




**(b) Symbol**

**Fig. 8.13 : Simple SR latch**

**Truth Table**

| Inputs | | | Outputs | |
|---|---|---|---|---|
| CLK | S | R | $Q_{n+1}$ | $\overline{Q}_{n+1}$ |
| 0 (⌐_⌐) | X | X | NC ($Q_n$) | NC ($\overline{Q}_n$) |
| 1 (_⌐_) | 0 | 0 | NC ($Q_n$) | NC ($\overline{Q}_n$) |
| 1 (_⌐_) | 0 | 1 | 0 | 1 |
| 1 (_⌐_) | 1 | 0 | 1 | 0 |
| 1 (_⌐_) | 1 | 1 | RACE | RACE |

_⌐_ = HIGH level

⌐_⌐ = LOW level

**Table 8.3 : Truth table of level triggered SR FF**

Let's analyse the circuit behaviour and verify truth table.

Scanned by CamScanner

**Case I :** Refer truth table of simple SR FF (Fig. 8.13(b)), when CLK level = 0 irrespective of S and R input S' = R' = 1. Therefore $Q_n$ and $\bar{Q}_n$ will not change. $\therefore Q_{n+1} = Q_n$, $\overline{Q_{n+1}} = \bar{Q}_n$

**Case II :** When CLK = 1, S = 0, R = 0.

Output S' = 1 and R' = 1. Therefore again $Q_n$ and $\bar{Q}_n$ will not change.

$\therefore Q_{n+1} = Q_n, \bar{Q}_{n+1} = \bar{Q}_n$

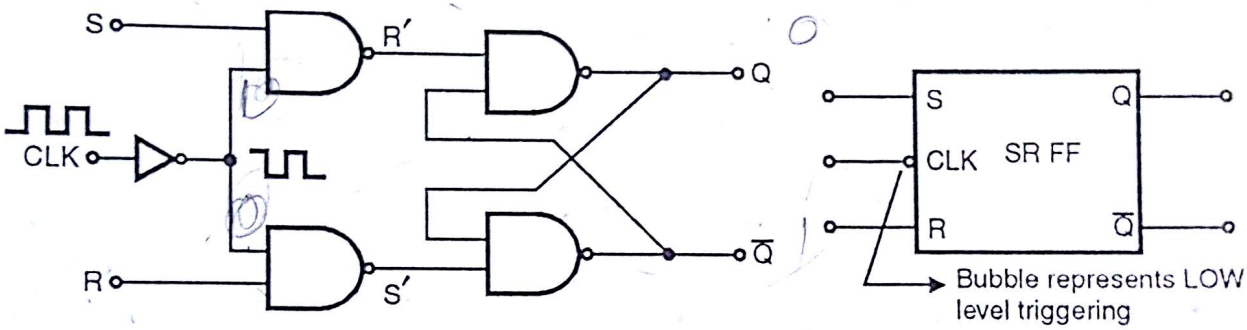
**Case III :** CLK = 1, S = 0, R = 1.

$\therefore$ S' = 0, R' = 1, Therefore $Q_{n+1} = 0, \bar{Q}_{n+1} = 1$. This condition is *'RESET'* condition.

**Case IV :** CLK = 1, S = 1, R = 0. $\therefore$ S' = 1, R' = 0, $\therefore Q_{n+1} = 1, \bar{Q} = 0$. This condition is *SET* condition.

**Case V :** CLK = 1, S = 1, R = 1. $\therefore$ S' = 0, R' = 0, $\therefore$ *RACE* condition.

For low level or negative clocked SR FF, insert an inverter at CLK input as shown in Fig. 8.14

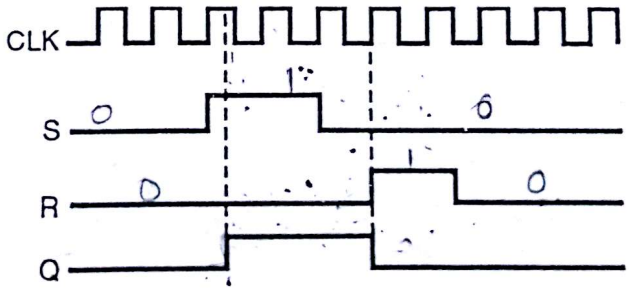



(a) Low level clocked SR FF (b) Symbol

**Fig. 8.14**

**Ex. 1 :** Draw output waveform for following input signals :

**Soln. :**



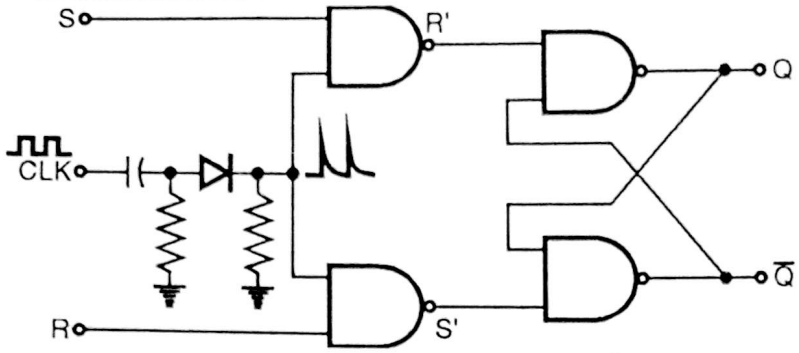

(Initially Q is considered at 0 level)

**Fig. Ex. 8.1**

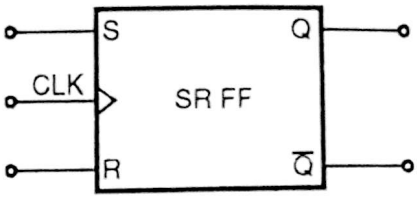
## 8.3.4 Edge Triggered Nand Latch : [ASKED IN EXAM - DEC. 97, 98, JUNE 99 !!! ]

In previous discussion we saw that, when there is HIGH/LOW level of the CLK, depending upon SR input, output Q and $\bar{Q}$ changes. But our requirement is such that FF should respond only when CLK changes the state, i.e. either on rising edge or on falling edge. Fig. 8.15 depicts edge triggered SR FF.

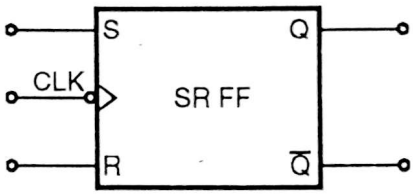
Scanned by CamScanner




(a) Positive edge triggered SR FF




(b) Symbol of positive edge triggered SR FF




(c) Symbol of negative edge triggered SR FF

**Fig. 8.15**

As shown output of differentiator is differentiation of CLK pulses. Time duration 't' depends upon RC time constant. $V_{out}$ pulses are ***Sharp spikes*** of height approx. ($\pm V_{CC}$).

One has to clip negative pulse and give the clipped output to common terminal of NAND latch as shown in Fig. 8.15(a).

$\uparrow \Rightarrow$ Positive edge

$\downarrow \Rightarrow$ Negative edge

| Inputs | | | Outputs | |
|---|---|---|---|---|
| CLK | S | R | $Q_{n+1}$ | $\overline{Q}_{n+1}$ |
| 0 | X | X | NC ($Q_n$) | NC ($\overline{Q}_n$) |
| 1 | X | X | NC ($Q_n$) | NC ($\overline{Q}_n$) |
| $\downarrow$ | X | X | NC ($Q_n$) | NC ($\overline{Q}_n$) |
| $\uparrow$ | 0 | 0 | NC ($Q_n$) | NC ($\overline{Q}_n$) |
| $\uparrow$ | 0 | 1 | 0 | 1 |
| $\uparrow$ | 1 | 0 | 1 | 0 |
| $\uparrow$ | 1 | 1 | RACE | RACE |

**Table 8.4 : Truth Table of SR FF**

**Ex. 2 :** Draw output waveform for given S and R input, for positive edge triggered SR FF.

**Soln. :**

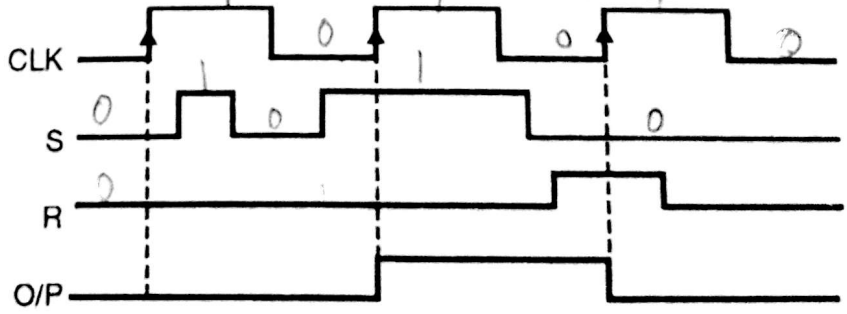



**Fig. Ex. 8.2**

Scanned by CamScanner

## 8.3.5 Preset and Clear :

**Problem in Working with Flip–Flop**

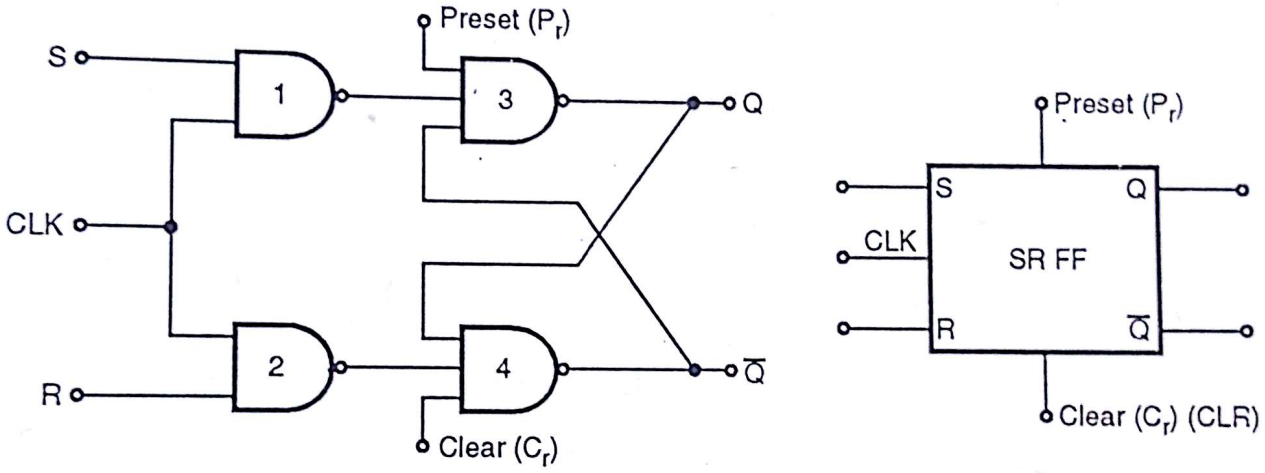
When power supply is switched on, the state of circuit is uncertain. Output is set or reset, that is unpredictable. In some application(s), user want Q to be very specific either SET or RESET. Therefore one has to consider initial condition also. ***How to accomplish required initial condition*** ? Solution, is to use Preset (also called ***direct set***) and clear (also called as ***direct reset***) terminal.

Preset and clear inputs can be given in two ways :

(1) Synchronism with CLK (called synchronous Preset/clear)

(2) Asynchronism with CLK (called asynchronous Preset/clear)

**Asynchronous Preset and Clear**




**(a) SR FF with preset and clear** **(b) Symbol**

**Fig. 8.16**

| Input | | | | | Outputs | |
|---|---|---|---|---|---|---|
| $P_r$ | $C_r$ | CLK | S | R | $Q_{n+1}$ | $\overline{Q}_{n+1}$ |
| 0 | 0 | X | X | X | RACE | RACE |
| 0 | 1 | X | X | X | 1 | 0 |
| 1 | 0 | X | X | X | 0 | 1 |
| 1 | 1 | 0 | X | X | NC | NC |
| 1 | 1 | 1 | 0 | 0 | NC | NC |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | RACE | RACE |

**Table 8.5 : Truth Table of SR FF with preset and clear**

Scanned by CamScanner

Let's analyse working :

(1) $P_r = C_r = 0$, it gives RACE condition, because output of NAND3/4 = 1. So avoid this condition.

(2) $P_r = 1$, $C_r = 0$. This is ***RESET*** condition of FF.

As $C_r = 0$, $\bar{Q} = 1$. As input to NAND3 is $P_r = 1$, $\bar{Q} = 1$ and output of NAND1 is $Q = 0$.

(3) $P_r = 0$, $C_r = 1$. This is SET condition of FF.

As $P_r = 0$, $Q = 1$. As input to NAND4 is $C_r = 1$, $Q = 1$ and output of NAND2 = 1 forces $\bar{Q} = 0$

(4) $P_r = C_r = 1$. In this condition SR FF acts as normal and truth table matches with SR FF without preset/clear terminal.

**Synchronous Preset and Clear**

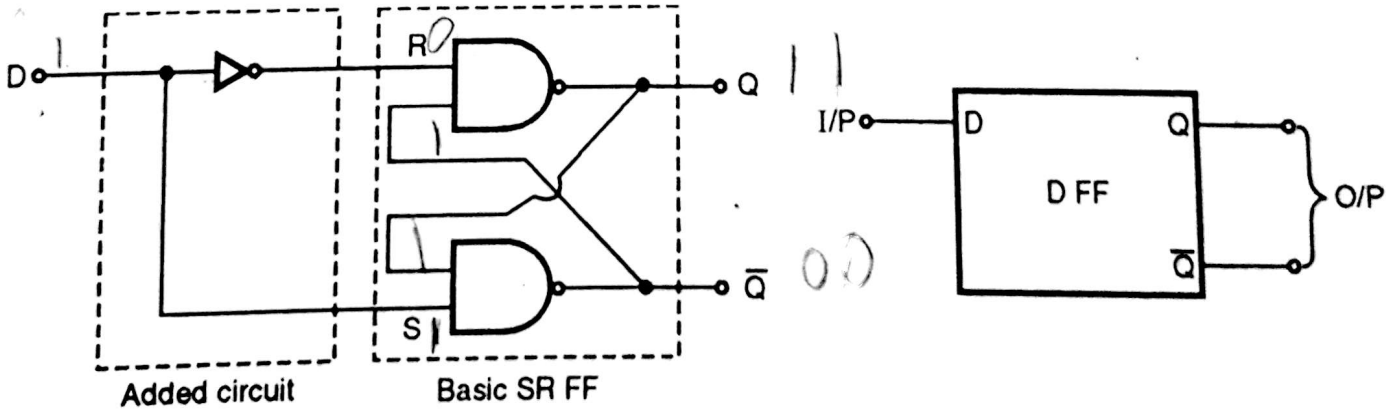
We will directly learn truth table for the same. SR FF with preset/clear (synchronous input).

| Inputs | | | | | Outputs | |
|---|---|---|---|---|---|---|
| $P_r$ | $C_r$ | CLK | S | R | $Q_{n+1}$ | $\bar{Q}_{n+1}$ |
| X | X | 0 | X | X | NC | NC |
| 0 | 0 | 1 | X | X | RACE | RACE |
| 0 | 1 | 1 | X | X | 0 | 1 |
| 1 | 0 | 1 | X | X | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | NC | NC |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | RACE | RACE |

**Table 8.6 : Truth Table of SR FF**

## 8.4 D FF (Delay Flip-Flop) :

To avoid possibility of race condition in SR FF, we will change the design of FF. The resulting latch is called 'D latch', as shown in Fig. 8.17.




**(a) D FF (Unclocked)** **(b) Unclocked D latch**

**Fig. 8.17**

Scanned by CamScanner

**Working of Unclocked D Latch**

**Case I :** If D = 0, R = 1 and S = 0. Therefore Q = 0, $\bar{Q}$ = 1.

**Case II :** If D = 1, R = 0 and S = 1. Therefore Q = 1, $\bar{Q}$ = 0.

**Note :** Output Q changes as D changes, irrespective of previous state.

From above analysis, we conclude that there is no possibility of occurring S = R = 0 or S = R = 1

| D input | $Q_{n+1}$ | $\bar{Q}_{n+1}$ |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

**Table 8.7 : Truth table of DFF**

Analyse truth table shown in Table 8.7, you will find that $Q_{n+1}$ is simply copy of D input (data input). So functionally D FF produces output which is equal to input but with some 'Delay'. Therefore it is called as ***delay FF***. Most widely used in computer system.

Now we will see clocked (level and edge triggered) D FF. For you, it is self explanatory.

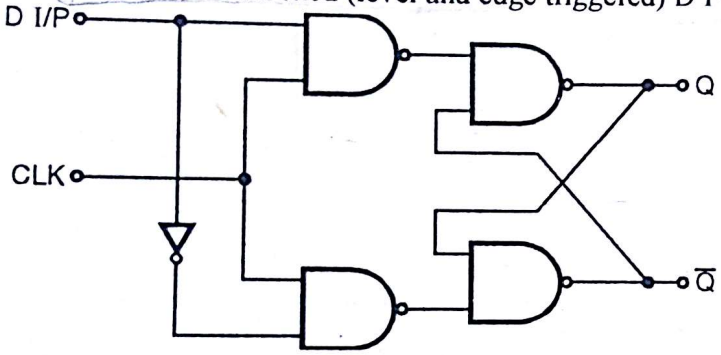



**Fig. 8.18 (a) : Level triggered D FF**

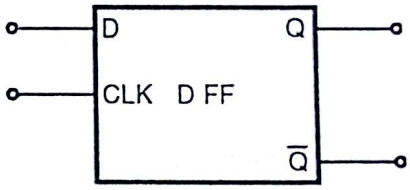



**Fig. 8.18 (b) : Symbol**

| Input | | Output | |
|---|---|---|---|
| CLK | D | $Q_{n+1}$ | $\bar{Q}_{n+1}$ |
| 0 | X | NC ($Q_n$) | NC ($\bar{Q}_n$) |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Fig. 8.18(c) : Truth Table**

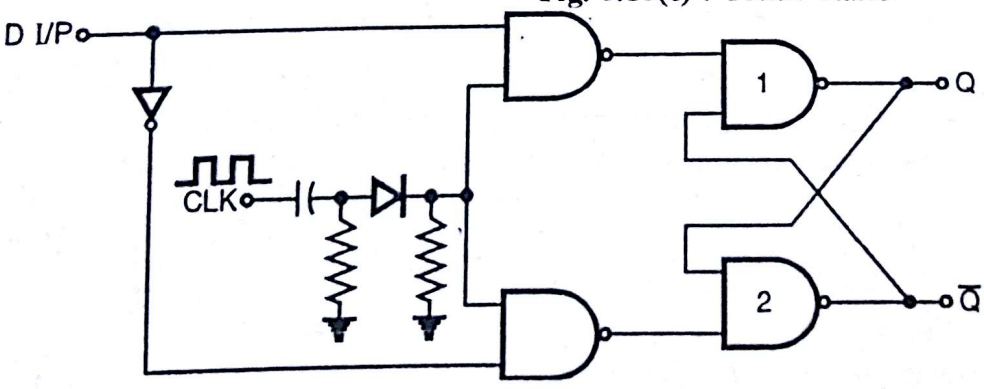



**Fig. 8.19(a) : Edge triggered D FF**

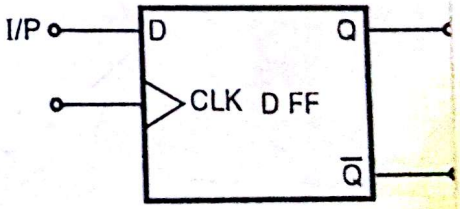



**Fig. 8.19(b) : Symbol**

Scanned by CamScanner

| Input | | Output | |
|---|---|---|---|
| CLK | D | $Q_{n+1}$ | $\bar{Q}_{n+1}$ |
| 0 | X | NC | NC |
| 1 | X | NC | NC |
| ↓ | X | NC | NC |
| ↑ | 0 | 0 | 1 |
| ↑ | 1 | 1 | 0 |

**Fig. 8.19(c) : Truth table**

**State table**

| Present State | Next State | |
|---|---|---|
| Q(n) | D = 0 | D = 1 |
| 0 | 0 | 1 |
| 1 | 0 | 1 |

**Table 8.8 : State table for D FF**

**State diagram**

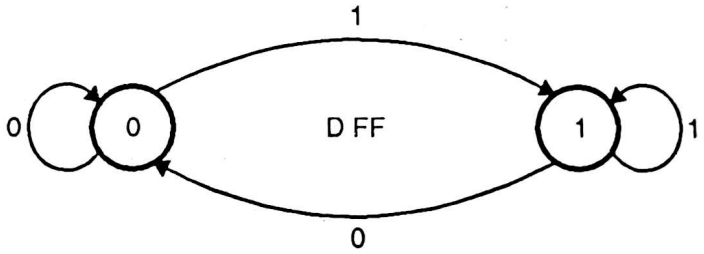



**Fig. 8.20 : D FF state diagram**

| Previous state | to | Next state | If | D |
|---|---|---|---|---|
| 0 | | 0 | | 0 |
| 0 | | 1 | | 1 |
| 1 | | 0 | | 0 |
| 1 | | 1 | | 1 |

**Table 8.9 : Excitation table for D FF**

**State equation**

$$Q(n+1) = D$$

**Ex. 3 :** Draw output waveform for given D input, for positive edge triggered D FF.

Scanned by CamScanner

**Soln. :**

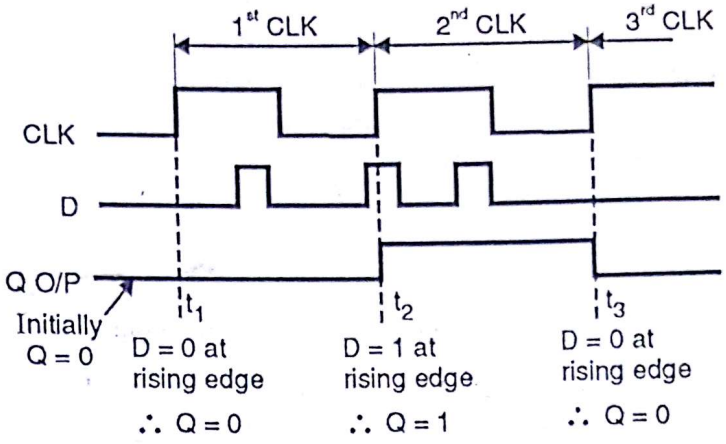



**Fig. Ex. 8.3**

Available D FF chips are :

(1) Positive edge triggered D type FF – IC 7474
(2) Positive level triggered D type FF – IC 7475 (transparent latch)
(3) Quad D flip-flop with clear – IC 74175
(4) Octal D latch – IC 74373

## 8.5 Flip-Flop Timing Consideration :

Manufacturer of flip-flop IC will specify several important timing parameters and characteristics that must be considered before FF is used in any circuit.

For explaining the same we will use D FF.

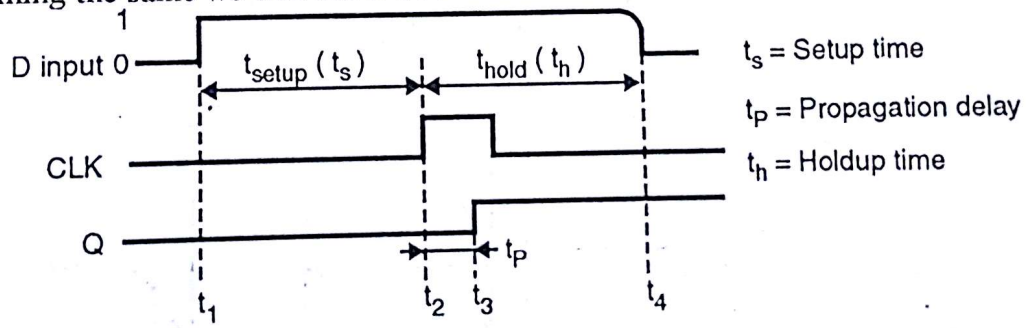



**Fig. 8.21**

**Propagation Delay (Refer Fig. 8.21)**

In IC's we use diodes and transistors. Transistor is operated either in saturation or cut off. Transistor takes time to switch from saturation to cutoff or V.V. Same way diode also takes small time to turn ON/OFF. This switching time is in nanosecond. But is main cause for propagation delay ($t_p$).

**Definition :** It is the amount of time it takes for output of gate or FF to change state after the input changes or CLK edge hits. Typical value = 10 – 20 η sec.

This time should be considered when digital circuit used in high speed circuit. Propagation delay is measured for both, LOW to HIGH and HIGH to LOW transition.

$t_p = t_3 - t_2$

Scanned by CamScanner

$t_{PLH}$ → From CLK to Q (LH → LOW to HIGH)

$t_{PHL}$ → From CLK to Q (HL → HIGH to LOW)

$t_{PLH}$ → $\overline{PRE}$ to Q } $\overline{PRE}$ – Preset

$t_{PHL}$ → $\overline{CLR}$ to Q } $\overline{CLR}$ – Clear

**Setup Time ($t_s$) ($t_{setup}$)** ➥ [ASKED IN EXAM - DEC. 99, DEC. 98 !!! ]

The setup time ($t_2 - t_1$) is the minimum length of time input should be present before CLK edge arrives. The main cause of setup time is stray capacitance at the input side. If this time requirement is not met, the FF may not respond reliably when CLK edge arrives. Typical timing is from 5 to 40 η sec.

**Holdup Time ($t_h$) ($t_{hold}$)** ➥ [ASKED IN EXAM - DEC. 99, DEC. 98 !!! ]

The holdup time ($t_4 - t_2$) is minimum length of time, input must be present after CLK edge has struck. The main cause of holding input is because of switching time of internal transistors and diodes. If this time is not met, one cannot assure about switching of internal circuit and FF will not work properly.

Typical holdup time is ranging from 0 to 10 η sec.

**Maximum CLK Frequency ($f_{max}$)**

This is the maximum CLK frequency that can be applied to CLK input of FF, for proper and stable operation.

$$f_{max} = 1/t_c = \frac{1}{t_{setup} + t_{pd(FF)} + t_{ns}}$$

where, $t_c$ = Clock period

$t_{ns}$ =Propagation delay of next state decoder

$t_{setup}$ =Setup time

$t_{pd\ (FF)}$ =Flip flop propagation delay

Refer Fig. 8.22.

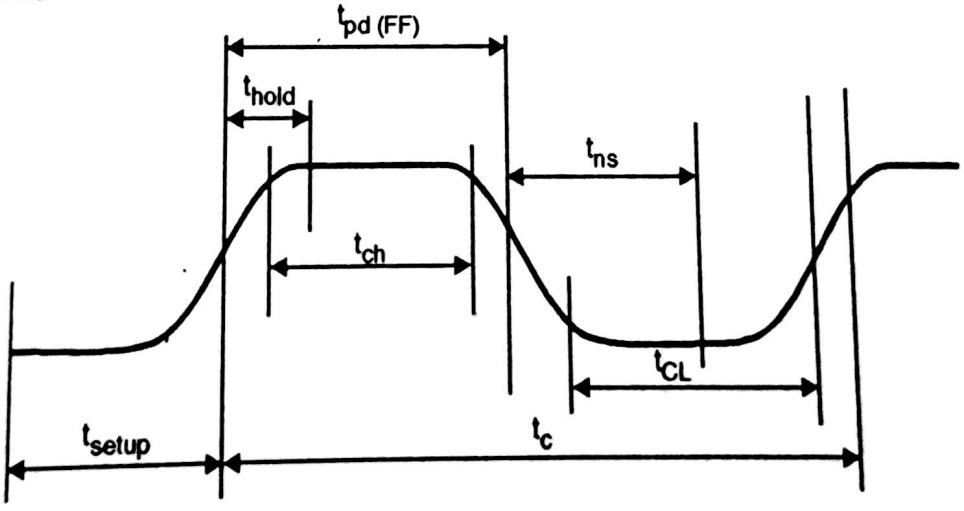



**Fig. 8.22**

**Clock Pulse HIGH and LOW Time**

**Refer Fig. 8.22. The manufacturer specifies minimum time duration that the CLK signal must remain LOW before it goes HIGH, it is called as CLK pulse low time ($t_{CL}$ or $t_{WCL}$) and minimum time duration**

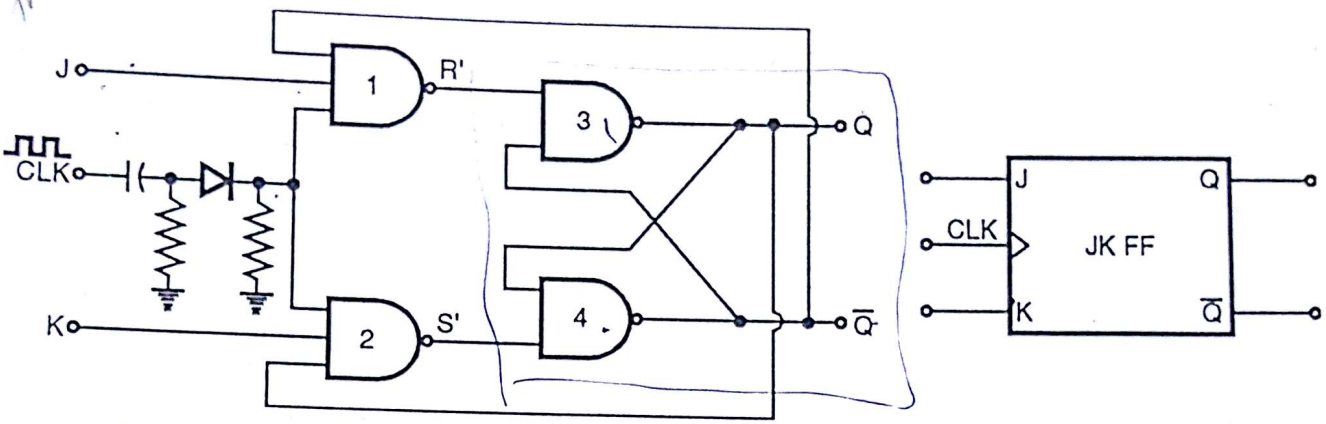
Scanned by CamScanner

that the CLK signal must remain HIGH before it returns LOW, is called CLK pulse high time ($t_{CH}$ or $t_{WCH}$) typical timing for $t_{W(H)}$ and $t_{W(L)}$ are 15 η sec. to 100 η sec.

**Asynchronous Active Pulse Width**

As we have seen in section 8.3.5 preset and clear are asynchronous inputs. For the inputs manufacturer specify minimum time duration. $t_{W(L)} \longrightarrow$ Preset or clear (For active low asynchronous input.).

## 8.6 JK Flip-Flop :




(a) Positive edge triggered JK FF (b) Symbol

**Fig. 8.23**

As shown in Fig. 8.23(a) NAND 3/4 forms simple SR FF. Inputs to the NAND1 are J, $\bar{Q}$ and CLK (edge). Inputs to NAND2 are K, Q and CLK (edge). Now let's see how JK FF works and derive truth table.

**Case I** : When CLK is at ***HIGH*** or ***LOW*** level, output Q and $\bar{Q}$ remains in previous state. Because NAND1/2 produces '1' output, irrespective of Q, $\bar{Q}$, J and K inputs, as input to NAND1/2 is '0', after differentiator circuit.

**Case II** : When CLK = ↓ i.e. falling edge, output Q and $\bar{Q}$ are unchanged.

**Case III** : When CLK = ↑ and J = 0, K = 0,

S' = R' = 1. ∴ output does not change.

**Conclusion :** Above three cases are called ***Inactive*** condition of JK FF.

**Case IV** : CLK = ↑, J = 0 and K = 1.

Let's consider previous state of Q = 1 and $\bar{Q}$ = 0

$$R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{0 \cdot 0 \cdot 1} = 1.$$

$$S' = \overline{K \cdot Q \cdot CLK} = \overline{1 \cdot 1 \cdot 1} = 0.$$

∴ Q changes from 1 to 0. Q = 0, $\bar{Q}$ = 1 (Refer Table 8.8(b) of SR FF). We will again start analysis because Q and $\bar{Q}$ has been changed.

$$\text{Now, } R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{0 \cdot 1 \cdot 1} = 1$$

$$S' = \overline{K \cdot Q \cdot CLK} = \overline{1 \cdot 0 \cdot 1} = 1$$

Q and $\bar{Q}$ will remain unchanged unless and until next JK input is given with CLK. This state is stable ***'RESET'*** condition of JK FF.

**Conclusion :** When J = 0, K = 1 and CLK hits, irrespective of previous condition, Q = 0 and $\bar{Q}$ = 1.

Scanned by CamScanner

**Case V** : CLK = ↑, J = 1, K = 0

Let's consider previous state of Q = 0 and $\bar{Q} = 1$

$R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{1 \cdot 1 \cdot 1} = 0$

$S' = \overline{K \cdot Q \cdot CLK} = \overline{0 \cdot 0 \cdot 1} = 1$

∴ Q changes from 0 to 1. Q = 1, $\bar{Q} = 0$

Now $R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{1 \cdot 0 \cdot 1} = 1$

$S' = \overline{K \cdot Q \cdot CLK} = \overline{0 \cdot 1 \cdot 1} = 1$

∴ Q and $\bar{Q}$ will remain unchanged. This state is stable ***'SET'*** condition of JK FF.

**Conclusion :** When J = 1, K = 0 and CLK hits, irrespective of previous condition Q = 1 and $\bar{Q} = 0$.

**Case VI** : When CLK = ↑, J = K = 1, this condition is more interesting. Here we are going to consider 2 CLK pulses.

(i) Let's say previous condition of Q = 1, $\bar{Q} = 0$

∴ $R' = \overline{J.\bar{Q}.CLK} = \overline{1 \cdot 0 \cdot 1} = 1$

∴ $S' = \overline{K.Q.CLK} = \overline{1 \cdot 1 \cdot 1} = 0$

∴ $Q_{n+1} = 0$ and $\overline{Q_{n+1}} = 1$ ($Q_{n+1} = \bar{Q}_n$, $Q_{n+1} = \bar{1} = 0$)

(ii) Now let's say previous condition of Q = 0, $\bar{Q} = 1$ (If you note $Q_{n+1}$ of (i) = $Q_n$).

∴ $R' = \overline{J \cdot \bar{Q} \cdot CLK} = \overline{1 \cdot 1 \cdot 1} = 0$

∴ $S' = \overline{K \cdot Q \cdot CLK} = \overline{1 \cdot 0 \cdot 1} = 1$

∴ $Q_{n+1} = 1$, $\overline{Q_{n+1}} = 0$ ($Q_{n+1} = \bar{Q}_n = \bar{0} = 1$)

From above two cases (i) and (ii) we conclude that when J = K = 1 and CLK hits $Q_{n+1} = \bar{Q}_n$ (means output toggles).

**Conclusion :** This condition of JK FF gives us ***TOGGLING*** mode. Now let's summarize everything and make truth table :

| Inputs | | | Outputs | | |
|---|---|---|---|---|---|
| **CLK** | **J** | **K** | $Q_{n+1}$ | $\overline{Q_{n+1}}$ | |
| 0 | X | X | NC | NC | Case I |
| 1 | X | X | NC | NC | Case I |
| ↓ | X | X | NC | NC | Case II |
| ↑ | 0 | 0 | NC | NC | Case III |
| ↑ | 0 | 1 | 0 | 1 | Case IV |
| ↑ | 1 | 0 | 1 | 0 | Case V |
| ↑ | 1 | 1 | $\bar{Q}_n$ | $Q_n$ | Case VI |

Scanned by CamScanner

**State table**

| Present state Q (n) | Next state Q (n + 1) | | | |
|---|---|---|---|---|
| | JK = 00 | JK = 01 | JK = 10 | JK = 11 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |

**Table 8.10 : State table for JK flip-flop**

**State diagram**

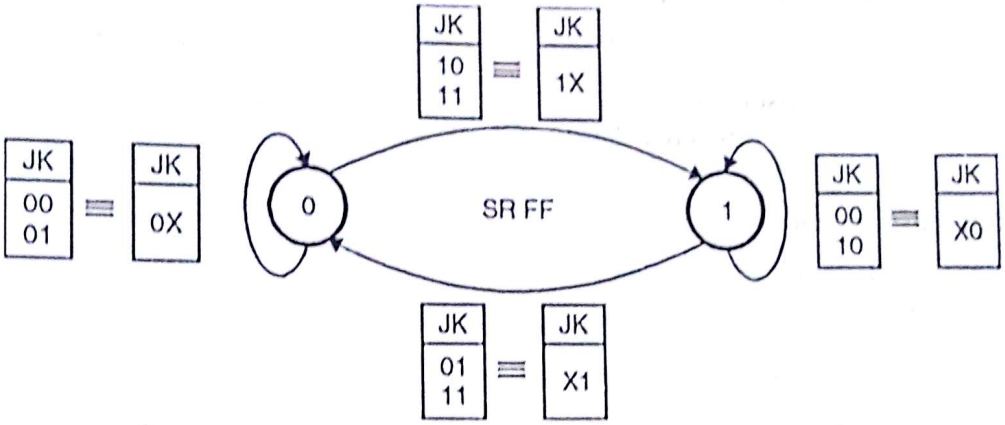



**Fig. 8.24 : State diagram for JK-FF**

| Previous state | to | Next state | If | J | K |
|---|---|---|---|---|---|
| 0 | | 0 | | 0 | X |
| 0 | | 1 | | 1 | X |
| 1 | | 0 | | X | 1 |
| 1 | | 1 | | X | 0 |

**Table 8.11 : Excitation table for JK–FF**

**State Equation**

$$Q(n+1) = J\,\overline{Q(n)} + \overline{K}\,Q(n)$$

**Ex. 4 :** Determine output Q for Fig. Ex. 8.4 :

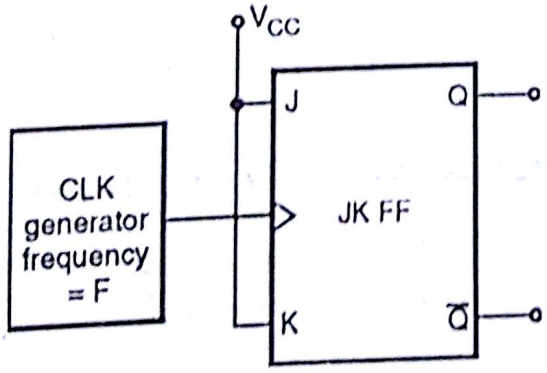



**Fig. Ex. 8.4**

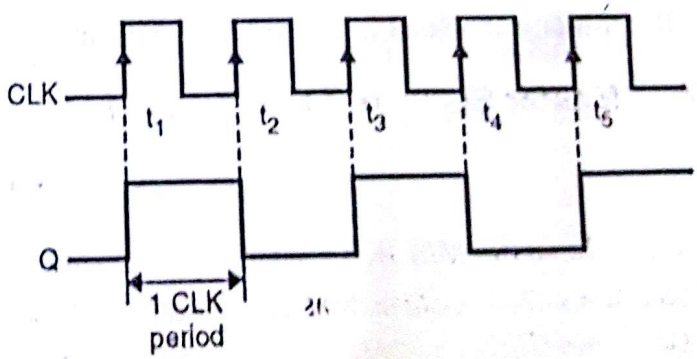



**Fig. Ex. 8.4(a)**

Scanned by CamScanner

**Soln. :** When J = K = 1, output Q toggles.

Initially Q = 0.

When, 1st CLK edge hits $Q_{n+1} = \bar{Q}_n$ $Q_{n+1} = \bar{0} = 1$ $(t_1)$

2nd CLK edge hits $Q_{n+1} = \bar{Q}_n$ $Q_{n+1} = \bar{1} = 0$ $(t_2)$

3rd CLK edge hits $Q_{n+1} = \bar{Q}_n$ $Q_{n+1} = \bar{0} = 1$ $(t_3)$

and the chain will repeat.

The frequency of the Q output is F/2. Therefore this circuit is called as ÷ 2 circuit.

### 8.6.1 Race Condition in JK FF (Race Around Condition) :

➥ **[ASKED IN EXAM - DEC. 96, MAY 97, DEC. 97 !!! ]**

We have started our discussion on JK FF, directly with positive edge triggered type. But how JK is going to behave if FF is level triggered ? Here we are going to see interesting RACE AROUND phenomena.

Remember three important points :

(1) When J = K = 1, JK FF output Q toggles

(2) When FF is level triggered, FF is active till ***active level*** is present

(3) Output Q and $\bar{Q}$ is fed back to input NAND gate in JK FF.

Refer Fig. 8.23(a) (Remove differentiator circuit, then analyse). Draw output Q when J = K = 1 for level triggered JK FF.

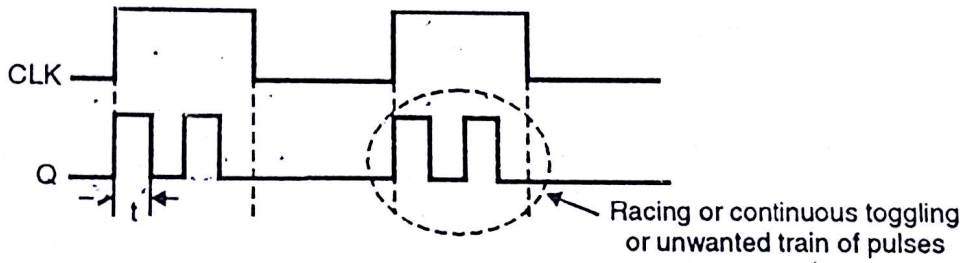



**Fig. 8.25**

Let's consider initially Q = 0. So when CLK = 1, Q changes state, Q = 1. Now new state of Q propagate through NAND1, NAND2 and will get settled and satisfy setup/hold up time for NAND2/3. The total time of propagation is represented as t in Fig. 8.25. After time t, as CLK = J = K = 1. Q again toggles. ∴ Q = 0. This chain will continue till CLK = 0.

We are going to get ***unwanted train of pulses*** when CLK = J = K = 1. This is drawback of level triggered JK FF.

But if user wants level triggered JK FF without racing, then the answer is MASTER SLAVE JK FF.

## 8.7 Master Slave JK FF (M/S JK FF) :

➥ **[ASKED IN EXAM - DEC. 96, MAY 98, !!! ]**

Fig. 8.26 shows M/S JK FF which is combination of two clocked latches; first is called as ***Master*** and second is called as ***Slave***. Master is positively clocked and slave is negatively clocked i.e. :

(1) When CLK = 1 Master is active, slave is inactive

(2) When CLK = 0 Master is inactive, slave is active.

**Table 8.12 : State table of TFF**

Scanned by CamScanner

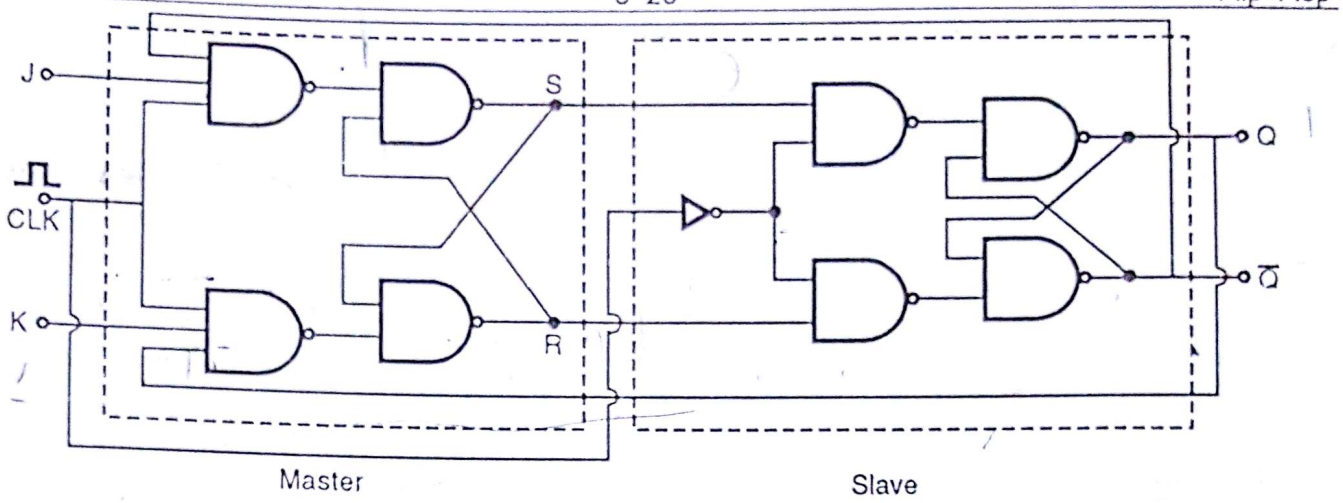



Fig. 8.26 : Master slave JK FF

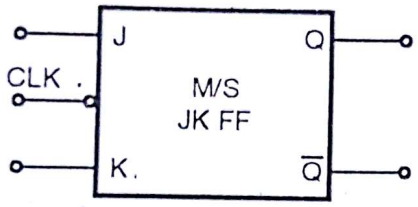



Fig. 8.27 : Symbol

| Inputs | | | Outputs | |
|---|---|---|---|---|
| CLK | J | K | $Q_{n+1}$ | $\bar{Q}_{n+1}$ |
| X | 0 | 0 | $Q_n$ (NC) | $\bar{Q}_n$ (NC) |
| ⎍ | 0 | 1 | 0 | 1 |
| ⎍ | 1 | 0 | 1 | 0 |
| ⎍ | 1 | 1 | Toggle ($\bar{Q}_n$) | Toggle ($Q_n$) |

Fig. 8.28 : Truth Table

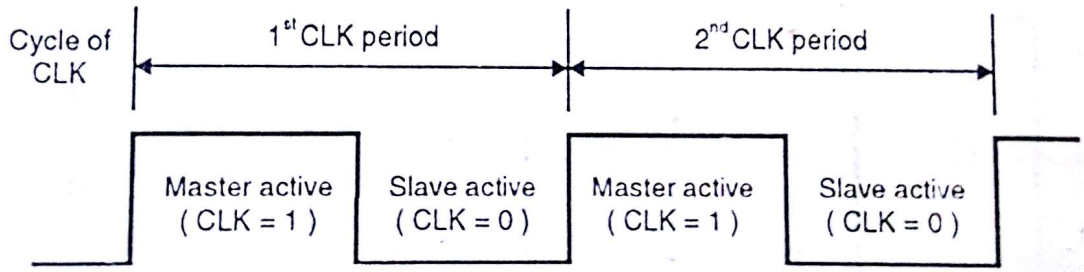



Fig. 8.29

You should remember mainly above two points and one more point that ***whatever master does, slave copies it, but in next half cycle of CLK.***

**Working**

**Case I :** $J = K = 0$ Previous state $Q_n = 0$, $\bar{Q}_n = 1$

(i) $CLK = 1$, $\therefore$ Master active, Slave inactive

$\therefore$ output of Master FF, $S = Q_n$, $R = \bar{Q}_n$

(ii) $CLK = 0$, Master inactive, Slave active

$\therefore$ output of Slave FF, $Q = Q_n$, $\bar{Q} = \bar{Q}_n$

**Conclusion : No change condition.**

Scanned by CamScanner

**Case II :** $J = 0, K = 1, Q_n = 1, \bar{Q}_n = 0$

(i) CLK = 1, Master active, Slave inactive

∴ output of Master, S = 0 R = 1.

(ii) CLK = 0, Master inactive, Slave active

∴ output of Slave, Q = 0. $\bar{Q} = 1 \rightarrow Q_{n+1}$ changed .

(iii) CLK = 1, Master active, Slave inactive

∴ output of Master, S = 0 R = 1

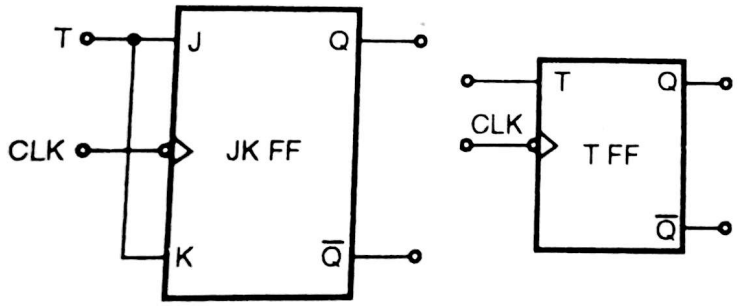
∴ Now the state is stable as (i) and (iii) are same.

Available J – K and M/S JK FF

| | | |
|---|---|---|
| (1) | Edge triggered JK FF | – 7470 |
| (2) | JK Master Slave FF | – 7472 |
| (3) | Dual JK M/S FF | – 7473 |
| (4) | Master Slave JK FF | – 7476 |
| (5) | JK M/S FF | – 74104/105 |
| (6) | Dual JK positive edge triggered FF | – 74109 |

## 8.8 T FF (Toggle FF) :

The type T FF changes state (toggles) for each CLK pulse, if $T_{input} = 1$. In J – K FF, if J = K we get type T FF. It has only one input, referred to as T input.




(a) JK converted to T (b) Logic symbol

| CLK | T | $Q_{n+1}$ | $\bar{Q}_{n+1}$ |
|---|---|---|---|
| ↓ | 0 | $Q_n$ | $\bar{Q}_n$ |
| ↓ | 1 | $\bar{Q}_n$ | $Q_n$ |
| ↑ | X | $Q_n$ | $\bar{Q}_n$ |
| 1 | X | $Q_n$ | $\bar{Q}_n$ |
| 0 | X | $Q_n$ | $\bar{Q}_n$ |

(c) Truth Table

**Fig. 8.30**

**State table**

| Previous state Q(n) | Next State Q(n + 1) | |
|---|---|---|
| | T = 0 | T = 1 |
| 0 | 0 | 1 |
| 1 | 1 | 0 |

**Table 8.12 : State table of TFF**

Scanned by CamScanner

**State diagram**

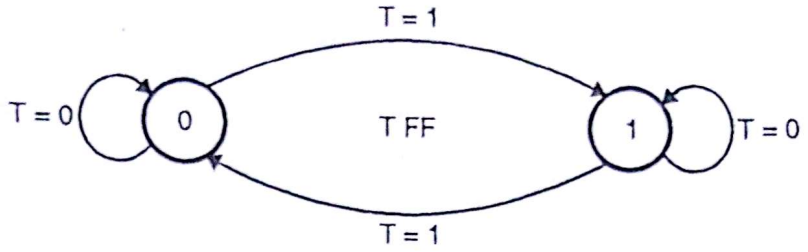



**Fig. 8.31 : State diagram of TFF**

| Present state | to | Next state | If T |
|---|---|---|---|
| 0 | | 0 | 0 |
| 0 | | 1 | 1 |
| 1 | | 0 | 1 |
| 1 | | 1 | 0 |

**Table 8.13 : Excitation table of TFF**

**State Equation**

$$Q(n+1) = \overline{T} \cdot Q(n) + \overline{Q(n)} \cdot T$$

---

**Ex. 5 :** Draw output waveform for Fig. Ex. 8.5 :

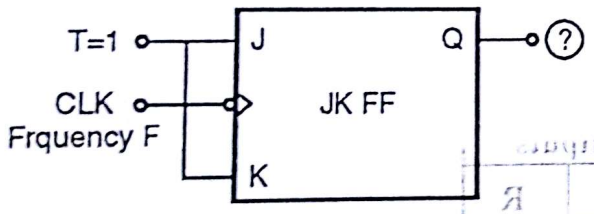



**Fig. Ex. 8.5**

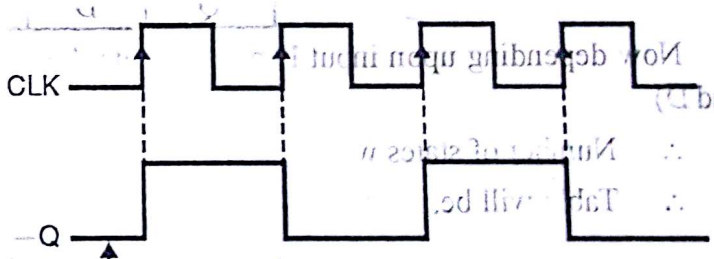



**Fig. Ex. 8.5(a)**

**Soln. :** Refer Fig. Ex. 8.5(a).

## 8.9 Conversion from one Type of FF to Another :

If we want to convert one type of FF to another type of FF, there is systematic approach implementing the same using K-Map. Fig. 8.32 shows generalized model for conversion from one type of FF to another.

Scanned by CamScanner

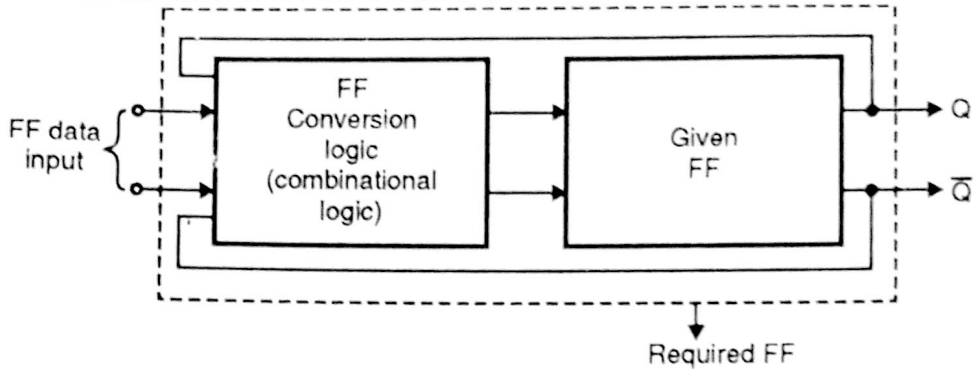



**Fig. 8.32 : General model of conversion from one FF to another**

As shown in Fig. 8.32 required FF is combination of given FF and FF conversion logic. FF conversion logic is nothing but combinational circuit.

To design this we require excitation table. Here one should make truth table which contains.

(1) Input lines :

This contains data inputs of required FF and Q as an input.

(2) Output lines :

Consider input terminal of the given FF as output.

For example, let's say we want to convert SR FF to D FF. than input links will be D and Q. Output lines will be S and R. ∴ Table will be

| Inputs | | Outputs | |
|---|---|---|---|
| Q | D | S | R |

Now depending upon input lines you write down number of states. Presently input lines are two (Q and D)

∴ Number of states will be 4.

∴ Table will be,

| Inputs | | Outputs | |
|---|---|---|---|
| **Q** | **D** | **S** | **R** |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

Now next step is very important you have to ask question to yourself, ***that previous output is this (Q), but because of new input (D) , what will be new state (say i). Then to change from previous output (Q) to new state (i), what should be state of output terminals (i.e. S and R).***

The states you can find from excitation table.

Let's take the same example.

∴ Previous output is '0' (Q), but if new input D = 0 new state will be i = 0.

∴ Transition is from Previous state Q = 0 to next state i = 0

∴ S = 0, R = X (excitation table) ; where X = don't care condition

∴ Previous output is ' 0' (Q) but if new input D = 1, new state will be i = 1.

Scanned by CamScanner

∴ Transition is from pervious state Q = 0 to new state i = 1.

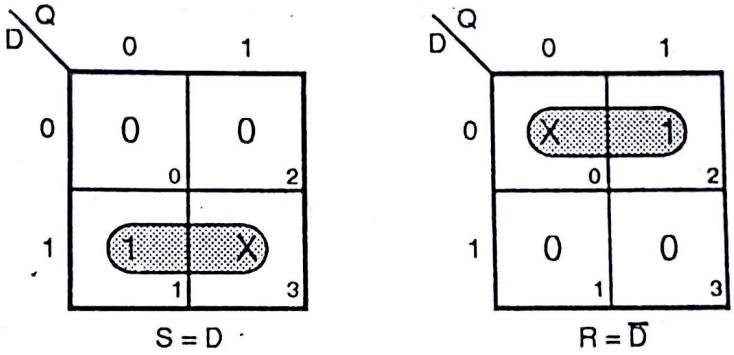
∴ S = 1, R = 0.

Thus you switch over to next inputs and find SR outputs

∴ We get,

| Inputs | | Transition | Output | |
|---|---|---|---|---|
| Q | D | Q to i | S | R |
| 0 | 0 | 0 → 0 | 0 | X |
| 0 | 1 | 0 → 1 | 1 | 0 |
| 1 | 0 | 1 → 0 | 0 | 1 |
| 1 | 1 | 1 → 1 | X | 0 |

Now draw K-Map.




∴ **Circuit is,**

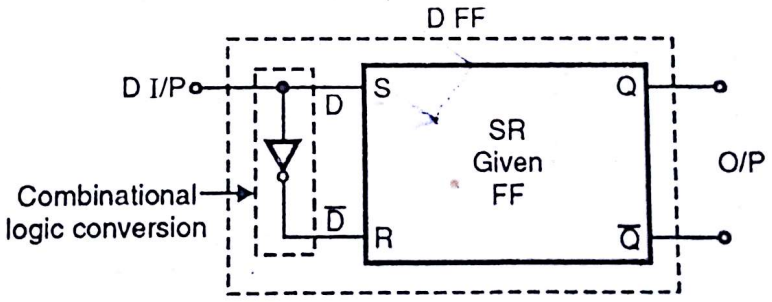



**Fig. 8.33 : Conversion of SR to D FF**

---

**Ex. 6 :** Represent following with the help of K-Map :

(i) JK FF to work as T FF

(ii) SR FF to work as T FF.

(iii) D FF to work as T FF.

**Soln. :** (i) JK to T

∴ **Table will be,**

Scanned by CamScanner

| Inputs | | Transition | Outputs | |
|---|---|---|---|---|
| Q | T | Q to i (new state) | J | K |
| 0 | 0 | 0 ⟶ 0 | 0 | X |
| 0 | 1 | 0 ⟶ 1 | 1 | X |
| 1 | 0 | 1 ⟶ 1 | X | 0 |
| 1 | 1 | 1 ⟶ 0 | X | 1 |

X = Don't care

**K-Map :**

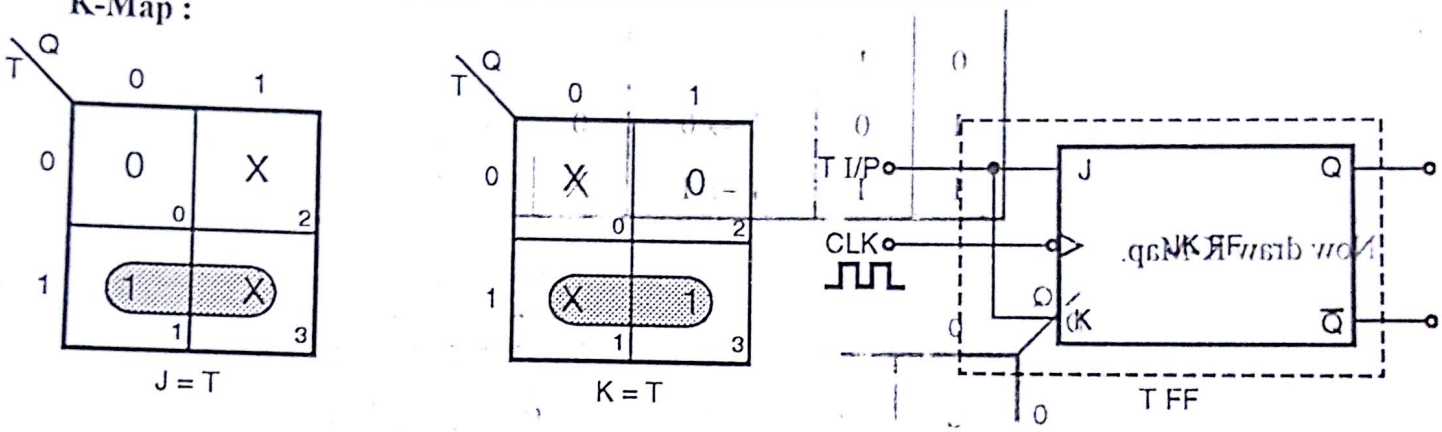


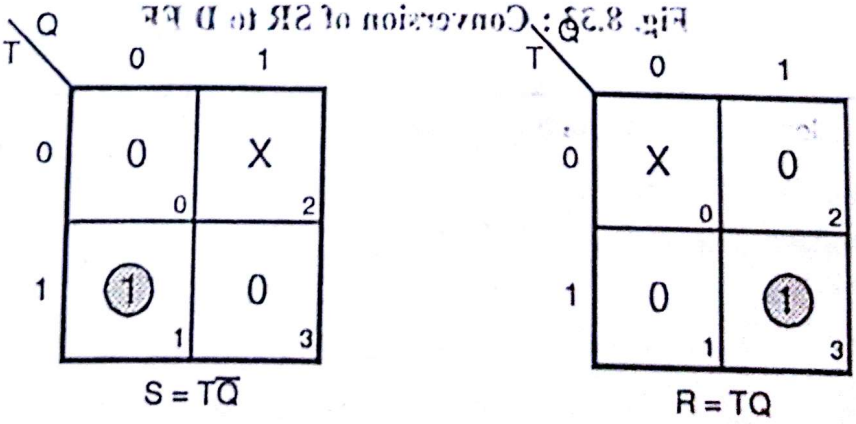

**Fig. Ex. 8.6(a)**

(ii) SR to T ∴ Table will be, **(Dec. 97, June 99, 8 Marks)**

| Inputs | | Transition | Outputs | |
|---|---|---|---|---|
| Q | T | Q to i | S | R |
| 0 | 0 | 0 ⟶ 0 | 0 | X |
| 0 | 1 | 0 ⟶ 1 | 1 | X |
| 1 | 0 | 1 ⟶ 1 | X | 0 |
| 1 | 1 | 1 ⟶ 0 | 0 | 1 |

X = Don't care

∴ **K-Map :**




Scanned by CamScanner

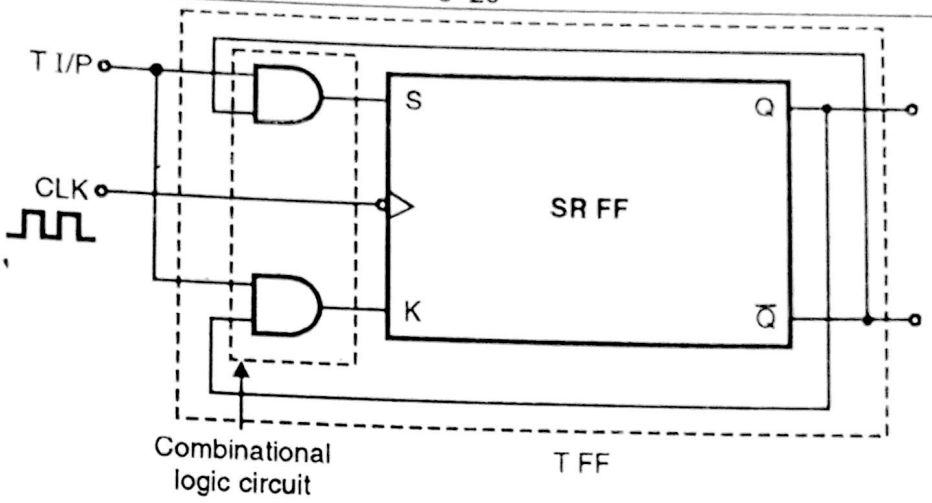



**Fig. Ex. 8.6(b)**

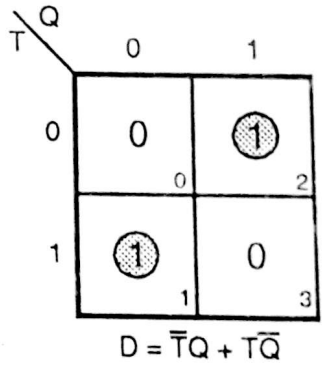
(iii) D FF to T FF

**(Dec. 96, 8 Marks, Dec. 98, 99, 6 Marks)**

∴ Table is,

| Inputs | | Transition | O/P |
|---|---|---|---|
| Q | T | Q to i (new state) | D |
| 0 | 0 | 0 → 0 | 0 |
| 0 | 1 | 0 → 1 | 1 |
| 1 | 0 | 1 → 1 | 1 |
| 1 | 1 | 1 → 0 | 0 |

**K-Map :**




$D = \overline{T}Q + T\overline{Q}$

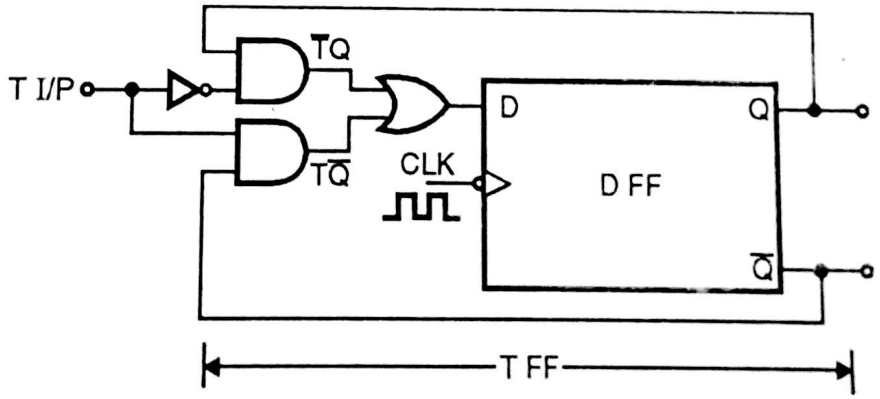



**Fig. Ex. 8.6(c)**

---

**Ex. 7 :** Represent the following with help of K-Map :

(i) SR to JK (ii) JK to D.

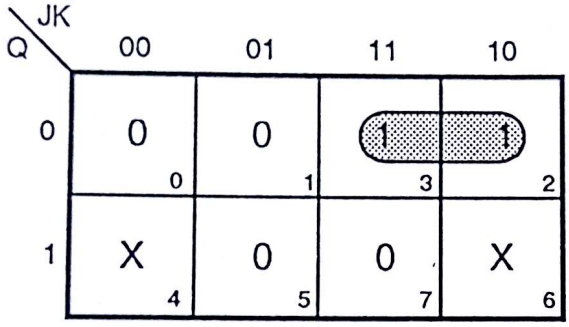
**Soln. : (i) SR to JK :**

Scanned by CamScanner

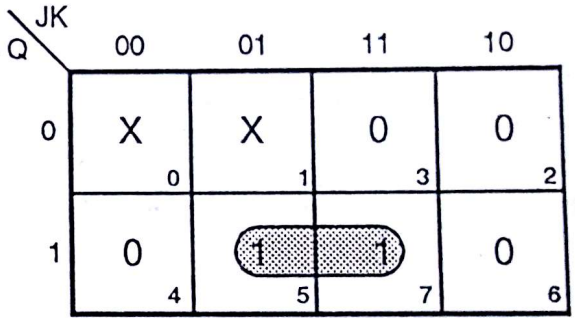
∴ Table is,

| | | | Transition | Outputs | |
|---|---|---|---|---|---|
| Q | J | K | Q to i | S | R |
| 0 | 0 | 0 | 0 ⟶ 0 | 0 | X |
| 0 | 0 | 1 | 0 ⟶ 0 | 0 | X |
| 0 | 1 | 0 | 0 ⟶ 1 | 1 | 0 |
| 0 | 1 | 1 | 0 ⟶ 1 | 1 | 0 |
| 1 | 0 | 0 | 0 ⟶ 1 | X | 0 |
| 1 | 0 | 1 | 1 ⟶ 0 | 1 | 0 |
| 1 | 1 | 0 | 1 ⟶ 1 | X | 0 |
| 1 | 1 | 1 | 1 ⟶ 0 | 0 | 1 |

∴ **K-Map :**




$S = J\overline{Q}$




$R = KQ$

∴ **Circuit :**

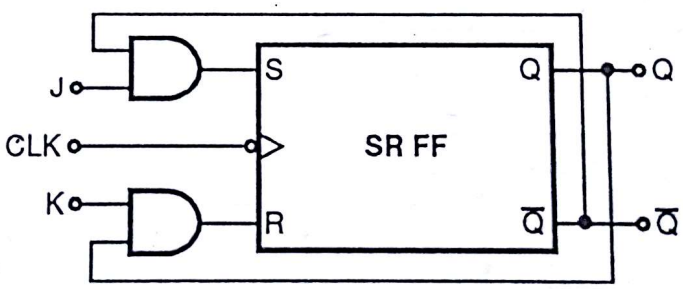



**Fig. Ex. 8.7(a)**

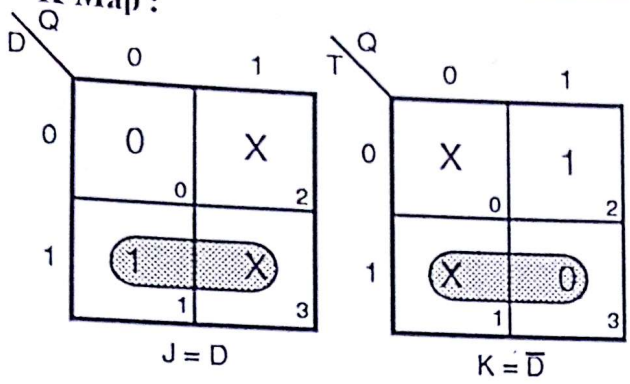
**(ii) JK to D :**

∴ Table is,

| Inputs | | Transition | Outputs | |
|---|---|---|---|---|
| Q | D | Q to i | J | K |
| 0 | 0 | 0 → 0 | 0 | X |
| 0 | 1 | 0 → 1 | 1 | X |
| 1 | 0 | 1 → 0 | X | 1 |
| 1 | 1 | 1 → 1 | X | 0 |

Scanned by CamScanner

**K-Map :**




**Circuit :**

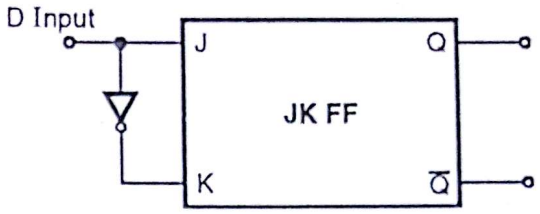



**Fig. Ex. 8.7(b)**

**Ex. 8 :** Show that following circuit acts as T type FF :

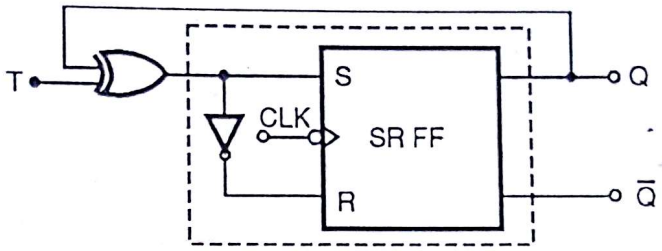



**Fig. Ex. 8.8**

**Soln. :** The circuit can be redrawn to analyse it easily. As shown in Fig. Ex. 8.8, dotted part is nothing but –ve edge triggered D type FF.

∴ Circuit becomes,

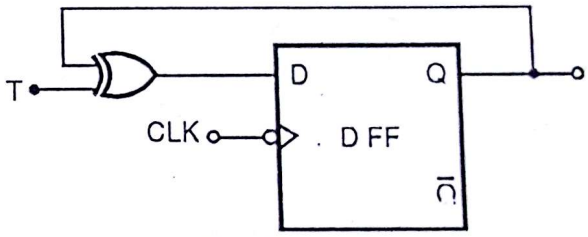



**Fig. Ex. 8.8(a)**

To analyse circuit refresh truth table of D FF as well as EX–OR gate. We will make truth table for same.

| Input | | | | Output | |
|---|---|---|---|---|---|
| **T input** | **CLK** | $Q_n$ | $D = (T \oplus Q_n)$ | $Q_{n+1}$ | |
| 0 | ↓ | 0 | 0 | $0 \Rightarrow (Q_n)$ | Part I |
| 0 | ↓ | 1 | 1 | $1 \Rightarrow (Q_n)$ | |
| 1 | ↓ | 0 | 1 | $1\ (\overline{Q}_n)$ | Part II |
| 1 | ↓ | 1 | 0 | $0(\overline{Q}_n)$ | |

Observe part I, in above table, As shown when T = 0, even though we apply clock pulse output remains as it is i.e. $Q_n$ (previous state).

Observe part II, As T = 1, for each clock edge output is complement of previous state means output toggles at each CLK edge.

∴ Truth table is exactly matching with T FF truth table.

Scanned by CamScanner

## 8.10 Applications of Flip-Flops :

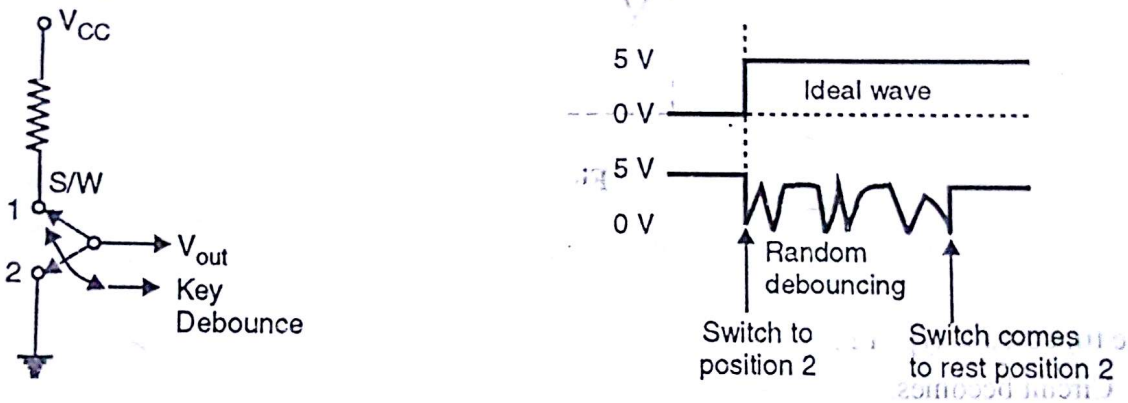
Until now we have studied all types of FF, symbols and their truth table. It is obvious to know the application of FF. FF can be used in :

(1) Counter/Timer,
(2) Memory element,
(3) Register and
(4) Eliminating keyboard debouncing etc.

Application (1) to (3) we are going to cover in next chapters. Now concentrate on application (4).

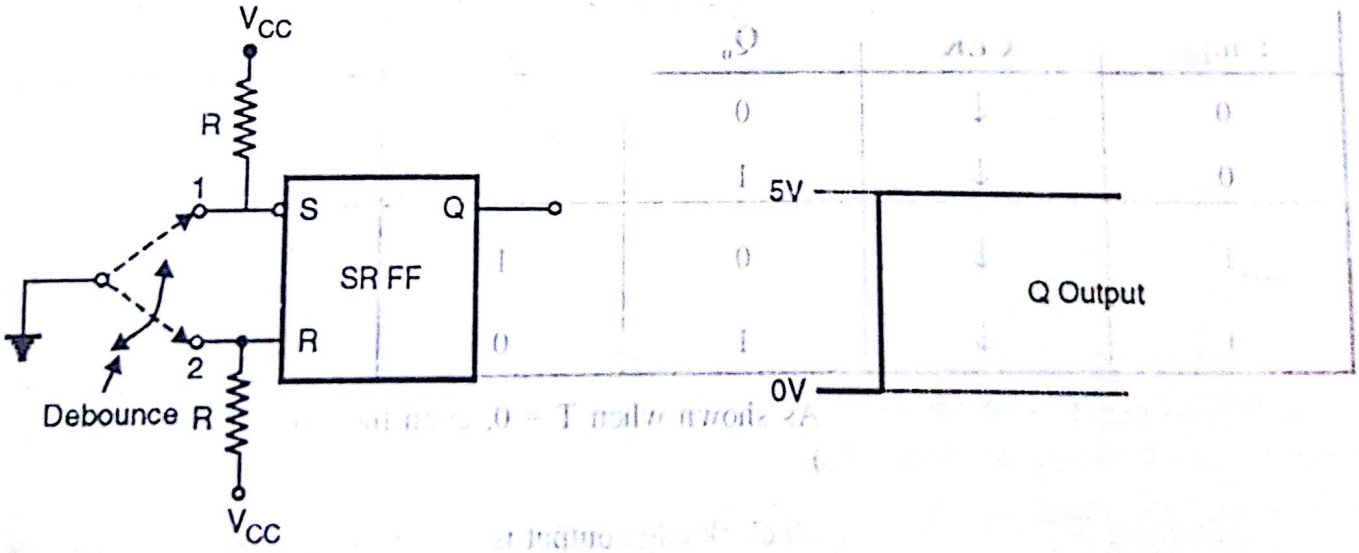
### 8.10.1 Elimination of Keyboard Debounce :

As we have seen that in digital electronics we require '0' and '1' logic. Same can be generated either through oscillator circuit or switch. At student level switch is most commonly used. The circuit is as shown in Fig. 8.34(a).




(a) Switch circuit (b) Switch to position

Fig. 8.34




(a) (b)

Fig. 8.35

Scanned by CamScanner

As shown in Fig. 8.34, (b) waveform should change from 0 to 5V (or v.v) instantly and should get settle down. But it is virtually impossible to obtain clear voltage transition from mechanical switch, because of ***contact debounce*** effect. Contact debounce means switch makes and breaks contact, before coming to rest on contact point. The multiple transition last only for few milliseconds, but for many application this is not acceptable. Answer to the problem is simple SR FF.

**Analyse working of circuit**

Initially key is resting on position 1,

$\therefore$ S = 0, R = 1, $\therefore$ Q = 0.

When switch is moved to position 2, S = 1 and R = 0, as the switch first makes the contact.

$\therefore$ Q = 1 (refer Fig. 8.32(b)).

Now if switch debounces off contact 2, S = R = 1,

$\therefore$ $Q = Q_n$ i.e. No change in output.

$\therefore$ Q = 1. After some make and break switch will come to rest on contact 2. So problem is solved.

---

**Ex. 9 :** An edge triggered D flip-flop can be made to operate in the toggle mode by connecting it as shown in Fig. Ex. 8.9. Assume that Q = 0, initially and determine the Q waveform.

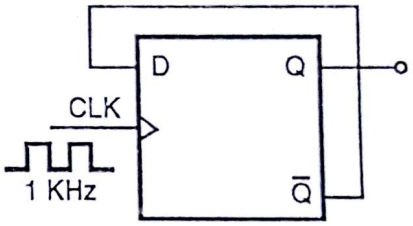



Fig. Ex. 8.9

**Soln. :** Initially Q = 0, Now we will prepare table.

| CLK | D | $Q_n$ | $\bar{Q}_n$ | $Q_{n+1}$ |
|---|---|---|---|---|
| $\uparrow$ ($1^{st}$) | 1 | 0 | 1 | 1 |
| $\uparrow$ ($2^{nd}$) | 0 | 1 | 0 | 0 |
| $\uparrow$ ($3^{rd}$) | 1 | 0 | 1 | 1 |
| $\uparrow$ ($4^{th}$) | 0 | 1 | 0 | 0 |
| $\uparrow$ ($5^{th}$) | 1 | 0 | 1 | 1 |
| $\uparrow$ ($6^{th}$) | 0 | 1 | 0 | 0 |
| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |

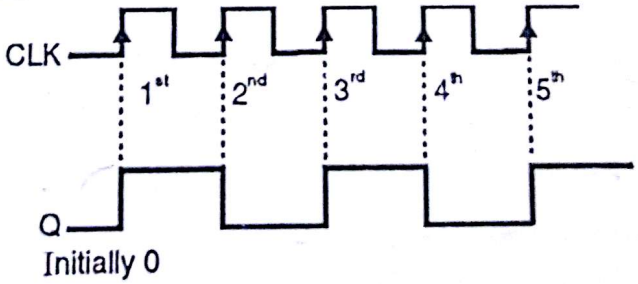



**Fig. Ex. 8.9(a)**

Scanned by CamScanner

## 8.11 Examples from University Papers :

1. **Explain diference between combinational logic circuit and sequential logic circuit. Give two examples of each of these types.** **(May 96, 3 Marks)**

**Ans.:** Refer Section 8.1.

Example :

Combinational circuit typically use AND–OR, EX-OR, NAND–NOR.

Sequential circuit uses flip-flops plus all above mentioned logic gates.

2. **Explain the following terms for a flip-flop :**

**(i) Race around condition.**

**Ans.:** Refer Section 8.6.1.

**(ii) Propagation delay**

**Ans.:** Refer Section 8.5.

**(iii) Edge trigger and level trigger.** **(Dec. 96, 6 Marks, Dec. 97, 5 Marks)**

**Ans.:** Refer Section 8.3.2 and 8.3.4.

3. **Convert T Filp–flop to D flip–flop.** **(May 96, May 97, 8 Marks)**

**Ans. :** To design the conversion logic we need to combine the truth table of require table of required filp–flop and excitation table of given flip–flop.

**Truth table of D flip–flop :**

| D | $Q_n$ | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Table 1**

**Excitation table of T flip–flop :**

| $Q_n$ | → | $Q_{n+1}$ | T |
|---|---|---|---|
| 0 | → | 0 | 0 |
| 0 | → | 1 | 1 |
| 1 | → | 0 | 1 |
| 1 | → | 1 | 0 |

**Table 2**

Here T is given flip flop and D is required flip flop. So Q will be output and D will be new input.

If $Q = 0, D = 0$ then next state $Q_{n+1} = 0$

So transition is from $Q = 0$ to $Q_{n+1} = 0$

$\therefore \quad T = 0$

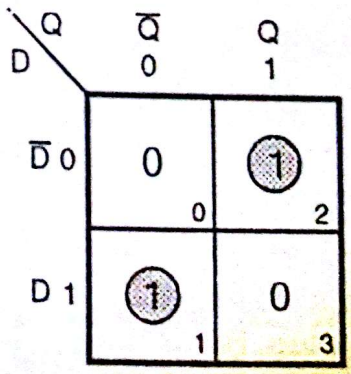
Similarly we can make transition table for other states also.

**Transition table :**

| Inputs | | Transition | Output |
|---|---|---|---|
| Q | D | Q to $Q_{n+1}$ (new state) | T |
| 0 | 0 | 0 → 0 | 0 |
| 0 | 1 | 0 → 1 | 1 |
| 1 | 0 | 1 → 0 | 1 |
| 1 | 1 | 1 → 1 | 0 |

**Table 3**

**K- map**




Scanned by CamScanner

Output equation for K–map will be $T = D\bar{Q} + \bar{D}Q$

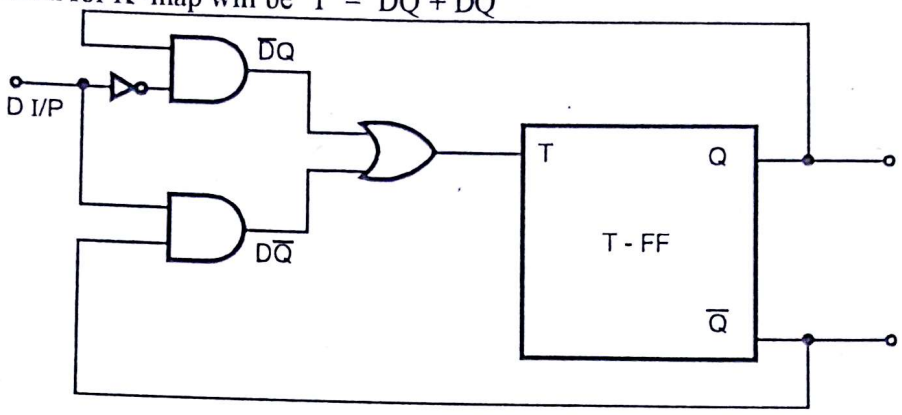



**Fig. Q. 8.3**

**4. Explain what is race around condition in flip–flop and how to avoid malfunction due to such condition ?** **(Dec. 96, 6 Marks)**

**OR**

**Explain what is race around condition in J–K flip–flop. How this problem could be solved ?**

**(May 97, Dec. 97, 6 Marks)**

**Ans.:** Refer Section 8.6.1

**5. Convert D flip–flop to T flip–flop.** **(Dec. 96, 8 Marks, Dec. 98, Dec. 99, 6 Marks)**

**Ans.:** Refer Section 8.9, Ex. 6 (iii).

**6. Explain what is master–slave flip–flop and give its applications.** **(Dec. 96, 4 Marks)**

**Ans.:** Refer Section 8.7. Application : Most widely used in counters.

**7. Filp–flops with propagation delay of 30 nsec. are used in 6 stage asynchronous counter. If the strobe pulse width is 20 nsec, find the maximum frequency of operation of the counter.**

**(May 97, 4 Marks)**

**Ans. :** Following equation provides relationship between clock period, number of flip–flops used and propagation delay of flip–flop.

$$\text{Time period of clock } (T_{CLK}) = N \times (t_{pd} + \text{strobe pulse width})$$

Minimum time delay for each flip–flop = Propagation delay + Strobe pulse width

$$\therefore T_d = 30 \text{ nsec} + 20 \text{ nsec}$$

$$= 50 \text{ nsec}$$

$$N = \text{number of flip–flops used}$$

As it is 6–stage counter

$$N = 6$$

$$\therefore T_{CLK} = 6 \times 50 \text{ nsec} = 300 \text{ nsec.}$$

Scanned by CamScanner

∴ Maximum frequency of operation of counter

$$f_{max} = \frac{1}{T_{CLK}}$$

$$= \frac{1}{300 \text{ nsec}} = \frac{1}{300 \times 10^{-9}} = 3.33 \text{ MHz}$$

∴ $\boxed{f_{max} = 3.33 \text{ MHz}}$ ...Ans.

**8. Convert RS flip–flop to T flip–flop.** **(Dec. 97, June 99, 8 Marks)**

**Ans.:** Refer Section 8.9, Ex. 6 (ii).

**9. What is master slave flop–flop and what advantage it offers in logic system?**

**(May 98, 4 Marks)**

**Ans.:** Refer Section 8.7

**10. The clock and 'X' input applied to different flip–flop is shown in Fig. Q.8.10(a).**

**(May 98, 6 Marks)**

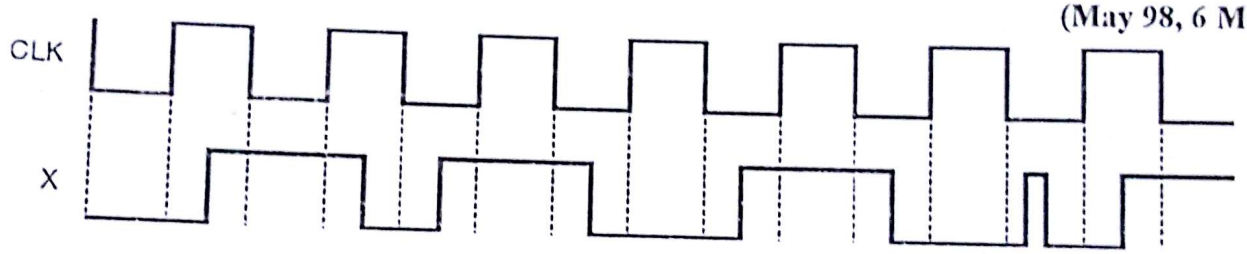


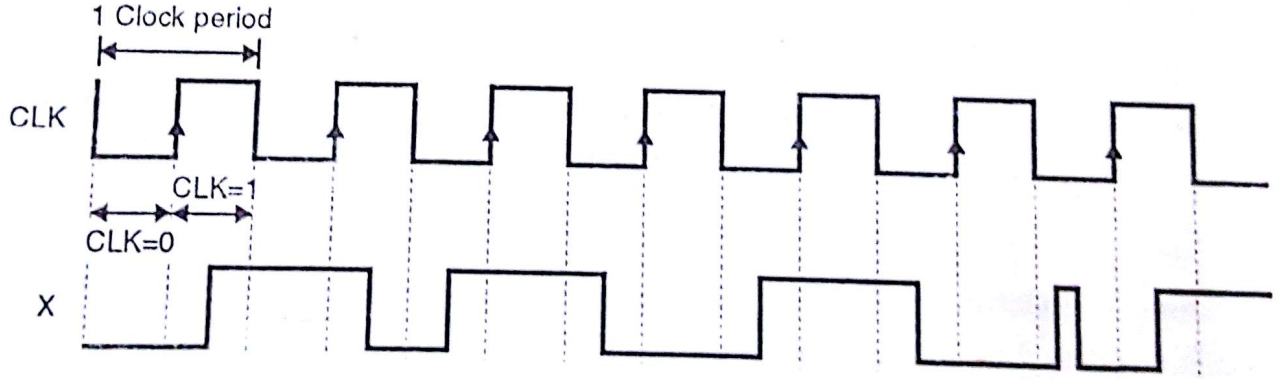

**Fig. Q. 8.10 (a)**

**Draw the output waveforms for the following :**

**(i) J–K master-slave flip–flop. The input 'K' is held LOW and input 'X' applied to 'J'.**

**(ii) 'X' is connected to level triggered 'D' flip–flop.**

**Ans. :**



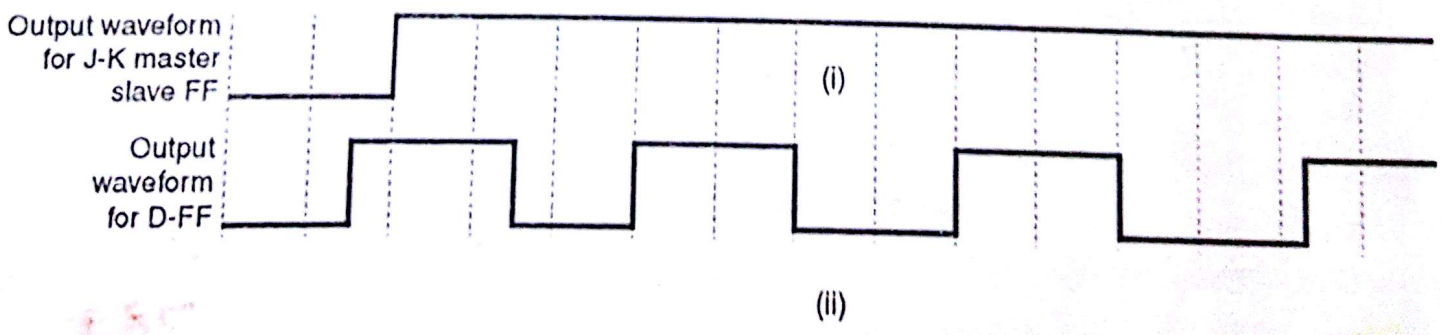




**Fig. Q. 8.10 (b)**

Scanned by CamScanner

**Explanation for output waveforms of J–K master slave :**

Here CLK pulse is given to J–K master–slave FF. The input K is held LOW, so K = 0. Input 'X' is applied to J.

When CLK is 0, X = 0, K = 0

Output will be zero as shown in waveform but as soon as clock pulse start CLK = 1, X = 1, K = 0.

For next instant output = 1 (From truth table of JK m/s FF, for X = 1, K = 0, output Y = 1)

When X becomes zero there will be no change in output. So output continues to be 1. Because for X = 0, K = 0, Y = No change.

Now again X becomes 1, K = 0 output will remain 1.

**Explanation for output waveforms of level triggered D flip–flop :**

From truth table of D–flip flop,

If D = 0, CLK = 1; output = 0

D = 1, CLK = 1; output = 1

So output depends on input applied to D.

For first clock pulse CLK = 1, X = 1, when X becomes 1 at that instant (For high level of clock) output of D flip–flop becomes 1.

It remains 1 till X becomes zero. So output becomes zero at the instant X goes down. It remains zero till X = 0.

But now X becomes 1 when clock is zero. As it is level triggered flip–flop. So its output becomes 1 at high level of clock. So there is some time delay between input X and output of level triggered D flip–flop.

But output depends on input X only.

**11. Whether a master–slave flip–flop acts as an edge triggered F/F? Why or why not?**

**(Dec. 98, 6 Marks)**

**Ans. :** Acutally, master slave FF(M/S FF) is level triggered FF. But as slave follows master, it appears or acts as an edge triggered FF. Let's study the operation to clear the idea.

Refer Section 8.7, page 8–22.

**12. The Fig. Q. 8.12 represents a certain flip–flop function. Identify that flip–flop.**

**(Dec. 98, 3 Marks)**

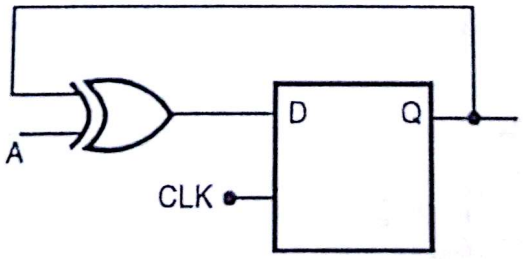



**Fig. Q. 8.12**

COLLEGE OF COMMERCE LIBRARY BSc. IT 743

Scanned by CamScanner

**Ans.:**

| Input | | | | Output | |
|---|---|---|---|---|---|
| A input | CLK | $Q_n$ | $D = A \oplus Q_n$ | $Q_{n+1}$ | |
| 0 | ↑ | 0 | 0 | $0 \rightarrow 0(Q_n)$ | Part I |
| 0 | ↑ | 1 | 1 | $1 \rightarrow 1(Q_n)$ | |
| 1 | ↑ | 0 | 1 | $1\ (\bar{Q}_n)$ | Part II |
| 1 | ↑ | 1 | 0 | $0\ (\bar{Q}_n)$ | |

**Truth Table 4**

First observe part I in the above table. As shown, when A = 0, even though we apply clock pulse output remains as it is i.e. $Q_n$ (previous state) observe part II. As A = 1 for each clock edge, output is complement of previous state, means output toggles at each clock edge.

Therefore truth table is exactly matching with truth table of T flip–flop.

∴ The flip–flop is **T flip-flop**.

**13. Modify an asynchronous RS–FF so that when R = S = 1 the FF is set. (Dec. 98, 4 Marks)**

**OR**

**Design a modified SR FF in which the flip-flop is set when R = S = 1, using a NAND memory cell and additional gates. (Dec. 99, 10 Marks)**

**Ans.:** In SR flip–flop when S = R = 1, flip–flop is in undetermined state (Q = 1, $\bar{Q}$ = 1). Second problem is that, when power supply is switched on, the state of circuit is uncertain. Output is set or reset, that is unpredictable.

Solution is to use preset (also called direct set) and clear (also called as 'direct reset') terminal $P_r \rightarrow$ Preset, $C_r$ = clear.

1. $P_r = C_r = 0$, it give Race condition, because output of NAND 3/4 = 1, so avoid this condition.
2. $P_r = 1$, $C_r = 0$, this is reset condition of flip-flop. As $C_r = 0$, $\bar{Q} = 1$. As input to NAND 3 is $P_r = 1$, $\bar{Q} = 1$ and output of NAND 1 is Q = 0.
3. $P_r = 0$, $C_r = 1$. This is set condition of flip-flop. As $P_r = 0$ and S = R = 1; Q = 1. As input to NAND 4 is $C_r = 1$, Q = 1 and output of NAND 2 = 1(R = 1, CLK = 1) forces $\bar{Q} = 0$.

Therefore using preset and clear terminal we can set flip–flop when R = S = 1.

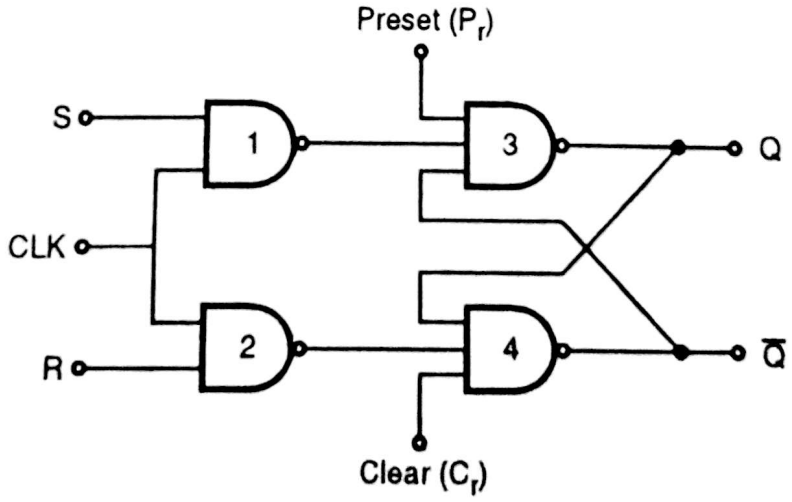



**Fig. Q. 8.13 (a)**

Scanned by CamScanner

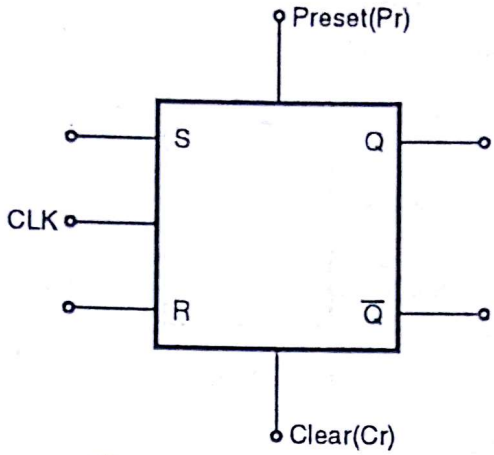



**Fig. Q. 8.13 (b) : Block diagram of modified RS flip-flop**

One more remedy over above problem is, simply use an inverter keeping other funtions of SR FF as it is.

| Inputs | | Outputs | |
|---|---|---|---|
| S | R | Q | $\overline{Q}$ |
| 0 | 0 | NC | NC |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | RACE | RACE |

SR is set when, S = 1 and R = 0. So one can simply put inverter at input to R.

**Table 4A**

| Inputs | | Outputs | |
|---|---|---|---|
| S | R' | Q | $\overline{Q}$ |
| 0 | 0 | 0 | 1 |
| 0 | 1 | NC | NC |
| 1 | 0 | RACE | RACE |
| 1 | 1 | 1 | 0 |

→ expected

**Table 4B**

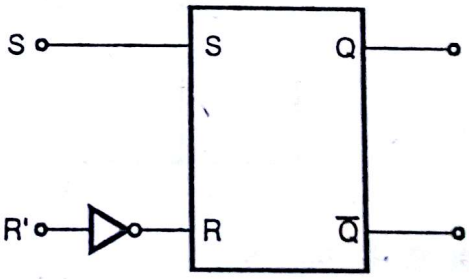



**Fig. Q.8.13(c)**

**14. What is meant by following terms in a FF :** **(June 99, 4 Marks)**

**(i) Edge triggered.**

**Ans.:** Refer Section 8.3.4

**(ii) Set up time and hold time.** **(Dec. 99, 2 Marks)**

**Ans.:** Refer Section 8.5.

**15. Consider MN–FF in the following Fig. Q. 8.15 which is a modified JK–FF.**

Scanned by CamScanner

**(i) Obtain its characteristic table.** **(Dec. 98, 5 Marks ; Dec. 99, 3 Marks)**

**(ii) Write excitation table of MN-FF.** **(Dec. 99, 3 Marks)**

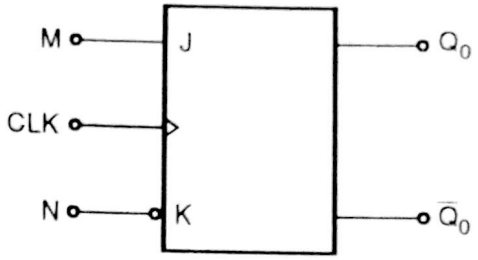



**Fig. Q. 8.15**

**Ans. :** First write the truth table for J–K FF

| | Input | | Output |
|---|---|---|---|
| **CLK** | **J** | **K** | $Q_{n+1}$ |
| X | 0 | 0 | $Q_{n(NC)}$ |
| ⎍ | 0 | 1 | 0 |
| ⎍ | 1 | 0 | 1 |
| ⎍ | 1 | 1 | $\overline{Q}_n$ |

**Table 5**

In Fig. Q. 8.15 M is directly connected to input J of J–K FF. Therefore inputs applied at M are directly fed to J input but N is connected to K through a bubble (inverter). So whatever we are applying at N is inverted and fed to K. So characteristic table will change. Output will be different.

**Characteristic table for MN–flip–flop :**

| Input | | | Output |
|---|---|---|---|
| **CLK** | **M(J)** | **N(K)** | **Y** |
| X | 0 | 0 → (1) | 0 |
| ⎍ | 0 | 1 → (0) | $Q_n$ |
| ⎍ | 1 | 0 → (1) | $\overline{Q}_n$ |
| ⎍ | 1 | 1 → (0) | 1 |

**Table 6**

**Excitation table of JK flip-flop :**

| $Q_n$ | $Q_{n+1}$ | J | K |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

**Table 7**

**From Table 7 write truth table for MN flip-flop.**

Scanned by CamScanner

| Input | | | | Output | |
|---|---|---|---|---|---|
| M | N | J | K | $Q_n$ | $Q_{n+1}$ |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 |

Table 8

∴ **Excitation table for MN flip-flop :**

| $Q_n$ | $Q_{n+1}$ | M | N |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 0 |
| 1 | 1 | X | 1 |

Table 9

**16. How does the operation of an asynchronous input differ from that of a synchronous input in a flip–flop ?** **(Dec. 98, June 99, 4 Marks)**

**Ans.:** Asynchronous input immediately affects the output, when it is applied. But synchronous input will affect output after CLK edge hits. Thus, there is time delay.

**17. The Fig. Q.8.17 represents certain F/F function. Identify that FF.** **(Dec. 99, 3 Marks)**

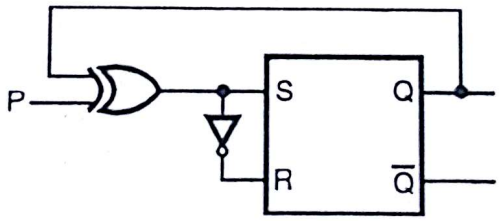



**Fig. Q. 8.17**

**Ans. :** Refer Section 8.9, Ex. 8.



Scanned by CamScanner

# Counters

9

**Syllabus :**

Counters – Synchronous / Asynchronous. Counter design

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Introduction | 9.1 | ✓ | – |
| • Asynchronous counter | 9.2 | ✓ | ✓ |
| • Terms related to counter and problems in ripple counter | 9.3 | ✓ | ✓ |
| • IC 7493 | 9.4 | ✓ | ✓ |
| • Synchronous counter design using TFF, DFF, JKFF with Bushing concept | 9.5, 9.6, 9.7, 9.8 | ✓ | ✓ |
| • Controlled synchronous counter | 9.9 | – | ✓ |
| • Presettable counter | 9.10 | ✓ | ✓ |
| • IC 7490, IC 7492 | 9.11, 9.12 | – | ✓ |
| • IC 74190, IC 74191 | 9.13 | – | ✓ |
| • Analysis of counter circuit | 9.14 | – | ✓ |



**Comment**

- Very very very important chapter.
- Concentrate on full chapter very corectly.

Scanned by CamScanner

## 9.1 Introduction :

➥ [ASKED IN EXAM - MAY 96, DEC. 96, DEC. 97 !!! ]

The most wide application of FF is counters and registers. The FF is basic element of sequential logic system. Using FF and combinationl circuit, any sequential system can be designed.

Counter is most useful and versatile subsystem of digital branch. As name suggests counter is going to count number of CLK pulses applied to it. Basically CLK is timing parameter we introduced in FF, therefore CLK pulses occurs at regular time interval, so that counter can be used to measure time or frequency.

Counters can be sub-divided into two groups :

Counter

| Asynchronous counter (Also called as serial) | Synchronous counter (Also called as parallel) |
|---|---|
| (1) Very simple and straight forward in operation. | (1) Complex in comparison with asynchronous. |
| (2) Has speed limitation. | (2) Speed is more. |
| (3) Next FF is triggered by previous FF. | (3) Each FF is triggered by CLK in synchronism. |
| (4) Problem of glitch. | (4) No problem of glitch. |
| (5) Settling time is more. | (5) Settling time is less. |

Synchronous / asynchronous counter can be sub-divided in following sub-group :

(1) **Regular Counter :** For building counter, FFs are used. There is direct relation between number of FFs used and number of states of counter.

Let's say No. of state are N, number of FFs are m then relation is given by,

$$N = 2^m \quad \ldots[1]$$

We will consider one more variable n, where n is number of actual states of counter.

In regular counter number of actual states of counter (n) and number of states given by eqn. 1 (N) are equal.

$$\boxed{n = N}$$

Let's say FF are 3, $\therefore N = 2^m = 2^3 = 8$. $\therefore n = 8$.

(2) **Truncated Counter :** In this type of counter number of actual states of counter (n) are always less than number of states given by Equation (1) (N).

$$\boxed{n < N}$$

If FFs are 3. $N = 2^m = 8$ but $n < 8$.

(3) **Sequential Counter :** As name suggests, states of counter are sequential i.e. 0, 1, 2, 3, 4, 5, ... so on.

(4) **Non-Sequential Counter :** Sequence or states of counter are not sequential but irregular. e.g. 0, 3, 9, 8, 2, 1, 7.

## 9.2 Asynchronous Counter :

**Let's start discussing asynchronous counter. Fig. 9.1 shows generalized block schematic of asynchronous counter. Asynchronous counter is also called as *Ripple Counter*.**

Scanned by CamScanner

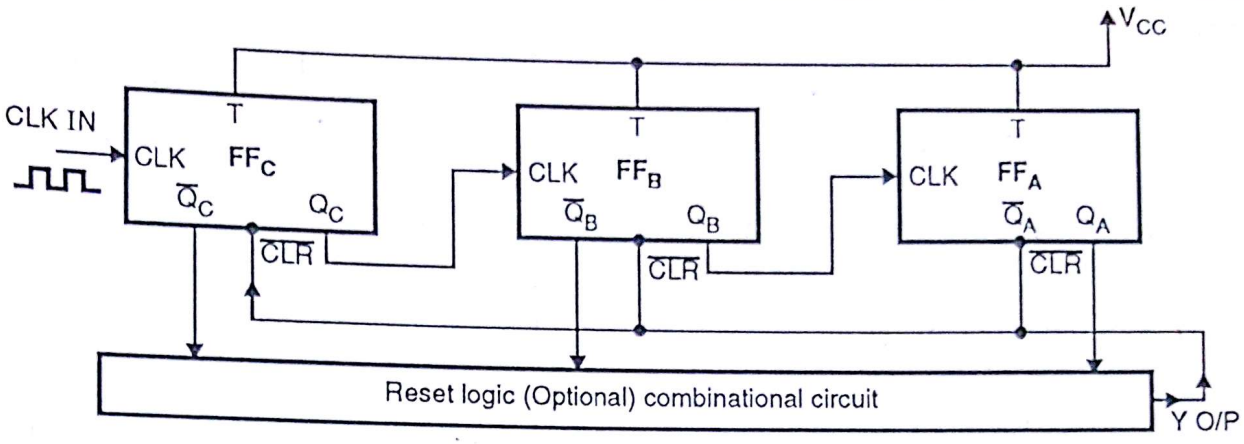



**Fig. 9.1**

Observations from Fig. 9.1.

(1) CLK IN is given to only one FF.

(2) Normally T type FF is preferred in ripple counter design.

(3) If you want 'n' states less than 'N' i.e. you want to stop counter in between and restart from '0' (reset) state, ***reset logic*** should be used.

Reset logic is nothing but combinational circuit. This block is optional.

(4) $Q/\overline{Q}$ of previous FF.given to next FF, CLK input. Means one FF drives another, therefore this counter is called as ***ripple counter***.

To elaborate the point, remind old Hindi movie, where hero is waiting for heroin at the bank of the river. For time pass, he throws stones in the river. If you observe technical part (forget love part), throwing stone in water generates circles. One circle drives $2^{nd}$, $2^{nd}$ drives $3^{rd}$ and so on. It is also called as ripple generated in water. The same concept, driving next by previous, is used in asynchronous counter $\therefore$ called a ripple counter.

## 9.2.1 3 Bit Ripple Counter :

To understand ripple counter let's start with 3 bit ripple counter.

Refer truth table of T FF, and Fig. 9.2.

**Observation 1 :** (1) FF's are ***negative edge triggered.***

(2) O/P taken from Q (not $\overline{Q}$)

(3) Cascaded O/P s are Q (not $\overline{Q}$)

**Steps :**

(1) No. of states $N = 2^m = 2^3 = 8$.

(2) $\therefore$ Take three FF., Tie T input of all FF to $V_{cc}$.

(3) Apply CLK IN to one of the FF (Normally left most is selected and it will become LSB). Last FF becomes MSB. Refer Fig. 9.2. C = MSB A = LSB.

(4) O/p Q of one FF given to next FF. It is obvious that you will start from LSB and end with MSB.

Scanned by CamScanner

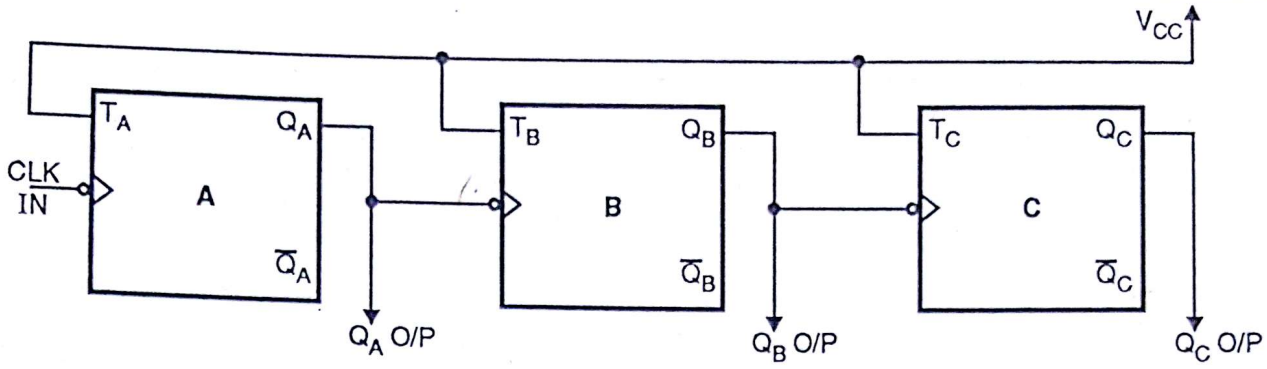



**Fig. 9.2 : 3 Bit ripple counter**

First we will draw waveform and write truth table, then we will see working.

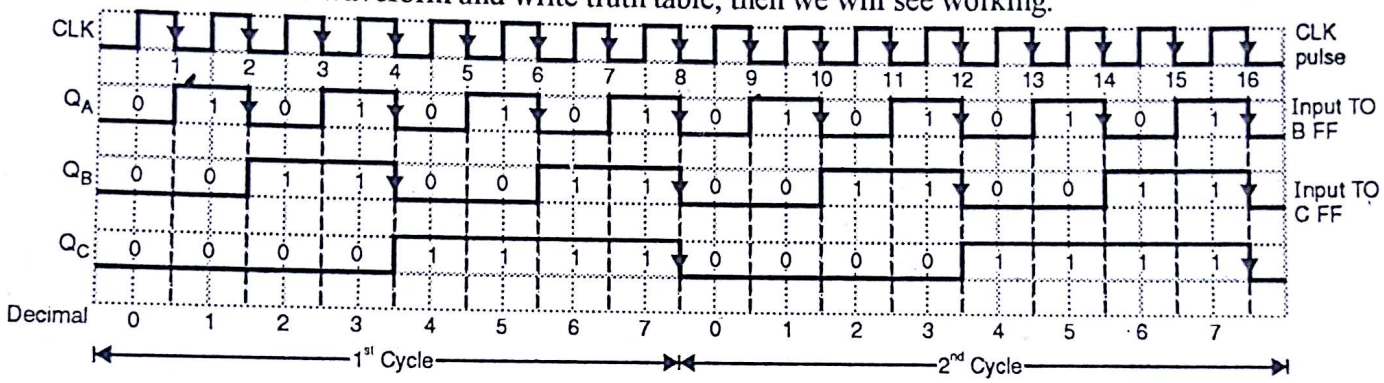



**Fig. 9.3(a)**

**Truth table :**

| States of Counter | | | |
|---|---|---|---|
| $Q_C$ | $Q_B$ | $Q_A$ | Decimal Equivalent |
| 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 |
| 0 | 0 | 0 | 0 |

**Fig. 9.3(b)**

**Working :**

Please refer Fig. 9.3 (a) and (b) [$Q_C\ Q_B\ Q_A$ = CBA]

(1) Initially all FF's are reset (assumed to be at) CBA = 000

(2) When 1st CLK edge hits, A will toggle from 0 → 1 (rising edge),
∴ B and C are unchanged. ∴ CBA = 001.

(3) When 2nd CLK edge hits, A will toggle from 1 → 0 (falling edge, shown in waveform),

Scanned by CamScanner

$\therefore$ B toggle from $0 \rightarrow 1$ (rising edge),
$\therefore$ C is unchanged. Finally CBA = 010.

(4) When 3rd CLK edge hits. A will toggle from $0 \rightarrow 1$ (rising edge)
$\therefore$ B and C are unchanged.
$\therefore$ CBA = 011.

(5) At 4th CLK edge, A will toggle from $1 \rightarrow 0$ (falling edge),
$\therefore$ B toggles from $1 \rightarrow 0$
(falling edge, shown in the waveform),
$\therefore$ C changes from $0 \rightarrow 1$. $\therefore$ CBA = 100.

(6) At 5th CLK edge, A will toggle from $0 \rightarrow 1$ (rising edge)
$\therefore$ B and C are unchanged.
$\therefore$ CBA = 101

(7) At 6th CLK edge, A will toggle from $1 \rightarrow 0$ (falling edge), this will force output B to toggle. B will toggle from $0 \rightarrow 1$ (rising edge). As B provides rising edge, C is unchanged.
$\therefore$ CBA = 110.

(8) At 7th CLK edge, A will toggle from $0 \rightarrow 1$ (rising edge),
$\therefore$ B and C both are unchanged.
$\therefore$ CBA = 111.

(9) Finally at 8th CLK edge, A will toggle from $1 \rightarrow 0$ (falling edge), this will force output B to toggle. B will toggle from $1 \rightarrow 0$ (falling edge). As B provides falling edge to C, C also toggles. C changes state from $1 \rightarrow 0$. Thus 3 FF changes state from $1 \rightarrow 0$.
$\therefore$ CBA = 000. We have started our discussion with CBA = 000.
So cycle will repeat from point No. 1.

**Note :** (1) Falling edge of A FF changes state of B FF **ONLY.**
(2) Falling edge of B FF changes state of C FF **ONLY.**
(3) REMEMBER THAT ONE FF DRIVES ANOTHER.

Counter states are represented in Fig. 9.3(b).

**Conclusion :**

(1) No. of actual states of counters are n = 8 as $N = 2^3 = 8$, $\therefore$ regular counter. $\therefore$ it is also called as Modulo 8 counter. (Reference Section 9.3.2)

(2) Counter is counting up. 0, 1, 2, 3, ...

**Note :** Counter will count up or down depends upon transition (+ or –) which drives the FF ; Q or $\bar{Q}$ used for cascading and finally, the output we take. Let's represent it symbolically.

where, $U_P$ = 1 –up counter
= 0 –down counter
$p_n$ = 1 –negative transition (FF used is negative edge triggered)
= 0 –positive transition (FF used is positive edge triggered)
$Q_t$ = 1 –Q cascaded
= 0 $-\bar{Q}$ cascaded
$Q_1$ = 1 –output from Q
= 0 –output from $\bar{Q}$

We can write a simple boolean equation for this, i.e.,

$$U_P = p_n \oplus Q_t \oplus Q_1$$

We will have different cases, as listed follows :

Scanned by CamScanner

**Case :**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| I | $\Rightarrow$ | $p_n = 1$ | $Q_t = 1$ | $Q_1 = 1$ | $\therefore$ | $U_p$ | $= 1 \oplus 1 \oplus 1 = 1$ | | | $\therefore$ counter is up |
| II | $\Rightarrow$ | $p_n = 1$ | $Q_t = 1$ | $Q_1 = 0$ | $\therefore$ | $U_p$ | $= 1 \oplus 1 \oplus 0 = 0$ | | | $\therefore$ counter is down. |
| III | $\Rightarrow$ | $p_n = 0$ | $Q_t = 1$ | $Q_1 = 1$ | $\therefore$ | $U_p$ | $= 0 \oplus 1 \oplus 1 = 0$ | | | $\therefore$ counter is down. |
| IV | $\Rightarrow$ | $p_n = 0$ | $Q_t = 1$ | $Q_1 = 0$ | $\therefore$ | $U_p$ | $= 0 \oplus 1 \oplus 0 = 1$ | | | $\therefore$ counter is up. |
| V | $\Rightarrow$ | $p_n = 1$ | $Q_t = 0$ | $Q_1 = 0$ | $\therefore$ | $U_p$ | $= 1 \oplus 0 \oplus 0 = 1$ | | | $\therefore$ counter is up. |
| VI | $\Rightarrow$ | $p_n = 1$ | $Q_t = 0$ | $Q_1 = 1$ | $\therefore$ | $U_p$ | $= 1 \oplus 0 \oplus 1 = 0$ | | | $\therefore$ counter is down. |

## 9.3 Terms Related to Counter :

Before we proceed further, we would like to study few terms related to counter.

### 9.3.1 State Diagram :

State diagram means graphically representing the states of counter circuit.

e.g. for 3 bit binary ripple up counter state diagram can be drawn as,

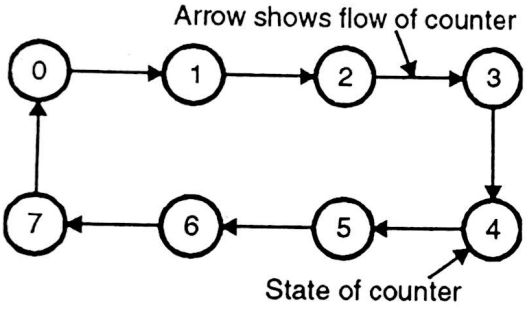



**Fig. 9.4**

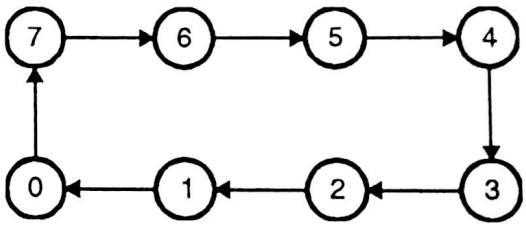



**Fig. 9.5**

For 3 bit binary ripple down counter, state diagram is as shown in Fig. 9.5.

**Note :** As shown, in counter, we don't have any input(s) or output(s). We are simply concerned with the ***state of the counter.*** Therefore, on the top of the transition arrow, showing change of state, we haven't shown any input / output combination(s).

### 9.3.2 Module N Counter :

If it is desired to have modulo N (mod – N) counter, the number of F.F. required is determined by,

$$N \leq 2^m \qquad m = \text{Number of F.F.}$$

Let say $m = 4$ $\therefore N = 16$. But if we require only 10 states out of 16, then it is called as moduls – 10 (Mod – 10) counter, but required FFs will be 4 only.

Mod – N can be achieved by reseting the FF. This should be done at the N state. In Mod – 10, counter will count from 0 to 9 , and at $10^{th}$ state it will reset back to 0. Mod – 10 counte is also called as ***decade counter.***

Uptill now we have seen that counter is sequential and number of states provided are $N = 2^m$ . Now let's take case of truncated ripple counter when n (number of actual state) < N.

**Ex. 1 :** Design mod – 3 ripple (asynchronous) counter.

Scanned by CamScanner

**Soln. :** (1) Draw state diagram. No. of states are 3.

(2) No of actual states are n = 3

$\therefore n < N = 2^m \quad 3 \leq 2^m \therefore \boxed{m = 2}$

$\therefore$ 2 FFs are required.

(3) Here FF used should have ***clear*** terminal for resetting or clearing.

(4) Assumption : Considering FF and gates are having '0' propogation delay.

(5) Reset logic circuit required to terminate count after 2.

(6) 2 FFs are used $\therefore$ N = 4.

| $Q_B$ | $Q_A$ | |
|---|---|---|
| 0 | 0 | → 0 |
| 0 | 1 | → 1 |
| 1 | 0 | → 2 |
| 1 | 1 | → 3 |

} Valid states

Invalid state so whenever this state occurs counter should be ***CLEARED*** (Reset).

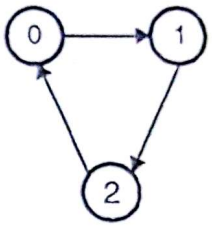



**Fig. Ex. 9.1(a)**

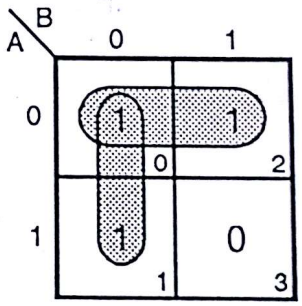
Normally CLR terminal is ACTIVE LOW $\therefore$ Design a reset logic combinational circuit such that output Y (Refer Fig. 9.1, block schematic) should be 1 when valid states are there and Y = 0 when Invalid states are present.

**Truth Table,**

| $Q_B$ | $Q_A$ | Y O/P |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

→ Reset FF

**K-Map :**




$Y = \overline{A} + \overline{B}$ ...(a)

$= \overline{AB}$ ...(b)

**Circuit :**

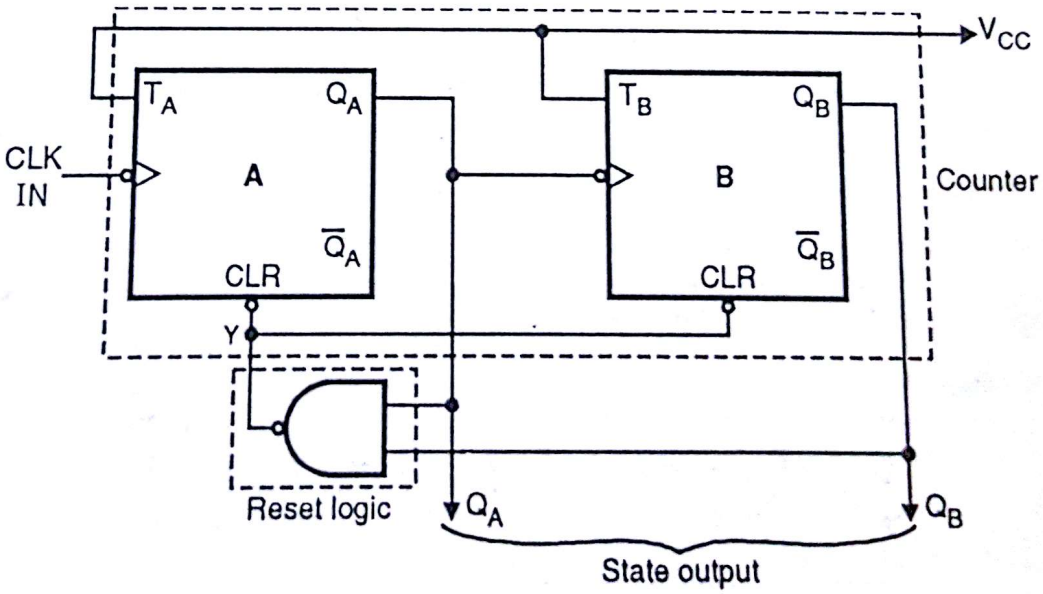



**Fig. Ex. 9.1(b)**

Scanned by CamScanner

Counter is up counter. Therefore FF should be negative edge triggered, cascade Q O/P and final O/P is taken from Q. The final circuit is therefore shown in Fig. Ex. 9.1(b). Observe waveform.

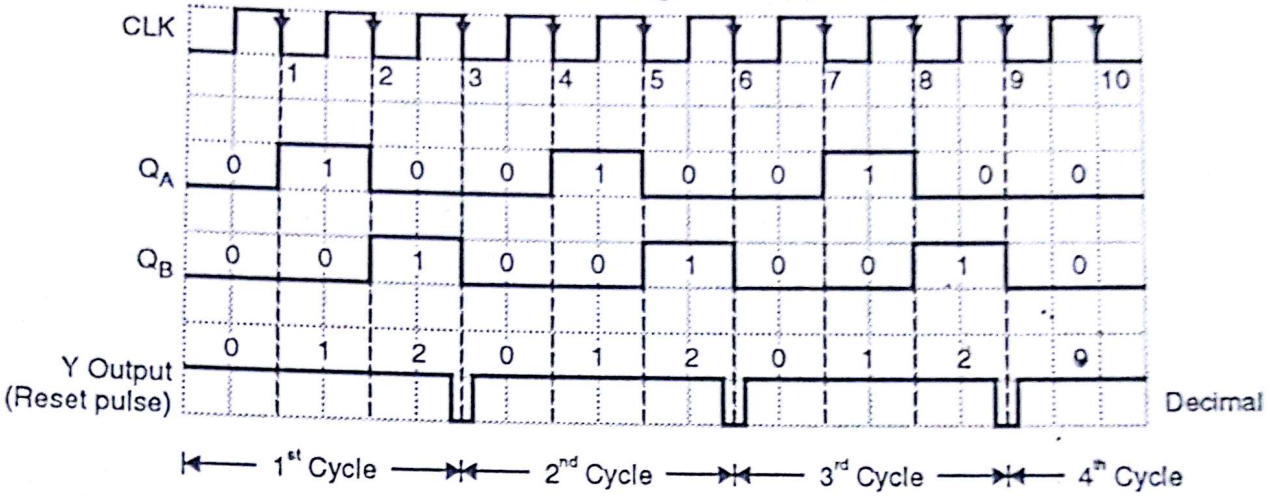



**Fig. Ex. 9.1(c)**

**Working :** Refer Fig. Ex. 9.1(b) and Fig. Ex. 9.1(c) ($Q_B\ Q_A$ = BA)

(1) Initally all FFs are cleared.

$\therefore$ BA = 00 ; $\therefore$ Y = 1

(2) When 1st negative CLK edge hits, A toggles

$\therefore$ A = 1, $\therefore$ BA = 01 ; $\therefore$ Y = 1

(3) At 2nd hit of CLK edge, A toggles from 1 $\rightarrow$ 0 (negative edge),

$\therefore$ B also toggles from 0 $\rightarrow$ 1.

$\therefore$ BA = 10 $\therefore$ Y = 1.

(4) When 3rd CLK edge hits, A will toggle from 0 $\rightarrow$ 1 (positive edge)

$\therefore$ B = unchanged $\therefore$ BA = 11

$\therefore$ Y = 0. As Y = 0 , all FFs will be cleared.

and BA = 00. As BA = 00 Y = 1 again.

The sequence from BA = 11 to BA = 00 , is so fast (as propogation delay is considered 0 $\eta$ sec.) that, it is not possible to observe 11 condition on waveform. Y waveform pulse is also very sharp and of very very small duration.

Thus counter will run 0 $\rightarrow$ 1 $\rightarrow$ 2 $\rightarrow$ 0 .

---

**Ex. 2 :** Design modulo 6 ripple counter.

**Soln. :** (1) State diagram = 6 states $\Rightarrow$ 0 to 5

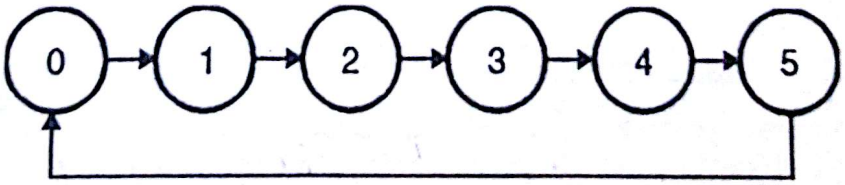



**Fig. Ex. 9.2(a)**

(2) No. of states N = 6 $\therefore$ n < N = $2^m$ $6 < 2^m$ $\therefore$ $\boxed{m = 3}$

Scanned by CamScanner

(3) Design of reset circuit.

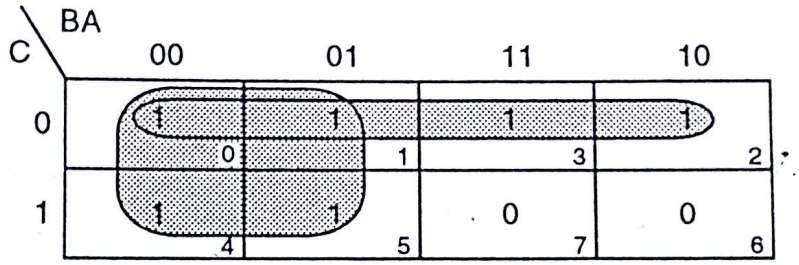
Truth table is,

$N = 2^m = 2^3 = 8$

but $n = 6$

| | C | B | A | Y O/P |
|---|---|---|---|---|
| Valid | 0 | 0 | 0 | 1 |
| | 0 | 0 | 1 | 1 |
| | 0 | 1 | 0 | 1 |
| | 0 | 1 | 1 | 1 |
| | 1 | 0 | 0 | 1 |
| | 1 | 0 | 1 | 1 |
| Invalid | 1 | 1 | 0 | 0 |
| | 1 | 1 | 1 | 0 |

**K-map :**




$$Y = \overline{C} + \overline{B}$$

**Points :**

(1) Counter is up counter, ∴ we have to cascade Q output, – ve edge triggered FF and final output from Q.

(2) FF should have ACTIVE LOW reset (clear) terminal.

(3) Propogation delay presently assumed 0 n sec

**Circuit :**

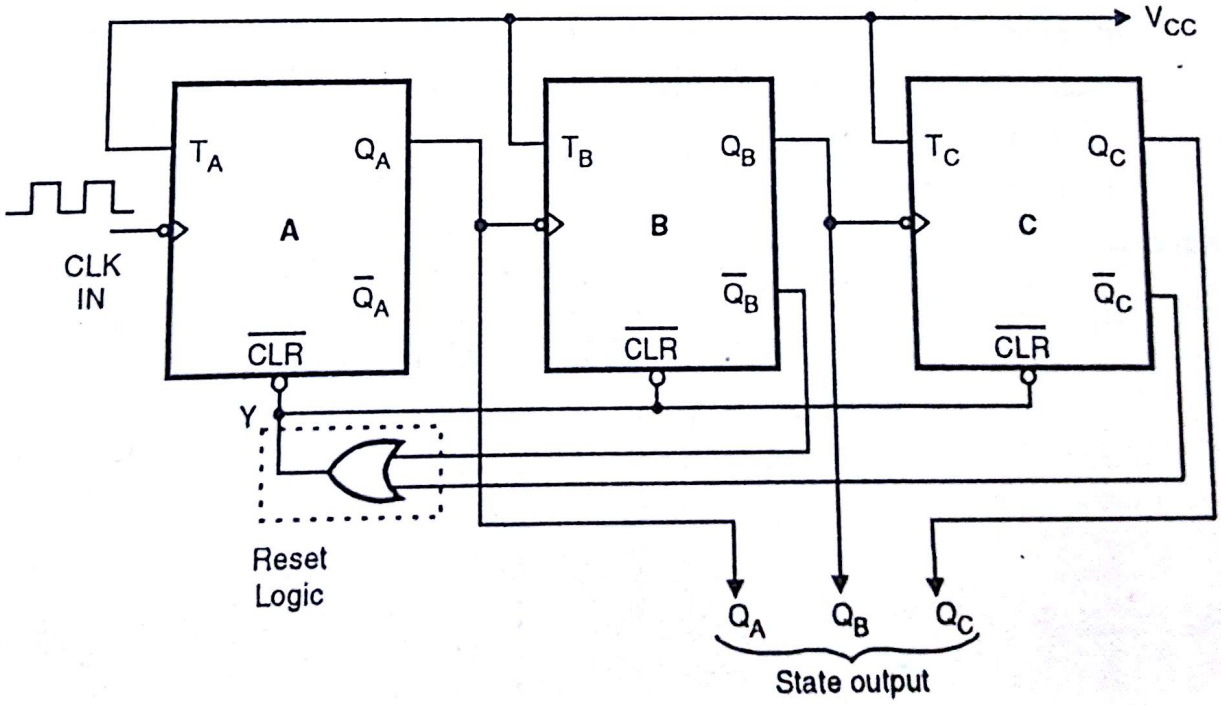



**Fig. Ex. 9.2(b)**

Scanned by CamScanner

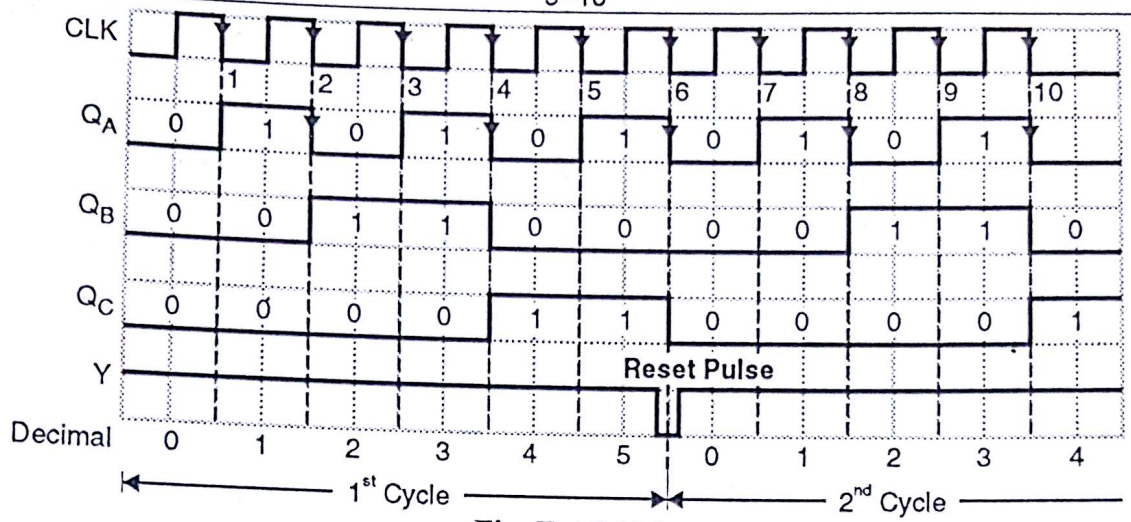



Fig. Ex. 9.2(c)

**Ex. 3 :** Design ripple counter for state diagram shown in Fig. Ex. 9.3. **(June 99, 10 Marks)**

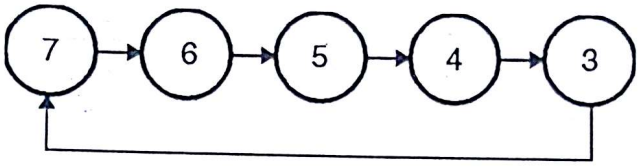



Fig. Ex. 9.3

**Soln. :** Observe state diagram.

**Conclusion :**

(1) Counter is down counter

(2) Valid states are 7, 6, 5, 4, 3.

(3) Invalid states are 2, 1, 0.

(4) After 3, counter jumps back to 7 state. ∴ Counter should be resettable type i.e. FF should have RESET terminal. Assume that RESET is ACTIVE LOW.

(1) No. of FFs required

Max. state is 7 ∴ $7 \leq 2^m$ ∴ $\boxed{m = 3}$

As m = 3, $N = 2^m = 2^3 = 8$ (0 to 7)

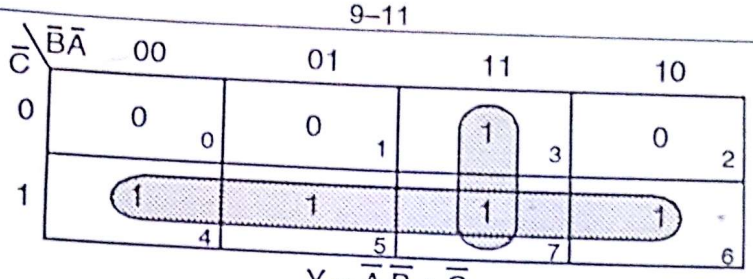
(2) Design of reset circuit.

Truth table

| | $\bar{C}$ | $\bar{B}$ | $\bar{A}$ | Y output |
|---|---|---|---|---|
| Invalid states | 0 | 0 | 0 | 0 |
| | 0 | 0 | 1 | 0 |
| | 0 | 1 | 0 | 0 |
| Valid states | 0 | 1 | 1 | 1 |
| | 1 | 1 | 0 | 1 |
| | 1 | 0 | 1 | 1 |
| | 1 | 0 | 0 | 1 |
| | 1 | 1 | 1 | 1 |

Scanned by CamScanner

**K-map :**




$Y = \bar{A}\,\bar{B} + \bar{C}$

While drawing circuit diagram : (1) Use FF with negative edge triggering.

(2) Cascade Q output ; (3) Final output from $\bar{Q}$

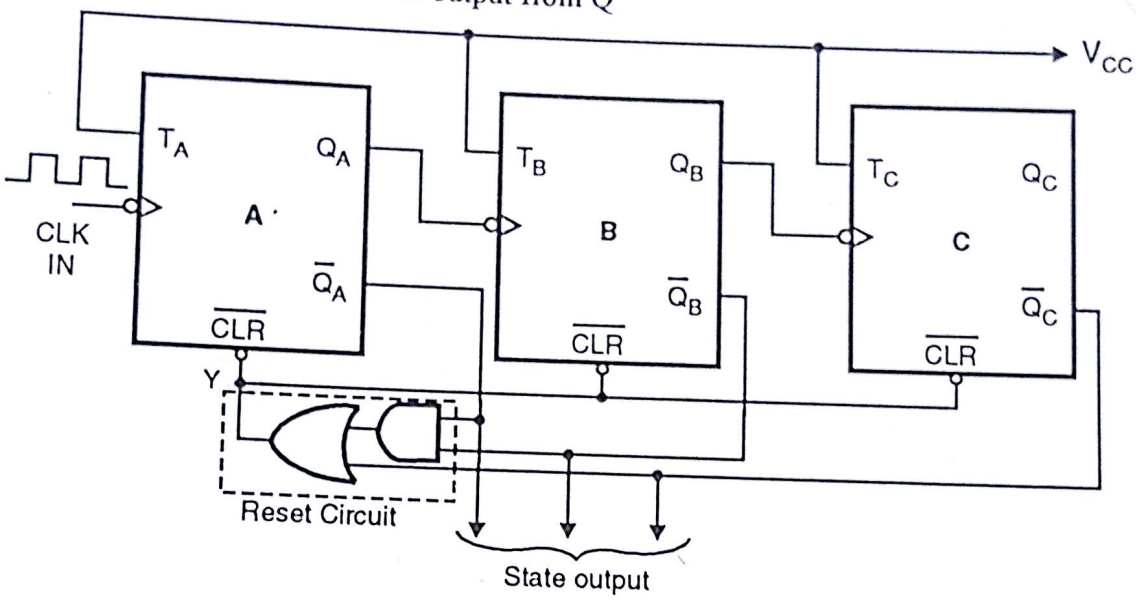



**Fig. Ex. 9.3(a) : Circuit diagram**

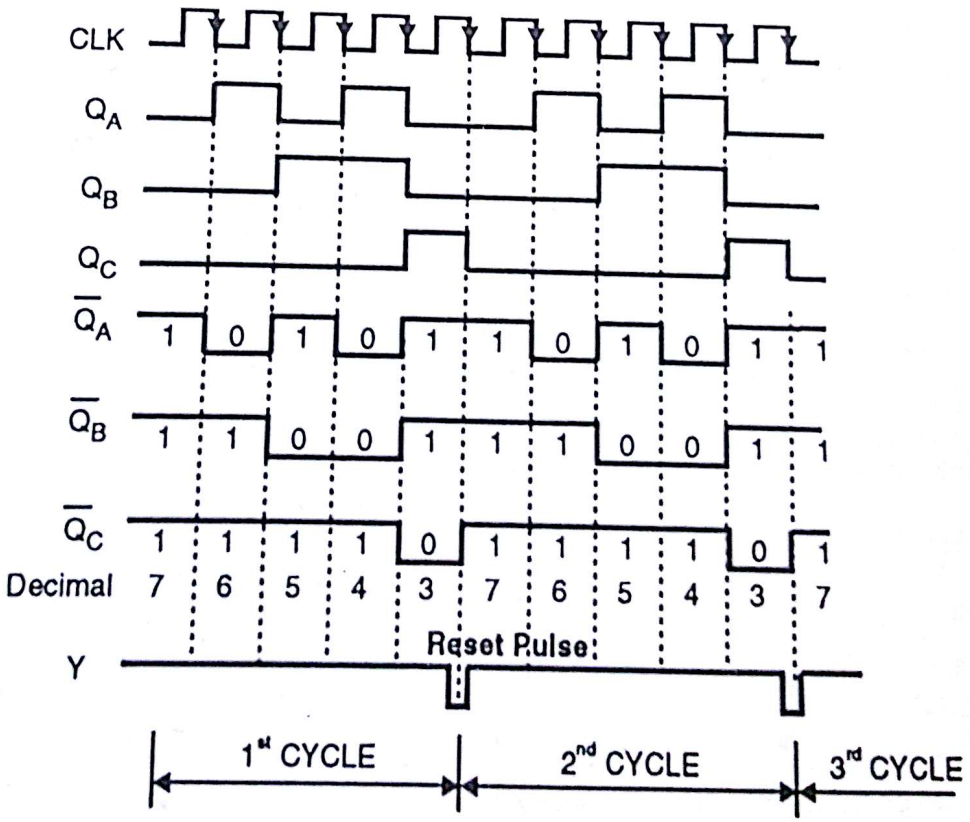



**Fig. Ex. 9.3(b)**

Scanned by CamScanner

### 9.3.3 Problems Involved In Ripple Counter :

➥ [ASKED IN EXAM - MAY 96, DEC. 97, MAY 97, MAY 98, JUNE 99, DEC. 99!!! ]

Mainly two problems are attached with ripple counter.

(1) **Glitch** ⟶ Propagation delay of *Reset logic*.

(2) **Propagation delay** ⟶ propagation delay of FF.

**GLITCH :**

In example 1 assumption was made that propagation delay through FF and reset circuit is 0 η sec. But practically this situation doesnot exist, therefore ***glitch*** occurs.

Let's consider propagation delay of reset logic.

For analysis let's consider example 1 states are 0, 1, 2. Redraw the waveforms considering propagation delay of reset CLK.

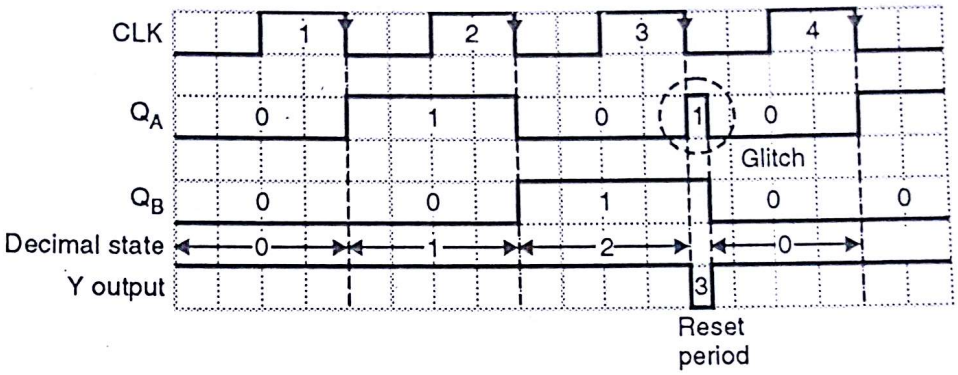



**Fig. 9.6**

**Working :**

Refer Fig. 9.6 and Ex. 1 ($Q_A$ = A $Q_B$ = B)

(1) Initially all FFs are reset. ∴ BA = 00 ∴ Y = 1

(2) When 1st CLK edge hits, A will toggle ∴ BA = 01, ∴ Y =1

(3) At 2nd CLK edge, A will toggle from 1 → 0 (negative edge), ∴ B also toggles. ∴ BA = 10. ∴ Y = 1

(4) When 3rd CLK edge hits, A will toggle from 0 → 1 (positive edge), ∴ B is unchanged. ∴ BA = 11

Reset circuit is designed in such a way that, when 11 occurs Y = 0. But appearing of logic 11 at NAND input, getting settled, then propagate through NAND and appear at output, then resetting the FF will take sometime in η sec. During this period BA = 11. This condition is invalid condition and as in Fig. 9.6, $Q_A$ produces unwanted short duration pulse called ***glitch***.

**Propagation Delay in Ripple Counter**

Basic principle of operation of asynchronous counter is, each FF is triggered by the transition at the output of preceding FF. Each FF has internal propagation delay, means second FF will not respond unless and untill a propagation delay time, after the first FF receives an active clock transition. ∴ Third FF will not respond untill a time equal to twice propagation delay ($2 \times t_{pd}$) after that clock transition.

Thus propagation delay of FF accumulates so that $N^{th}$ FF cannot change state unless and untill, time equals to $N \times t_{pd}$ after the CLK transition occurs.

To understand this refer waveforms in Fig. 9.7.

∴ $Q_C$ misses one CLK period ∴ we get random states.

Scanned by CamScanner

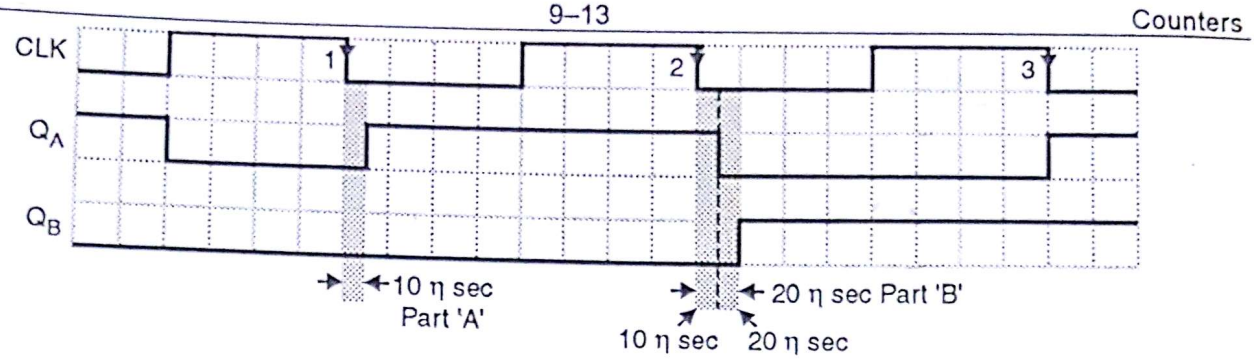



**Fig. 9.7**

Refer Fig. 9.7 ($Q_A$ = A, $Q_B$ = B) Propagation delay of one FF = 10 η sec.

(1) Initially both output BA = 00

(2) When first CLK edge hits, A changes from 0 → 1. As shown $Q_A$ changes states 10 η sec after CLK edge hits. (Part (a))

(3) In second CLK edge $Q_A$ changes from 1 → 0. Transition occurs 10 η sec after CLK edge hits.
$Q_A$ provides negative transition (1 → 0) to B FF
∴ $Q_B$ changes the state from 0 → 1, 10 η sec after the transition of $Q_A$.
∴ with respect to CLK edge $Q_B$ changes state after 20 η sec. (part (b)).
i.e. 2 FF × 10 η sec = 20 η sec.

This propagation delay causes limitation on CLK frequency input, let's see in next example.

Say CLK pulse duration is 100 η sec. Propagation delay of each FF is 50 η sec.

The waveform in Fig. 9.8 are for 3 bit ripple counter. Actually states should be 0, 1, 2, 3, 4, 5, 6, 7. But due to propagation delay of FF and high CLK frequency states are random.

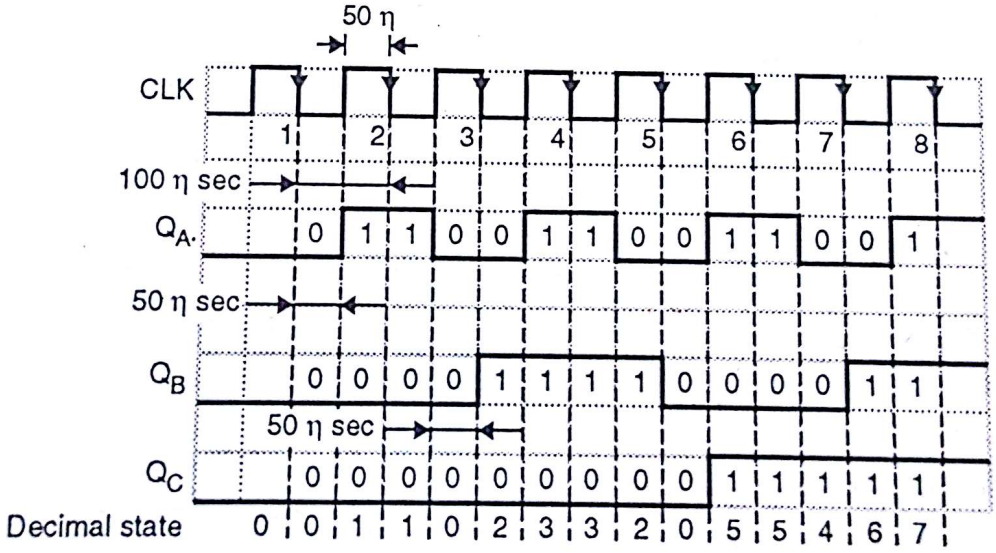



**Fig. 9.8**

***Observe the waveform.***

(1) $Q_A$ changes the state (0 → 1 or 1 → 0) 50 η sec after the CLK edge hits. Means $Q_A$ changes in the mid of the CLK.

(2) ∴ $Q_B$ changes the state, 50 η sec, *after* $Q_A$ changes,
∴ with respect to CLK edge $Q_B$ changes after 100 η sec.
But as CLK is of 100 η sec duration, we expect that after each 100 η sec, output should change but that is not achieved.

(3) ∴ $Q_C$ changes the state, 50 η sec *after* $Q_B$ changes.
∴ with respect to CLK edge, $Q_C$ changes after ***150 η sec.***
∴ $Q_C$ misses one CLK period ∴ we get random states.

Scanned by CamScanner

For ripple counter, following equation provides relationship between CLK period, number of FFs used and propagation delay of FF.

Time period of CLK $(T_{CLK}) \geq N \times t_{pd}$

where, $t_{pd}$ = Propagation delay

$N$ = No. of FFs

$\therefore$ $f_{max} = \frac{1}{CLK} = \frac{1}{N\, t_{pd}}$

### 9.3.4 Power On Reset Circuit :

Whenever counter working was discussed, it was assumed that initially everything is reset. Whenever power supply is switched ON ; one cannot predict the output state of FF. Therefore Clear terminal is provided. But clear terminal requires some external circuitry to force it self 'LOW' so that it can provide Q = 0 (some known state). This external circuitry is nothing but power ON reset circuit. Refer Fig. 9.9.

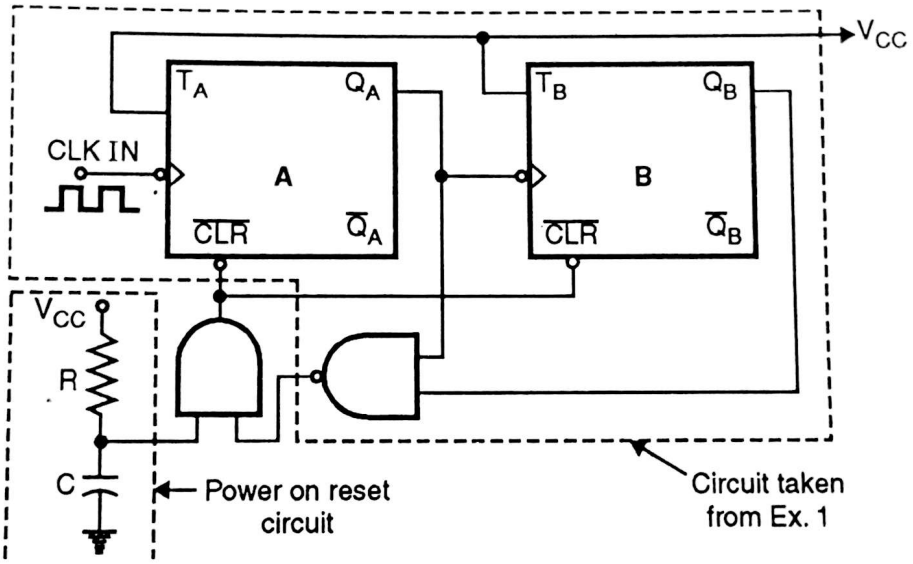



**Fig. 9.9**

As shown in Fig. 9.9, (counter states are 0, 1, 2). In addition one RC n/w with AND gate is used. So either of the input is LOW output of AND is LOW, FFs will be cleared.

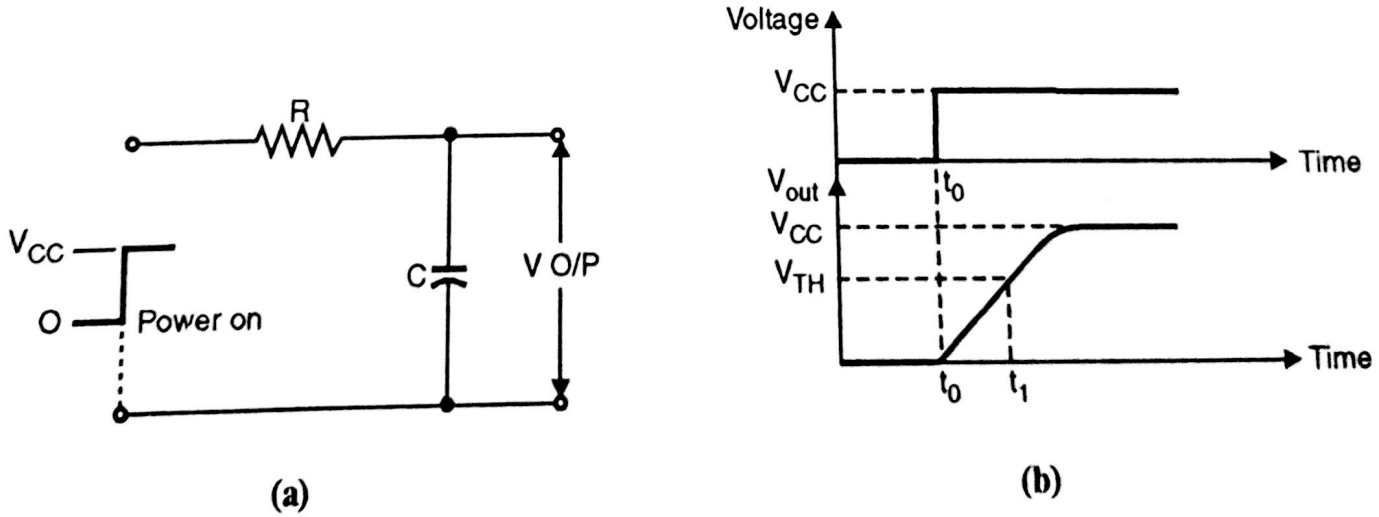



**Fig. 9.10**

Scanned by CamScanner

**Working :**

Refer Fig. 9.10 (a) and (b).

When power supply is switched ON, Vcc will appear across RC n/w at time $t_0$. ∴ Capacitor will start charging. The charging slope depends upon RC time constant. When capacitor voltage ($V_{output}$) reaches to level $V_{TH}$ (Threshold voltage), chip (IC) will consider it as HIGH logic. Below threshold it is considered as LOW logic. ∴ For duration $t_1 - t_0$ output of AND gate is '0' ∴ FFs are cleared.

When counter steps through states and AB = 11, output of NAND is equal to 0.

∴ $\overline{\text{CLEAR} \cdot \text{(CLK)}} = 0$, as AND output is '0' ∴ FF will reset.

**Ex. 4 :** Design 4 bit binary up/down ripple counter with control input of up/dn. (dn = down).

**Soln. :** Requirement :

(1) 4 bit ; ∴ 4 FFs are required.

(2) ripple counter.

(3) up/dn terminal ∴ up/dn. = 0 → up operation
up/dn. = 1 → down operation.

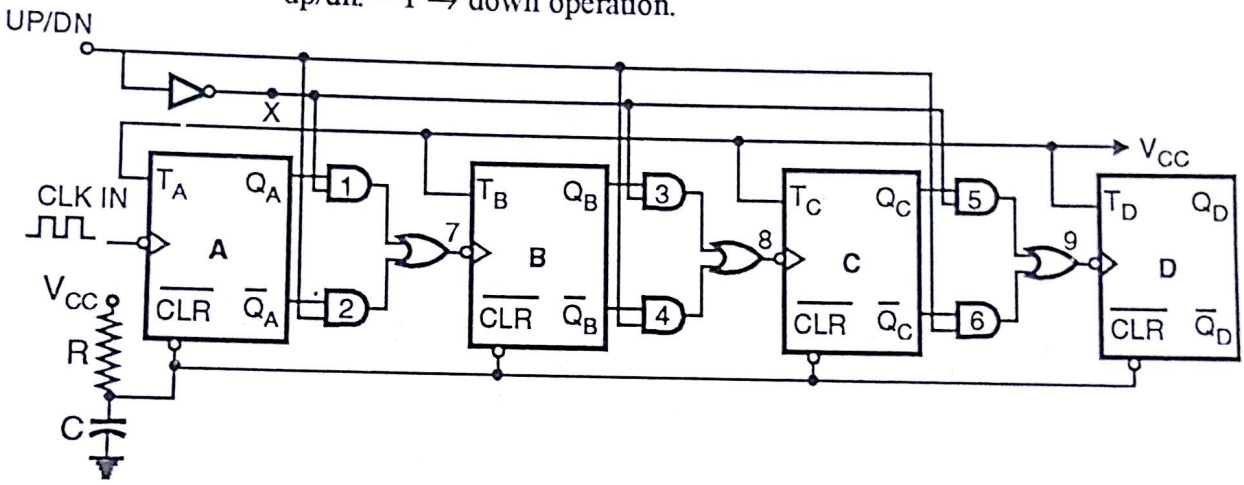



**Fig. Ex. 9.4(a) : 4 bit binary up/dn counter**

Regarding points (1) and (2), you are aware. Regarding point (3), we want one control terminal in such a way that counter will start either up or down counting. Recall equation studied in Section 9.2.1. Just by changing cascading from Q to $\bar{Q}$, counter changes from up to down counting. We are going to use same fundamental.

**Case I :** When up/dn control terminal = 0, x = 1. Therefore AND gate 1, 3, and 5 will be transparent, thus cascades Q output. Whereas AND gate 2, 4, and 6 produces continuous 'LOW' output irrespective of $\bar{Q}$. Therefore counter counts up.

$P_n = 1\ Q_1 = 1\ Q_t = 1,\ U_p = 1 \oplus 1 \oplus 1 = 1 \rightarrow$ up counter

**Case II :** When up/dn control terminal = 1, x = 0. Therefore AND gate 2, 4 and 6 will be transparent, thus cascades $\bar{Q}$ output. Whereas AND gate 1, 3 and 5 produces continuously 'LOW' output irrespective of Q. Therefore, counter counts down.

$P_n = 1\ Q_1 = 1\ Q_t = 0\ \therefore U_p = 1 \oplus 1 \oplus 0 = 0 \rightarrow$ down counter.

## 9.4 IC 7493 (4 bit binary counter) :

Let's study TTC MSI circuit of 4 bit ripple counter implemented in IC 7493.

Scanned by CamScanner

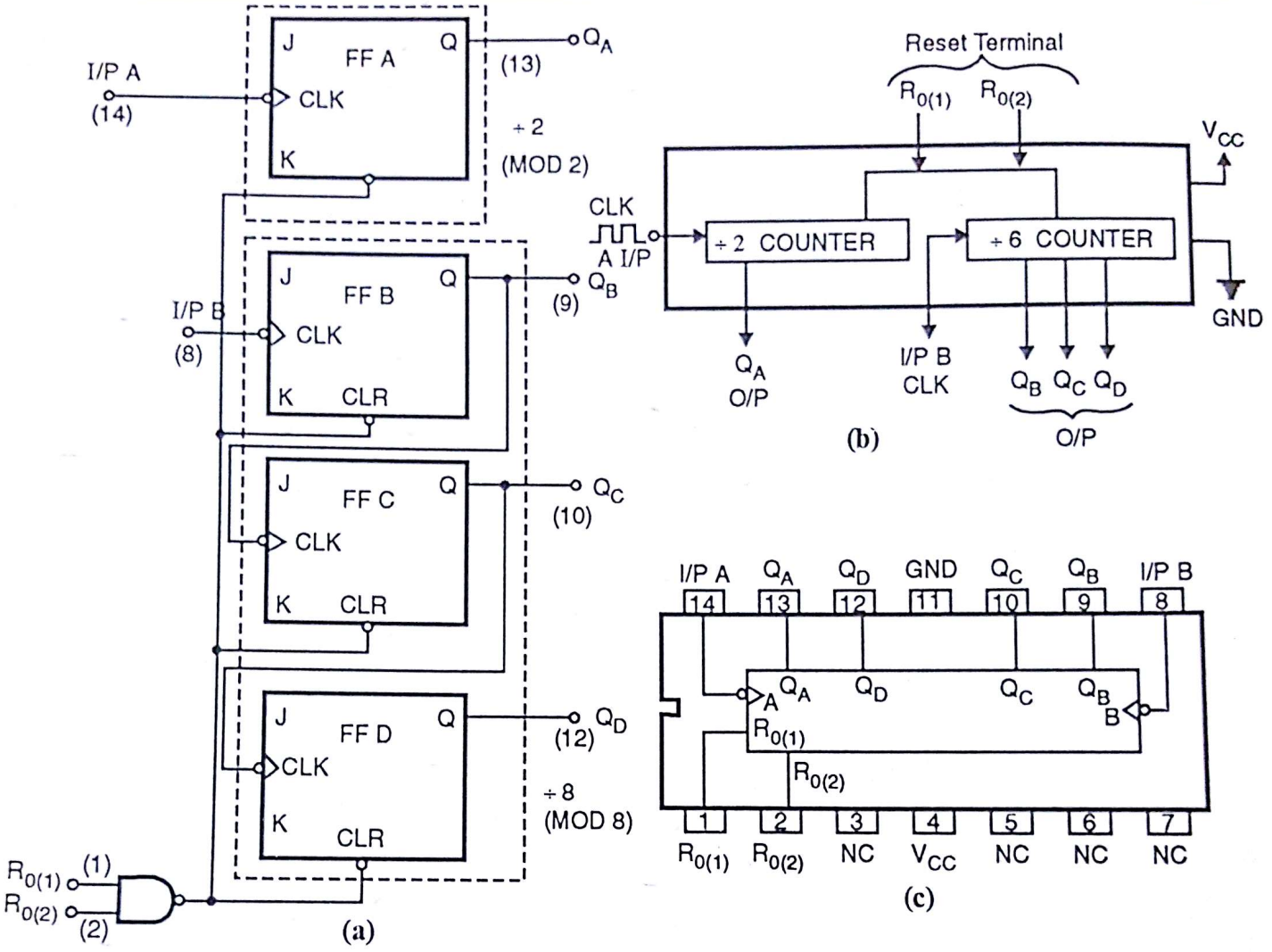



**Fig. 9.11**

In IC 7493,

$Q_D$ $Q_C$ $Q_B$ $Q_A$ ⇒ Binary output of FF.

Input A ⇒ CLK IN of FFA

Input B ⇒ CLK IN of FFB

$R_{0\,(1)}$, $R_{0\,(2)}$ ⇒ Reset pin

When $R_{0\,(1)} = R_{0\,(2)} = 1$ Reset will occur.

Fig. 9.11 shows logic diagram of IC 7493. It shows that FFA is independent i.e. output is not cascaded to next FF. But FF B, C and D are cascaded. ∴ we have individual, internally two counters, ÷ 2 (Mod 2) and ÷ 8 (Mod 8). If we combine both counter we get 2 × 8 = 16 state counter or mod 16 counter or ÷ 16 counter.

When we connect $Q_A$ output to input B and CLK IN given to Input A only, it becomes simple 4 bit binary ripple counter. Truth table as follows.

Scanned by CamScanner

| $R_{0(1)} / R_{0(2)}$ | CLK | $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ |
|---|---|---|---|---|---|
| 1 | × | 0 | 0 | 0 | 0 |
| 0 | ↓ | 0 | 0 | 0 | 0 |
| 0 | ↓ | 0 | 0 | 0 | 1 |
| 0 | ↓ | 0 | 0 | 1 | 0 |
| 0 | ↓ | 0 | 0 | 1 | 1 |
| 0 | ↓ | 0 | 1 | 0 | 0 |
| 0 | ↓ | 0 | 1 | 0 | 1 |
| 0 | ↓ | 0 | 1 | 1 | 0 |
| 0 | ↓ | 0 | 1 | 1 | 1 |
| 0 | ↓ | 1 | 0 | 0 | 0 |
| 0 | ↓ | 1 | 0 | 0 | 1 |
| 0 | ↓ | 1 | 0 | 1 | 0 |
| 0 | ↓ | 1 | 0 | 1 | 1 |
| 0 | ↓ | 1 | 1 | 0 | 0 |
| 0 | ↓ | 1 | 1 | 0 | 1 |
| 0 | ↓ | 1 | 1 | 1 | 0 |
| 0 | ↓ | 1 | 1 | 1 | 1 |

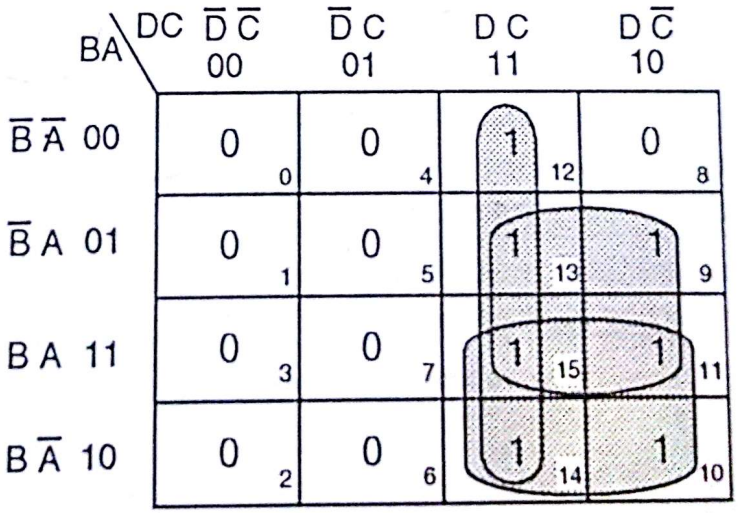
**Ex. 5 :** Implement ÷ 9 counter using IC 7493.

**Soln. :** Procedure is same as when we design mod – N ripple counter.

(1) ÷ 9 means mod – 9, ∴ required states are 9 (0 to 8) ∴ FF used are 4.

(2) Reset circuit should generate '0' output for valid states (0 to 8) and '1' output for invalid states (9 to 15), because if $R_{0\,(1)}$ and $R_{0\,(2)}$ both are 1, then only IC 7493 resets.

(3) Design reset circuit. Draw K-map.




$Y = BD + AD + CD = D\,[A + B + C]$

Scanned by CamScanner

(4) **Circuit :**

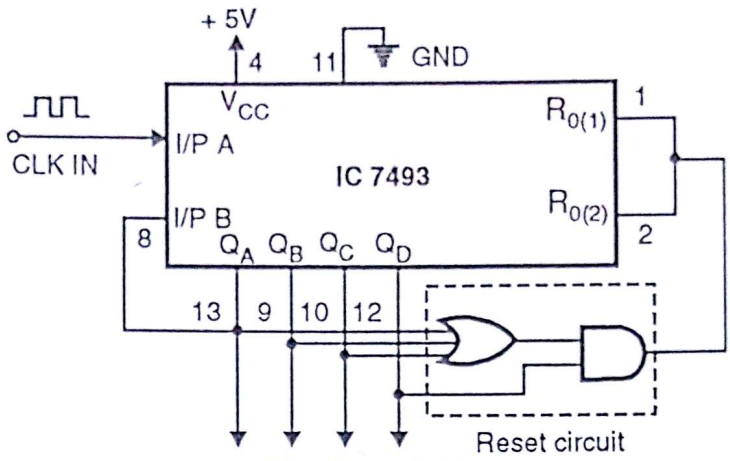



**Fig. Ex. 9.5 (a)**

## 9.5 Synchronous Counter :

In asynchronous counter mainly two problems were present, glitch and delay in counter. To avoid this, synchronous counter came into picture. Synchronous counter is now most widely used. Generalized block schematic of synchronous counter is shown in Fig. 9.12.

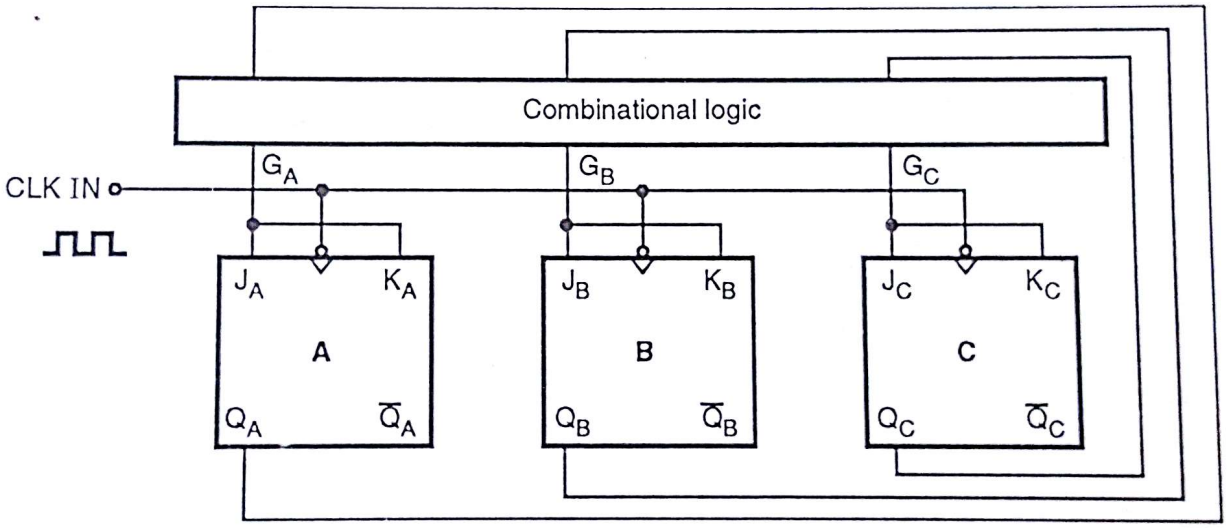



**Fig. 9.12 : Block schematic of synchronous counter**

Observation from block schematic :

(1) CLK pin of all FFs are tied together, so that FF changes output in synchronism.

(2) O/p Q given to combinational logic circuit. This circuit is designed in such a way that $G_A$ $G_B$ $G_C$ generated from it, applies proper logic to input of FF, so that we get correct next state.

(3) O/p Q depends upon previous input of FF, when CLK edge hits.

**Note :** Presently in block diagram, JK is tied together. But this will not be the case every time. JK can be controlled separately by combinational circuit.

Basically we are going to design synchronous counter using,

(1) T FF

(2) D FF

(3) JK FF (also JK master slave)

**Note :** Refresh excitation table of each FF.

Scanned by CamScanner

### 9.5.1 Type T Design :

Let's start designing synchronous counter using T FF.

**Ex. 6 :** Design mod – 4 regular sequential synchronous up counter. Using T FF.

**Soln. :**

(1) No. of states are 4, for mod – 4 counter.

(2) Regular means n = N.

(3) No. of FF required will be $4 \leq 2^m$ $\therefore$ $\boxed{m = 2}$

(4) Sequential and up means 0, 1, 2, 3.

(5) $\therefore$ state diagram is Fig. Ex. 9.6(a).

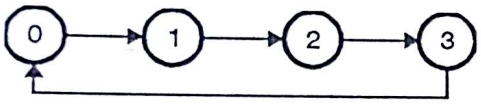



Fig. Ex. 9.6(a)

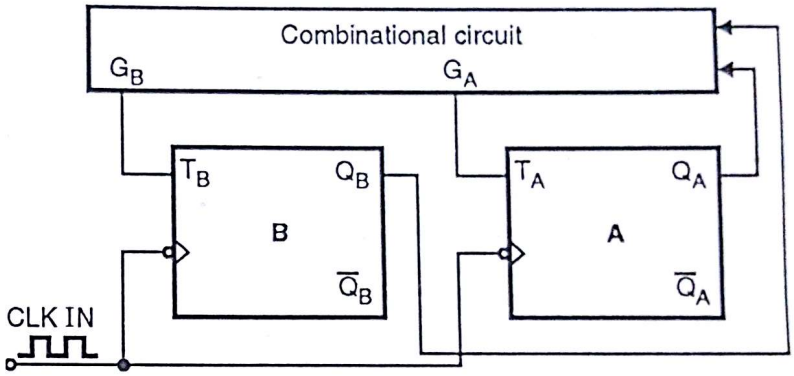



Fig. Ex. 9.6(b)

(6) FF A and B are used. $G_A$ and $G_B$ are output from combinational circuit to provide proper input to T FF. $\therefore$ circuit will be as shown in Fig. Ex. 9.6(b).

(7) Design of combination circuit to generate $G_A$, $G_B$.

**Step 1 :** For generating truth table, depending upon number of bits (FF), write down ***sequentially*** all states.

In our case 2 bits (FF) are used, $\therefore$ states are 4.

| | | | | | | |
|---|---|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 | ; | A = MSB |
| B | 0 | 1 | 0 | 1 | ; | B = LSB |
| decimal | 0 | 1 | 2 | 3 | | |

Here $Q_A = A, Q_B = B$ for simplicity

**Step 2 :** Now below this you write down required output.

| | | | | |
|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 1 |
| | (Prev.) | (Next) | | |
| $G_A$ | | | | |
| $G_B$ | | | | |

**Note :** Prev. = Previous

**Step 3 :** Now start from first state of counter. Treat this state as previous state. Take next state of counter from state diagram, and it will be obviously next state.

Scanned by CamScanner

In our case 1st state = AB = 00 ⇒ Previous state.

2nd state = AB = 01 ⇒ Next state.

**Step 4 :** Now refer excitation table of T FF, treat change of state for A and B ***individually*** and write down $G_A$ and $G_B$. The step is explained as follows :

| | | | | | |
|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 0 | 1 |
| $G_A$ | 0 | | | | |
| $G_B$ | 1 | | | | |

As A changes from $0 \rightarrow 0$ $\therefore G_A\ (T_A) = 0$

B changes from $0 \rightarrow 1$ $\therefore G_B\ (T_B) = 1$

**Step 5 :** Now treat previously next as previous state. i.e. AB = 01 which was next state previously, will be *NOW* previous state. Refer state diagram and find next state. In our case it will be AB = 10. Again the procedure is same as STEP 4.

| | | | | |
|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 1 |
| | | [Prev.] | [Next] | |
| $G_A$ | 0 | 1 | | |
| $G_B$ | 1 | 1 | | |

As A changes from $0 \rightarrow 1$, $\therefore G_A\ (T_A) = 1$

B changes from $1 \rightarrow 0$, $\therefore G_B\ (T_B) = 1$

**Step 6 :** Now previous state = AB = 10 and Next state = AB = 11. again follow STEP 4.

| | | | | |
|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 1 |
| | | | [Prev.] | [Next] |
| $G_A$ | 0 | 1 | 0 | |
| $G_B$ | 1 | 1 | 1 | |

As A changes from $1 \rightarrow 1$, $\therefore G_A\ (T_A) = 0$

B changes from $0 \rightarrow 1$, $\therefore G_B\ (T_B) = 1$

**Step 7 :** Finally previous state = 11 and next state = 0 0. Next we are completing full chain of counter. Follow step 4.

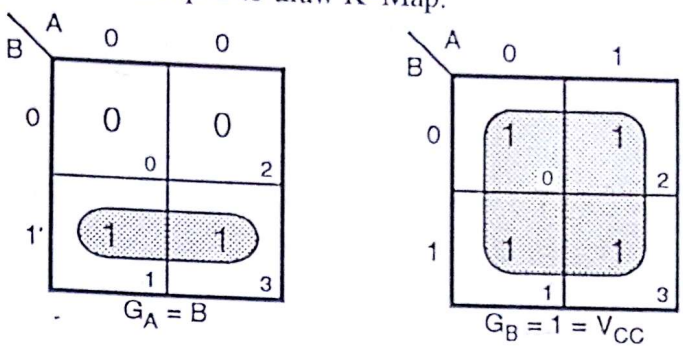
| | | | | |
|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 1 |
| | [Next] | | | [Prev.] |
| $G_A$ | 0 | 1 | 0 | 1 |
| $G_B$ | 1 | 1 | 1 | 1 |

As A changes from $1 \rightarrow 0$, $\therefore G_A\ (T_A) = 1$

B changes from $1 \rightarrow 0$, $\therefore G_B\ (T_B) = 1$.

Scanned by CamScanner

**Step 8 :** After filling truth table, next step is to draw K–Map.




**Step 9 :** Draw final circuit diagram.

Already we have drawn half circuit as shown in Fig. Ex. 9.6(b). Only thing is to replace combinational block by directly some connection.

**Note :** In Ex. 6 is so easy that we haven't got complicated boolean expression for $G_A$ and $G_B$. In next example we will see that, block is replaced by AND – OR – INV Matrix.

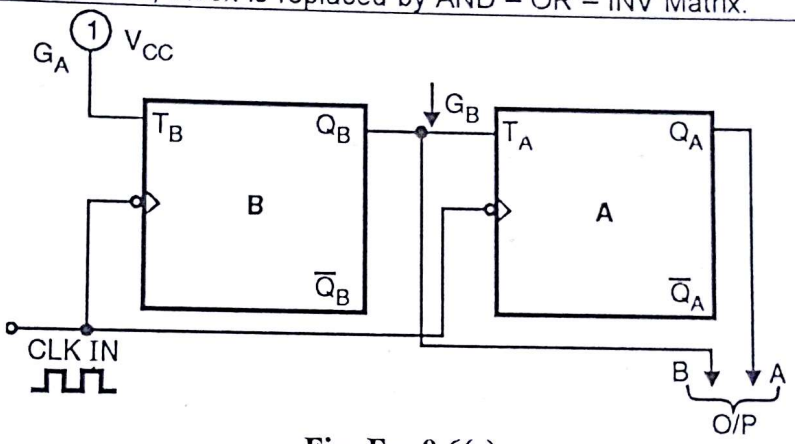



**Fig. Ex. 9.6(c)**

Let's draw the waveforms and analyse working of circuit.

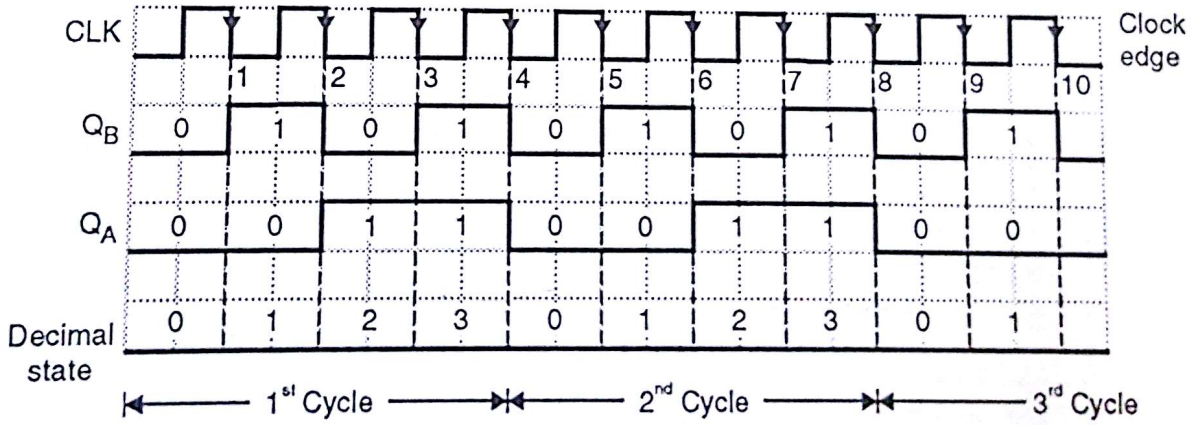



**Fig. Ex. 9.6(d)**

**Working :**

(1) Initially counter is Reset, $\therefore$ **AB = 00**

(2) At 1st CLK edge, $T_A = 0$ as B = 0, $T_B = 1$.
$\therefore$ A remains constant as $Q_B$ toggles from 0 → 1. $\therefore$ **AB = 01**

(3) At 2nd CLK edge, as $T_A = 1$ (because B = 1) and $T_B = 1$,
A and B both will toggle, giving $\therefore$ **AB = 10**

Scanned by CamScanner

(4) At 3rd CLK edge, $T_A = 0$ as B = 0 and $T_A = 1$,

$\therefore$ A = unchanged, B = toggle $\therefore$ **AB = 11**

(5) Finally at 4th CLK edge, $T_A = 1$ and B = 1, $T_B = 1$

$\therefore$ A and B both will toggle from $1 \rightarrow 0$. $\therefore$ **AB = 00**

Now the point (2), (3), (4) and (5) will continue till CLK is present, and counter generates required states.

If you compare the waveform in Fig. Ex. 9.6(d) with waveforms of ripple counter, you will find that falling edge is shown only on CLK, not on $Q_A$ or $Q_B$ output, because CLK is synchronized. Counter changes state ONLY when CLK edge hits.

Here you MAY HAVE DOUBT OR QUERY that when 1st CLK edge hits, B changes from $0 \rightarrow 1$. $\therefore T_A = 1$ and $\therefore$ A should also toggle.

The answer for the doubt is B changes from $0 \rightarrow 1$ ***after propagation delay of 1 FF,*** then it gets settled down and after that signal is capable to toggle A. But till B gets settled to 1, ***CLK edge duration*** is over, because edges are sharp and of short duration. $\therefore$ A ***cannot*** toggle depending upon next state of B, but changes according to previous state of B. This point is very very important.

**Note :** Now onwards we are going to solve more example. But the methodology will be same either it is T FF or D FF or JK FF or SR FF.

**Ex. 7 :** Design synchronous counter for state diagram shown in Fig. Ex. 9.7.

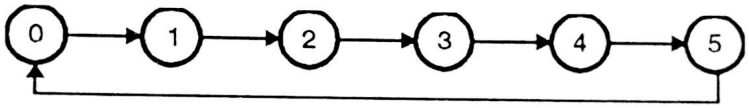



**Fig. Ex. 9.7**

**Soln. :**

(1) Counter is sequential, but truncated

(2) No. of states n = 6. $\therefore n = 6 \leq 2^m$, $\therefore m = 3$

As m = 3, $2^3 = 8 = N$ $\therefore n < N$ (truncated counter).

(3) Up counter (observe from given sate diagram)

(4) Truth table for designing combinational logic circuit.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | 0 → 0 → 0 → 0 → 1 → 1 | | | | | | 1 | 1 |
| B | 0 → 0 → 1 → 1 → 0 → 0 | | | | | | 1 | 1 |
| C | 0 → 1 → 0 → 1 → 0 → 1 | | | | | | 0 | 1 |
| $G_A$ | 0 | 0 | 0 | 1 | 0 | 1 | X | X |
| $G_B$ | 0 | 1 | 0 | 1 | 0 | 0 | X | X |
| $G_C$ | 1 | 1 | 1 | 1 | 1 | 1 | X | X |

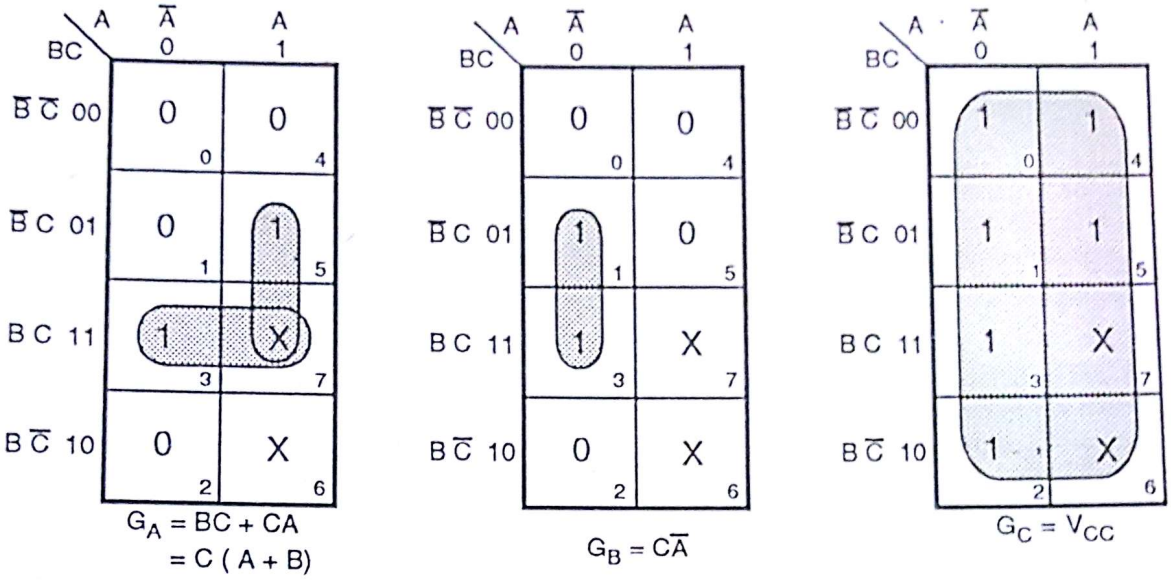
$\rightarrow V_{CC}$

**Fig. Ex. 9.7(a)**

**As shown in Fig. Ex. 9.7(a) truth table, valid states are only 0, 1, 2, 3, 4, and 5. Therefore, for invalid states i.e. 6 and 7, $G_A$, $G_B$ and $G_C$ are takes *don't care* conditions because we know that these states are not going to occur.**

Scanned by CamScanner

(5) K-Map for $G_A$, $G_B$ and $G_C$ :




(6) Circuit :

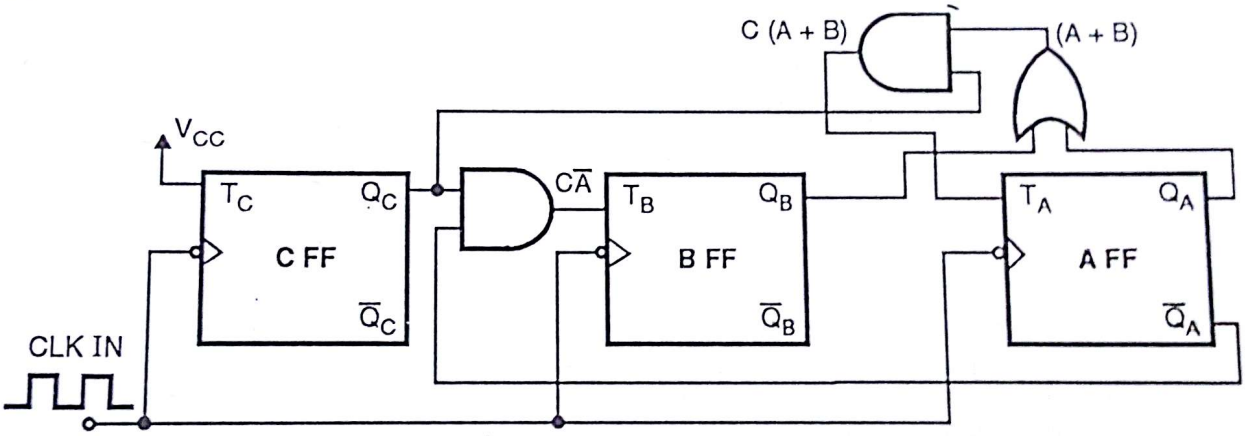



Fig. Ex. 9.7(b)

(7) Waveforms :

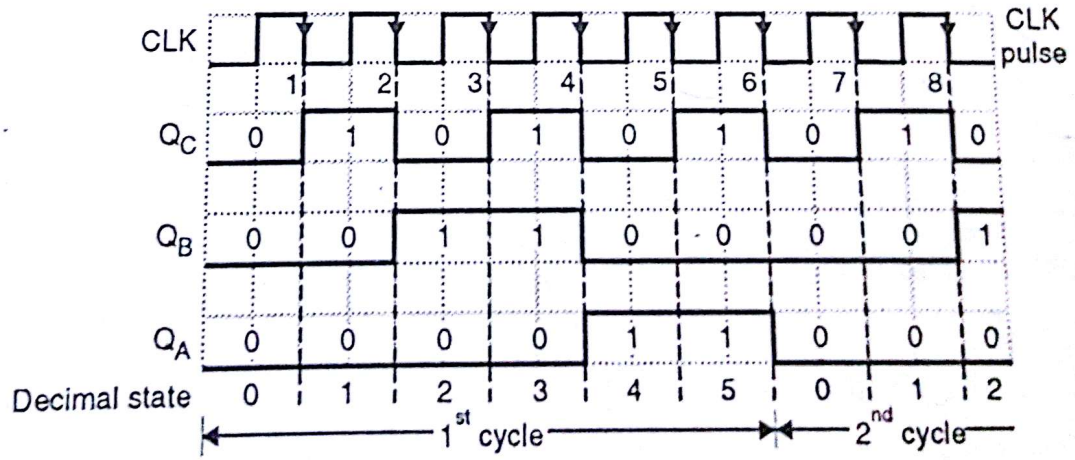



Fig. Ex. 9.7(c)

**Ex. 8 :** Design synchronous, non-sequential counter.

Scanned by CamScanner

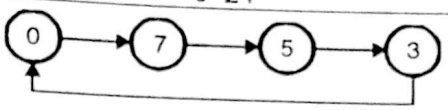



Fig. Ex. 9.8

**Soln. :**

(1) Counter is nonsequential as well as truncated.

(2) While designing non sequential counter, instead of number of state you should consider maximum count of the state. In our case maximum count of a state is 7. To represent 7 in binary we require 3 bits (111) and therefore $\boxed{m = 3}$.

As m = 3. $n = 2^3 = 8$

(3) While writing truth table as told in Ex. 6, (7), step I, depending upon number of bits (FF), write down states ***sequentially***. In our case N = 8 as m = 3.
So we are suppose to write down 0 to 7 sequentially, refer Fig. Ex. 9.8(a)(Truth table)

(4) Truth table.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0–7 ⓐ | | | | | | 7–5 ⓑ | | |
| A | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | → MSB |
| | | 3–0 ⓓ | | | 5–3 ⓒ | | | | |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| C | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | → LSB |
| | 0 – 7 | | | 3 – 0 | | 5 – 3 | | 7 – 5 | |
| $G_A$ | 1 | X | X | 0 | X | 1 | X | 1 | |
| $G_B$ | 1 | X | X | 1 | X | 1 | X | 1 | → $V_{CC}$ |
| $G_C$ | 1 | X | X | 1 | X | 0 | X | 1 | |

**Fig. Ex. 9.8(a)**

**Step 1 :** Write standard table A, B, C and $G_A$, $G_B$, $G_C$.

**Step 2 :** Write 0 – 7 states ***sequentially*** under A, B, C.

**Step 3 :** Write down X (don't care) for invalid states. For our case invalid states are 1, 2, 4, 6, ∴ for 1, 2, 4, 6, $G_A$, $G_B$, and $G_C$ are X (don't care).

**Step 4 :** Now the steps are standard.
Start from 1st state of state diagram i.e. 0. Next state is 7.
So the transition is as shown in Fig. Ex. 9.8(a) by (a). Previous state of ABC = 000, Next state of ABC = 111.
∴ $G_A$, $G_B$ and $G_C$ = 111, written under column '0 – 7' (shown in Fig. Ex. 9.8(a)).

**Step 5 :** 2nd transition is from 7 – 5 (shown by arrow (b) ).
ABC changes from 111 to 101. ∴ $G_A = 0$, $G_B = 1$ and $G_C = 0$, written under column '7 – 5'

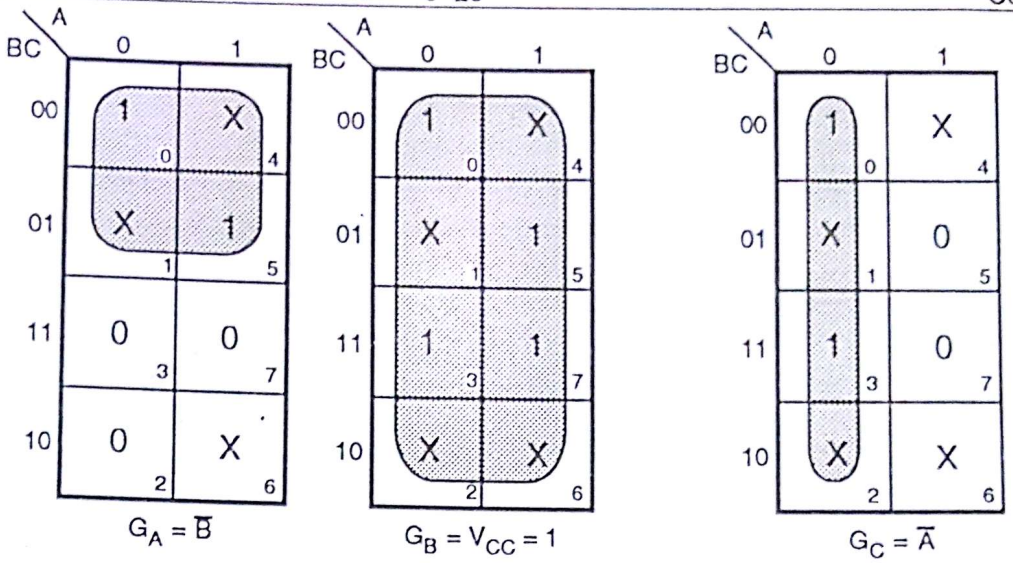
**Step 6 :** 3rd transition is from 5 – 3 (shown by arrow (c) ) ABC changes from 101 to 011. ∴ $G_A = 1$, $G_B = 1$ and $G_C = 0$, written under column '5 – 3'.

**Step 7 :** Final transition is from 3 to 0 (3 → 0) (shown by arrow (a) ).
ABC changes from 011 to 000. ∴ $G_A = 0$ $G_B = 1$ $G_C = 1$, written under column '3 – 0'

**Step 8 :** After going through one full cycle draw K-Map.

Scanned by CamScanner

(5) K-Map :




(6) Circuit diagram :

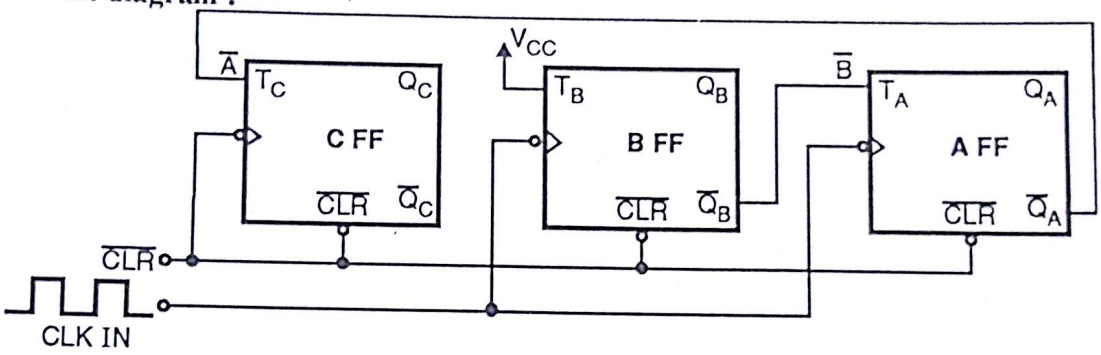



**Fig. Ex. 9.8(b)**

## 9.6 Bushing : ➥ [ASKED IN EXAM - MAY 96, MAY 97, DEC. 97 !!! ]

If you observe examples solved uptill now, in full sequence '0' is always present and once counter enters in valid state, it will continue the chain unless and untill power supply is switched off. To enter in the chain we use power on reset circuit, therefore first state is normally '0'.

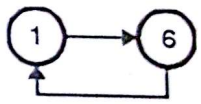



**Fig. 9.13(a)**

**Problem :**

(1) In some of the applications we don't require '0' state at all.

(2) Secondly due to power supply fluctuation (glitch of power supply), Electromagnetic interference and RFI (radio frequency interference), it may happen that counter will enter in invalid state. In Ex. 8. invalid states are 1, 2, 4, and 6.

***If due to problem stated above it enters in to invalid condition then what will happen ?***

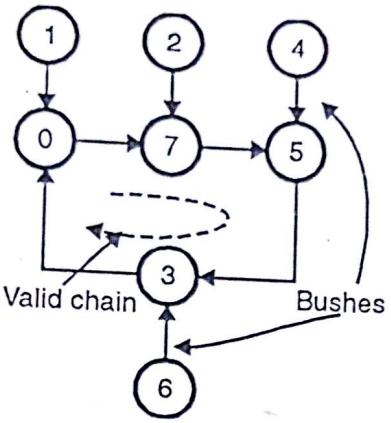
Here your luck factor counts.

**(1) *LOCK OUT STATE*** (IF LUCK IS BAD).

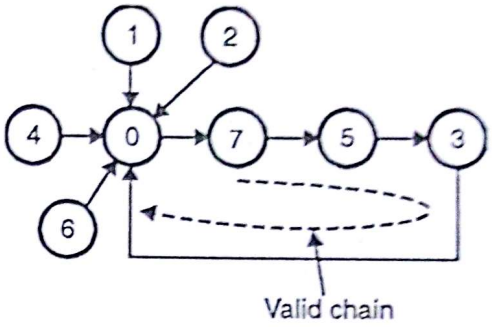
**It may happen that counter will lock itself in invalid states only. Let's say in above case counter enters into state 1, because of logic circuit next state happens to be 6. After 6 next state happens to be 1 (oh, no). Then counter will toggle between two state only, as shown in state diagram.**

Scanned by CamScanner

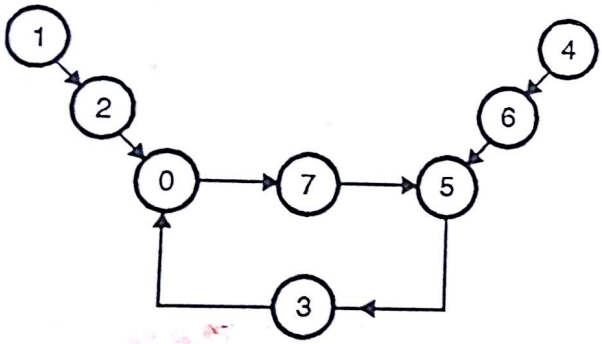
(2) If you are lucky enough after entering into invalid state it may switch over to valid state. But don't try to check your luck. Take precaution. As a precautionary step we implement *bushing* to state diagram. Bushing means branches. We are going to branch invalid states in such a way that even though it enters into invalid state after one, two or three CLK ticks it will enter into valid state chain. Following are some of the examples for implementing bushing in Ex. 8 :



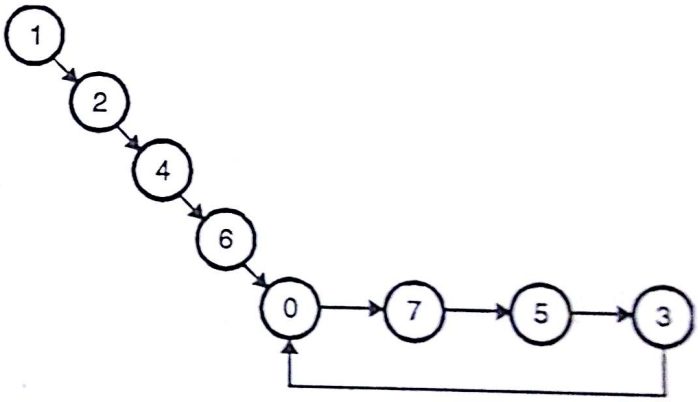

(b) Enters in valid chain after one CLK




(c) Enter in valid chain after one CLK




(d) Enters into valid state after maximum 2 CLK ticks




(e) Enters after 4 CLK tick

**Fig. 9.13**

Thus one can have 'n' number of combinations. Choice depends upon designer.

**Ex. 9 :** Implement synchronous counter for state diagram shown in Fig. Ex. 9.9.

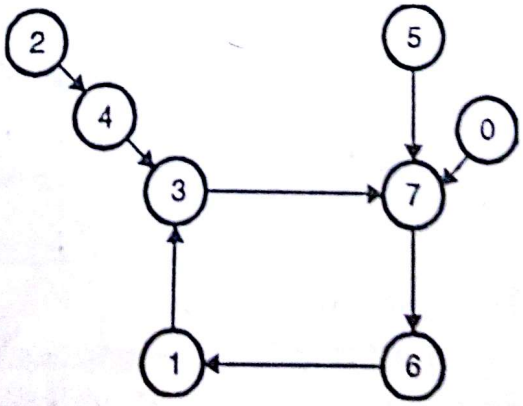



**Fig. Ex. 9.9**

Scanned by CamScanner

**Soln. :** Procedure for finding FF, truth table is same as previous example. The problem is states with bushing.

(1) No. of FFs are 3 as max. state of counter is '7'.

(2) Busing branches are (2) → (4) and (5) and (0).

(3) **Truth table is as follows :**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| C | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | 0 – 7 | 1 – 3 | 2 – 4 | 3 – 7 | 4 – 3 | 5 – 7 | 6 – 1 | 7 – 6 |
| $G_A$ | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| $G_B$ | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| $G_C$ | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

0–7, 6–1, 2–4, 4–3, 1–3, 3–7, 5–7

**Fig. Ex. 9.9(a)**

As shown in Fig. Ex. 9.9(a) truth table, you can start initially with main loop i.e. 3, 7, 6, 1, 3. Procedure is same as previous one. The figure shows 0 – 7 means previous state is 0 and next is 7.

After completing main loop, you can take branch by branch.

For our case,

First branch is 2, 4, 3. $2 \xrightarrow{\text{to}} 4$ and $4 \xrightarrow{\text{to}} 3$.

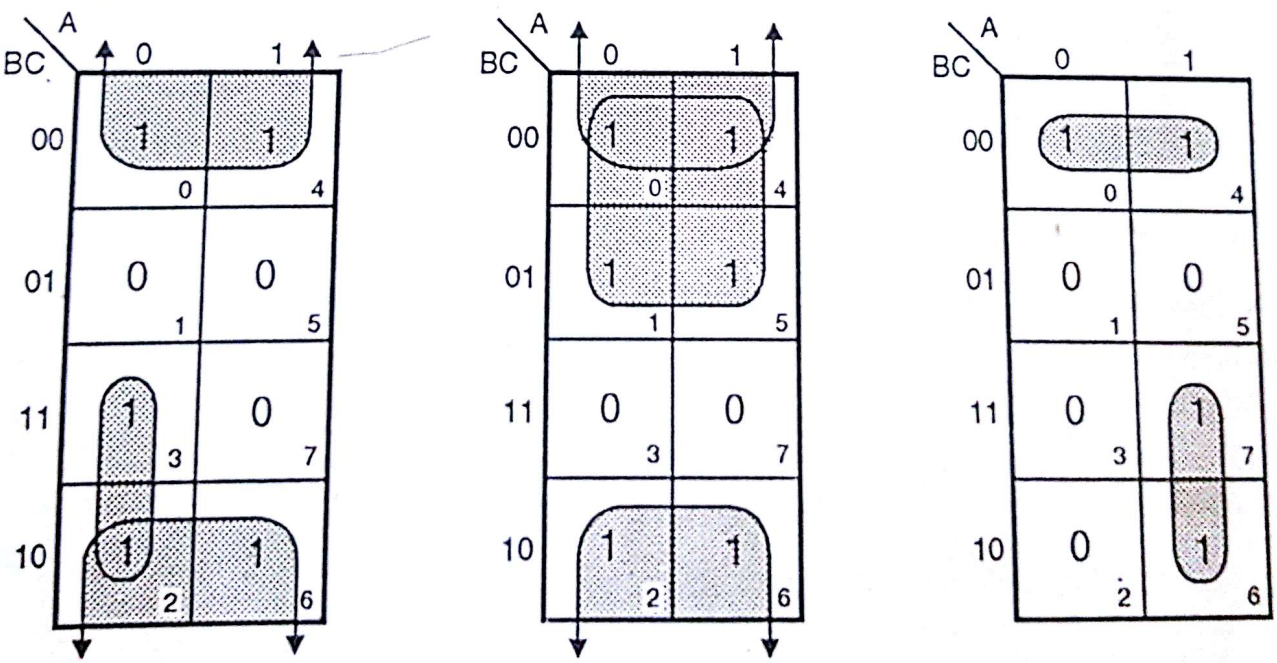
Second branch is 5, 7. $5 \xrightarrow{\text{to}} 7$

Third branch is 0, 7 $3 \xrightarrow{\text{to}} 7$

Accordingly you write down $G_A$, $G_B$ and $G_C$.

MULUND COLLEGE OF COMMERCE LIBRARY DSC 743

**(4) K-Map :**




$G_A = \overline{C} + \overline{A}B$ $G_B = \overline{B} + \overline{C}$ $G_C = AB + \overline{BC}$

Scanned by CamScanner

(5) **Circuit diagram :**

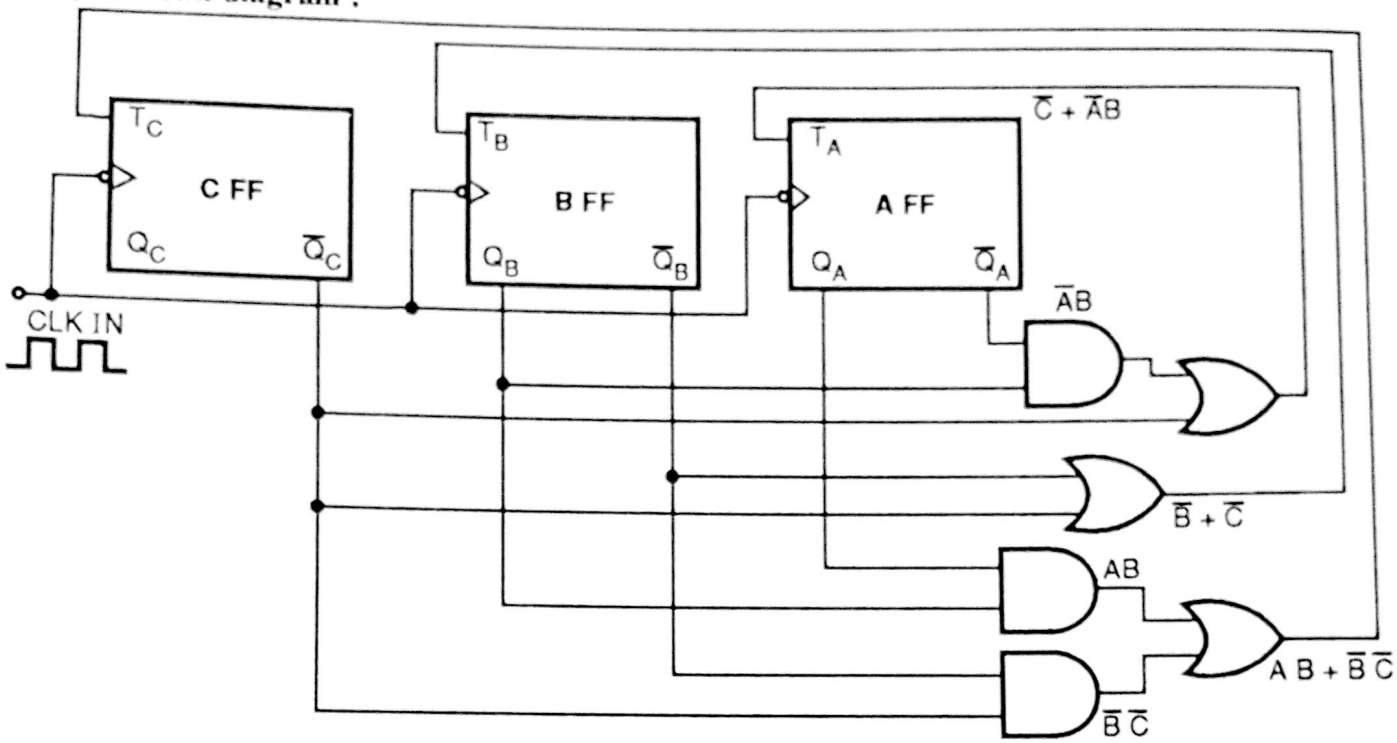



**Fig. Ex. 9.9(b)**

## 9.7 Type D Design :

Excitation table for D FF is already known to us. Methodology for designing counter using D FF is same as that of T FF. You track for changes in the state and accordingly write $G_A$, $G_B$ and $G_C$ and so on.

**Ex. 10 :** Design synchronous counter for state diagram shown in Fig. Ex. 9.10 using D FF.

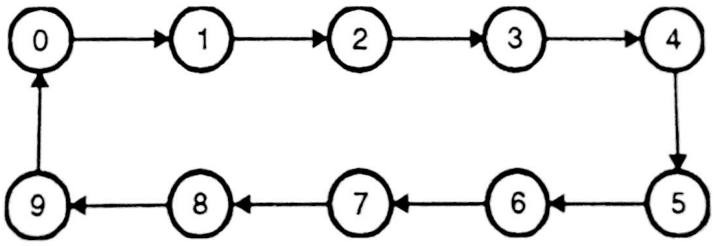



**Fig. Ex. 9.10**

**Soln. :** (1) Given counter is Mod – 10 or decade counter.

(2) $9 \leq 2^m$

$\therefore \boxed{m = 4}$ No. of FFs are '4'.

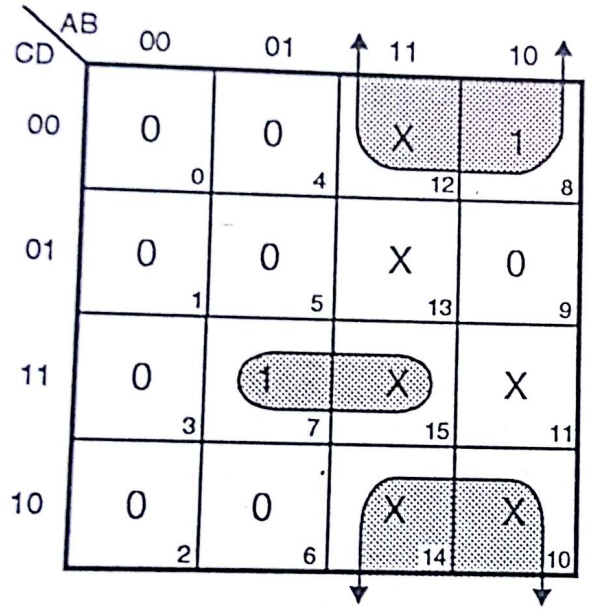
$\therefore N = 2^m = 2^4 = 16.$

(3) While writing truth table write all states sequentially.

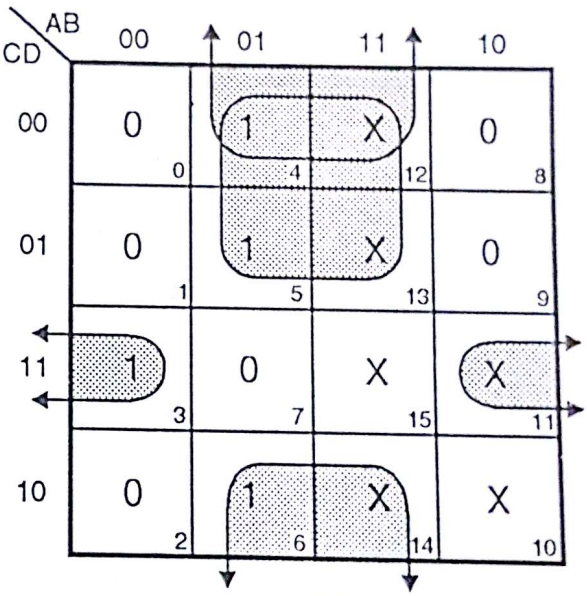
Scanned by CamScanner

| | | | | | | | | | | | Invalid States | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | 1 | 1 | 1 | 1 | 1 | 1 |
| B | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| C | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| D | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $G_A$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X |
| $G_B$ | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | X | X | X | X | X | X |
| $G_C$ | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | X | X | X | X | X | X |
| $G_D$ | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | X | X | X | X | X | X |

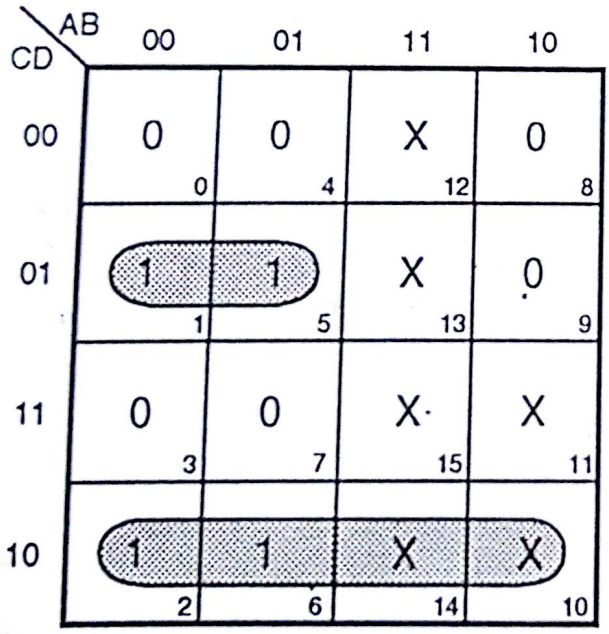
**K-Map :**



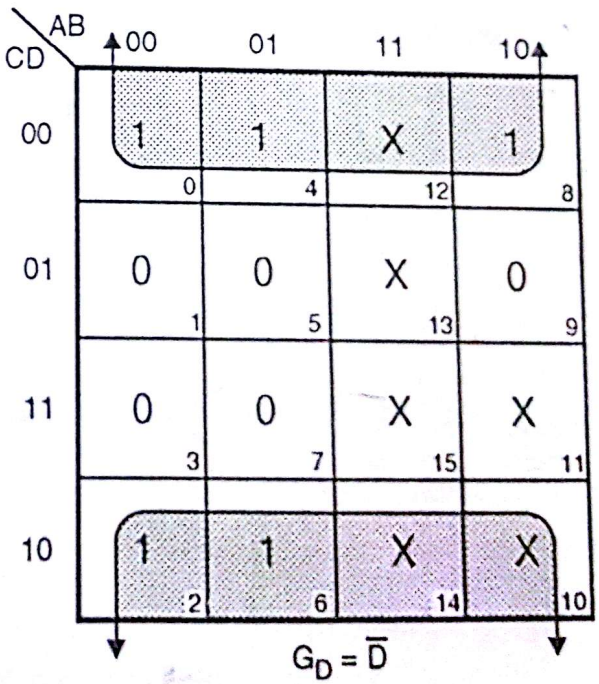

$G_A = A\overline{D} + BCD$




$G_B = B\overline{C} + \overline{B}CD + B\overline{D}$




$G_C = C\overline{D} + \overline{C}D\overline{A}$




$G_D = \overline{D}$

Scanned by CamScanner

**Circuit diagram :**

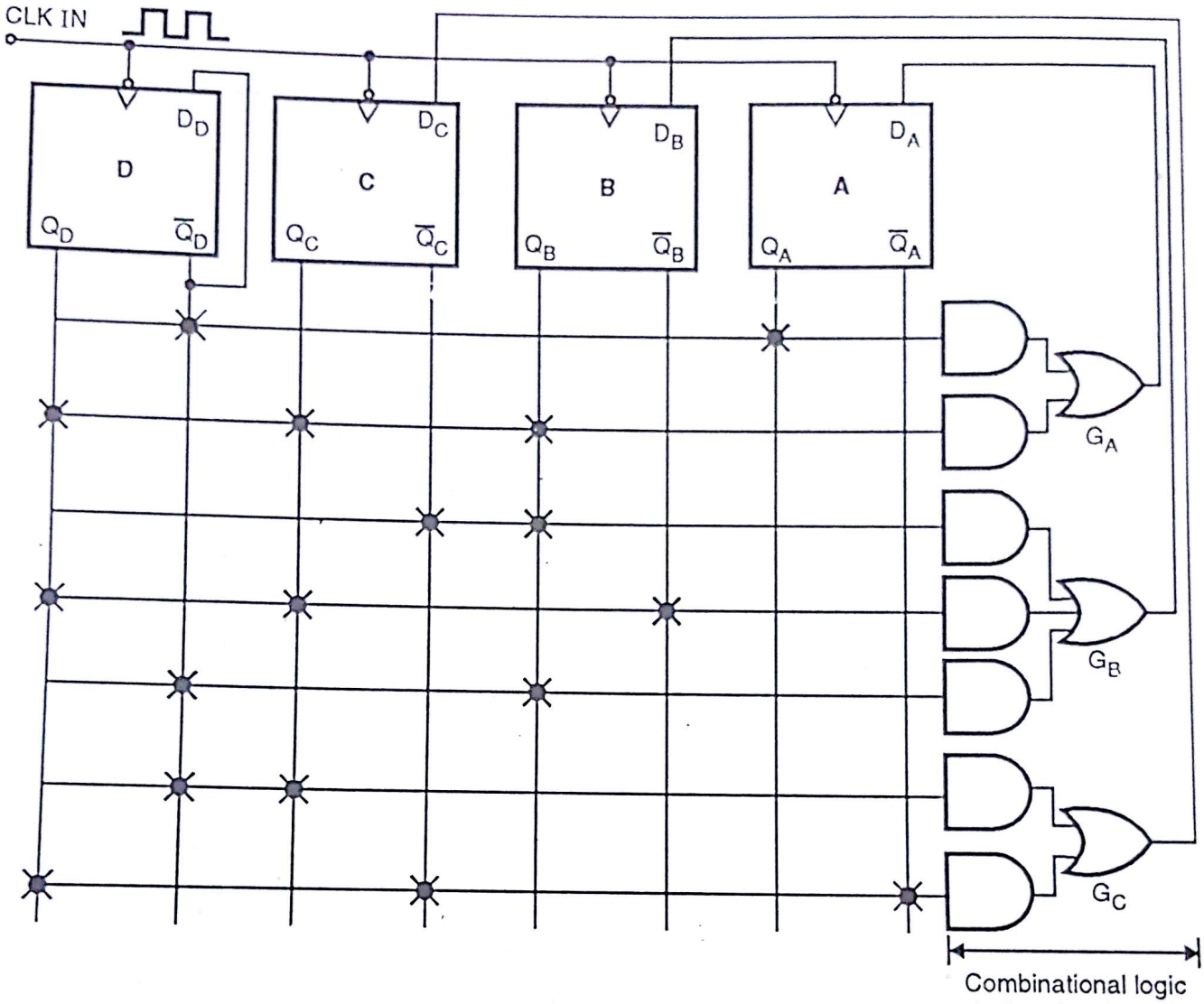



**Fig. Ex. 9.10(a)**

---

**Ex. 11 :** Design synchronous counter using D FF.

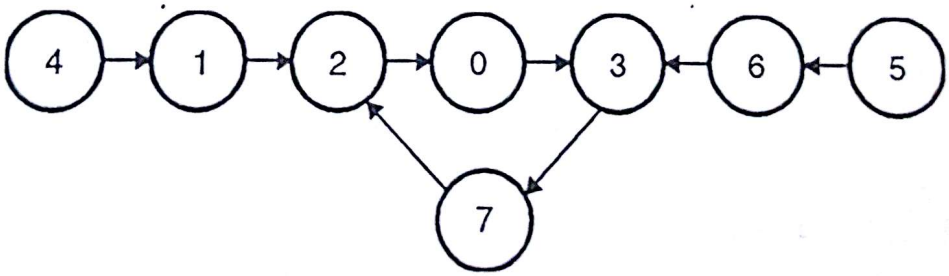



**Fig. Ex. 9.11**

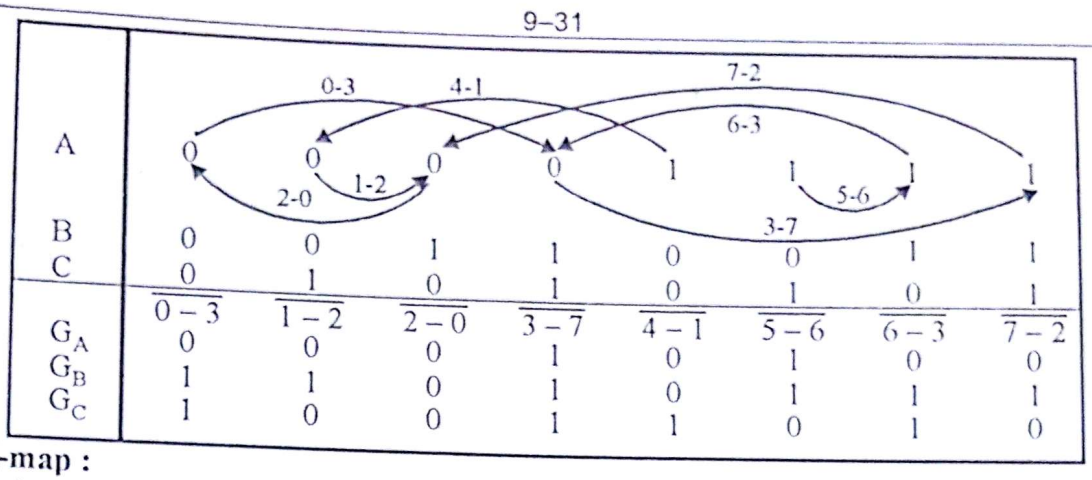
**Soln. :**

(1) Counter is non sequential with busing.

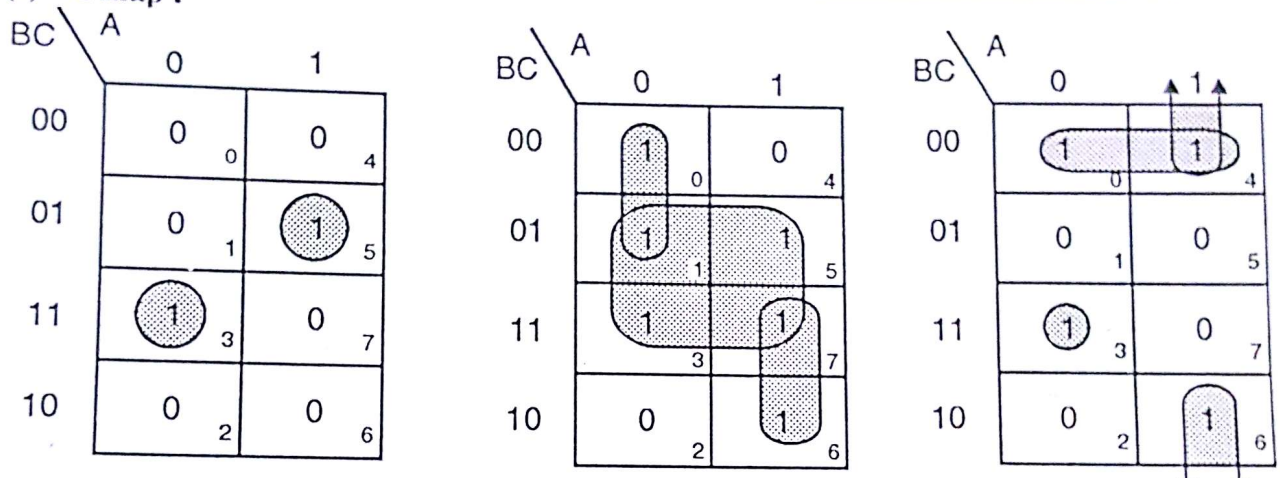
(2) No. of FFs required are $7 \leq 2^m$ $\therefore$ m = 3.
$2^m = N = 2^3 = 8 \therefore$ 0 – 7 states are present.

(3) Truth table.

Scanned by CamScanner




(4) **K-map :**




$G_A = \overline{A}BC + A\overline{B}C$
$= C(\overline{A}B + A\overline{B}) = C(A \oplus B)$

$G_B = C + \overline{A}\overline{B} + AB$
$= C(\overline{A \oplus B})$

$G_C = \overline{B}\overline{C} + \overline{A}BC + A\overline{C}$
$= \overline{C}(A + \overline{B}) + \overline{A}BC$

(5) **Circuit diagram :**

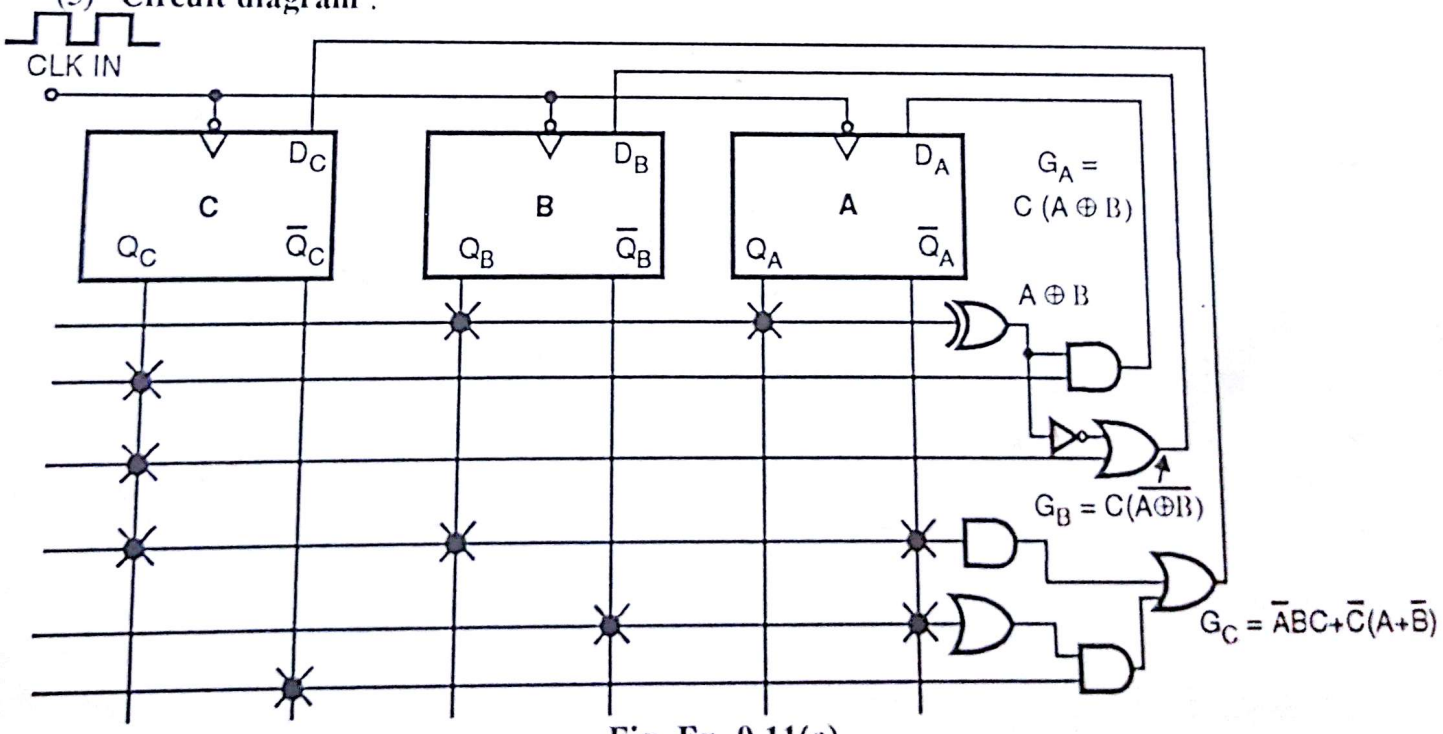



**Fig. Ex. 9.11(a)**

**Ex. 12 :** Design counter to go through the following states 0–1–3–4–6–0. Draw the logic diagram.

Scanned by CamScanner

**Soln. :** (1) No. of FFs required will be 3.

(2) **State diagram** : Refer Fig. Ex. 9.12.

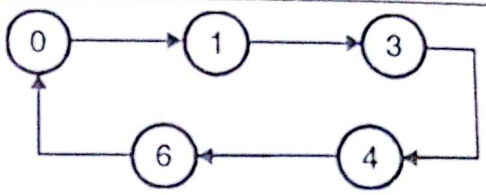
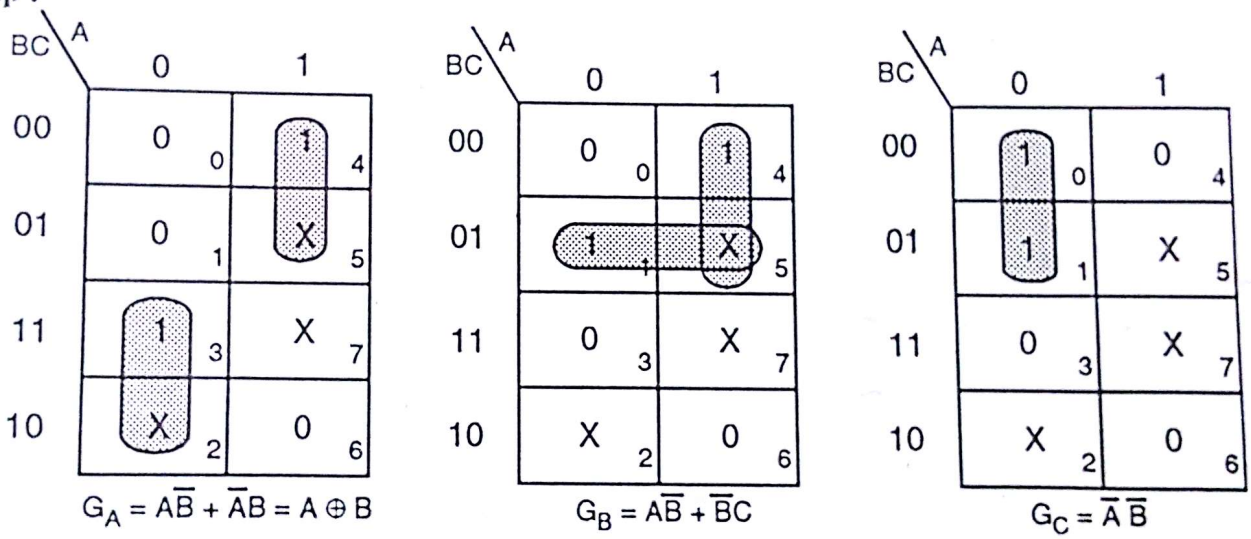



Fig. Ex. 9.12

(3) **Truth table :**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| C | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $G_A$ | 0 | 0 | X | 1 | 1 | X | 0 | X |
| $G_B$ | 0 | 1 | X | 0 | 1 | X | 0 | X |
| $G_C$ | 1 | 1 | X | 0 | 0 | X | 0 | X |

(4) **K-map :**




(5) **Circuit diagram :**

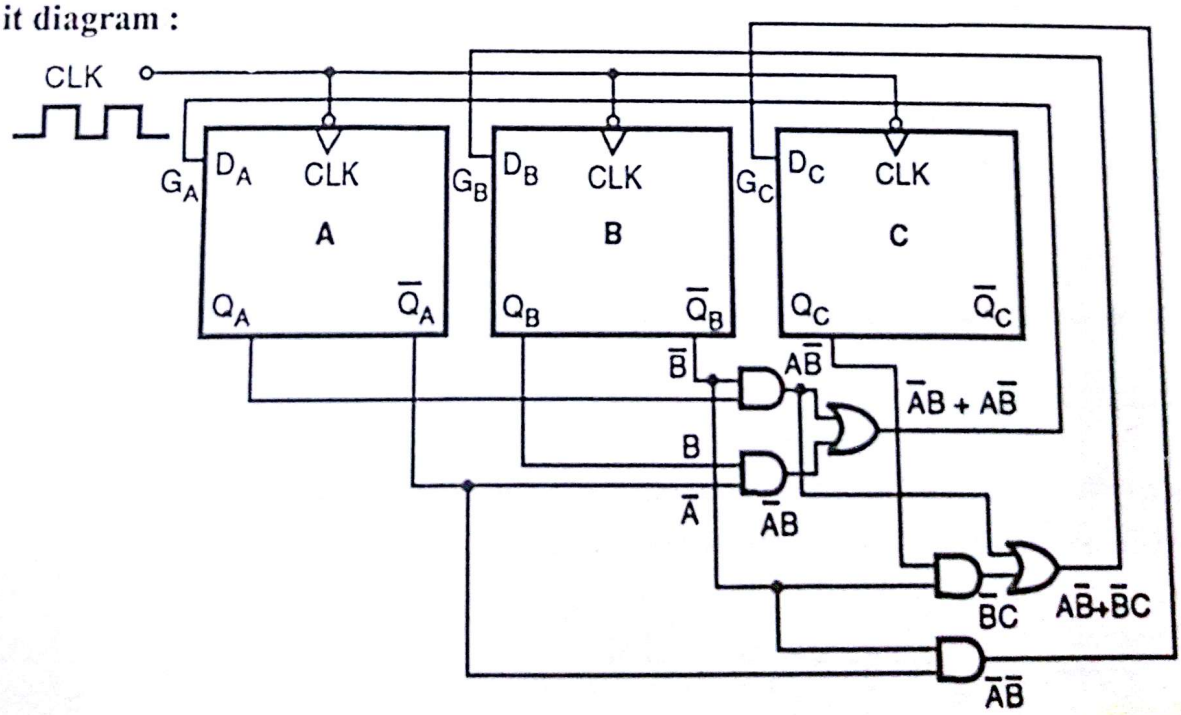



**Fig. Ex. 9.12(a)**

Scanned by CamScanner

## 9.8 Type JK Design :

Counter design using JK-FF is most widely used. Here J and K terminals are controlled separately. We have already generated excitation table for JK-FF.

As far as counter design goes, method is same as we followed uptill now. Fig. 9.14 shows generalized block schematic for counter using JK.

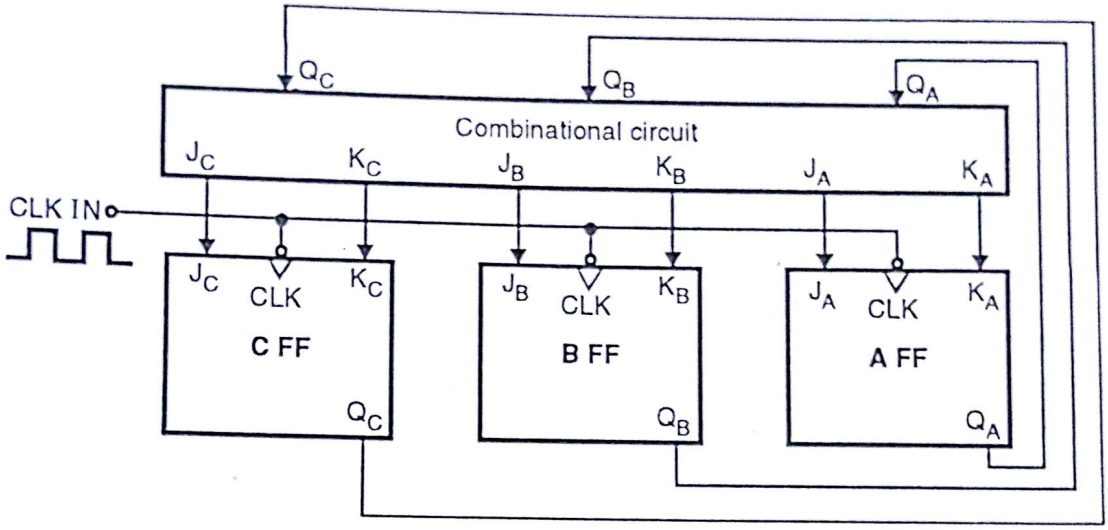



**Fig. 9.14**

As shown in Fig. 9.14.

(1) CLK IN of all FFs are tied together. (synchronous counter)

(2) $Q_A$, $Q_B$, $Q_C$ outputs are fed to combinational circuit.

(3) Depending upon next state combinational circuit generates proper J and K input for A, B and C FF.

**Note :** Number of FFs may be of any numbers.

---

**Ex. 13 :** Design a modulo – 6 counter using JK FF. Explain its action by writing a truth table and draw wavefoms for the outputs of the flip flops.

**Soln. :** (1) State diagram. Modulo 6 means 0 to 5.

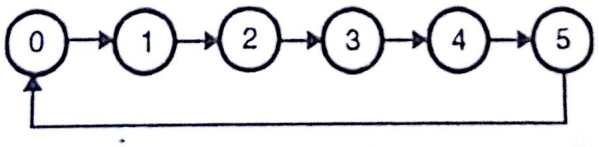



**Fig. Ex. 9.13(a)**

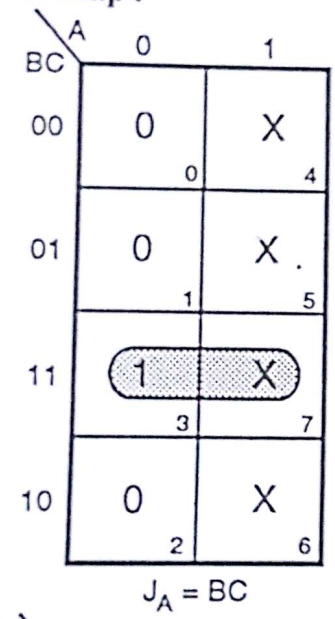
**(2) No. of FFs required will be 3.** $\therefore N = 2^m = 2^3 = 8.$

Scanned by CamScanner

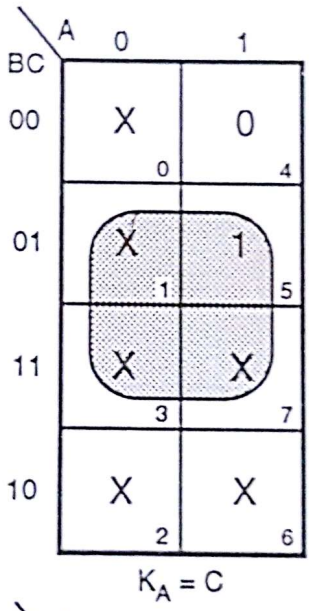
(3) **Truth Table :**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Invalid states | | |
| A | 0 → | 0 → | 0 → | 0 → | 1 → | 1 | 1 | 1 | |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| C | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| $J_A$ | 0 | 0 | 0 | 1 | X | X | X | X | |
| $K_A$ | X | X | X | X | 0 | 1 | X | X | |
| $J_B$ | 0 | 1 | X | X | 0 | 0 | X | X | |
| $K_B$ | X | X | 0 | 1 | X | X | X | X | |
| $J_C$ | 1 | X | 1 | X | 1 | X | X | X | $\rightarrow V_{CC}$ |
| $K_C$ | X | 1 | X | 1 | X | 1 | X | X | $\rightarrow V_{CC}$ |

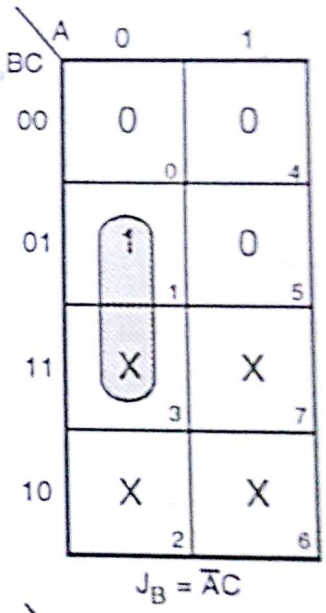
(4) **K-Map :**



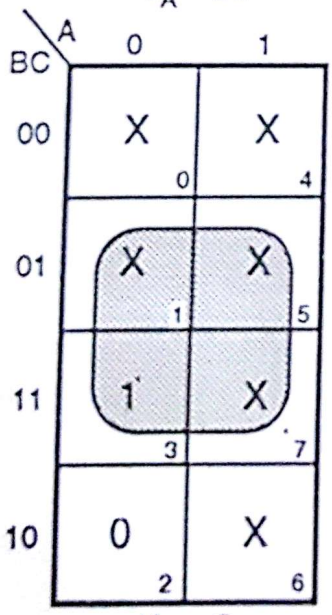

$J_A = BC$



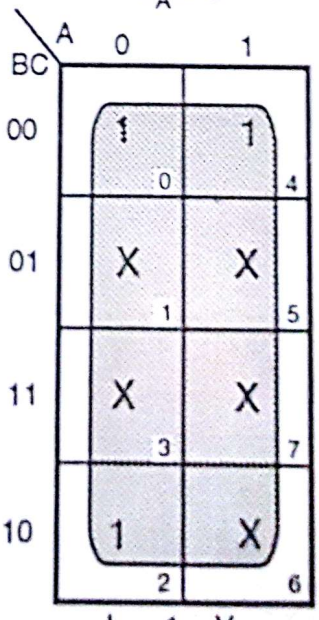

$K_A = C$



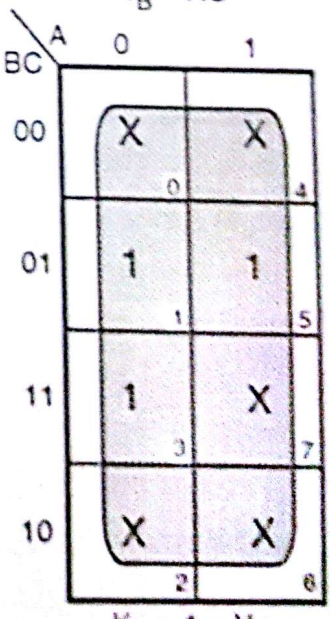

$J_B = \bar{A}C$




$K_B = C$




$J_C = 1 = V_{CC}$




$K_C = 1 = V_{CC}$

Scanned by CamScanner

(5) **Circuit diagram :**

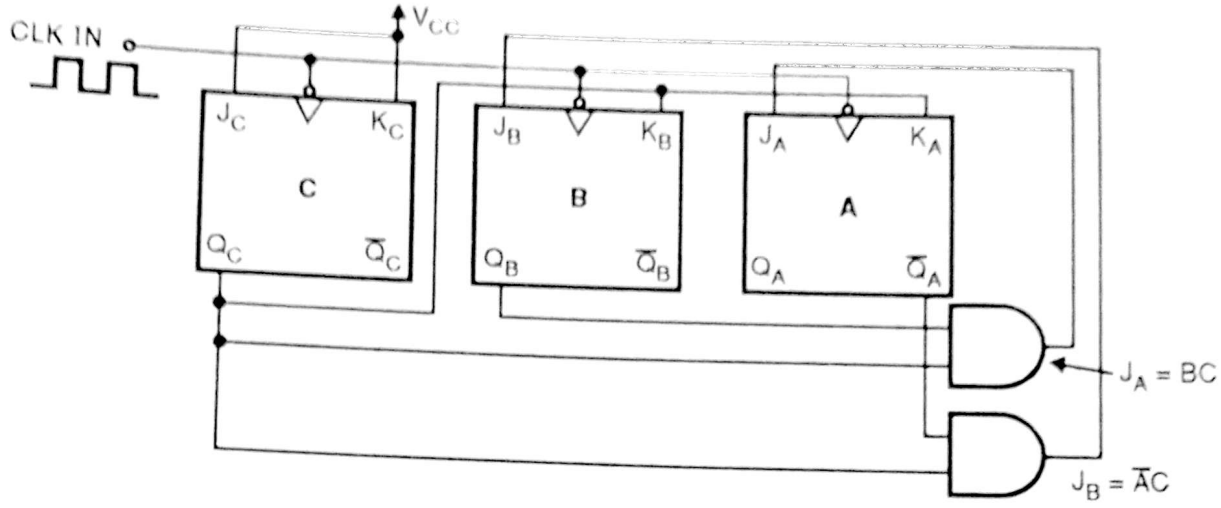



**Fig. Ex. 9.13(b)**

**Waveforms :**

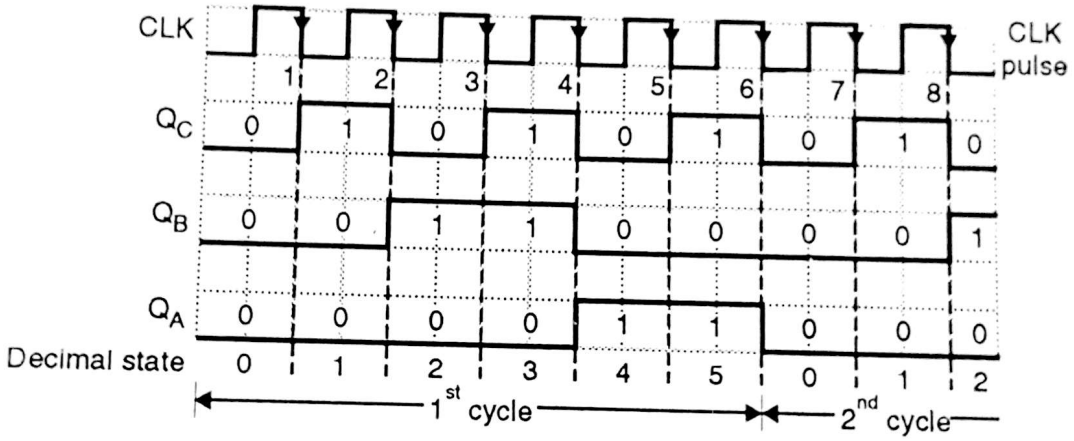



**Fig. Ex. 9.13(c)**

**Ex. 14 :** Design sync counter using M/S JK FF to count sequence in four bit XS–3.

**Soln. :** (1) In XS–3 valid codes are from decimal 3 to 12.

(2) Invalid states are 0, 1, 2, 13, 14, 15.

(3) No. of bits are 4. ∴ FFs are required.

$N = 2^m = 2^4 = 16$

(4) State diagram (shown in binary).

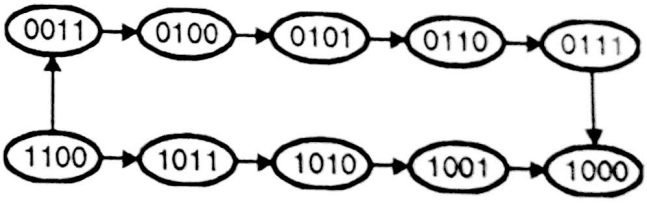


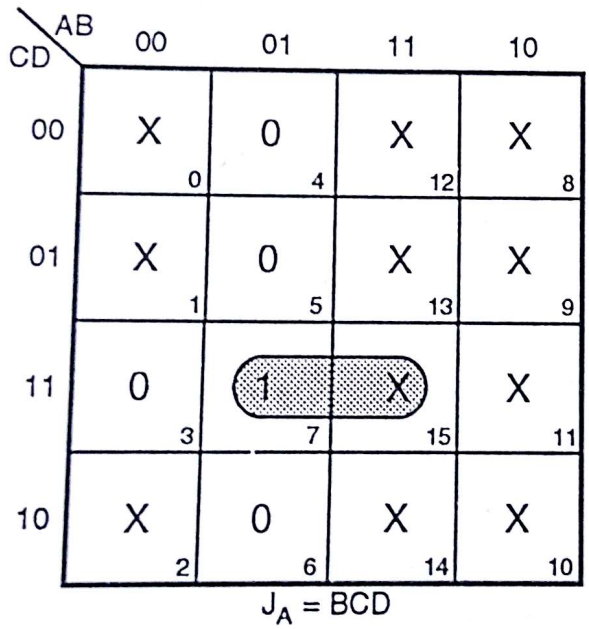

**Fig. Ex. 9.14(a)**

**(5) Truth table :**

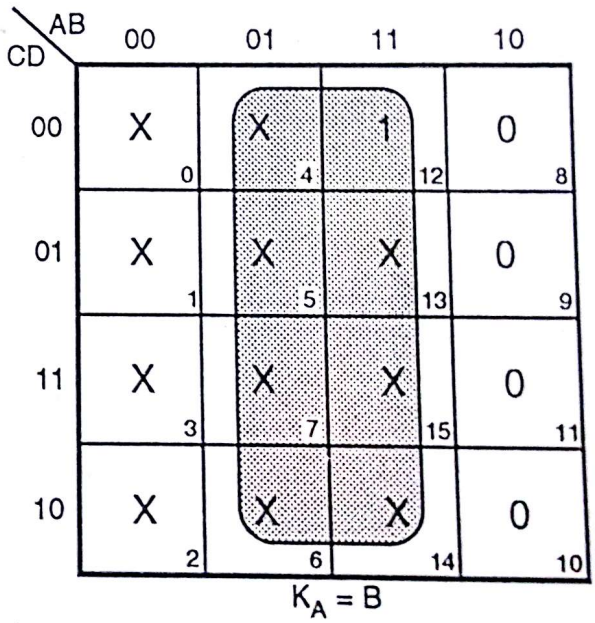
Scanned by CamScanner

| Decimal Count | Count Sequence | | | | Inputs of FFs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | $J_A$ | $K_A$ | $J_B$ | $K_B$ | $J_C$ | $K_C$ | $J_D$ | $K_D$ |
| 0 | 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | X |
| 1 | 0 | 0 | 0 | 1 | X | X | X | X | X | X | X | X |
| 2 | 0 | 0 | 1 | 0 | X | X | X | X | X | X | X | X |
| 3 | 0 | 0 | 1 | 1 | 0 | X | 1 | X | X | 1 | X | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | X | X | 0 | 0 | X | 1 | X |
| 5 | 0 | 1 | 0 | 1 | 0 | X | X | 0 | 1 | X | X | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | X | X | 0 | X | 0 | 1 | X |
| 7 | 0 | 1 | 1 | 1 | 1 | X | X | 1 | X | 1 | X | 1 |
| 8 | 1 | 0 | 0 | 0 | X | 0 | 0 | X | 0 | X | 1 | X |
| 9 | 1 | 0 | 0 | 1 | X | 0 | 0 | X | 1 | X | X | 1 |
| 10 | 1 | 0 | 1 | 0 | X | 0 | 0 | X | X | 0 | 1 | X |
| 11 | 1 | 0 | 1 | 1 | X | 0 | 1 | X | X | 1 | X | 1 |
| 12 | 1 | 1 | 0 | 0 | X | 1 | X | 1 | 1 | X | 1 | X |
| 13 | 1 | 1 | 0 | 1 | X | X | X | X | X | X | X | X |
| 14 | 1 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 15 | 1 | 1 | 1 | 1 | X | X | X | X | X | X | X | X |

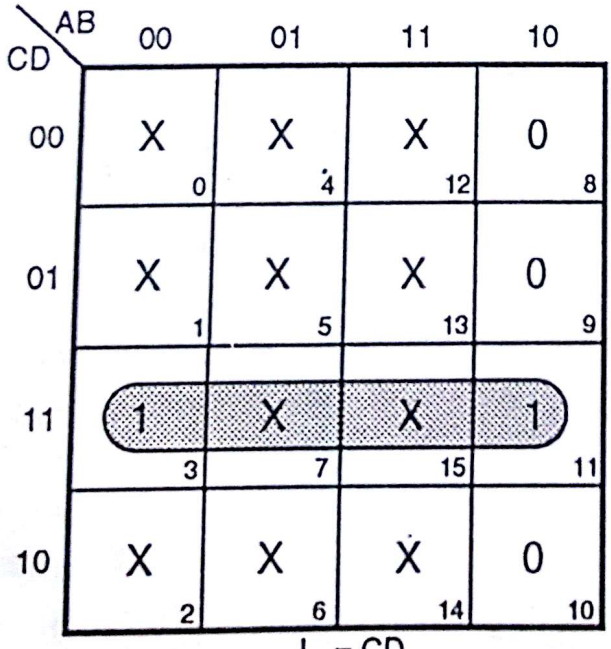
**K-Map :**



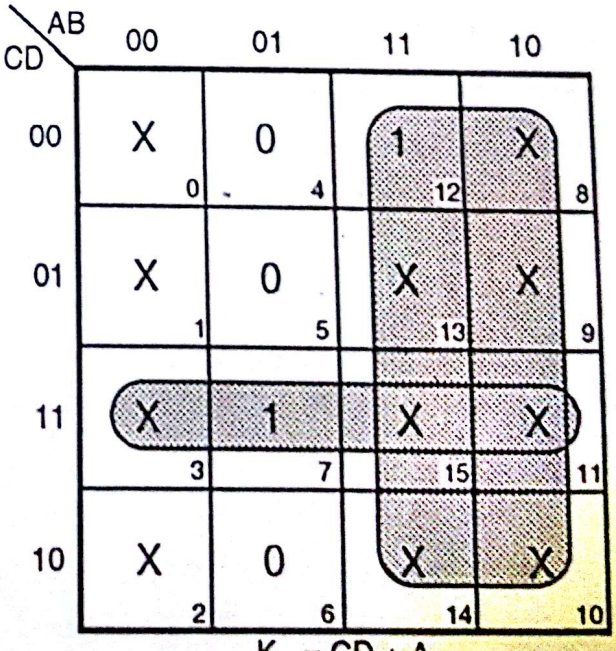

$J_A = BCD$




$K_A = B$

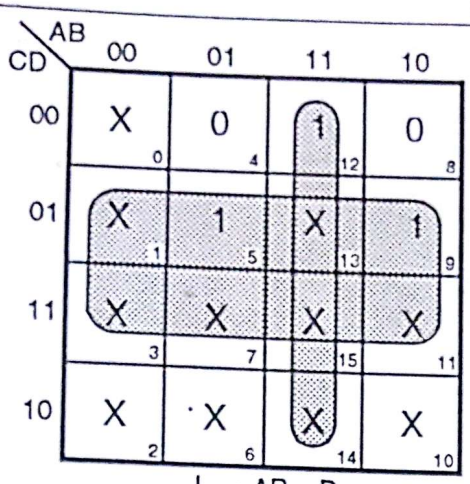



$J_B = CD$

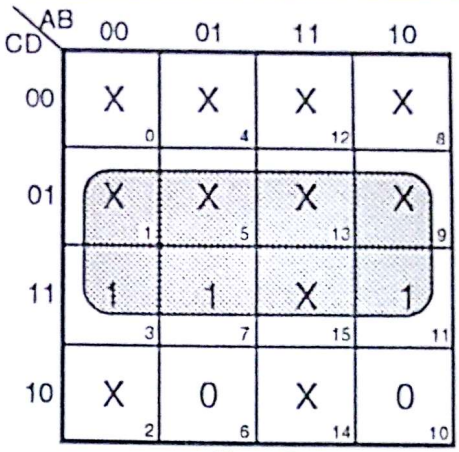



$K_B = CD + A$

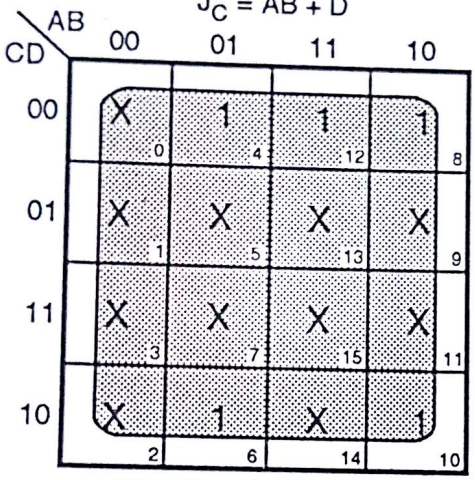
Scanned by CamScanner



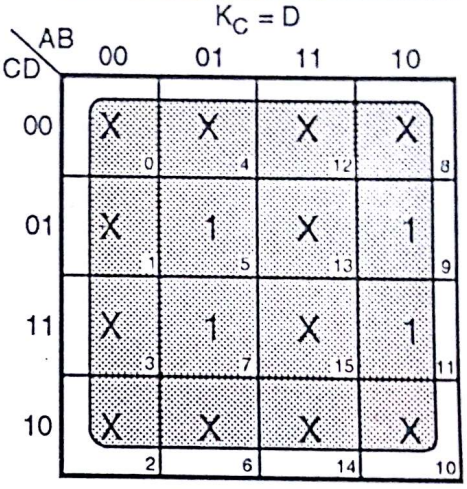

$J_C = AB + D$




$K_C = D$




$J_D = 1 = V_{CC}$




$K_D = 1 = V_{CC}$

**(6) Circuit diagram :**

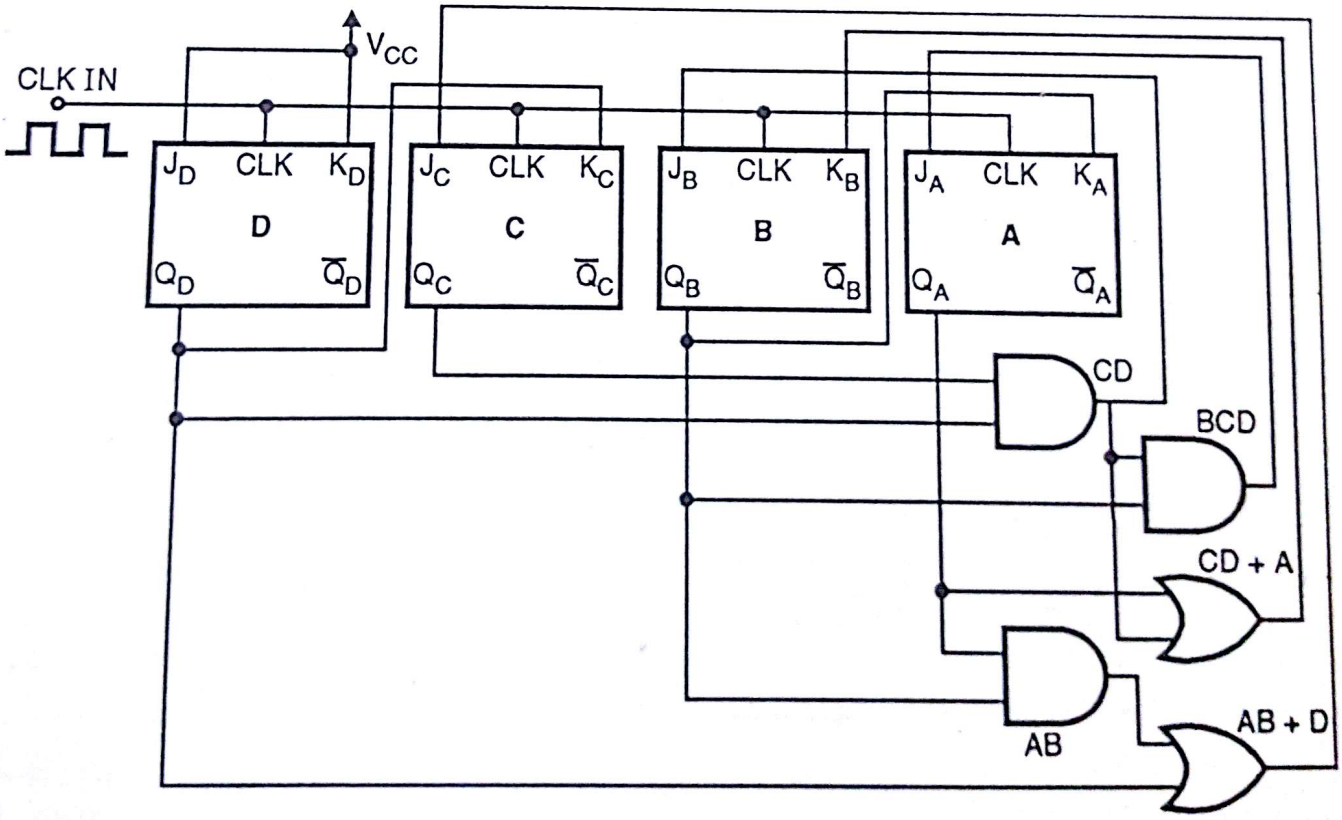



**Fig. Ex. 9.14(b)**

Scanned by CamScanner

**Ex. 15 :** Implement synchronous counter using JK FF. for state diagram shown in Fig. Ex. 9.15.

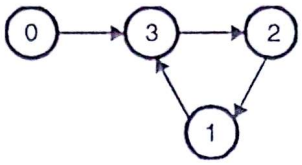



**Fig. Ex. 9.15**

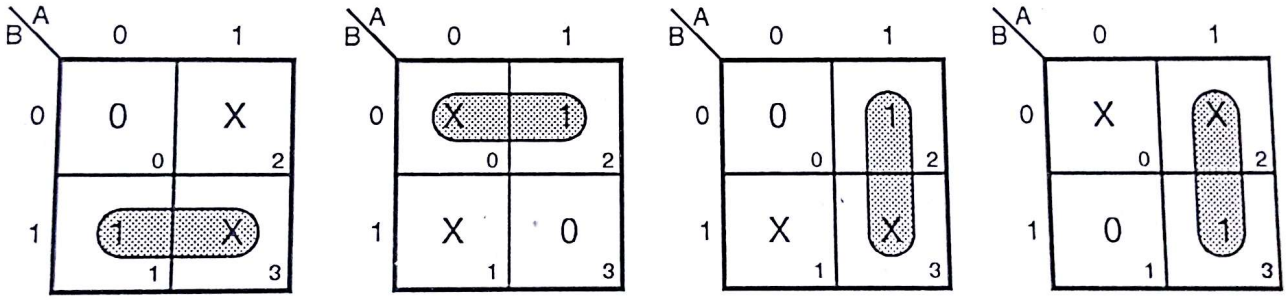
**Soln. :** (1) Counter with busing. (2) No. of FFs required will be 2.

**(3) Truth table :**

| | | | | |
|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 1 |
| $J_A$ | 0 | 1 | X | X |
| $K_A$ | X | X | 1 | 0 |
| $J_B$ | 0 | X | 1 | X |
| $K_B$ | X | 0 | X | 1 |

(2-1, 3-2, 1-3, 0-3)

**(4) K-Map :**




**(5) Circuit diagram :**

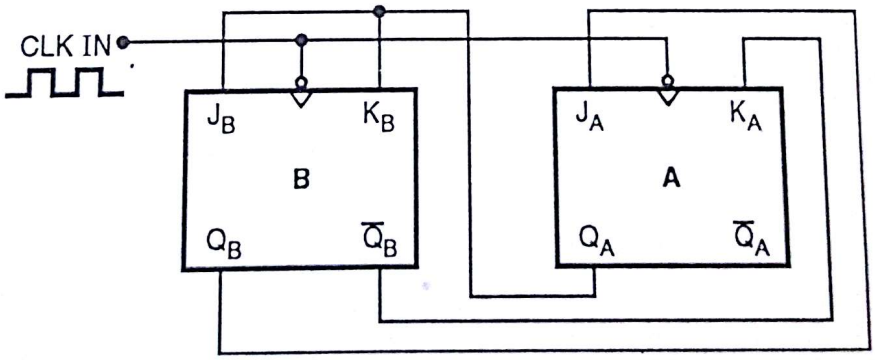



**Fig. Ex. 9.15(a)**

**Ex. 16 :** Design a decade counter using 4 FFs.

**Soln. :** Counter is decade counter or Modulo 10 counter. ∴ states are from 0 to 9.

Scanned by CamScanner

(1) State diagram.

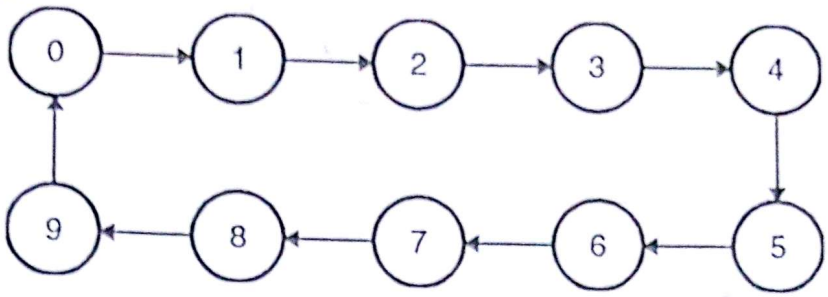



Fig. Ex. 9.16(a)

(2) No. of FFs are 4, $\therefore N = 2^m = 2^4 = 16$ states.

$\therefore$ states 10 to 15 are invalid states.

(3) Let's use M/S JK FF.

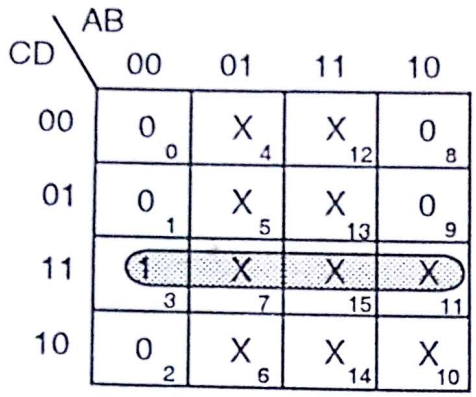
(4) Truth table.

Uptill now we have drawn truth table horizontally. For more states one can prefer vertical truth table also.

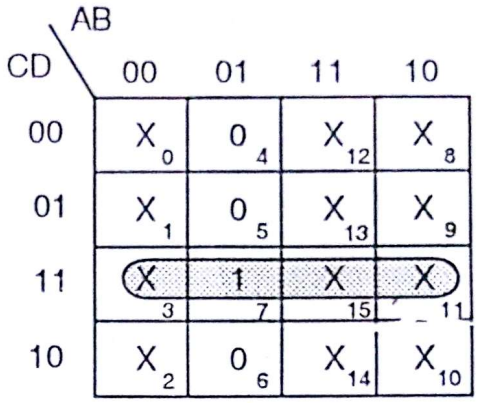
| Decimal count | Count sequence | | | | FF inputs required | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | $J_A$ | $K_A$ | $J_B$ | $K_B$ | $J_C$ | $K_C$ | $J_D$ | $K_D$ |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X | 0 | X | 1 | X |
| 1 | 0 | 0 | 0 | 1 | 0 | X | 0 | X | 1 | X | X | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | X | 0 | X | X | 0 | 1 | X |
| 3 | 0 | 0 | 1 | 1 | 0 | X | 1 | X | X | 1 | X | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | X | X | 0 | 0 | X | 1 | X |
| 5 | 0 | 1 | 0 | 1 | 0 | X | X | 0 | 1 | X | X | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | X | X | 0 | X | 0 | 1 | X |
| 7 | 0 | 1 | 1 | 1 | 1 | X | X | 1 | X | 1 | X | 1 |
| 8 | 1 | 0 | 0 | 0 | X | 0 | 0 | X | 0 | X | 1 | X |
| 9 | 1 | 0 | 0 | 1 | X | 1 | 0 | X | 0 | X | X | 1 |
| 10 | 1 | 0 | 1 | 0 | X | X | X | X | X | X | X | X |
| 11 | 1 | 0 | 1 | 1 | X | X | X | X | X | X | X | X |
| 12 | 1 | 1 | 0 | 0 | X | X | X | X | X | X | X | X |
| 13 | 1 | 1 | 0 | 1 | X | X | X | X | X | X | X | X |
| 14 | 1 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 15 | 1 | 1 | 1 | 1 | X | X | X | X | X | X | X | X |

Scanned by CamScanner

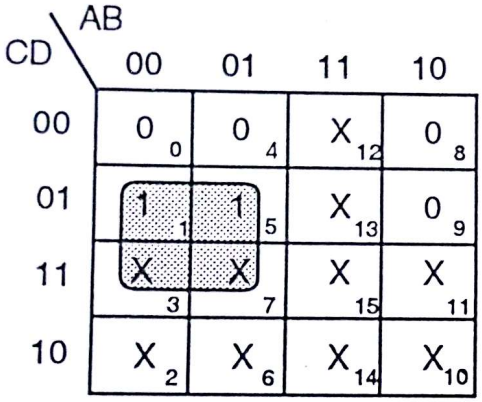
(5) **K-map → chart :**



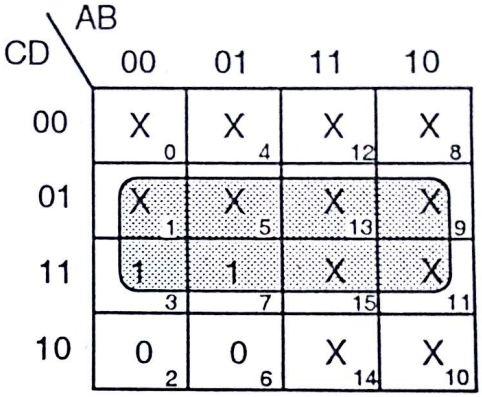

$J_B = CD$



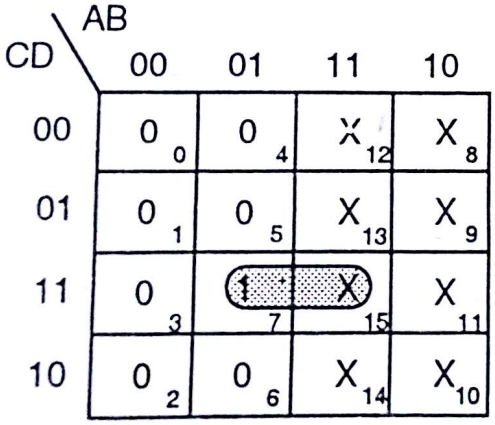

$K_B = CD$



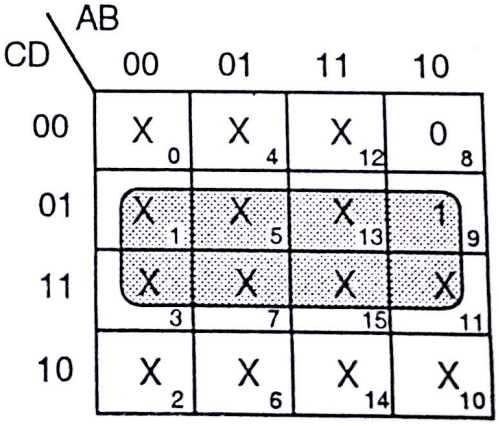

$J_C = \bar{A}D$



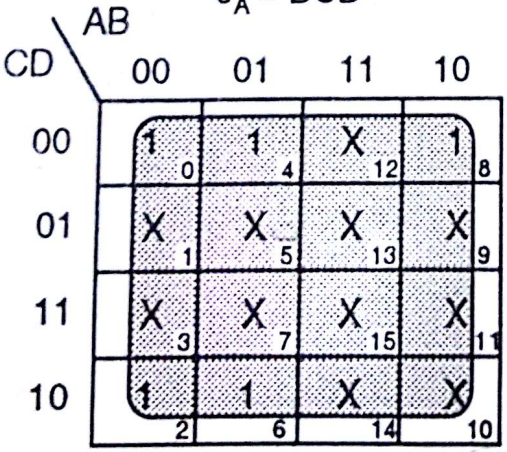

$K_C = D$



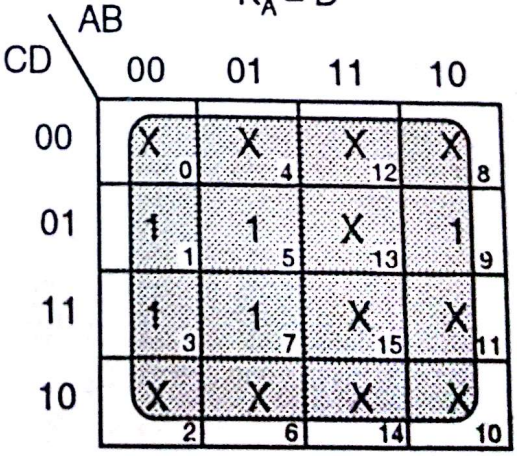

$J_A = BCD$




$K_A = D$




$J_D = 1 = V_{CC}$




$K_D = 1 = V_{CC}$

Scanned by CamScanner

(6) **Circuit diagram :**

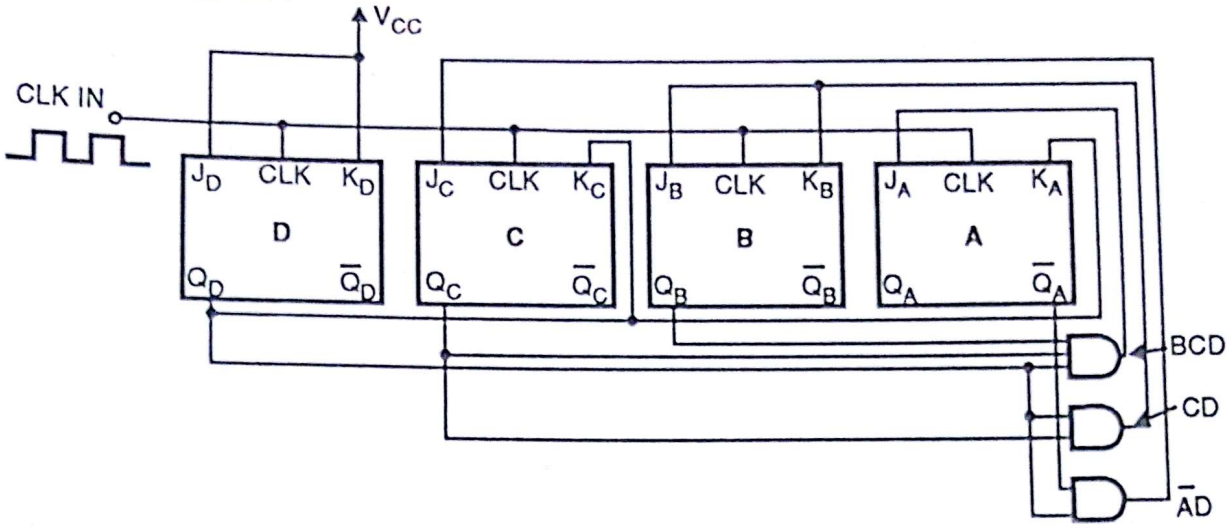



**Fig. Ex. 9.16(b)**

**Ex. 17 :** By using JK FF, design counter to go through states.

0 – 1 – 3 – 5 – 7 – 9 – 11 – 13 – 15 – 0.

**Soln. : (1) State diagram :**

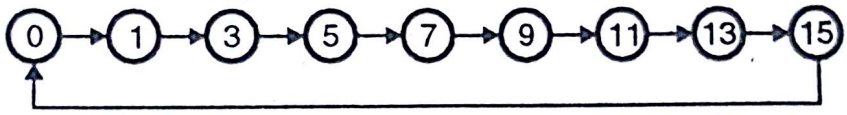



**Fig. Ex. 9.17(a)**

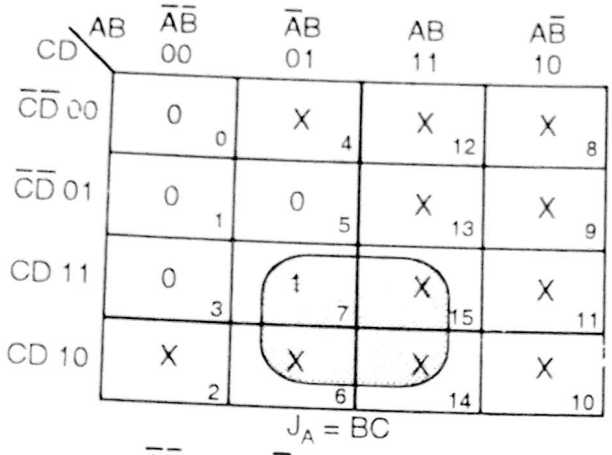
(2) States are $15 \leq 2^m$

$\therefore$ $m = 4$ $\therefore$ $N = 2^m = 2^4 = 16$ $\therefore$ 0 to 15

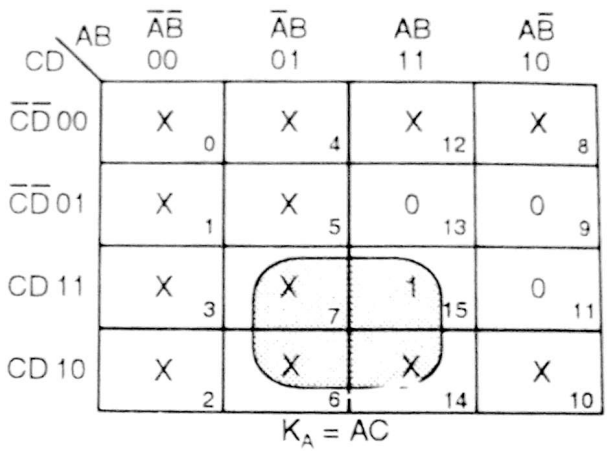
(3) Truth table :

| Decimal count | Count sequence | | | | Inputs to FF | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | $J_A$ | $K_A$ | $J_B$ | $K_B$ | $J_C$ | $K_C$ | $J_D$ | $K_D$ |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X | 0 | X | 1 | X |
| 1 | 0 | 0 | 0 | 1 | 0 | X | 0 | X | 1 | X | X | 0 |
| 2 | 0 | 0 | 1 | 0 | X | X | X | X | X | X | X | X |
| 3 | 0 | 0 | 1 | 1 | 0 | X | 1 | X | X | 1 | X | 0 |
| 4 | 0 | 1 | 0 | 0 | X | X | X | X | X | X | X | X |
| 5 | 0 | 1 | 0 | 1 | 0 | X | X | 0 | 1 | X | X | 0 |
| 6 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 7 | 0 | 1 | 1 | 1 | 1 | X | X | 1 | X | 1 | X | 0 |
| 8 | 1 | 0 | 0 | 0 | X | X | X | X | X | X | X | X |
| 9 | 1 | 0 | 0 | 1 | X | 0 | 0 | X | 1 | X | X | 0 |
| 10 | 1 | 0 | 1 | 0 | X | X | X | X | X | X | X | X |
| 11 | 1 | 0 | 1 | 1 | X | 0 | 1 | X | X | 1 | X | 0 |
| 12 | 1 | 1 | 0 | 0 | X | X | X | X | X | X | X | X |
| 13 | 1 | 1 | 0 | 1 | X | 0 | X | 0 | 1 | X | X | 0 |
| 14 | 1 | 1 | 1 | 0 | X | X | X | X | X | X | X | X |
| 15 | 1 | 1 | 1 | 1 | X | 1 | X | 1 | X | 1 | X | 1 |
| | | | | | | | | | | 1 | | 1 |
| | | | | | | | | | | $V_{CC}$ | | $V_{CC}$ |

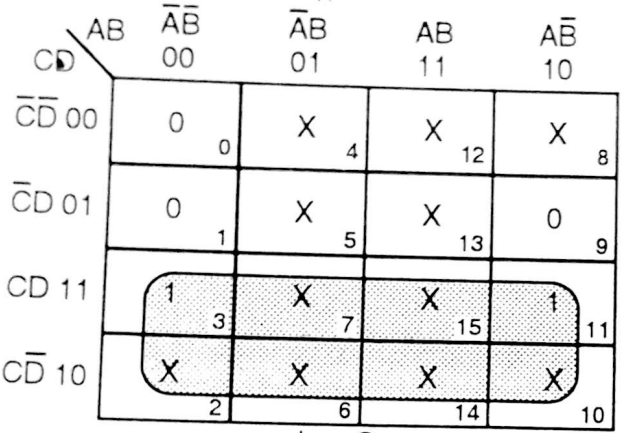
Scanned by CamScanner



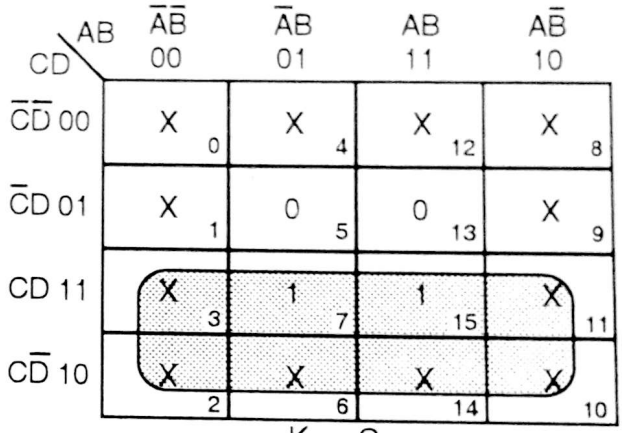

$J_A = BC$



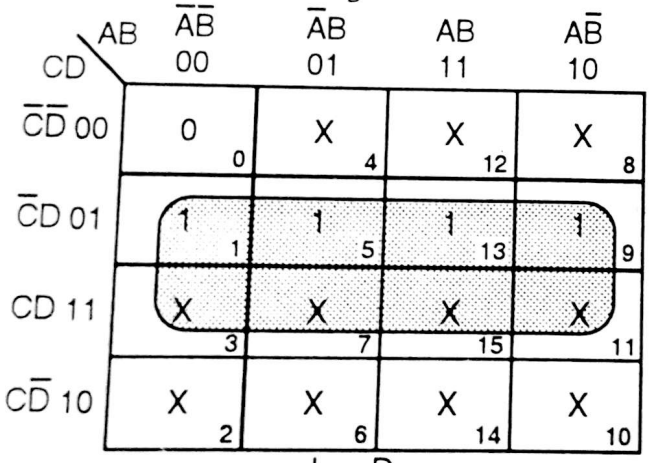

$K_A = AC$



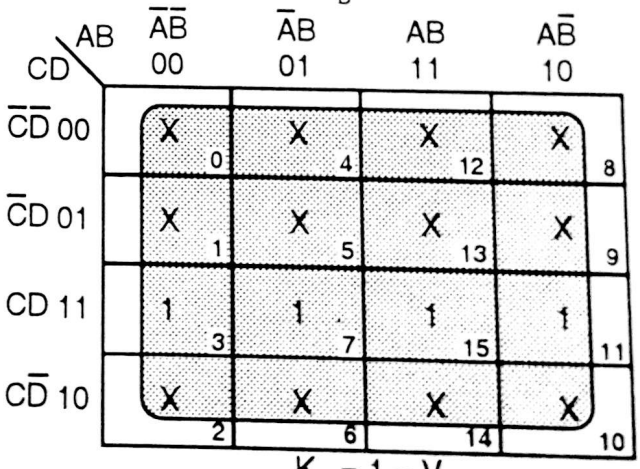

$J_B = C$



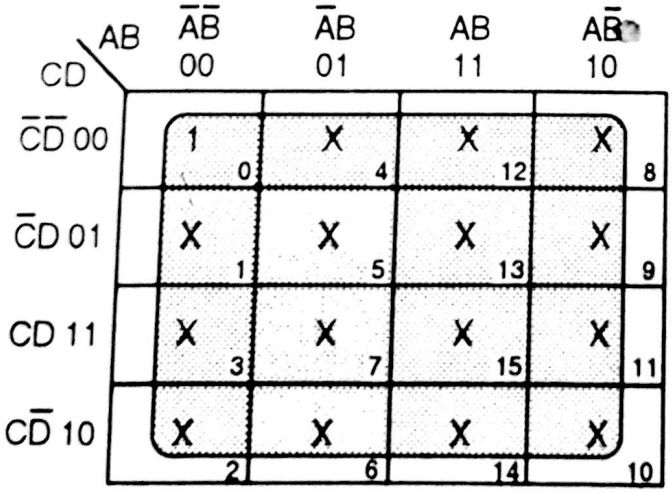

$K_B = C$



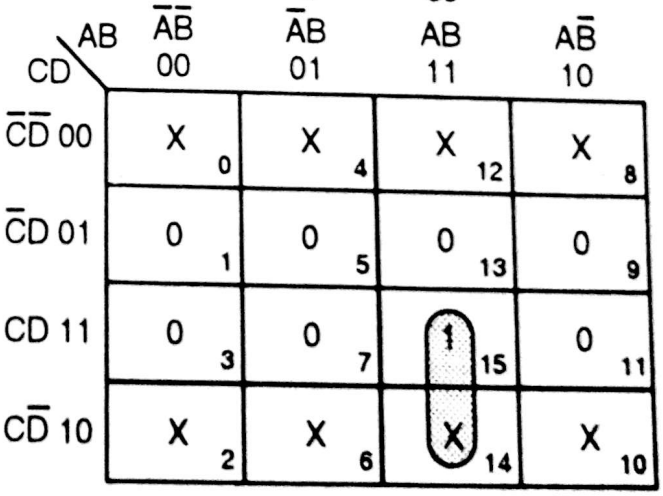

$J_C = D$




$K_C = 1 = V_{CC}$




$J_D = 1 = V_{CC}$




$K_D = ABC$

Scanned by CamScanner

**Circuit diagram :**

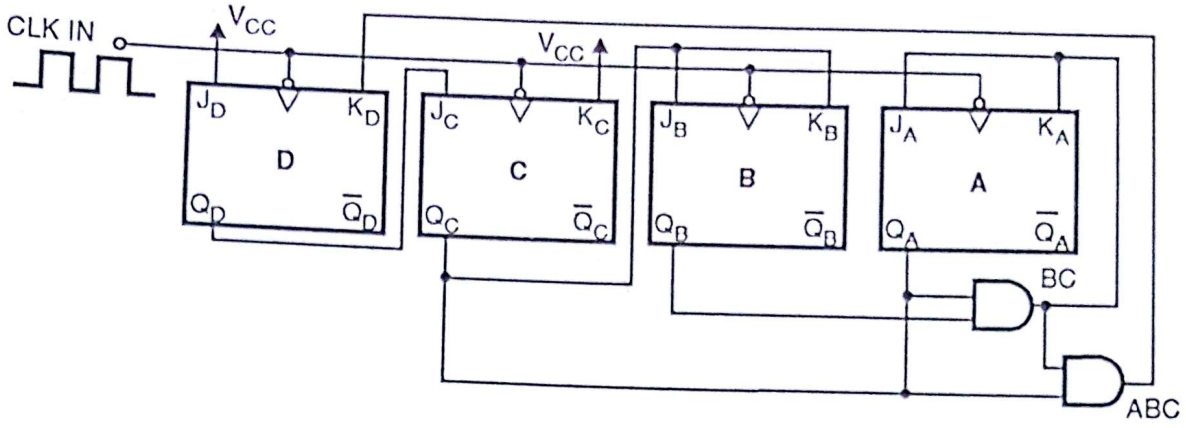



**Fig. Ex. 9.17(b)**

## 9.9 Controlled Counter :

In some of the applications, we may require one or more count sequences. Question is that how to switch over to other sequence, for the standard hardware. Let's say we want sequence.

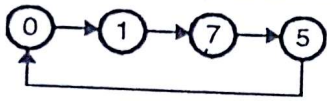



**Fig. 9.15**

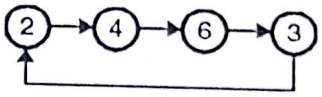



**Fig. 9.16**

These has to be smooth switch over from one sequence to other. Here we are going to use CONTROL I/P.

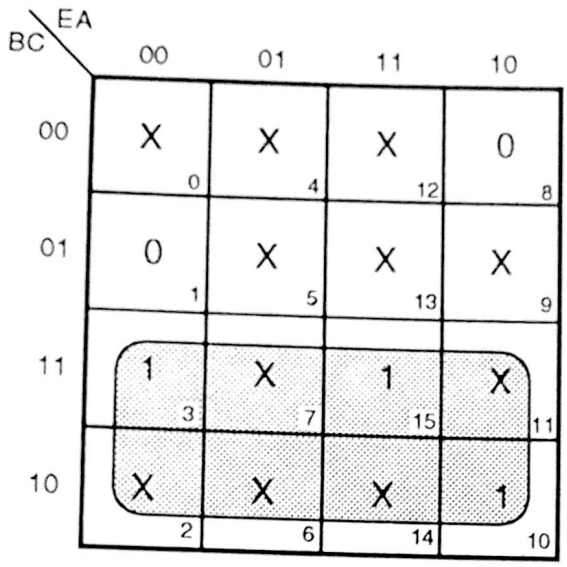
**Ex. 18 :** Design 3 bit counter with input control signal E. The counter will undergo the steps. 0 – 2 – 4 – 6 – 0 if control signal e is held high. The counter will undergo steps 1 – 3 – 5 – 7 – 1 if control signal E is held low. **(June 99, 16 Marks)**

**Soln. :**

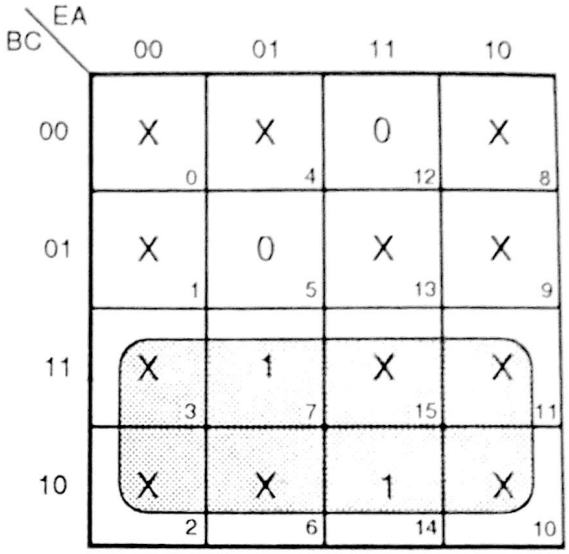
| Control E MSB | A | B | C LSB | $J_A$ | $K_A$ | $J_B$ | $K_B$ | $J_C$ | $K_C$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | X | X | X | X | X | X | |
| 0 | 0 | 0 | 1 | 0 | X | 1 | X | X | 0 | |
| 0 | 0 | 1 | 0 | X | X | X | X | X | X | |
| 0 | 0 | 1 | 1 | 1 | X | X | 1 | X | 0 | Table for sequence 1, 3, 5, 7, 1 E = 0 E should be taken MSB |
| 0 | 1 | 0 | 0 | X | X | X | X | X | X | |
| 0 | 1 | 0 | 1 | X | 0 | 1 | X | X | 0 | |
| 0 | 1 | 1 | 0 | X | X | X | X | X | X | |
| 0 | 1 | 1 | 1 | X | 1 | X | 1 | X | 0 | |
| 1 | 0 | 0 | 0 | 0 | X | 1 | X | 0 | X | |
| 1 | 0 | 0 | 1 | X | X | X | X | X | X | |
| 1 | 0 | 1 | 0 | 1 | X | X | 1 | 0 | X | |
| 1 | 0 | 1 | 1 | X | X | X | X | X | X | Count sequence 0, 2, 4, 6, 0 (concentrate on ABC only) E = 1 |
| 1 | 1 | 0 | 0 | X | 0 | 1 | X | 0 | X | |
| 1 | 1 | 0 | 1 | X | X | X | X | X | X | |
| 1 | 1 | 1 | 0 | X | 1 | X | 1 | 0 | X | |
| 1 | 1 | 1 | 1 | X | X | X | X | X | X | |

Scanned by CamScanner

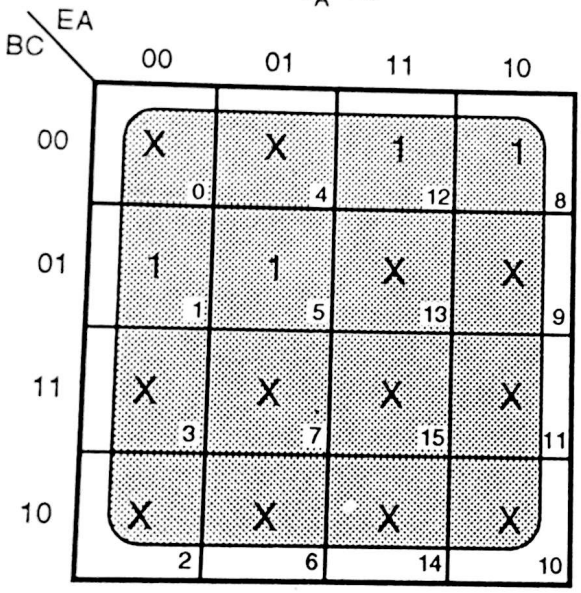
**K-Map :**



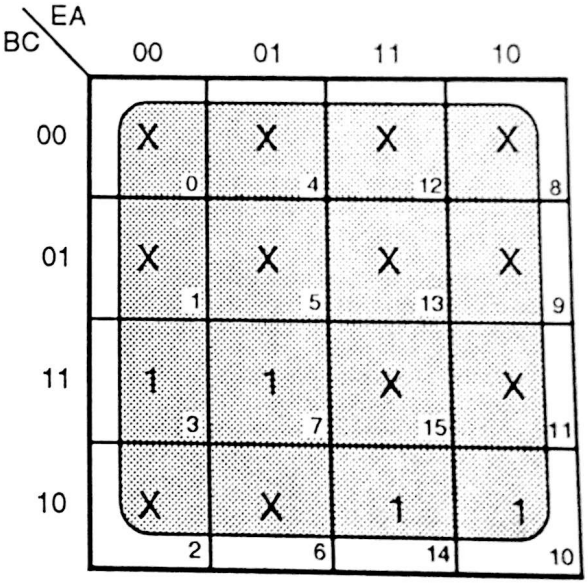

$J_A = B$



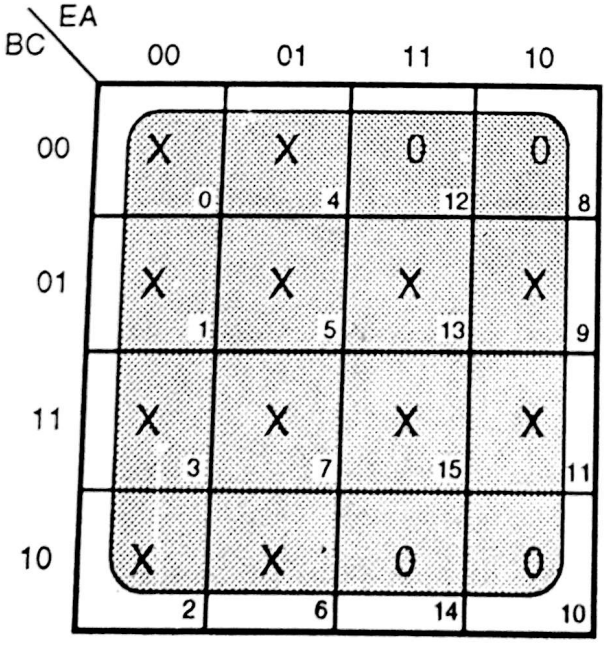

$K_A = B$



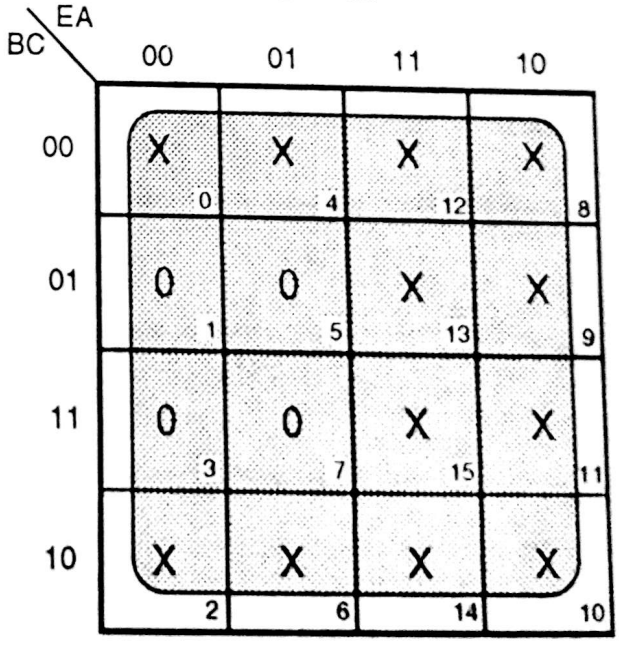

$J_B = V_{CC} = 1$




$K_B = V_{CC} = 1$




$J_C = GND = 0$




$K_C = GND = 0$

Scanned by CamScanner

(1) No. of FFs required will be $7 \leq 2^m$ $\therefore m = 3$

(2) Sequence 1 is 0, 2, 4, 6, 0 ; 2 is 1, 3, 5, 7, 1

(3) While making truth table take into account E (control signal).

From equation we see that master control line E is not coming into picture. So have a look to sequence.

| E = 0 | | | E = 1 | | |
|---|---|---|---|---|---|
| A | B | C | A | B | C |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | .0 | 1 |

C = 0 permanently as E = 0

C = 1 permanently as E = 1

$\therefore$ Connect E to preset and clear of last FF. (C) FF.

**Circuit diagram :**

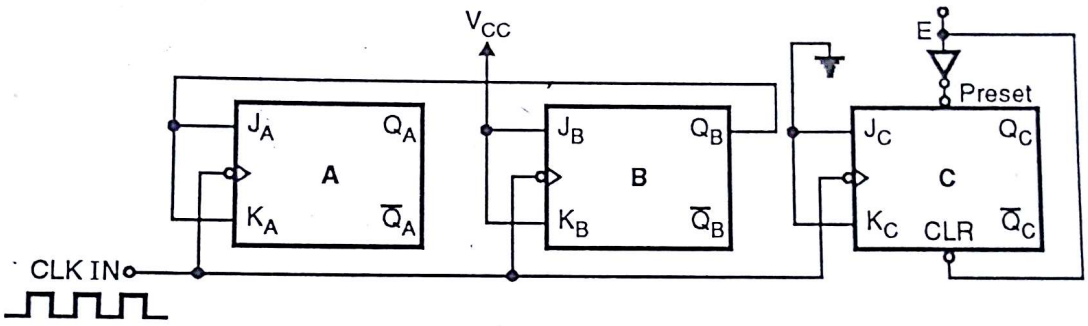



**Fig. Ex. 9.18(a)**

## 9.10 Presettable Counter :

As we have discussed previously that sometime counter will not start from '0' state. It will start from some predefined state. Let's say we want to down count from decimal 10. So *preset* counter to start 10, apply CLK pulses, so it will count down to 9, 8, 7, ... to 0. After that it will again start from 10.

Counters can be preset to any desired starting count either synchronously or asynchronously. This can be done using ***PRESET*** and ***CLEAR*** terminals of FF.

Fig. 9.17 shows 3 bit parallel presettable up counter.

(1) Counter is synchronous

(2) Asynchronous inputs, preset and clear, are controlled through *load* signal.

(3) Steps for 'Preset',

(a) Apply required count (step) to $P_2$, $P_1$, $P_0$. Let's say $P_2$, $P_1P_0$ = 101

(b) Apply 'LOW' going signal to 'LOAD' pin

(c) $\therefore$ output of NAND1, NAND3 = 0 and NAND2 = 1 at the same time NAND4, NAND6 = 1 and NAND5 = 0.

$\therefore$ FF B will be reset and AC will be preset

: ABC = 101.

(d) CLK is not going to affect above action.

Scanned by CamScanner

(4) After presetting apply CLK pulse so that counter can start next count.

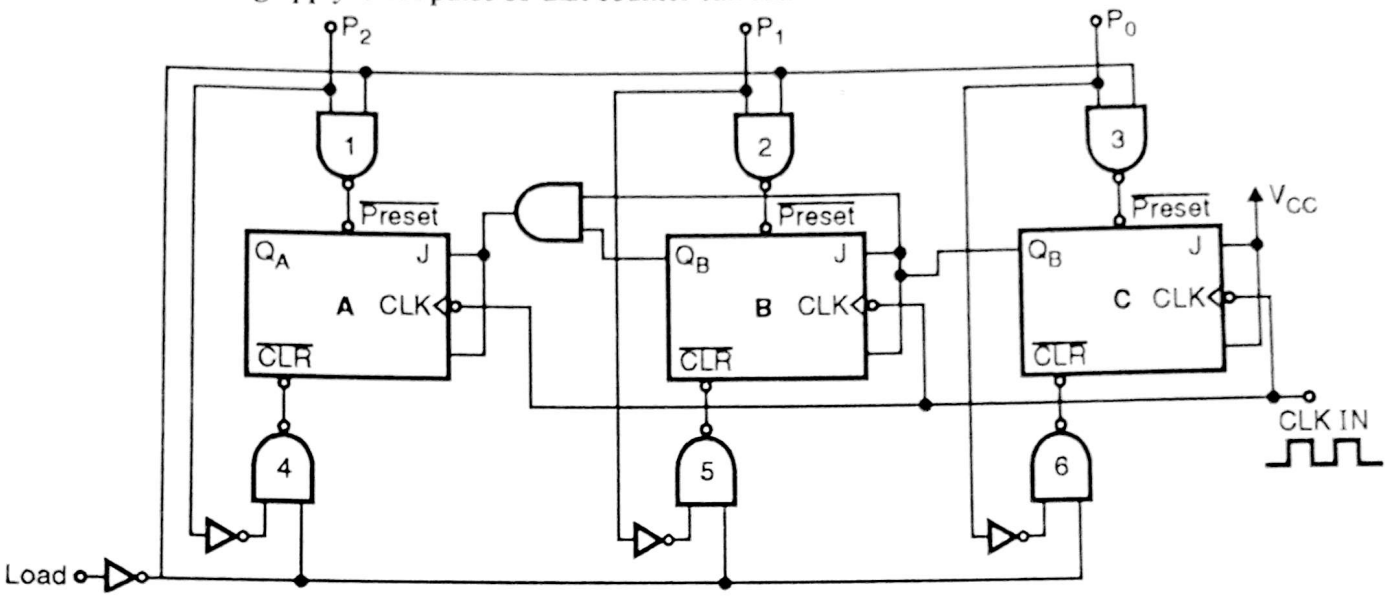



**Fig. 9.17**

## 9.11 IC 7490 :

IC 7490 is TTl MSI ***decade counter***. It contains four M/S FF and additional gating to provide divide by 2 counter (Mod–2) and three stage binary counter which provides divide by 5 counter (Mod–5).

Fig. 9.18 shows block schematic of IC 7490.

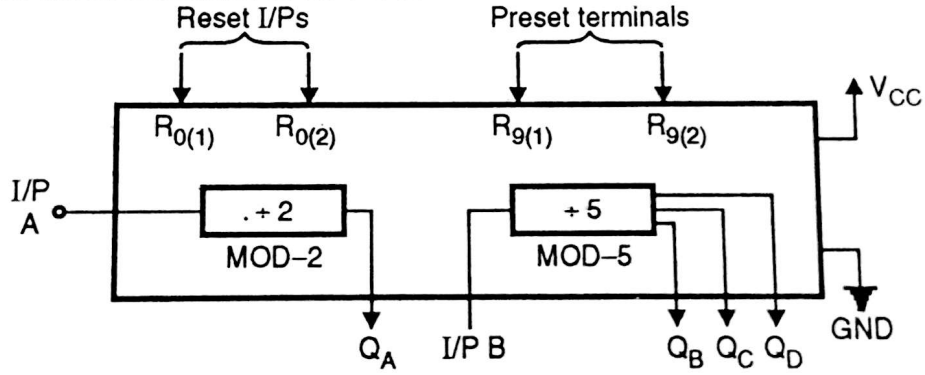



**Fig. 9.18**

7490 have gated zero reset ($R_{0(1)}$, $R_{0(2)}$) and gated set to nine inputs $R_{9(1)}$, $R_{9(2)}$ for use in BCD nine's complement applications.

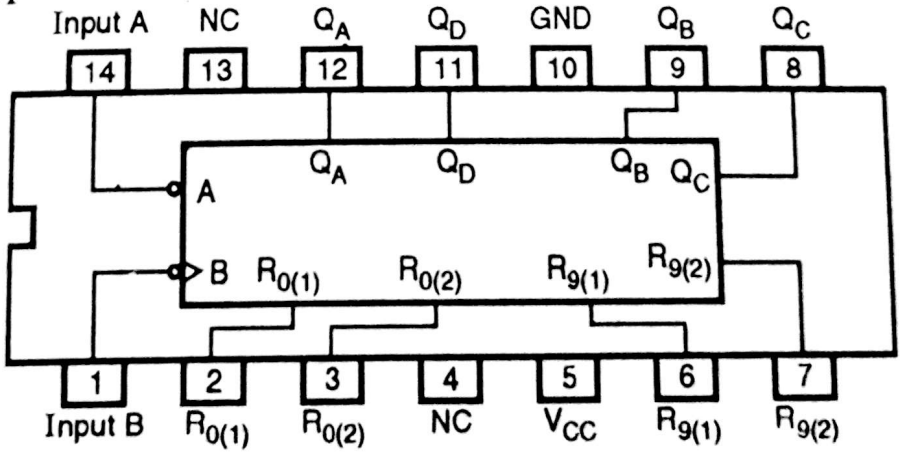



**Fig. 9.19 (a) : Pin configuration**

Scanned by CamScanner

Reset | count Truth table.

| Reset Inputs | | | | Output | | | | |
|---|---|---|---|---|---|---|---|---|
| $R_{0(1)}$ | $R_{0(2)}$ | $R_{9(1)}$ | $R_{9(2)}$ | $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ | |
| 1 | 1 | 0 | X | 0 | 0 | 0 | 0 | |
| 1 | 1 | X | 0 | 0 | 0 | 0 | 0 | → Reset to 0000 |
| X | X | 1 | 1 | 1 | 0 | 0 | 1 | → Set to 1001 |
| X | 0 | X | 0 | COUNTER | | | | |
| 0 | X | 0 | X | COUNTER | | | | If CLK applied it will count or step from previous to next step |
| 0 | X | X | 0 | COUNTER | | | | |
| X | 0 | 0 | X | COUNTER | | | | |

**Fig. 9.19(b)**

Fig. 9.19(b) Shows truth table.

As shown in truth table when $R_{0(1)} = R_{0(2)} = 1$ IC 7490 ***Reset***.

When $R_{9(1)} = R_{9(2)} =$ IC 7490 sets to 1001 (9 decimal).

So one has to provide logic 1 to these input temporarily, set/reset it, and lower the input terminals so IC 7490 can start counting as CLK pulses applied to it.

**Note :** (1) $Q_A$ O/P toggle at each ***Falling edge*** of CLK applied to Input A pin.

(2) States of mod – 5 counters are 0, 1, 2, 3, 4, 0.

**Ex. 19 :** Implement sequential decade counter using IC 7490 and write counter states.

**Soln. :** In IC 7490 mod–2 counter and mod – 5 counter is available $\therefore 2 \times 5 = 10 \therefore$ gives decade counter.

As mentioned in above example, requirement is ***sequential*** decade counter. $\therefore$ Apply CLK IN to INPUT A pin. Output $Q_A$ is connected to INPUT B i.e. input of mod – 5 counter. Output terminals are $Q_D$ $Q_C$ $Q_B$ $Q_A$, where $Q_D$ = MSB, $Q_B$ = LSB connections are shown in Fig. Ex. 9.19(a).

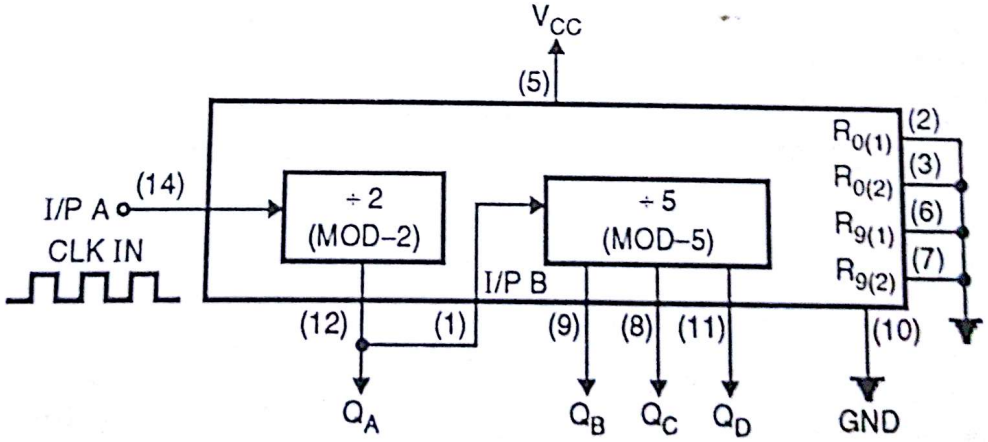



**Fig. Ex. 9.19(a)**

**Important point to be remembered is, now at each *falling edge* of $Q_A$ O/P, mod 5 counter will increment. And $Q_A$ change at falling edge of CLK input.**

Scanned by CamScanner

The count sequence can be written as,

| Count | $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ | Falling edge of $Q_A$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 |

Change

**Fig. Ex. 9.19(b)**

If you observe Fig. Ex. 9.19(b). concentrate on $Q_D$, $Q_B$, $Q_C$ and $Q_A$ seperately initially $Q_D\ Q_C\ Q_B$ = 000. When $Q_A$ changed from $1 \rightarrow Q_D\ Q_C\ Q_B$ changed to 001. For 2nd CLK period state of $Q_D\ Q_C\ Q_B$ remains unchanged.

**Ex. 20 :** Generate bi – quinary count using 7490.

In Ex. 19 CLK IN was given to mod 2 and output of mod 2 cascaded to mod 5 counter. To get bi – quinary count reverse the block i.e. give CLK IN to mod – 5 first and then output of mod – 5 ($Q_D$) cascade to mod – 2 counter. The diagram is as shown in Fig. Ex. 9.20.

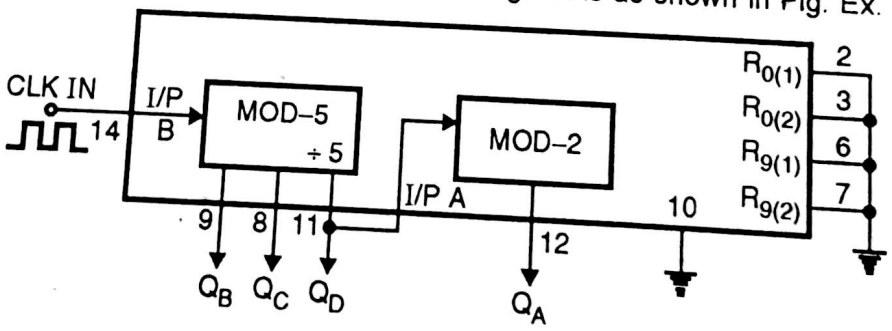



**Fig. Ex. 9.20**

**Soln. :** Remember now that HIGH to LOW transition of $Q_D$ will toggle $Q_A$ output and Also remember that mod – 5 counts from 0, 1, 2, 3, 4, 0. Binary count table.

Scanned by CamScanner

| Decimal equivalent | $Q_A$ | $Q_D$ | $Q_C$ | $Q_B$ | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Initially reset |
| 1 | 0 | 0 | 0 | 1 | |
| 2 | 0 | 0 | 1 | 0 | |
| 3 | 0 | 0 | 1 | 1 | |
| 4 | toggle 0 | transition 1 | 0 | 0 | |
| 8 | 1 | 0 | 0 | 0 | |
| 9 | 1 | 0 | 0 | 1 | |
| 10 | 1 | 0 | 1 | 0 | |
| 11 | 1 | 0 | 1 | 1 | |
| 12 | 1 | 1 | 0 | 0 | |

**Ex. 21 :** Design module 7 counter using IC 7490

**Soln. :** Modulo – 7 means states are from 0, 1, 2, 3, 4, 5, 6, 0.

∴ We have to reset counter IC 7490 when $Q_D$, $Q_C$, $Q_B$ $Q_A$ = 0111.

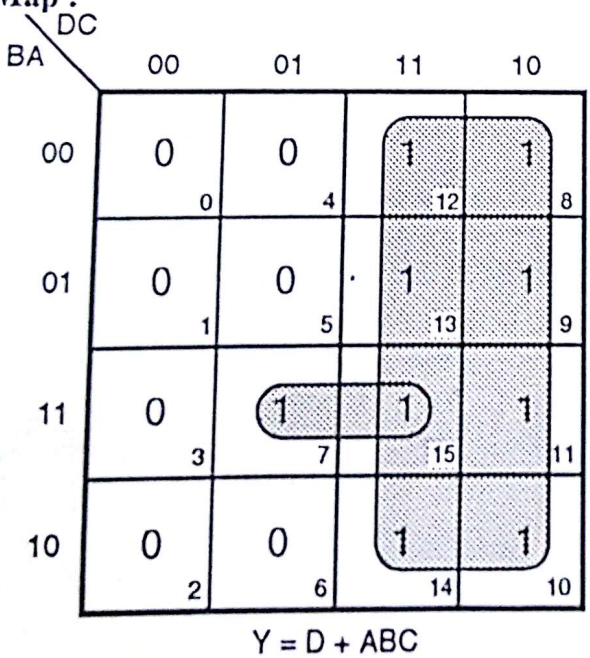
Design Reset Circuit.

The way of designing reset circuit is same as that of asynchronous counter. O/p of reset circuit should be HIGH because $R_{0(1)}$ and $R_{0(2)}$ are 'active high' inputs.

∴ output should be low for states 0 to 6.

Output should be HIGH for states 7 onwards.

**K-Map :**




Y = D + ABC

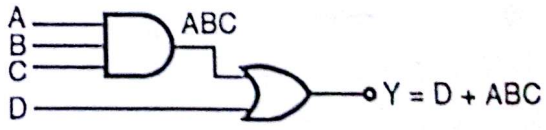



**Fig. Ex. 9.21(a)**

**Normally, we connect Y output to Reset terminal ∴ final diagram is,**

Scanned by CamScanner

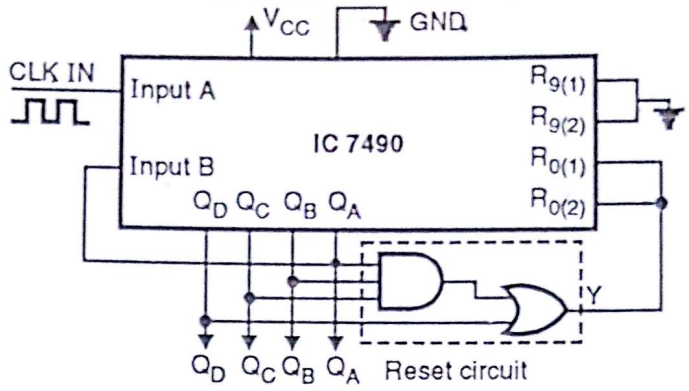



**Fig. Ex. 9.21(b) : Reset circuit**

---

**Ex. 22 :** Design divided by 100 using IC 7490.

**Soln. :**

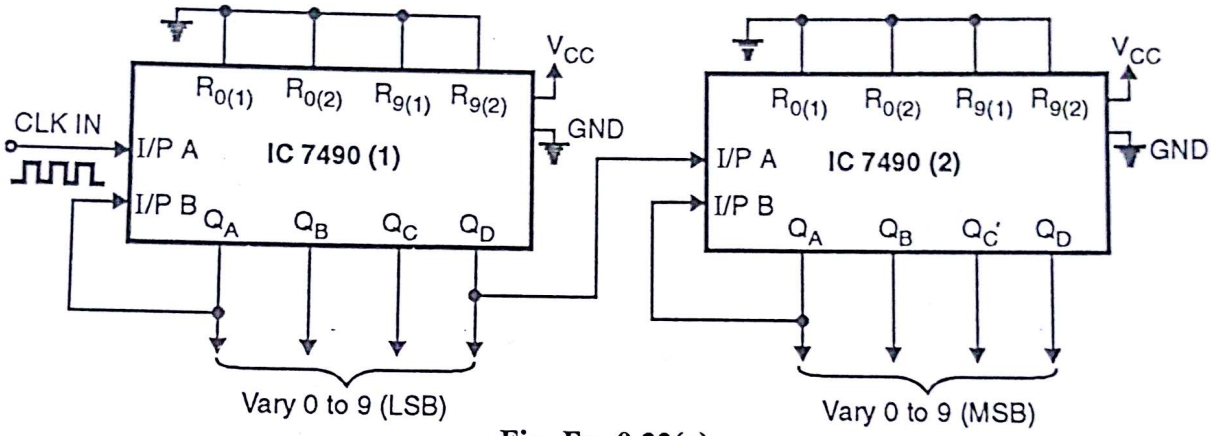



**Fig. Ex. 9.22(a)**

The problem is nothing but asks for cascade of IC 7490.

Each IC 7490 gives ÷ 10 count (Mod 10) ∴ 2 IC gives $10 \times 10 = 100$ counts.

Observe Fig. Ex. 9.22(a). IC 7490 is in standard ÷ 10 mode i.e. CLK output to Input A, $Q_A$ connected to Input B and $R_{0(1)}$, $R_{0(2)}$, $R_{9(1)}$, $R_{9(2)}$ are grounded. For cascading simply connect $Q_D$ of 1st stage (IC 7490 (1)) to Input A of next stage (IC 7490 (2)). Both IC combinely provides count from 00 to 99 decimal i.e. total 100 states. ∴ ÷ 100 counter.

---

**Ex. 23 :** Design divide by 96 counter using 7490 IC's.

**Soln. :** The problem asks for cascading of IC 7490 as well as resetting in between.

As far as cascading is concern circuit will be same as that of ..... except $R_{0(1)}$ $R_{0(2)}$ will not be grounded. It will be given to ***reset circuit***.

Normally for designing reset circuit, we draw K-Map. For 4 to 5 variable drawing K-Map is OK. But for 96 count drawing K-Map will be unnecessary complicated.

∴ Implement simple logic.

Scanned by CamScanner

Remember each count is unique count, it will occur only once thro'out full cycle of 00 to 99.

Divided by 96 means mod – 96 ∴ circuit should reset at count 96.

Write 96 in binary

9 6
↓ ↓
1001 0110

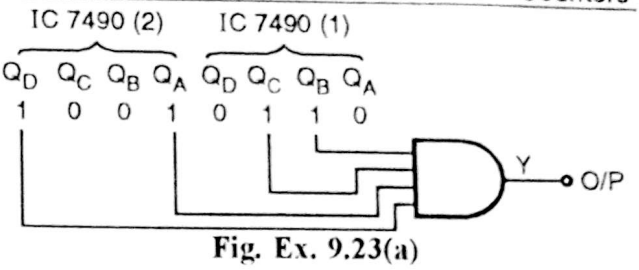



Fig. Ex. 9.23(a)

Select ones and give that to AND gate as output required is HIGH to reset IC 7490.

∴ Final circuit diagram,

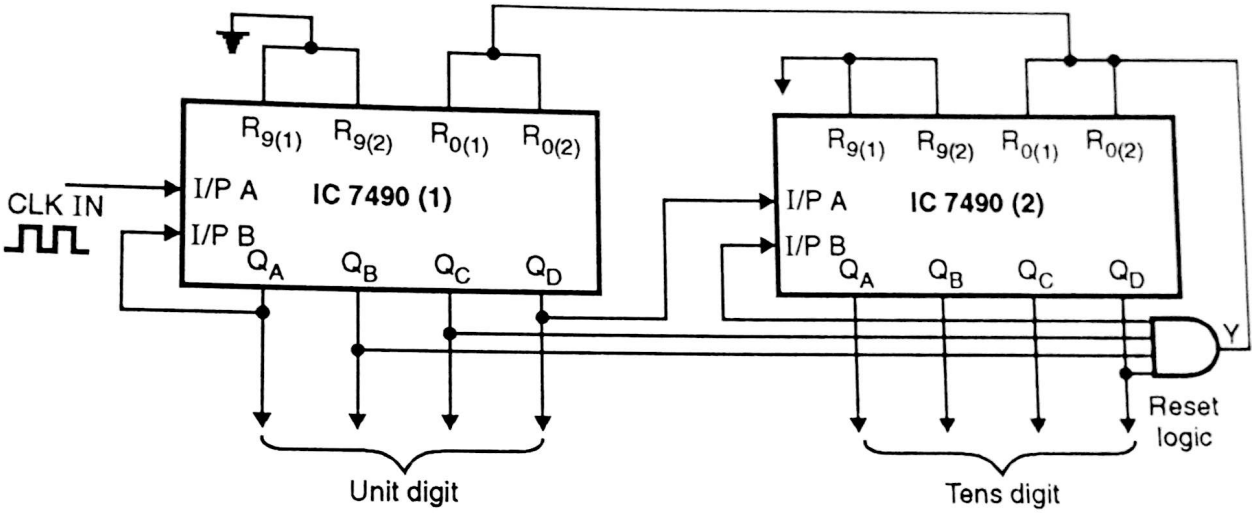



Fig. Ex. 9.23(b)

## 9.12 IC 7492 :

IC 7492 is TTL, MSI divide by 12 counter. IC 7492 is having 2 counter inside it. mod 2 (÷ 2) and mod 6 (÷ 6) counter.

∴ Combination of both gives $2 \times 6 = 12$ ∴ ÷ 12 counter.

**Block schematic of IC 7492 :**

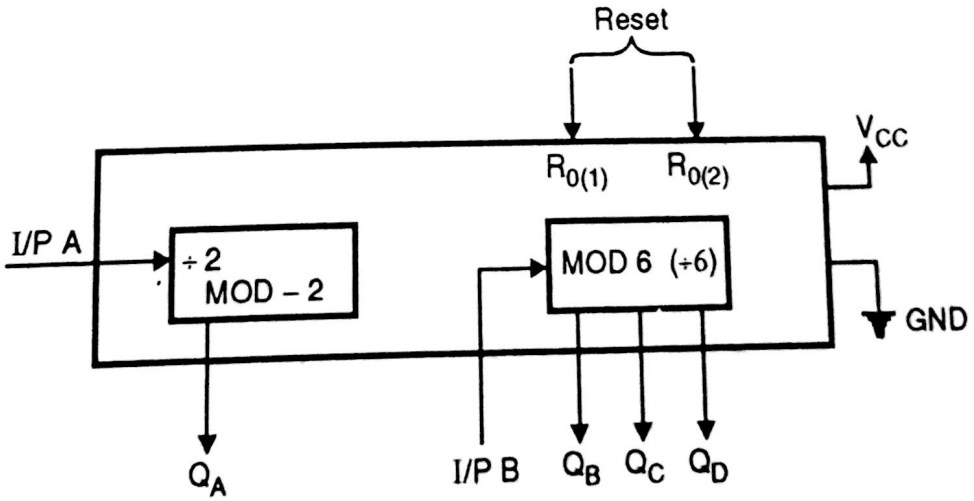



Fig. 9.20

Scanned by CamScanner

Fig. 9.20 shows block schematic of IC 7492. For mod 6 counter IC 7492 counts from 0, 1, 2, 4, 5, 6, 0. (Observe sequence carefully, state 3 is absent).

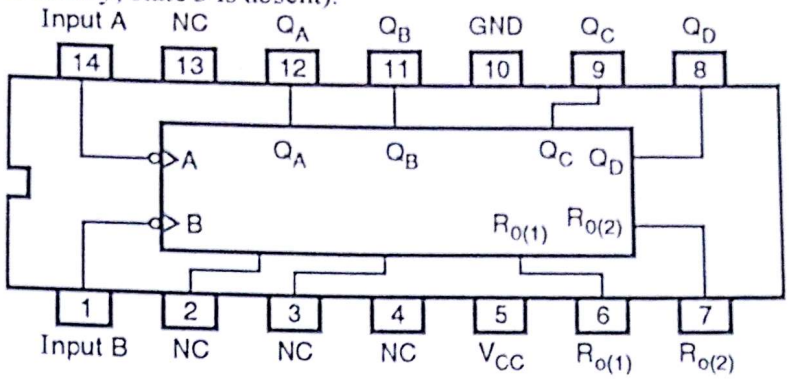



**Fig. 9.21(a)**

**Ex. 24 :** (a) Write down truth table for Mod 6 counter in IC 7492.
(b) Write down truth table for Mod – 12 counter in IC 7492, if CLK IN is given to Input A and Input B $Q_A$ is shorted.

**Soln. :** (a) Truth table of IC 7492 for mod 6 counter. As we know that internally IC has mod 2 and mod 6 counter. So use only mod 6 counter. CLK IN gives to Input B and outputs are $Q_D$, $Q_C$, $Q_B$ only. State are as shown in Fig. Ex. 9.24(a)

| CLK edge | $Q_D$ | $Q_C$ | $Q_B$ | Decimal equivalent |
|---|---|---|---|---|
| 1st | 0 | 0 | 0 | 0 |
| 2nd | 0 | 0 | 1 | 1 |
| 3rd | 0 | 1 | 0 | 2 |
| 4th | 1 | 0 | 0 | 4 |
| 5th | 1 | 0 | 1 | 5 |
| 6th | 1 | 1 | 0 | 6 |
| 7th | 0 | 0 | 0 | 0 |

**Fig. Ex. 9.24(a)**

(b) In this part,
(i) Give CLK IN to Input A
(ii) Connect Input B to $Q_A$.
∴ Circuit will be
as shown in Fig. Ex. 9.24(b)

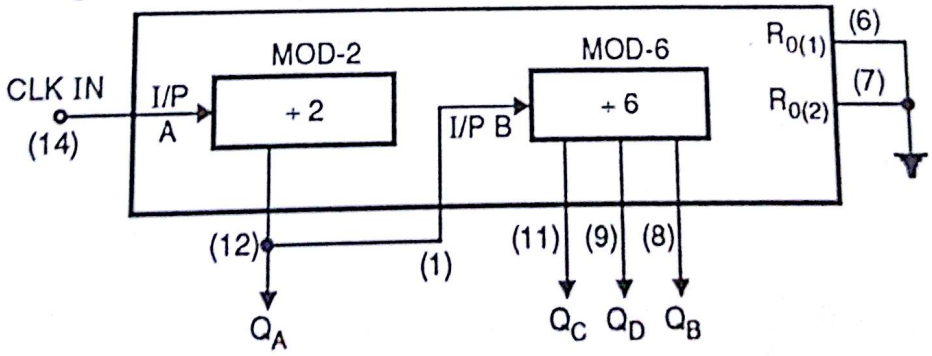



**Fig. Ex. 9.24(b)**

Point to be remembered.
(1) $Q_A$ toggles at each falling edge of CLK IN.

Scanned by CamScanner

(2) $Q_B$ $Q_C$ $Q_D$ will change state at each ***falling edge*** of $Q_A$.

**Working :**

Refer Fig. Ex. 9.24(a).

(1) Initially $Q_D$ $Q_C$ $Q_B$ $Q_A$ (DCBA) = 0000

(2) At first CLK edge, a changes from 0 → 1 (rising edge,)
∴ = 000 ∴ DCBA = 0001.

(3) At 2nd CLK edge, A toggles from 1 → 0 (falling edge,)
∴ DCB will change from 000 to 001. (from Fig. Ex. 9.24(a). DCBA = 0010.

(4) 3rd CLK edge, A changes from 0 → 1 (rising edge),
∴ DCBA = 0011.

(5) 4th CLK edge, A changes from 1 → 0 (falling edge),
∴ DCB changes from 001 to 010. ∴ DCBA = 0100.

(6) At 5th CLK edge, A changes from 0 → 1 (rising edge),
∴ DCB = unchanged ∴ DCBA = 0101.

(7) At 6th CLK edge, a changes from 1 → 0 (falling edge),
∴ DCB changes from 010 to 100 (Decimal count 011 is not present in mod 6 refer Fig. Ex. 9.24(a)
∴ DCBA = 1000

(8) At 7th CLK edge, a change from f → 1,
∴ DCB = unchanged. ∴ DCBA = 1001.

(9) At 8th edge, a changes from 1 → 0,
∴ DCB changes from 100 to 101.
∴ DCBA = 1010

(10) At 9th CLK edge. a changes from 0 → 1,
∴ DCB unchanged, ∴ DCBA = 1011.

(11) At 10th CLK edge, A changes from 1 → 0,
∴ DCB changes from 101 to 110
∴ DCBA = 110

(12) Finally at 11th CLK edge, A changes from 0 → 1,
∴ DCB unchanged, ∴ DCBA = 1101.

Truth table is written as follows.

| CLK edge | $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ | Decimal equivalent |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 0 | 2 |
| 4 | 9 | 9 | 1 | 1 | 3 |
| 5 | 0 | 1 | 0 | 0 | 4 |
| 6 | 0 | 1 | 0 | 1 | 5 |
| 7 | 1 | 0 | 0 | 0 | 8 |
| 8 | 1 | 0 | 0 | 1 | 9 |
| 9 | 1 | 0 | 1 | 0 | 10 |
| 10 | 1 | 0 | 1 | 1 | 11 |
| 11 | 1 | 1 | 0 | 0 | 12 |
| 12 | 1 | 1 | 0 | 1 | 13 |
| 13 | 0 | 0 | 0 | 0 | 0 |

**Fig. Ex. 9.24(c) : Truth table for mod 12 counter**

Scanned by CamScanner

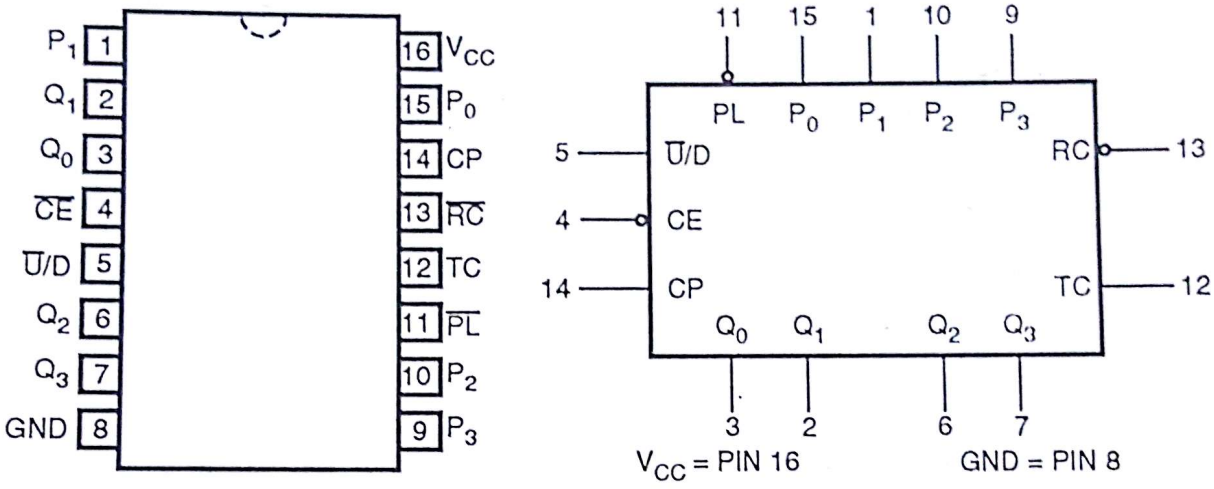
## 9.13 Synchronous Counter ICs :

Let's study some synchronous counter ICs.

### IC 74190

IC 74190 is UP/DOWN decade counter with preset and ripple clock. It is a reversible BCD (8421) decade counter featuring synchronous counting and asynchronous presetting. Reversible BCD means it can count 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 i.e. up and 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 9 i.e. down. Asynchronous presetting means parallel input data can be loaded at any instant, it is not dependent upon CLK input.

Pin configuration and logic symbol is as shown in Fig. 9.22.




(a) Pin (b) Logic Symbol

**Fig. 9.22**

Let's see the pin names and function.

| PIN NAMES | DESCRIPTION |
|---|---|
| $\overline{CE}$ | Count enable input (Active low) |
| CP | Clock pulse input (Active rising edge) |
| $P_0 - P_3$ | Parallel data input |
| $\overline{PL}$ | Asynchronous parallel load input (Active low) |
| $\overline{U}/D$ | Up/Down count control input<br>0 – up count<br>1 – down count |
| $Q_0 - Q_3$ | Flip Flop outputs |
| $\overline{RC}$ | Ripple clock output (Active low) |
| TC | Terminal count output (Active HIGH) |

Basically 74190 is decade counter. Internally four JK FFs are used, so output is 4 bit $Q_0 - Q_3$. It may happen because of supply fluctuation or noise it may enter in invalid state i.e. 10 to 15, then how 74190 behaves ? This point can be explained by state diagram.

Scanned by CamScanner

Refer Fig. 9.23

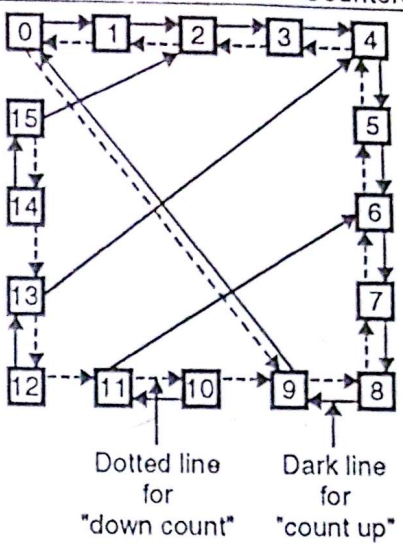



Fig. 9.23

**(1) When counter counts up (Shown by dark line)**

Main loop is → .....0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 ...

Bushing branches are

(1) 14, 15, 2

(2) 12, 13, 4

(3) 10, 11, 6

Where 2, 4 and 6 are valid states. Worst case counter will take maximum three CLK cycle to enter in main loop.

**(2) When counter counts down (Shown by dotted line)**

Main loop is → ..... 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 9 ...

Bushing branch is → 15, 14, 13, 12, 11, 9.

Here we have only one branch. In worst case situation counter will take maximum 7 CLK cycles to enter valid states.

## Different Modes of 74190 :

74190 has different modes of operation. Modes are selected by $\overline{PL}$ , $\overline{CE}$ , $\bar{U}/D$ pins.

**Mode Select Table**

| Inputs | | | | MODE |
|---|---|---|---|---|
| $\overline{PL}$ | $\overline{CE}$ | $\bar{U}/D$ | CP | |
| 1 | 0 | 0 | ⌐¬ | Count Up |
| 1 | 0 | 1 | ⌐¬ | Count down |
| 0 | X | X | X | Preset (Async.) |
| 1 | 1 | X | X | NC (Hold) |

We will study all these modes step by step.

### COUNT UP

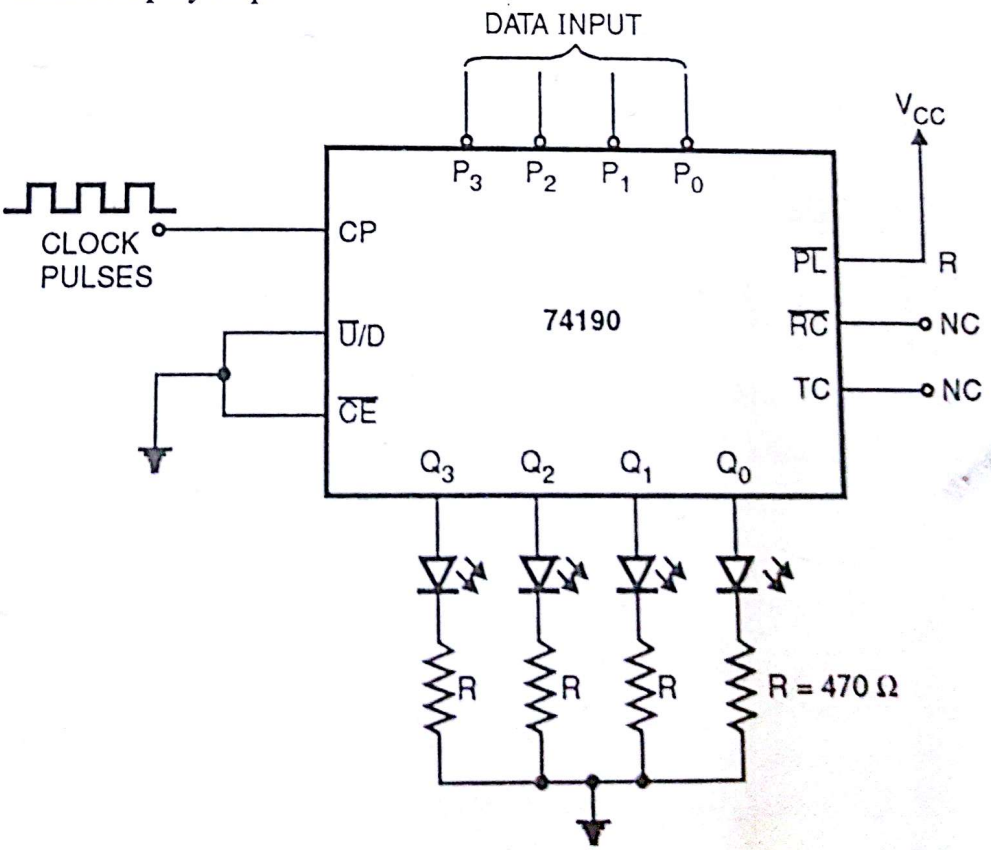



**Fig. 9.24 : UP Counter Using IC 74190**

Scanned by CamScanner

For counter counting up one has to tie $\overline{PL}$ = 1 and $\overline{CE}$ = $\bar{U}$/D = 0. Apply clock pulses and counter counts up. The circuit is as shown as Fig. 9.24.

## COUNT DOWN

For counter counting down tie $\overline{PL}$ = 1 = $V_{CC}$ and $\bar{U}$/D = 1 = $V_{CC}$, $\overline{CE}$ = 0. Apply clock pulses and counter will start counting down. The circuit is as shown in Fig. 9.25.

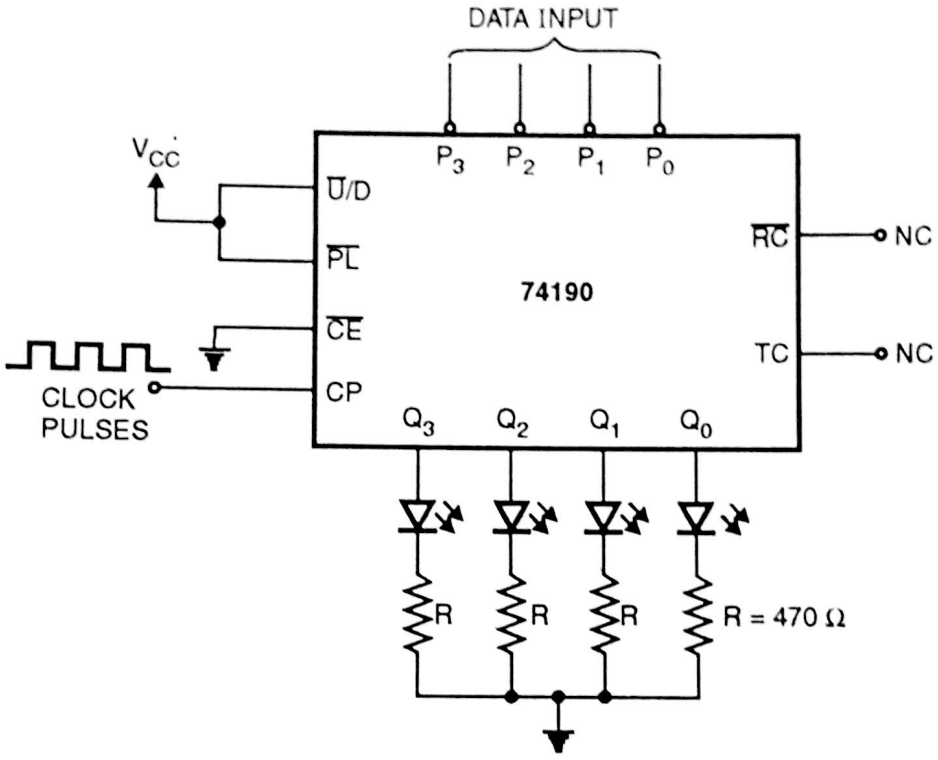



**Fig. 9.25 : Down Counter Using IC 74190**

## PRESET MODE

When you need programmable divider one should PRESET output $Q_0 - Q_3$ to the required count and start counting.

The Fig. 9.26 shows experimental set up for studying presettable Up/down counter.

**Case 1 :** Presettable Up counter.

**Step 1 :** Close switch $S_2$.

**Step 2 :** Apply the count to which you want to PRESET the counter to $P_0 - P_3$ input.

**Step 3 :** Close switch $S_1$. Moment switch is closed $P_0 - P_3$ contents will appear at $Q_3 - Q_0$. You can see it as LED glows.

**Step 4 :** Apply clock pulses. (The pulses should be of low frequency so that you can see output LED changing)

**Step 5 :** Release (open) $S_1$ switch

**Step 6 :** Now counter will start counting from presetted count. Let's say $Q_3 - Q_0 = 0101 = (5)_{10}$.

Scanned by CamScanner

Then it will count 5, 6, 7, 8, 9 and back to '0'. It won't again preset to zero unless and untill S1 is closed again.

If we want that after 9, counter again preset to '5' (or presetted count) then we have to add combinational circuit. We will study it within no time.

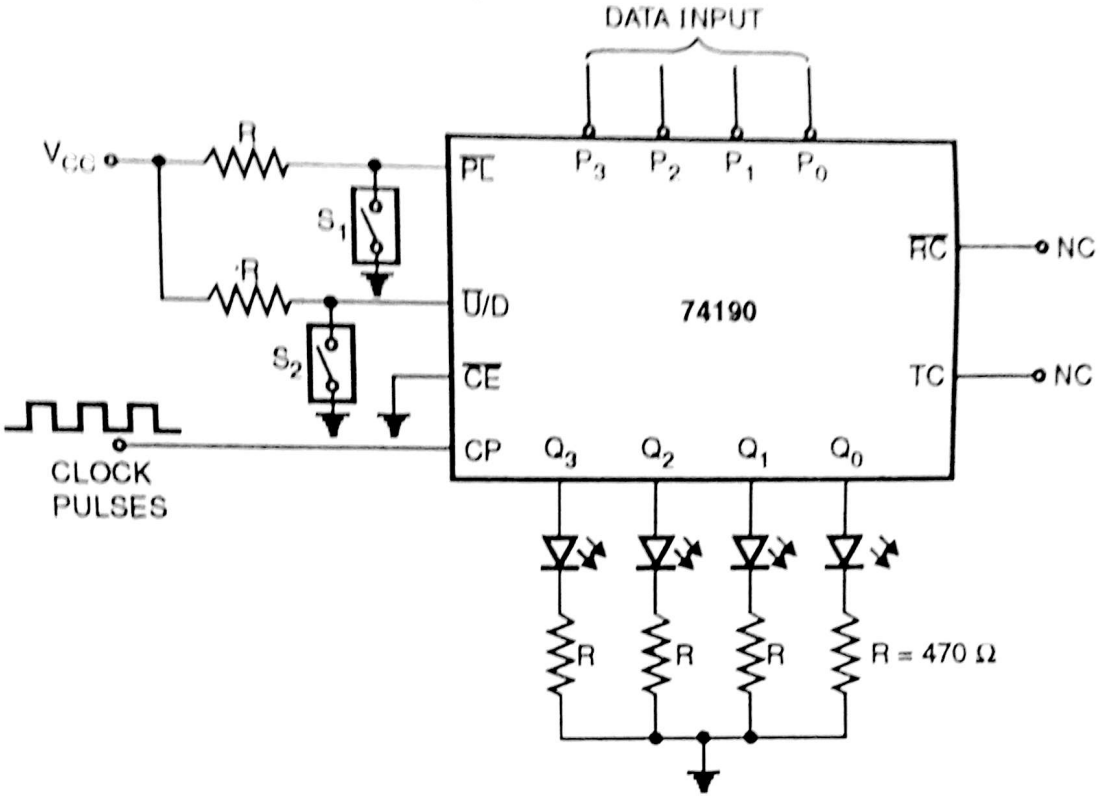



**Fig. 9.26 : Presettable up / down counter**

**Case 2 :** Presettable down counter.

**Step 1 :** Open switch $S_2$.

**Step 2 :** Steps written in case 1 from (2) to (5) is same here also.

**Step 3 :** Now counter will start counting from presetted count.

Let's say $Q_3 - Q_0 = 0101 = (5)_{10}$

Then counter will count 5, 4, 3, 2, 1, 0 and back to 9. It won't preset again to 5 in this circuit unless and untill $S_1$ is closed again.

**HOLD**

To HOLD or LATCH the count in the counter make $\overline{PL}$ = $\overline{CE}$ = 1 $\bar{U}$/D and CP, both pins are treated ***don't care***.

---

**Ex. 25 :** Design presettable up counter, free running using 74190.

**Soln. :**

(1) Let's assume presettable count = 5 = $(0101)_2$

(2) States of counter will be ... 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5 ...

(3) Design of combinational circuit.

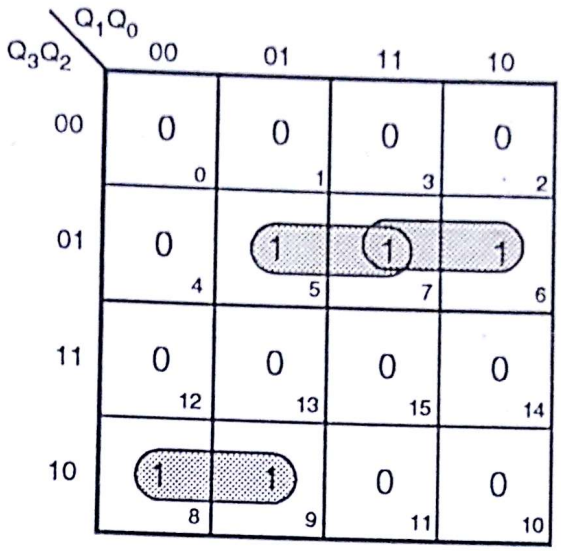
One has to design combinational circuit in such a way that when valid states are present output of the circuit should be '1' and for invalid states output should be '0', because we are going to tie output of combinational circuit to $\overline{PL}$.

$\overline{PL}$ = 0 → Preset count

$\overline{PL}$ = 1 → Counter Counts.

Scanned by CamScanner

(4) **K – Map :**




$$Y = Q_3\overline{Q}_2\overline{Q}_1 + \overline{Q}_3Q_2(Q_0 + Q_1)$$

(5) **Circuit diagram :**

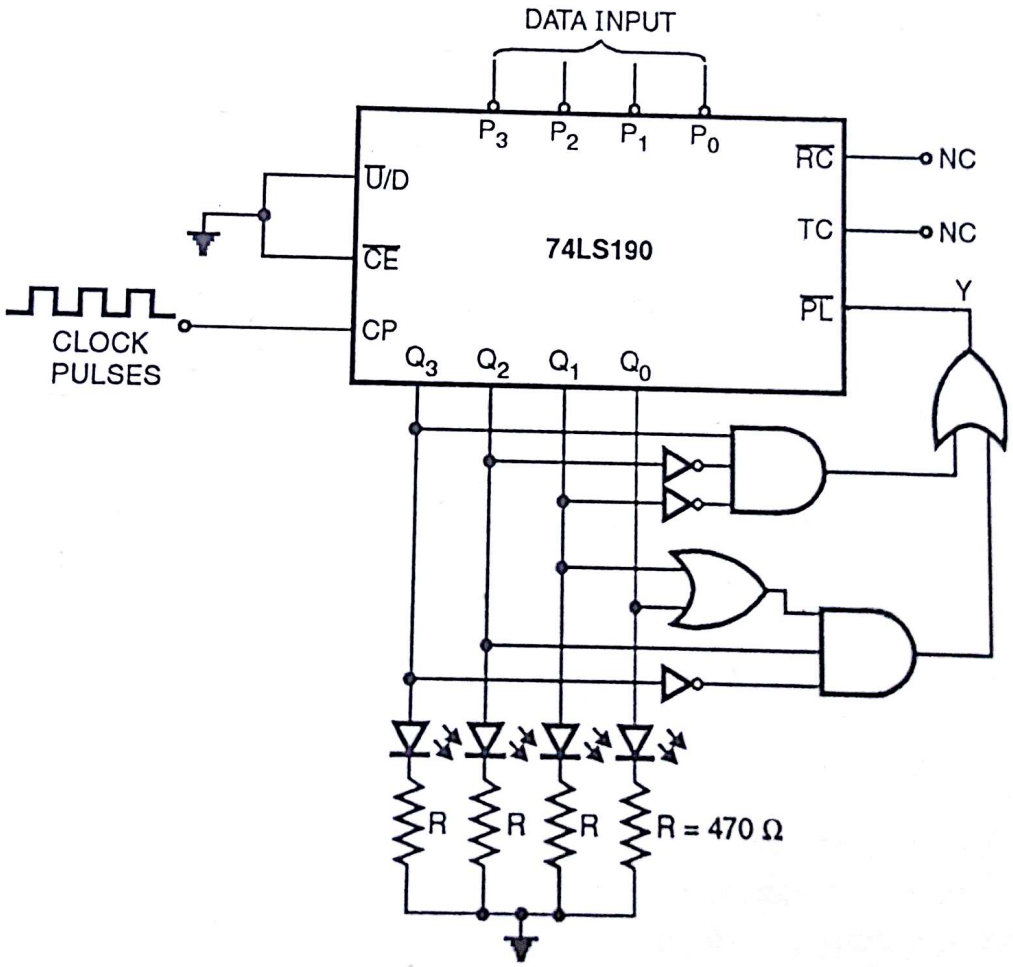



**Fig. Ex. 9.25(a)**

**Ex. 26 :** Design presettable down counter using 74190.

**Soln. :**

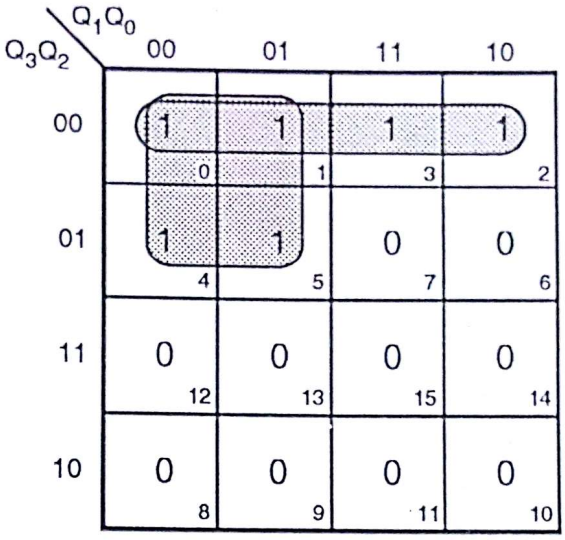
(1) Let's assume presttable state = 5 = $(0101)_2$

Scanned by CamScanner

(2) States of counter will be ... 5, 4, 3, 2, 1, 0, 5, 4, 3, 2, 1, 0, 5 ...

(3) Design of combinational circuit.
It should produce HIGH output for above valid states and low output for remaining Invalid states.

(4) **K Map :**




$$Y = \overline{Q}_3\overline{Q}_2 + \overline{Q}_3\overline{Q}_1$$
$$= \overline{Q}_3(\overline{Q}_2 + \overline{Q}_1)$$

(5) **Circuit diagram :**

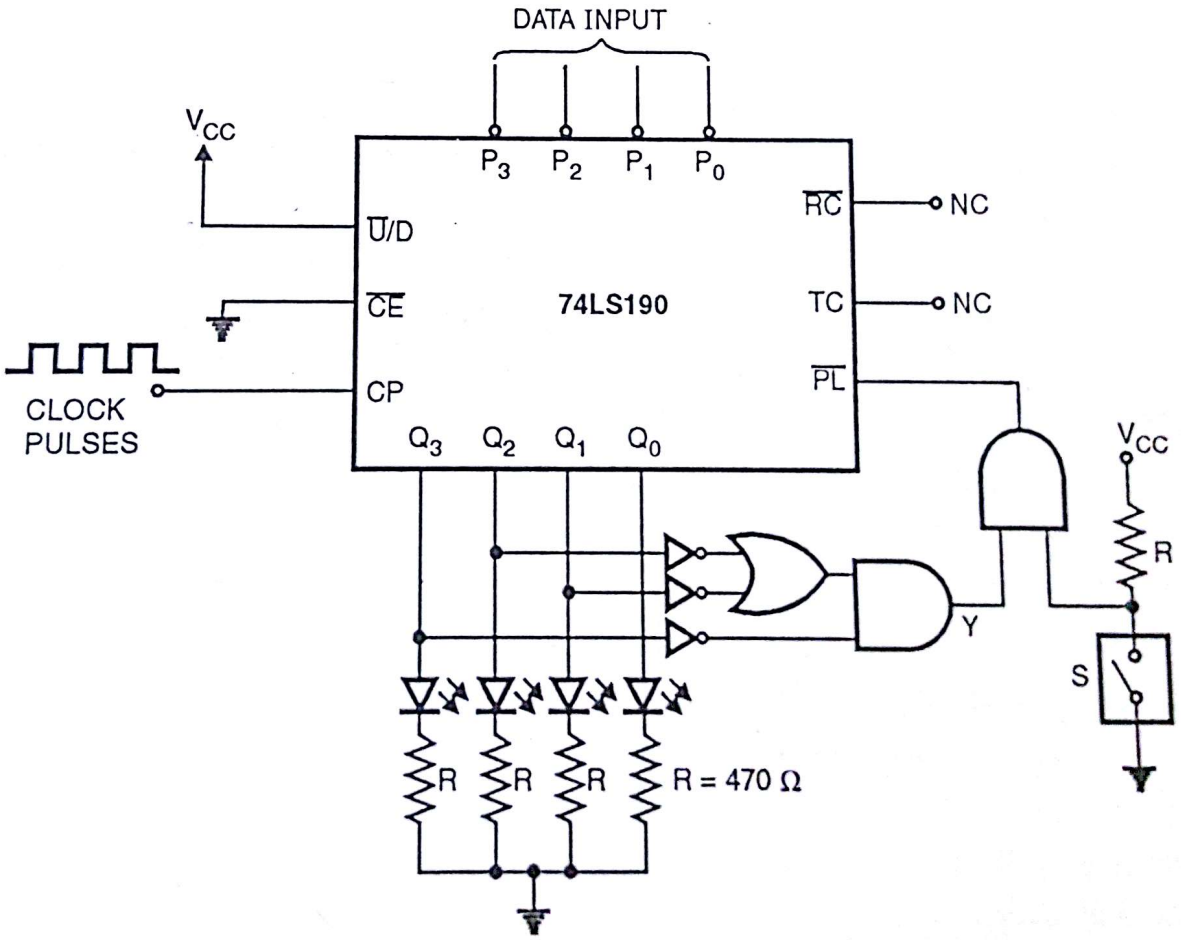



**Fig. Ex. 9.26(a)**

(6) Initially keep switch S closed, apply clock pulses. $Q_3 - Q_0 = 0101$.
Now open switch S, so counter will start counting.

Scanned by CamScanner

**Function of TC and RC :**

Two types of outputs are provided as overflow/under flow indicators. The TC (terminal count) output is normally low and goes HIGH when a circuit reaches zero in the count down mode or reaches maximum in count up mode. The TC output remains HIGH untill state change occurs, whether by counting or presetting or until $\overline{U}/D$ is changed.

**Precaution :** The TC output should not be used as clock signal because it is subject to decoding spikes. The function of TC is for cascading the counter or for mutilatage counter.

The TC signal is also used internally to enable ***ripple clock*** output. The ripple clock ( $\overline{RC}$ ) output is normally HIGH. When $\overline{CE}$ is LOW and TC is HIGH, the $\overline{RC}$ output will go LOW when the clock next goes low and will stay low untill CLK goes HIGH again. The truth table for $\overline{RC}$ output is shown, as follows.

RC Truth table,

| Inputs | | | Output |
|---|---|---|---|
| $\overline{CE}$ | TC* | CP | $\overline{RC}$ |
| 0 | 1 | ⊓ | ⊓ |
| 1 | X | X | 1 |
| X | 0 | X | 1 |

* TC is generated Internally.

We will learn two circuit, Of multistage counter using $\overline{RC}$ pin for cascading.

**1. N – stage counter using ripple CLK :**

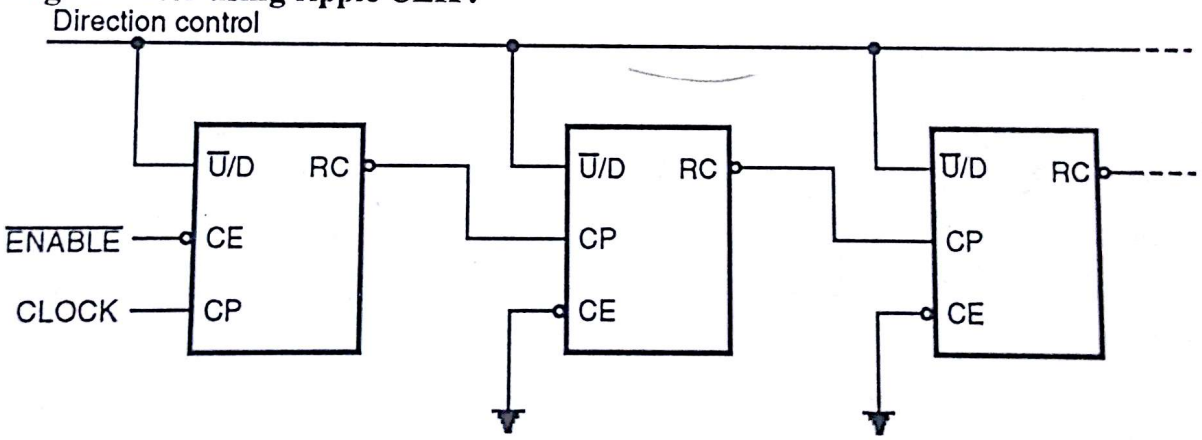



**Fig. 9.27 : N stage counter using ripple clock**

Refer Fig. 9.27

Each $\overline{RC}$ output is used as CLK input for the next higher stage. This configuration is particularly advantageous when the CLK source has a limited drive capability, since it drives only first stage. To prevent counting in all stages it is only necessary to inhibit the first stage, since a HIGH signal on $\overline{CE}$ inhibits the $\overline{RC}$ output pulse.

Disadvantage of this configuration, in some applications, is the timing skew between state changes in the first and last stage. This represents cumulative delay of the CLK as it ripples through the preceding stages, because it is basically ripple or asynchronous counter.

Scanned by CamScanner

**2. Synchronous N – Stage Counter using Ripple Carry/borrow :**

A method of causing state changes to occur simultaneously in all stages is shown in Fig. 9.28.

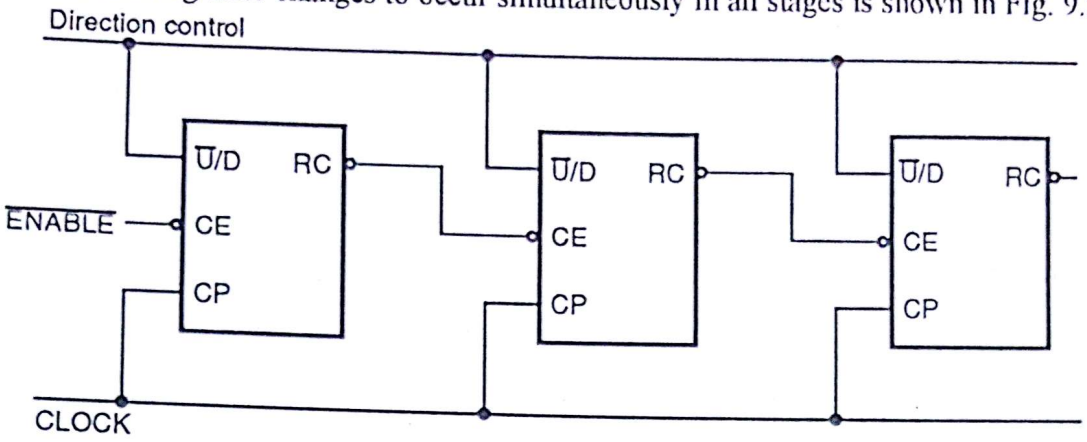



**Fig. 9.28 : Synchronous N stage counter using ripple carry / borrow**

All clock inputs are driven in parallel and $\overline{RC}$ output propagates the carry/borrow signals in the ripple fashion. In this configuration the low state duration of the clock must be long enough to allow the negative–going edge of the carry/borrow signal to ripple through to the last stage before the clock goes HIGH. There is no such restriction on the HIGH state duration of the CLK. Since the $\overline{RC}$ output of any package goes HIGH shortly after its CP input goes HIGH.

Let's study multistage counter using TC as cascading line.

**(1) Synchronous N – Stage Counter with Parallel Gated Carry/borrow :**

To avoid ripple delays and their associated restrictions, the following Fig. 9.29 should be used.

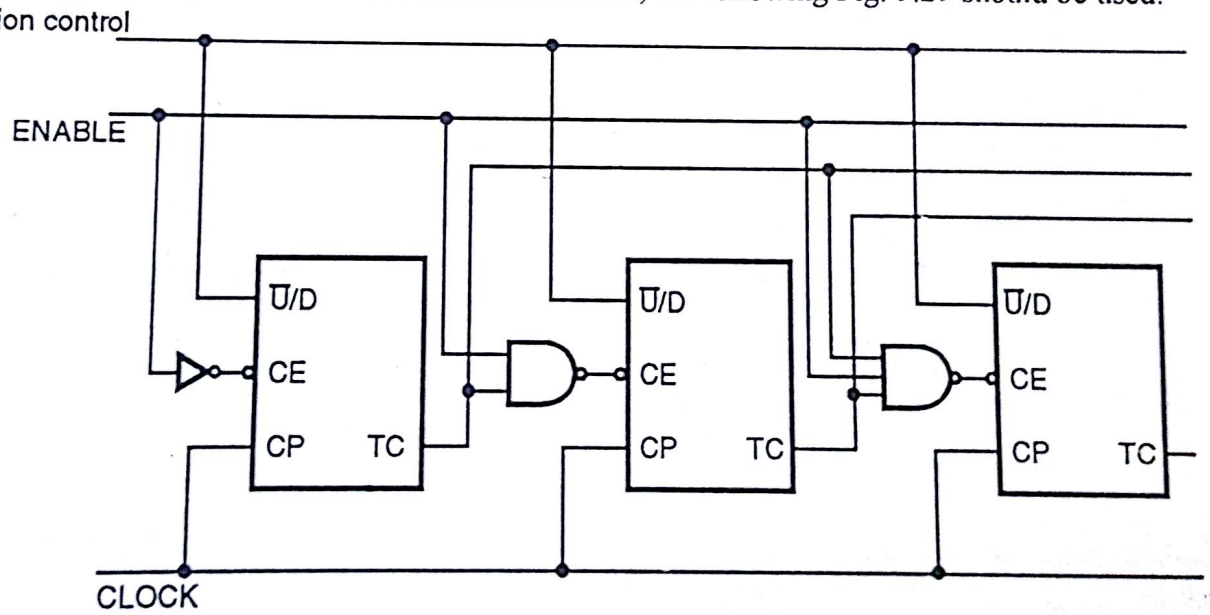



**Fig. 9.29**

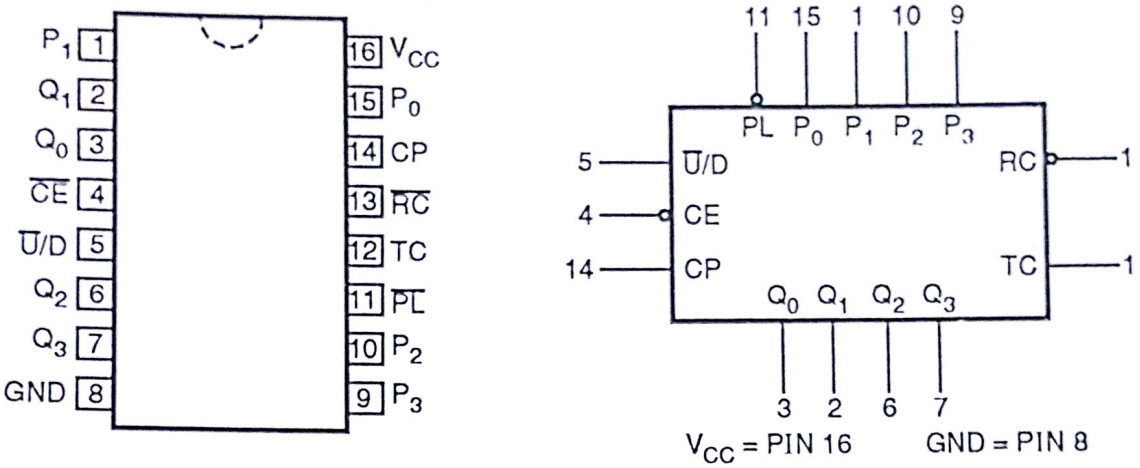
(1) Clock terminal of all ICs are tied together.

(2) The $\overline{CE}$ input for a given stage is formed by combining the TC signals from all the preceding stages. In order to inhibit counting an enable signal must be included in each carry gate. This is done because TC output of a given stage is not affected by its own $\overline{CE}$ .

Scanned by CamScanner

## IC 74191

IC 74191 is binary up/down counter with preset and ripple clock. If you compare it with 74190, IC 74190 is decade counter, where as 74191 counts from 0 to 15. This IC can count up or down synchronously and has a synchronous preset. Preset feature allows the chip to be used in dividers. The pin configuration and logic symbol is as shown in Fig. 9.30.




**Pinout** **Logic symbol**

**Fig. 9.30**

Pin names are as follows :

| Pin Names | Description |
|---|---|
| $\overline{CE}$ | Count enable input (Active low) |
| CP | Clock pulse input (Active rising edge) |
| $P_0 - P_3$ | Parallel data inputs |
| $\overline{PL}$ | Asynchronous parallel load input (Active low) |
| $\overline{U/D}$ | Up/Down count control input |
| $Q_0 - Q_3$ | Flip Flop outputs |
| $\overline{RC}$ | Ripple clock output (Active low) |
| TC | Terminal count output (Active HIGH) |

State diagram for the chip is as shown in Fig. 9.31

74191 counts up from 0 to 15 (shown by dark line) and also counts down 15 to 0 (shown by dotted line).

74191 can be operated in different modes.

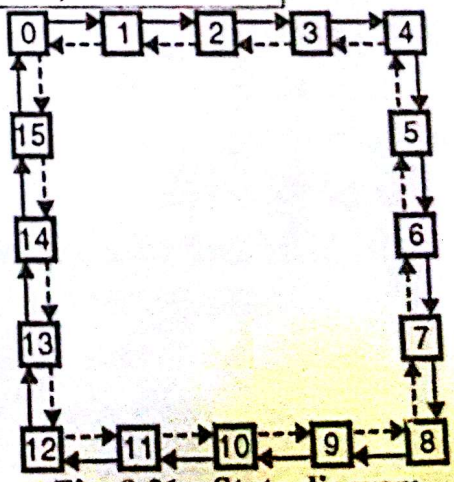



**Fig. 9.31 : State diagram**

Scanned by CamScanner

Mode select table.

| INPUTS | | | | MODE |
|---|---|---|---|---|
| PL | $\overline{CE}$ | U /D | CP | |
| 1 | 0 | 0 | _\|‾ | Count Up |
| 1 | 0 | 1 | _\|‾ | Count down |
| 0 | X | X | X | Preset (Async) |
| 1 | 1 | X | X | No change (HOLD) |

RC Truth Table

| Inputs | | | Output |
|---|---|---|---|
| $\overline{CE}$ | TC | CP | $\overline{RC}$ |
| 0 | 1 | | |
| 1 | X | X | 1 |
| X | 0 | X | 1 |

The modes are same as 74190, there for explanation given in 74190 holds for 74191 also.

For cascading or multistage counter, the configuration explained in 74190 holds for 74191 also. RC truth table is also similar.

## 9.14 Analysis of Counter Circuit :

Uptill now we have solved some example based on T, D and JK FF design. This part was designing the counter circuit or synthesis of counter, where we were given state diagram and we have to find circuit for the same.

Now it is analysis, i.e. circuit will be given to you and we have to find state diagram, and some time unused state will be given to us and we have to find how counter is going to behave.

The method is very simple. Explained point by point.

(1) Firstly present state will be given to you e.g. say presently counter is in CBA = 010, now find next states.

By chance if the state is not given, you assume some state. The thumb rule is to start from 000.

(2) This present state is now previous state for us.

(3) You have to now find out, I/ps to all the FFs, because of this previous state.

(4) After finding out inputs of all FFs. assume that CLK edge hits. Now you already know the input, you know the truth table of each FF. So find out what will be Q output of FF. This 'new' state will be ***next state*** of the FF.

(5) Now you assume the ***next state*** as previous and start from point 3.

This chain should be continued unless and untill you get ***next state***, which is already existing previously. For example. We started with state 4. From 4 next state is 6. From 6 next is 8. From 8 next is 9 and from 9 next state is 4. Means '4' has occurred twice so chain is complete. If you want you can draw state diagram as shown in Fig. 9.32.

Scanned by CamScanner

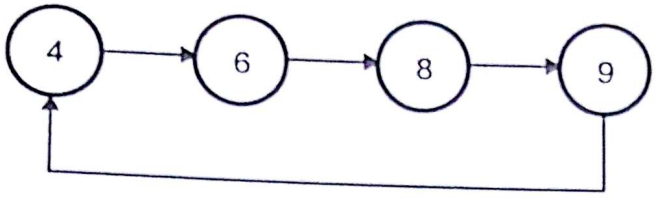



Fig. 9.32

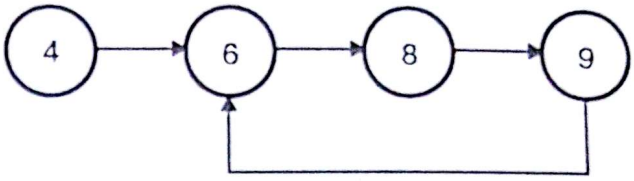



Fig. 9.33

In above example it *may* happen that you get 6, after 9. 6 is already present. But now state diagram will be different. It is as shown in Fig. 9.33. ∴ 4 is ***bushing***.

Thus you go on trying different combination and find ***next states***. To make analysis simple, all 5 points can be combined in 'one' table. You keep the table standard now onwards.

**Table**

| | Previous State | | | | FF inputs | | | | ∴ Next state | | | Decimal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | $\bar{A}$ | B | $\bar{B}$ | $J_A$ = | $K_A$ = | $J_B$ = | $K_B$ = | A | B | C | equiv. |
| Start | | | | | | | | | | | | |

Note : In above table $J_A$ = _____, $K_A$ = _____. The blank space is left for writing Boolean expression (s).

**Ex. 27 :** The logical diagram of counter is given in Fig. Ex. 9.27. If the initial condition of the counter is ABC = 110. Find the states through which the counter goes when clock is run. What happens if the counter is initially found in 011 state.

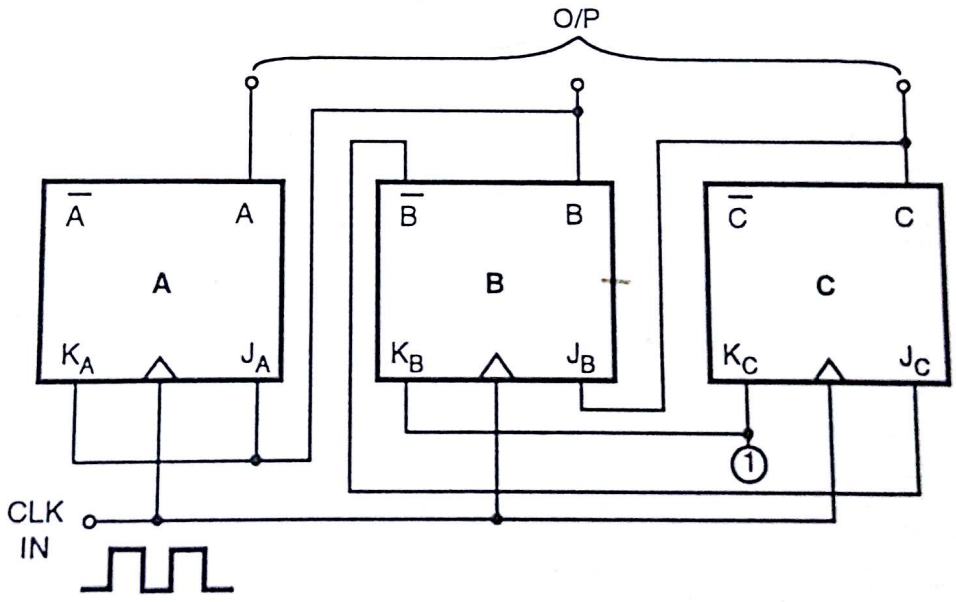



**Fig. Ex. 9.27(a)**

**Soln. : Step – I :** First write down Boolean expression for FFs inputs.

(1) $J_A = B$

(2) $K_A = B$

(3) $J_B = C$

(4) $K_B = 1$ (Permanently Vcc)

Scanned by CamScanner

(5) $J_C = \bar{B}$

(6) $K_C = 1$ (Permanently Vcc.)

**Step II :** Write down initial condition ABC = 110

$\therefore$ A = MSB C = LSB.

**Step III :** Make table.

| | Previous state | | | | | | FF Inputs | | | | | | Next State | | | Decimal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | $\bar{A}$ | B | $\bar{B}$ | C | $\bar{C}$ | $J_A = B$ | $K_A = B$ | $J_B = C$ | $K_B = 1$ | $J_C = \bar{B}$ | $K_C = 1$ | A | B | C | Count |
| start → | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | | | | |
| | | | | | | | | | | | | | | | | |

**Fig. Ex. 9.27(b)**

and as shown in Fig Ex. 9.27(b), we have written ABC = 110 and complement of ABC.

As B = 1, $J_A = K_A = B = 1$. As C = 0, $J_B = 0$. As $J_C = \bar{B}$ and $\bar{B} = 0$, $J_C = 0$. $K_B = K_C = 1$ (Permanent).

Now assume CLK edge hits. (Refer Truth table of JK FF).

(1) $J_A = 1$ $K_A = 1$ $\therefore$ A will toggle $\therefore$ A changes from $1 \rightarrow 0$.

(2) $J_B = 0$ $K_B = 1$ $\therefore$ B = 0 (irrespective of previous state.)

(3) $J_C = 0$ $K_C = 1$ $\therefore$ C = 0 (irrespective of pervious state.)

$\therefore$ Next state = ABC = 000

Write this state under heading of ***Next state*** in Fig. Ex. 9.27(b).

This was 1st cycle.

Now in 2nd cycle take next state as previous state.

$\therefore$ 0 0 0 = ABC is now previous state. Write it under heading of ***previous state*** in Fig. Ex. 9.27(b). Find complement of ABC. Find out $J_A$, $K_A$ ... and so on. Again assume CLK edge hits. and find out ***new*** next state.

Continue the same till you get same decimal count. The full table is shown below. Start = ABC = 110 = 6 (decimal).

| Previous state | | | | | | FF Inputs | | | | | | Next State | | | decimal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | $\bar{A}$ | B | $\bar{B}$ | C | $\bar{C}$ | $J_A = B$ | $K_A = B$ | $J_B = C$ | $K_B = 1$ | $J_C = \bar{B}$ | $K_C = 1$ | A | B | C | Count |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 4 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 5 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 6 |

$\therefore$ States are $6 \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$

↑ start ↑ start

Scanned by CamScanner

∴ state diagram

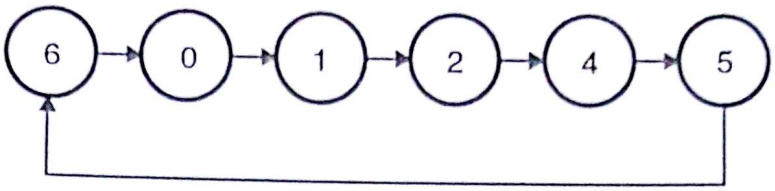



Fig. Ex. 9.27(c)

Second part of the problem is if the counter is initially found in 011 state then what will be best states.

Procedure is again same.

Truth table : Start ABC = 011 = 3 (decimal).

| Previous State | | | | | | FF Inputs | | | | | | Next State | | | Decimal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | $\bar{A}$ | B | $\bar{B}$ | C | $\bar{C}$ | $J_A = B$ | $K_A = B$ | $J_B = C$ | $K_B = 1$ | $J_C = \bar{B}$ | $K_C = 1$ | A | B | C | Count |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 4 |

Fig. Ex. 9.27(d)

If you see Fig. Ex. 9.27(d), After ABC = 011 (3) next state is ABC = 100 (4). 4 is present in counter chain ∴ you stop. Even though you proceed further next state will be 5 then 6 then 0 then 1 and so on.

∴ Final state diagram is,

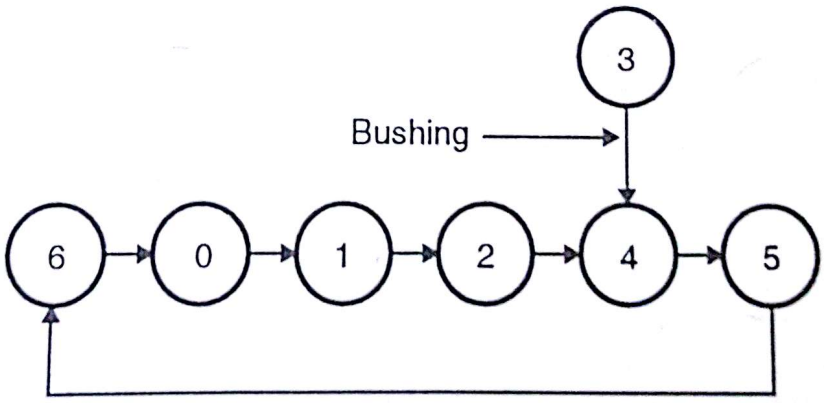



Fig. Ex. 9.27(e)

**Ex. 28 :** The schematic arrangement of synchronous counter which uses master / slave JK FF is shown in Fig. Ex. 9.28. If the Initial condition of the counter is ABCD = 0001, Find the states through which the counter goes when CLK is seen.

What happens if the counter is found initially in state ABCD = 1000 ?

Scanned by CamScanner

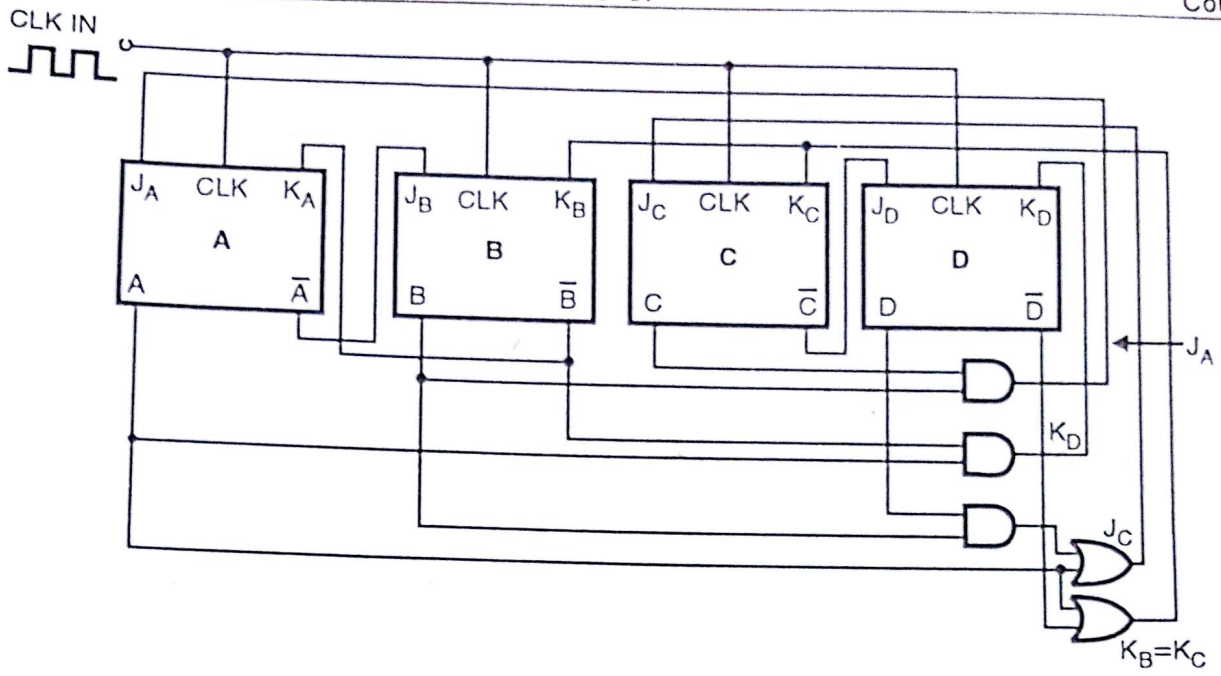



**Fig. Ex. 9.28**

**Soln. :**

**Step I :** Don't worry [I am with you]

**Step II :** Write Boolean expression for inputs of FFs.

$J_A = BC$ $\quad J_C = BD + A$

$K_A = \bar{B}$ $\quad K_C = A + \bar{D}$

$J_B = \bar{A}$ $\quad J_D = \bar{C}$

$K_B = A + \bar{D}$ $\quad K_D = A\bar{B}$

COLLEGE OF COMMERCE LIBRARY Acc No 743

**Step III :** Make table. and start from state ABCD = 0001 (1 decimal). Refer Table Ex. 9.28(1).

$\therefore$ State diagram : Fig. Ex. 9.28(a)

Second part is, if counter is initially ABCD = 1000

Same table.

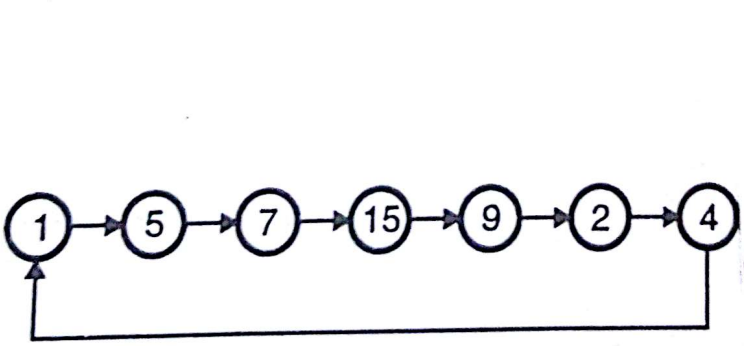



**Fig. Ex. 9.28(a)**

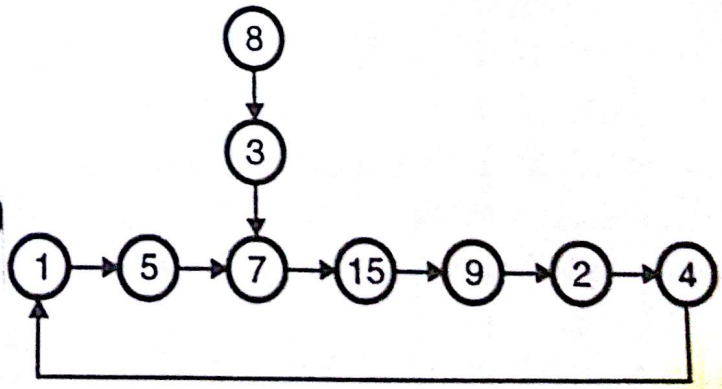



**Fig. Ex. 9.28(b)**

**States are (8) → (3) → (7). $\therefore$ Final state diagram is as shown in Fig. Ex. 9.28(b).**

Scanned by CamScanner

| Previous State | | | | | | | | FF Inputs | | | | | | | | Next states | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | $\bar{A}$ | B | $\bar{B}$ | C | $\bar{C}$ | D | $\bar{D}$ | $J_A$= BC | $K_A$ = B | $J_B$ = $\bar{A}$ | $K_B$ = A + $\bar{D}$ | $J_C$ = A + BD | $K_C$ = A + $\bar{D}$ | $J\bar{D}$ = $\bar{C}$ | $K_D$ = A $\bar{B}$ | A | B | C | D | Decimal count |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 7 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 15 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 2 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1← Stop |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 7← Stop |

**Table Ex. 9.28(1)**

Scanned by CamScanner

**Ex. 29 :** Fig. Ex. 9.29 shows counter which must count through more than one set of states as determined by the control line.

Find the states through which counter goes when the control line (i) low and (ii) high

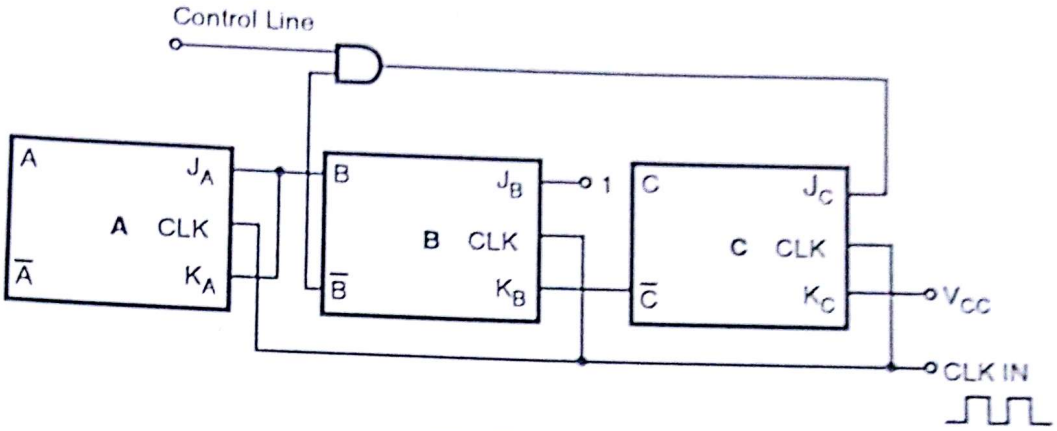



**Fig. Ex. 9.29**

**Soln. :**

**Case I :** Control line is HIGH.

Procedure is same as we followed in last example.

$J_A = K_A = B$ $\quad J_C = (\text{control line}).\ \bar{B} = (\text{con})\ \bar{B}$

$J_B = 1$ $\quad K_C = 1$

$K_B = \bar{C}$

Control line = CON = 1 Assume initial state = 0 0 0 = ABC

| Previous state | | | | | | Inputs of FF | | | | | | Next state | | | Decimal counter |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | $\bar{A}$ | B | $\bar{B}$ | C | $\bar{C}$ | $J_A = B$ | $K_A = B$ | $J_B = 1$ | $K_B = \bar{C}$ | $J_C = (CON.)\cdot\bar{B}$ | $K_C = 1$ | A | B | C | |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 3 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 6 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

**∴ State diagram :**

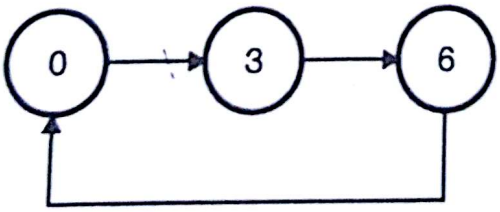



**Fig. Ex. 9.29(a)**

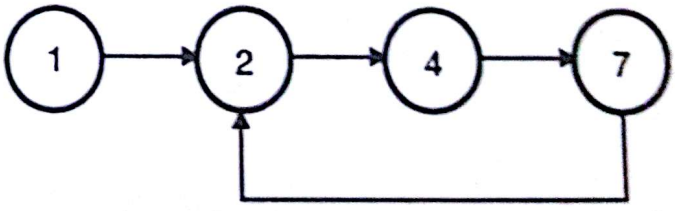



**Fig. Ex. 9.29(b)**

Only three states are coming. So let's start with assumption that initially counter is at 001 = ABC = decimal 1.

Scanned by CamScanner

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 4 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |

∴ state diagram is as shown in Fig. 9.29(b)

But still state 5 is missing. so start with state 5 ABC = 101

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| start | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 6 |

∴ We have 2 state diagram

If start ABC = 00

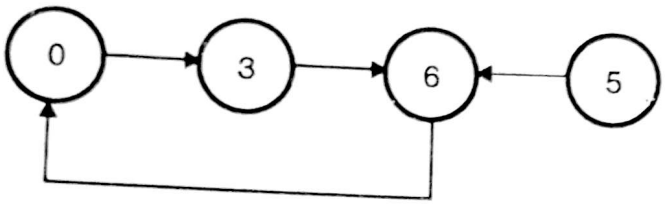



Fig. Ex. 9.29(c)

If ABC = 001

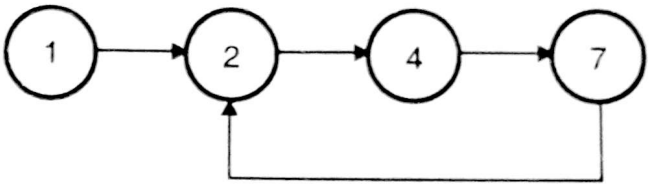



Fig. Ex. 9.29(d)

Here I would like to Indicate that if you consider state diagram Fig. Ex. 9.29(c) as valid sequence, then because of supply fluctuation or EMZ or RFZ interference if counter enters into state diagram of Fig. Ex. 9.29(d) it will never come out of that chain and will lockout it self in invalid states.

Now let's make control line = 0 Initially ABC = 000 = 0 decimal

| | Previous state | | | | | | Input of FF | | | | | | Next state | | | Decimal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | $\bar{A}$ | B | $\bar{B}$ | C | $\bar{C}$ | $J_A$ = B | $K_A$ = B | $J_B$ = 1 | $K_B$ = $\bar{C}$ | $J_C$= CON $\bar{B}$ | $K_C$ = 1 | A | B | C | Count |
| start→ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 2 |
| | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 4 |
| | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 6 |
| | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

Now take Initial condition ABC = 001 = 1 decimal

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| start→ | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |

Now take Initial condition ABC = 011 = 3 decimal

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| start→ | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 6 |

Now take Initial condition ABC = 101 = 5 decimal

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| start→ | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 6 |

Now take initial condition ABC 111 = 7 decimal

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| start→ | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |

Fig. Ex. 9.29(e)

Scanned by CamScanner

If you observe Fig. Ex. 9.29(e) you will find that main chain of counter is 0, 2, 4, 6, 0. If you select 1, next state is 2 (valid state of counter). Same way for 3, next state is 6. For 5, next state is also 6. For 7, next state is 2. ∴ Final state diagram is

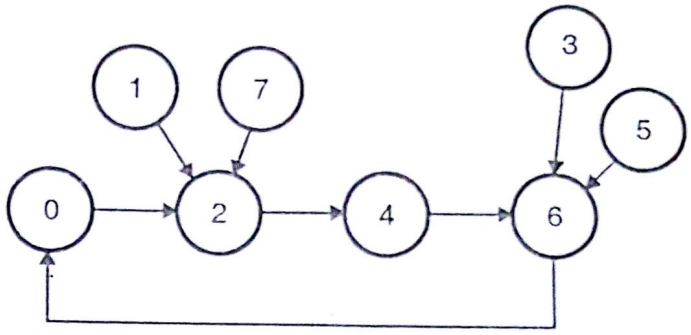



Fig. Ex. 9.29(f)

**Ex. 30 :** A counter using negative edge triggered flip flop is shown in Fig. Ex. 9.30. Find the states through which counter goes when CLK is run. Also draw tining diagram showing relation between CLK input and output of FF.

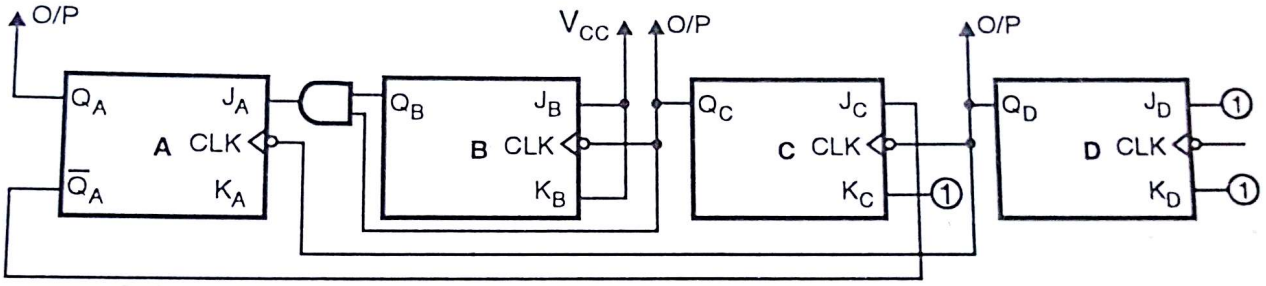



**Fig. Ex. 9.30**

**Soln. :** (1) The counter in Fig. Ex. 9.30 is ripple counter. The way of analysis is same.

(2) Remembers : CLK is given to only one FF and output changes on 1 → 0 (Falling edge) of the CLK input.

(3) Truth table : Initially ABCD = 0 0 0 0 = 0 decimal. $CLK_A = CLK_C = D$, $CLK_B = C$

**Explanation :**

(1) Initially ABCD = 0 0 0 0 (Refer Fig. Ex. 9.30(a)

(2) At 1st CLK edge hit, D changes from 0 → 1 (rising edge) ∴ ABC are unchanged. ABCD = 0001

(3) At 2nd CLK edge hit, D changes from 1 → 0 (falling edge – shown by arrow in D column under next state heading.).

∴ C changes state from 0 → 1 (rising edge) as $J_C = 1$ $K_C = 1$.

A remains unchanged even though CLK is present because $J_A = 0$ $K_A = 1$ ∴ ABCD = 0010.

(4) At 3rd CLK 'D changes from 0 → 1 (rising edge). ∴ ABC are unchanged ∴ ABCD = 0011.

(5) At 4th CLK edge, D changes from 1 → 0 (falling edge),

∴ C changes from 1 → 0 (falling edge) as $J_C = 1$ $K_C = 1$, this will force B to change from 0 → 1 as $J_B = 1$ $K_B = 1$. A remains unchanged as $J_A = 0$ $K_A = 1$ ∴ ABCD = 0 100.

Thus cycle will repeat and counter will count 0101, 0110, 0111, 1000 and 1001. After 1001 it will reset to 0.

Scanned by CamScanner

| Previous state | | | | | | | Input of FF(s) | | | | | | | | | Next state | | | | CLK | Decimal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | $\overline{A}$ | B | $\overline{B}$ | C | $\overline{C}$ | D | $\overline{D}$ | $J_A = BC$ | $K_A = 1$ | $J_B = 1$ | $K_B = 1$ | $J_C = \overline{A}$ | $K_C = 1$ | $J_D = 1$ | $K_D = 1$ | A | B | C | D | edge ↓ | count |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1st | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 2nd | 2 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 3rd | 3 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 4th | 4 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 5th | 5 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 6th | 6 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 7th | 7 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 8th | 8 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 9th | 9 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 10th | 0 |

**Fig. Ex. 9.30(a)**

Scanned by CamScanner

**Waveform** : Refer Fig. Ex. 9.30(b).

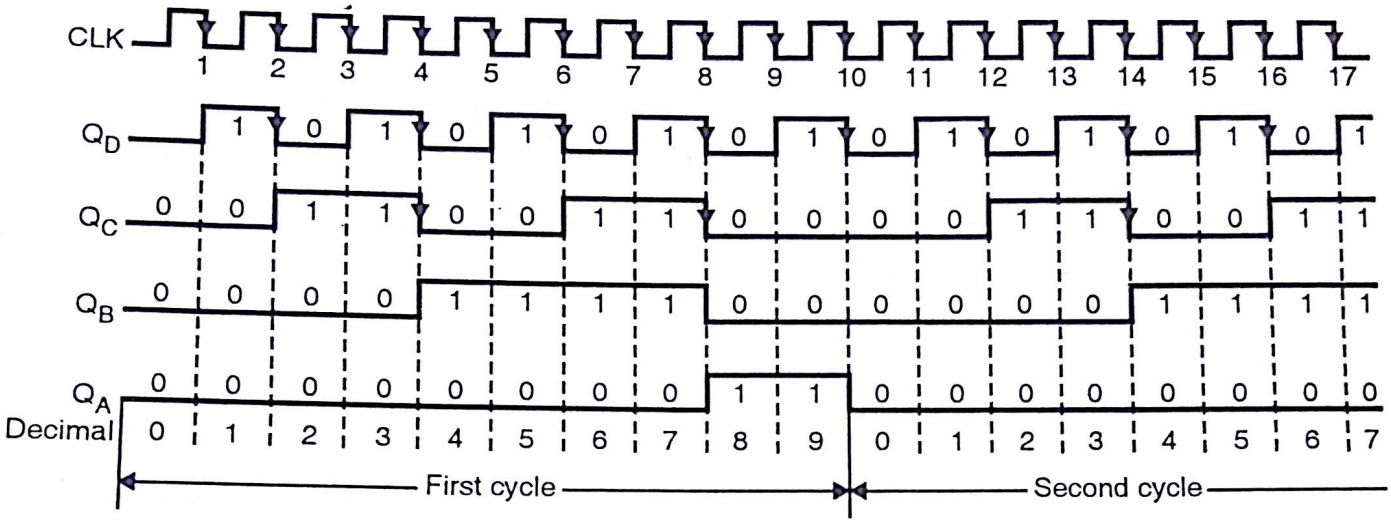



**Fig. Ex. 9.30(b)**

Scanned by CamScanner

## 9.15 Examples from University Papers :

1. For the logic circuit shown in Fig. Q. 9.1 (a) draw the timing diagram of CP, $Q_2$ and $Q_1$ assuming initial condition $Q_2 = Q_1 = 0$. (May 96, 10 Marks)

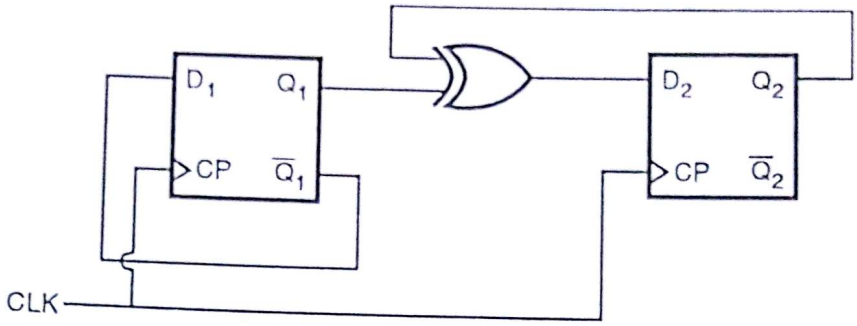



Fig. Q. 9.1 (a)

**Ans. :** Waveforms for CP, $Q_1$ and $Q_2$ are given in Fig. Q.9.1(b) :

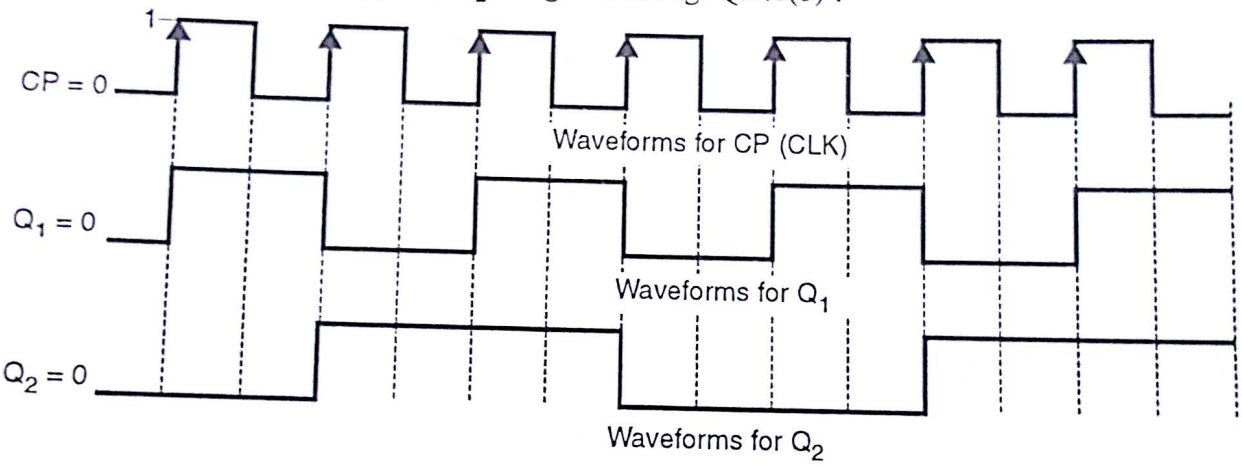



Fig. Q. 9.1 (b)

In Fig. Q. 9.1 (a) two D–type flip–flops are given. $\overline{Q}_1$ is connected to $D_1$. $Q_2$ and $Q_1$ are connected to $D_2$ through Ex–OR gate. CLK is common to both the flip–flops.

**Explanation of the waveforms :**

| Previous State | | | | Inputs | | Next State | | Decimal state |
|---|---|---|---|---|---|---|---|---|
| $Q_1$ | $\overline{Q}_1$ | $Q_2$ | $\overline{Q}_2$ | $D_1 = \overline{Q}_1$ | $D_2 = (Q_1 \oplus Q_2)$ | $Q_{1+1}$ | $Q_{2+1}$ | |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 3 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 3 |

Table 1

Scanned by CamScanner

**State Diagram :**

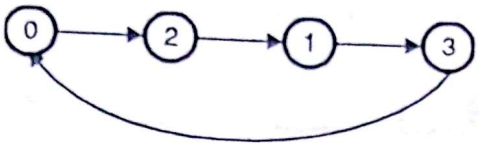



Fig. Q.9.1(c)

**2. Give difference between synchronous counter and asynchronous counter. Give merits and demerits of these counters.** (May 96, Dec. 96, 4 Marks, Dec. 97, 5 Marks)

**Ans.:** Refer Section 9.1

**3. Explain what is glitch problem as encountered in asynchronous counter. How could glitches be avoided ?** (May 96, 4 Marks)

**Ans.:** Refer Section 9.3.3.

**4. A ripple counter is required to operate at a frequency of 5MHz. If propagation delay time of each flip–flop is 20 nsec and strobe pulse width is 10 nsec. Calculate the maximum number of stages the counter can have.** (May 96, 4 Marks)

**Ans. :** $f = 5 \times 10^6$ Hz $\quad t_{pd} = 20$ nsec

$t_{sb} = 10$ nsec. $\quad N = ?$

Maximum time delay for a signal (T)

= propagation delay time of each flip-flop + strobe pulse delay

= 20 nsec + 10 nsec = 30 nsec.

$$f = \frac{1}{N \times T}$$

$$\therefore \quad N = \text{No. of flip–flops} = \frac{1}{\text{frequency} \times \text{total time delay}}$$

$$\therefore \quad N = \frac{1}{5 \times 10^6 \times 30 \times 10^{-9}} = 6$$

$\therefore$ Maximum number of stages the counter can have is 6 ...Ans.

**5. Explain what is 'lockout' in a sequential circuit. How it can be avoided?**

(May 96, May 97, Dec. 97, 4 Marks)

**Ans.:** Refer Section 9.6.

**6. For the logic circuit shown in Fig. Q. 9.6 (a) draw timing diagram of CP, $Q_2$ and $Q_1$ assuming initial condition $Q_2 = Q_1 = 0$.** (Dec. 96, 10 Marks)

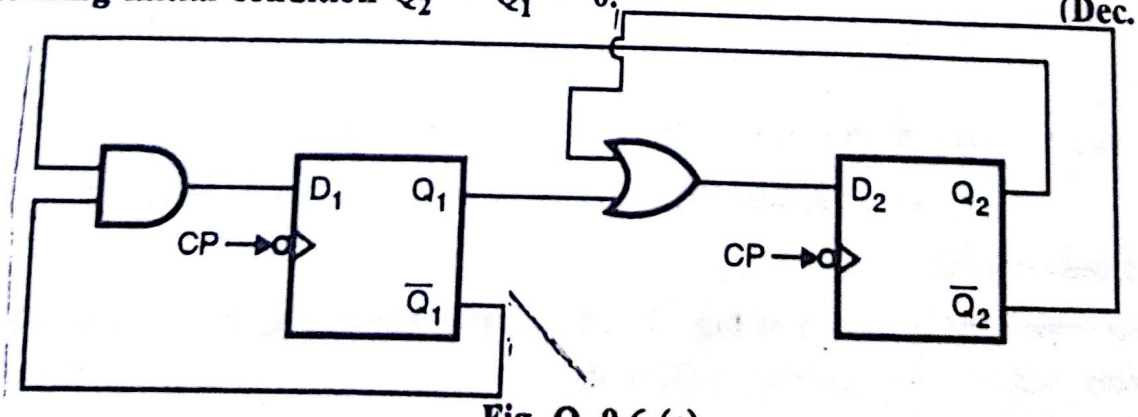



Fig. Q. 9.6 (a)

Scanned by CamScanner

**Ans. :** Waveforms for CP, $Q_1$ and $Q_2$ are as as shown in Fig. Q.9.6 (b) :

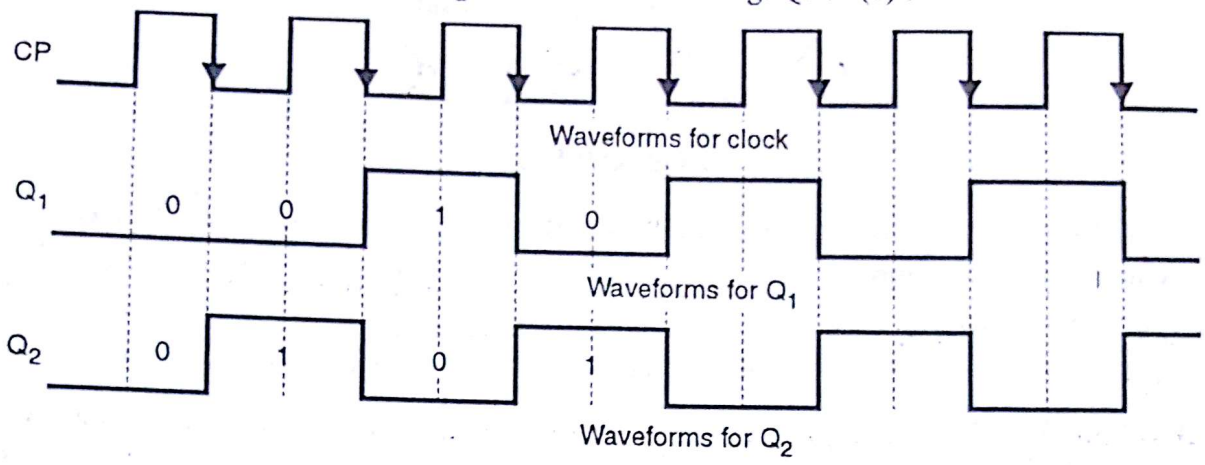



**Fig.Q. 9.6 (b)**

The waveforms are drawn according to Table 2.

| Previous State | | | | Inputs | | Next State | | Decimal state |
|---|---|---|---|---|---|---|---|---|
| $Q_1$ | $\bar{Q}_1$ | $Q_2$ | $\bar{Q}_2$ | $D_1 = (\bar{Q}_1 \cdot Q_2)$ | $D_2 = (Q_1 + \bar{Q}_2)$ | $Q_{1+1}$ | $Q_{2+1}$ | |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |

**Table 2**

**State Diagram :**

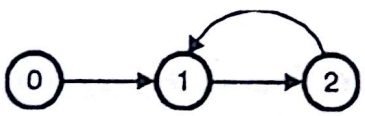



**Fig. Q.9.6(c)**

**7. What are decoding glitches & how these can be eliminated ?**

**(Dec. 96, Dec. 98, 8 Marks, May 97, June 99, 6 Marks, Dec. 99, 5 Marks)**

**Ans.:** Refer answer of Q.3.

**8. For the logic circuit shown in Fig. Q. 9.8 (a) draw the timing diagram of CP, $Q_2$ and $Q_1$ assuming initial condition $Q_2 = Q_1 = 0$.** **(May 97, 10 Marks)**

Scanned by CamScanner

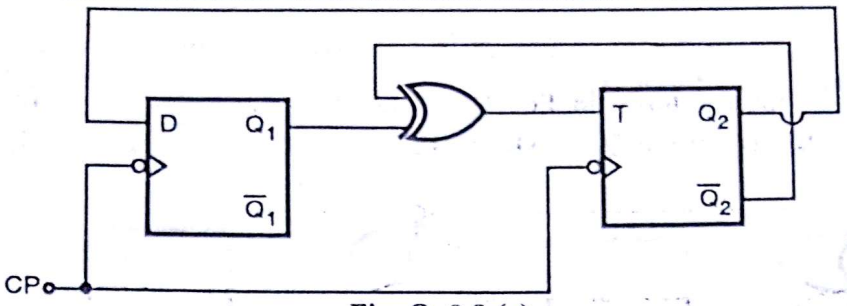



Fig. Q. 9.8 (a)

**Ans. :** In Fig. Q. 9.8 (a) two types of flip–flops are used : **D and T.**

**The waveforms for CP, $Q_1$ and $Q_2$ are as shown below :**

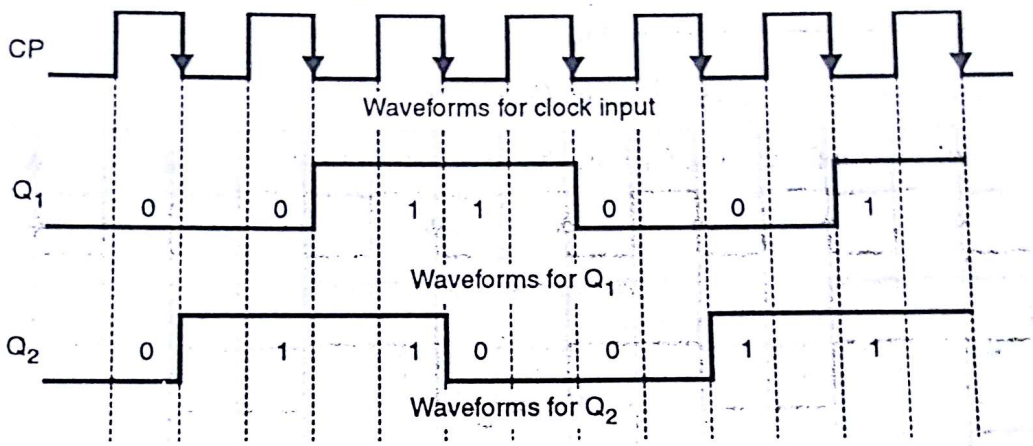



Fig. Q. 9.8 (b)

**Explanation for above waveforms :**

| Previous State | | | | Inputs | | Next State | | Decimal state |
|---|---|---|---|---|---|---|---|---|
| $Q_1$ | $\overline{Q}_1$ | $Q_2$ | $\overline{Q}_2$ | $D_1 = Q_2$ | $T = Q_1 \oplus \overline{Q}_2$ | $Q_{1+1}$ | $Q_{2+1}$ | |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 3 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 3 |

Table 3

**State Diagram :**

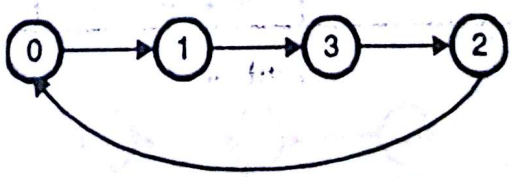



Fig. Q.9.8(c)

Scanned by CamScanner

9. For the logic circuit shown in Fig. Q. 9.9 (a) draw the timing diagram of CP, $Q_2$, $Q_1$ assuming initial condition $Q_2 = Q_1 = '1'$. (Dec. 97, 10 Marks)

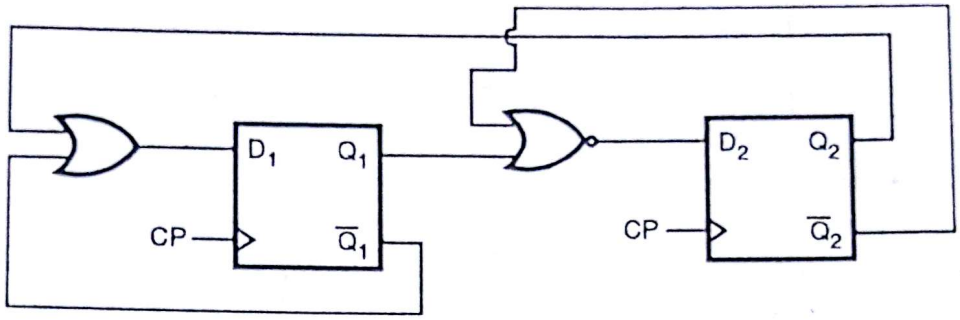



Fig. Q. 9.9 (a)

Ans. : Timing diagram for $C_p$, $Q_2$, $Q_1$ is as shown in Fig. Q.9.9 (b).

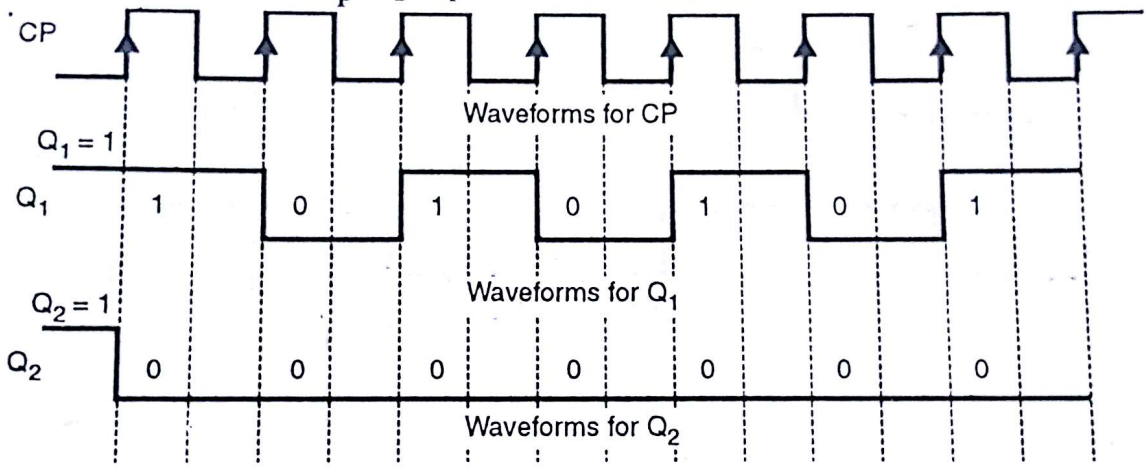



Fig. Q. 9.9 (b)

**Explanation for waveforms :**

| Previous State | | | | Inputs | | Next State | | Decimal state |
|---|---|---|---|---|---|---|---|---|
| $Q_1$ | $\bar{Q}_1$ | $Q_2$ | $\bar{Q}_2$ | $D_1 = \bar{Q}_1 + Q_2$ | $D_2 = \overline{Q_1 + \bar{Q}_2}$ | $Q_{1+1}$ | $Q_{2+1}$ | |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 2 |

Table 4

**State Diagram :**

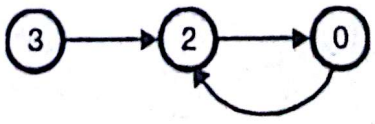



Fig. Q.9.9(c)

Scanned by CamScanner

**10. What is 'glitch' as observed in counter decoder scheme ? How it can be eliminated ? In what way glitch is harmful in a logic system ?** **(Dec. 97, 6 Marks)**

**Ans.:** Refer answer of Q.3.

**11. A sequential circuit employing two D–type flip–flops is shown in Fig. Q. 9.16. Analyse this circuit and write equations for flip–flop outputs and circuit output. Hence derive all possible entries of present state/next state table for this circuit.** **(May 98, 10 Marks)**

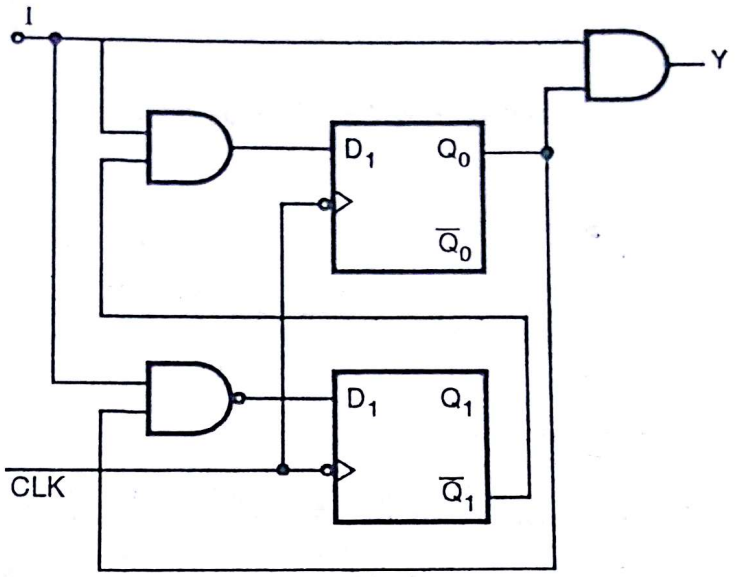



**Fig. Q. 9.11**

**Ans. :** First we will write present state/next state table for this circuit.

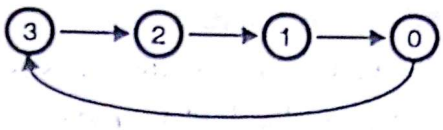
| Present State | | | | | Decimal | I/P to D FF | | Next State | | | | Decimal | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | $\bar{Q}_1$ | $Q_1$ | $\bar{Q}_0$ | $Q_0$ | | $D_1 = I\bar{Q}_0$ | $D_0 = I\bar{Q}_1$ | $\overline{Q_{1+1}}$ | $Q_{1+1}$ | $\overline{Q_{0+1}}$ | $Q_{0+1}$ | | $I \cdot Q_0$ |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 2 | 0 |
| 0 | 0 | 1 | 1 | 0 | 2 | 1 | 0 | 0 | 1 | 1 | 0 | 2 | 0 |
| 0 | 0 | 1 | 0 | 1 | 3 | 1 | 0 | 0 | 1 | 1 | 0 | 2 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 3 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 2 | 1 | 0 | 0 | 1 | 1 | 0 | 2 | 0 |
| 1 | 0 | 1 | 0 | 1 | 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

**Table 5**

**State diagram :** (i) If I = 0, the counter willl lockout in state 2.

(ii) If I = 1 the states of counter are as shown in state diagram below :

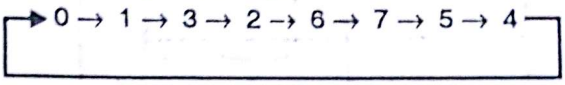
Scanned by CamScanner




**12. Using J–K flip–flops realise a three bit Grey Code counter. Use K-maps for reduction.**

**(May 98, 10 Marks)**

**Ans. :** As it is a 3 bit Grey Code counter, it will go through states




As it is 3 bit counter, we will use three J–K flip–flops.

The following table gives present state/next state for 3 bit grey code counter.

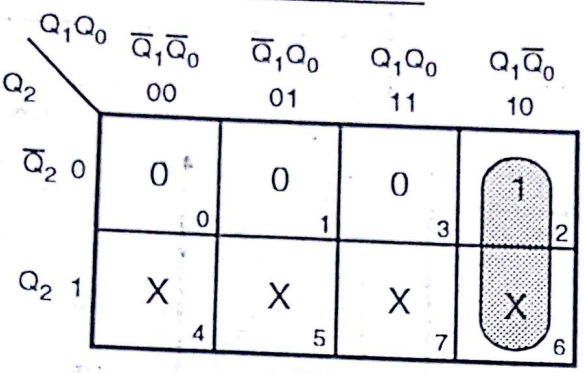
| Present state | | | Inputs | | | | | | Next states | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_{2n}$ | $Q_{1n}$ | $Q_{0n}$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ | $J_0$ | $K_0$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| 0 | 0 | 0 ($m_0$) | 0 | X | 0 | X | 1 | X | 0 | 0 | 1 |
| 0 | 0 | 1 ($m_1$) | 0 | X | 1 | X | X | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 ($m_3$) | 0 | X | X | 0 | X | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 ($m_2$) | 1 | X | X | 0 | 0 | X | 1 | 1 | 0 |
| 1 | 1 | 0 ($m_6$) | X | 0 | X | 0 | 1 | X | 1 | 1 | 1 |
| 1 | 1 | 1 ($m_7$) | X | 0 | X | 1 | X | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 ($m_5$) | X | 0 | 0 | X | X | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 ($m_4$) | X | 1 | 0 | X | 0 | X | 0 | 0 | 0 |

**Table 6**

*Using Table 2 we have to realise grey code counter.* **For reduction we will use K–maps.**

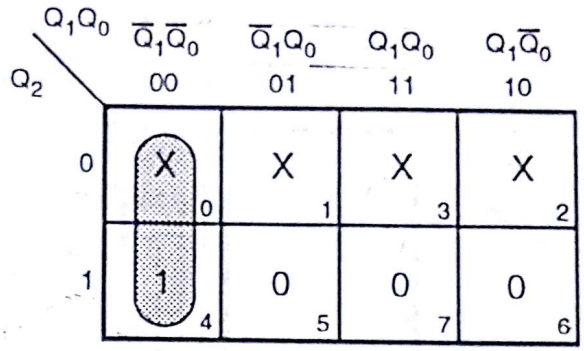
Scanned by CamScanner

**K- map for $J_2$**



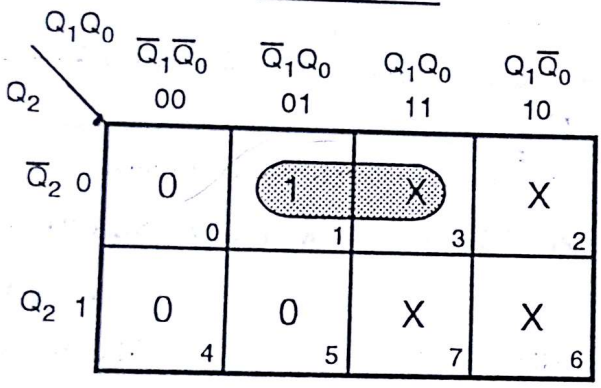

$J_2 = Q_1 \cdot \overline{Q}_0$

**K- map for $K_2$**



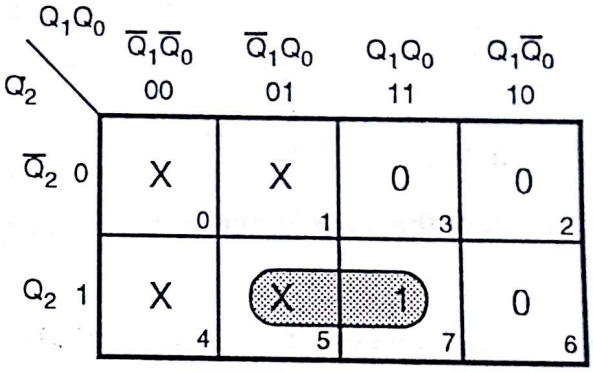

$K_2 = \overline{Q}_1 \overline{Q}_0$

**K- map for $J_1$**



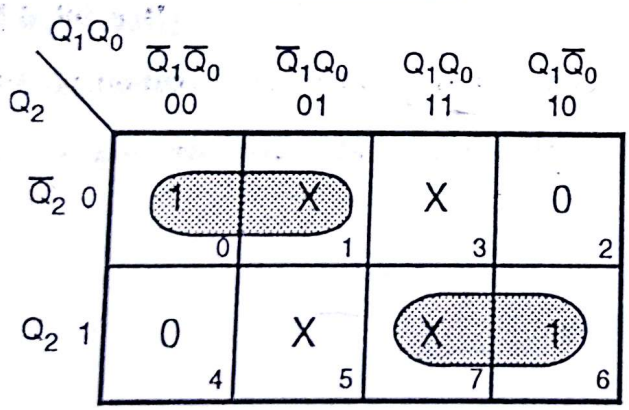

$J_1 = \overline{Q}_2 Q_0$

**K- map for $K_1$**




$K_1 = Q_2 Q_0$

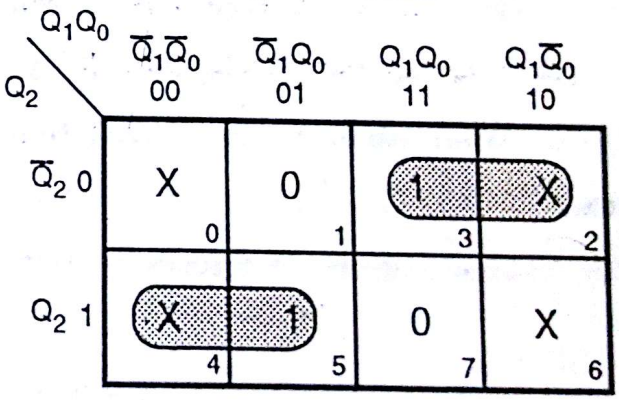
**K- map for $J_0$**




$J_0 = \overline{Q}_2 \overline{Q}_1 + Q_2 Q_1$

$= \overline{Q_1 \oplus Q_2}$

**K- map for $K_0$**




$K_0 = Q_2 \overline{Q}_1 + \overline{Q}_2 Q_1$

$= Q_1 \oplus Q_2$

Scanned by CamScanner

**Circuit Diagram for 3 bit Grey code counter using JK flip–flop :**

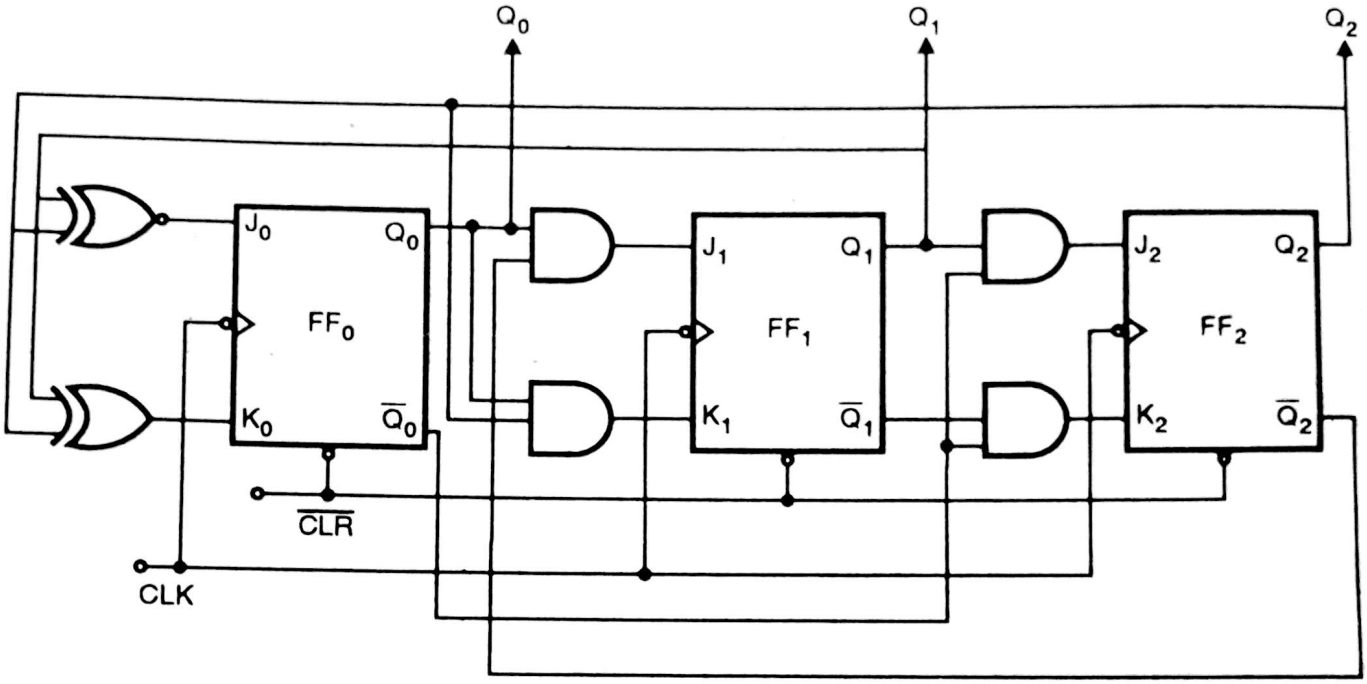



**Fig. Q. 9.12**

**13. Explain the term 'glitch' w.r.t. a ripple counter. Explain drawing waveforms.**

**(May 98, 4 Marks)**

**Ans.:** Refer Section 9.3.3.

**14. On what factors the speed of a ripple counter depends ?** **(May 98, 3 Marks)**

**Ans.:** Speed of operation of a ripple counter depends upon "propagation delay" of FF and clock frequency. If frequency is more and propagation delay is large, we come across invalid states.

**15. (a) Obtain a mod–6 asynchronous counter.** **(Dec. 98, 6 Marks)**

**(b) Give the decoding logic required.** **(Dec. 98, 3 Marks)**

**(c) What will be maximum clock frequency for the above mod–6 asynchronous counter?**

**Ans. :**

**Step 1 :** State diagram = 6 states $\Rightarrow$ 0 to 5

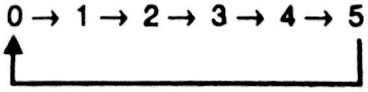



**Fig. Q. 9.15**

**No. of states N = 6 $\therefore$ n < N = $2^m$**

$$6 < 2^m$$

$\therefore$ $\boxed{m = 3}$ → No. of flip–flops

Scanned by CamScanner

**Step 2 : For design of reset circuit :**

Truth table is,

$N = 2^m = 2^3 = 8$

But $n = 6$

0 to 5 are valid states. So output will be one for 0 to 5.

6 and 7 are invalid states. So output will be zero for 6 and 7 states.

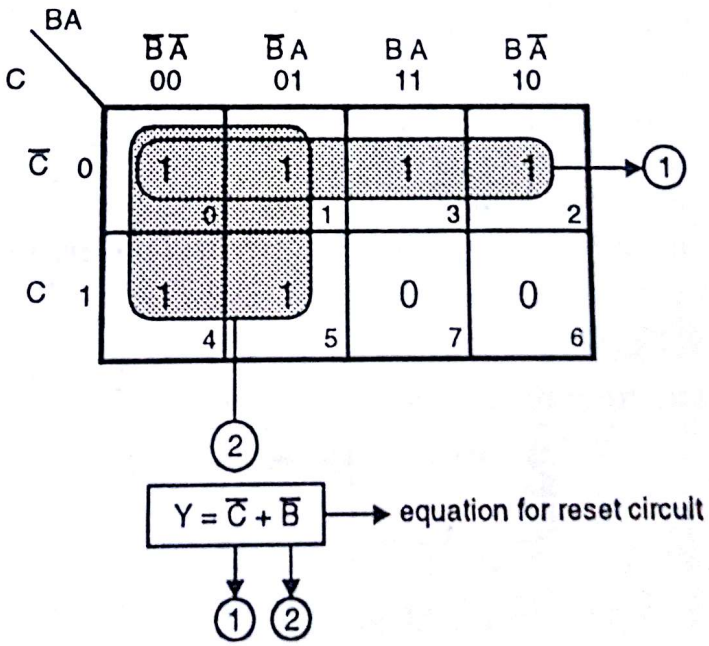
Truth table is as given in Table 7.

| | C | B | A | Y O/P |
|---|---|---|---|---|
| Valid | 0 | 0 | 0 | 1 |
| | 0 | 0 | 1 | 1 |
| | 0 | 1 | 0 | 1 |
| | 0 | 1 | 1 | 1 |
| | 1 | 0 | 0 | 1 |
| | 1 | 0 | 1 | 1 |
| Invalid | 1 | 1 | 0 | 0 |
| | 1 | 1 | 1 | 0 |

**Table 7**

**Step 3 :** First we will make K–map for above circuit :

**K- map**




**Points :** (1) Counter is up counter. Therefore we have to cascade Q output negative edge triggered filp–flop and final output from Q.

(2) Flip–flops should have active low reset (clear) terminal.

(3) Propagation delay presently assumed 0 nsec.

**Step 4 : Circuit Diagram for mod–6 asynchronous counter is as follows :**

Scanned by CamScanner

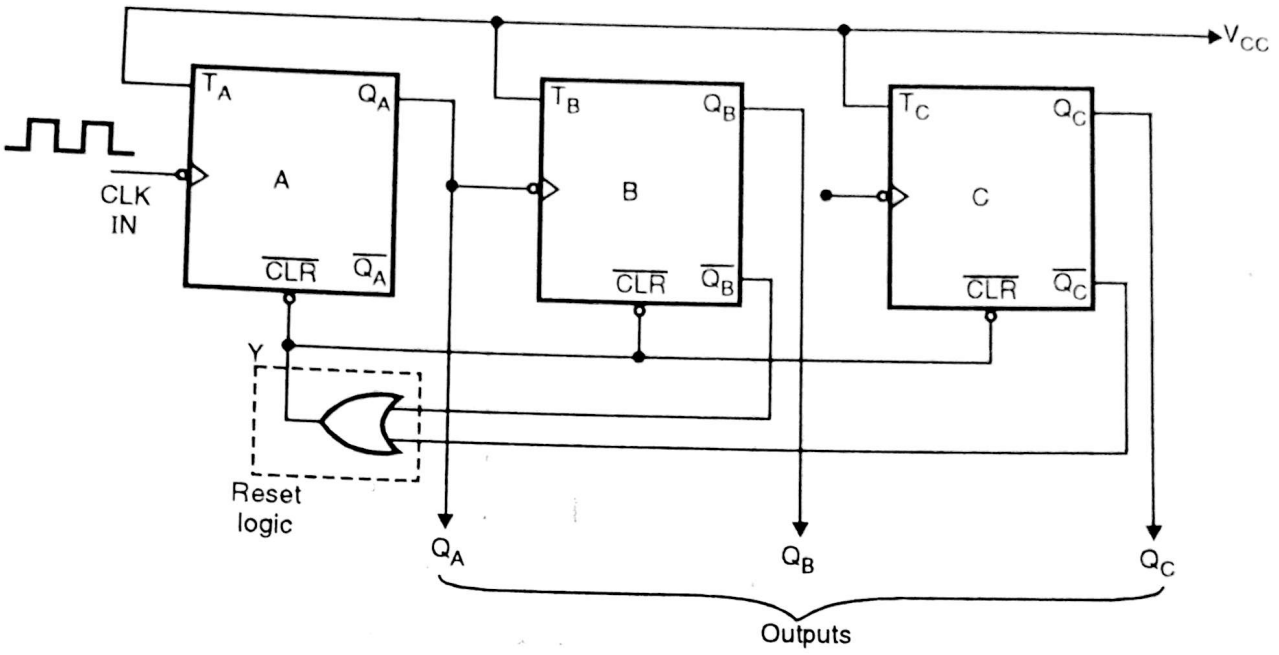



**Fig. Q. 9.15(a)**

As it is mod–6 counter, it has 0 to 5 states. So when counter reaches 110 state; input for OR gate (Reset logic) is $\overline{C}$ and $\overline{B}$; that is 00, giving output zero. This zero output is given as $\overline{CLR}$ to all counters, which clears all the counters giving 000 state.

(c) Time period of CLK $(T_{CLK}) \geq N \times t_{pd}$

$t_{pd}$ = propogative delay

N = No. of FFs

$\therefore$ Maximum clock frequency $f_{max} = \frac{1}{CLK} = \frac{1}{Nt_{pd}}$ **...Ans.**

**16. Design a 3–bit counter with input control signal E. The counter will undergo the steps 0–2–4–6–0 if control signal is held high. The counter will undergo steps 1–3–5–7–1 if control signal is held low. (June 99, 16 Marks)**

**Ans.:** Refer Section 9.9, Ex. 18.

**17. Design ripple counter for state diagram shown in Fig. 9.17. (June 99, 10 Marks)**

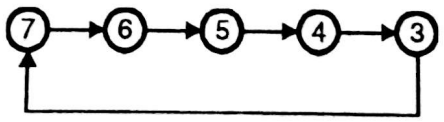



**Fig. 9.17**

**Ans.:** Refer Section 9.3.2, Ex. 3.

**18. For asynchronous sequence counter with sequence –2–6–5–3–1–0–2. (June 99, 16 Marks)**

**(a) Give present state/next state table.**

**(b) Set up excitation table for input of FF, (use D FF).**

**(c) Construct K–maps for excitation inputs of flip-flop, simplify and realize the circuit.**

Scanned by CamScanner

(d) If the counter enters any unused state will it go to lockout condition?

**Ans. :** Given sequence is,

$2 \rightarrow 6 \rightarrow 5 \rightarrow 3 \rightarrow 1 \rightarrow 0 \rightarrow 2$

First we will write present state/next state table according to sequence.

| | Present state | | | Next state | | |
|---|---|---|---|---|---|---|
| | $Q_{2n}$ | $Q_{1n}$ | $Q_{0n}$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| Valid states | 0 | 1 | 0 | 1 | 1 | 0 |
| | 1 | 1 | 0 | 1 | 0 | 1 |
| | 1 | 0 | 1 | 0 | 1 | 1 |
| | 0 | 1 | 1 | 0 | 0 | 1 |
| | 0 | 0 | 1 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 | 0 |
| Invalid states | 1 | 0 | 0 | X | X | X |
| | 1 | 1 | 1 | X | X | X |

**Table 8**

States 4 and 7 are unused states. If counter enters any unused state we will write don't care for next state.

So for 100 and 111 we will write don't care.

(b) From Table 8 we can construct excitation table for input of flip–flop.

Here we are using D flip–flop.

As there are 6 used states, so value of N = 3

So we are using 3 D type flip–flops.

**Excitation table :**

| | Present state | | | D inputs | | | Next state | | |
|---|---|---|---|---|---|---|---|---|---|
| | $Q_{2n}$ | $Q_{1n}$ | $Q_{0n}$ | $D_2$ | $D_1$ | $D_0$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| Used states | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| Unused states | 1 | 0 | 0 | X | X | X | X | X | X |
| | 1 | 1 | 1 | X | X | X | X | X | X |

**Table 9**

(c) From Table 9 we have to construct K–maps for excitation inputs of D flip–flop and design the circuit.

Scanned by CamScanner

**K- map for $D_2$**

| $Q_2$ \ $Q_1Q_0$ | $\bar{Q}_1\bar{Q}_0$ 00 | $\bar{Q}_1Q_0$ 01 | $Q_1Q_0$ 11 | $Q_1\bar{Q}_0$ 10 |
|---|---|---|---|---|
| $\bar{Q}_2$ 0 | 0 (0) | 0 (1) | 0 (3) | 1 (2) |
| $Q_2$ 1 | X (4) | 0 (5) | X (7) | 1 (6) |

$$D_2 = Q_1\bar{Q}_0$$

**K- map for $D_1$**

| $Q_2$ \ $Q_1Q_0$ | $\bar{Q}_1\bar{Q}_0$ 00 | $\bar{Q}_1Q_0$ 01 | $Q_1Q_0$ 11 | $Q_1\bar{Q}_0$ 10 |
|---|---|---|---|---|
| $\bar{Q}_2$ 0 | 1 (0) | 0 (1) | 0 (3) | 1 (2) |
| $Q_2$ 1 | X (4) | 1 (5) | X (7) | 0 (6) |

$$D_1 = \bar{Q}_2\bar{Q}_0 + \bar{Q}_1\bar{Q}_0 + Q_2Q_0$$

$$= \bar{Q}_2\bar{Q}_0 + Q_2Q_0 + \bar{Q}_1\bar{Q}_0$$

$$D_1 = \overline{Q_2 \oplus Q_0} + \bar{Q}_1\bar{Q}_0$$

**K- map for $D_0$**

| $Q_2$ \ $Q_1Q_0$ | $\bar{Q}_1\bar{Q}_0$ 00 | $\bar{Q}_1Q_0$ 01 | $Q_1Q_0$ 11 | $Q_1\bar{Q}_0$ 10 |
|---|---|---|---|---|
| $\bar{Q}_2$ 0 | 0 (0) | 0 (1) | 1 (3) | 0 (2) |
| $Q_2$ 1 | X (4) | 1 (5) | X (7) | 1 (6) |

$$D_0 = Q_2 + Q_1Q_0$$

We have to design the circuit according to $D_2$, $D_1$ and $D_0$.
We are using negative edge triggered flip–flop.

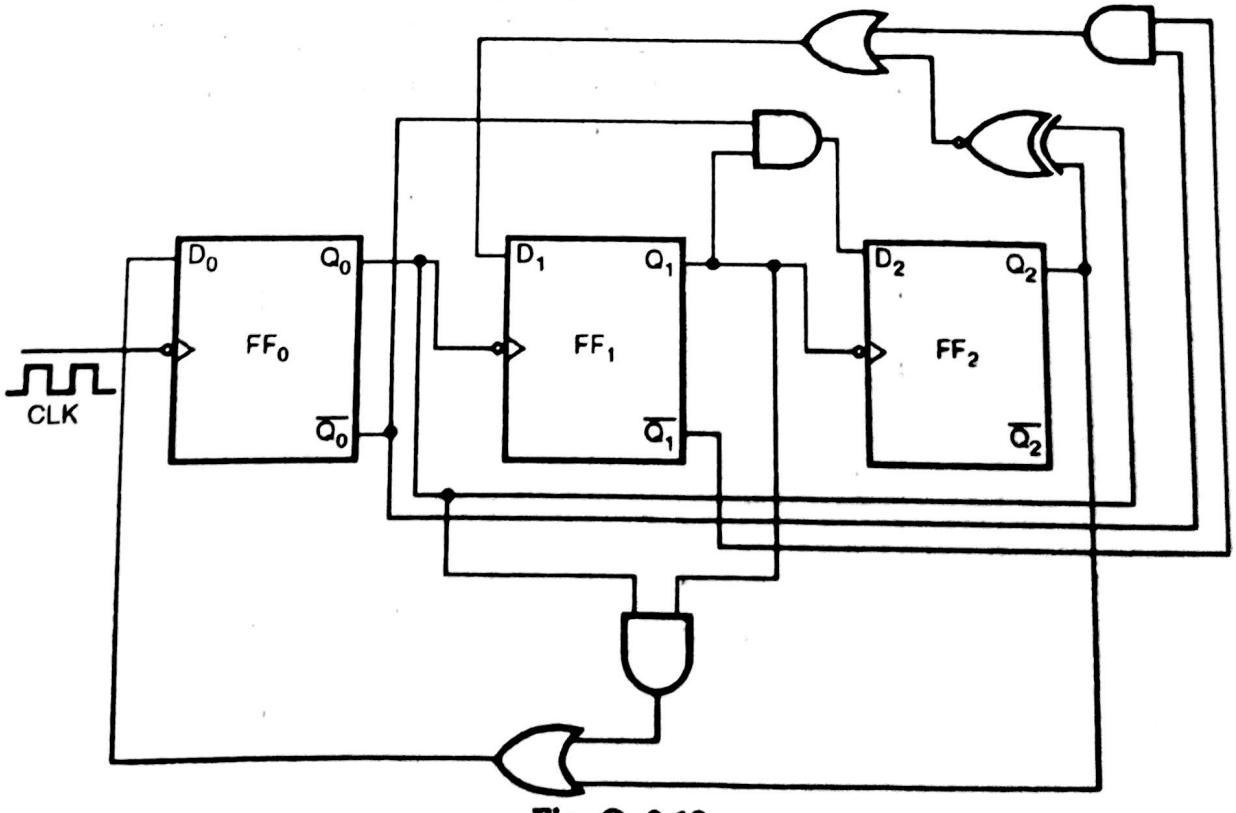



**Fig. Q. 9.18**

Scanned by CamScanner

(d) In above sequence 4 and 7 are invalid states. If the counter enters states 4 and 7, we will check from circuit diagram whether it will go to lockout condition or not.

| Present state | | | Input of D FF | | | Next State | | |
|---|---|---|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $D_2$ | $D_1$ | $D_0$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |

**Table 10**

From Table 10 we can conclude that if counter enters state 4, next state is 1 and if it enters state 7, next state is 3. So counter go back to valid states. So it will not go to lockout condition.

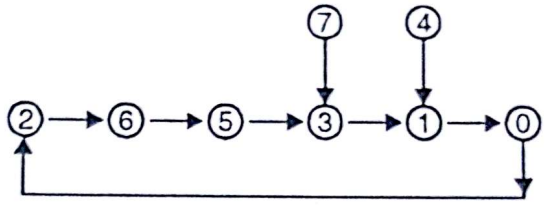



**Fig. Q. 9.18 (a)**

❑❑❑

Scanned by CamScanner

# Shift Register

10

**Syllabus :**

Serial–in–Serial out register, shift register operations.

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Register [Parallel and shift register] | 10.1 | ✓ | – |
| • Serial Shifting | 10.2 | ✓ | – |
| • Serial in – Serial out (SISO) | 10.3 | ✓ | – |
| • Serial in – Parallel out | 10.4 | ✓ | – |
| • Parallel in – Parallel out | 10.5 | ✓ | – |
| • Ring Counter | 10.6 | ✓ | – |
| • Johnson Counter (Twisted Ring Counter) | 10.7 | ✓ | – |
| • Applications of Shift Register | 10.8 | – | ✓ |
| • Pseudo Random Binary Sequence (PRBS) Generator | 10.9 | ✓ | – |
| • IC 7495 | 10.10 | – | ✓ |
| • Analysis of Shift Counter | 10.11 | – | ✓ |



**Comments**

- Section 10.1 / 10.2 / 10.6 / 10.7 / 10.10 / 10.11 to be concentrated more.
- Most widely used in microprocessor.

Scanned by CamScanner

## 10.1 Register [Parallel and Shift register] :

**Introduction**

Till this point, we have studied different flip-flops and conversion from one to another. Now, let's concentrate on application part of flip-flops. The most common use of FF is a simple ***Register*** (OR a n bit memory storage device). In next section, we will start with basic definition of register and slowly move on to shift register.

Register is a group of memory element that work together as a unit. Register simply stores a binary word. When register accepts parallel data and outputs parallel data, the same is referred as ***Parallel register or buffer register.*** About shift register, it is nothing but memory element with facility of shifting left and right, bit by bit. Before going deep into shift register, let's see structure of buffer register.

### 10.1.1 Buffer Register :

Buffer register is simplest kind of register and stores digital word. The structure of buffer register, built with the edge triggered D type FF is shown in Fig. 10.1.

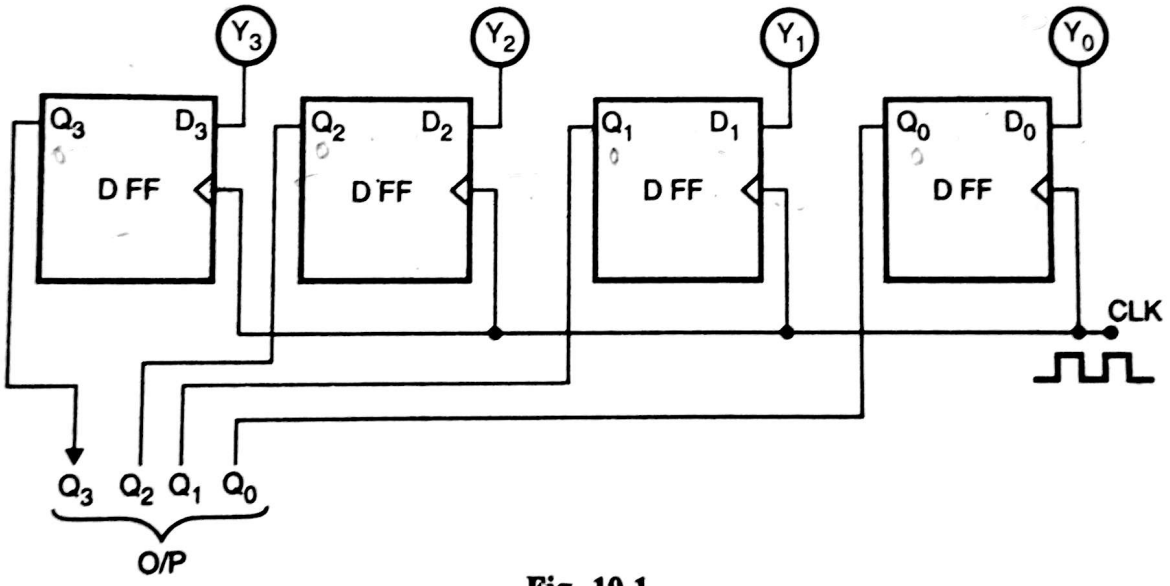



**Fig. 10.1**

**Working :**

(1) Initially, consider $Q_3\ Q_2\ Q_1\ Q_0 = 0\ 0\ 0\ 0$

(2) Apply data to be stored at input terminals $Y_3 - Y_o$ . Let's say $Y_3\ Y_2\ Y_1\ Y_o = 1010$.

(3) Apply clock pulse.

(4) When first clock edge arrives, $Y_3\ Y_2\ Y_1\ Y_o = Q_3 Q_2\ Q_1\ Q_o = 1010$

(5) Even though if now $Y_3\ Y_2\ Y_1\ Y_o = 0\ 0\ 0\ 0$, $Q_3 Q_2\ Q_1\ Q_0 = 1010$ i.e. latched.

**Conclusion :**

From Fig. 10.1 and WORKING we conclude that,

(1) There must be one FF for each bit in binary number. ∴ for 4 bit we require 4 FFs.

(2) Simultaneously $Y_3 - Y_o$ was applied and loaded to $Q_3 - Q_0$ by single clock pulse. i.e. at a time $Q_3$ to $Q_0$ was loaded.

(3) Therefore, the way we applied input and taken output, is called ***parallel in – parallel out*** (PIPO). The block schematic representation is shown in Fig. 10.2.

Scanned by CamScanner

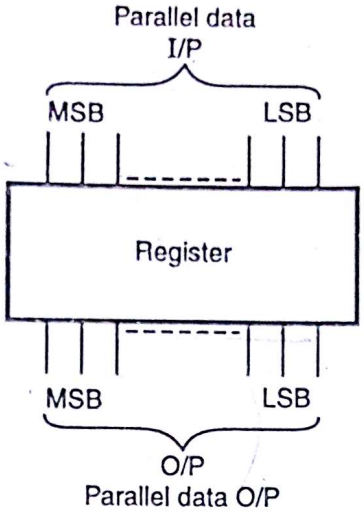



**Fig. 10.2**

This is also referred as ***Parallel Shifting***.

## 10.2 Serial Shifting :

One more method of loading or shifting the data is called ***Serial Shifting***.

***What do you mean by serial ?***

Serial means bit by bit data flow, serially, on single line. Serial shifting has only single bit data line, not 4 or 8 as in parallel loading. The serial shifting can be represented block schematically, as shown in Fig. 10.3.

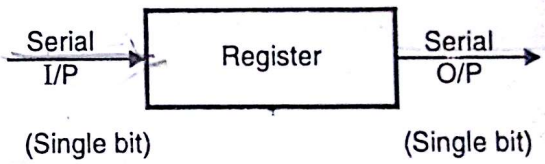
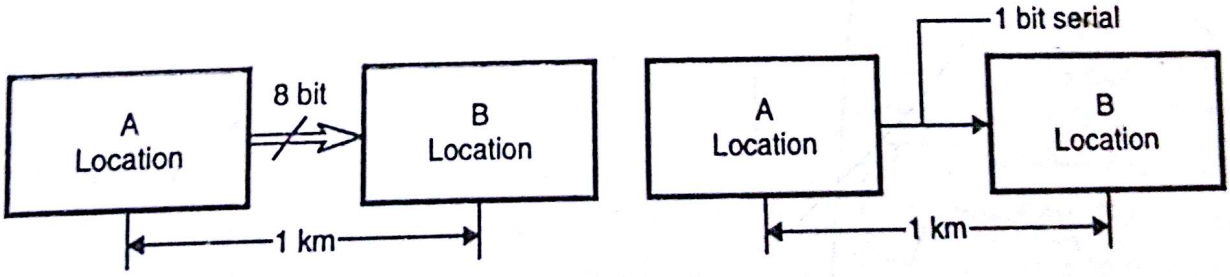



**Fig. 10.3**

Single bit data is entered in register and serially single data bit is taken out, through register. A group of flip-flops connected to provide serial out, when serial input is given, is called ***Shift register.***

Basically if you see, loading parallel data is much faster than the serial. Secondly, normally we work with 8 bit, 16 bit, and 32 bit parallel data bits. ***So one can ask, why you think of serial data*** ? The answer for the question is very simple. If you see, uptill now we are looking of loading the registers, but now think of ***loading or transmitting data*** at distant end. Fig. 10.4 shows parallel and serial data transmission.




**(a) Parallel data transmission** **(b) Serial data transmission**

**Fig. 10.4**

Scanned by CamScanner

As shown in Fig. 10.4 parallel transmission requires 8 lines, serial requires single line. Now think of distance between A and B location is in meters and kilometers. At such a large distance just increasing length of the wire is not the solutions because now wire is Transmission line, so reflection, impedance matching, power required etc. parameter, we have to consider. Increasing power will increase total power consumption of the system. In parallel, power for 8 lines have to be increased, ∴ Power consumed is much, in comparision with single line of serial. ∴ Normally serial transmission is preferred over parallel.

At this point you must be eager to know how serial data looks like. It is shown in Fig. 10.5.

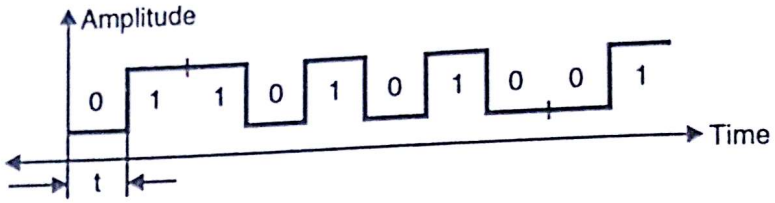



**Fig. 10.5 : Format of Serial data**

As shown in Fig. 10.5, data is represented on time axis is bit by bit, not full 8 bit or 16 bit as in parallel. Time 't' for each bit is same. Data stream shown is ... 0 , 1, 1, 0, 1, 0, 1, 0, 0, 1 ....

Therefore serial communication has become common and most widely used media for communication. But our systems are normally dealing with parallel (8 bit, 16 bit and soon), ∴ one has to convert parallel to serial. This function is also called as **parallel input and serial output** (PISO).

Serial data is now transmitted.to location B. On receiving side (B side), one has to convert serial data back to parallel. This function of conversion is called **serial in and parallel out** (SIPO). Finally one may require **serial in and serial out** (SISO). Fig. 10.6 shows generalised block schematic of four functions.

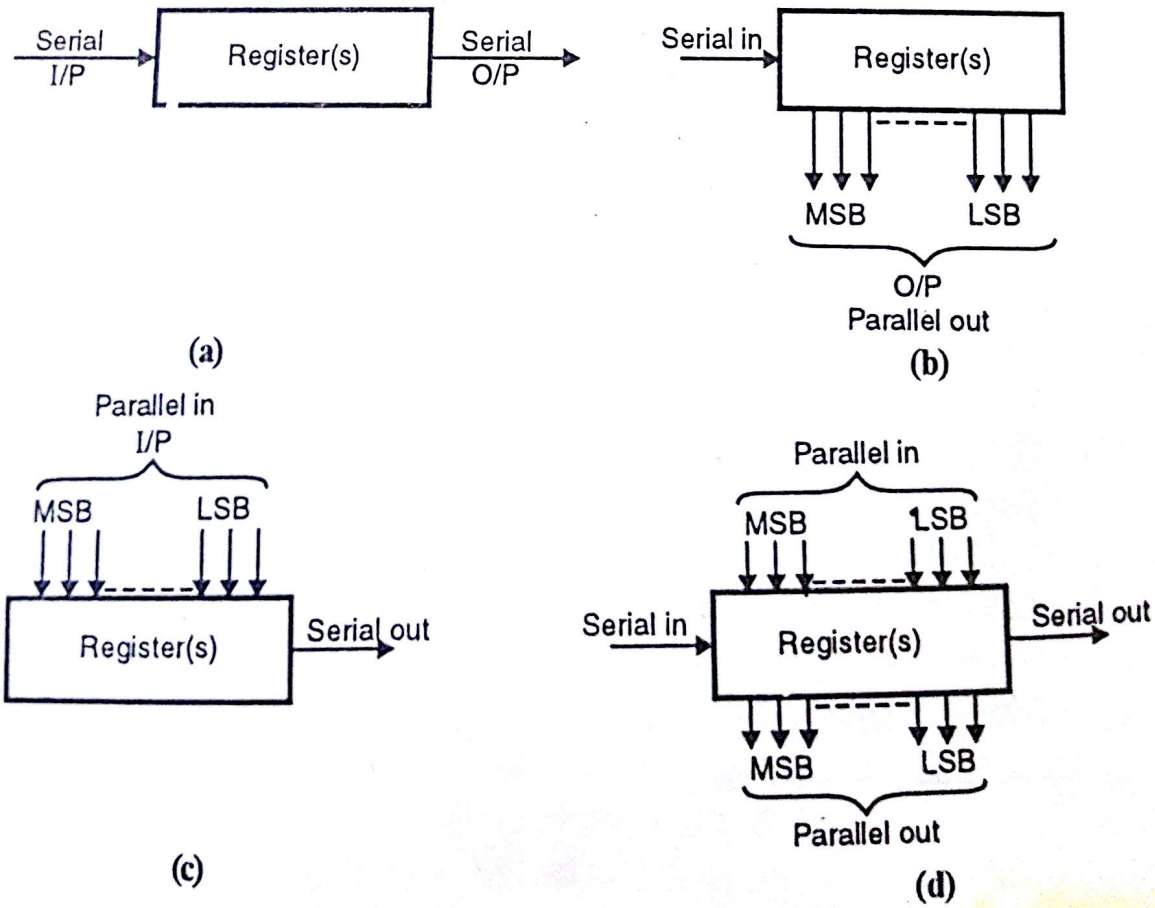



**Fig. 10.6**

Scanned by CamScanner

## 10.3 Serial in – Serial out (SISO) :

In SISO, two main operations are performed, serial left shifting and serial right shifting.

**Serial Left Shift**

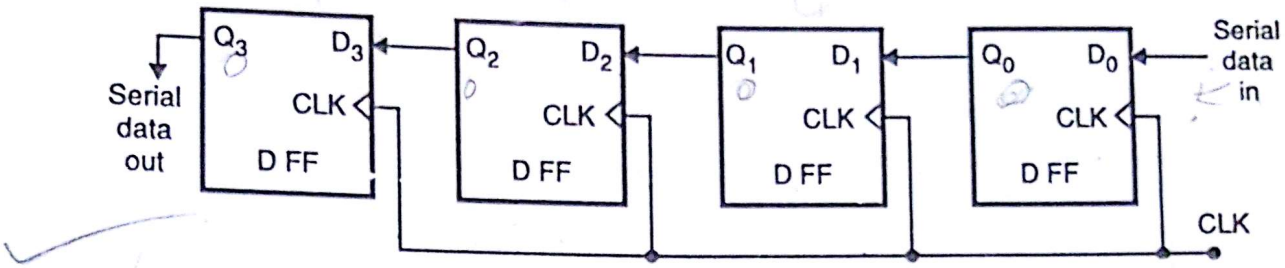



**Fig. 10.7**

Fig. 10.7 shows serial operation, built using D Flip-Flop. To know data shifts left we temporarily take output of $Q_2$ $Q_1$ and $Q_0$ also.

**Working :**

(1) Initially all FFs are reset $\therefore$ $Q_3\ Q_2\ Q_1\ Q_0 = 0000$

(2) Make Data IN = 1.

(a) At first CLOCK edge, output will be $Q_3\ Q_2\ Q_1\ Q_0 = 0001$
(Refer truth table of D FF).

(b) At 2nd CLOCK edge, output will be $Q_3\ Q_2\ Q_1\ Q_0 = 0011$

(c) At 3rd CLOCK edge, output will be $Q_3\ Q_2\ Q_1\ Q_0 = 0111$

(d) At 4th CLOCK edge, output will be $Q_3\ Q_2\ Q_1\ Q_0 = 1111$

(3) Now the Data IN = 0.

(e) At 5th CLOCK edge, output will be $Q_3\ Q_2\ Q_1\ Q_0 = 1110$

(f) At 6th CLOCK edge, output will be $Q_3\ Q_2\ Q_1\ Q_0 = 1100$

(g) At 7th CLOCK edge, output will be $Q_3\ Q_2\ Q_1\ Q_0 = 1000$

(h) At 8th CLOCK edge, output will be $Q_3\ Q_2\ Q_1\ Q_0 = 0000$

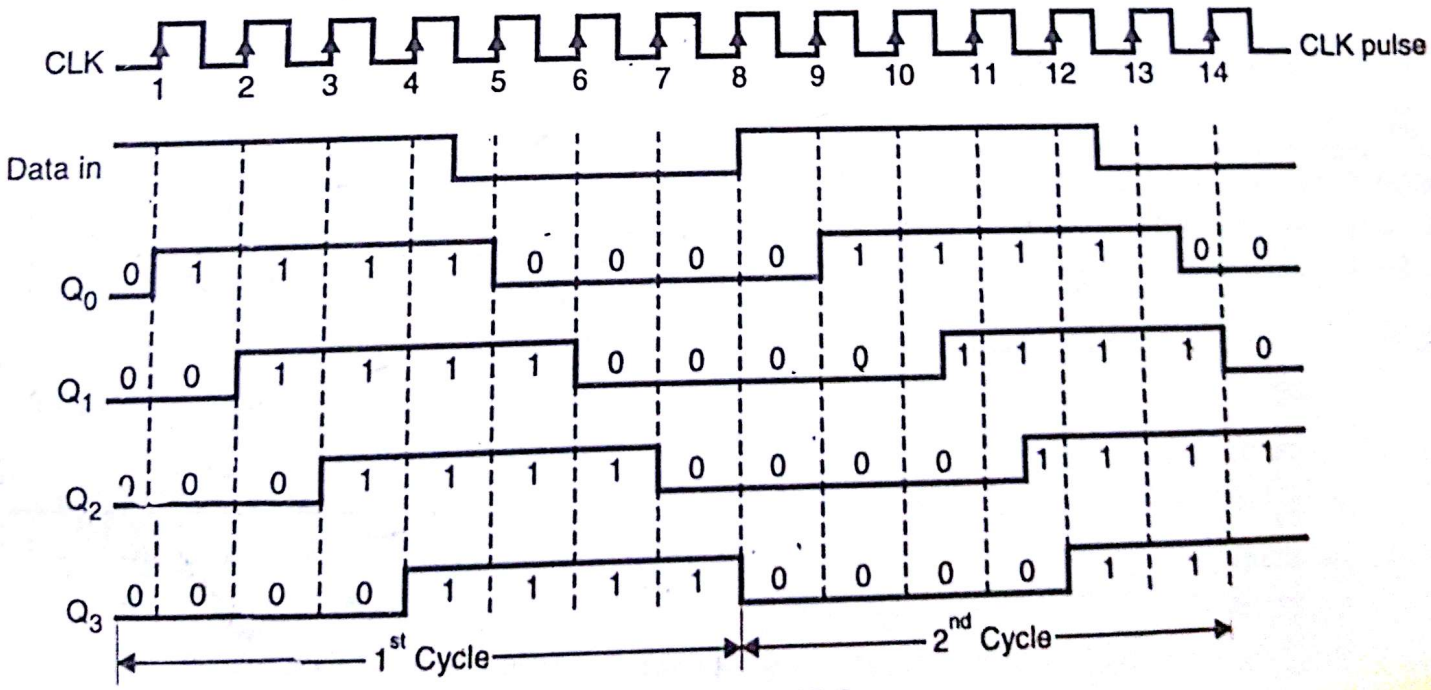



**Fig. 10.8**

Scanned by CamScanner

If you observe, 1 or 0 travels from right hand to left hand, ∴ called ***left shifting***. Let's observe waveforms for the same. Waveforms are self explanatory. Refer Fig. 10.8.

**Shift Right**

Fig. 10.9 shows serial right circuit, built using D FF.

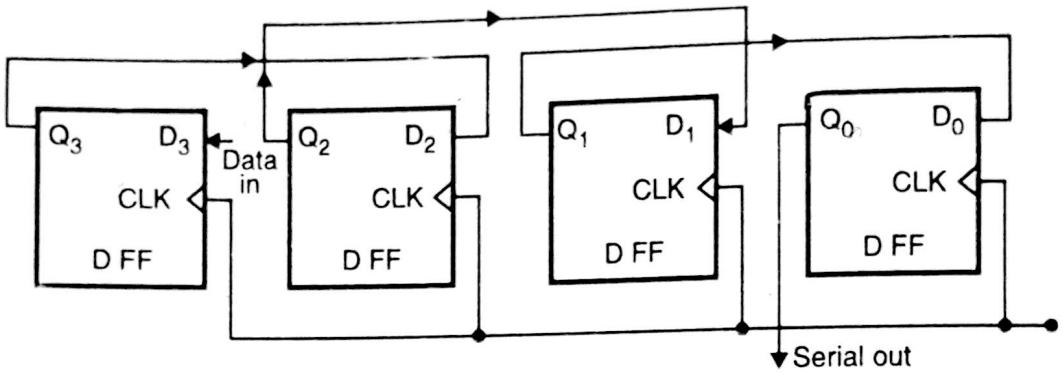



**Fig. 10.9**

**Working :**

(1) Initially all FFs are cleared. ∴ $Q_3\ Q_2\ Q_1\ Q_o = 0000$

(2) Make Data In = 1

(a) 1st CLOCK edge, output will be, $Q_3\ Q_2\ Q_1\ Q_0 = 1000$

(b) 2nd CLOCK edge, output will be, $Q_3\ Q_2\ Q_1\ Q_0 = 1100$

(c) 3rd CLOCK edge, output will be, $Q_3\ Q_2\ Q_1\ Q_0 = 1110$

(d) 4thCLOCK edge, output will be, $Q_3\ Q_2\ Q_1\ Q_0 = 1111.$

(3) Now apply Data In = 0 .

(e) 5th CLOCK edge, output will be, $Q_3\ Q_2\ Q_1\ Q_0 = 0111$

(f) 6thCLOCK edge, output will be, $Q_3\ Q_2Q_1\ Q_0 = 0011$

(g) 7th CLOCK edge, output will be, $Q_3\ Q_2\ Q_1\ Q_0 = 0001$

(h) 8th CLOCK edge, output will be, $Q_3\ Q_2Q_1\ Q_0 = 0000.$

Thus 1 and 0, travel from left to right, ∴ right shifting.

## 10.4 Serial in–Parallel out :

In this function, data is entered serially and taken out in parallel To take data parallely out, it is necessary to have all data bits available as output, at the same time and therefore output of each FF is taken.

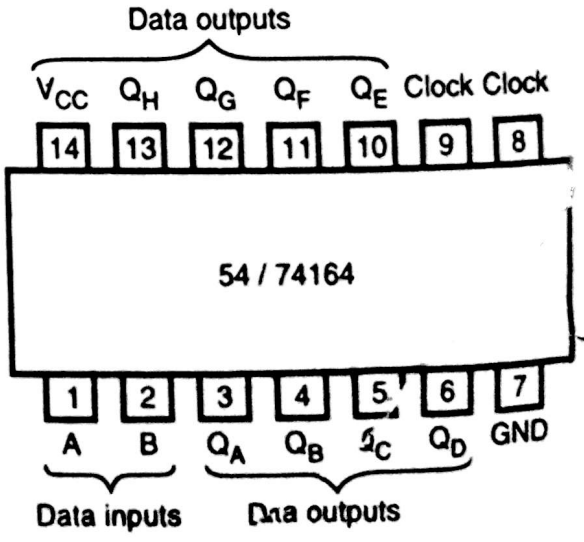



**Fig. 10.10 : DIP pin out**

Scanned by CamScanner

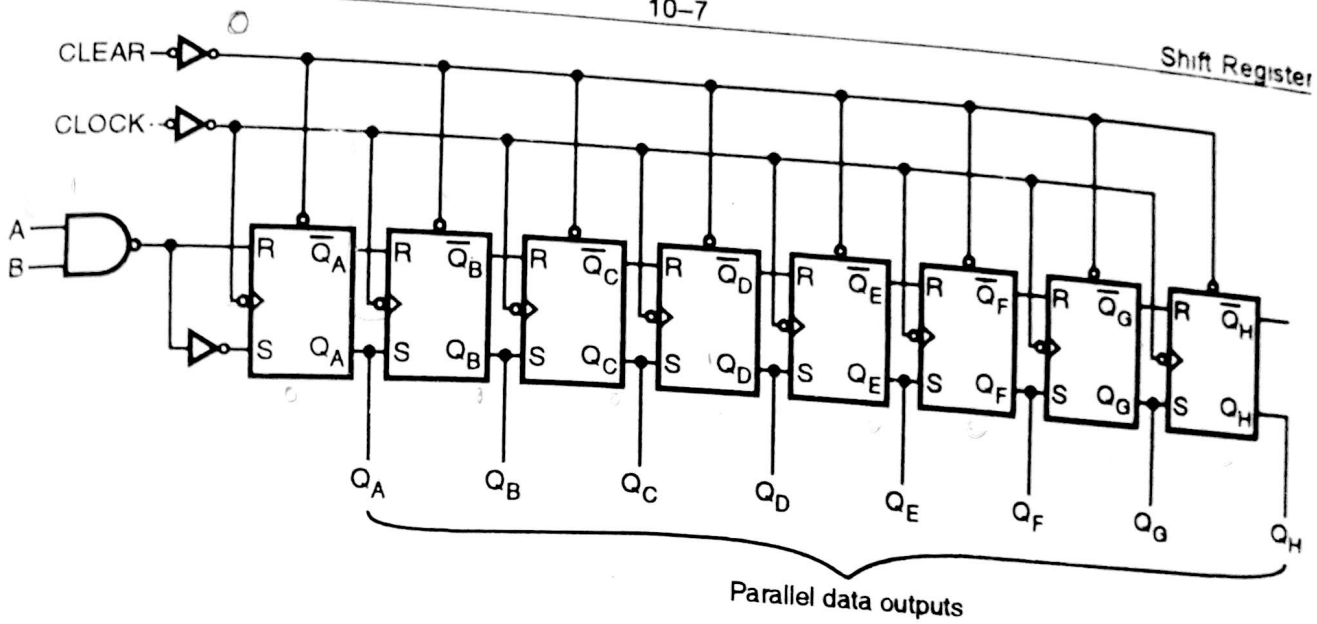



**Fig. 10.11 : Circuit diagram**

If you observe the diagram carefully you will find that,

(1) Clock of all FFs are tied together, therefore all FF changes state at same time.

(2) Asynchronous input, CLEAR, is given to all FF. So that when CLEAR = 0. all FFs are resetted.

∴ output Q = 0000 0000.

(3) A and B are data inputs, either of the terminal can be used as ***control line***.

When A = 0, irrespective of B, input to FF will be 0. At each CLOCK edge 0 travels from left to right and after 8 CLOCK pulses. Q output = 0000 0000. It means if B = Data In, data will get inhibited.

When A = 1, NANDgets enabled and data Input B passes through NANDgate inverted. The input data shifted serially in the register.

(4) Output of all FFs are taken as final output. O/p is Q, not $\overline{Q}$ . Output appears at a time, ∴ it is parallel output.

(5) Data inputs may be changed while CLOCK is either low or high.

## 10.5 Parallel in – Parallel out :

This type we have already studied in section 10.1.1, buffer register. In that $Y_3$–$Y_o$ was inputed parallely and $Q_3$ to $Q_0$ taken out parallely.

## 10.6 Ring Counter :

➥ **[ASKED IN EXAM - MAY 98 III ]**

Refer Fig. 10.7. In that circuit if $Q_3$ is connected to ***Serial Data In***, then circuit is going to circulate an injected pulse. The way circuit behaves, it is named ***'Ring Counter'***.

Scanned by CamScanner

**Circuit :**

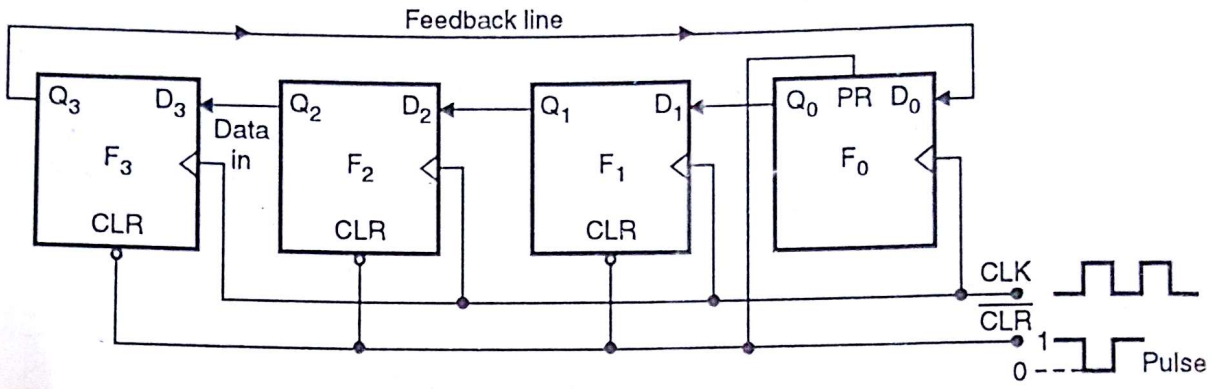



**Fig. 10.12 : Ring counter**

**Working :**

Basically ring counter resembles shift register because the bits are shifted left one position per positive clock edge. Only change is ***feedback line***. i.e. output of $Q_3$ given to $Q_0$.

(1) Initially CLOCK goes low then back to high. ∴ $F_3$, $F_2$, $F_1$ will be reset and $F_0$ is preset giving output $Q_3\ Q_2\ Q_1\ Q_0 = 0001$.

(2) At first positive CLOCK edge, MSB moves to LSB, ($Q_3 \rightarrow Q_0$), $Q_2 \rightarrow Q_3$, $Q_1 \rightarrow Q_2$ and $Q_0 \rightarrow Q_1$.
∴ output $Q_3\ Q_2\ Q_1\ Q_0 = 0010$

(3) At 2nd positive CLOCK edge, with respect to second point, bits will shift left.
∴ output $Q_3\ Q_2\ Q_1\ Q_0 = 0100$.

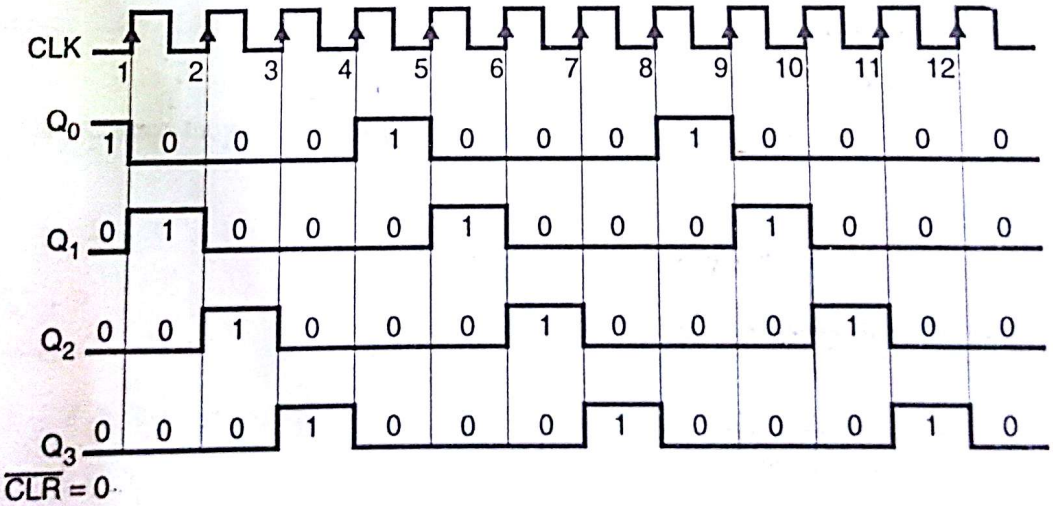



**Fig. 10.13 : Waveforms for ring counter**

(4) At 3rd positive CLOCK edge, with respect to 3rd point, bits will shift left
∴ output $Q_3\ Q_2\ Q_1\ Q_0 = 1000$

(5) At 4th positive CLOCK edge, with respect to 4th point, bits will shift left
∴ output $Q_3\ Q_2\ Q_1\ Q_0 = 0001$

Scanned by CamScanner

Thus presetted '1' follows circular path or makes ***ring***. ∴ it is called as ring counter.

Observe waveform for the same. Refer Fig. 10.13.

**Conclusion :**

(1) Number of FFs used are 4.

(2) Number of states for ring counter are also 4. i.e. 0001, 0010, 0100, 1000 and chain repeats.

(3) Number of 1 s is single.

Let's write output Q for 8 bit. So for 8 bit 8 FFs will be used.

Initially Q = 0000 0010

1st clock edges, Q = 0000 0010

2nd clock edges, Q = 0000 0100

3rd clock edges, Q = 0000 1000

4th clock edges, Q = 0001 0000

5th clock edges, Q = 0010 0000

6th clock edges, Q = 0100 0000

7th clock edges, Q = 1000 0000

8th clock edges, Q = 0000 0001 → Repeat,

If you require you can also take output from $\bar{Q}$.

∴ for 4 bit Refer Fig. 10.12

Initially when $\overline{CLOCK}$ = 0 Q = 0001 ∴ $\bar{Q}$ = 1110

1st CLOCK edge $\bar{Q}$ = 1101

2nd CLOCK edge $\bar{Q}$ = 1011

3rd CLOCK edge $\bar{Q}$ = 0111

4th CLOCK edge $\bar{Q}$ = 1110

**Application :**

Ring counters are invaluable when it is necessary to control sequence of operation.

Mainly ring counters are used in microprocessor.

**Relationship of number of states and FFs :**

***In ring counter number of FFs and number of states are equal.***

Ring counter using JK FF is shown in Fig. 10.14.

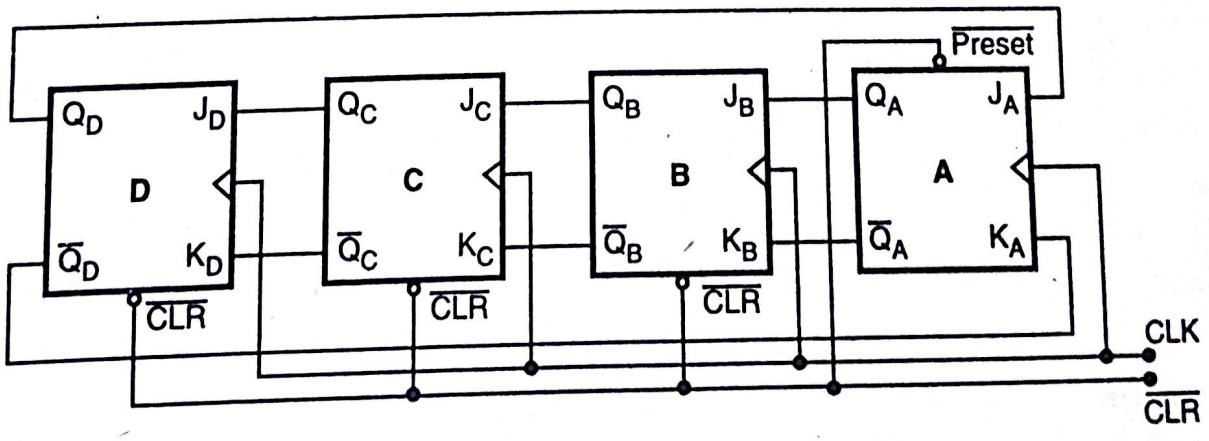



**Fig. 10.14**

The working of circuit is same as that of, when D FF is used.

Scanned by CamScanner

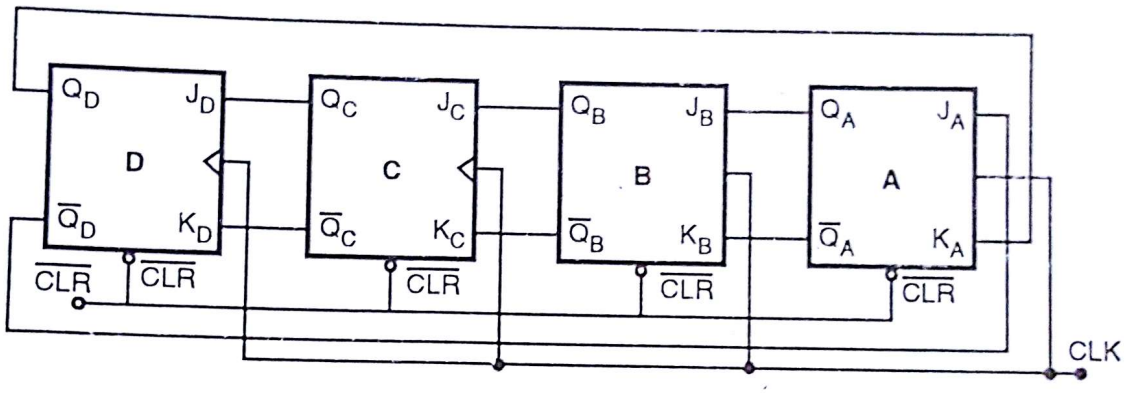
## 10.7 Johnson Counter (Twisted Ring Counter) :

➥ [ASKED IN EXAM - MAY 96, DEC. 96, MAY 97 !!! ]

In ring counter output shift register was fed back to input of first FF, that technique was refered as ***direct feedback***. But if outputs of last FF are crossed and then connected to inputs of first FF, the technique is called ***inverse feedback***. The counter we get from this technique is called Johnson counter or twisted ring counter.

### Circuit :

Let's draw circuit of Johnson counter using positive edge triggered JK FF.




**(a) 4 bit Johnson counter**

| CLOCK | $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ | State | Decimal equivalent |
|---|---|---|---|---|---|---|
| Initially | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 1 | 3 | 3 |
| 3 | 0 | 1 | 1 | 1 | 4 | 7 |
| 4 | 1 | 1 | 1 | 1 | 5 | 15 |
| 5 | 1 | 1 | 1 | 0 | 6 | 14 |
| 6 | 1 | 1 | 0 | 0 | 7 | 12 |
| 7 | 1 | 0 | 0 | 0 | 8 | 8 |
| 8 | 0 | 0 | 0 | 0 | 1 | 0 |

**(b) Truth Table**

**Fig. 10.15**

### Working :

(Refer Fig. 10.15 (a) and (b).)

1 Initially CLOCK is made low and then high after some time.

∴ all FFs will be cleared and output $Q_D\ Q_C\ Q_B\ Q_A$ = DCBA = 0000.

Scanned by CamScanner

2. Now Inputs $J_D = J_C = J_B = 0$, $K_D = K_C = K_B = 1$, and $J_A = 1$, $K_A = 0$. So when CLOCK edge hits, A will change from $0 \rightarrow 1$ BCD are unchanged. $\therefore$ DCBA = 0001.
3. Inputs $J_D = J_C = 0$, $K_D = K_C = 1$. But $J_A = J_B = 1$, $K_A = K_B = 0$. $\therefore$ the moment CLOCK edge hits B and C are unchanged. A is also unchanged. But B changes from $0 \rightarrow 1$.
   $\therefore$ DCBA = 0011.
4. Inputs $J_D = 0$ $K_D = 0$, $J_A = J_B = J_C = 1$ and $K_A = K_B = K_C = 0$. $\therefore$ the CLOCK edge will change output C from $0 \rightarrow 1$, A, B and D are unchanged. $\therefore$ DCBA = 0111.
5. Input $J_D = J_C = J_B = J_A = 1$ and $K_D = K_C = K_B = K_A = 0$.
   $\therefore$ next CLOCK edge changes D from $0 \rightarrow 1$, $\therefore$ DCBA = 1111
6. Because of change of state of D from $0 \rightarrow 1$. Now I/P to $J_A = 0$ and $K_A = 1$. $J_D = J_C = J_B = 1$ and $K_D = K_C = K_B = 0$. $\therefore$ next CLOCK edge changes A from $1 \rightarrow 0$. $\therefore$ DCBA = 1110.
7. Now input to $J_A = J_B = 0$ and $K_A = K_B = 1$ and $J_D = J_C = 1$ and $K_D = K_C = 1$. $\therefore$ next CLOCK edge changes B from $1 \rightarrow 0$. $\therefore$ DCBA = 1100 means A, D, C are unchanged.
8. At this stage input to $J_A = J_B = J_C = 0$ and $K_A = K_B = K_C = 1$, $J_D = 1$ and $K_D = 0$. $\therefore$ next CLOCK edge keeps A, B and D unchanged, C changes from $1 \rightarrow 0$. $\therefore$ DCBA = 1000.
9. Finally input to $J_A = J_B = J_C = J_D = 0$ and $K_A = K_B = K_C = K_D = 1$ $\therefore$ at next CLOCK edge D changes from $1 \rightarrow 0$. $\therefore$ DCBA = 0 0 0 0. Now cycle will repeat from point 2 onwards.

As we have seen in working of Johnson counter and observing truth table, we conclude that number of states of Johnson counter are double of number of FFs. Therefore for 4 FFs states will be 8, 5 FFs states will be 10 and for 8 FFs states will be 16.

Johnson counter using DFF,

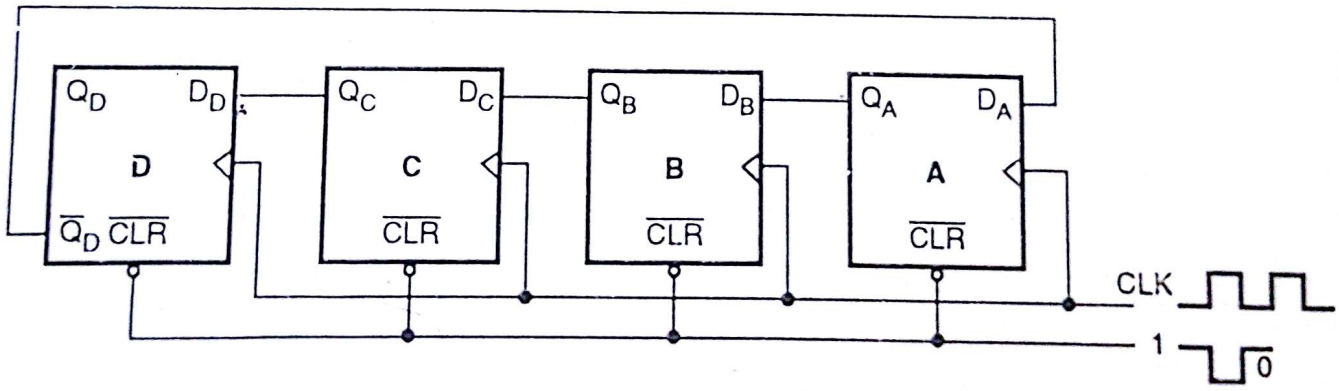



**Fig. 10.16**

The working of the circuit is same as what we have seen.

## 10.8 Applications of Shift Register :

Shift registers can be used for :

***(1) Generating sequence (pulse generator)***

***(2) Counters***

***(3) Random bit generator.***

Scanned by CamScanner

**1. Sequence Generator (Pulse Generator) :**

Block schematic structure is shown in Fig. 10.17.

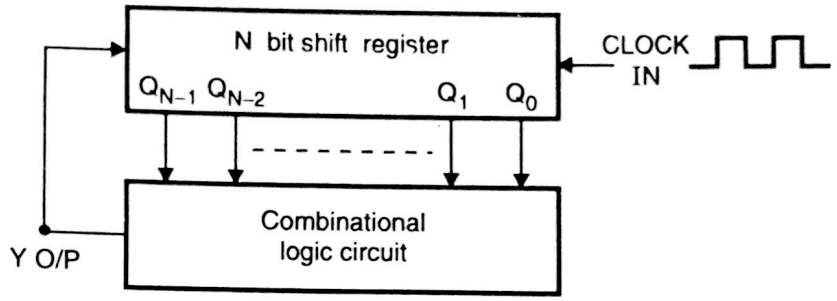



**Fig. 10.17**

Basically shift register sequence generator contain basic two blocks :

(1) 'N' bit shift register (SIPO type).

(2) Combination logic circuit.

Combinational logic circuit has input $Q_{N-1}$ to $Q_0$. From this it will decide, what should be the next serial input for shift register block, should be generated, so that new state or sequence will appear. Now we will take an example to design sequence generator.

---

**Ex. 1 :** The following pulse train is to be generated by using a shift register. Explain the steps in designing and draw logic diagram.

pulse train ... 1011010110 ...

**Soln. :**

(1) Observe the pulse train carefully and you will find that main pulse train, which is repeating, is 10110.

i.e. ... $\underbrace{10110}_{\text{Main}}\underbrace{10110}_{\text{Repeating}}$ ...

(2) Calculate number of distinct timing intervals. In short you calculate number of bits of pulse train. In our case it is,

5 ( 1 0 1 1 0 )

1 2 3 4 5 ...

(3) Number of FFs can be given by equation,

$$S \leq 2^N - 1$$

where $N$ = Number of FFs.

$S$ = Number of distinct timing intervals.

$\therefore \quad S \leq 2^N - 1$

$\therefore \quad 6 \leq 2^N \quad \therefore N \cong 3$

$\therefore$ 3 FFs are required.

**Note :** It may happen that if we sequence the pulse train properly, might be number of states given by 3 FF will be of repeating nature. Therefore we ***may*** have to increase number of FFs.

**(4) Preparing state table for sequence generator.**

Make table, containing number of states, decimal equivalent and 3 output of FF (N = 3)

Scanned by CamScanner

| State | O/P s of FF $Q_C$ (MSB) | $Q_B$ | $Q_A$ (LSB) | Decimal equivalent |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 5 |
| 2 | 0 | 1 | 1 | 3 |
| 3 | 1 | 1 | 0 | 6 |
| 4 | 1 | 0 | 1 | 5 |
| 5 | 0 | 1 | 0 | 2 |

1st 2nd 3rd

LIBRARY Dept. IT 743

**Fig. 10.18(a) : State table**

Procedure for preparing state table.

(a) Write the format of table as shown in Fig.10.18(a). i.e. states, output of 'N' number of FF and decimal equivalent.

(b) Under column 1st (Normally MSB is selected) write down sequence 'vertically'.

(c) Now just imagine that sequence is having ring structure and shift the full sequence by one bit.

∴ if sequence is 10110, rotate it left by one bit so we will get 01101. Write down shifted version under column 2.

(d) Again shift the sequence by one bit. Therefore with respect to original it will be shifted twice.

∴ sequence under column 2 is 01101. shift left by one bit. ∴ sequence is 11010. Write it under column 3.

(e) Continue shifting of sequence unless you reach to LSB column.

After shifting (Rolling) sequence now find out decimal equivalent of binary sequence. For our case we got 5, 3, 6, 5, 2.

In this decimal sequence, decimal number 5 is repeated twice, ∴ states are not distinct, ∴ we have to increase number of FFs to get distinct states.

Therefore make state table again with four FFs.

| State | O/P of FF $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ | Decimal Equivalent |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 11 |
| 2 | 0 | 1 | 1 | 0 | 6 |
| 3 | 1 | 1 | 0 | 1 | 13 |
| 4 | 1 | 0 | 1 | 0 | 10 |
| 5 | 0 | 1 | 0 | 1 | 5 |

**Fig.10.18(b)**

Scanned by CamScanner

Procedure for making truth table is same as we discussed just now. If we observe the decimal equivalent, it is coming out to be 11, 6, 13, 10, 5. All states are distinct.

**(5) Design of combinational circuit :**

Here we have to find out O/P Y of combinational circuit, given to shift register, so that state changes from previous to next.

For example let's say DCBA = 1011 (1 decimal), what should be input to shift register, so that DCBA will be 0110. Means we have to apply '0' to input of shift register or in short Y should be 0. Same way you find Y for each state. The format of table is shown in Fig.10.18(c).

| State | $Q_D$ | $Q_C$ | $Q_B$ | $Q_A$ | O/P Y |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 | 1 |
| 3 | 1 | 1 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 |

MSB column

LSB column

**Fig.10.18(c)**

As shown in Fig. 10.18(c), output Y of combinational circuit is again nothing but 1 bit shifted version of LSB column.

**(6) K-Map :**

Valid states are 11, 6, 13, 10, 5. remaining states are don't care. states reached to 13, four variable K-Map should be used.

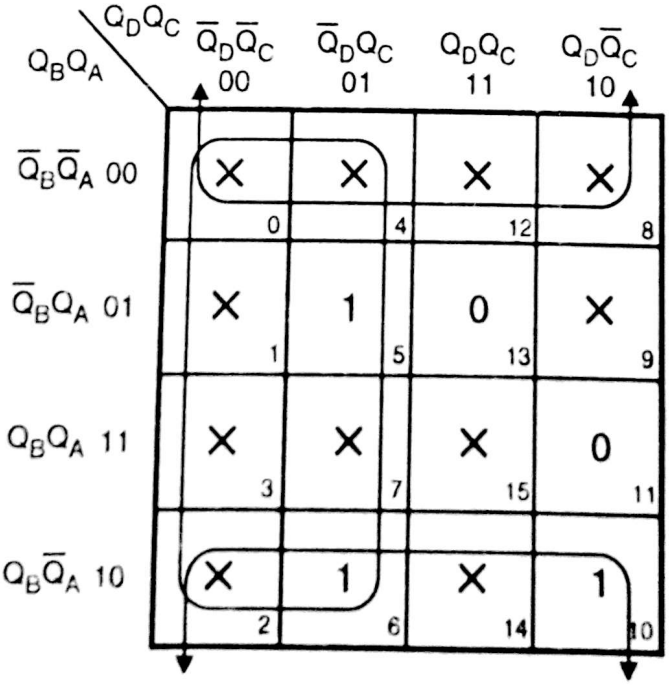



**Fig. 10.18(d)**

Scanned by CamScanner

(7) **Logic diagram :**

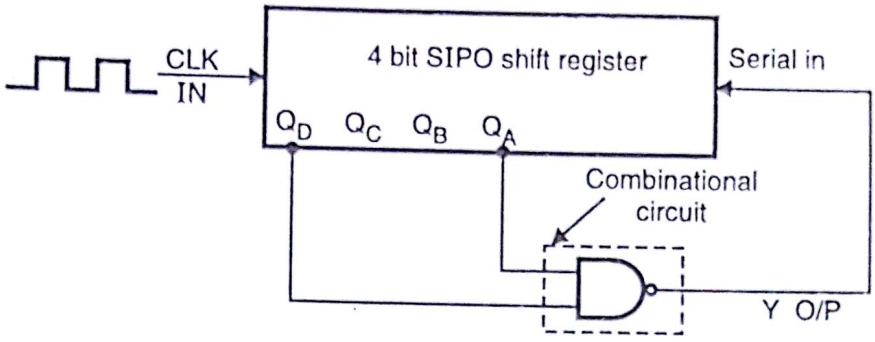



**Fig. 10.18(c)**

## 10.9 Pseudo Random Binary Sequence (PRBS) Generator :

One important application of shift register is pseudo-random generator. It is basically used to generate random sequence. Due to random sequence generation the same can be used for generating noise ; as well as used for data encryption. By generating random noise, one can test ***immunity given by his / her design to the noise.*** By encrypting data, it becomes difficult job for a person who hacks the data in between. Basically, the sequence generated is not truely random because it cycles through all possible combinations once every $2^n - 1$ clock cycles (n = number of FFs).

Generation of pseudo random sequence is based on feedback given to shift register through ***Combinational logic circuit.*** Fig. 10.19 depicts noise generator based on pseudo-random sequence. The combinational circuit used is simple EX-OR gate.

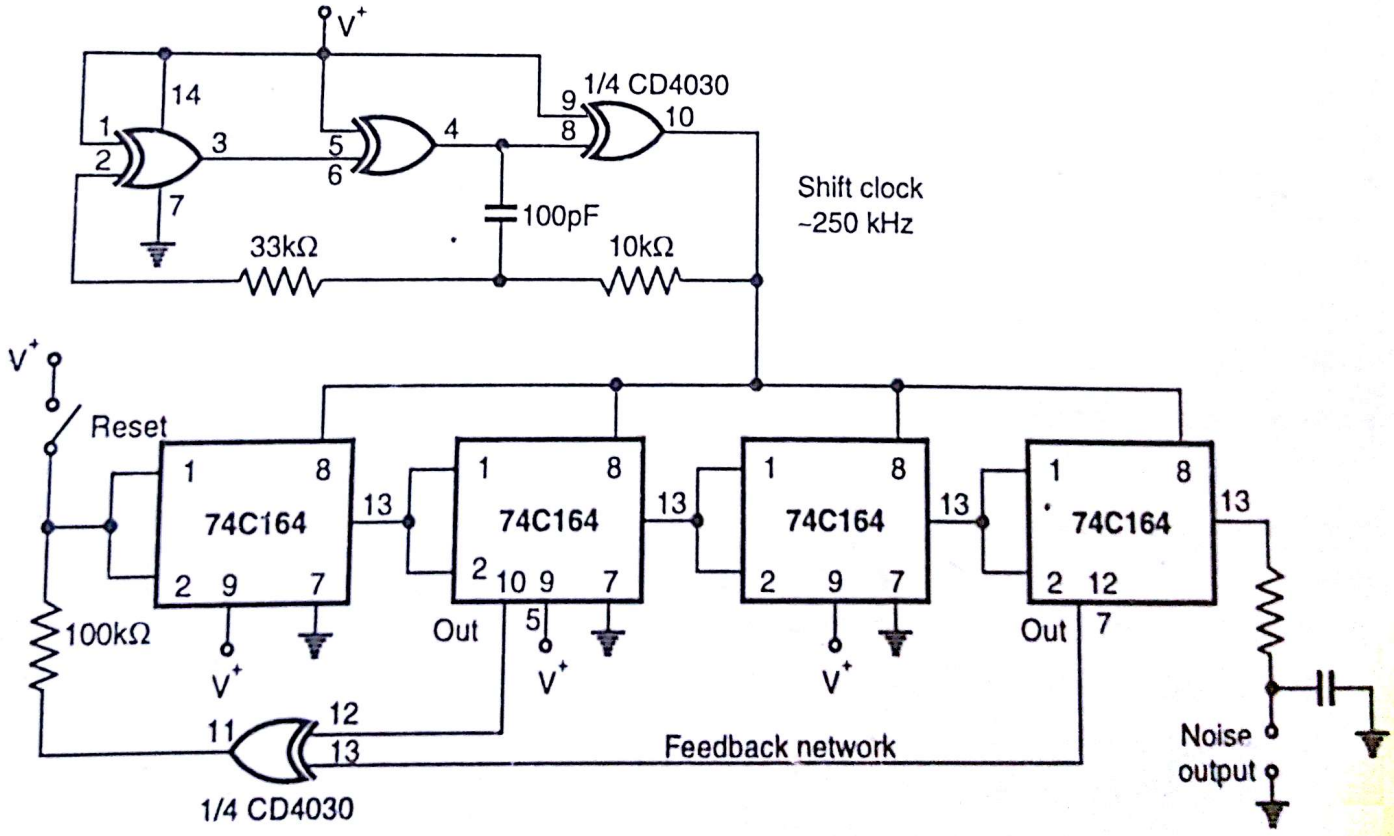



**Fig. 10.19 : Noise generator based on pseudo-random sequence**

Scanned by CamScanner

It uses, 31 stage shift register provided with linear feedback to produce maximum length pseudo-random bit sequence. One of the significant features of pseudo-random sequence is that the noise produced from there is repeatable. The spectral density of the noise output of this circuit is uniform to within ±1 dB over the frequency range 20Hz to 20 kHz.

The maximum length of sequence (MLS) generated will be $2^n - 1$.

$\therefore \quad 2^n - 1 = 2^{31} - 1$, sufficient length to encrypt the data.

## 10.10 IC 7495 :

➥ [ASKED IN EXAM - DEC. 98 !!! ]

We are going to study now TTL MSI Chip 7495, 4 bit shift register with serial and parallel synchronous operating mode. Features of the chip are:

(1) Synchronous shift left capability.

(2) Synchronous parallel load.

(3) Separate shift and load clock inputs.

(4) Synchronous expandable shift right.

Let's see Internal block of IC 7495 and Pin configuration.

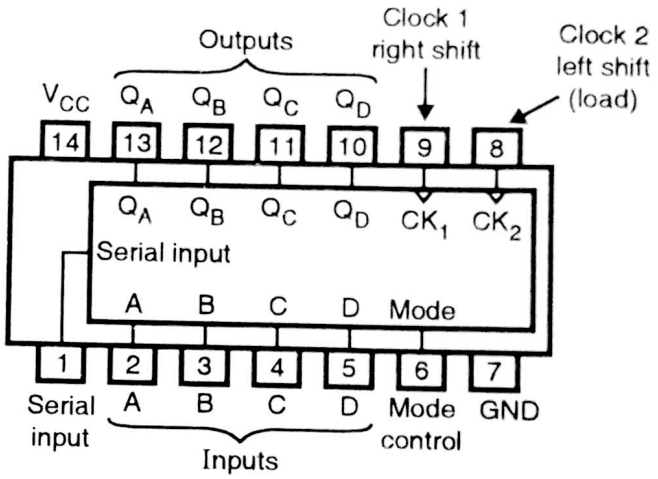



**Fig. 10.20 (a) : Pin diagram**

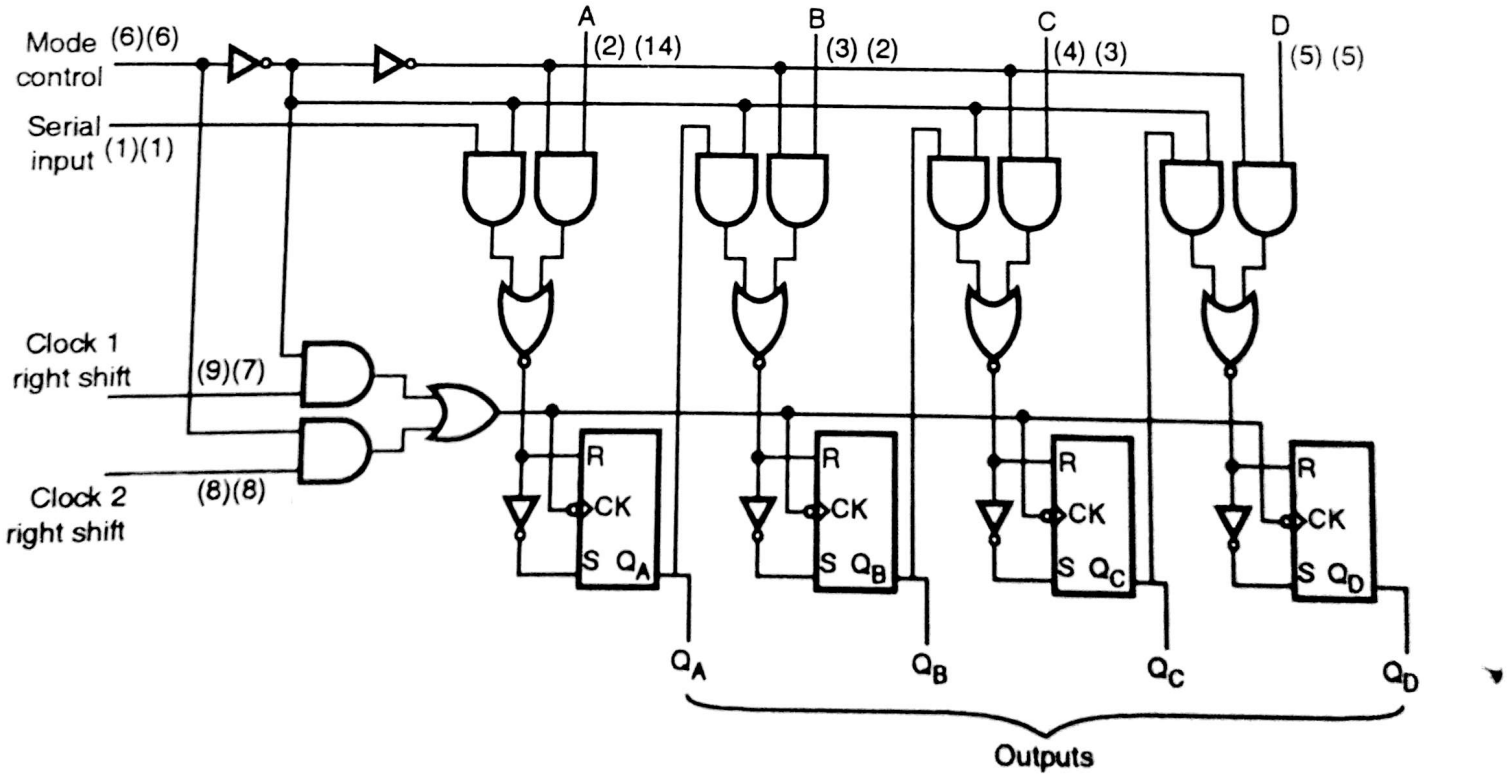



**Fig. 10.20 (b) : Logic diagram (also called universal shift register)**

Scanned by CamScanner

We are going to perform basically three functions using 7495 :

(1) Parallel load.

(2) Shift left.

(3) Shift right.

### 1. Parallel Load

(a) Make mode control = 1, therefore AND gate 2, 4, 6, 8 will get enabled, and gates 1, 3, 5, 7 will get disabled.

(b) Apply input ABCD.

(c) In this case clock 2 input (pin number 8) is enabled. Therefore a HIGH to LOW transition on clock and input will transfer parallel data ABCD inputs to $Q_0 - Q_3$ outputs.

(d) Here clock 1 input is don't care. DS is also don't care.

### 2. Shift Right

(a) Make mode control = 0, therefore AND gates 1, 3, 5, 7 will be enabled and AND gate 2, 4, 6, 8 will get disabled.

(b) The data input to FF $Q_A$ is now at serial input ($D_S$).

(c) Clock and input is don't care. ABCD inputs are don't care.

(d) Apply CLOCK input to clock 1.

(e) A HIGH to low transition on enabled clock 1 input transfers data serially from $D_S$ to $Q_A$, $Q_A$ to $Q_B$, $Q_B$ to $Q_C$ and $Q_C$ to $Q_D$ respectively (right shift.)

### 3. Shift Left

(a) For shift left operation external connection are to be performed.
Connect $Q_D$ to C, $Q_C$ to B and $Q_B$ to A.
As shown in Fig. 10.21.

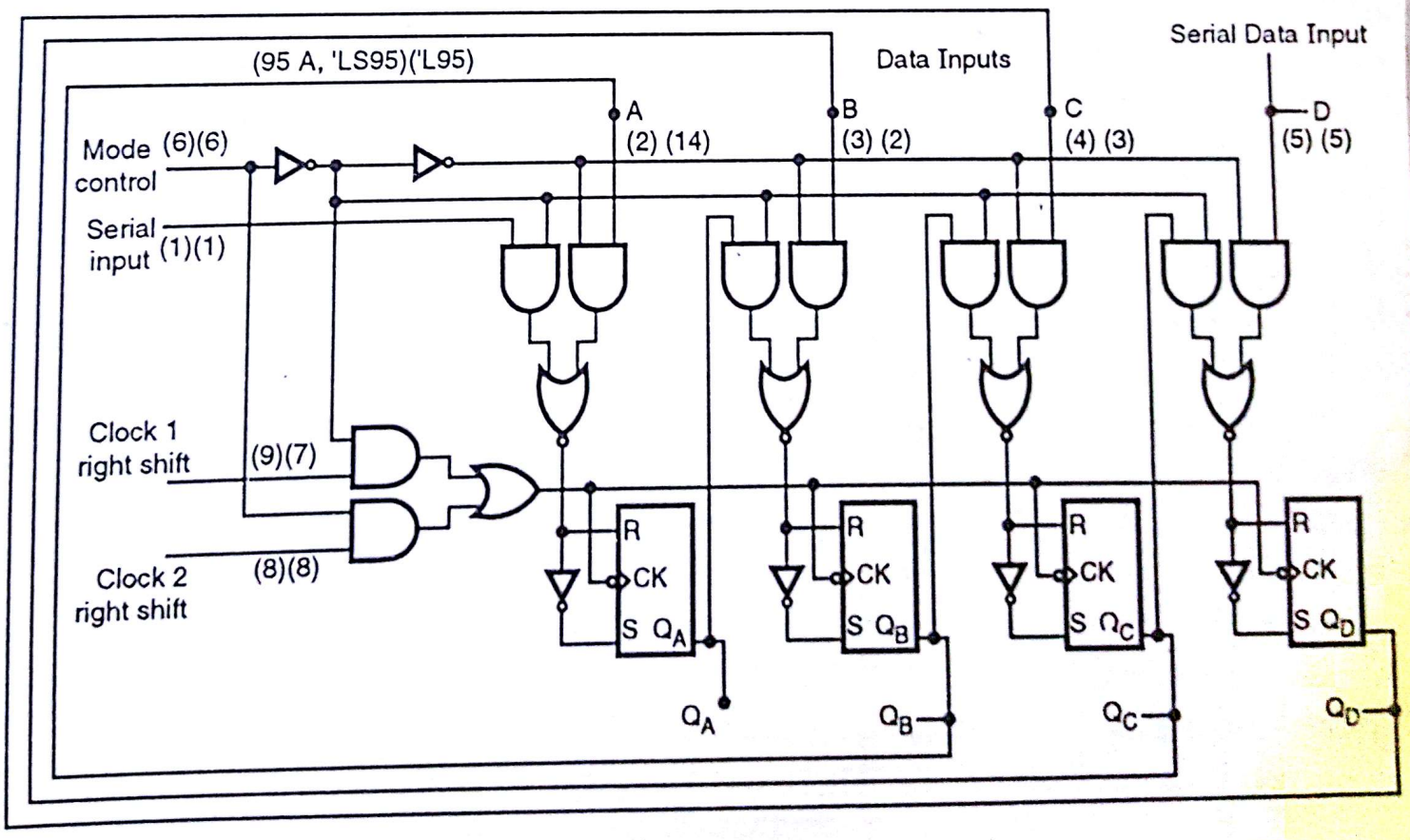



**Fig. 10.21**

Scanned by CamScanner

(b) Make mode control = 1

(c) Apply serial data to input D (pin no. 5).

(d) Clock input 1 is don't care.

(e) Apply CLOCK input to clock 2 (left shift).

(f) Each HIGH to LOW transition will transfer data from D to $Q_D$, $Q_D$ to $Q_C$, $Q_C$ to $Q_B$ and $Q_B$ to $Q_A$ i.e. left shift operation performed.

**Ex. 2 :** Generate the following pulse train using shift register.

11001110.

**Soln. :**

(1) Pulse train is 11001110.

(2) No. of distinct timing intervals are 8. ∴ No. of FFs required are

$S \leq 2^N - 1$

$8 \leq 2^N - 1$

$9 \leq 2^N$ ∴ $\boxed{N = 4}$

(3) State table.

| State | D | C | B | A | Decimal equivalent |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 12 |
| 2 | 1 | 0 | 0 | 1 | 9 |
| 3 | 0 | 0 | 1 | 1 | 3 |
| 4 | 0 | 1 | 1 | 1 | 7 |
| 5 | 1 | 1 | 1 | 0 | 14 |
| 6 | 1 | 1 | 0 | 1 | 13 |
| 7 | 1 | 0 | 1 | 1 | 11 |
| 8 | 0 | 1 | 1 | 0 | 6 |

**Fig. 10.22**

No. of states are distinct i.e. 12, 9, 3, 7, 14, 13, 11, and 6.

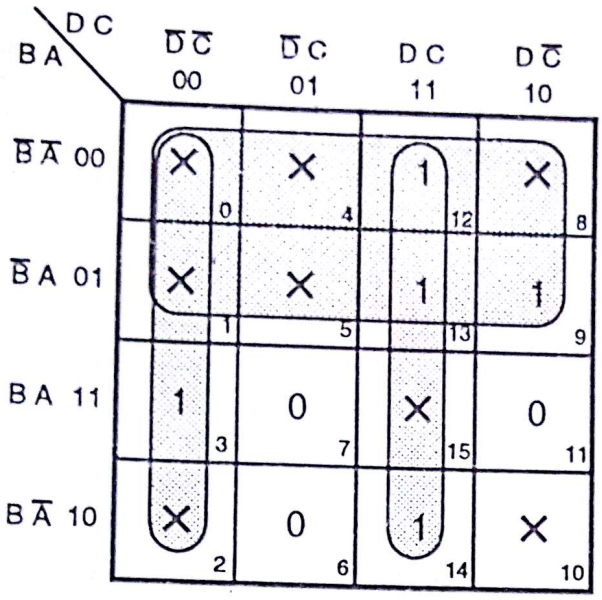
(4) Combinational circuit design.

| State | D | C | B | A | Y |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 1 | 1 | 0 |
| 5 | 1 | 1 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 | 1 | 1 |
| 7 | 1 | 0 | 1 | 1 | 0 |
| 8 | 0 | 1 | 1 | 0 | 0 |

**Fig. 10.23**

Scanned by CamScanner

**K map**




$Y = \overline{B} + \overline{D}\,\overline{C} + DC = \overline{B} + (D \oplus C)$

**(5) Logic diagram.**

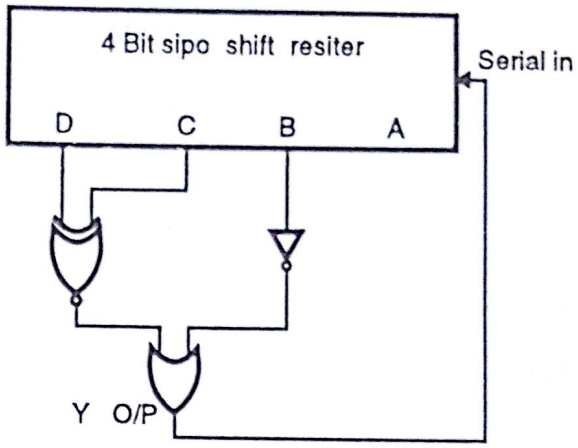



**Fig. 10.24**

**Ex. 3 :** The following pulse train is to be generated by using shift register. Explain the steps in designing and draw logic diagram.
Pulse train 10101, 10101, ...

**Soln. :**

(1) Pulse train is 10101

(2) No. of. timing Intervals are 5.

(3) No. of FFs $\leq\leq 2^N - 1$ where N = no. of FFs.
$5 + 1 \leq 2^N\ 6 \leq 2^N$, $\therefore$ N = 3

(4) State table :

| State | C | B | A | Decimal equation |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 5 |
| 2 | 0 | 1 | 0 | 2 |
| 3 | 1 | 0 | 1 | 5 |
| 4 | 0 | 1 | 1 | 3 |
| 5 | 1 | 1 | 0 | 2 |

**Fig. 10.25**

Decimal equivalent states are 5, 2, 5, 3, 7. $\therefore$ 5 is occurring twice in the sequence.
$\therefore$ Now try with 4 FFs.

| State | D | C | B | A | Decimal equivalent |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 10 |
| 2 | 0 | 1 | 0 | 1 | 5 |
| 3 | 1 | 0 | 1 | 1 | 11 |
| 4 | 0 | 1 | 1 | 0 | 6 |
| 5 | 1 | 1 | 0 | 1 | 13 |

**Fig. 10.26**

Scanned by CamScanner

States of counter are unique 10, 5, 11, 6, 13.

(5) Design of combinatioal circuit.

| State | D | C | B | A | Y |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | 1 |
| 3 | 1 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 1 | 0 | 1 |
| 5 | 1 | 1 | 0 | 1 | 0 |

Fig. 10.27

**K map :**

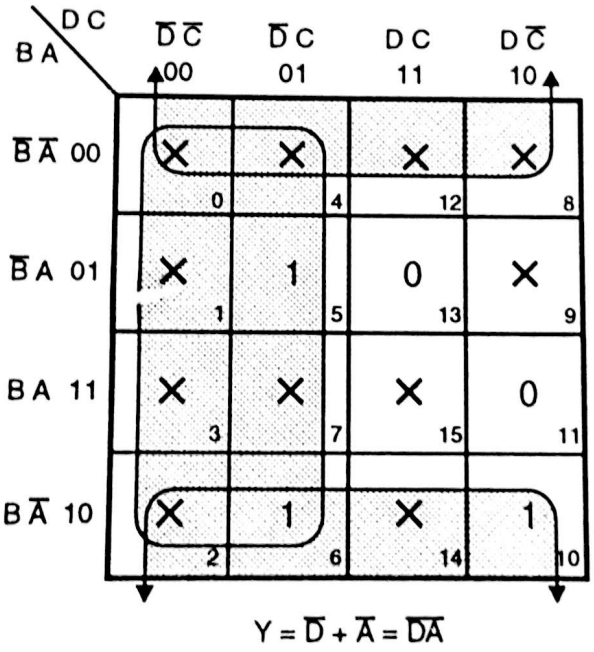



Fig. 10.28.

(6) Logic diagram

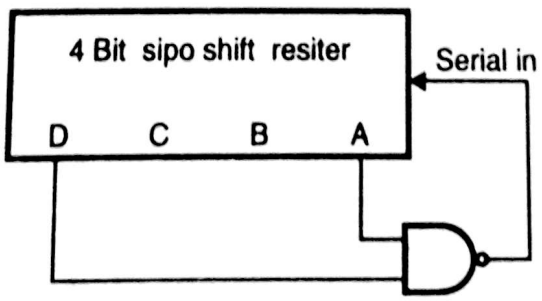



Fig. 10.29

Scanned by CamScanner

## 10.11 Analysis of Shift Counter :

In this, circuit will be given and one has to find states through which shift register passes. To understand this let's solve example.

**Ex. 4 :** A shift register with associated combinational logic circuit is shown in Fig. 10.30. Explain its action by drawing waveforms for the outputs of shift register (ABCD). Assume that Initially shift register is in the state ABCD = 1111.

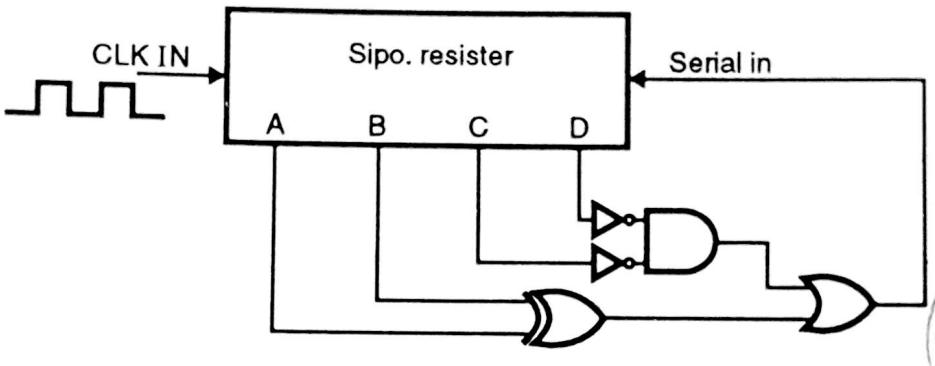



**Fig. 10.30**

**Soln. :**

(1) Write equation for output Y.

(2) Basically output Y is serial in for shift register, shift is left shift, ∴ serial in placed in LSB i.e. D.

So make table.

| CLOC K edge | Serial In | A | B | C | D | O/P $Y = (A \oplus B) + \overline{C}\,\overline{D}$ |
|---|---|---|---|---|---|---|
| – | – | 1 | 1 | 1 | 1 | 0 |
| 1st | 0 | 1 | 1 | 1 | 0 | 0 |
| 2nd | 0 | 1 | 1 | 0 | 0 | 1 |
| 3rd | 1 | 1 | 0 | 0 | 1 | 1 |
| 4th | 1 | 0 | 0 | 1 | 1 | 0 |
| 5th | 0 | 0 | 1 | 1 | 0 | 1 |
| 6th | 1 | 1 | 1 | 0 | 1 | 0 |
| 7th | 0 | 1 | 0 | 1 | 0 | 1 |
| 8th | 1 | 0 | 1 | 0 | 1 | 1 |
| 9th | 1 | 1 | 0 | 1 | 1 | 1 |
| 10th | 1 | 0 | 1 | 1 | 1 | 1 |
| 11th | 1 | 1 | 1 | 1 | 1 | 0 |

**Fig. 10.31**

Scanned by CamScanner

Explaination : (Refer Fig. 10.32)

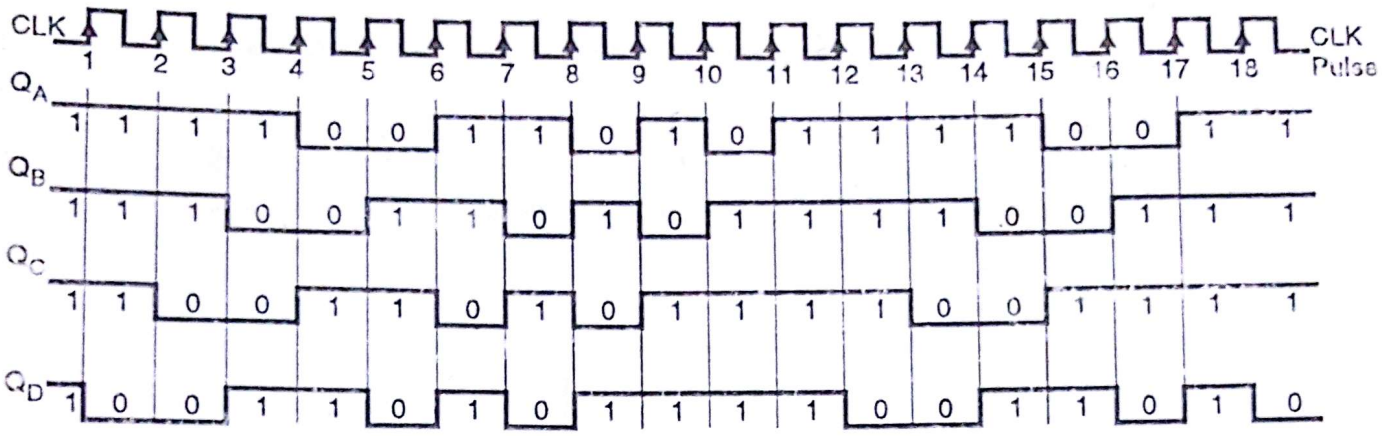



**Fig. 10.32**

Initially ABCD = 1111, $\therefore$ Y = $(1 \oplus 1) + \bar{1}\,\bar{1} = 0 + 0 = 0$
$\therefore$ Y = serial In = 0, $\therefore$ when 1st CLOCK edge hits, ABCD = 1110 (Left shift).

Now ABCD = 1110, $\therefore$ Y = $(1 \oplus 1) + \bar{1}\bar{0} = 0 + 0 = 0$
$\therefore$ Y = serial In = 0, $\therefore$ when 2nd CLOCK edge hits, ABCD = 1100
Thus you proceed till you get initial condition ABCD = 1111.
From table (Fig 10.31) we found that counter, counts through 15, 14, 12, 9, 3, 6, 13, 10, 5, 11, 7 and 15.
Thus generates sequence → 111100110101
Waveforms (Waveforms are as shown in Fig. 10.32).

---

**Ex. 5 :** Using a shift register design a pulse train generator to generate the waveform shown in Fig. 10.33. Also examine the response of the circuit for all illegal states.

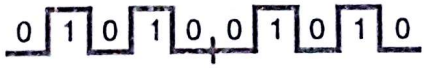



**Fig. 10.33**

**Soln. :**

(1) The pulse train is 01010.
(2) No. of distinct time intervals are 5.
(3) No. of FFs required will be

$5 \leq 2^N - 1$
$6 \leq 2^N$ $\therefore$ $\boxed{N \equiv 3}$

(4) State table

| State | C | B | A | Y O/P | Decimal equivalent |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | – | 2 |
| 2 | 1 | 0 | 1 | – | 5 |
| 3 | 0 | 1 | 0 | – | 2 |
| 4 | 1 | 0 | 0 | – | 4 |
| 5 | 0 | 0 | 1 | – | 1 |

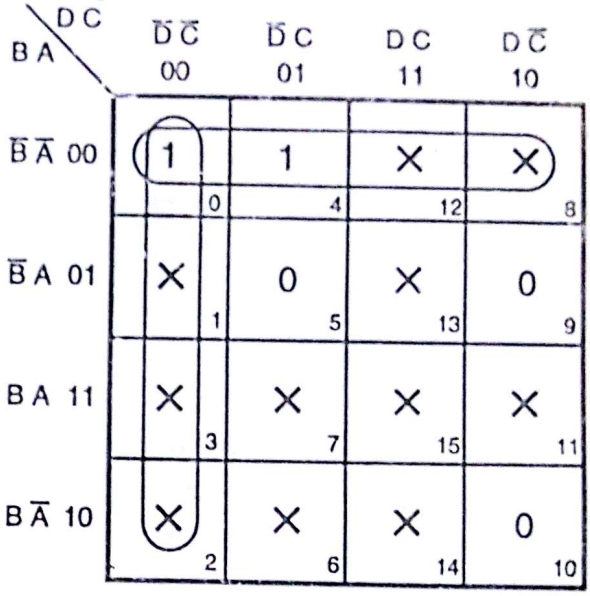
Scanned by CamScanner

State 2 is repeating. ∴ increase one FF.

State table.

| State | D | C | B | A | Y O/P | Decimal equivalent |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 5 |
| 2 | 1 | 0 | 1 | 0 | 0 | 10 |
| 3 | 0 | 1 | 0 | 0 | 1 | 4 |
| 4 | 1 | 0 | 0 | 1 | 0 | 9 |
| 5 | 0 | 0 | 1 | 0 | 1 | 2 |

(5) ∴ K–map :




$Y = \overline{B}\,\overline{A} + \overline{D}\,\overline{C}$

(6) Let's check for invalid states. Analysis method is same as followed in Ex. 4 consider initially state is 0000 (Invalid state)

| | CLK edge | Serial In | D | C | B | A | O/P Y $= \overline{B}\overline{A} + \overline{D}\overline{C}$ | Decimal states | |
|---|---|---|---|---|---|---|---|---|---|
| Initially | – | – | 0 | 0 | 0 | 0 | 1 | 0 | |
| | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | Invalid |
| | 2 | 1 | 0 | 0 | 1 | 1 | 1 | 3 | |
| | 3 | 1 | 0 | 1 | 1 | 1 | 0 | 7 | |
| | 4 | 1 | 1 | 1 | 1 | 0 | 0 | 14 | |
| | 5 | 0 | 1 | 1 | 0 | 0 | 1 | 12 | |
| | 6 | 1 | 1 | 0 | 0 | 1 | 0 | 9 | Valid |
| Initially | – | – | 0 | 1 | 1 | 0 | 0 | 6 | Invalid |
| | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 12 | |
| | 2 | 1 | 1 | 0 | 0 | 1 | 0 | 9 | → Valid |
| Initially | – | – | 1 | 0 | 0 | 0 | 1 | 8 | Invalid |
| | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | | | | | | | After this states will be 3, 7, 14, 12, and 9 | |
| Initially | – | – | 1 | 1 | 0 | 1 | 0 | 11 | → Invalid |
| | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 10 | → Valid |
| Initially | – | – | 1 | 0 | 1 | 1 | 0 | 13 | Invalid |
| | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 7 | |
| | | | | | | | | After this states will be 14, 12 and 9 | Valid |
| Initially | – | – | 1 | 1 | 1 | 1 | 0 | 15 | → Invalid |
| | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 14 → Stop | |

**Fig. 10.34**

Scanned by CamScanner

Total FFs are 4. ∴ No. of states will be 16.

Out of 16, 5 – 10 – 4 – 9 – 2 are valid states. Remaining all i.e. 0 – 1 – 3 – 6 – 7 – 8 – 11 – 12 – 13 – 14 – 15 are invalid.

We are suppose to find the sequence when invalid states are present. So start initially with DCBA = 0 0 0 0 (decimal 0).

From table we came to know that after 0 next state is 1, then 3, 7, 14, 12 and 9. After getting valid state, you have to stop cause if you continue, you will loop your self in valid states.

After this you now look for remaining invalid states.

Remaining are now 6, 8, 11, 13, 15.

Take each state one by one.

When 6 is taken, next state is 12 and then 9. So again stop.

When 8 is taken, next state is 1 which is invalid. But you are now already aware what will be the next state after 1. So stop these.

Now select state 11, next state is valid 10 so again stop. Select state 13, next state is 7. We know that after 7 next states will be 14, 12, and 9. So stop at 7.

Finally take state 15, next state is 14. After 14 it will be 12 and 9. So stop here.

Just as a curiosity if we draw state diagram for above problem, it will be.

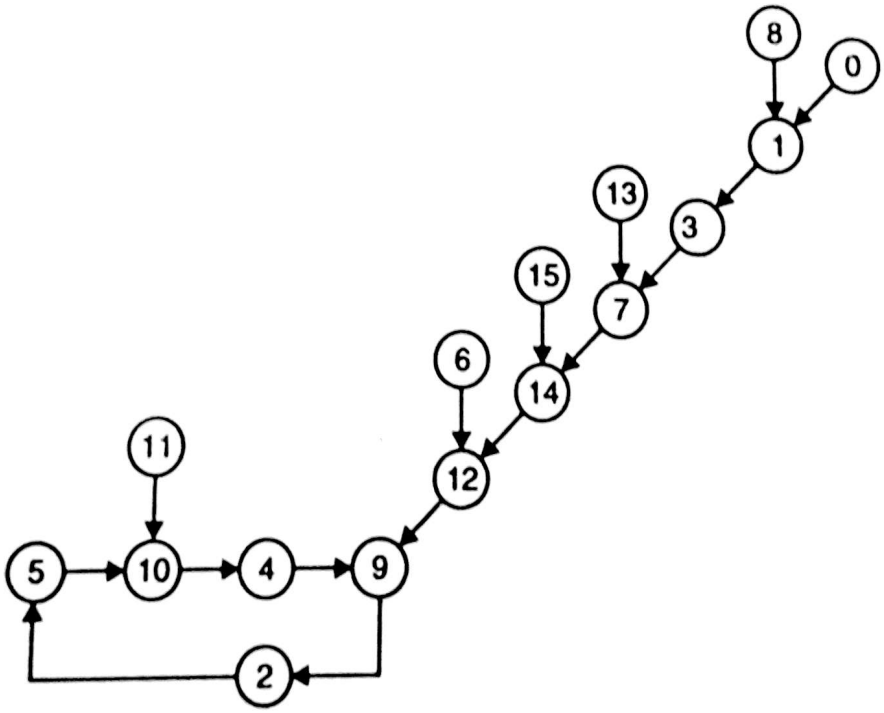



**Fig. 10.35 : State diagram**

Scanned by CamScanner

## 10.12 Examples from University Papers :

1. **Draw a schematic diagram of 4–bit bi–directional shift register using R–S flip–flops. Explain its working.**

**(May 96, 8 Marks, May 97, 10 Marks, Dec. 97, 8 Marks, June 99, 8 Marks)**

**Ans. :** Diagram of 4 bit bi-directional shift register using R–S flip-flop is as follows :

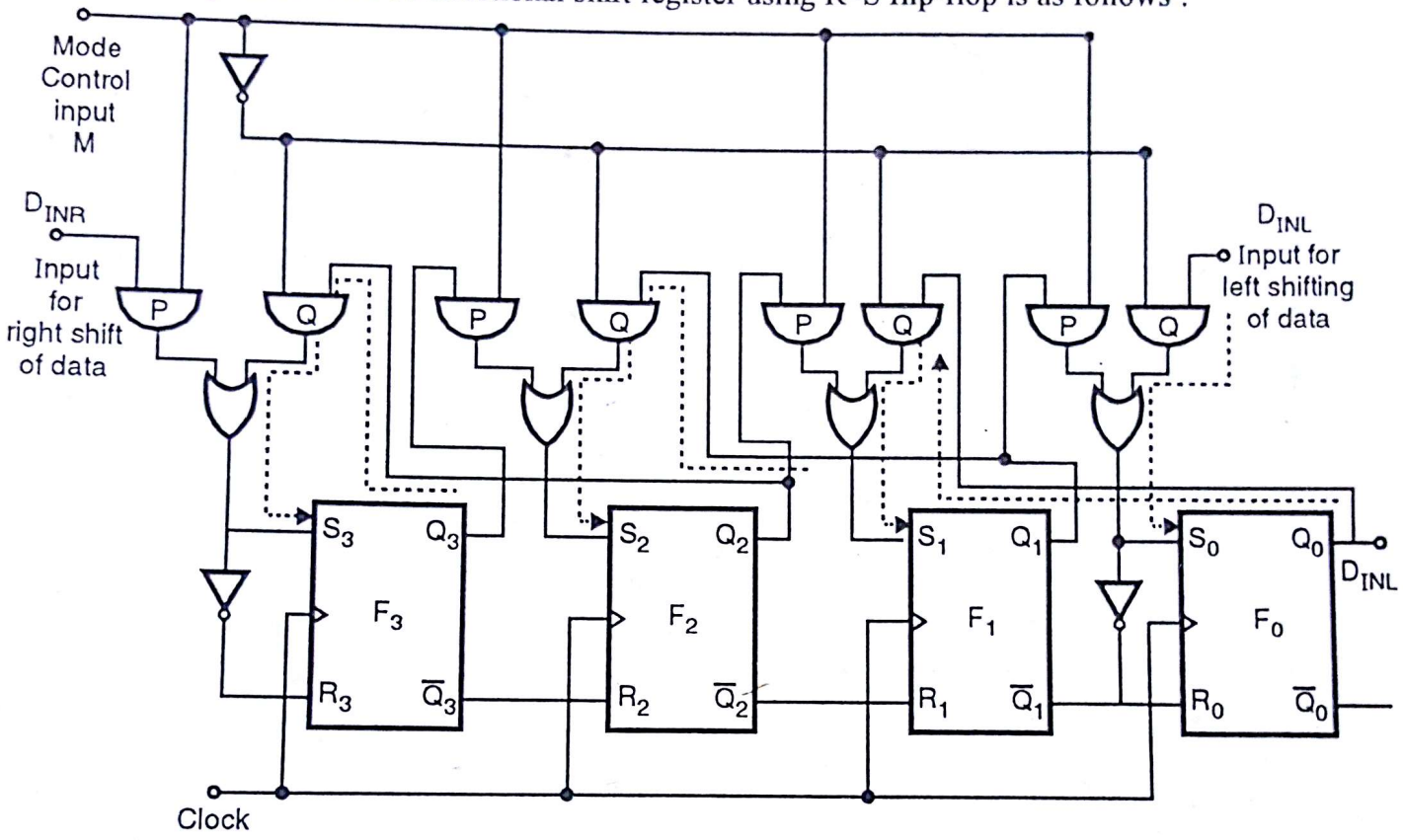



**Fig. Q. 10.1**

**Step I** : Fig. Q.10.1 shows 4 bit bidirection shift register. So data is shifted in both directions.

**Step II** : **Shift right operation :**

Control input M = 1. AND gates P are enabled.

Data is applied at $D_{INR}$. Data D is shifted towards right when CLK pulses are applied.

The output for shift right operation is taken at $Q_0$.

**Step III : For shift left operation :**

Control inut M = 0. AND gates marked Q, are enabled.

Data is applied at $D_{INL}$. Data $D_{INL}$ is shifted towards left [Shown by dotted lines], when CLK pulses are applied. The output for shift left operation is taken at $Q_3$.

2. **Give truth table and schematic diagram of 4-bit twisted ring counter. Explain its working.**

**(May 96, 8 Marks, May 97, 10 Marks)**

**Ans.:** Refer Section 10.7.

3. **For conversion of 4 bit parallel input to serial output using D flip–flops. Give schematic diagram and relevant timing diagram.** **(May 96, Dec. 97, June 99, 8 Marks)**

Scanned by CamScanner

**Ans. :**

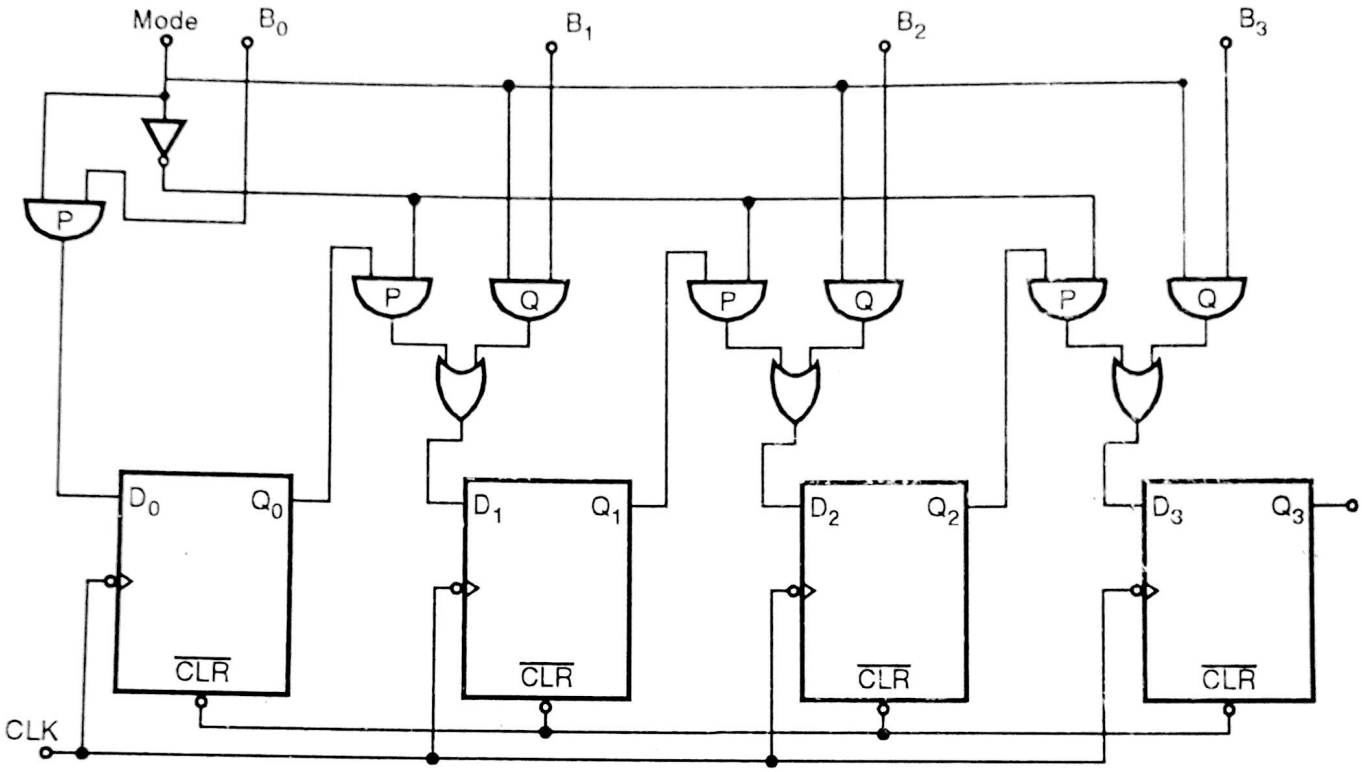



**Fig.Q. 10.3 (a)**

Fig. Q. 10.3.(a) gives us schematic diagram for parallel input serial output.

(i) Inputs are taken at parallel port $B_0B_1B_2B_3$.

(ii) When mode is 1, AND gates marked Q are enabled.

(iii) When mode is 0, AND gates marked P are enabled.

(iv) Output is taken at $Q_3$.

**The relevant timing diagram is as shown in Fig. Q. 10.3(b).**

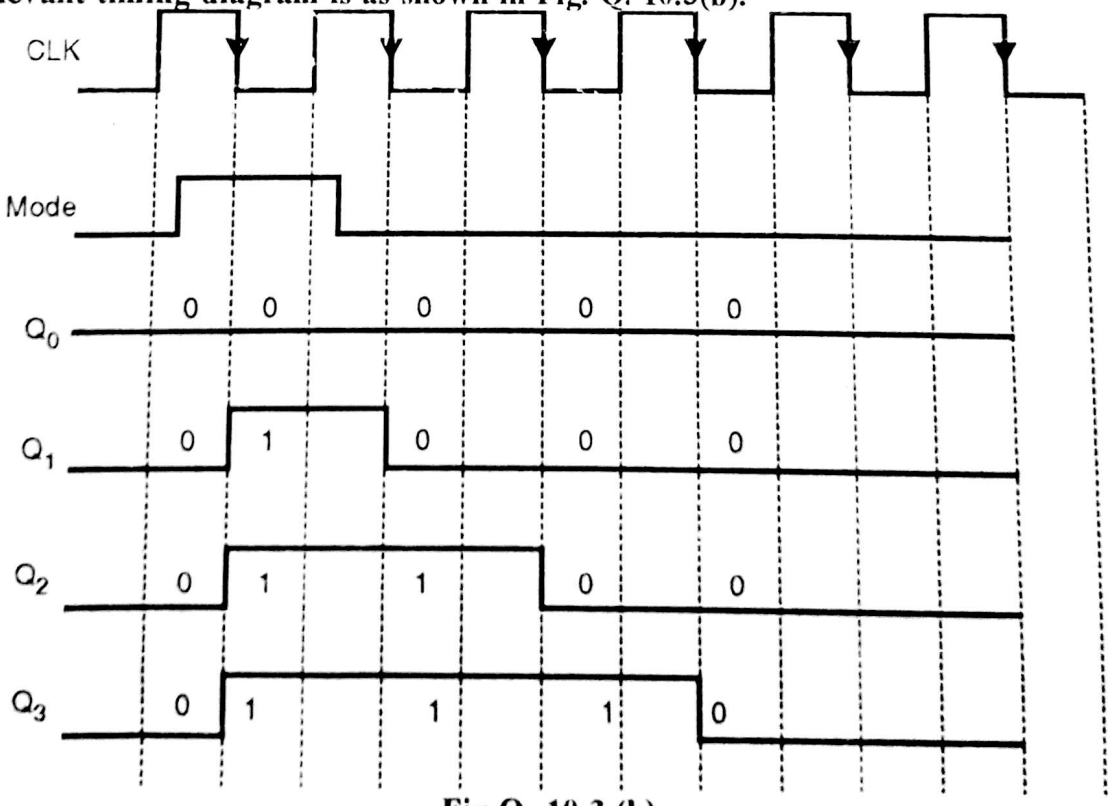



**Fig.Q. 10.3.(b)**

Scanned by CamScanner

At first input applied is $B_0B_1B_2B_3 = 0111$

Mode is = 1

(1) Mode = 1 $\therefore$ $B_0B_1B_2B_3 = 0111$

So after first clock edge, $Q_0Q_1Q_2Q_3 = 0111$

(2) When mode = 0 $Q_3 = 1$

(3) At second clock edge, As $Q_2 = 1$, $Q_3 = 1$

As $Q_1 = 1$ $\therefore$ $Q_2 = 1$

As $Q_0 = 0$ $\therefore$ $Q_1 = 0$

(4) At third clock edge As $Q_2 = 1$, $Q_3 = 1$

As $Q_1 = 0$ $\therefore$ $Q_2 = 0$

As $Q_0 = 0$ $\therefore$ $Q_1 = 0$

**4. Draw a logic diagram of 4 bit bi–directional shift register using D flip–flops. Explain its working.**

**(Dec. 96, 8 Marks)**

**Ans. : The logic diagram of 4–bit bi-directional shift register is as follows :**

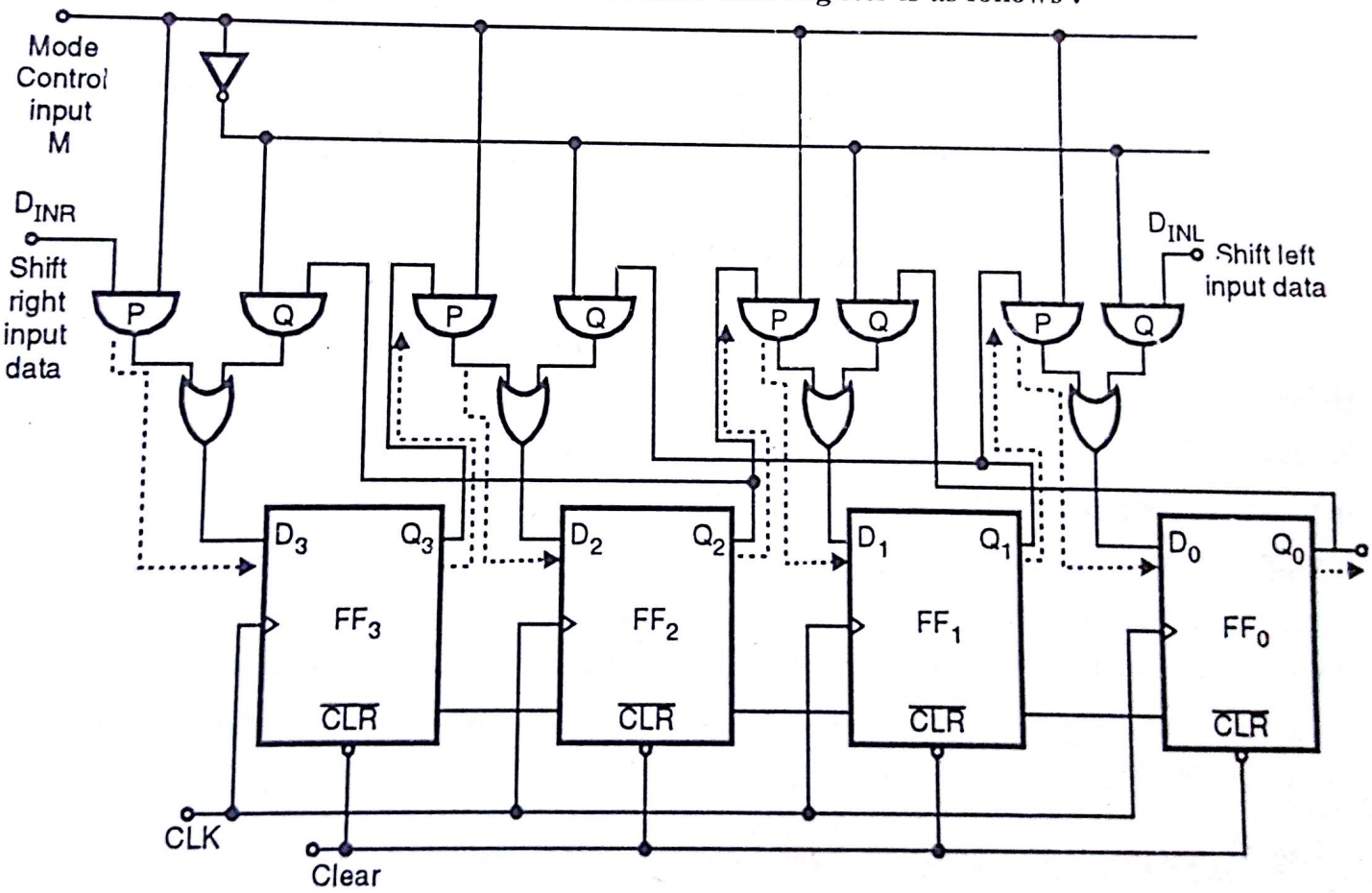



**Fig. Q. 10.4**

**Step I :** The Fig. Q. 10.4 shows 4–bit bi-directional shift register. So the data is shifted in both directions.

Scanned by CamScanner

**Step II : SHIFT RIGHT OPERATION**

Control input M = 1. So AND gates P are enabled. Data is applied at $D_{INR}$. Data is shifted towards right (shown by dotted lines), when CLK pulses are applied. The output for shift right operation is taken at $Q_0$.

**Step III :** For shift left operation control input M is made zero. So AND gates marked, Q are enabled. Data is applied at $D_{INL}$. Data is shifted towards left when clock pulses are applied. The output for shift left operation is taken at $Q_3$.

**5. Convert 4 bit serial input to parallel output using J–K flip–flops. (Dec. 96, 8 Marks)**

**Ans.** Circuit diagram for 4 bit serial input to parallel output is as shown in Fig. Q. 10.5.

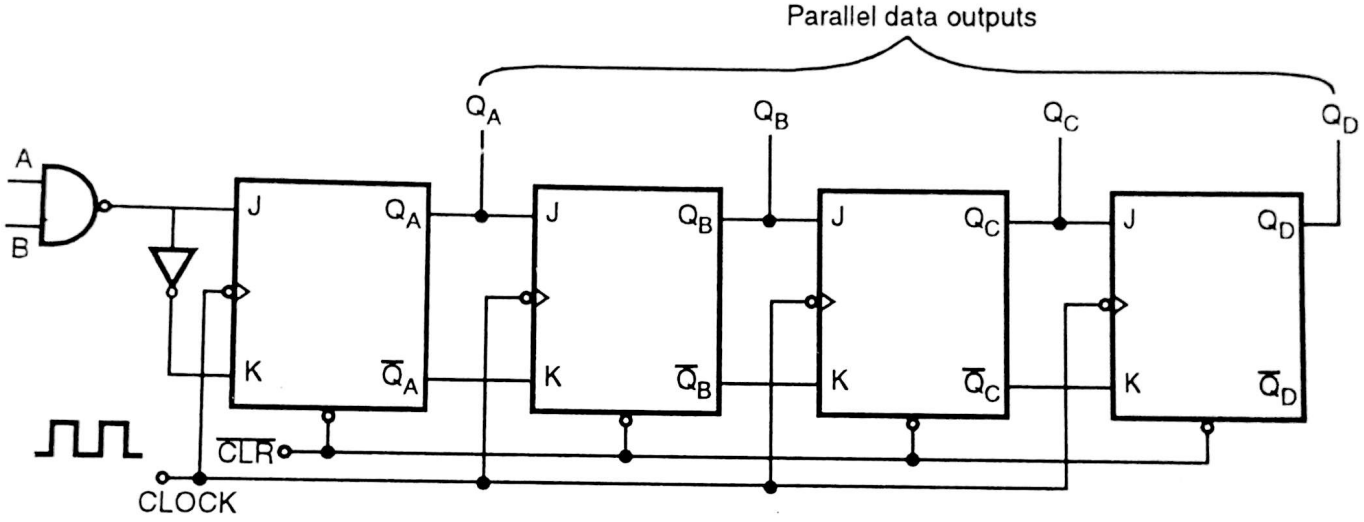



**Fig. Q.10.5**

**Step I :** In Fig. Q. 10.5 clock of all flip–flops are tied together, therefore all filp–flops changes state at same time.

**Step II :** Asynchronous input clear is given to all flip–flops. Hence when clear ( $\overline{CLR}$ ) = 0, all flip–flops reset. ∴ Output Q = 0000

**Step III :** A and B are data inputs, either of the terminal can be used as control line. When A = 0, irrespective of B, input to filp–flop will be 0. At each clock edge, 0 travels from left to right after four clock pulses Q output = 0000. It means that if B = data in, data will get inhibited.

**Step IV :** When A = 1, NAND gate is enabled and data input B passes through NAND gate ; inverted. The input data shifted serially in register.

**Step V :** Output of all flip–flops are taken as final output. Output is $Q_1$, not $\overline{Q}$. Output appears at a time ∴ it is parallel output.

**6. Give truth table and logic diagram of twisted ring counter. (Dec. 96, 6 Marks)**

**Ans.:** Refer Section 10.7.

**7. Draw a block diagram of a four bit bi–directional shift register.**

Scanned by CamScanner

**A four bit input 1011 is applied to this shift register. What will be the four different states that occur as the parallel port outputs of this shift register?**

**(i) When 'shift to right' is applied.** **(May 98, 3 Marks)**

**(ii) When 'shift to left' is applied.** **(May 98, 3 Marks)**

**Ans. :** For block diagram and explanation of a four bit bi–directional shift register,

Refer to section 10.10 IC7495, Fig. 10.20 (b). Universal shift register itself acts as a four bit bi-directional shift register.

(i) When shift to right is applied, initially 1011 is applied to inputs A, B, C, D of this shift register.

So at next negative clock edge outputs will be,

(a) $Q_A Q_B Q_C Q_D = 1\ 1\ 0\ 1$

(b) $Q_A Q_B Q_C Q_D = 1\ 1\ 1\ 0$

(c) $Q_A Q_B Q_C Q_D = 1\ 1\ 1\ 1$

(ii) When shift to left is applied, initially inputs applied to A, B, C, D is 1 0 1 1

$\therefore$ At next negative clock edge, outputs will be,

(a) $Q_A Q_B Q_C Q_D = 0\ 1\ 1\ 1$

(b) $Q_A Q_B Q_C Q_D = 1\ 1\ 1\ 1$

(c) $Q_A Q_B Q_C Q_D = 1\ 1\ 1\ 1$

**8. Draw a block diagram and explain how a shift register can be used as counter (any type).**

**(May 98, 5 Marks)**

**Ans.:** Refer Section 10.6.

**9. Explain the working of an universal shift resigster using FF's.** **(Dec. 98, 6 Marks)**

**Ans.:** Refer Section 10.10.

**10. Explain how universal shift register can be used as a parallel to serial and serial to parallel converter.** **(Dec. 98, 10 Marks, June 99, 8 Marks)**

**Ans. :** Refer to Page 10-17

**11. Show that an n–bit binary counter connected to n to $2^n$ line decoder is equivalent to a ring counter with $2^n$ FF. Draw block diagram of both circuits for n = 3.**

**Ans.** Fig. Q. 10.11(a) shows the block diagram of 3 bit binary counter connected to 3 : 8 decoder.

As $n = 3$ , $2^n = 8$

$\therefore$ 8 output lines will be there, for decoder.

**Truth table for Fig. Q.10.11 (a).**

Scanned by CamScanner

| Counter States | | | Output of 3:8 decoder | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Y_7$ | $Y_6$ | $Y_5$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 1**

So here 1 is shifted diagonally and we get respective output of decoder.

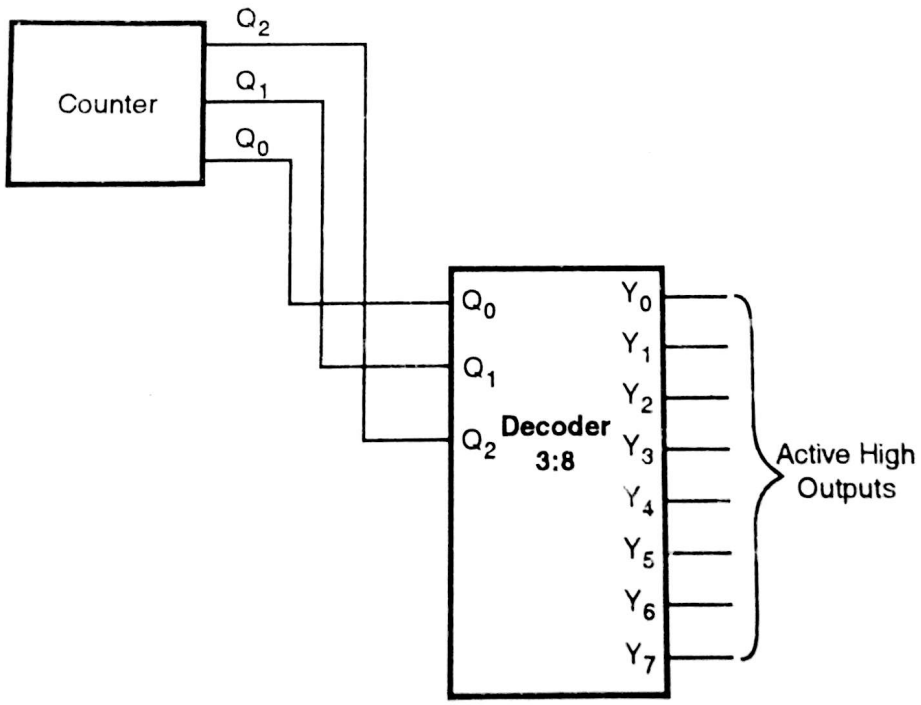



**Fig. 10.11 (a)**

In Fig. Q.10.11 (a), according to counter state, decoder output is selected and is made high. As it is 3 bit counter, it selects output of decoder in sequence from $(0\ 0\ 0)_2$ to $(1\ 1\ 1)_2$ and back to $(0\ 0\ 0)_2$. It appears that logic 1 is roling in a loop. This output is same as output of 8-bit ring counter.

In ring counter, number of FFS and number of states are equal. Therefore we will design 8-bit ring counter using 8 flip-flops.

Scanned by CamScanner

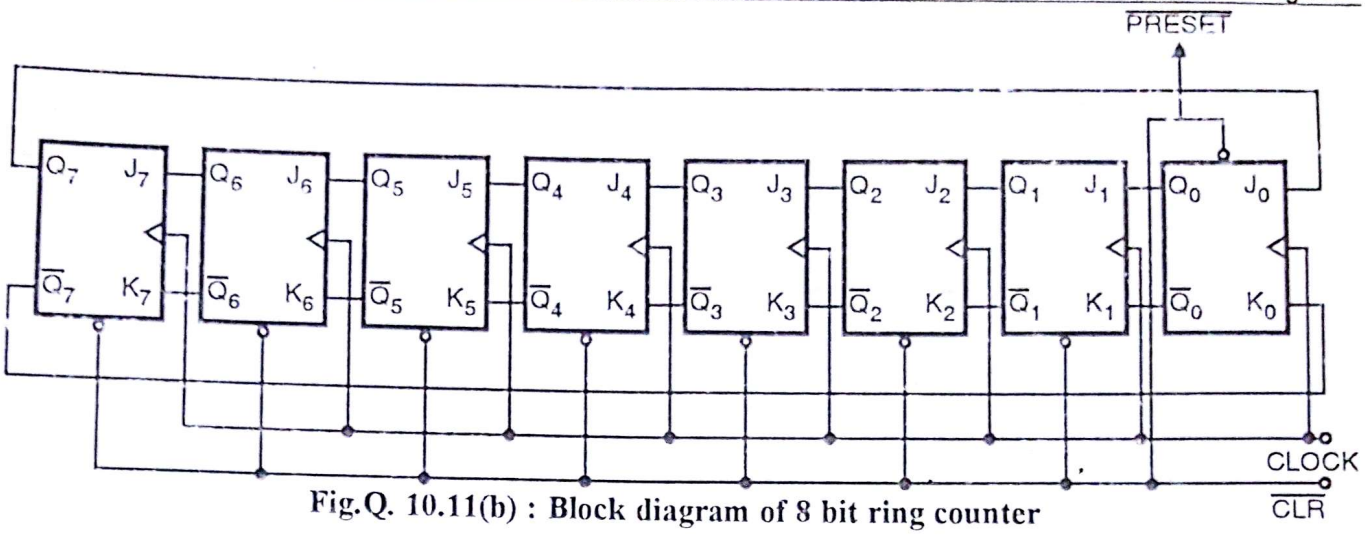



**Fig.Q. 10.11(b) : Block diagram of 8 bit ring counter**

**12. Draw logic diagram of 3–bit bi-directional shift register using S–R F/F. (Dec.99, 8 Marks)**

**Ans.** In three bit bi-directional shift register, we are using three flip–flops. The flip–flops used are R–S flip–flops. An inverter is connected between R and S. This is because, if a data bit x is to be clocked into the flip–flop, the compliment x must be present at the input. For instance, if $\bar{x} = 0$, then R = 0 and S = 1 and 1 will be clocked into the filp–flop when the clock transitions.

The addition of two AND gates and inverter allow the left operation or right operation. It depends on control input M. If control input M is made high, AND gates P are enabled and data is shifted towards right. Whereas if M = 0, then AND gates Q are enabled and data is shifted towards left.

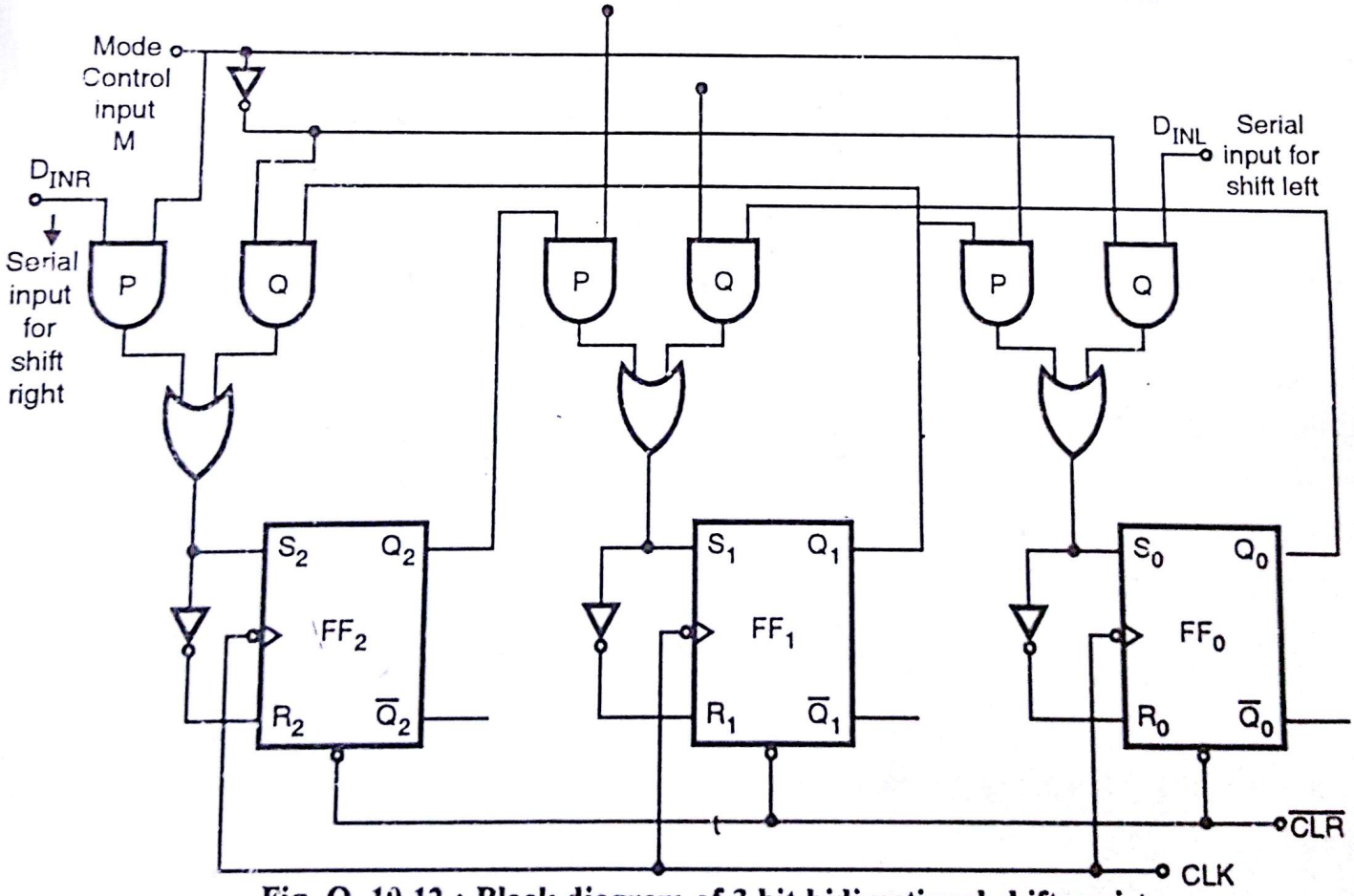



**Fig. Q. 10.12 : Block diagram of 3 bit bidirectional shift register**

**13. Design a mod–6 twisted ring counter using JK FF, with 'power on start' and draw the timing diagram for the same. (Dec. 99, 8 Marks)**

Scanned by CamScanner

**Ans. :** First draw circuit of mod–6 twisted ring counter using JK FF.

As it is a mod 6 twisted ring counter, so we will need three flip–flops.

In twisted ring counter, number of states are double of number of FFs.

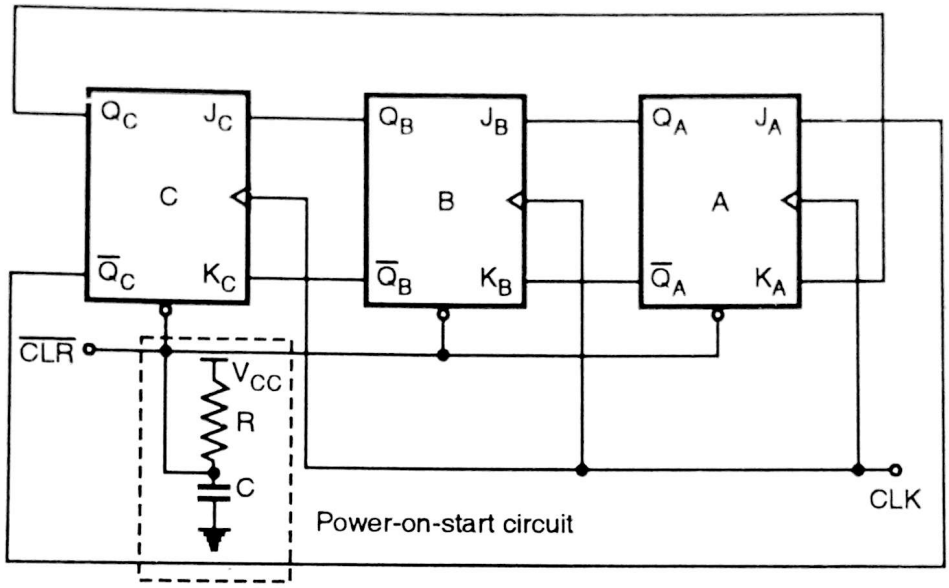



**Fig. Q.10.13 (a) : Diagram of mod–6 twisted ring counter**

Truth table for mod 6 twisted ring counter is as follows :

| CLOCK | $Q_C$ | $Q_B$ | $Q_A$ | State |
|---|---|---|---|---|
| Initially | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 1 | 3 |
| 3 | 1 | 1 | 1 | 4 |
| 4 | 1 | 1 | 0 | 5 |
| 5 | 1 | 0 | 0 | 6 |
| 6 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 1 | 1 |

**Table 2**

As it is a mod–6 counter, 0 to 5 are valid states. So after $6^{th}$ clock pulse counter comes back to initial state.

**Note :** For explaination of **power on start** circuit refer to Chapter 9, Section 9.3.4

Scanned by CamScanner

**Timing diagram for mod–6, twisted ring counter is as follows :**

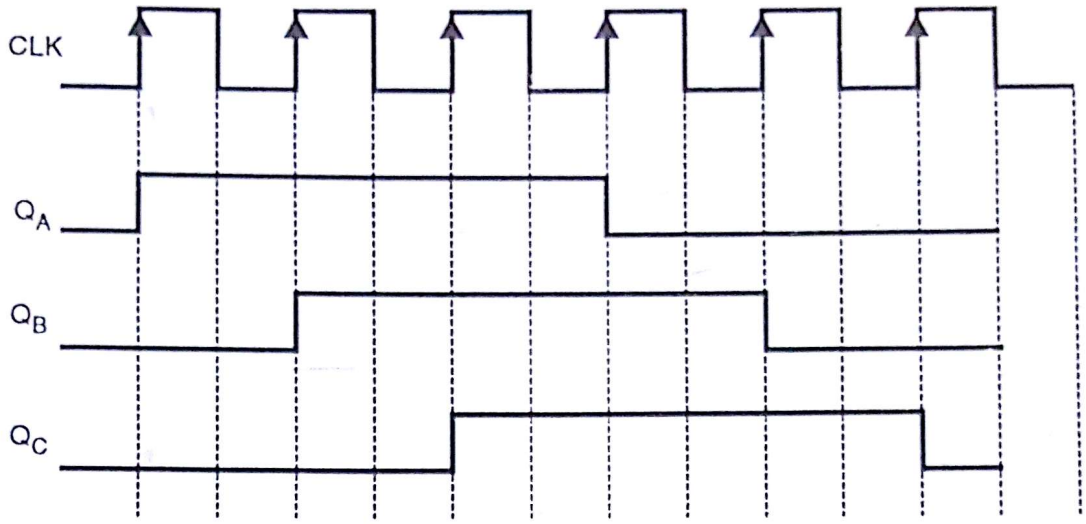



**Fig. Q. 10.13 (b)**

(i) From truth Table 1, we can say that $Q_A$ is high till third clock pulse and at fourth clock edge it falls to zero.

(ii) Similarly $Q_B$ and $Q_C$ becomes high at second and third clock edge respectively and becomes zero at fifth and sixth clock edge respectively.

❑❑❑

Scanned by CamScanner

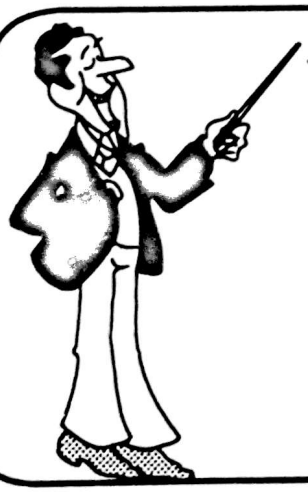
# Analysis and Synthesis of Sequential Circuits

11

**Syllabus :**

State diagrams and tables, Transition table, excitation table and equations. Examples using flip-flops. Simple Synchronous and asynchronous sequential circuit analysis. Construction of state diagram and counter design.

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Introduction | 11.1 | ✓ | – |
| • Sequential Circuit Structure | 11.2 | ✓ | – |
| • State Diagram : Moore Model | 11.3 | ✓ | ✓ |
| • General Procedure for Analysing Synchronous Flip-flop Circuits | 11.4 | ✓ | ✓ |
| • State Reduction and Assignment | 11.5 | – | ✓ |
| • Sequential Circuit Synthesis | 11.6 | – | ✓ |
| • Design of counters. | 11.7 | – | ✓ |
| • Multi-input Synchronous Sequential Circuit Design | 11.8 | – | ✓ |



**Comment**

- Very very very important chapter.
- Considering today's trend i.e. VHDL(VLSI). This chapter is the base for that topic.
- Concentrate more on Sec. 11.5 to Sec. 11.7.
- Clear each and every concept related to moore model.

Scanned by CamScanner

## 11.1 Introduction :

We have studied combinational logic. In that, output solely depends upon input present. In sequential logic circuit, their output signals depends upon past as well as present input signals. Refer Fig. 11.1.

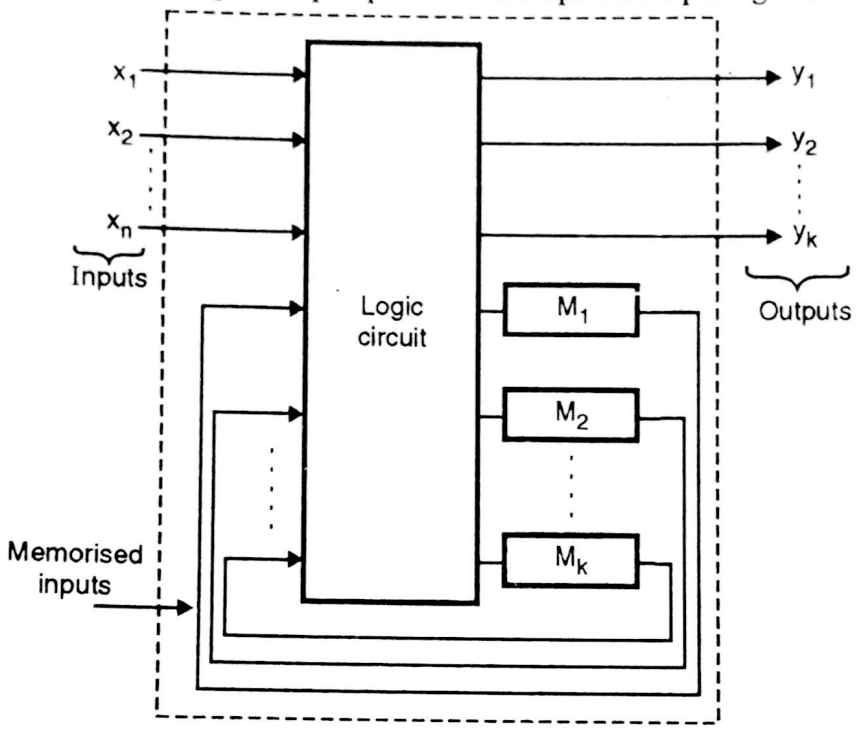



**Fig. 11.1 : General model of sequential system**

From Fig. 11.1 we can conclude that sequential circuit has memory [ $M_1$ to $M_k$], and so can store information derived from past inputs. The two information constitutes the circuit internal state i.e. x input and memorised input. In sequential circuit we have stored sub groups i.e.,

***(1) Synchronous Sequential Circuit***

***(2) Asynchronous Sequential Circuit.***

In synchronous circuit we have clocked flip-flops as memories and use a periodic clock signal to synchronize the times at which all flip-flops change their individual states, and therefore the state of the entire circuit. In asynchronous circuit, clock is not used.

## 11.2 Sequential Circuit Structure :

The way we use gates and flip-flops in combinational logic design, the same way basic building blocks of sequential circuit is gates and flip-flops. One can connect gates and flip flops in many ways to form a sequential circuit. Here flip-flop constitutes memory M and gates forming one or more combinational sub circuit $C_1$, $C_2$, ....., $C_q$. Now let's refer different structure in sequential circuit.

Fig. 11.2 shows sequential circuit, here memory M and combinational circuit C is used. Here we haven't used any feedback. Because of no feedback present, next state is independent of present state. The next state is a function of primary inputs X only and that is determined by input logic $C_1$. Output Z is determined by output logic $C_2$. Sequential circuit of this type are quite limited in their behaviour and can, in most cases, use gated latches instead of flip-flops. More general circuit is shown in Fig. 11.3.

Scanned by CamScanner

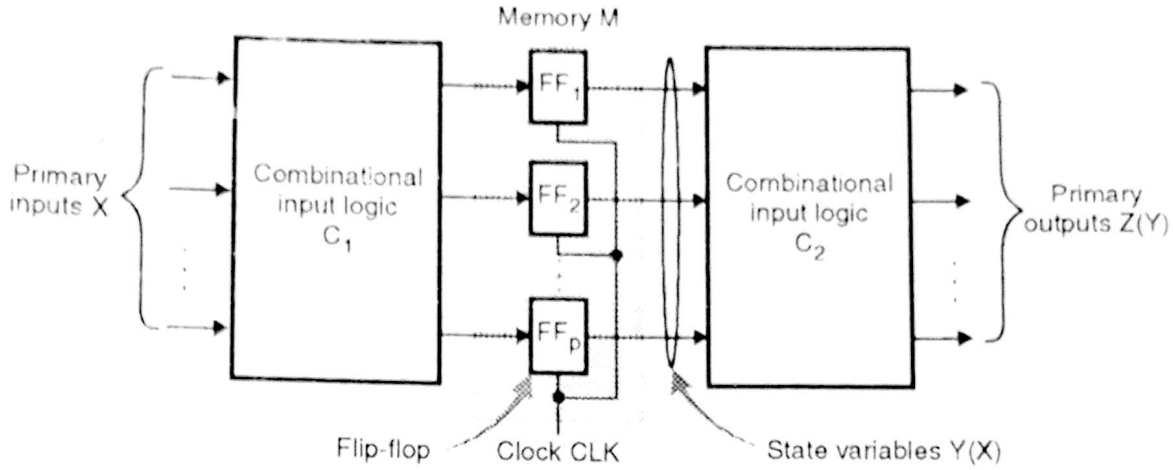



**Fig. 11.2 : Synchronous sequential circuit, with no global feedback**

As shown in Fig. 11.3, we have two models in sequential circuits :

*(1) Moore circuit (model)*

*(2) Mealy circuit (model).*

In Fig. 11.3 (a) feedback is from memory M, to input combinational logic circuit $C_1$. Therefore, next state is function of both, X and Y. In this output logic $C_2$ receives, inputs from M (memory) only. This model is called **'Moore'**, named after Edward F. Moore. ***The most common example for Moore circuit is counters. In counters, output just depends upon present count.***

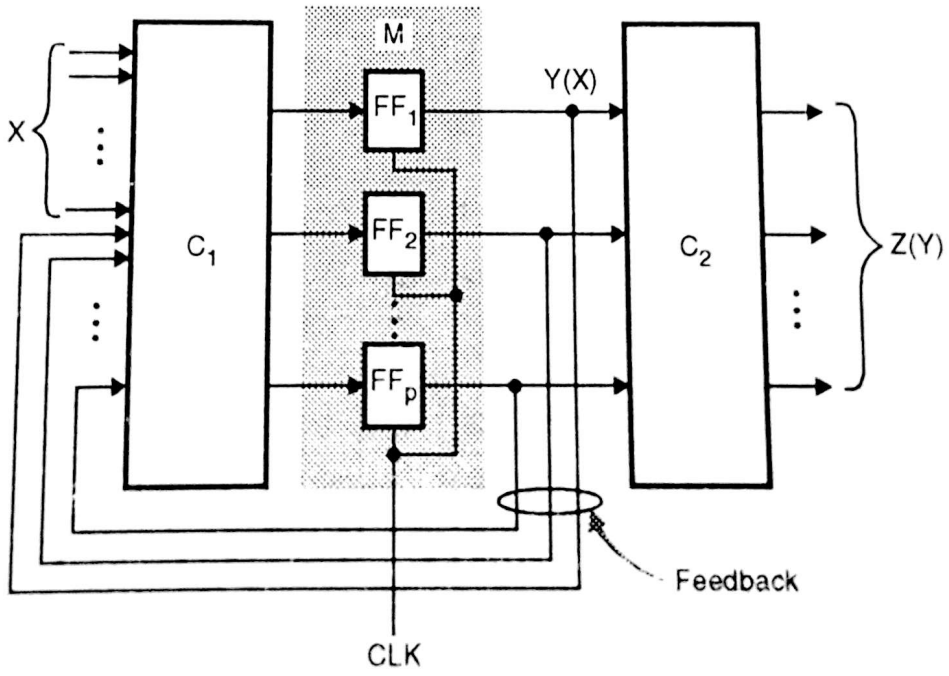



**Fig. 11.3 (a) : Primary output independent of primary inputs (Moore Model)**

Fig. 11.3 (b), shows most general case, here output as well as next state, is function of both, X and Y. This model is called Mealy model named after G.H. Mealy.

Scanned by CamScanner

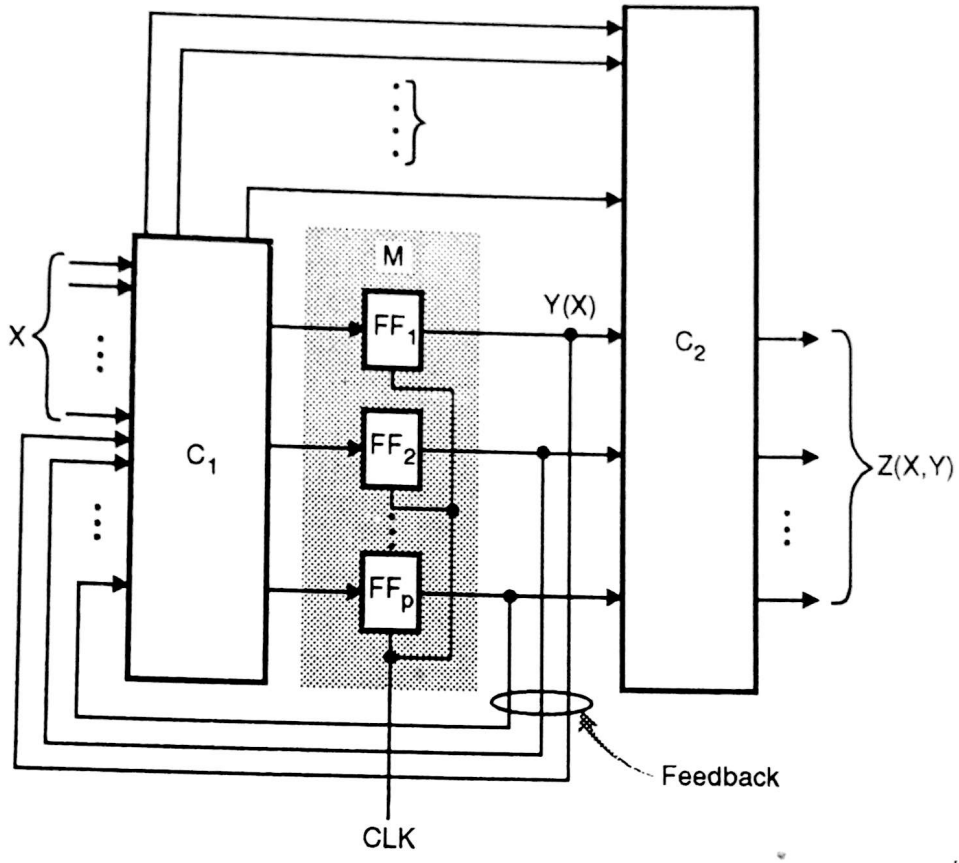



**Fig. 11.3 (b) : A general case (Mealy Model)**

Mealy circuit can be characterised by quinteuple.

$$M = (\Sigma, Q, Z, \delta, \lambda)$$

where, $\Sigma$ = Finite non-empty set of input symbols $x_1, x_2, \ldots\ldots x_l$

$Q$ = Finite non-empty set of states $q_1, q_2, \ldots\ldots q_n$

$Z$ = Finite non-empty set of output symbols $z_1, z_2, \ldots\ldots z_m$.

$\delta$ = Next state function, which maps

$$Q \times \Sigma \longrightarrow Q$$

$\lambda$ = Output function which maps

$$Q \times \Sigma \longrightarrow Z$$

Same way in Moore model, next state and output functions are expressed as,

$$\delta\,(q\,(n), x\,(n)) = q\,(n+1)$$

and $$\lambda\,(q\,(n), x\,(n)) = z\,(n)$$

where, $x\,(n)$ = Present input

$q\,(n)$ = Present state

$q\,(n+1)$ = Next state

$z\,(n)$ = Present output

If we combine $c_1$ and $c_2$ in Fig. 11.3 (b) and the connections between them into single combinational circuit C, we get Huffman model as shown in Fig. 11.4.

Scanned by CamScanner

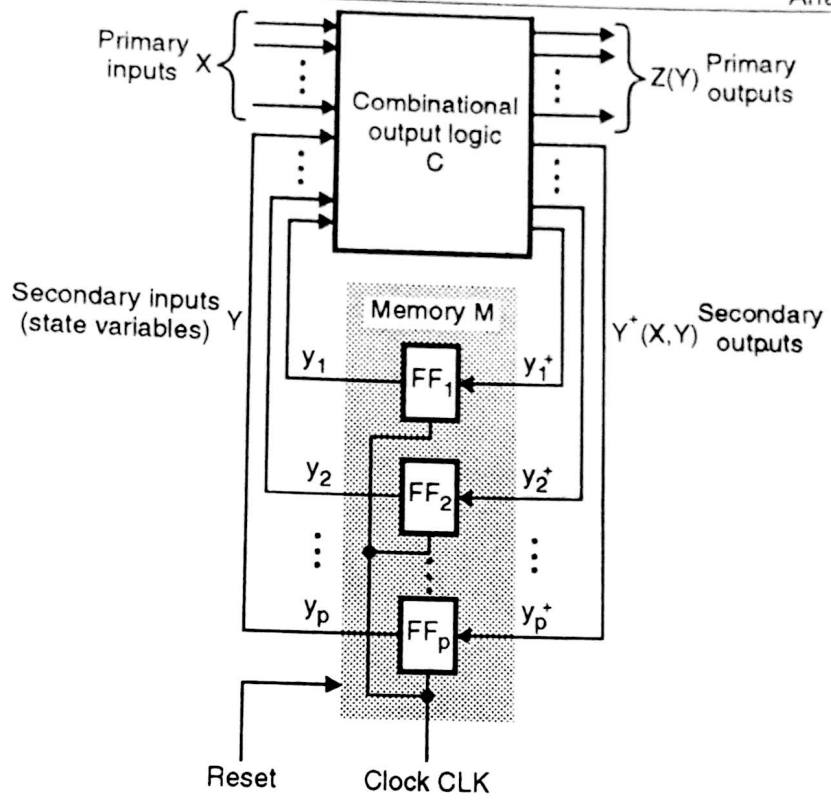



**Fig. 11.4 : Huffman model in sequential circuit**

Main signal types in sequential circuit are defined as follows :

$X = x_1, x_2, \ldots\ldots, x_n$ ⟶ Primary Input Variables

$Z = z_1, z_2, \ldots\ldots, z_m$ ⟶ Primary Output Variables

$Y = y_1, y_2, \ldots\ldots, y_p$ ⟶ Secondary Input Variables OR (internal) State Variable

$Y^+ = y_1^+, y_2^+, \ldots\ldots\ldots y_p^+$ ⟶ Next State Outputs

Therefore, Mealy and Huffman models of sequential circuits are essentially the same. The structure in Fig. 11.3 (b) and Fig. 11.4 covers all the main synchronous sequential circuits.

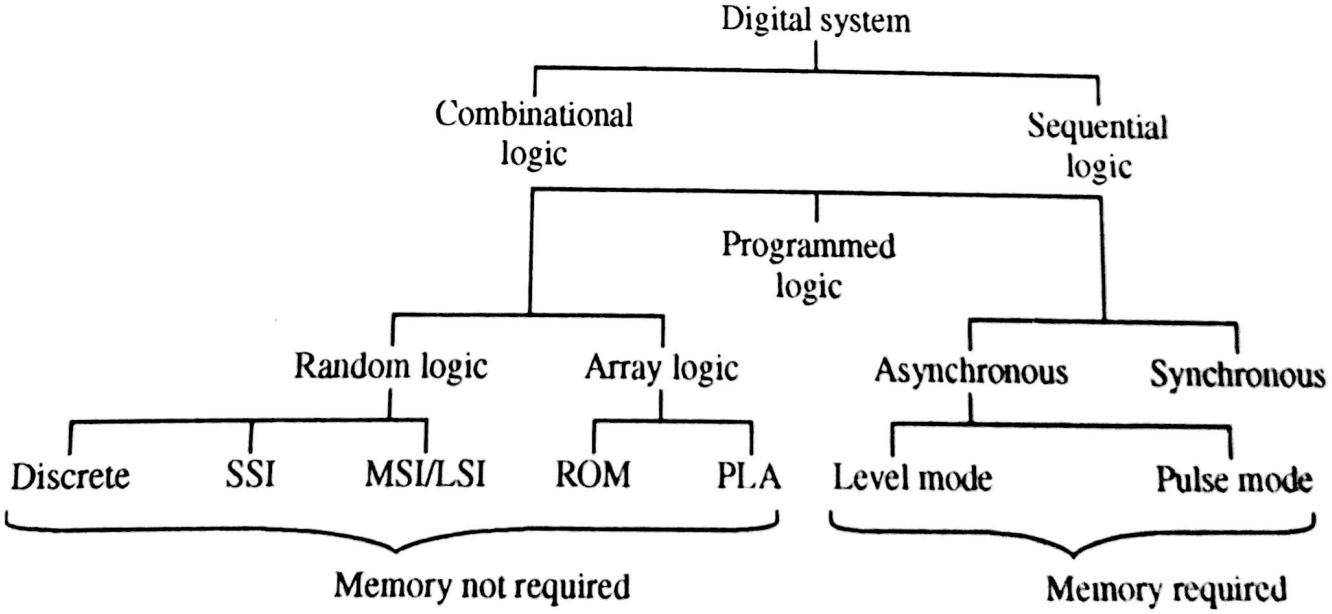



**Fig. 11.5 : Overview of digital system**

Scanned by CamScanner

**Note :** Figs. 11.3 (a) and (b), Fig. 11.4 are synchronous sequential circuits. To have asynchronous simply remove clock signal.

At this state if you try to overview full digital system it will be as shown in Fig. 11.5.

**Important Point :**

Behaviour of any sequential circuit can be described by sequence of events which occur at discrete instants i.e. at time $t_1$, $t_2$, $t_3$ and so on. We are aware with the fact that output of sequential circuit depends upon input x (t) and state of the circuit. It may happen that circuit might have infinite varieties of possible histories. To memorize these histories we require memory of infinite size. It is impossible to design or realise circuit with infinite storage capabilities. Therefore in many cases or all the cases you will find, we use finite memory elements. Therefore we memorise finite post states of the circuit. Therefore normally circuits in Fig. 11.3 and 11.4 are referred as ***finite-state sequential circuits***.

## 11.3 State Diagram : Moore Model :

State diagrams which we had studied in Flip-flop chapter, were for Mealy model. Presentation of state diagram will be slightly different for Moore model.

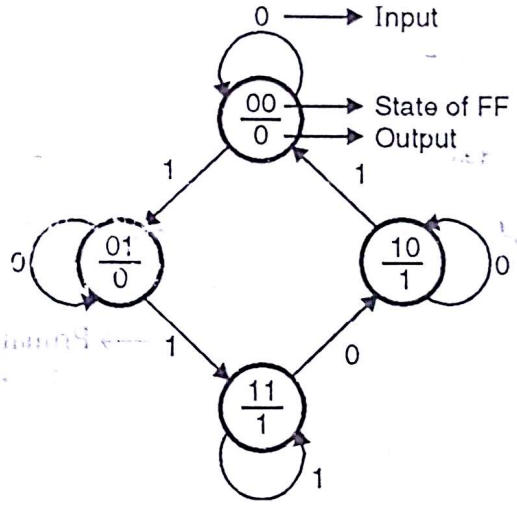



**Fig. 11.6 : Moore model state diagram (An example)**

As shown in Fig. 11.6, state transition arrow is labeled with only one variable, and that is, state of the input that causes transition. Output state is indicated within the circle, below the present state because in Moore model, output depends only upon present state and not on the input.

One more important difference between Mealy and Moore model. In Moore model, the output does depends on the input, since the state depends on the input, but the effect on the output is delayed untill after the next clock.

## 11.4 General Procedure for Analysing Synchronous Flip-flop Circuits :

Under this section we are going to analyse, given sequential circuit. Analysis means we are interested to find state table, state diagram for a given sequential circuit.

***Hint :*** Remember operation of flip flops and its characteristic equation.

Fig. 11.7 specifies flowchart for analysing given sequential circuit. Let's take one example for analysis.

Scanned by CamScanner

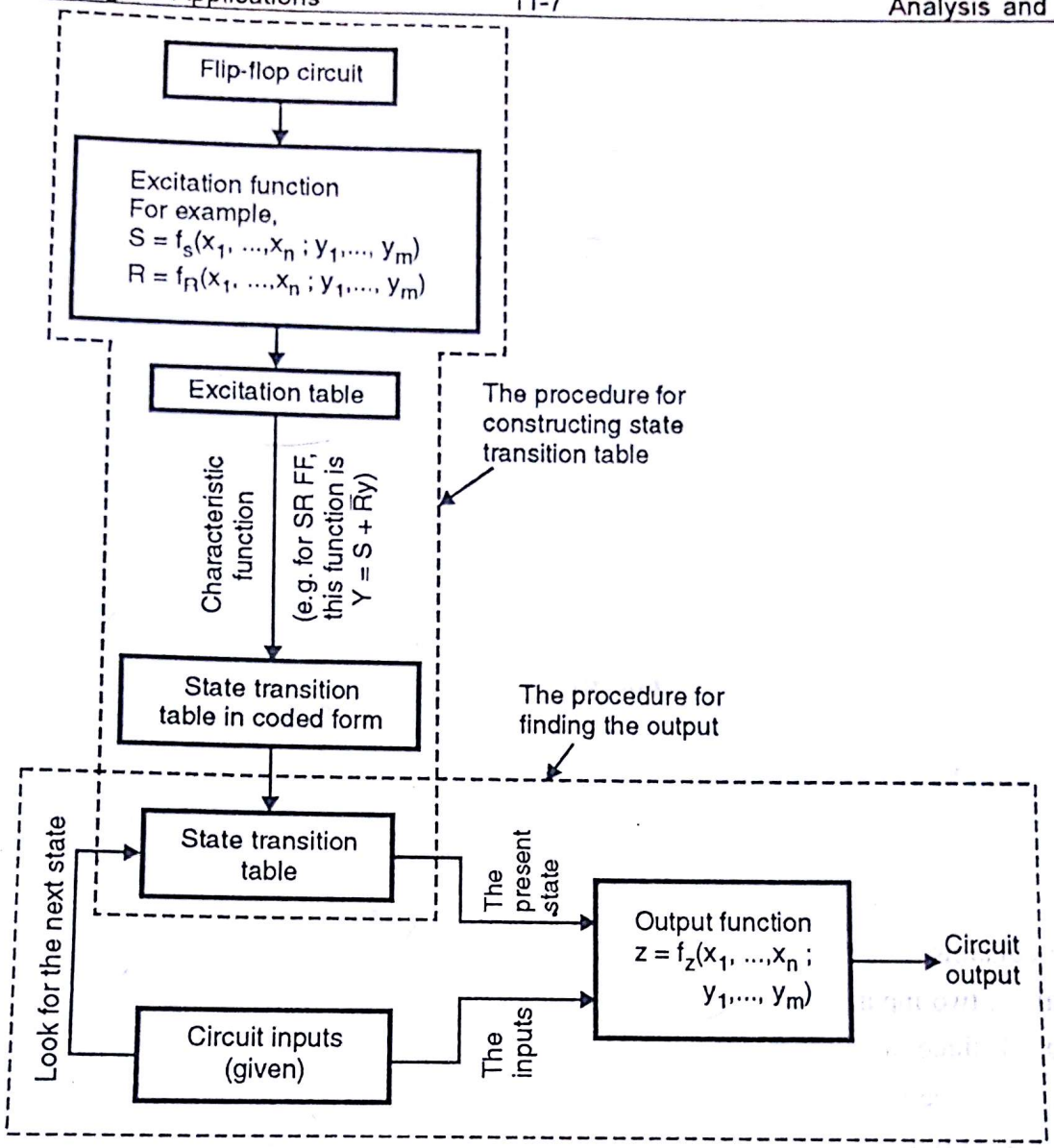



**Fig. 11.7 : General analysis procedure**

**Ex. 1 :**

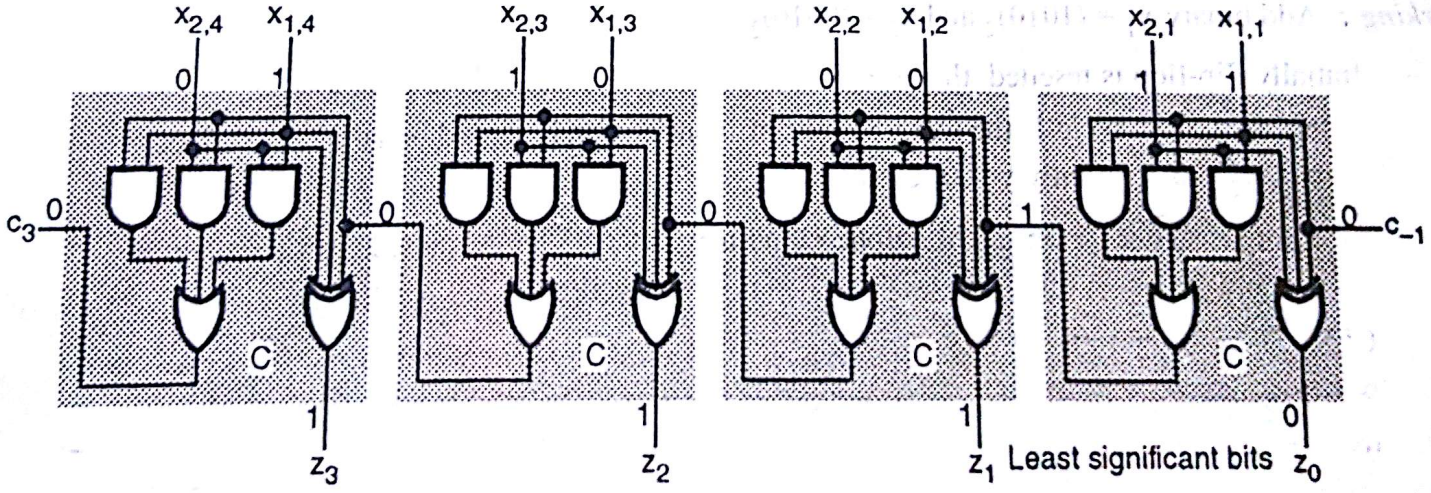



**Fig. Ex. 11.1(a)**

Scanned by CamScanner

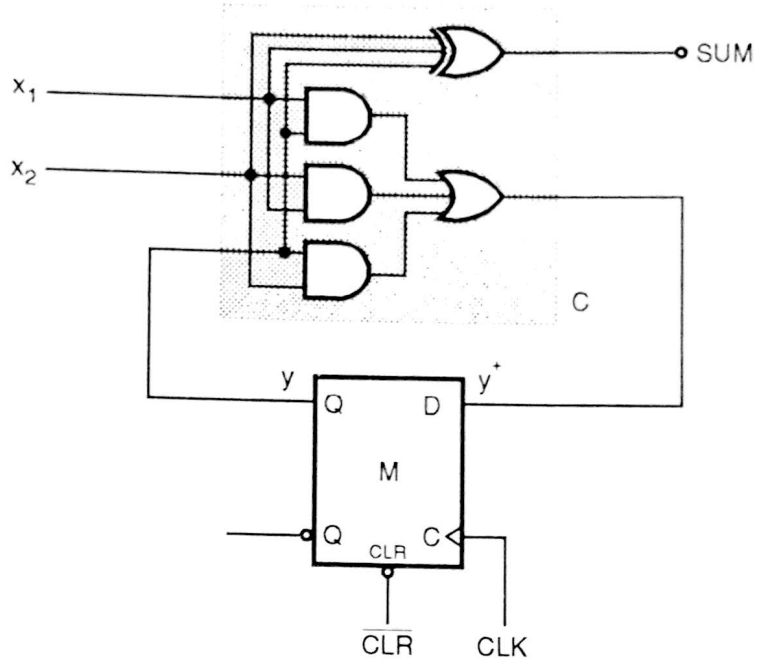



**Fig. Ex. 11.1(b)**

Fig. Ex. 11.1(a) shows parallel adder i.e. combinational circuit. It performs 4 bit binary addition. Fig. Ex. 11.1(b) shows sequential version of the same. Perform 4 bit addition, using the same circuit. Draw timing waveforms also.

**Soln. :** (1) The given circuit is nothing but serial adder.

(2) It contains :

(a) 3, two input AND gate

(b) 1, three input XOR gate

(c) 1, three input OR gate

(d) 1, D type flip flop

(3) Flip-flop memorises 'Carry bit' *C*.

(4) Circuit has two inputs $x_1$ and $x_2$, i.e. operands $x_1$ and $x_2$ bits should be applied bit by bit, with LSB first.

***Working :*** Add binary $x_1 = (1010)_2$ and $x_2 = (0010)_2$

- Initially flip-flop is resetted, therefore y = 0
- After the start of cycle 1, the two LSB bits of input operands should be applied to adder. $x_1 = 0$, $x_2 = 0$. Presently y = 0, c = 0

  Resulting sum bit '0' appears at output z, and new carry bit '0' is applied to D FF input $y^+$.
- Now apply 2nd CLK edge. Presently $x_1 = 1$, $x_2 = 1$. New carry bit get loaded at positive edge of CLK. Therefore y = 0.

  Resulting sum bit '0' will appear at output Z, and new carry bit '1' will appear at input to D FF, input $y^+$.
- Now when positive edge of 3rd CLK appears y = 1, i.e. nothing but $y^+$.
  Next two operand bits will appear at $x_1$ and $x_2$. $x_1 = 0$, $x_2 = 0$ and y = 1.

Scanned by CamScanner

Therefore resulting sum bit '1' appears at output Z, and new carry bit '0' appears at $y^+$ input of D FF.

New carry bit will get loaded in D FF, when 4th CLK pulse appears, $y = 0$.

Next two operands bits will appear at $x_1$ and $x_2$.

$x_1 = 1$, $x_2 = 0$ and $y = 0$.

Therefore resulting sum bit '1' will appear at output Z.

Finally carry bit will appear at $y^+$ input of D FF i.e. '0'.

As shown we get sequential answer :

0 0 1 1 ∴ $(1100)_2$ is final answer

↑ ↑

LSB MSB

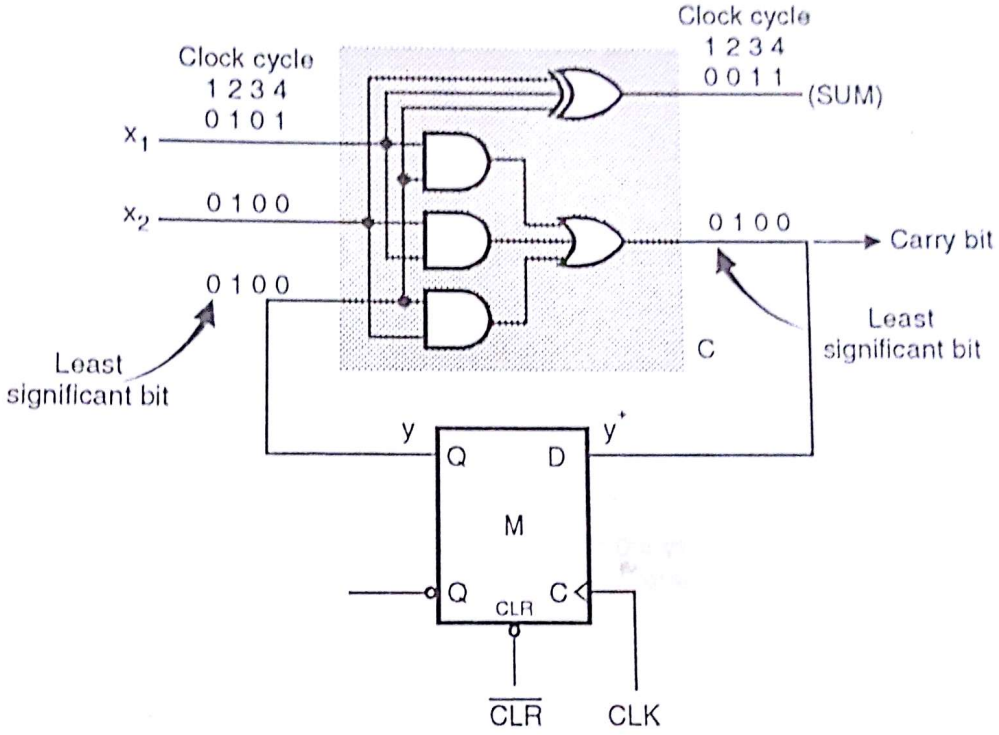



**Fig. Ex. 11.1 (c) : Serial adder working**

**Precautions :** For proper operation of the circuit :

(1) Initial value of the state variable y should be '0' before the first LSB bits of $x_1$ and $x_2$ are applied.

(2) Clear D FF, before each new n bit addition begins.

Now let's prepare state table.

The equations are, $z = x_1 \oplus x_2 \oplus y$

$y^+ = x_1x_2 + x_1y + x_2y$

Now for each $x_1$ $x_2$ and y, we have to calculate $y^+$ and z.

Let's say $X = x_1\ x_2 = 11$ and $Y = y = 0$

∴ $Z = x_1 \oplus x_2 \oplus y$

$= 1 \oplus 1 \oplus 0 = 0$

$y^+ = x_1\ x_2 + x_1\ y + x_2\ y$

$= 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 0 = 1$

Scanned by CamScanner

**State table**

| Present state (y) | Next state | | | | Output ($y^+$) | | | |
|---|---|---|---|---|---|---|---|---|
| | $x_1 x_2$ 0 0 | $x_1 x_2$ 0 1 | $x_1 x_2$ 1 1 | $x_1 x_2$ 1 0 | $x_1 x_2$ 0 0 | $x_1 x_2$ 0 1 | $x_1 x_2$ 1 1 | $x_1 x_2$ 1 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

If we consider $q_0$ = 0 state and $q_1$ = 1 state we will get state table as

| Present state (y) | Next state | | | | Output ($y^+$) | | | |
|---|---|---|---|---|---|---|---|---|
| | $x_1x_2$ 0 0 | $x_1x_2$ 0 1 | $x_1x_2$ 1 1 | $x_1x_2$ 1 0 | $x_1x_2$ 0 0 | $x_1x_2$ 0 1 | $x_1x_2$ 1 1 | $x_1x_2$ 1 0 |
| 0 | $q_0$ | $q_0$ | $q_1$ | $q_0$ | 0 | 1 | 0 | 1 |
| 1 | $q_0$ | $q_1$ | $q_1$ | $q_1$ | 1 | 0 | 1 | 0 |

Let's draw state diagram for the same.

Now let's draw state diagram for the same.

(1) Consider initial state y = 0 ($q_0$).

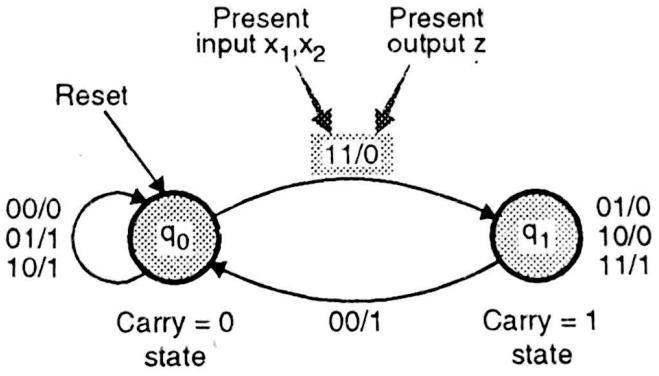



**Fig. 11.8 : State diagram for serial adder**

(2) When state is y = 0, If $x_1$ $x_2$ is 00/0, 01/1 or 10/1, circuit remains in state y = 0.

Here

| | | | | |
|---|---|---|---|---|
| 0 | 0 | / | 0 |
| 0 | 1 | / | 1 |
| 1 | 0 | / | 1 |

↓ Present input $x_1$ $x_2$ ↓ Present output z

(3) To switch over from y = 0, to y = 1, $x_1x_2/z$ is 11/0.

(4) Now circuit will be at state y = 1, till $x_1x_2/z$ is 01/0, 10/0 or 11/1.

(5) Circuit changes state from y = 1 to y = 0 when $x_1x_2/z$ is 00/1.

**Note :** The circuit is Mealy circuit. Waveforms given below will clear your idea for the same.

The waveforms are drawn for, $x_1$ = 1001 and $x_2$ = 0101.

| | | | | | |
|---|---|---|---|---|---|
| $x_1$ | = | 1 | 0 | 0 | 1 |
| + | | | | | |
| $x_2$ | = | 0 | 1 | 0 | 1 |
| | | 1 | 1 | 1 | 0 |

Fig. 11.9(a) represents timing cycles for the same.

Scanned by CamScanner

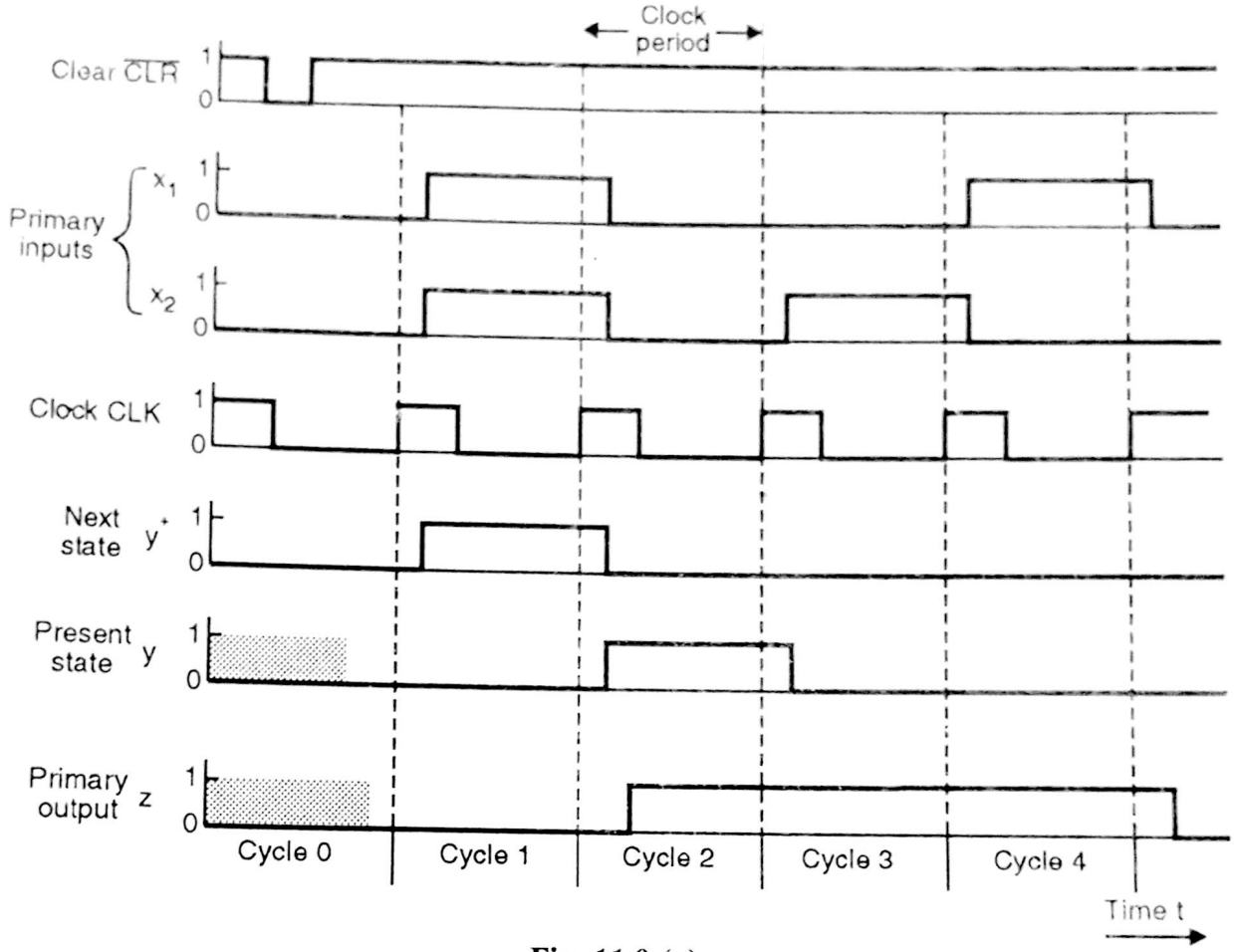



**Fig. 11.9 (a)**

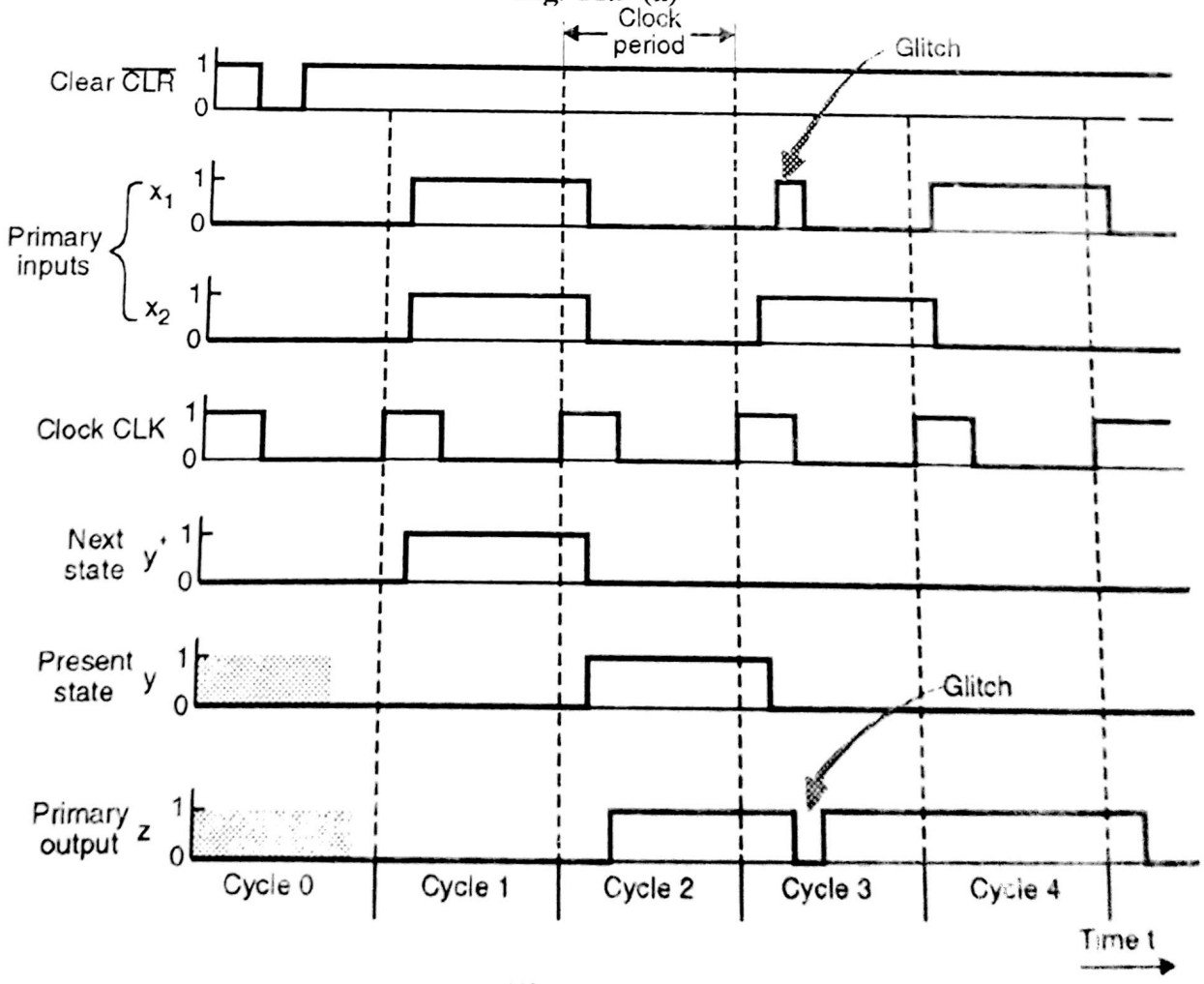



**Fig. 11.9(b)**

Scanned by CamScanner

**Note :** Refer Fig. 11.9.

(1) In cycle 0, everything gets cleared.

(2) In cycle 1, $x_1 = x_2 = 1$. Therefore $y^+ = 1$ and $z = 0$. Status of $y^+$ will not immediately appear at y. It will appear at next positive CLK edge.

(3) In cycle 2, status of $y^+$ appears at output y of D FF. Now $x_1 = x_2 = 0$, and $y = 1$. Therefore $y^+ = 0$ and $z = 1$.

(4) In cycle 3, status of $y^+ = 0$, will appear at output y of D FF. Now $x_1 = 0$ and $x_2 = 1$, with $y = 0$, we get, $y^+ = 0$ and $z = 1$.

(5) Finally in cycle 4, $y^+ = 0$., $y = 0$. $x_1 = 1$ and $x_2 = 0$, $\therefore$ $y^+ = 0$ and $z = 1$.

This describes full working of circuit.

*Observation :*

(1) Output of D FF changes only when CLK arrives. It won't change, eventhough input $y^+$ changes.

(2) Output of combinational circuit i.e. z changes, as soon as either $x_1$ or $x_2$ or y changes. In general, a primary output signal $z_k$ is function of $x_j$, so the change in $x_j$ immediately affects $z_k$.

To understand point 2 of observation, refer Fig. 11.9(b).

Refer Fig. 11.9(b). As shown in Fig. 11.9(b), a narrow 1 pulse in $x_1$, which is an externally generated glitch in the middle of cycle 3, produces a corresponding glitch in form of '0' pulse in z. Because at that instant,

$x_1 = 1$, $x_2 = 1$, $y = 0$ $\therefore$ $x_1 + x_2 + y = z = 1 + 1 + 0 = 0$ with carry.

$\therefore$ We get $z = 0$. Glitch remains there for very short span. Therefore this 1 pulse (glitch) does not affect y, because it has disappeared by the time the flip flop is triggered at the start of cycle 4. Therefore to produce a single well defined output value or state change in any CLK period, each data input should be held constant for most of the clock period. Therefore for proper operation of the circuit, all input signals $x_j$ should be well defined '0' or '1', pulses. Normally input should be maintained for full one clock period. More detail regarding the same will be given to you afterwards.

**Ex. 2 :** For given sequential circuit, draw state diagram and state table. Refer Fig. Ex. 11.2(a).

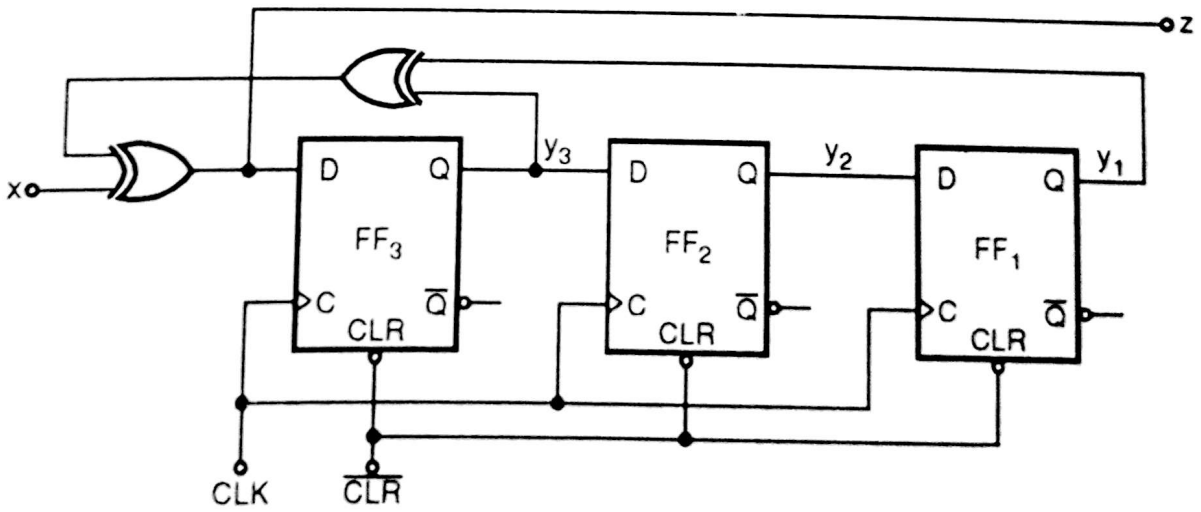



**Fig. Ex. 11.2(a) : Sequential circuit**

**Soln. :** (1) First we will write boolean expressions for the same.

Scanned by CamScanner

$$z = x \oplus y_1 \oplus y_3$$
$$y_1^+ = y_2$$
$$y_2^+ = y_3$$
$$y_3^+ = x \oplus y_1 \oplus y_3 = z$$

(2) First we will write state transition table.
In our problem,

$z$ = Output
$y_1\, y_2\, y_3 \longrightarrow$ Provides previous state
$y_1^+\, y_2^+\, y_3^+ \longrightarrow$ Provides next state
$x$ = Input

$\therefore$ We can write, $y_1\, y_2\, y_3 \xrightarrow{x/z} y_1^+\ y_2^+\ y_3^+$

$\therefore$ We get table.

| Present state | | | Next state ($y_1^+\ y_2^+\ y_3^+$) | | | | | | Ouput (z) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | x = 0 | | | x = 1 | | | x = 0 | x = 1 |
| $y_1$ | $y_2$ | $y_3$ | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |

(3) Next step we follow is simply rearrange transition table to get state table. Only the important point here is we define labels to the state.

$\therefore$ State

| | | | |
|---|---|---|---|
| 0 0 0 | $\rightarrow$ | $S_0$ | 1 0 0 $\rightarrow$ $S_4$ |
| 0 0 1 | $\rightarrow$ | $S_1$ | 1 0 1 $\rightarrow$ $S_5$ |
| 0 1 0 | $\rightarrow$ | $S_2$ | 1 1 0 $\rightarrow$ $S_6$ |
| 0 1 1 | $\rightarrow$ | $S_3$ | 1 1 1 $\rightarrow$ $S_7$ |

**State table**

| Present state | Next state | | Output | |
|---|---|---|---|---|
| | x = 0 | x = 1 | x = 0 | x = 1 |
| $S_0$ | $S_0$ | $S_1$ | 0 | 1 |
| $S_1$ | $S_3$ | $S_2$ | 1 | 0 |
| $S_2$ | $S_4$ | $S_5$ | 0 | 1 |
| $S_3$ | $S_7$ | $S_6$ | 1 | 0 |
| $S_4$ | $S_1$ | $S_0$ | 1 | 0 |
| $S_5$ | $S_2$ | $S_3$ | 1 | 1 |
| $S_6$ | $S_5$ | $S_4$ | 1 | 0 |
| $S_7$ | $S_6$ | $S_7$ | 0 | 0 |

Scanned by CamScanner

(4) The last step is to draw state diagram from state table

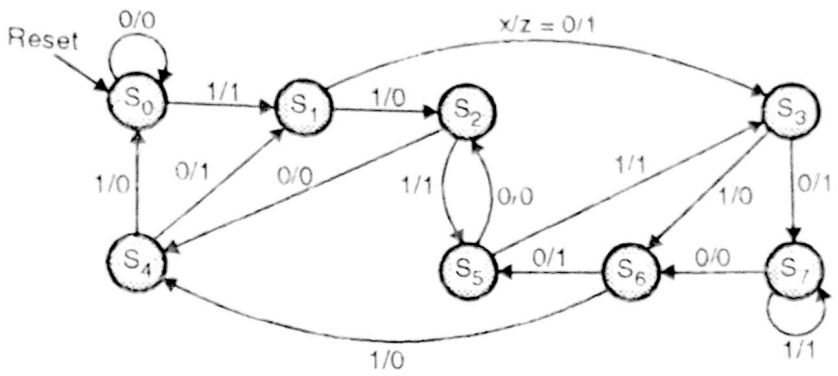



**Fig. Ex. 11.2(b) : State diagram**

Note : Till this point if you have observed carefully, we have used DFF. DFF is simple flip-flop, because, whatever is input, that will be the output. But for TFF, JK, FF, SRFF, this is not the case. We have to take into account input to the FF, and then we have to decide next state.

Let's start with simple TFF.

**Ex. 3 :** Consider TFF circuit as shown in Fig. Ex. 11.3(a). Find out state table and diagram for the same.

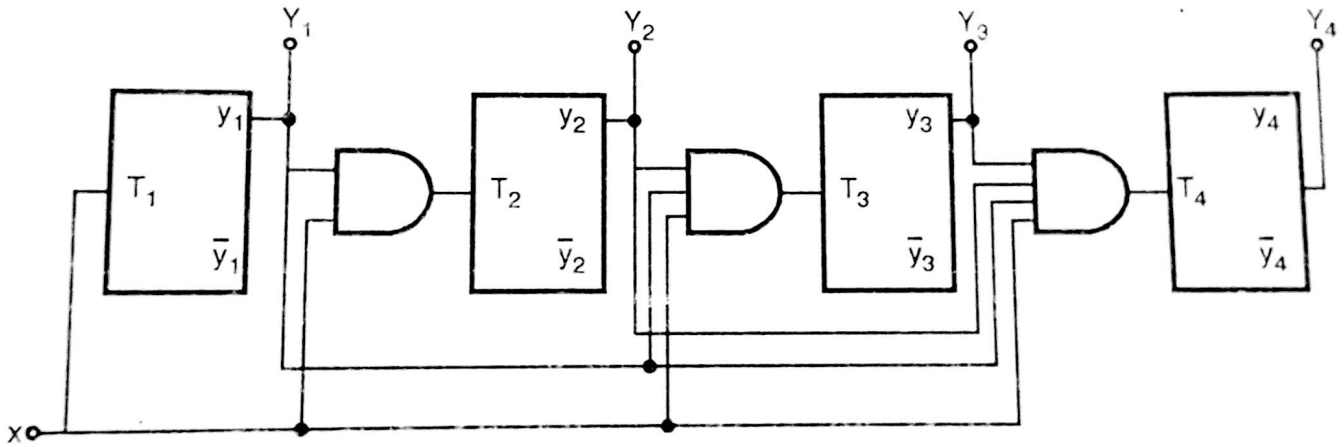



**Fig. Ex. 11.3(a) : T FF sequential circuit**

**Soln. :** (1) First step is, we will write expressions for $T_1$, $T_2$, $T_3$ and $T_4$

$$T_1 = x$$

$$T_2 = xy_1$$

$$T_3 = xy_1y_2$$

$$T_4 = xy_1y_2y_3$$

(2) Output states are $Y_4$, $Y_3$, $Y_2$ and $Y_1$. Input line is x.

$\therefore$ We will first write excitation table.

Scanned by CamScanner

| Present state | | | | Flip-flop inputs | | | | | | | | Next state ($Y_i = T_i \oplus y_i$) i = 1, 2, 3, 4 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | x = 0 | | | | x = 1 | | | | x = 0 | | | | x = 1 | | | |
| $y_4$ | $y_3$ | $y_2$ | $y_1$ | $T_4$ | $T_3$ | $T_2$ | $T_1$ | $T_4$ | $T_3$ | $T_2$ | $T_1$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_4$ | $Y_3$ | $Y_2$ | $Y_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

∴ It is module 16 counter. Secondly, if input x = 0, counter latches count and if x = 1, counter is free running.

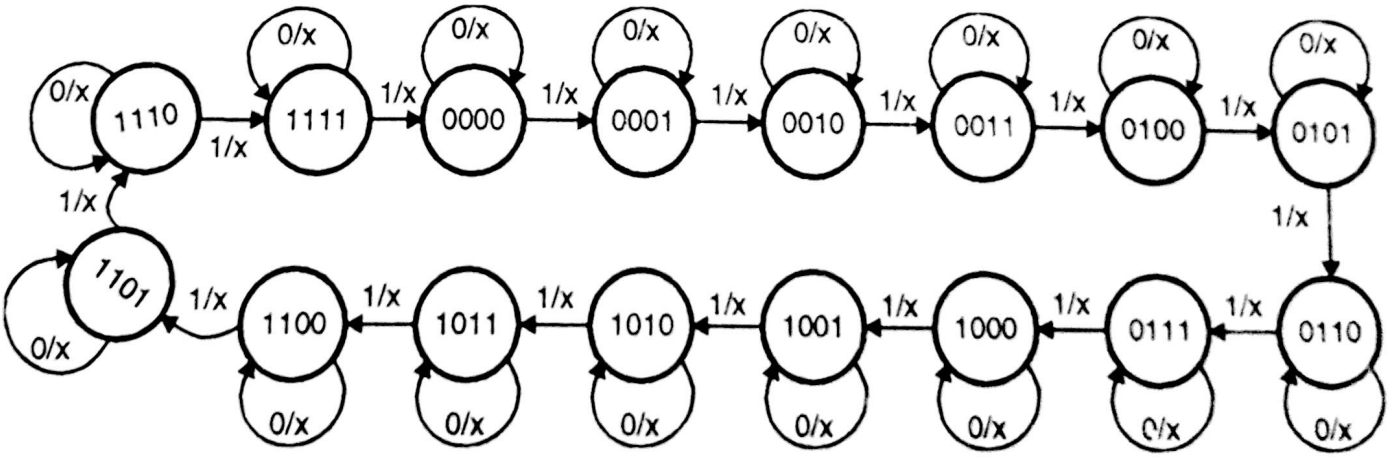



**Fig. Ex. 11.3(b) : State diagram**

**Ex. 4 :** For given circuit as shown in Fig. Ex. 11.4(a) find out state diagram and state table

Scanned by CamScanner

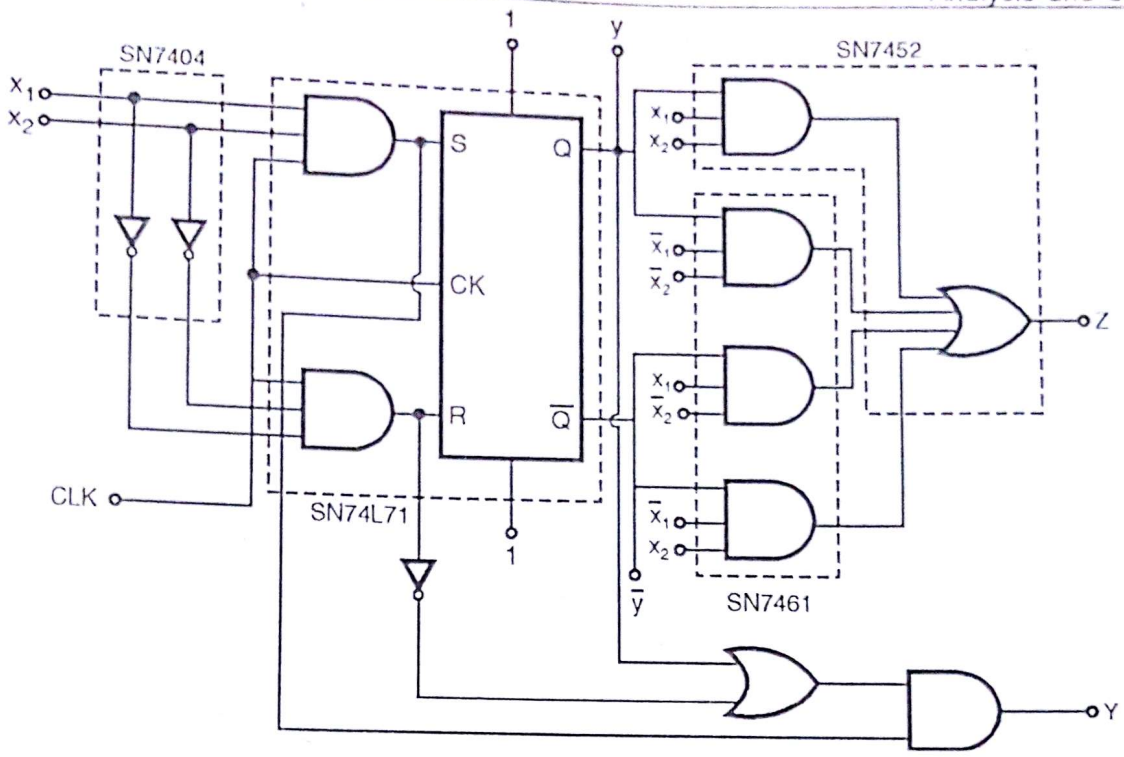



**Fig. Ex. 11.4(a)**

**Soln. :** Observing the circuit you will find that it is nothing but serial adder only. But the circuit uses SR instead of D FF as in Ex. 1. Now let's analyse the same, We should get same state table and state diagram, as in Ex. 1.

**Step 1 :** Firstly we have to find excitation functions in terms of circuit inputs and flip-flop outputs from given circuit diagram.

$S = x_1 x_2 \qquad R = \bar{x}_1 \bar{x}_2$

**Step 2 :** Now let's obtain binary transition table, from excitation table. Binary transition table consists of Y and Z. Where $Y = S + \bar{R}y$ and $Z = x_1 x_2 y + \bar{x}_1 \bar{x}_2 y + x_1 \bar{x}_2 \bar{y} + \bar{x}_1 x_2 \bar{y}$

Now we will combine all the states, as well as output and draw final state table.

| Present state (y) | Flip-flop inputs (S/R) $S = x_1 x_2$; $R = \bar{x}_1 \bar{x}_2$ | | | | Next state $Y = S + \bar{R} Y$ | | | | Output z $Z = x_1 x_2 y + \bar{x}_1 \bar{x}_2 y + x_1 \bar{x}_2 \bar{y} + \bar{x}_1 x_2 \bar{y}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $x_1 x_2$ 0 0 | $x_1 x_2$ 0 1 | $x_1 x_2$ 1 1 | $x_1 x_2$ 1 0 | $x_1 x_2$ 0 0 | $x_1 x_2$ 0 1 | $x_1 x_2$ 1 1 | $x_1 x_2$ 1 0 | $x_1 x_2$ 0 0 | $x_1 x_2$ 0 1 | $x_1 x_2$ 1 1 | $x_1 x_2$ 1 0 |
| 0 | 0/1 | 0/0 | 1/0 | 0/0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0/1 | 0/0 | 1/0 | 0/0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

To check how we got output Y and Z, we will take one example.

$x_1 = 1$ ; $x_2 = 1$ and $y = 0$

$\therefore \qquad Y = S + \bar{R} y$

$= x_1 x_2 + \overline{\bar{x}_1 \bar{x}_2}\, y = 1 + \bar{0} \cdot 0 = 1$

Scanned by CamScanner

$$Z = x_1 x_2 y + \bar{x}_1 \bar{x}_2 y + x_1 \bar{x}_2 \bar{y} + \bar{x}_1 x_2 \bar{y}$$
$$= 1 \cdot 1 \cdot 0 + \bar{1} \cdot \bar{1} \cdot 0 + 1 \cdot \bar{1} \cdot \bar{0} + \bar{1} \cdot 1 \cdot \bar{0}$$
$$= 0$$

If we take state '0' as $q_0$ and state 1 as $q_1$ we get transition table as,

| Present state (y) | Flip-flop inputs (S/R) $S = x_1 x_2$; $R = \bar{x}_1 \bar{x}_2$ | | | | Next state $Y = S + \bar{R} Y$ | | | | Output Z $Z = x_1 x_2 y + \bar{x}_1 \bar{x}_2 y + x_1 \bar{x}_2 \bar{y} + \bar{x}_1 x_2 \bar{y}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $x_1 x_2$ 0 0 | $x_1 x_2$ 0 1 | $x_1 x_2$ 1 1 | $x_1 x_2$ 1 0 | $x_1 x_2$ 0 0 | $x_1 x_2$ 0 1 | $x_1 x_2$ 1 1 | $x_1 x_2$ 1 0 | $x_1 x_2$ 0 0 | $x_1 x_2$ 0 1 | $x_1 x_2$ 1 1 | $x_1 x_2$ 1 0 |
| 0 | 0/1 | 0/0 | 1/0 | 0/0 | $q_0$ | $q_0$ | $q_1$ | $q_0$ | 0 | 1 | 0 | 1 |
| 1 | 0/1 | 0/0 | 1/0 | 0/0 | $q_0$ | $q_1$ | $q_1$ | $q_1$ | 1 | 0 | 1 | 0 |

**State diagram :**

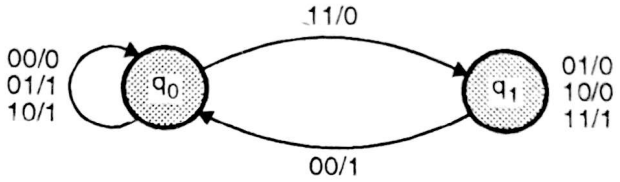



**Fig. Ex. 11.4(b)**

**Note :** In next example drawing pattern of state table is slightly changed, shown as 2D array.

**Ex. 5 :** Consider JK FF circuit shown in Fig. Ex. 11.5(a). Find out state table and state diagram for the same.

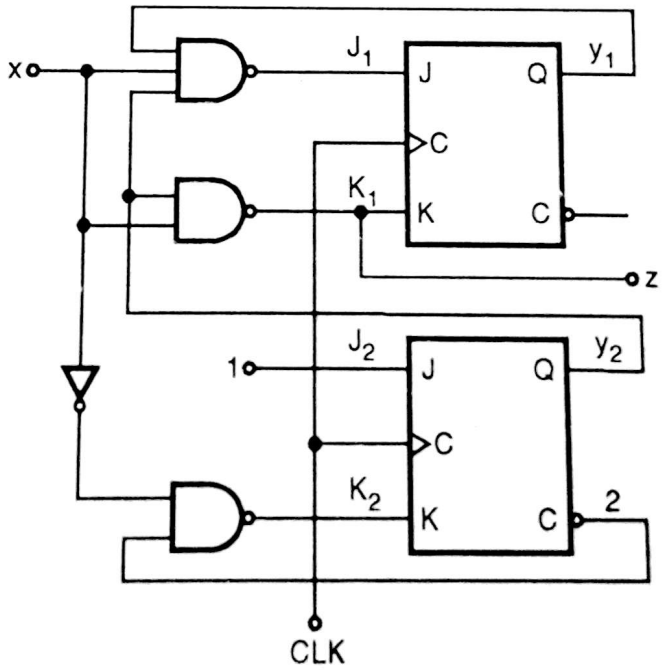



**Fig. Ex. 11.5(a) : Sequential circuit containing JK FF**

**Soln. :** (1) We will write expressions,

Scanned by CamScanner

$$Z = \bar{x} + \bar{y}_2$$

$$J_1 = \bar{x} + \bar{y}_1 + \bar{y}_2$$

$$K_1 = \bar{x} + \bar{y}_2$$

$$J_2 = 1$$

$$K_2 = x + y_2$$

(2) Characteristic equation for JK is,

$$Q(n+1) = J(n)\,\overline{Q(n)} + \overline{K(n)}\,Q(n)$$

$$\therefore \quad y_1^+ = J_1 \bar{Q}_1 + \bar{K}_1 Q_1 = (\bar{x} + \bar{y}_1 + \bar{y}_2)\,\bar{y}_1 + (\overline{\bar{x} + \bar{y}_2})\,y_1$$

Simplify, $\boxed{y_1^+ = \bar{y}_1 + xy_2}$

$$y_2^+ = J_2 \bar{Q}_2 + \bar{K}_2 Q_2 = 1.\bar{y}_2 + (\overline{x + y_2})\,y_2$$

$$= \bar{y}_2 + (\overline{x + y_2})\,y_2$$

Simplifying $\boxed{y_2^+ = \bar{y}_2}$

Transition table will be as shown :

| Present state $y_1 y_2$ | Input x = 0 | Input x = 1 |
|---|---|---|
| 0 0 | 1 1 , 1 | (1) (1) , (1) |
| 0 0 | 1 0 , 1 | 1 0 , 0 |
| 0 0 | 0 1 , 1 | 0 1 , 1 |
| 0 0 | 0 0 , 1 | 1 0 , 0 |

Next state: $y_1^+$, $y_2^+$, z

(3) Let's say,

0 0 → A

0 1 → B

1 0 → C

1 1 → D

∴ State table will be,

| Present state $y_1 y_2$ | Present input x = 0 | Present input x = 1 |
|---|---|---|
| A | D, 1 | D, 1 |
| B | C, 1 | C, 0 |
| C | B, 1 | B, 1 |
| D | A, 1 | C, 0 |

Scanned by CamScanner

**(4) State diagram :**

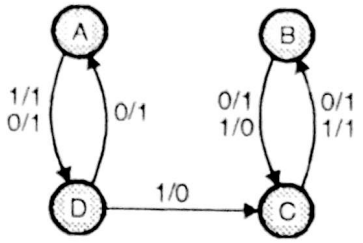



**Fig. Ex. 11.5(b)**

**Ex. 6 :** For given sequential circuit draw state table and state diagram. Refer Fig. Ex. 11.6(a).

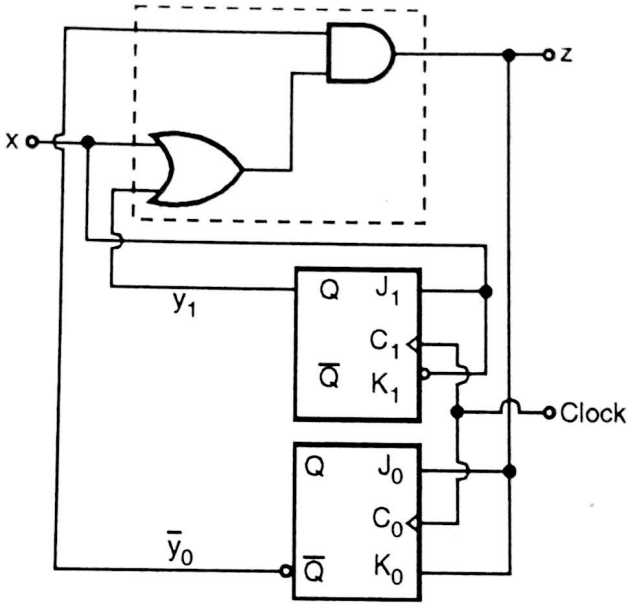



**Fig. Ex. 11.6(a)**

LIBRARY
Dept. IT
743

**Soln. :** (1) First step is to write equations :

$$J_1 = x \quad ; \quad K_1 = \bar{x}$$

$$J_0 = z = (x + y_1)\,\bar{y}_0$$

$$K_0 = \overline{x + y_1\,\bar{y}_0}$$

From characteristic equation of JK we can write down,

$$y_1^+ = J_1\,\bar{Q}_1 + \bar{K}_1 Q_1$$

$$= x\,\bar{y}_1 + \bar{x}\,y_1 = x\,\bar{y}_1 + x\,y_1$$

$$= x$$

$$y_0^+ = J_0\,\bar{Q}_0 + \bar{K}_0\,Q_0$$

$$= (x + y_1)\,\bar{y}_0 \cdot \bar{y}_0 + \overline{(x + y_1\,\bar{y}_0 \cdot y_0}$$

$$= (x + y_1)\,\bar{y}_0$$

Scanned by CamScanner

(2) Now write transition table.

| Present state $y_1 y_0$ | Present input x = 0 | Present input x = 1 |
|---|---|---|
| 0 0 | 0 0 , 0 | 1 1 , 1 |
| 0 1 | 0 0 , 0 | 1 0 , 0 |
| 1 1 | 0 0 , 0 | 1 0 , 0 |
| 1 0 | 0 1 , 1 | 1 1 , 1 |

($Y_1$ $Y_0$ Z)

(3) Define states :

$0\ 0 \longrightarrow q_0$

$0\ 1 \longrightarrow q_1$

$1\ 0 \longrightarrow q_2$

$1\ 1 \longrightarrow q_3$

We get state table :

| Present state $y_1\ y_0$ | Present input x = 0 | Present input x = 1 |
|---|---|---|
| 0 0 | $q_0$ , 0 | $q_3$ , 1 |
| 0 1 | $q_0$ , 0 | $q_2$ , 0 |
| 1 1 | $q_0$ , 0 | $q_2$ , 0 |
| 1 0 | $q_1$ , 1 | $q_3$ , 1 |

(4) **State diagram :**

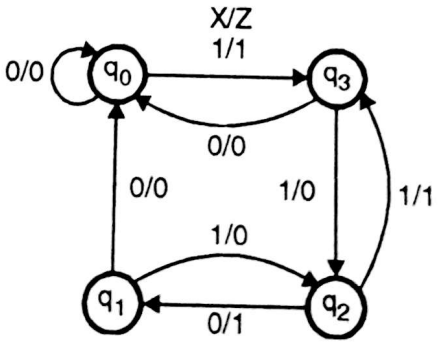



**Fig. Ex. 11.6(b)**

## 11.5 State Reduction and Assignment :

Whenever we design any circuit, in initial stage we say it as ***Raw*** design. Once the ***Raw*** design fulfills our requirements, we concentrate on reducing the same design. Reducing doesn't mean compromising for performance. Reducing means reduction in hardware. By reducing hardware one can reduce cost of the equipment and increase reliability of the design. Reliability increases with less components because failure rate decreases.

Scanned by CamScanner

In sequential system design we would also like to reduce the hardware. But for reducing the same we have to use procedure or algorithm, this process is called ***State table reduction***. The name involve state table just because we are going to take help of state table and reduce the number of states.

An algorithm for state reduction of completely specified state table is given as follows :

***Two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit either to same state or to an equivalent state. If two states are equivalent, one of them can be removed without altering the input-output relationships.***

To understand the concept more clearly let's take one example.

**Ex. 7 :** For given state diagram, prepare state table and reduce the same using "State table reduction" technique. Refer Fig. Ex. 11.7(a).

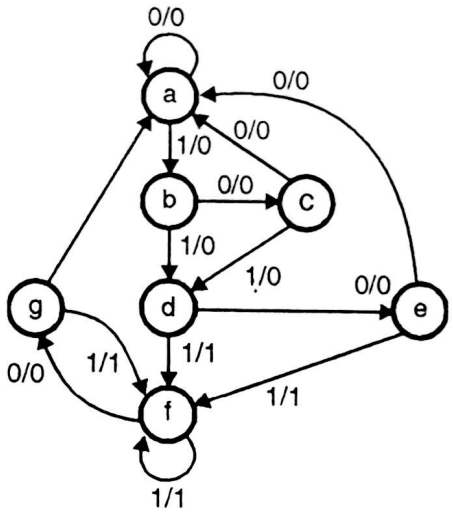



**Fig. Ex. 11.7(a) : State diagram**

**Soln. :** (1) The first step we will follow is prepare state table. The state table is as follows :

| Present state | Next state | | Output | |
|---|---|---|---|---|
| | x = 0 | x = 1 | x = 0 | x = 1 |
| a | a | b | 0 | 0 |
| b | c | d | 0 | 0 |
| c | a | d | 0 | 0 |
| d | e | f | 0 | 1 |
| e | a | f | 0 | 1 |
| f | g | f | 0 | 1 |
| g | a | f | 0 | 1 |

Now observe the state table carefully. Can you see ***equivalent state(s)*** ? Yes, states g and e are equivalent states; they both go to states a and f, and have outputs of 0 and 1, for x = 0 and x = 1.

∴ Change in the state table is ***discard*** state g. And replace state g by e. therefore we have new table as,

Scanned by CamScanner

| Present state | Next state | | Output | |
|---|---|---|---|---|
| | x = 0 | x = 1 | x = 0 | x = 1 |
| a | a | b | 0 | 0 |
| b | c | d | 0 | 0 |
| c | a | d | 0 | 0 |
| d | e | f | 0 | 1 |
| e | a | f | 0 | 1 |
| f | ⓒ | f | 0 | 1 |

↓

'g' state replaced by 'e'

Now again observe this new table carefully. Can you find any "***equivalent state***" ? Yes, 'd' and 'f' both are same. Therefore now discard state f, and replace state f in state diagram by state 'd'. Therefore next version of state table is as follows :

| Present state | Next state | | Output | |
|---|---|---|---|---|
| | x = 0 | x = 1 | x = 0 | x = 1 |
| a | a | b | 0 | 0 |
| b | c | d | 0 | 0 |
| c | a | d | 0 | 0 |
| d | e | d | 0 | 1 |
| e | a | d | 0 | 1 |

Now prepare state diagram for reduced state table.

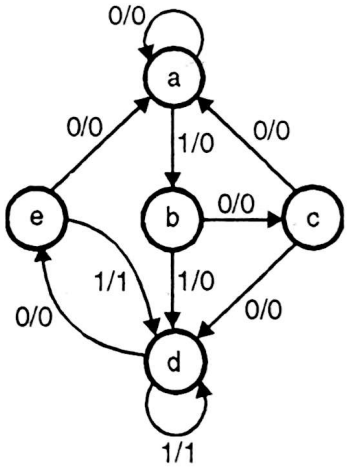



**Fig. Ex. 11.7(b) : State diagram for reduced state table**

## Cross Verification :

This is just cross verifying that whatever reduction we have done is correct. This can be checked by applying an input sequence x to the circuit. Check output produced by original state table and reduced state table. Let's say we apply x = 010101110.

(1) States and output produced by original state table when x is applied. Prepare table for the same.

| Input x | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| State | a | → a | b | c | d | e | f | f | f | g |
| Output | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

↑ Next State Transition

Initially state is a, output = 0

Scanned by CamScanner

Input applied is x = 0. Therefore next state is 'a'.

(2) Now we will check for output and states generated by reduced state table, when an input x is applied. We will prepare table for the same :

| Input x | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| State | a | a | b | c | d | e | d | d | d | e |
| Output | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

↑ Next State Transition

Initially state is a, output = 0

Input applied is x = 0 therefore next state is a.

Observing outputs in both the cases, we found that both are matching. Therefore reduction method is perfect. Here you may ask about output states, because that are not matching. Answer for this doubt is, we are least concerned with the states circuit goes through. We are interested in final output.

**State Assignment :**

State assignment procedures are concerned with methods for assigning binary values to states in such a way as to reduce the cost of the combinational circuit. Actually one can assign any binary value to state, because as I told you, we are concerned with output only, no way to states.

Regarding minimum hardware cost, one has to go for trial and error method to see that in which type we require less hardware.

## 11.6 Sequential Circuit Synthesis :

In this section we are going to study design procedure for sequential circuit. We are going to learn systematic technique for sequential circuit implementation. Steps in this section we follow is :

- Construct state table
- Construct state diagram
- Assign binary value to states
- Depending upon requirement of designer, use SR, T, JK FF, and draw logic diagram.
- Minimisation technique used where it was required.

### 11.6.1 Design Process :

Our target is to design a sequential circuit and to derive structure that has ***specified*** behaviour. One has to see to it that circuit designed should meet behavioural specification or in short it should ensure functional correctness.

The main steps required to design sequential circuit is explained step by step in following section. One can reduce (hopefully) or shorten the steps with his/her experience. Now let's start with the design steps.

1. ***State Behaviour Specification :*** In this one has to specify state diagrams or state table or timing diagram or other pertinent information. Only requirement is, one should describe precisely the input/output behaviour required for target circuit. Basically this point specifies internal states needed and the transition that can occur between the states.
2. ***State Reduction :*** If state diagram is given, prepare state table or directly state table will be given. After that the next step is to reduce number of states using state reduction algorithm/technique.

Scanned by CamScanner

This is done because sequential circuit can be characterised by input-output relationships independent of the number of states.

3. ***State Assignment :*** After we have complete state table, we have to assign binary value to the states. Number of binary state variables (bits) required will be dependent of number of states in state table. Let's say number of states in state table are S. Then number of binary bits B can be given by, $B \geq \log_2 S$.
4. ***Transition Table :*** This table specifies required state behaviour in binary form. This is constructed from state assignment and state table or diagram.
5. ***Determination of FF :*** Determine number of flip–flops needed and assign some letter symbol to each.
6. ***Choose Type of FF :*** In this we are suppose to choose FF from varieties available i.e. SR, D, JK, T. etc.
7. ***Circuit Excitation :*** From state table derive circuit excitation and output table.
8. ***Input/Output Functions :*** Using map or any other simplified method derive circuit output functions. We also have to define the input function that must be applied to flip–flops to produce the required next-state values on their outputs.
9. ***Logic Diagram :*** Now finally draw logic diagram.

This general procedure for synthesizing synchronous flip–flops circuit, is represented in flowchart manner as follows :

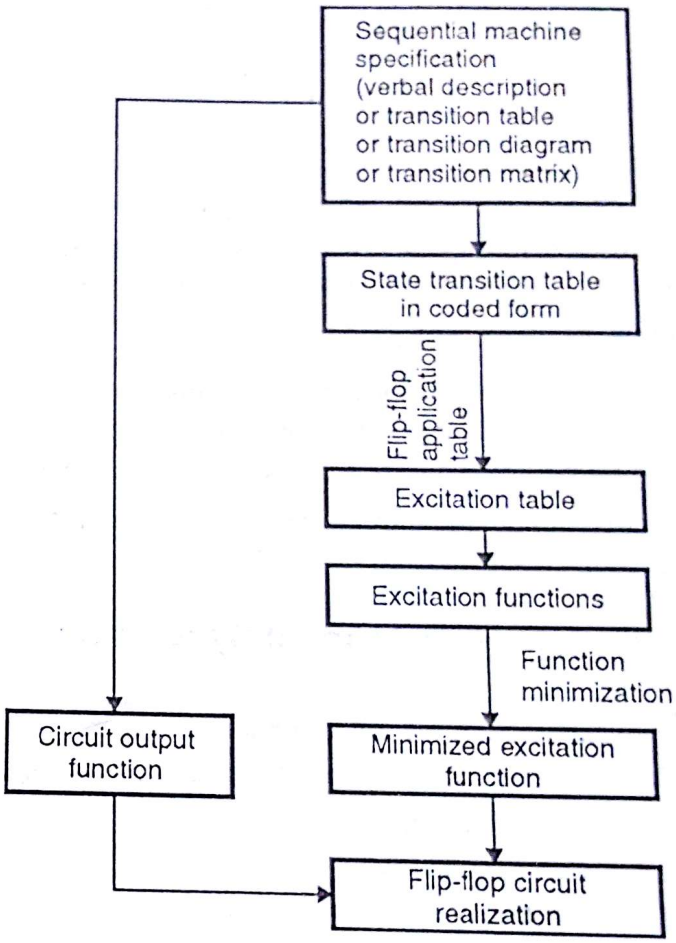



**Fig. 11.10 : General synthesis procedure**

Let's solve some problems for the same.

---

**Ex. 8 :** Design sequential circuit for given state diagram using JKFF. Refer Fig. Ex. 11.8(a)

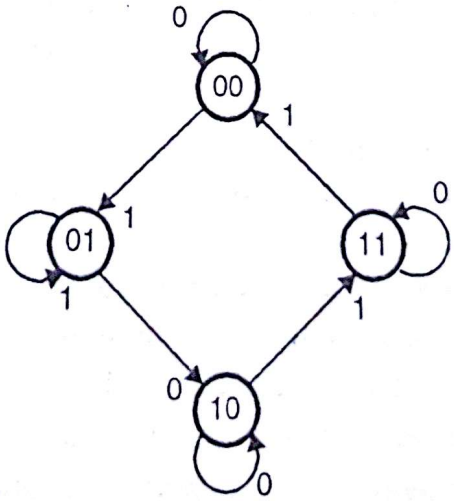



**Fig. Ex. 11.8(a) : State diagram**

Step 2 : ... to write *excitation table* for given ...

Scanned by CamScanner

**Soln. :**

**Step 1 :** For given state diagram, derive state table.

| Present state | | Next state | | | |
|---|---|---|---|---|---|
| | | x = 0 | | x = 1 | |
| A | B | A | B | A | B |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

If you observe state diagram it consists of four states with binary value already given. Transition lines are marked with single binary digit without a '1', it means that there is one input variable and no output variables.

∴ State of flip-flop ***may*** be treated as an output.

**Step 2 :** Number of flip-flops required.

Number of states in state table are '4'

∴ $B = 4$

∴ $B = \log_2 (4) = \log_2 (2)^2 = 2 \log_2 2 = 2$

∴ As two binary bits are required, we require two flip-flops.

At this stage block diagram will look like as shown in Fig. Ex. 11.8(b).

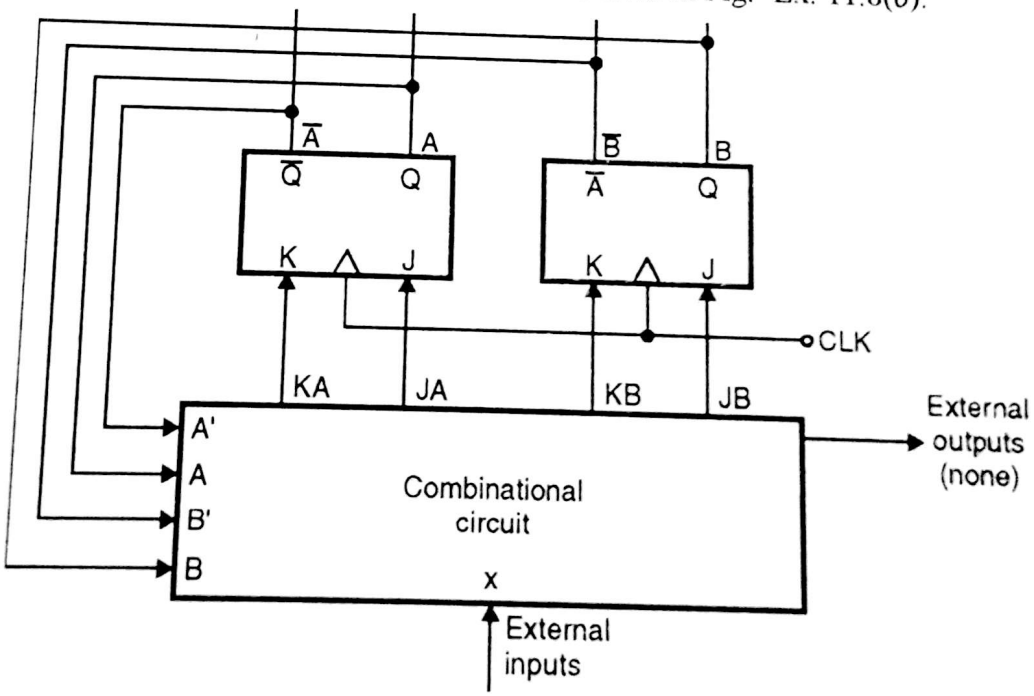



**Fig. Ex. 11.8(b)**

**Step 3 :** Recall excitation table of JKFF.

**Step 4 :** Now we are going to write ***excitation table*** for given state diagram. Basically this table will contain :

Scanned by CamScanner

(1) Inputs of combinational circuit :
   (a) Present state
   (b) Input x.
(2) Next state
(3) Output of combinational circuit i.e. referring to Fig. Ex. 11.8(b). JA, KA, JB, KB and so on.

The same is shown as follows :

| Inputs of Combinational Circuit | | | Next state | | Output of Combinational Circuit | | | |
|---|---|---|---|---|---|---|---|---|
| Present state | | Input | | | Flip-flop Inputs | | | |
| A | B | x | A | B | JA | KA | JB | KB |
| | | | | | | | | |

Now we will fill up I and J column, referring to state table, therefore we get,

| Inputs of Combinational Circuit | | | Next state | | Outputs of Combinational Circuit | | | |
|---|---|---|---|---|---|---|---|---|
| Present state | | Input | | | Flip-flop Inputs | | | |
| A | B | x | A | B | JA | KA | JB | KB |
| (0) | 0 (i) | 0 | (0) | 0 | 0 | X | | |
| 0 | (0) (P) | 1 | 0 | (1) | | | 1 | X |
| 0 | 1 | 0 | 1 | 0 | | | | |
| 0 | 1 | 1 | 0 | 1 | | | | |
| 1 | 0 | 0 | 1 | 0 | | | | |
| 1 | 0 | 1 | 1 | 1 | | | | |
| 1 | 1 | 0 | 1 | 1 | | | | |
| 1 | 1 | 1 | 0 | 0 | | | | |

**Table 1**

You will find entries shown by dotted box. Now let's see how we got it. To get this you have to ask one question to yourself i.e.,

Scanned by CamScanner

***If previous state is 'z', required next state is w, then what should be J and K inputs ?***. For example, refer 'i' transition line in Table 1. We will ask ***If previous state is '0', required next state is '0', then what will be J-K inputs ?***

The answer is J = 0 and K = X. Write the same in that row and in column of ***Flip-flop output*** (Shown in dotted box, in Table 1).

Take one more example. Refer transition line 'p' in Table 1.

***If previous state is '0', required next state is '1', then J = 1 and K = X***. Fill value of J-K in Table 1 under JB and KB.

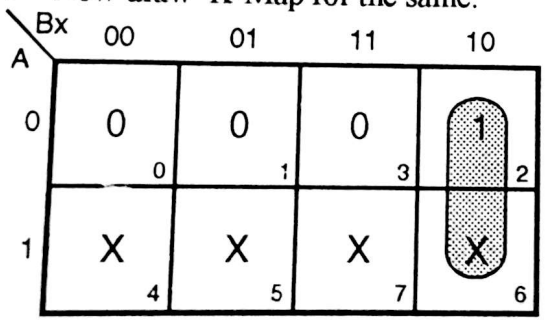
This way you start filling the full table step by step for 'A' and 'B' variable

Finally you get,

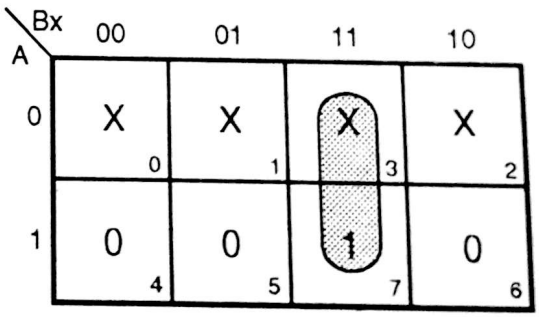
| Inputs of Combinational circuit | | | Next state | | Outputs of Combinational circuit | | | |
|---|---|---|---|---|---|---|---|---|
| Present state | | Input | | | Flip–flop inputs | | | |
| A | B | x | A | B | JA | KA | JB | KB |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X |
| 0 | 0 | 1 | 0 | 1 | 0 | X | 1 | X |
| 0 | 1 | 0 | 1 | 0 | 1 | X | X | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | X | X | 0 |
| 1 | 0 | 0 | 1 | 0 | X | 0 | 0 | X |
| 1 | 0 | 1 | 1 | 1 | X | 0 | 1 | X |
| 1 | 1 | 0 | 1 | 1 | X | 0 | X | 0 |
| 1 | 1 | 1 | 0 | 0 | X | 1 | X | 1 |

**Table 2**

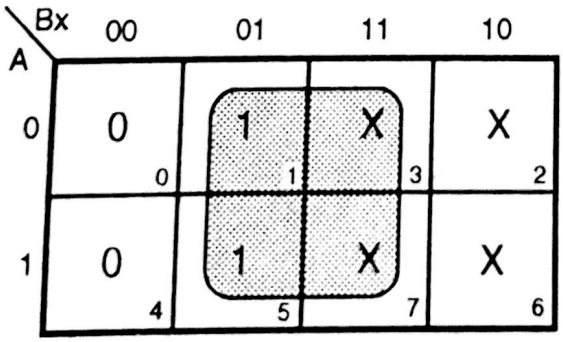
**Step V :** Now draw K-Map for the same.



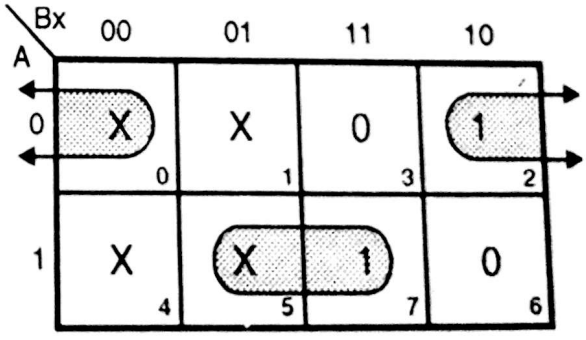

$JA = B\bar{x}$




$KA = Bx$




$JA = x$




$KB = Ax + \bar{A}\bar{x}$

$= A \odot x$

**Fig. Ex. 11.8(c)**

Scanned by CamScanner

**Step VI :** Now finally draw logic diagram.

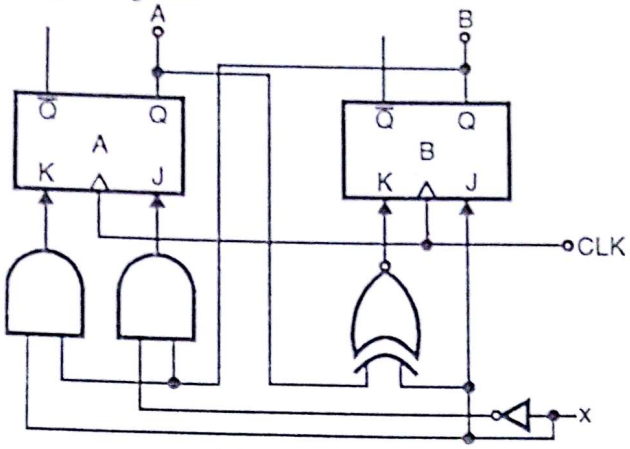



**Fig. Ex. 11.8(d)**

---

**Ex. 9 :** Design serial adder. Use JK FF.

**Soln. :**

**Step 1 :** We have already with us state table for serial adder.
Let us redraw to recall the same.
For state diagram refer Fig. 11.8.

| | $x_1x_2$ | Present input $x_1x_2$ | | | |
|---|---|---|---|---|---|
| | y | 00 | 01 | 11 | 10 |
| Present state | 0 | $q_0, 0$ | $q_0, 1$ | $q_1, 0$ | $q_0, 1$ |
| | 1 | $q_0, 1$ | $q_1, 0$ | $q_1, 1$ | $q_1, 0$ |

where $q_0 = 0$ and $q_1 = 1$

$\therefore$ Single binary bit required therefore we require only one flip-flop.

**Step 2 :** Now let us redraw state table, as we have done in previous example. (hint Table 3) (Refer Table 1 i.e. JK FF transition table).

| Inputs of Combinational Circuit | | | | Outputs of Combinational Circuit | |
|---|---|---|---|---|---|
| Present state | Input | | Next state | Flip-flop inputs | |
| y | $x_1$ | $x_2$ | $y^+$ | J | K |
| 0 | 0 | 0 | 0 | 0 | X |
| 0 | 0 | 1 | 0 | 0 | X |
| 0 | 1 | 0 | 0 | 0 | X |
| 0 | 1 | 1 | 1 | 1 | X |
| 1 | 0 | 0 | 0 | X | 1 |
| 1 | 0 | 1 | 1 | X | 0 |
| 1 | 1 | 0 | 1 | X | 0 |
| 1 | 1 | 1 | 1 | X | 0 |

**Step 3 :** Now draw K-Map for J and K.

Scanned by CamScanner

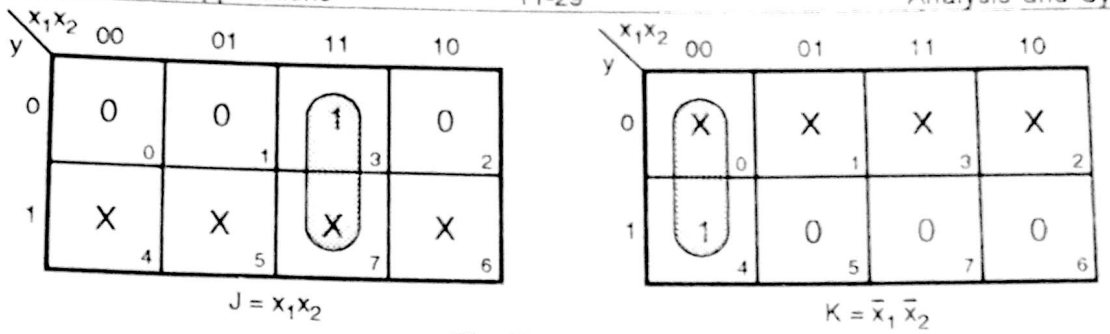
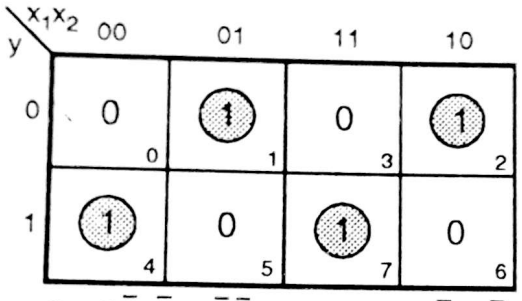



**Fig. Ex. 11.9(a)**

Same way we can also draw K map for z output. See Fig. Ex. 11.9(b)

**Step 4** : Logic diagram. Refer Fig. Ex. 11.9(c)




$$z = y\,\bar{x}_1\,\bar{x}_2 + \bar{y}\,\bar{x}_1\,x_2 + y\,x_1\,x_2 + \bar{y}\,x_1\,\bar{x}_2$$
$$= y\,(\bar{x}_1\,\bar{x}_2 + x_1\,x_2) + \bar{y}\,(\bar{x}_1\,x_2 + x_1\,\bar{x}_2)$$
$$= y\,(\overline{x_1 \oplus x_2}) + \bar{y}\,(x_1 \oplus x_2)$$
$$= x_1 \oplus x_2 \oplus y$$

**Fig. Ex. 11.9(b)**

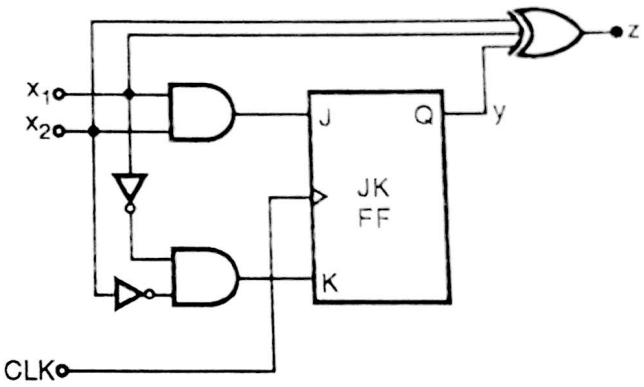



**Fig. Ex. 11.9(c)**

---

**Ex. 10 :** Draw the state diagram for the given state table. Reduce the state table if possible and obtain the universal circuit implementation.

| PS | NS / Output | |
|---|---|---|
| | x = 0, | x = 1 |
| a | f, 0 | b, 0 |
| b | d, 0 | c, 0 |
| c | f, 0 | e, 0 |
| d | g, 1 | a, 0 |
| e | d, 0 | c, 0 |
| f | f, 1 | b, 1 |
| g | g, 0 | h, 1 |
| h | g, 1 | a, 0 |

PS – Present state
NS – Next state.

d = 100

Scanned by CamScanner

**Soln. :**

(1) Draw state diagram for the same.

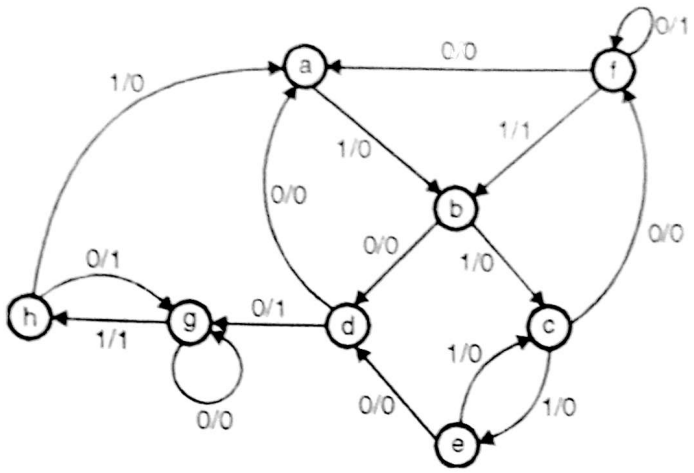



**Fig. Ex. 11.10(a)**

(2) Now observe state table, for getting equivalent state. h = d, we can reduce the states.

| PS | NS / Output | |
|---|---|---|
| | x = 0, | x = 1 |
| a | f, 0 | b, 0 |
| b | d, 0 | c, 0 |
| c | f, 0 | e, 0 |
| d | g, 1 | a, 0 |
| e | d, 0 | c, 0 |
| f | f, 1 | b, 1 |
| g | g, 0 | d, 1 |

Secondly, e = b, therefore replace 'e' by 'b'.

Therefore we have,

| PS | NS / Output | |
|---|---|---|
| | x = 0, | x = 1 |
| a | f, 0 | b, 0 |
| b | d, 0 | c, 0 |
| c | f, 0 | b, 0 |
| d | g, 1 | a, 0 |
| f | f, 1 | b, 1 |
| g | g, 0 | d, 1 |

Now a and c are matching therefore replace 'c' by 'a'.

| PS | NS / Output | |
|---|---|---|
| | x = 0, | x = 1 |
| a | f, 0 | b, 0 |
| b | d, 0 | a, 0 |
| d | g, 1 | a, 0 |
| f | f, 1 | b, 1 |
| g | g, 0 | d, 1 |

Scanned by CamScanner

(3) Now let's assign binary value to states. Number of states are a, b, d, f, and g. i.e. 5.

$\therefore$

$$B \geq \log_2 5$$

$$B \geq \log_2 5$$

$$B \geq 3$$

$\therefore$ Number of bits required will be 3.

Let us say,

$$a = 000 - (0)_{10}$$

$$b = 001 - (1)_{10}$$

$$d = 011 - (3)_{10}$$

$$f = 010 - (2)_{10}$$

$$g = 110 - (6)_{10}$$

$\therefore$ Unused states are 4, 5 and 7.

$\therefore$ State table will be,

| PS | NS / Output | |
|---|---|---|
| | x = 0 | x = 1 |
| 000 | 010, 0 | 001, 0 |
| 001 | 011, 0 | 000, 0 |
| 011 | 110, 1 | 000, 0 |
| 010 | 010, 1 | 001, 1 |
| 110 | 110, 0 | 011, 1 |

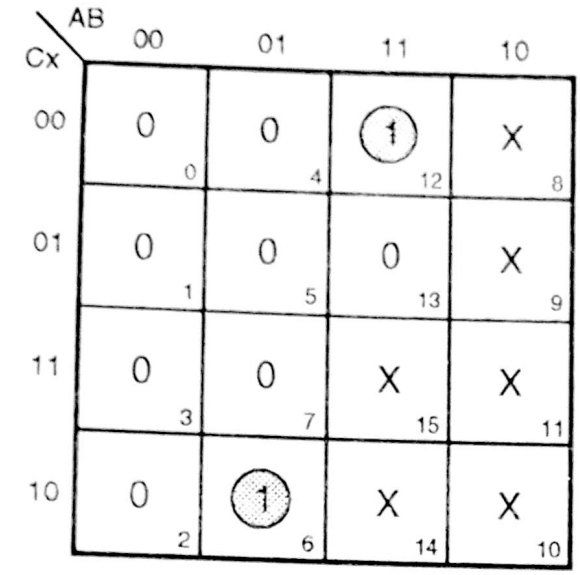
Now draw excitation table for above state assigned state table.

We will use D FF for simplified hardware.

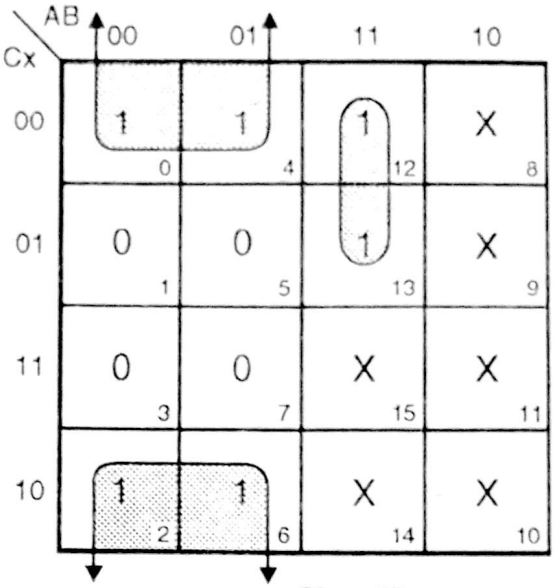
| Inputs of Combinational Circuit | | | | Next state | | | Output of Combinational Circuit | | | O/P |
|---|---|---|---|---|---|---|---|---|---|---|
| Present state | | | Input | | | | Flip-flop inputs | | | |
| A | B | C | x | A | B | C | $D_A$ | $D_B$ | $D_C$ | Y |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

Scanned by CamScanner

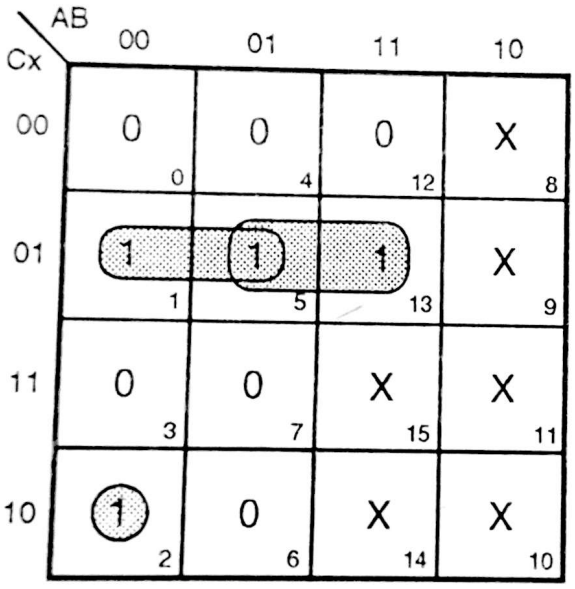
(4) **K-map :**



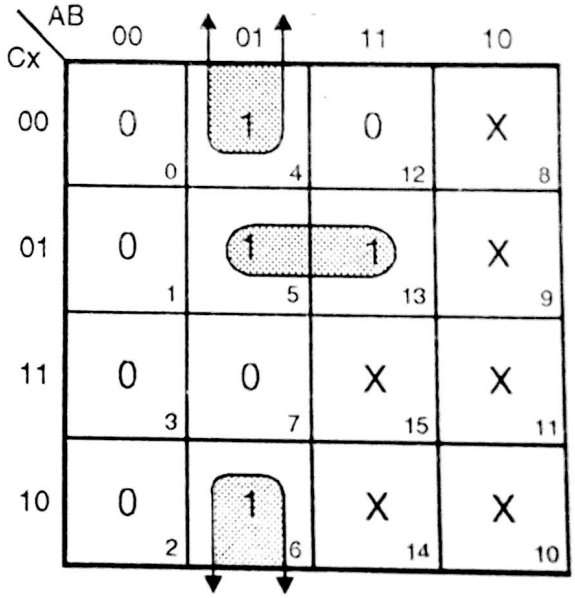

$D_A = A\,B\,\overline{C}\,\overline{x} + \overline{A}\,B\,C\,\overline{x}$




$D_B = A\,B\,\overline{C} + A\,\overline{x}$




$D_C = \overline{A}\,\overline{B}\,C\,\overline{x} + \overline{C}\,x\,(\overline{A} + B)$




$Y = \overline{A}\,B\,\overline{x} + B\,\overline{C}\,x$

**Fig. Ex. 11.10(b)**

**Note :** States those are not present considered 'x' don't care.

Scanned by CamScanner

(5) **Logic diagram :**

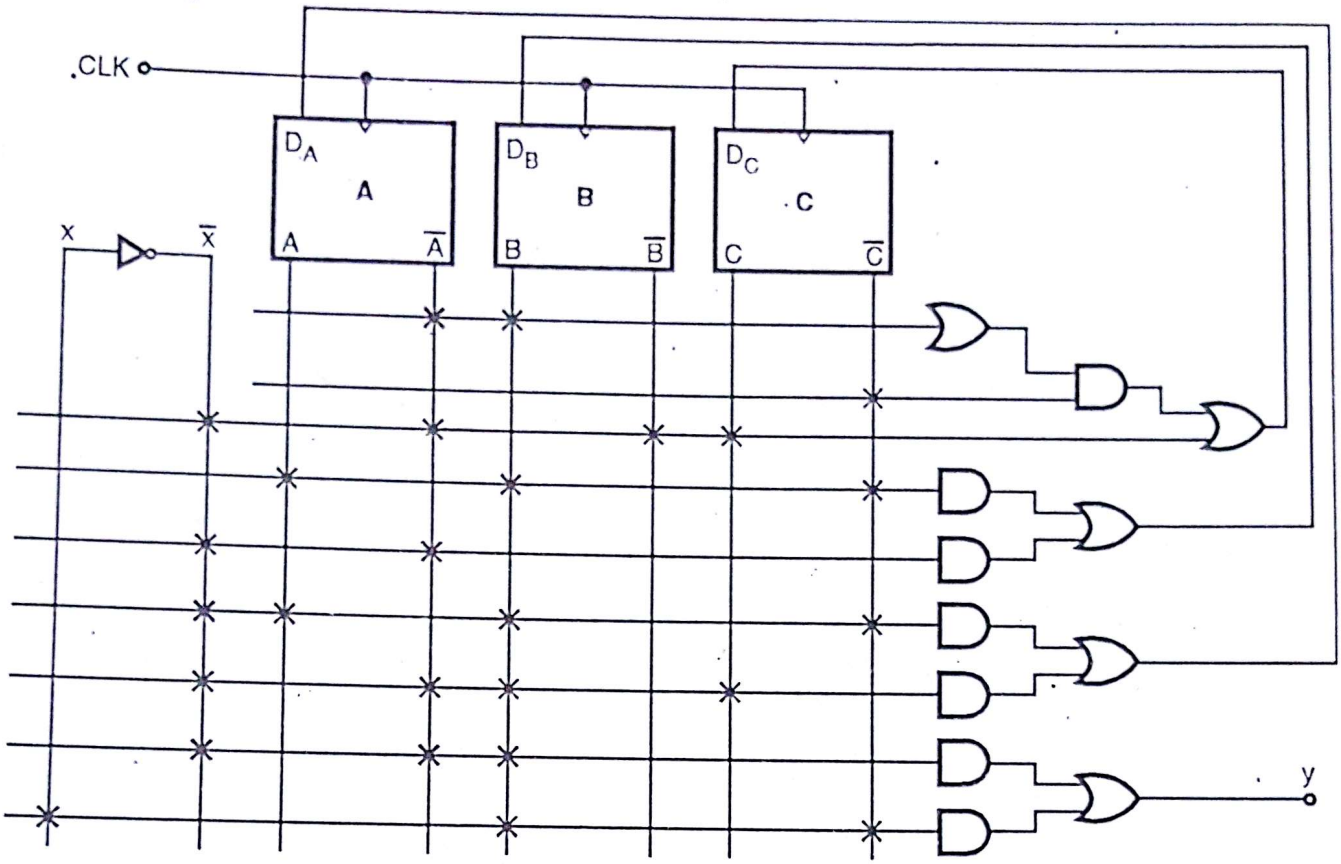



**Fig. Ex. 11.10(c)**

---

**Ex. 11 :** For state diagram given in Ex. 7, design circuit using RSFF.

**Soln. :** (1) Refer Fig. Ex. 11.7(a). It shows given state diagram in Ex. 7.

(2) After implementing reduction technique we have state diagram shown in Fig. Ex. 11.7(b) The state table for the same is shown as follows :

| Present state | Next state | | Output | |
|---|---|---|---|---|
| | x = 0, | x = 1 | x = 0 | x = 1 |
| a | a | b | 0 | 0 |
| b | c | d | 0 | 0 |
| c | a | d | 0 | 0 |
| d | e | d | 0 | 1 |
| e | a | d | 0 | 1 |

(3) Now assign binary value to states.

$$B \geq \log_2 5$$
$$\geq \log_2 5 \geq 3$$
$$\therefore \quad B \geq 3$$

$\therefore$ Three bits will be required say ABC.

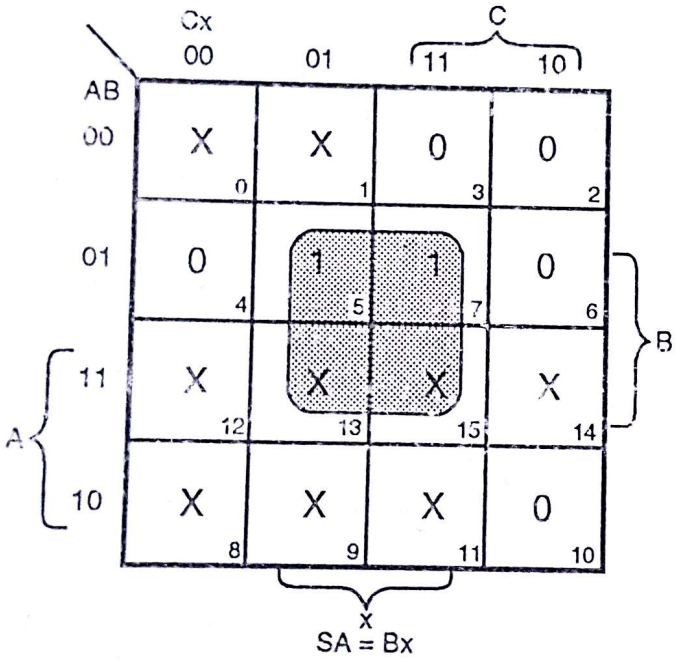
a = 001
b = 010
c = 011
d = 100
e = 101

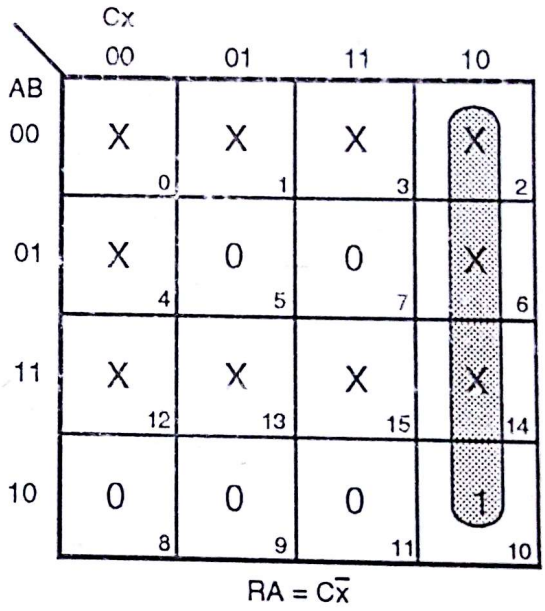
Scanned by CamScanner

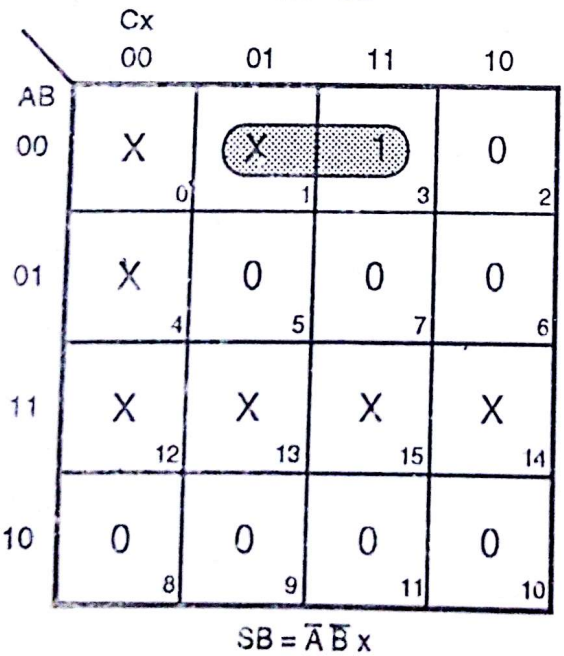
Now refer excitation table of RSFF and we get excitation table for state table as follows :

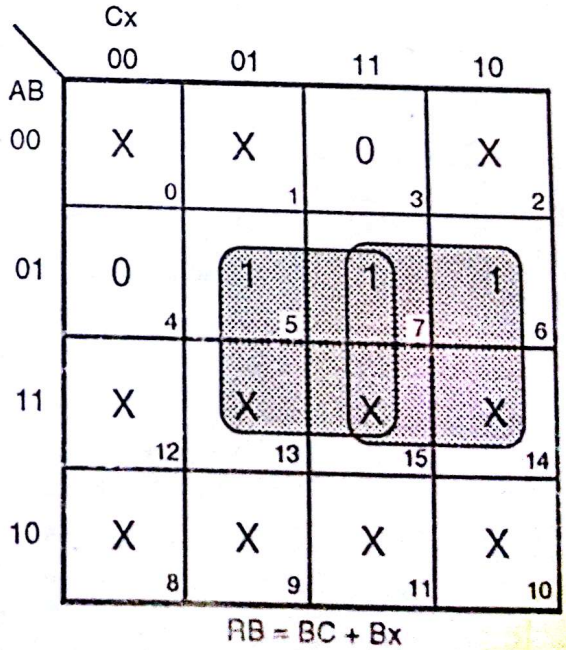
| Present state | | | Input | Next State | | | Flip-flop inputs | | | | | | Output |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | x | A | B | C | SA | RA | SB | RB | SC | RC | y |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | X | 0 | X | X | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | X | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | X | X | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | X | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | X | 0 | 1 | X | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | X | 0 | 0 | X | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | X | 0 | 0 | X | 0 | X | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | X | X | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | X | 0 | 0 | X | 0 | 1 | 1 |

(4) K-map :




Scanned by CamScanner

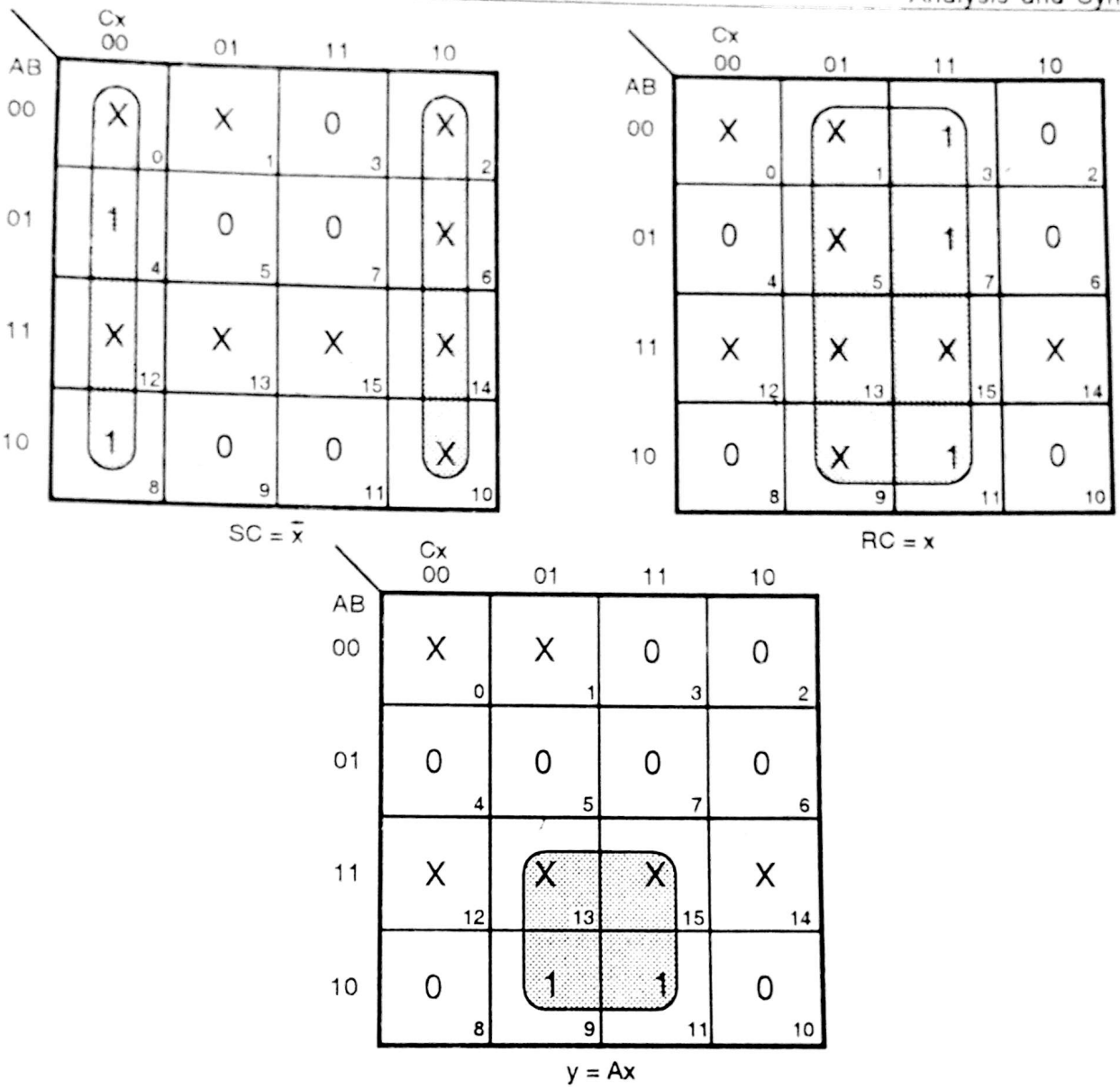



**Fig. Ex. 11.11(a)**

(5) **Logic diagram :**

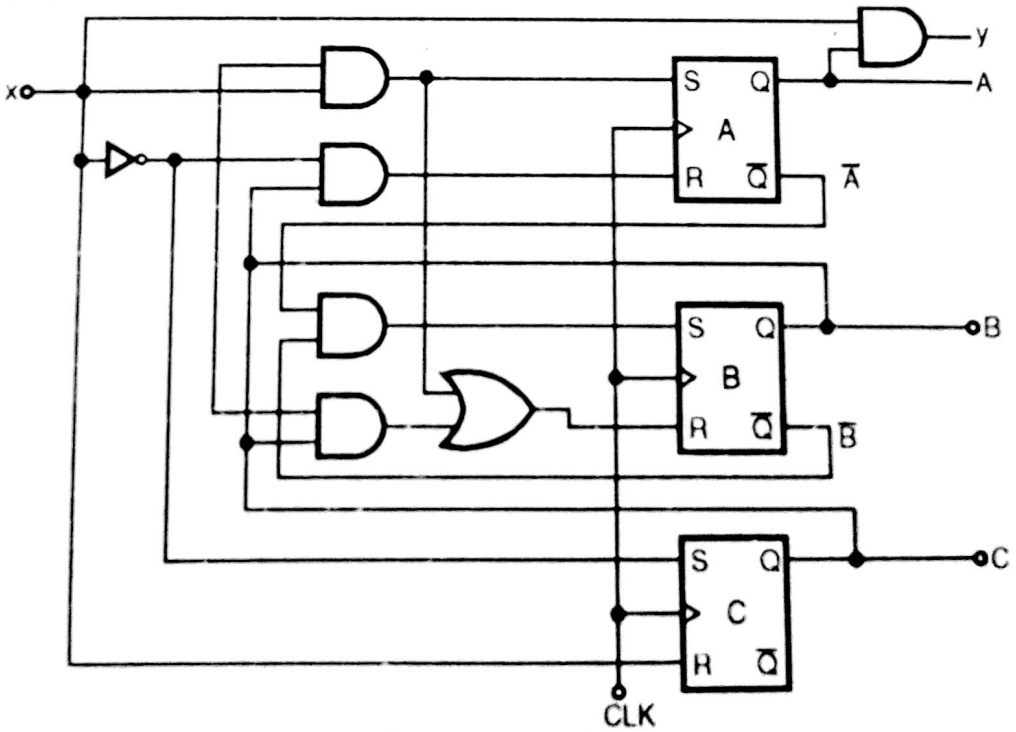



**Fig. Ex. 11.11(b)**

Scanned by CamScanner

**Ex. 12 :** Design a synchronous JK flip flop circuit having one input and one output. The circuit outputs a 1, whenever three consecutive 1's are observed. Otherwise output will be '0'.

**Soln. : Target :** Circuit should sense 3 consecutive 1's only and provide high output.

(1) Let's say initial condition/RESET condition is state $q_0$. If first, logic 1 is present state will change from $q_0$ to $q_1$. For second logical 1, it will change from state $q_1$ to $q_2$. Now for third logical 1, it will remain in $q_2$ only output will be high. The same is shown in Fig. 11.12 (a).

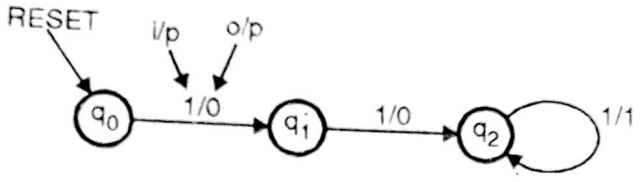



**Fig. Ex. 11.12(a)**

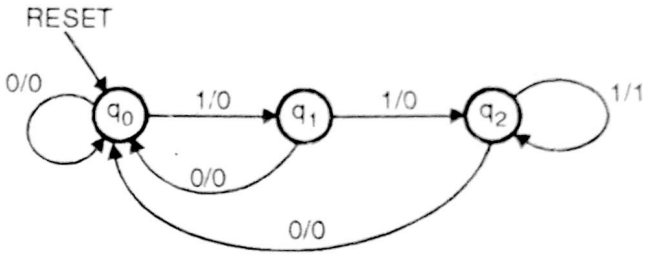



**Fig. Ex. 11.12(b)**

There obvious question you may ask what if in any state you get input as '0', not a logical 1. The answer is if in any state you get logical '0', switch over to $q_0$ state. The same is depicted in Fig. Ex. 11.12(b).

(2) Let us draw state table.

| | x | Present input x | |
|---|---|---|---|
| | q | 0 | 1 |
| Present state | $q_0$ | $q_0$, 0 | $q_1$, 0 |
| | $q_1$ | $q_0$, 0 | $q_2$, 0 |
| | $q_2$ | $q_0$, 0 | $q_2$, 1 |

(3) Number of states are three, i.e. $q_0$, $q_1$ and $q_2$

$$\therefore \quad S = 3$$

$$\therefore \quad B \geq \log_2 S$$

$$\geq \log_2 3$$

$$\geq 2$$

$\therefore$ We require 2 binary bits.

(4) Now assign binary value to state.

$q_0 = 00$
$q_1 = 01$
$q_2 = 11$
} Output states $y_1y_2$

$\therefore$ Transition table will be

| | x | Present input x | |
|---|---|---|---|
| | $q_{(AB)}$ | 0 | 1 |
| Present state | 00 | 00, 0 | 01, 0 |
| | 01 | 00, 0 | 11, 0 |
| | 11 | 00, 0 | 11, 1 |

(5) Now let us draw state table in different manner.

Scanned by CamScanner

| Inputs of Combinational Logic | | | Next state | | Output of Combinational Circuit | | | | Output y |
|---|---|---|---|---|---|---|---|---|---|
| Present state | | Input | | | Flip-flop inputs | | | | |
| A | B | x | A | B | $J_1$ | $K_1$ | $J_2$ | $K_2$ | |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | X | 1 | X | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | X | X | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | X | X | 0 | 0 |
| 1 | 0 | 0 | X | X | X | X | X | X | X |
| 1 | 0 | 1 | X | X | X | X | X | X | X |
| 1 | 1 | 0 | 0 | 0 | X | 1 | X | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | X | 0 | X | 0 | 1 |

**Note** : State '11' is not there, therefore considered as ***don't care*** (X).

(I) **K-map :**

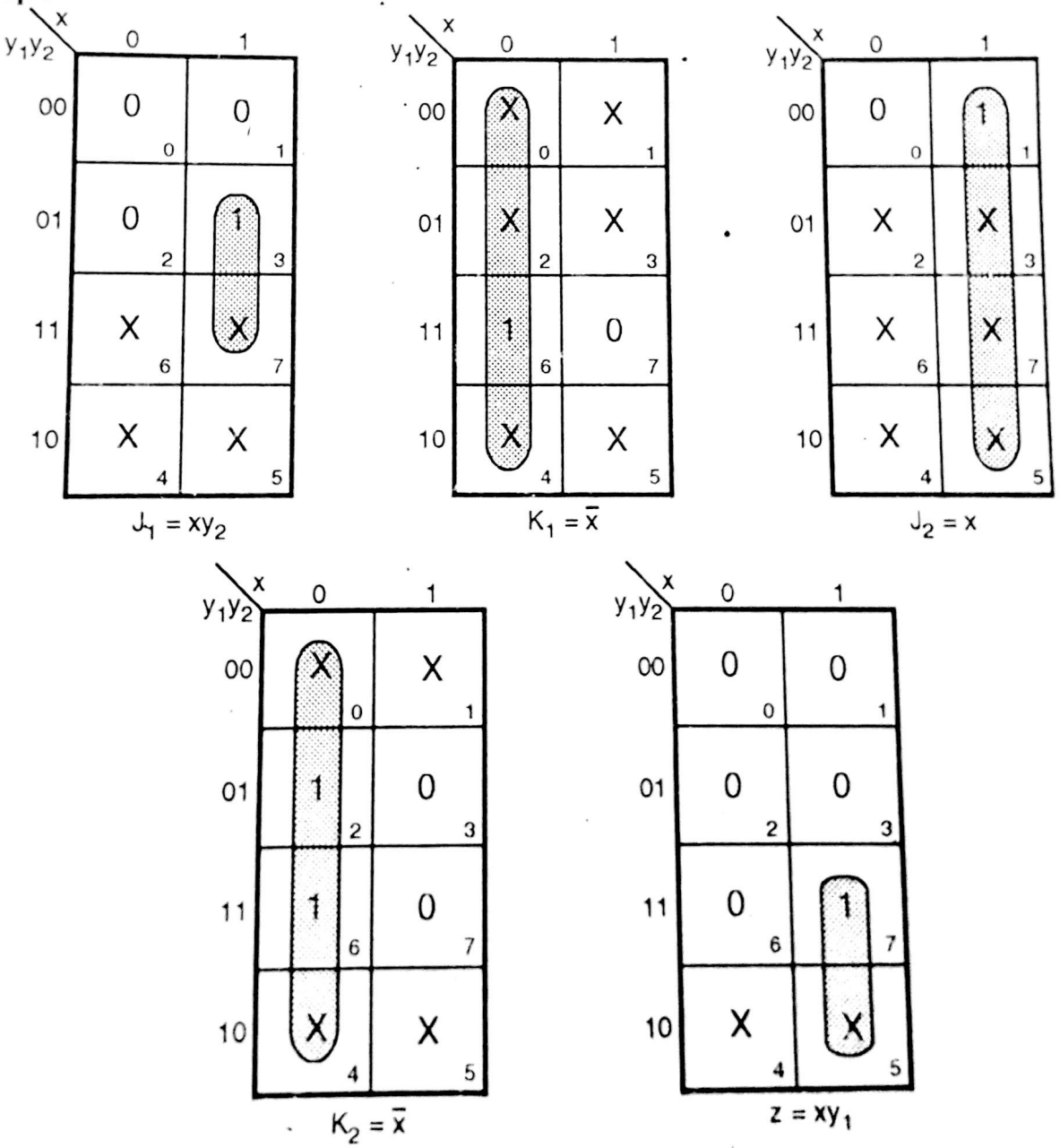



**Fig. Ex. 11.12(c)**

Scanned by CamScanner

(2) **Logic diagram :**

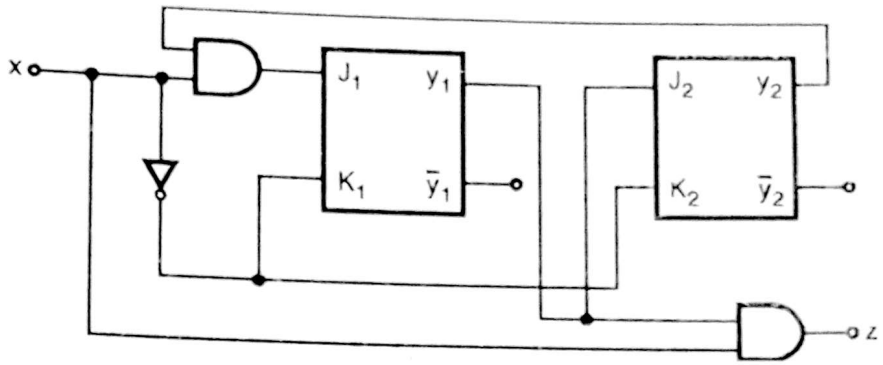



**Fig. Ex. 11.12(d)**

**Ex. 13 :** Design a sequential circuit using Mealy machine to detect an overlapping sequence as follows :

... 1010 ...

**Soln. :**

(1) Overlapping sequence means suppose we have,

1 0 1 0 1 0

over lapped

$\therefore$ 1 0 1 0 → one sequence

10 of previous and 10 of next combinely gives 1010 as new sequence

(2) Let's say we have initially state $q_0$ at RESET. If we get '1', new state will be $q_1$. After that if we get '0', new state will be $q_2$. Now if we get '1', we have new state $q_3$, therefore till this point we have received 101. Now if we get '0' we will switch over to state $q_2$, because we want overlapping sequence output also. The same is depicted in Fig. Ex. 11.13(a).

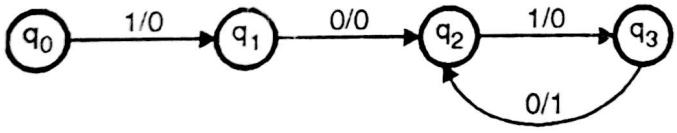



**Fig. Ex. 11.13(a)**

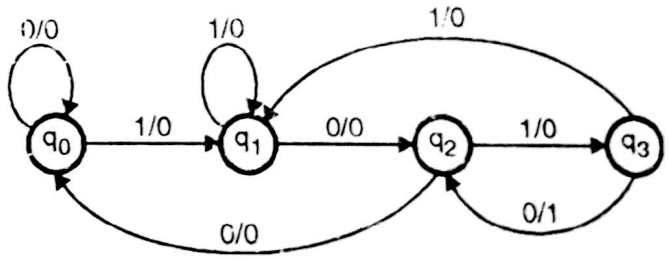



**Fig. Ex. 11.13(b)**

$\therefore$ After $q_3$ if we get 01. It will toggle between $q_3$ and $q_2$ only.

Now let's think about case other than that shown in Fig. 11.13 (a). Cases are as follows :

(1) Present state $q_0$. If we receive '0' input, we take next state as $q_0$ only, because we don't want to proceed till we get logical '1'.

(2) Present state is $q_1$, and we receive logical 1 input. We can think of next state as $q_1$ only, because from this point if we continue still we can trace sequence 1010.

(3) Present state is $q_2$, and we receive logical '0', then next state is $q_0$. This is but obvious because till $q_2$, we received input as '1' and '0', now start from scratch.

(4) Present state $q_3$, and we get logical 1

Scanned by CamScanner

The next state we can think of as $q_1$ cause we start from again first '1' of 1010

These all the cases are considered and final state diagram is shown in Fig. Ex. 11.13(b)

(3) ∴ In totality we have 4 state. We require 2 bits to represent the state

∴
$q_0 = 00$
$q_1 = 01$
$q_2 = 10$
$q_3 = 11$

∴ State table will be

| Present state | x / AB | Present input x = 0 | Present input x = 1 |
|---|---|---|---|
| | $q_0$ | $q_0$, 0 | $q_1$, 0 |
| | $q_1$ | $q_2$, 0 | $q_1$, 0 |
| | $q_2$ | $q_0$, 1 | $q_3$, 0 |
| | $q_3$ | $q_2$, 1 | $q_1$, 0 |

∴ State table with binary state assigned

| Present state | x / AB | Present input x = 0 | Present input x = 1 |
|---|---|---|---|
| | 00 | 00, 0 | 01, 0 |
| | 01 | 10, 0 | 01, 0 |
| | 10 | 00, 0 | 11, 0 |
| | 11 | 10, 1 | 01, 0 |

Redrawing the diagram with different format and considering inputs of flip-flop. We get table as follows. We are using JK FF for designing.

| **Inputs of Combinational Logic** | | | | | **Output of Combinational Circuit** | | | | **Output** |
|---|---|---|---|---|---|---|---|---|---|
| **Present state** | | **Input** | **Next state** | | **Flip-flop inputs** | | | | **y** |
| **A** | **B** | **x** | **A** | **B** | **JA** | **KA** | **JB** | **KB** | |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | X | 1 | X | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | X | X | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | X | X | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | X | 1 | 0 | X | 0 |
| 1 | 0 | 1 | 1 | 1 | X | 0 | 1 | X | 0 |
| 1 | 1 | 0 | 1 | 0 | X | 0 | X | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | X | 1 | X | 0 | 0 |

Scanned by CamScanner

(4) **K-map :**

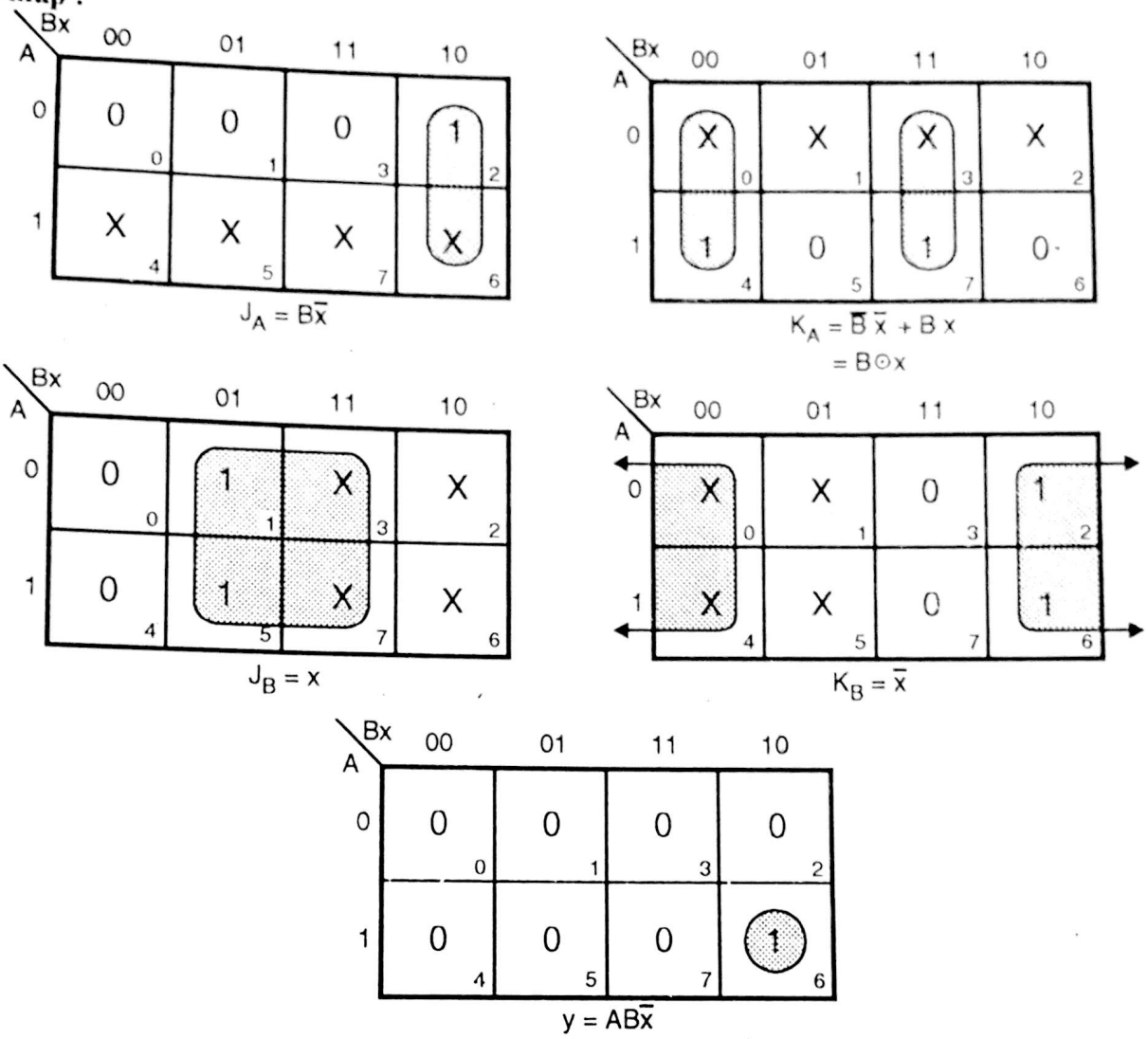



**Fig. 11.13(c)**

(5) **Logic diagram :**

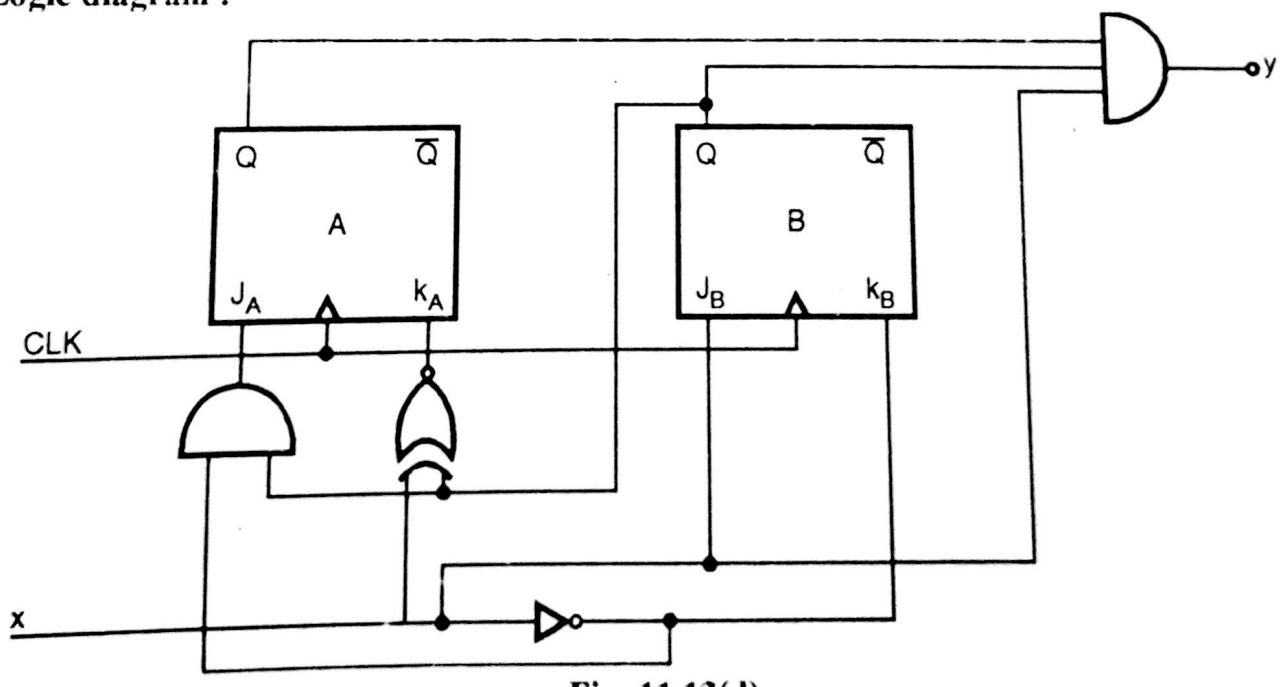



**Fig. 11.13(d)**

Scanned by CamScanner

**Ex. 14 :** Design sequential circuit for detecting and overlapping sequence 1101 Use JK flip-flop

**Soln. :** (1) Draw state diagram

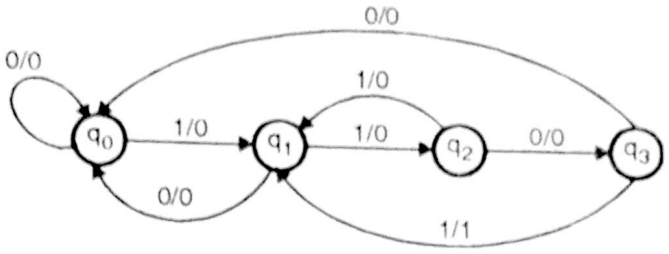



**Fig. 11.14(a)**

(2) State table for the same.

| Present state | AB \ x | Present input x = 0 | Present input x = 1 |
|---|---|---|---|
| | $q_0$ | $q_0, 0$ | $q_1, 0$ |
| | $q_1$ | $q_0, 0$ | $q_2, 0$ |
| | $q_2$ | $q_3, 0$ | $q_1, 0$ |
| | $q_3$ | $q_0, 0$ | $q_1, 0$ |

(3) Total states are four therefore number of bits required will be two. Therefore Flip-flop used will be 2.

Assign binary value to state.

$\therefore$ $q_0 = 00$, $q_1 = 01$, $q_2 = 10$, $q_3 = 11$

$\therefore$ State table will be

| Present state | AB \ x | Present input x = 0 | Present input x = 1 |
|---|---|---|---|
| | 00 | 00, 0 | 01, 0 |
| | 01 | 00,0 | 10,0 |
| | 10 | 11,0 | 01,0 |
| | 11 | 00,0 | 01,1 |

(4) Remember transition table for JK FF and redraw the table we have.

| Inputs of Combinational Logic | | | | | Output of Combinational Circuit | | | | Output y |
|---|---|---|---|---|---|---|---|---|---|
| Present state | | Input | Next state | | Flip-flop inputs | | | | |
| A | B | x | A | B | JA | KA | JB | KB | |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | X | 1 | X | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | X | X | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | X | X | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | X | 0 | 1 | X | 0 |
| 1 | 0 | 1 | 0 | 1 | X | 1 | 1 | X | 0 |
| 1 | 1 | 0 | 0 | 0 | X | 1 | X | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | X | 1 | X | 0 | 1 |

Scanned by CamScanner

(5) **K-map :**

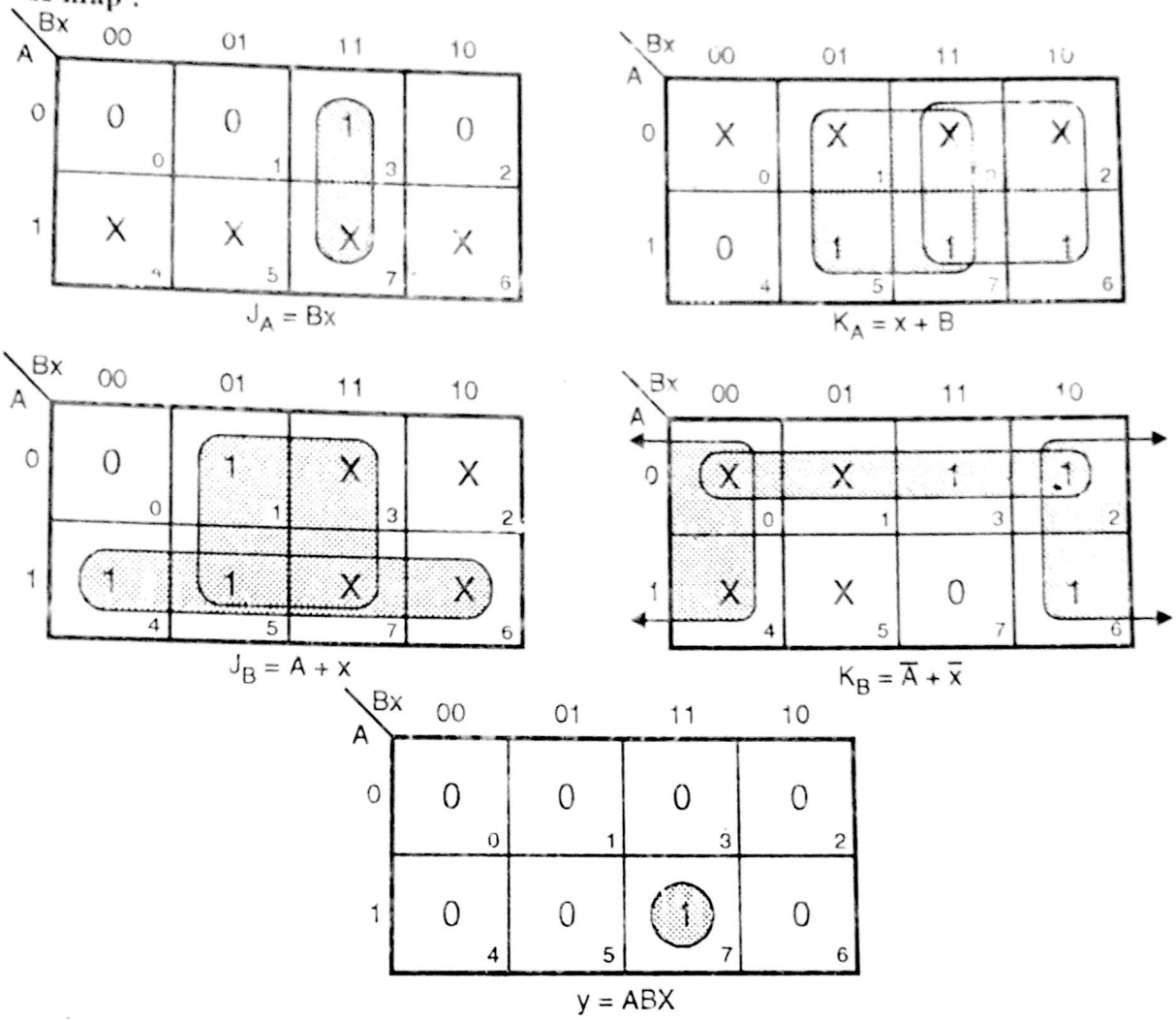



**Fig. Ex. 11.14(b)**

(6) **Circuit diagram :**

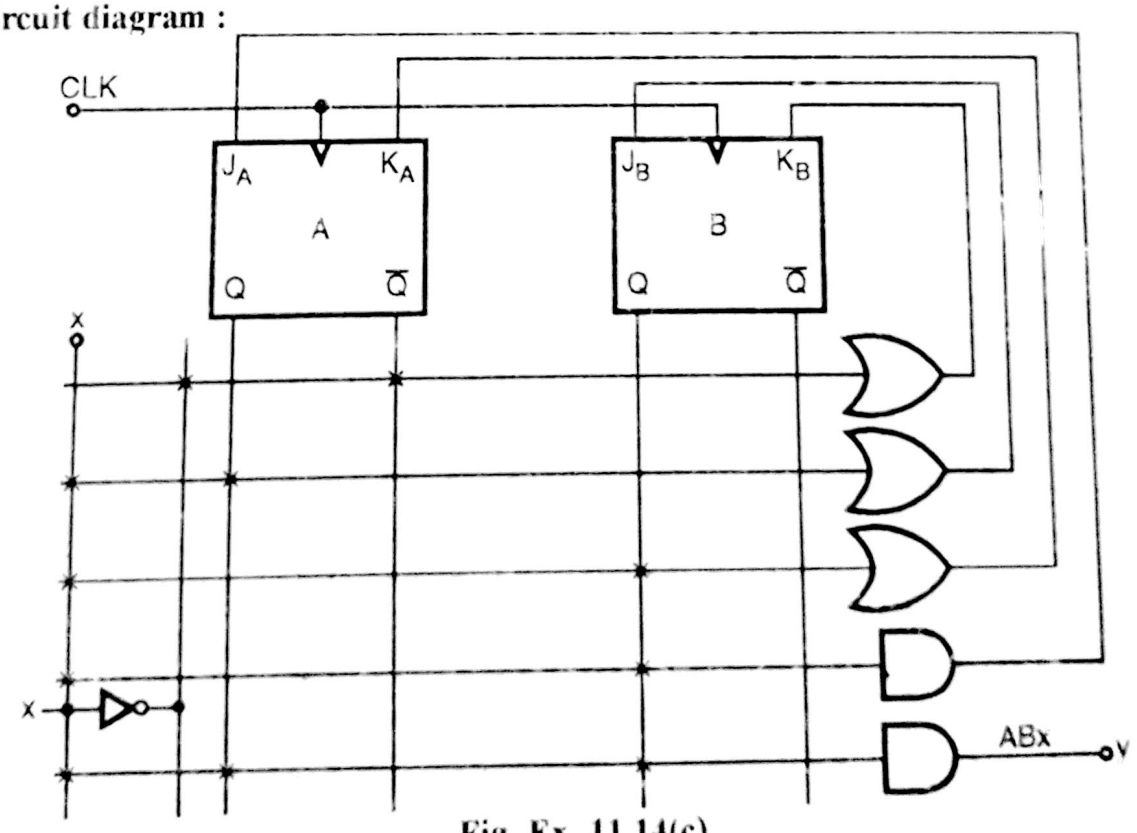



**Fig. Ex. 11.14(c)**

Scanned by CamScanner

**Ex. 15 :** Design a circuit to generate the output waveform as shown in Fig Ex 11 15(a).

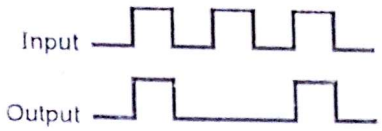



**Fig. Ex. 11.15(a)**

**Soln. :**

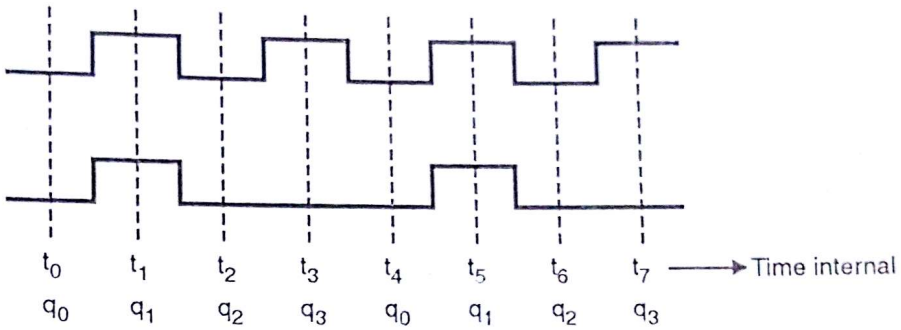



**Fig. Ex. 11.15(b)**

(1) First job we have to do is with respect to input, find out what will be an output. For that one has to record the instant where input change. When input changes at that instant what was output, the same should be taken into account. Fig. Ex. 11.15(b) shows graphical presentation of the same.

The output sequence is 01000100 with input sequence 0101010101 ... . If we observe the output sequence we get 0100 0100

$\therefore$ sequence actually is '0100'.

(2) Now we will draw state diagram. Let us say initial state is $q_0$. Refer Fig. Ex. 11.15(b) i.e. waveform. If input = 0, output = 0 and state is $q_0$. But if input = 1, output changes to 1 and state changes to $q_1$.

$\therefore \quad q_0 \xrightarrow{1/1} q_1$

Now when circuit is in $q_1$ state and input = 1 we get output = 1 and state is $q_1$ only. But if input = 0 output is also '0' and state changes to $q_2$. This way you proceed for $q_2$ to $q_3$ and $q_3$ to $q_0$, therefore, finally we get state diagram as shown in Fig. Ex. 11.15(c).

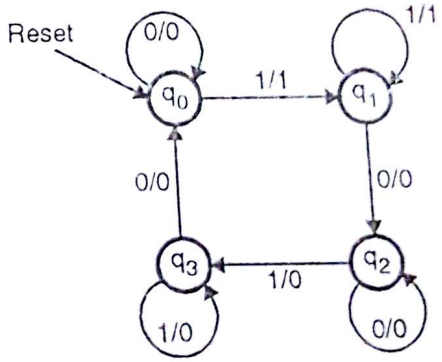



**Fig. Ex. 11.15(c)**

(3) Let us draw state table.

| Present state (AB) \ x | Present Input x = 0 | Present Input x = 1 |
|---|---|---|
| $q_0$ | $q_0$, 0 | $q_1$, 1 |
| $q_1$ | $q_2$, 0 | $q_1$, 0 |
| $q_2$ | $q_2$, 0 | $q_3$, 0 |
| $q_3$ | $q_0$, 1 | $q_3$, 0 |

(4) Total number of states are four i.e. $q_0$ to $q_3$. To represent these states we require two binary bits.

Scanned by CamScanner

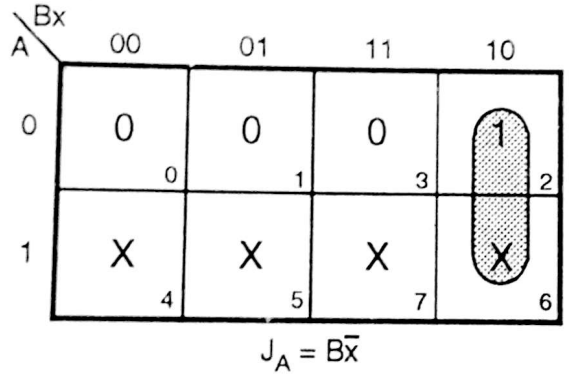
Therefore now assign binary value to the state

$q_0 = 00$
$q_1 = 01$
$q_2 = 10$
$q_3 = 11$

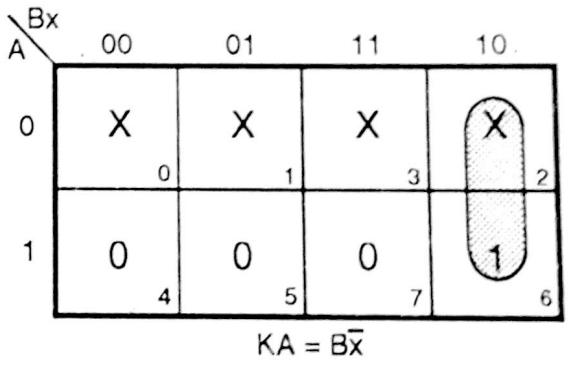
(5) Now finally draw state diagram in different method and write binary values

| Inputs of Combinational Circuit | | | | | Output from Combinational Circuit | | | | Output y |
|---|---|---|---|---|---|---|---|---|---|
| Present state | | Input | Next state | | Flip-flop Inputs | | | | |
| A | B | x | A | B | JA | KA | JB | KB | |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | X | 1 | X | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | X | X | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | X | X | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | X | 0 | 0 | X | 0 |
| 1 | 0 | 1 | 1 | 1 | X | 0 | 1 | X | 0 |
| 1 | 1 | 0 | 0 | 0 | X | 1 | X | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | X | 0 | X | 0 | 0 |

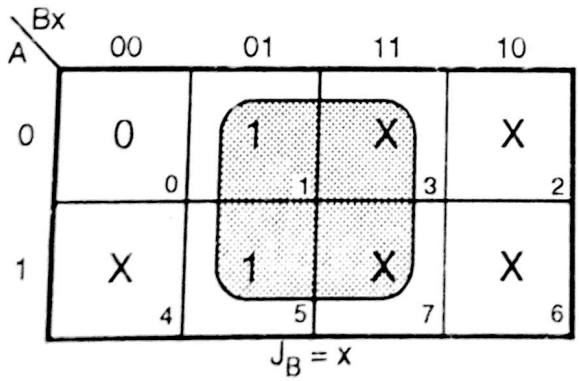
(6) **K-map :**



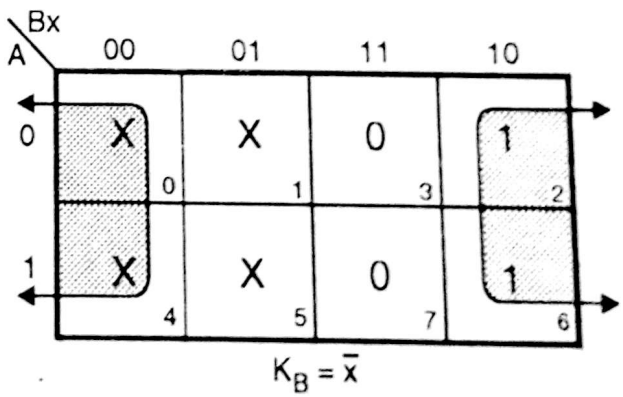

$J_A = B\bar{x}$



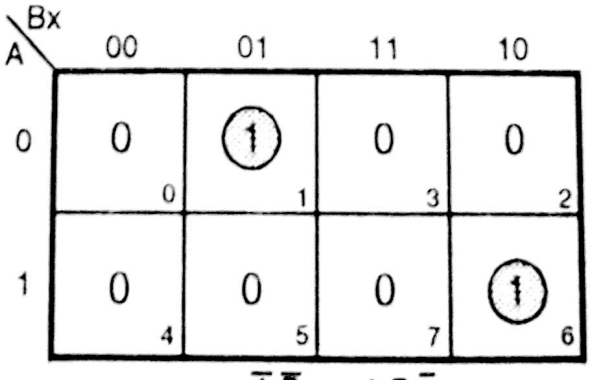

$KA = B\bar{x}$




$J_B = x$




$K_B = \bar{x}$




$y = \bar{A}\,\bar{B}\,x + A\,B\,\bar{x}$

**Fig. Ex. 11.15(d)**

Scanned by CamScanner

(7) **Logic diagram :**

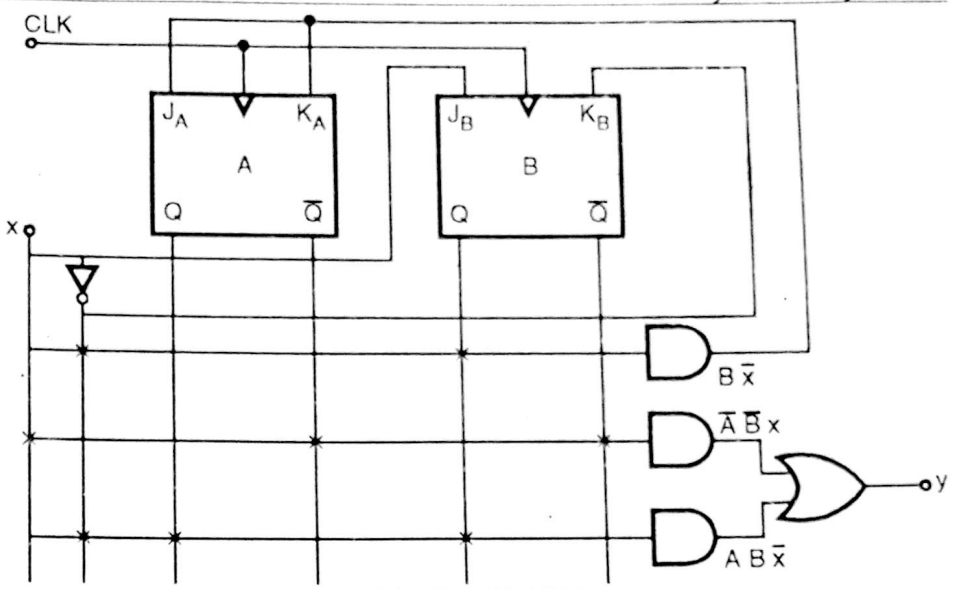



**Fig. Ex. 11.15(e)**

**Ex. 16 :** Fig. Ex. 11.16(a) layout of traffic light controller.

Our target is to design a digital system, which will control traffic light. This traffic controller (TC) is placed at intersection of two streets. The main job is to switch the lights ON and OFF in a sequence that responds to the flow of vehicles and pedestrians. We take single example where NS street intersects EW street. Function of TC is defined as follows :

(1) Whenever pedestrian want to cross the road he/she press Walk button.

(2) When no pedestrian is present, lights allow NS traffic to flow for one minute. Means NS-green and EX-Red.

(3) After that TC light changes automatically to allow EW traffic.to flow for one minute.

∴ EW-green, NS-Red.

(4) Thus green light alternate between NS and EW direction, at one minute interval.

(5) If 'walk' button is pressed, all the lights turned to Red, allowing pedestrian to walk for one minute.

No yellow light is considered.

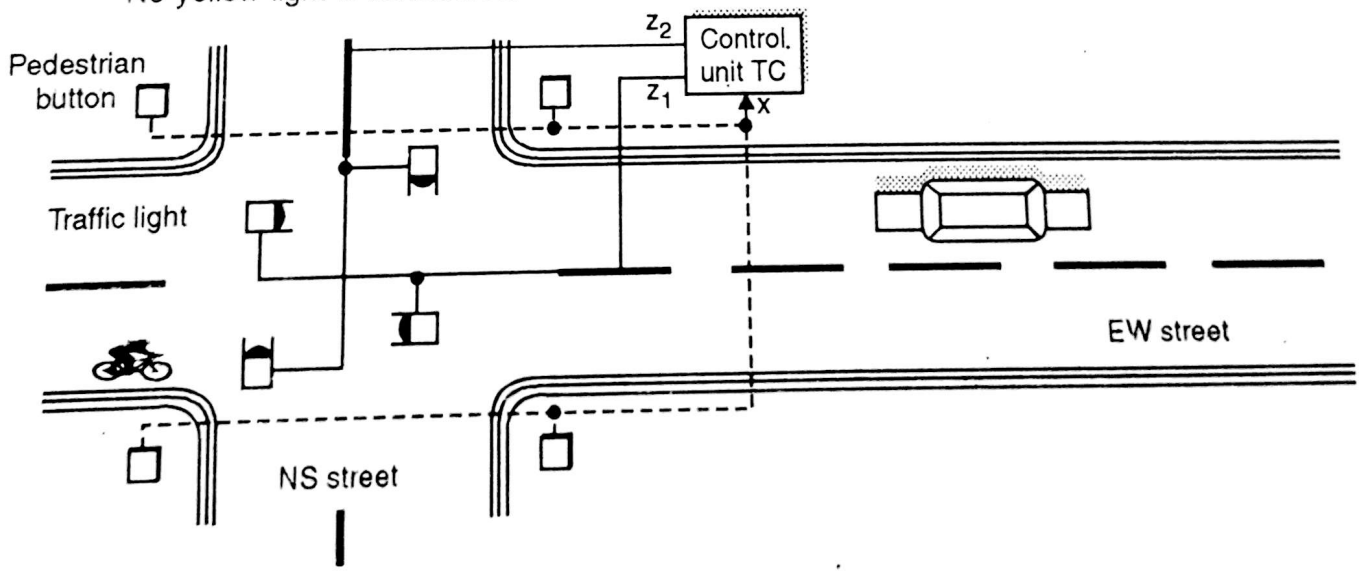



**Fig. Ex. 11.16(a)**

Scanned by CamScanner

**Soln. :** (1) Firstly we will analyse the problem and if possible, give some binary value to any particular case.

(a) Let us represent pedestrian button by single variable x.

$x = 1$ – Pedestrian has pressed

$x = 0$ – No pedestrians

(b) Circuit will have two outputs.

$z_1$ = Make light green in NS **OR** Red in EW.

$z_2$ = Make light Red in NS **OR** green in EW.

$\therefore$ All red colour situation means $z_1 = z_2 = 0$.

(c) Lastly the important point is *time*. We want to keep lights ON/OFF for 1 minute. The solution is, use clock whose period is of 1 minute.

(2) **State Assignment :** Now we have to think of states in this problem.

Just by going through the problem, you can say that, we have following number of states.

(1) $q_0 \rightarrow$ Green in NS and red in EW.

(2) $q_1 \rightarrow$ Green in EW and red in NS

(3) $q_2 \rightarrow$ Red in both, i.e. EW and NS.

These three states determine the output signals $z_1$ and $z_2$. State diagram is shown in Fig. Ex. 11.16(b).

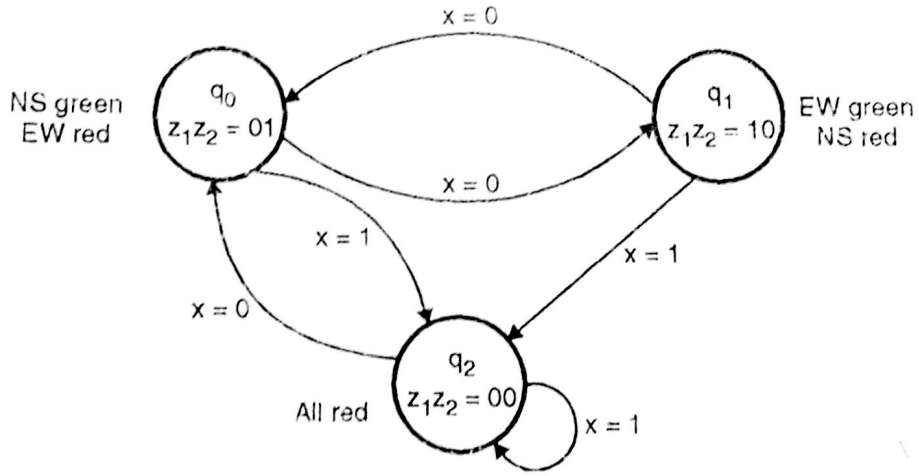



**Fig. Ex. 11.16(b)**

The drawn state diagram is functionally correct. Only important point is :

(1) When $q_2$ state is present and $x = 0$ (i.e. no pedestrian) then next state we prefer is $q_0$. This becomes bit unfair for vehicular traffic from EW. Both the direction should have equal opportunity for 'Red' as well as 'Green' light.

(2) Secondly, if number of pedestrians are more and each presses 'walk' button for long amount of time only pedestrian gets chance, vehicles will be standstill. To avoid these problem we draw next state diagram, which is as follows :

As shown in Fig. Ex. 11.16(c) if state is $q_0$ and $x = 1$, the next state will be $q_1$, i.e. lights will be red. After that if $x = 0$ or 1, after 1 minute it will switch over to state $q_1$.

Therefore Ex. 16 of point (1) is solved, i.e. after $q_0$ switch to $q_1$.

Sameway if present state is $q_1$, and $x = 1$, next state will be $q_3$. After $q_3$, $x = 0$ or 1, after exactly one minute it will switch over to state $q_0$.

Scanned by CamScanner

State $q_0$ and $q_1$ gives equal weightage

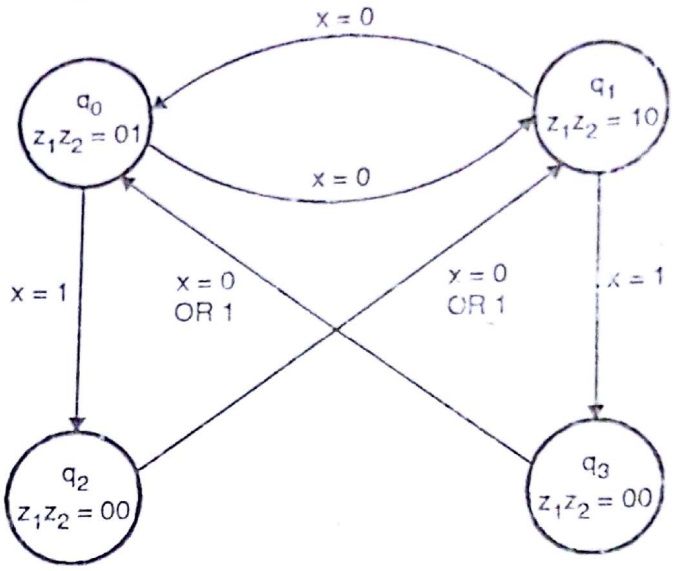



**Fig. Ex. 11.16(c)**

(3) Next job is to prepare state table.

| Present state | x / AB | Present input x = 0 | Present input x = 1 | Present output $z_1z_2$ |
|---|---|---|---|---|
| | $q_0$ | $q_1$ | $q_2$ | 01 |
| | $q_1$ | $q_0$ | $q_3$ | 10 |
| | $q_2$ | $q_1$ | $q_1$ | 00 |
| | $q_3$ | $q_0$ | $q_0$ | 00 |

Simply assign binary value to states. Total number of states are four, therefore we require two bits.

$q_0 = 00, \quad q_1 = 01, \quad q_2 = 10, \quad q_3 = 11$

(4) Therefore state table with binary value will be

| Present state | x / AB | Present Input x = 0 | Present Input x = 1 | Present output $z_1z_2$ |
|---|---|---|---|---|
| | 00 | 01 | 10 | 01 |
| | 01 | 00 | 11 | 10 |
| | 10 | 01 | 01 | 00 |
| | 11 | 00 | 00 | 00 |

(5) We are going to design this circuit using D FF. Therefore rearranging state table we get,

| Inputs of Combinational Logic | | | | | Output of Combinational Circuit | | | |
|---|---|---|---|---|---|---|---|---|
| Present state | | Input | Next state | | Flip-flop Inputs | | Outputs | |
| A | B | x | A | B | DA | DB | $z_1$ | $z_2$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Scanned by CamScanner

(6) **K-map :**

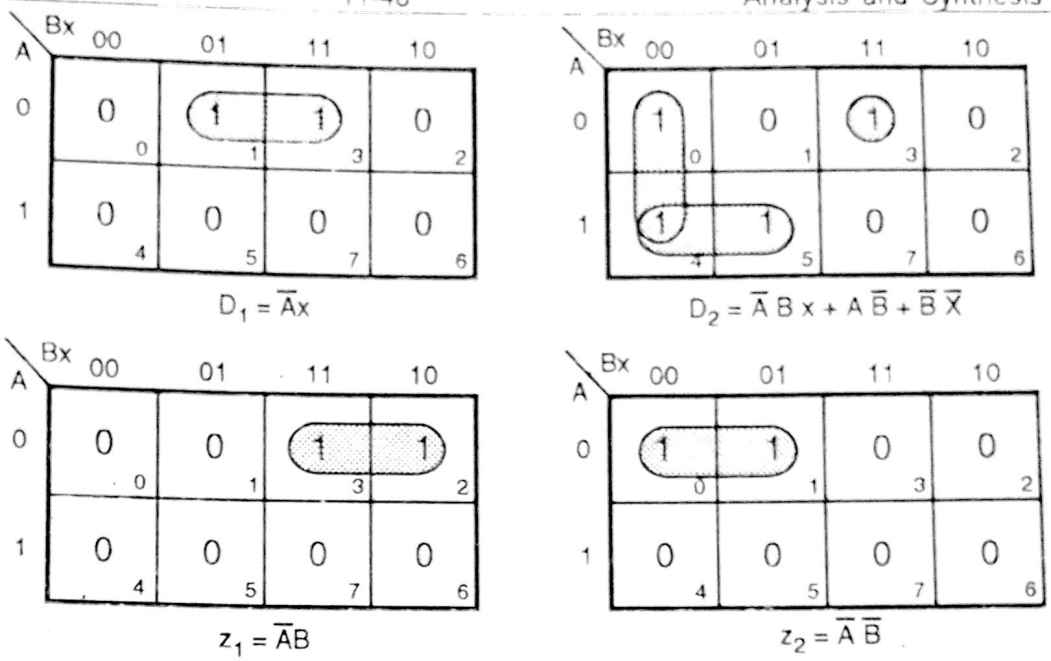



**Fig. Ex. 11.16(d)**

(7) **Logic diagram :**

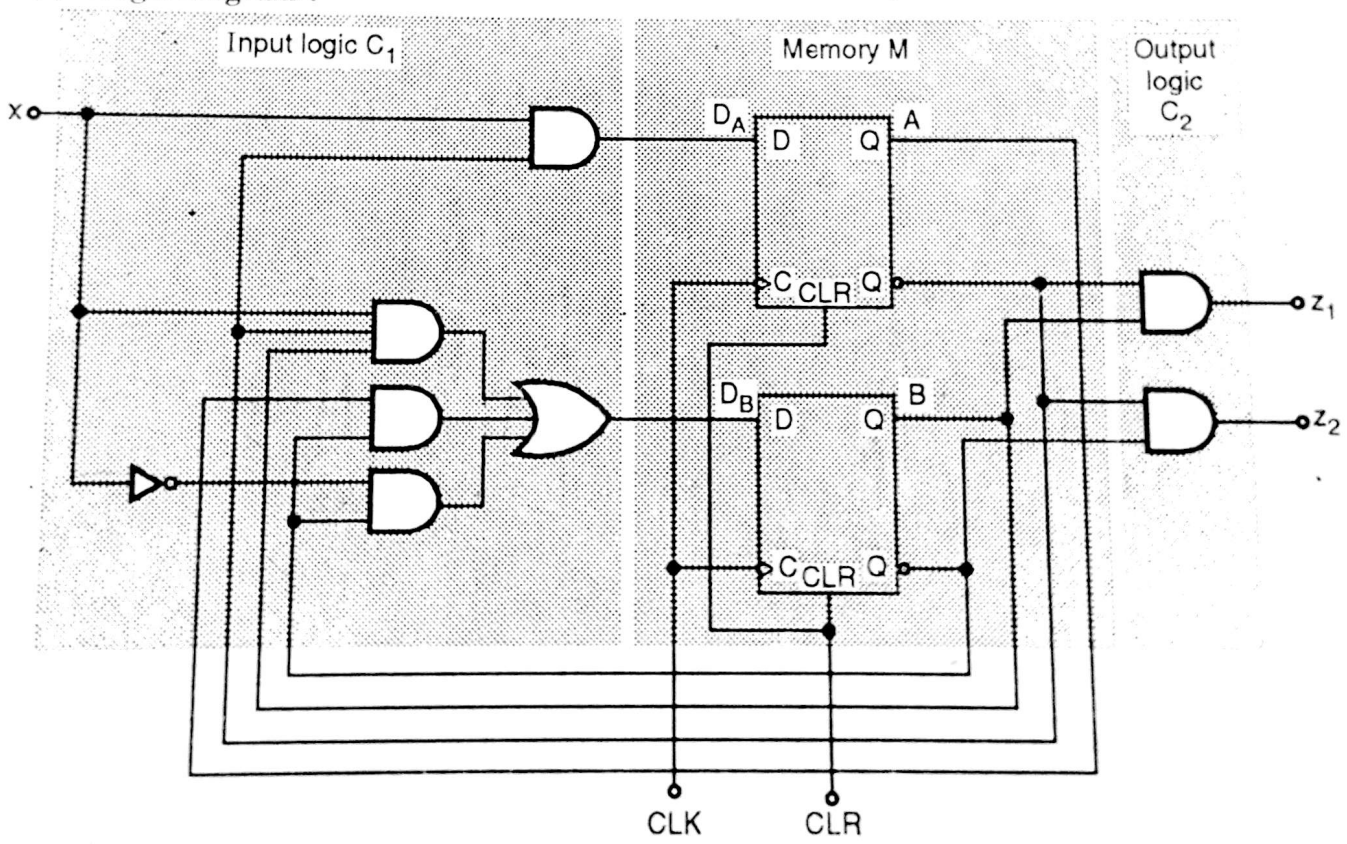



**Fig. Ex. 11.16(e)**

## 11.7 Design of Counters :

This you have already studied in Chapter 9. Counter is basically Moore circuit.

## 11.7(A) Design with State Equation (States) :

We can also design sequential circuit using state equation. State equation is nothing but an algebraic equation that gives the conditions for the next state as a function of present state and input variables. Information given in state table is given in, expression form by state equation.

Scanned by CamScanner

State equation method is convenient when the circuit is already specified in that form and when it is easy to derive expression from state table. If we use D FF, this method is preferred. This method is convenient. ***Sometime*** we use JK FF. Applying this procedure for circuit containing RS and T flip flop, involves considerable amount of algebraic manipulation. First we will see, designing using D FF.

**D FF design :**

The basic characteristic equation for D FF can be given as,

$$\underbrace{Q(n+1)}_{\text{Next state}} = \underbrace{D}_{\text{Input}}$$

It means that next state of flip flop is equal to present value of D input and is independent of the value of present state. It means that entries for the next state in state table are exactly same as that of D inputs.

Take one simple example if given is.

$$A = \Sigma (2, 4, 5, 6) = DA (A, B, x)$$
$$B = \Sigma (1, 3, 5, 6) = DB (A, B, x)$$

where A and B are FF outputs. DA and DB are flip flop input function. If we draw K map, we get,

$$DA = A\bar{B} + B\bar{x}$$
$$DB = \bar{A}B + \bar{B}x + AB\bar{x}$$

**JK FF design :**

The characteristic equation for JK FF is given by,

$$\underbrace{Q(n+1)}_{\text{Next state}} = \underbrace{J\bar{Q}}_{\text{Previous state}} + \underbrace{\bar{K}Q}_{\text{output}}$$

To get method of design we will take one example.

Suppose a sequential circuit with JK flip flops has to satisfy following state equations :

$$A(n+1) = \bar{A}\bar{B}CD + A\bar{B}\bar{C} + ACD + A\bar{C}\bar{D}$$
$$B(n+1) = \bar{A}C + C\bar{D} + \bar{A}B\bar{C}$$
$$C(n+1) = B$$
$$D(n+1) = \bar{D}$$

The input function for flip flop can be derived by following method :

***Arrange the state equation and match it with characteristic equation.***

Suppose we want input function for flip flop A, somehow manage to get form as $J\bar{A} + \bar{K}A$ (For moment treat J and K as simple variable).

$$A(n+1) = \bar{A}\bar{B}CD + A\bar{B}\bar{C} + ACD + A\bar{C}\bar{D}$$
$$= (\bar{B}CD + \bar{B}\bar{C}) \cdot \bar{A} + (CD + \bar{C}\bar{D})\, A$$
$$= (J_A)\,\bar{A} + (\bar{K}_A)\, A$$

$\therefore$ $J_A = \bar{B}CD + \bar{B}\bar{C}$

$\bar{K}_A = CD + \bar{C}\bar{D}$

$\therefore$ $K_A = \overline{CD + \bar{C}\bar{D}} = \bar{C}D + C\bar{D}$

**Same way for B (n + 1),**

Scanned by CamScanner

$$B(n+1) = \bar{A}C + C\bar{D} + \bar{A}B\bar{C}$$

$$= (\bar{A}C + C\bar{D}) + (\bar{A}\bar{C})\,B$$

$$= (\bar{A}C + C\bar{D})(\bar{B} + B) + (\bar{A}\bar{C})\,B$$

$$= (\bar{A}C + C\bar{D})\,\bar{B} + (\bar{A}C + \bar{A}\bar{C} + C\bar{D})\,B$$

$$= J_B\,\bar{B} + \bar{K}_B\,B$$

$$\therefore \quad J_B = \bar{A}C + C\bar{D}$$

$$\bar{K}_B = (\bar{A}C + \bar{A}\bar{C} + C\bar{D})$$

$$\bar{K}_B = \overline{(\bar{A}C + \bar{A}\bar{C} + C\bar{D})}$$

$$K_B = A\bar{C} + AD$$

$$C(n+1) = B$$

$$\Rightarrow B(C + \bar{C}) = BC + B\bar{C}$$

$$= J_C\,\bar{C} + \bar{K}_C\,C$$

$$\therefore \quad J_C = B$$

$$\bar{K}_C = B$$

$$\therefore \quad K_C = \bar{B}$$

$$D(n+1) = \bar{D} = 1 \cdot \bar{D} + 0 \cdot D = J_D\,\bar{D} + \bar{K}_D D$$

$$\therefore \quad J_D = K_D = 1 = V_{CC}$$

## 11.8 Multi-input Synchronous Sequential Circuit Design :

We have studied sequential synchronous circuit for designing application oriented circuit, counters etc. One important point is, uptill now we have taken single input line into account. Now in this section we will take more number of input lines, because sequential circuits are normally designed to work as controller for large systems, hence these systems are sometimes called as ***controller system***. In this section, we will study the design procedure for small system controllers. Let's solve some examples.

**Ex. 17 :** Design a controlled counter which has two inputs $x_1$ and $x_2$. The counter must count in following manner :

| $x_1$ | $x_2$ | States of counter |
|---|---|---|
| 0 | 0 | 3, 2, 1, 0 |
| 0 | 1 | 2, 0 |
| 1 | 0 | 1, 2 |
| 1 | 1 | 0, 1 |

Scanned by CamScanner

**Soln. :** 1. Diagrammatically circuit can be represented in block schematic as follows :

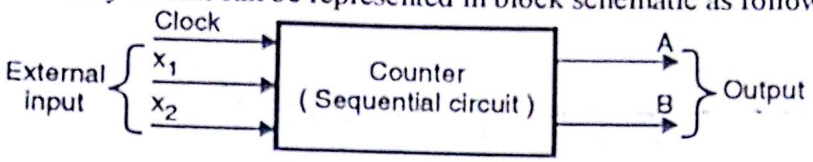
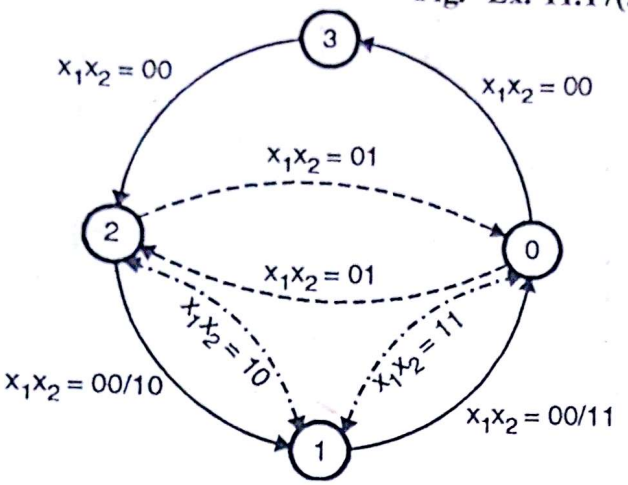



**Fig. Ex. 11.17(a)**




| | |
|---|---|
| ——— | $x_1x_2 = 00$ |
| - - - - - - | $x_1x_2 = 01$ |
| - · - · - · - | $x_1x_2 = 10$ |
| ——— · ——— | $x_1x_2 = 11$ |

**Fig. Ex. 11.17(b)**

2. Now observe given table carefully and concentrate on input and required sequence. Depending upon that draw state table. The same is shown in Fig. Ex. 11.17(b).
3. We are going to use D flip flop. Therefore now draw modified state table, consists of present state, present input, next state and flip flop inputs.

| | Present inputs | | Present state | | Next State | | Flip flop inputs | |
|---|---|---|---|---|---|---|---|---|
| | $x_1$ | $x_2$ | A | B | A | B | DA | DB |
| $x_1x_2 = 00$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| $x_1x_2 = 01$ | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| | 0 | 1 | 0 | 1 | X | X | X | X |
| | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 1 | 1 | 1 | X | X | X | X |
| $x_1x_2 = 10$ | 1 | 0 | 0 | 0 | X | X | X | X |
| | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| | 1 | 0 | 1 | 1 | X | X | X | X |
| $x_1x_2 = 11$ | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 1 | 1 | 1 | 0 | X | X | X | X |
| | 1 | 1 | 1 | 1 | X | X | X | X |

$= (01)_2$

Scanned by CamScanner

4. **K-Map :**

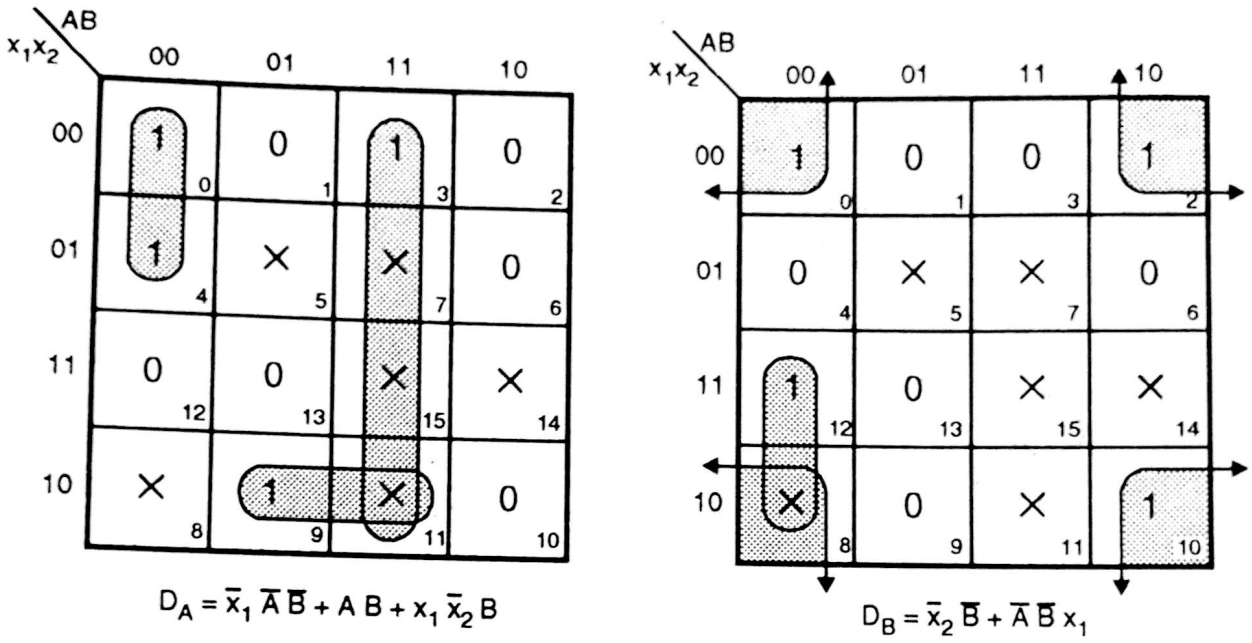



**Fig. Ex. 11.17(c)**

5. **Logic diagram :**

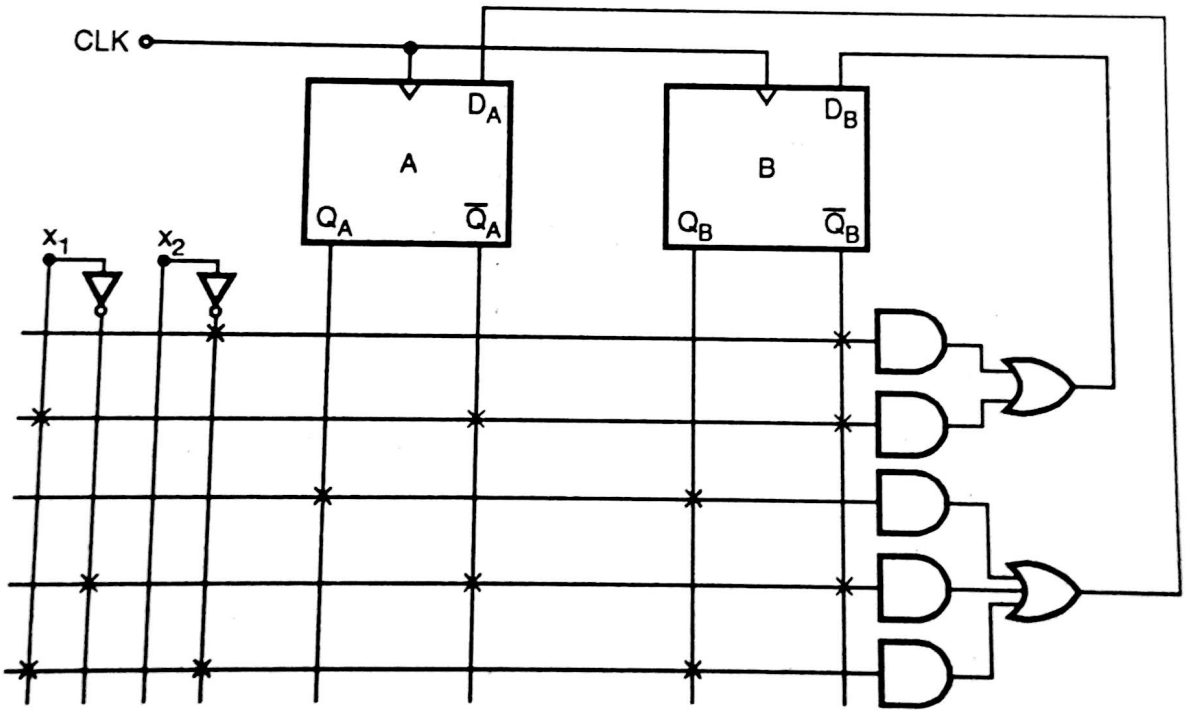



**Fig. Ex. 11.17(d)**

---

**Ex. 18 :** Design a fault detector with following description :

A fault signal 'F' activates an alarm bell, turns green light OFF and red light ON. The operator turns OFF the bell by pressing an acknowledge switch 'A'. When fault clears itself, red light turns OFF, green light turns ON and the bell is automatically reactivated to attract the operator's attention. The bell is turned OFF, when operator presses acknowledge button. Test button 'T' is used to test the circuit.

Scanned by CamScanner

**Soln. :**

1. Analyse problem

   It has three inputs :
   - Fault signal 'F'
   - Acknowledge switch 'A'
   - Test button 'T'

   It has three outputs :
   - Green light 'G'
   - Red light 'R'
   - Bell 'B'

2. Let's analyse state of circuit, one by one.

   (a) Initially consider that state so, where green light is ON and red is OFF, as well as bell is OFF.

   $S_0 \rightarrow G = 1, \quad R = 0, \quad B = 0$

   (b) In above state i.e. $S_0$, if we receive fault signal or test push button, green light will be OFF and Red light will be ON, with bell ON. At that time we have to change state, say $S_1$

   $S_1 \rightarrow G = 0, \quad R = B = 1.$

   (c) Now operator will press button 'A' (i.e. acknowledge). In response to it, bell will be OFF. Therefore circuit changes state to $S_2$. In $S_2 \rightarrow G = 0, R = 1, B = 0.$

   (d) Now circuit waits for fault clear condition. When fault is cleared, circuit switches to state $S_3$ and outputs $S_3 \rightarrow G = 1, R = 0, B = 1.$

   $\therefore$ Bell is activated.

   (e) Again circuit will expect acknowledge from operator. When operator acknowledge, by pressing button 'A', bell will stop, therefore circuit switches to $S_0$ state and $G = 1, R = 0, B = 0.$

Refer Fig. Ex. 11.18(a) for state diagram.

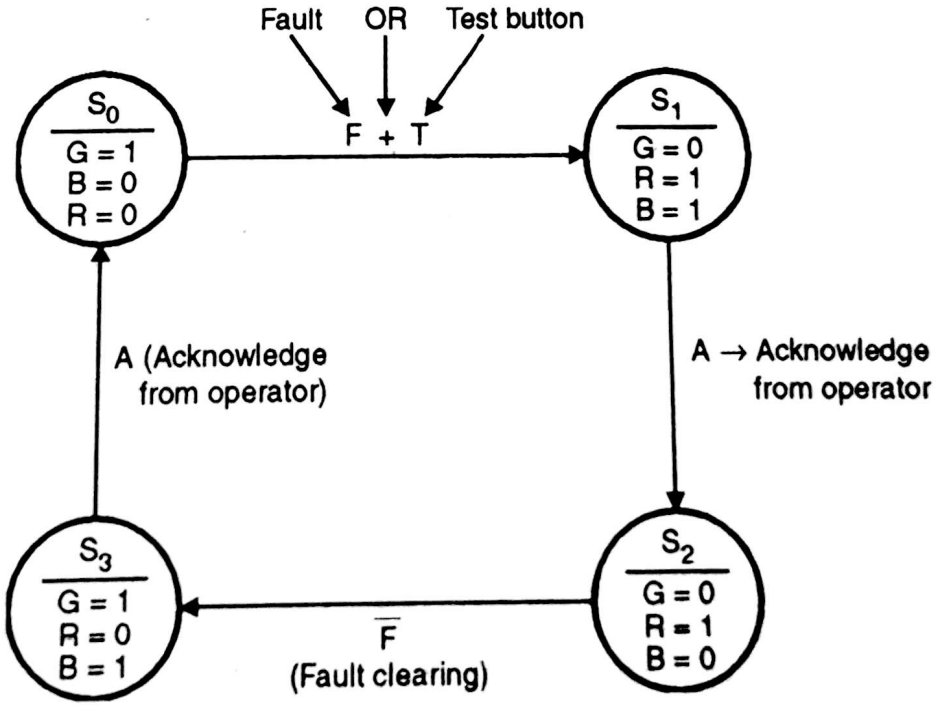



**Fig. Ex. 11.18(a)**

3. There are total four states $S_0 = (00)_2$, $S_1 = (01)_2$, $S_2\ (10)_2$ and $S_3\ (11)_2$.

Scanned by CamScanner

In totality we have five inputs, i.e. two bits for present state, fault, acknowledge and test. Three outputs are three G, R and B. Five inputs provides 32 combination. Drawing the same becomes bit complicated. To reduce that we follow bit short-cut method, which is explained as follows :

Now we have to prepare state table. The same is shown as follows :

| | Present state | | Inputs | | | Next state | | Output | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $y_1$ | $y_2$ | F | A | T | $Y_1$ | $Y_2$ | G | R | B |
| $m_1$ | 0 | 0 | 1 | × | 0 | 0 | 1 | 1 | 0 | 0 |
| $m_2$ | 0 | 0 | 0 | × | 1 | 0 | 1 | 1 | 0 | 0 |
| $m_3$ | 0 | 0 | 1 | × | 1 | 0 | 1 | 1 | 0 | 0 |
| $m_4$ | 0 | 1 | × | 1 | × | 1 | 0 | 0 | 1 | 1 |
| $m_5$ | 1 | 0 | 0 | × | × | 1 | 1 | 0 | 1 | 0 |
| $m_6$ | 1 | 1 | × | 1 | × | 0 | 0 | 1 | 0 | 1 |

Table shows only six rows, $m_1$ to $m_6$. Concentrate on $m_1$, $m_2$ and $m_3$. Suppose present state is '00' then if F or T either signal is present next state is '01'. Here signal 'A' is ***don't care***. F and T can have 01, 10 and 11 states. Same way when present state is '01', then only acknowledge A is important, F and T are considered as ***don't care***. Same way you go on filling the state table.

Next state $Y_1$ is function of $m_4$ and $m_5$.

$$\therefore \quad Y_1 = m_4 + m_5 = \bar{y}_1 y_2 A + y_1 \bar{y}_2 \bar{F}$$

Same way write equation for $Y_2$, G, R and B.

$$Y_2 = m_1 + m_2 + m_3 + m_5$$

$$= \bar{y}_1 \bar{y}_2 F\bar{T} + \bar{y}_1 \bar{y}_2 \bar{F}\bar{T} + \bar{y}_1 \bar{y}_2 FT + y_1 \bar{y}_2 \bar{F}$$

$$= \bar{y}_1 \bar{y}_2 F\bar{T} + \bar{y}_2 \bar{F} (y_1 + \bar{y}_1 T) + \bar{y}_1 \bar{y}_2 FT$$

$$= \bar{y}_1 \bar{y}_2 F (T + \bar{T}) + \bar{y}_2 \bar{F} (y_1 + T)$$

$$= \bar{y}_1 \bar{y}_2 F + \bar{y}_2 \bar{F} y_1 + \bar{y}_2 \bar{F} T = \bar{y}_2 (\bar{y}_1 F + y_1 \bar{F}) + \bar{y}_2 \bar{F} T$$

$$G = m_1 + m_2 + m_3 + m_6$$

$$= \bar{y}_1 \bar{y}_2 (\bar{F}\bar{T} + F\bar{T} + FT) + y_1 y_2 A$$

$$= \bar{y}_1 \bar{y}_2 (T + F\bar{T}) + y_1 y_2 A = \bar{y}_1 \bar{y}_2 (T + F) + y_1 y_2 A$$

$$R = m_4 + m_5 = Y_1$$

$$B = m_4 + m_6 = \bar{y}_1 y_2 A + y_1 y_2 A$$

$$= y_2 A (\bar{y}_1 + y_1) = y_2 A$$

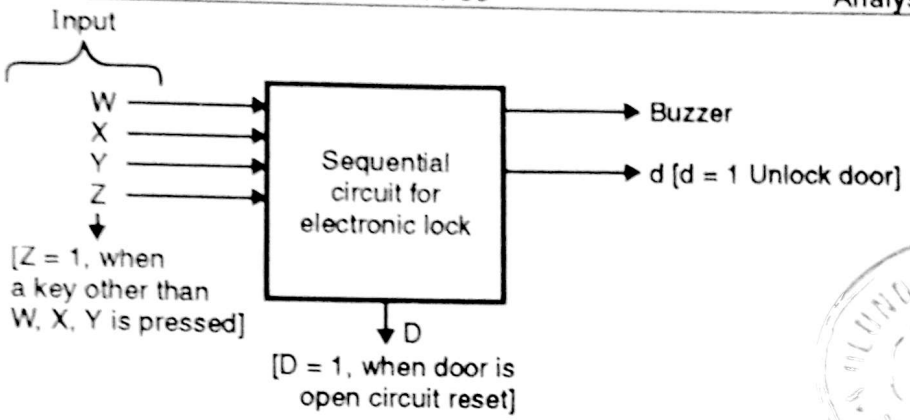
From boolean expression draw logic diagram.

---

**Ex. 19 :** Design a sequential circuit which unlocks when three specified keys on the keyboard are activated in sequence. If the wrong key is pressed at any time, the buzzer is turned on. Activating key in correct sequence when the buzzer is on, will turn off the buzzer and unlock the door, this allows circuit to accommodate genuine mistake.

**Soln. :** Let's first draw block schematic of the system.

Scanned by CamScanner




LIBRARY 743

**Fig. Ex. 11.19(a)**

1. We will consider that sequence has to be W, X and Y. This sequence is proper.
   Z is a signal which is generated when key other than W, X and Y is pressed.
   D is generated when door is open. It resets the circuit.
   Buzzer is for enabling/disabling buzzer.
   We will assume initial condition to be, door is closed and buzzer is OFF, circuit is in $S_0$ state.
2. Let's draw state diagram for the same.

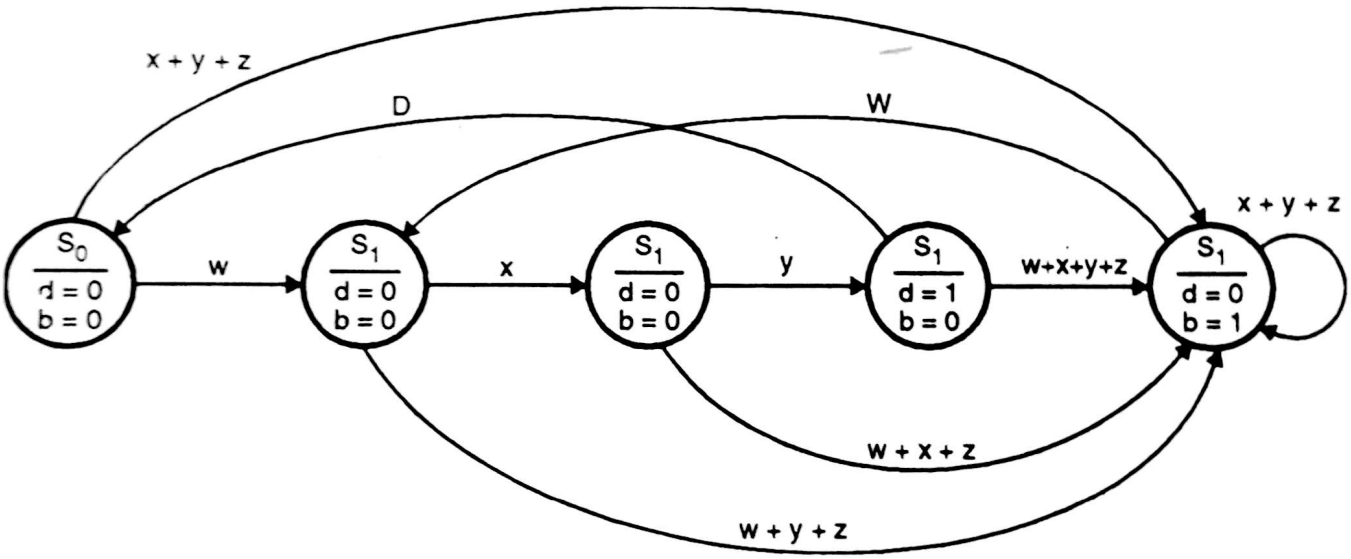



**Fig. Ex. 11.19(b)**

1. Initially state is $S_0$, therefore b = buzzer = 0 and d = 0.
2. If input is W, state changes from $S_0$ to $S_1$, else it will change to $S_4$. If state $S_1$, b = d = 0. But if state is $S_4$, b = 1, d = 0.
3. In state $S_1$, if input is x, next state will be $S_2$, else next state will be $S_4$. If next state is $S_2$, b = d = 0.
4. In state $S_2$, if input is y, next state will $S_3$ and in that state as proper sequence is generated d = 1 and b = 0. If key other than y is pressed next state will be $S_4$.

   Now door is open, therefore signal D will be generated and it will reset the full circuit.
   In state $S_3$, if we get key depression, next state will be $S_4$.
5. Finally when state is $S_4$ and W i.e. first correct key is pressed, next state will be $S_1$. This is nothing but accommodating genuine mistake. But if key other than W is pressed, it will be state $S_4$.

Scanned by CamScanner

3. Now let's draw state table.

| Present state | Present input (This checks only for true condition) W | X | Y | Z |
|---|---|---|---|---|
| $S_0$ | $S_1$, 0, 0, 0 | $S_4$, 0, 1, 0 | $S_4$, 0, 1, 0 | $S_4$, 0, 1, 0 |
| $S_1$ | $S_4$, 0, 1, 0 | $S_2$, 0, 0, 0 | $S_4$, 0, 1, 0 | $S_4$, 0, 1, 0 |
| $S_2$ | $S_4$, 0, 1, 0 | $S_4$, 0, 1, 0 | $S_3$, 1, 0, 1 | $S_4$, 0, 1, 0 |
| $S_3$ | $S_4$, 0, 1, 0 | $S_4$, 0, 1, 0 | $S_4$, 0, 1, 0 | $S_4$, 0, 1, 0 |
| $S_4$ | $S_4$, 0, 0, 0 | $S_4$, 0, 1, 0 | $S_4$, 0, 1, 0 | $S_4$, 0, 1, 0 |

↑ ↑ ↑

state d b

$S_0$ = 000

$S_1$ = 001

$S_2$ = 010

$S_3$ = 011

$S_4$ = 100

∴ State table will be

| Present state | | | Inputs | | | | Next state | | | Output | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $y_3$ | $y_2$ | $y_1$ | w | x | y | z | $Y_3$ | $Y_2$ | $Y_1$ | a | b | D |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | X | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | X | 1 | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | X | X | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | X | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | X | 0 | 1 | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | X | 0 | X | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | X | 0 | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | X | 1 | 0 | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | X | X | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | X | X | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | X | 1 | X | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | X | X | 1 | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | X | X | X | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | X | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | X | 1 | X | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | X | X | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

Now as we have done in previous example write equation for a, b and D.

Here input variables are 4 and state is 3 bit, so 7 bit K map is difficult to draw.

**Conclusion :** We have observed that for multiple input sequential circuit design we get 5, 6, 7 or more variable K map. Therefore design procedure is bit complicated. To write state table is also complicated. Therefore we take help of ASM (Algorithmic state machine) charts.

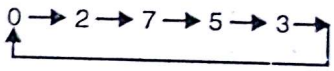
Scanned by CamScanner

## 11.9 Examples from University Papers :

1. For a synchronous counter with a sequence 0, 2, 7, 5, 3, 0
   (a) Set up the present state/next state table. (May 96, 4 Marks)
   (b) Set up exaction table for the inputs of flip–flops assuming D–type flip–flops. (May 96, 4 Marks)
   (c) Construct K–maps for the excitation inputs, simplify and realise the counter. (May 96, 6 Marks)
   (d) If the counter enters any of the unused states what will happen ? (May 96, 4 Marks)

**Ans. :**

**Step 1 :** The counter is non–sequential. Maximum count of the state is 7. To represent 7 in binary we require 3 bits (111). Therefore m = 3 and no. of flip–flops required are 3. We have to write don't care for invalid states. In above example invalid states are 1, 4 and 6.




**Step 2 :** We have to write excitation table for inputs of flip–flops using D–flip–flop.

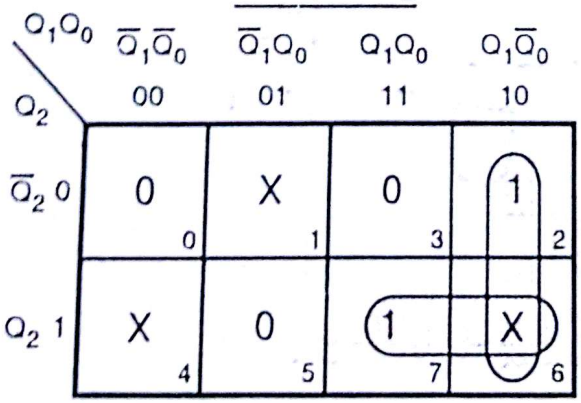
**Excitation table :**

| Present State | | | D inputs | | | Next State | | |
|---|---|---|---|---|---|---|---|---|
| $Q_{2n}$ | $Q_{1n}$ | $Q_{0n}$ | $D_2$ | $D_1$ | $D_0$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | X | X | X | X | X | X |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | X | X | X | X | X | X |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | X | X | X | X | X | X |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

**Table 1**

**Step 3 :** Now using Table 1 we have to construct K–maps for inputs of flip-flop.

K-map for $D_2$




$$D_2 = Q_1\overline{Q}_0 + Q_1 Q_2$$

Scanned by CamScanner

**K-map for $D_1$**

| $Q_2$ \ $Q_1Q_0$ | $\overline{Q}_1\overline{Q}_0$ 00 | $\overline{Q}_1Q_0$ 01 | $Q_1Q_0$ 11 | $Q_1\overline{Q}_0$ 10 |
|---|---|---|---|---|
| $\overline{Q}_2$ 0 | 1 (0) | X (1) | 0 (3) | 1 (2) |
| $Q_2$ 1 | X (4) | 1 (5) | 0 (7) | X (6) |

$$D_1 = \overline{Q}_1 + \overline{Q}_0$$

**K-map for $D_0$**

| $Q_2$ \ $Q_1Q_0$ | $\overline{Q}_1\overline{Q}_0$ 00 | $\overline{Q}_1Q_0$ 01 | $Q_1Q_0$ 11 | $Q_1\overline{Q}_0$ 10 |
|---|---|---|---|---|
| $\overline{Q}_2$ 0 | 0 (0) | X (1) | 0 (3) | 1 (2) |
| $Q_2$ 1 | X (4) | 1 (5) | 1 (7) | X (6) |

$$D_0 = Q_2 + Q_1\overline{Q}_0$$

**Circuit diagram for above sequence counter :**

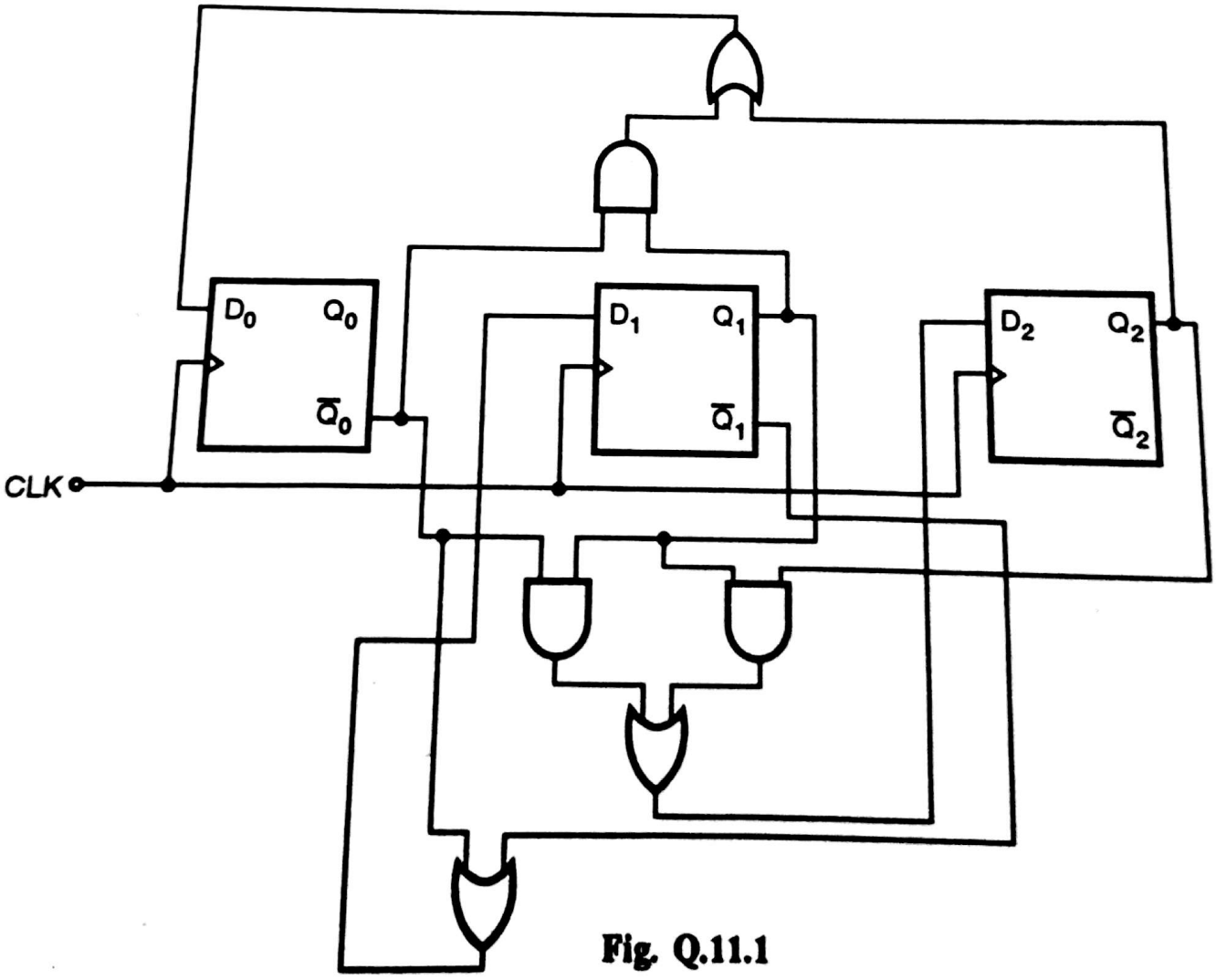



**Fig. Q.11.1**

Scanned by CamScanner

(d) The unused state of counter are 1, 4, 6.

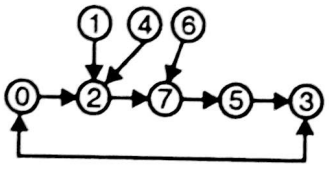
So we will have to check if counter enters any unused state, it will go to lockout condition or not. We will check this from logic circuit of above sequence counter.

| Decimal | Previous State | | | Input of D FF | | | Next State | | |
|---|---|---|---|---|---|---|---|---|---|
| | $Q_2$ | $Q_1$ | $Q_0$ | $D_2$ | $D_1$ | $Q_0$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

**Table 2**

From Table 2 if counter enters into any unused state it will not go to lockout condition. It returns to valid state.

The diagram of bushing is as follows :




**2. For a synchronous counter with a sequence 0, 1, 5, 6, 4, 2, 0.**

**(i) Set up present state/next state table.** **(May 97, Dec. 96, 3 Marks)**

**(ii) Set up excitation tables for the inputs of the flip–flops (assume D flip–flops).** **(May 97, Dec. 96, 4 Marks)**

**(iii) Construct K–maps for the excitation inputs, simplify and realise the counter.** **(May 97, Dec. 96, 6 Marks)**

**(iv) If the counter enters any of the unused states what will happen?** **(May 97, Dec. 96, 3 Marks)**

**Ans. :**

**Step 1 :** The counter is non–sequential. Maximum count of the state is 6.

$\therefore \quad 6 \leq 2^m$

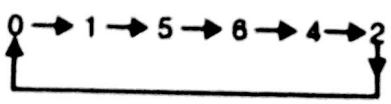
$\therefore \quad \boxed{m = 3}$

$\therefore$ No. of flip–flops are 3.

$$2^m = N = 2^3 = 8$$

$\therefore$ 0 – 7 states.

We have to write don't care for invalid states. In above example invalid states are 3 and 7.




**Step 2 : We have to write excitation table for inputs of flip–flops using D–flip–flop.**

Scanned by CamScanner

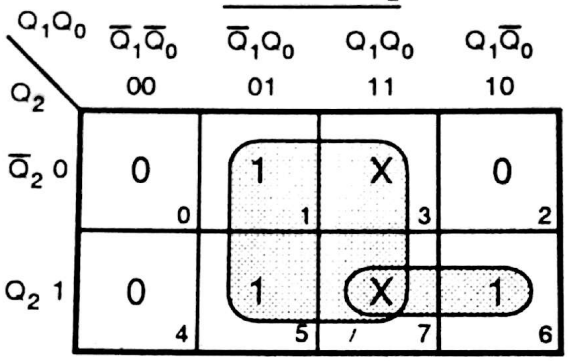
**Exciation table :**

| Present state | | | D FF inputs | | | Next state | | |
|---|---|---|---|---|---|---|---|---|
| $Q_{2n}$ | $Q_{1n}$ | $Q_{0n}$ | $D_2$ | $D_1$ | $D_0$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | X | X | X | X | X | X |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | X | X | X | X | X | X |

**Table 3**

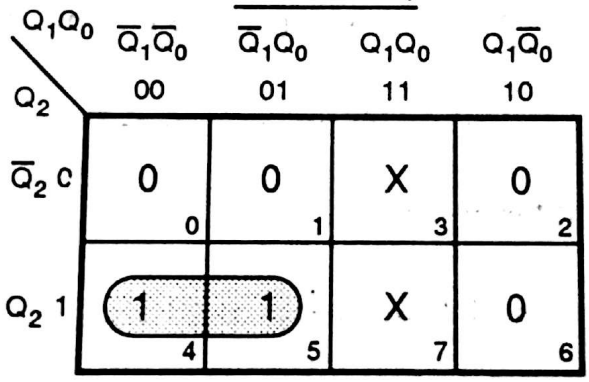
**Step 3 :** Now using Table 3 we have to construct K–maps for inputs of flip–flops as :

**K-map for $D_2$**



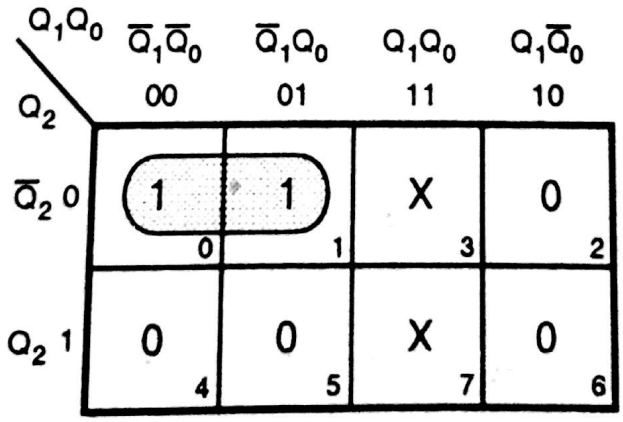

$D_2 = Q_0 + Q_2Q_1$

**K-map for $D_1$**




$D_1 = Q_2\overline{Q}_1$

**K-map for $D_0$**




$D_0 = \overline{Q}_2\overline{Q}_1$

**Step 4 : Circuit Diagram for above sequence counter**

Scanned by CamScanner

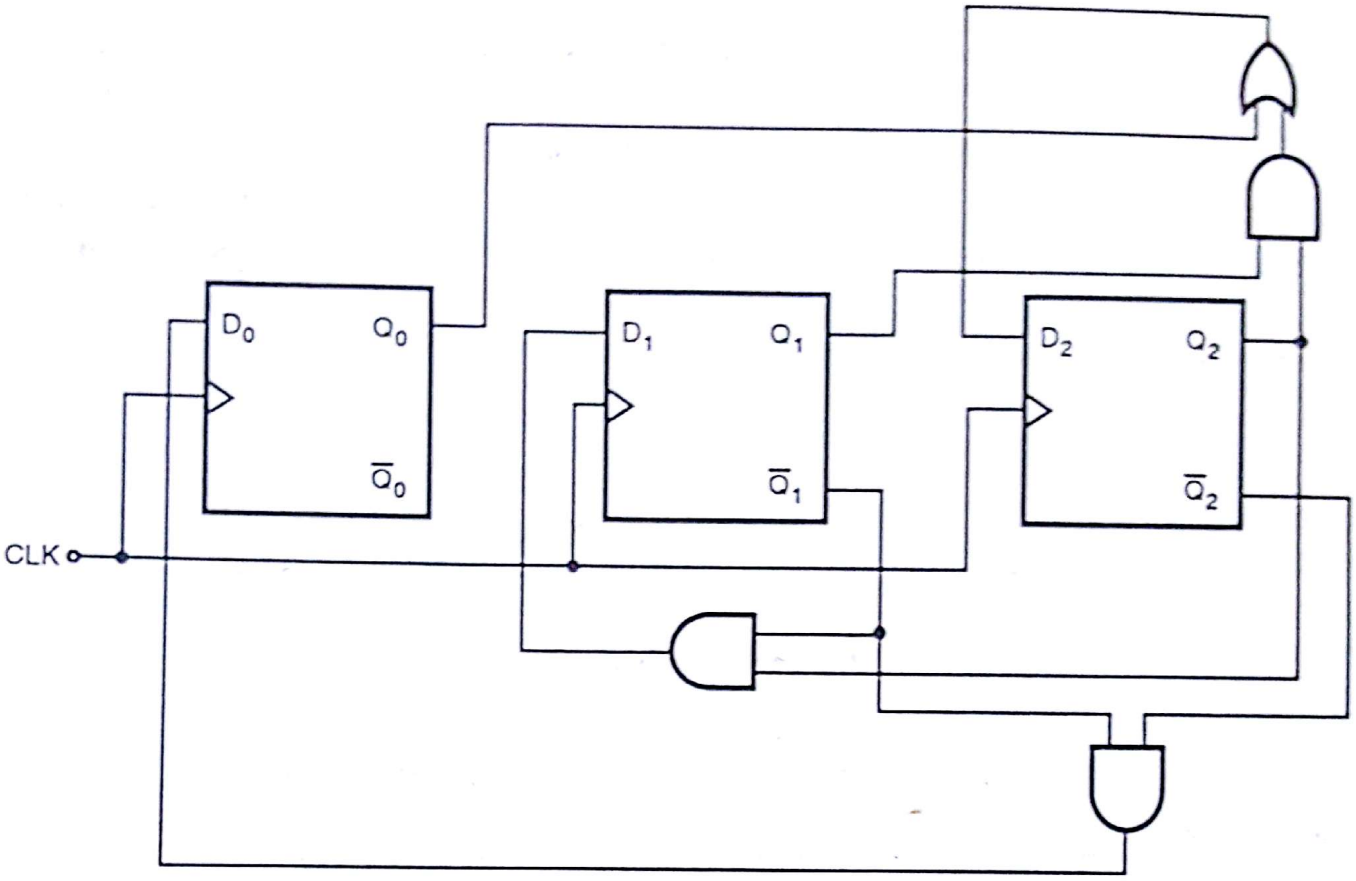



**Fig. Q.11.2**

**Step 5 :** The unused states of counter are 3 and 7.

So we will have to check if counter enters any unused state, it will go to lockout condition or not.

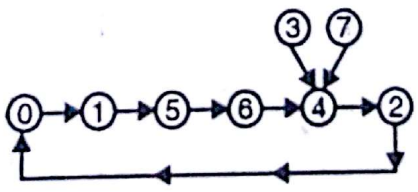
We will check this from circuit diagram of above sequence counter.

| Decimal | Previous State | | | Input to D FF | | | Next State | | |
|---|---|---|---|---|---|---|---|---|---|
| | $Q_2$ | $Q_1$ | $Q_0$ | $D_2$ | $D_1$ | $D_0$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| 3 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 7 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

**Table 4**

From Table 4, if counter enters any unused state it will not go to lockout condition, it returns to valid tate.

The diagram of bushing is as shown below :




**3. For a synchronous sequence counter with sequence 0, 2, 6, 5, 3, 1, 0**

**(a) Set up present state/next state table.** **(Dec. 97, 2 Marks)**

Scanned by CamScanner

**(b) Set up excitation table for inputs of flip–flop (assume D–flip–flop are used.)**
**(Dec. 97, 5 Marks)**

**(c) Construct K–maps for excitation inputs of flip–flop, simplify and realise the circuit.**
**(Dec. 97, 6 Marks)**

**(d) If the counter enters any unused state will it go to lockout condition ?(Dec. 97, 5 Marks)**

**Ans. :**

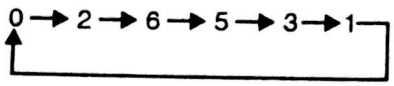
**Step 1 :** The counter is non–sequential. Maximum count of the state is 6.

$\therefore \quad 6 \leq 2^m$

$\therefore \quad \boxed{m = 3}$

Number of flip–flops are 3.

We have to write don't care for invalid states. In above example invalid states are 4 and 7.




**Step 2 :** We have to write excitation table for inputs of flip–flops using D–flip-flop.

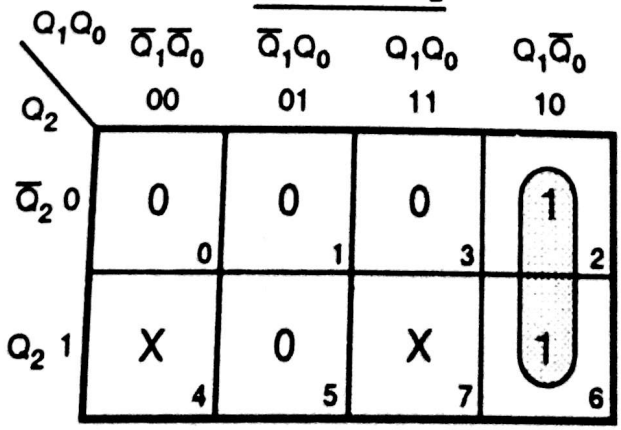
**Excitation table :**

| Present State | | | D FF Input | | | Next State | | |
|---|---|---|---|---|---|---|---|---|
| $Q_{2n}$ | $Q_{1n}$ | $Q_{0n}$ | $D_2$ | $D_1$ | $D_0$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | X | X | X | X | X | X |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | X | X | X | X | X | X |

**Table 5**

***Step 3 :*** *Using Table 5 we have to construct K–maps for inputs of D–type flip–flop.*

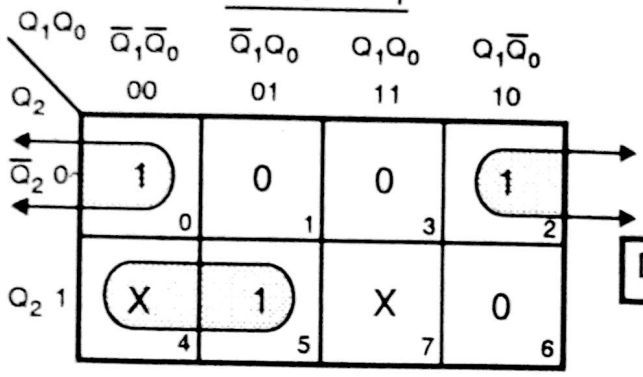
**K-map for $D_2$**




$\boxed{D_2 = Q_1\overline{Q}_0}$

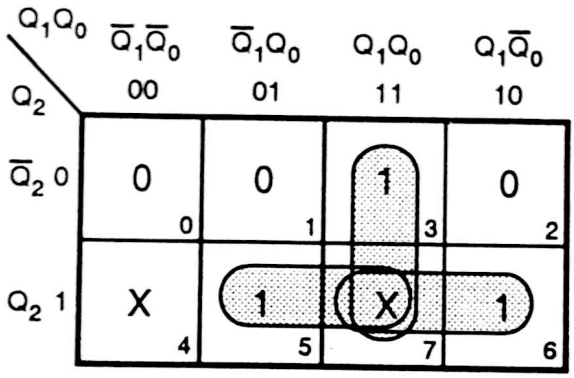
Scanned by CamScanner

**K-map for $D_1$**




$$D_1 = Q_2\overline{Q}_1 + \overline{Q}_2\overline{Q}_0$$

**K-map for $D_0$**




$$D_0 = Q_2 Q_0 + Q_2 Q_1 + Q_1 Q_0$$

$$D_0 = Q_2 (Q_0 + Q_1) + Q_1 Q_0$$

**Step 4 : Circuit Diagram :**

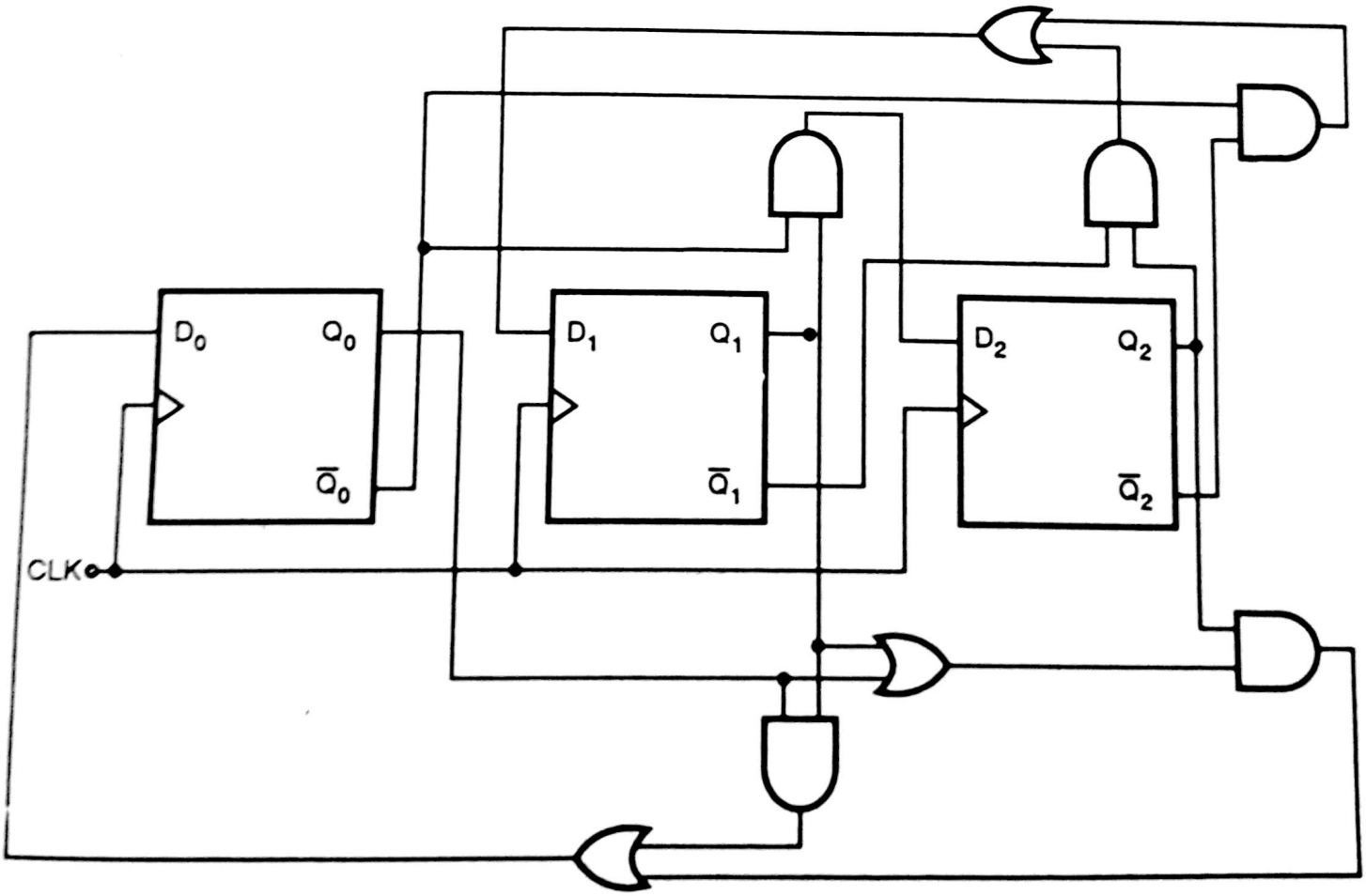



**Fig. O.11.3**

Scanned by CamScanner

**Step 5 :** The unused states of counter are 4 and 7.

So we will have to check if counter enters any unused state, it will go to lockout condition or not.

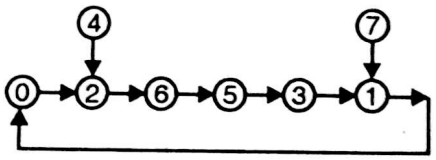
We will check this from circuit diagram of above sequence counter.

| Decimal | Previous State | | | Inputs to DFF | | | Next State | | |
|---|---|---|---|---|---|---|---|---|---|
| | $Q_2$ | $Q_1$ | $Q_0$ | $D_2$ | $D_1$ | $D_0$ | $Q_{2n+1}$ | $Q_{1n+1}$ | $Q_{0n+1}$ |
| 4 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 7 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

**Table 6**

FromTable 6, if counter enters any unused state ; it will not go to LOCKOUT condition. It returns to valid state.

The diagram of bushing is as shown below :




**4. A sequential circuit employing two J–K flip–flop is shown in Fig. Q. 11.4. Analyse the circuit and write equation for flip–flop output $Q_0$ and $Q_1$ and circuit output Z. Hence derive all possible entries of present state/next state table for this circuit.** **(June 99, Dec. 98, 10 Marks)**

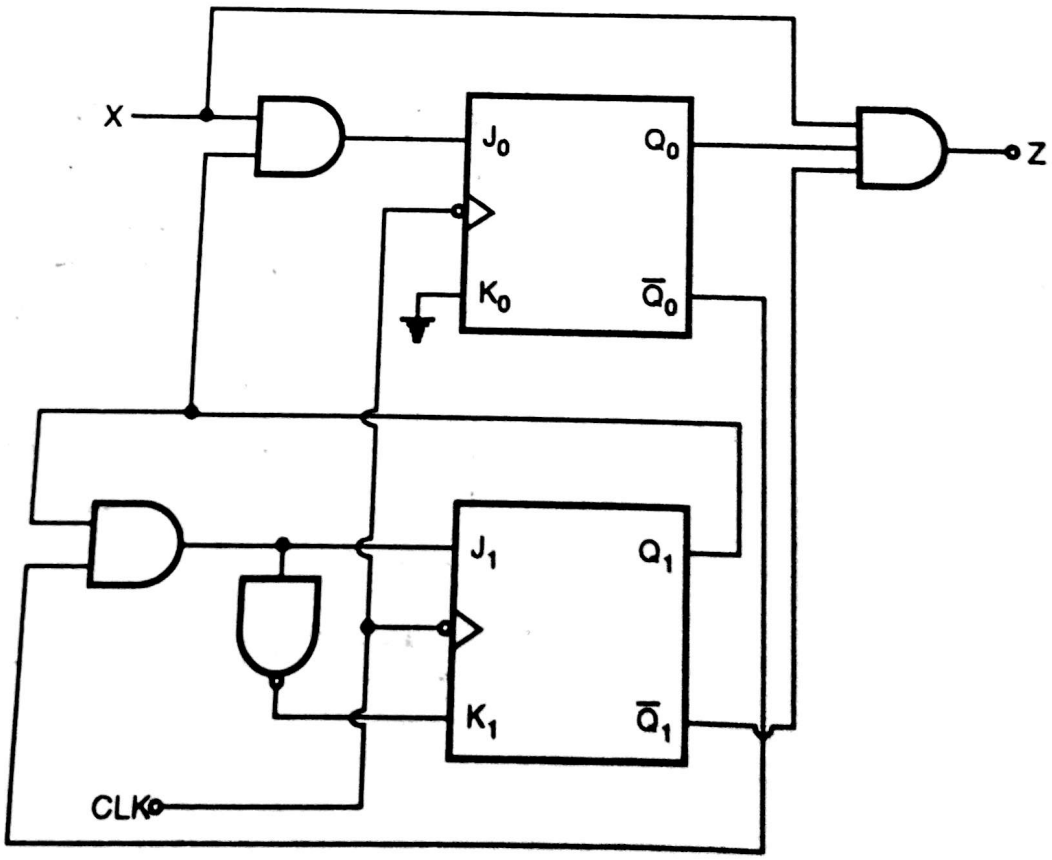



**Fig. Q. 11.4**

Scanned by CamScanner

**Ans. :** In the given circuit two J–K flip–flops are shown. Input to first flip–flop is X and $Q_1$ and given to $J_0$ through AND gate. Input to second flip–flop is $Q_1$ and $\bar{Q}_0$ which is given to $J_1$ through AND gate. Input to $J_1$ is inverted and given to $K_1$. $K_0$ is grounded.

**Present state/next state table :**

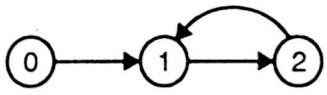
| Present State | | | | | dec | I/P to JK FF | | | | Next State | | | | Dec | $XQ_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | $\bar{Q}_1$ | $Q_1$ | $\bar{Q}_0$ | $Q_0$ | imal | $J_0 = xQ_1$ | $k_0 = 0$ | $J_1 = Q_1\bar{Q}_0$ | $k_1 = \bar{J}_1$ | $\bar{Q}_{1+1}$ | $Q_{1+1}$ | $\overline{Q_{0+1}}$ | $Q_{0+1}$ | imal | $\bar{Q}_1$ |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 2 | 0 |
| 0 | 0 | 1 | 0 | 1 | 3 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 3 | 0 |
| 1 | 0 | 1 | 0 | 1 | 3 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

**Table 7**

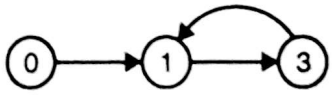
**State diagram :**

Either x = 0 or 1

If x = 0, the states of counter are as shown in state diagram below.




If x = 1, the states of counter are as shown in state diagram below :




Scanned by CamScanner

# PAL and PLA

12

**Syllabus :** Programmable logic devices, Programmable logic arrays and programmable array logic, Design using PAL, field programmable gate array.

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • PLA | 12.2 | ✓ | ✓ |
| • PAL | 12.3 | ✓ | ✓ |



**Comments**

- Base for PLD (Programmable logic devices)
- Need of present day

Scanned by CamScanner

## 12.1 Introduction :

While designing digital circuits we may require to implement AND/OR/INV circuits. There are chips available for AND/OR and INV. But whenever your circuit utilises more number of gates and complicated boolean functions, hardware, cost increases. The normal trend we follow is to go for custom IC chip, so that we can maintain secrecy. But in this case the affair becomes bit costlier if IC is not for long run. Because in custom built ICs one has to bare high start up cost. Considering this problem and to aid designer in using fewer chips and built complicated network IC manufacturers make semicustom chips in IC containers in which a basic two level gate network with many gates is produced and the gates can be interconnected on the chip as desired. These ICs are called Programmable logic arrays (PLAs) and Programmable Array Logic (PALs). Basically PAL is registered trademark of Monolithic Memories.

## 12.2 PLA (Programmable Logic Array) :

Basically PLA Consists of

(1) AND Matrix.

(2) OR Matrix.

Depending upon user's requirement connections in the Matrix are ***FUSED***.

These are two different ways of manufacturing PLA.

**I$^{st}$ :** Manufacturer places fused connections at every intersection point in the PLA between the inputs and the AND gates, and between AND and OR gates. Thus every possible connection is made when the PLA is manufactured and the desired connections are removed by blowing the fuses. This type of PLA often called as FPLA (Field Programmable Logic Array).

**II$^{nd}$ :** Desired connections are made during manufacturing. Manufacturer originally makes IC array layout so that any desired connections can be made, and the designer informs manufacturer which connections to make for a particular design. Manufacturer creates 'mask', which provides desired connections when layer of metalization are added to the chip during manufacturer. Setting of this mask costs for less than designing an entire new chip with precise logic array desired by logic designer.

Firstly we will study layout for small PLA.

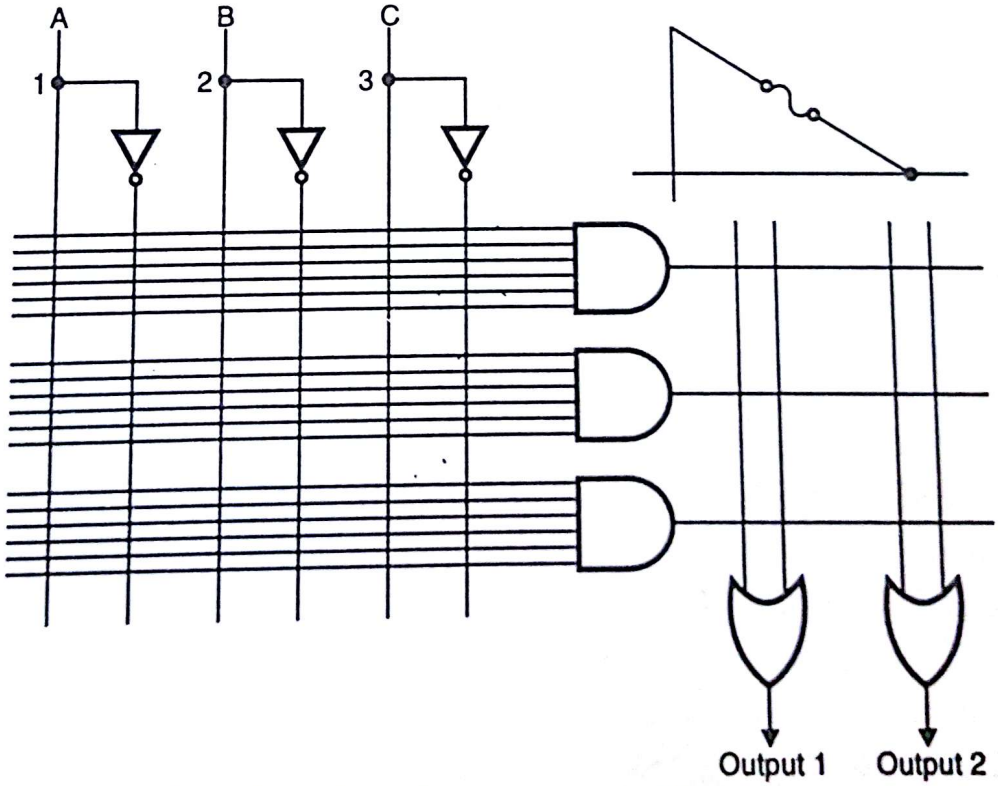



**Fig. 12.1 : Basic three input two output PLA**

Scanned by CamScanner

The PLA array has three AND gates and two OR gates. The connections from inputs A, B, C to the AND gates are not complete and that the AND gate outputs are not connected to OR gates. The connections are made as desired by gate network designer.

Suppose for this particular PLA we want

Output 1 = $\bar{A}B\bar{C} + \bar{A}\bar{B}$

Output 2 = $\bar{A}\bar{B} + B\,C$

Then circuit will be as shown in Fig. 12.2.

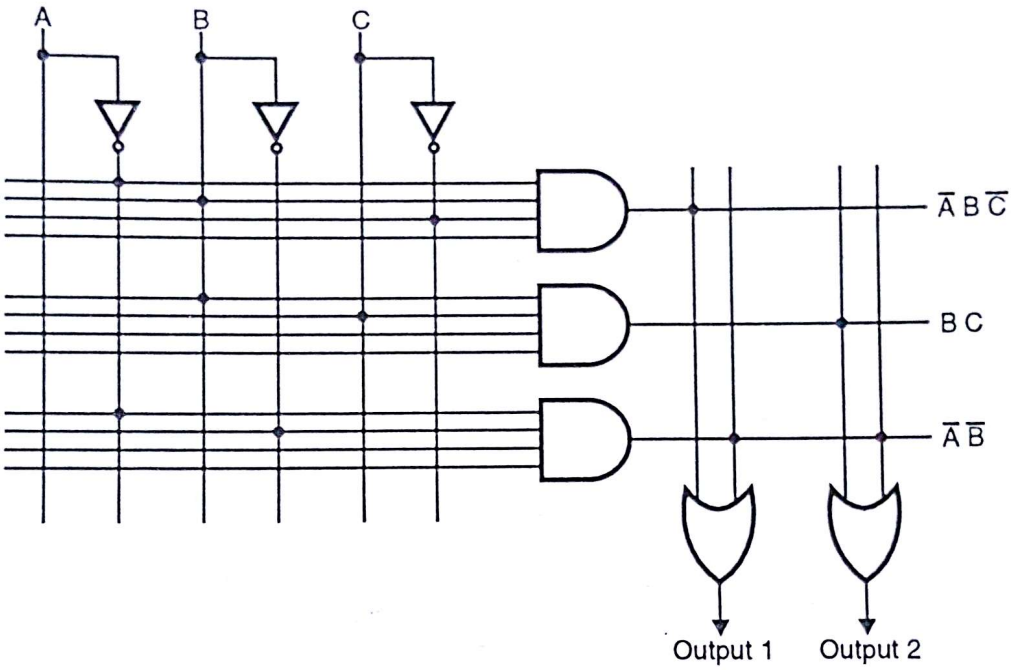



**Fig. 12.2 : Connection diagram for PLA**

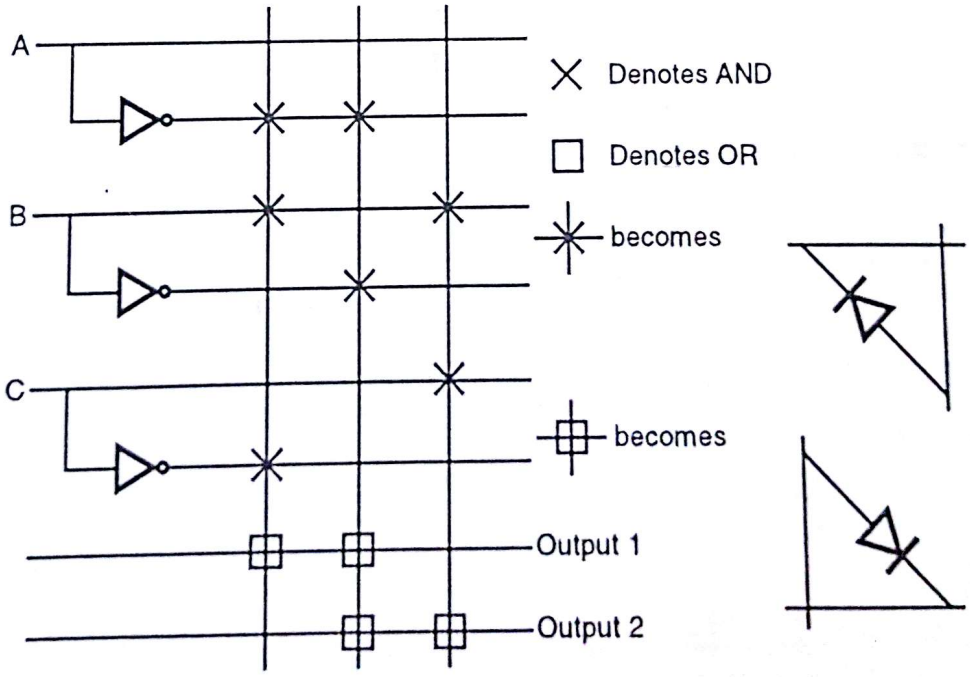



**Fig. 12.3 : Symbol used to draw PLA design**

Scanned by CamScanner

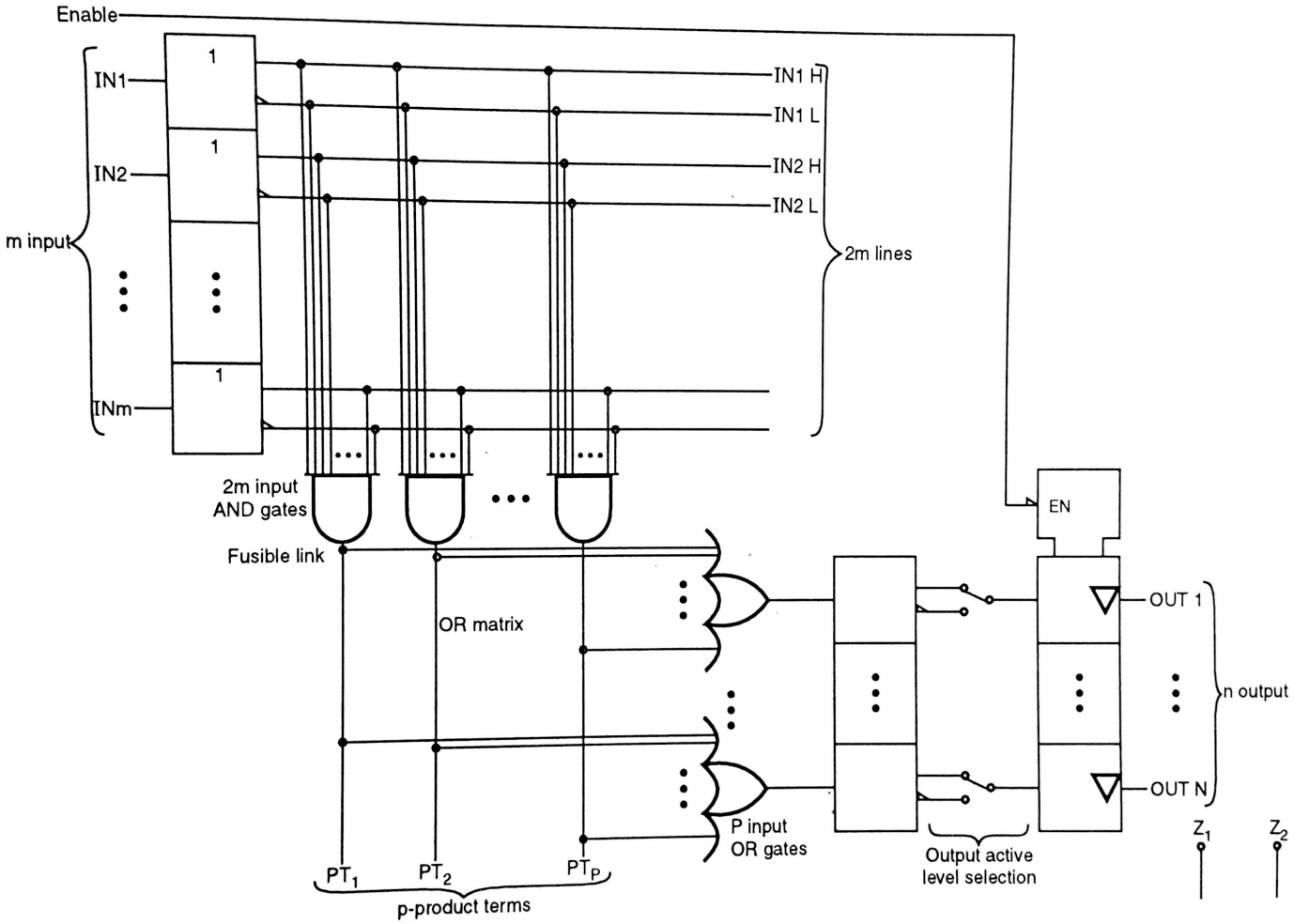



**Fig. 12.4**

Scanned by CamScanner

In practice we have larger PLA contain several hundred gates. Therefore to design larger arrays simplifying symbology is used.

Fig. 12.3 shows, crosses drawn on the function indicate ANDs, and square indicates ORs. The AND can be realised by single diode, the OR can be realised by diode pointing other way. In practice, manufactures lays out chip with diodes at every intersection of the lines, and only desired diode connections are made. Fig. 12.3 shows example of Fig. 12.2.

In FPLA we can also program active level of each output. ∴ Programming FPLA consists of

(1) Programming AND Matrix.

(2) Programming OR matrix.

(3) Programming active level of each output.

The logic of m × p × n FPLA is shown in Fig. 12.4.

Where m = number of inputs

p = number of product terms.

n = number of outputs.

P Product terms ANDs together, any number of the inputs (either active HIGH or active LOW).

Product term is not required to be function of all inputs. It can treat any number of inputs as don't care inputs.

Each of n outputs are formed by ORing together any number of (active HIGH) product terms. Finally we can select whether output is active HIGH or active LOW.

Let's see if we make m × p × n matrix how many fusible links we require. Here we normally have one fusible link to set active level of o/p.

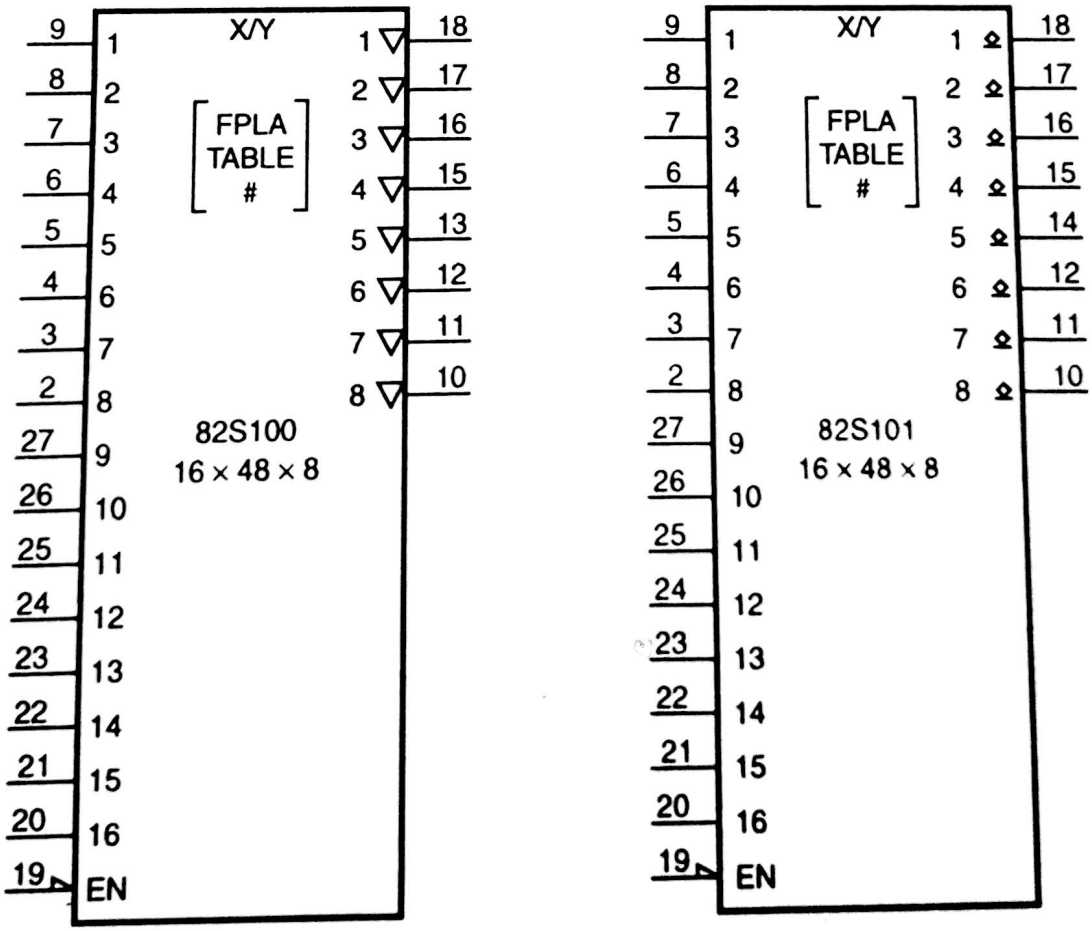



**Fig. 12.5 : FPAL 82S100 and 82S101**

Scanned by CamScanner

16 X 48X8 FPLA PROGRAM TABLE

| PROGRAM TABLE ENTRIES | | | | | | |
|---|---|---|---|---|---|---|
| INPUT VARIABLE | | | OUTPUT FUNCTION | | OUTPUT ACTIVE LEVEL | |
| Im. | $\overline{Im}$ | Don't Care | Prod.Term Present in Fp | Prod Term Not Present in Fp | Active High | Active Low |
| H | L | – (dash) | A | •(period) | H | L |
| NOTE Enter ↵ for unused inputs of used P-terms | | | NOTES 1)Entries independent of output polarity 2)Enter A for used Outputs of used P-terms | | NOTES 1)Polarity programmed once only 2)Enter(H) for all used outputs | |

| NO | PRODUCT TERM INPUT VARIABLE | | | | | | | | | | | | | | | | ACTIVE LEVEL OUTPUT FUNCTION | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | | | | | | | | | | | | | |
| 23 | | | | | | | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 | | | | | | | | | | | | | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | | | | | | | | | | | |
| 28 | | | | | | | | | | | | | | | | | | | | | | | | |
| 29 | | | | | | | | | | | | | | | | | | | | | | | | |
| 30 | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | | | | | | | | | | | | | | | | | | | | | | | | |
| 32 | | | | | | | | | | | | | | | | | | | | | | | | |
| 33 | | | | | | | | | | | | | | | | | | | | | | | | |
| 34 | | | | | | | | | | | | | | | | | | | | | | | | |
| 35 | | | | | | | | | | | | | | | | | | | | | | | | |
| 36 | | | | | | | | | | | | | | | | | | | | | | | | |
| 37 | | | | | | | | | | | | | | | | | | | | | | | | |
| 38 | | | | | | | | | | | | | | | | | | | | | | | | |
| 39 | | | | | | | | | | | | | | | | | | | | | | | | |
| 40 | | | | | | | | | | | | | | | | | | | | | | | | |
| 41 | | | | | | | | | | | | | | | | | | | | | | | | |
| 42 | | | | | | | | | | | | | | | | | | | | | | | | |
| 43 | | | | | | | | | | | | | | | | | | | | | | | | |
| 44 | | | | | | | | | | | | | | | | | | | | | | | | |
| 45 | | | | | | | | | | | | | | | | | | | | | | | | |
| 46 | | | | | | | | | | | | | | | | | | | | | | | | |
| 47 | | | | | | | | | | | | | | | | | | | | | | | | |

**Fig. 12.6 : Program table for a PLA**

Scanned by CamScanner

∴ Output fuses = n
AND Matrix fuses = $2 \times m \times p$
OR Matrix fuses = $p \times n$
Total = $n + 2mp + pn$

Now let's study some chips and programming methods for the same. Fig. 12.5 shows, FPLA from signetics i.e. 82S100 and 82S101.

We have to prepare truth table to specify contents of FPLA. The same is shown in Fig. 12.6.

We will use signetics programming notation.

1. Product terms
   - H = Active – high input
   - L = Active – low input
   - – = input ignored (don't care)
2. Output Functions
   - A = Product term included
   - . = Product term ignored (don't care)
3. Output active level
   - H = active – high output
   - L = active – low output.

We will take same previous example and try to prepare truth table.

16 X 48X8 FPLA PROGRAM TABLE

| PROGRAM TABLE ENTRIES | | | | | | |
|---|---|---|---|---|---|---|
| INPUT VARIABLE | | | OUTPUT FUNCTION | | OUTPUT ACTIVE LEVEL | |
| Im | $\overline{Im}$ | Don't Care | Prod.Term Present in Fp | Prod Term Not Present in Fp | Active High | Active Low |
| H | L | – (dash) | A | •(period) | H | L |
| NOTE Enter for unused inputs of used P-terms | | | NOTES 1)Entries independent of output polarity 2)Enter A for used Outputs of used P-terms | | NOTES 1)Polarity programmed once only 2)Enter(H) for all used outputs | |

| | PRODUCT TERM INPUT VARIABLE | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NO | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 6 | 3 | 2 | 1 | 0 |
| 0 | – | – | – | – | – | – | – | – | – | – | – | – | – | L | H | L |
| 1 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | L | L |
| 2 | – | – | – | – | – | – | – | – | – | – | – | – | – | H | H | – |
| 3 | | | | | | | | | | | | | | | | |
| 4 | • | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | |

| ACTIVE LEVEL OUTPUT FUNCTION | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| • | • | o | o | o | o | o | A |
| o | o | o | o | o | o | A | A |
| o | o | o | o | o | o | A | o |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

**Fig. 12.7**

(1) The logic input variables are identified with ***input variable*** number 0 to 15 on the table.
Let's say input 2 = C
input 1 = B
input 0 = A

(2) We are interested in $\bar{A}\,B\,\bar{C}$, $\bar{A}\bar{B}$ and $B\bar{C}$.
The Rule is
IF AN INPUT IS NOT COMPLEMENTED, 'H' IS WRITTEN IN THE TABLE. IF THE INPUT IS TO BE COMPLEMENTED, 'L' IS WRITTEN. IF INPUT NOT USED, '–' IS WRITTEN.

Scanned by CamScanner

We wish to form $\bar{A}B\bar{C}$, $\bar{A}\bar{B}$ and BC. So to form $\bar{A}B\bar{C}$ we will write L, H and L under column 2, 1 and 0.

(3) The OR gate inputs are written as follows. If an AND term in a particular row is to be used in an OR output, an A is written in the row, IF not, a '.' is written. In Fig. 12.7 OUTPUT FUNCTION LINE 0 in table associated with output 1 and OUTPUT FUNCTION LINE 1 corresponds to output 2.

Output 1 = $\bar{A}B\bar{C} + \bar{A}\bar{B}$

Output 2 = $\bar{A}\bar{B} + BC$.

## 12.3 PAL (Programmable Array Logic ) :

The PAL represents another alternative to PROMS. This device was introduced by Monolithic Memories. The PAL differs from the PROM in that the input AND gate connections are programmable in PAL, whereas the output OR gate connections are programmable in PROM. This difference usually results in reduction of gate count in PAL compared to PROM performing same function. PAL is in the form of standard two level logic circuit for evaluating an SOP logical expression. Fig. 12.8 shows logical diagram of small PAL.

Now let's see some PAL's available in the market and are famous.

As shown all chips are nothing but m × 16 × n FPLAS.

where, m = number of inputs

n = number of outputs

Each IC is packaged in 300 mil wide DIP package. Their totem pole output provide ease of use and fast access time.

In PAL numbers are,

| | | | |
|---|---|---|---|
| PAL | 16 | H | 2 |
| PAL | 16 | L | 2 |
| PAL | 16 | C | 1 |
| PAL | 14 | H | 4 |
| PAL | 14 | L | 4 |
| | ↑ | ↓ | ↓ |
| | Inputs | Active Output Level | Output |

Let's see inside of PAL, Fig. 12.10 shows one exmaple PAL structure.

Let's analyse the connection separately. Refer Fig. 12.11.

Scanned by CamScanner

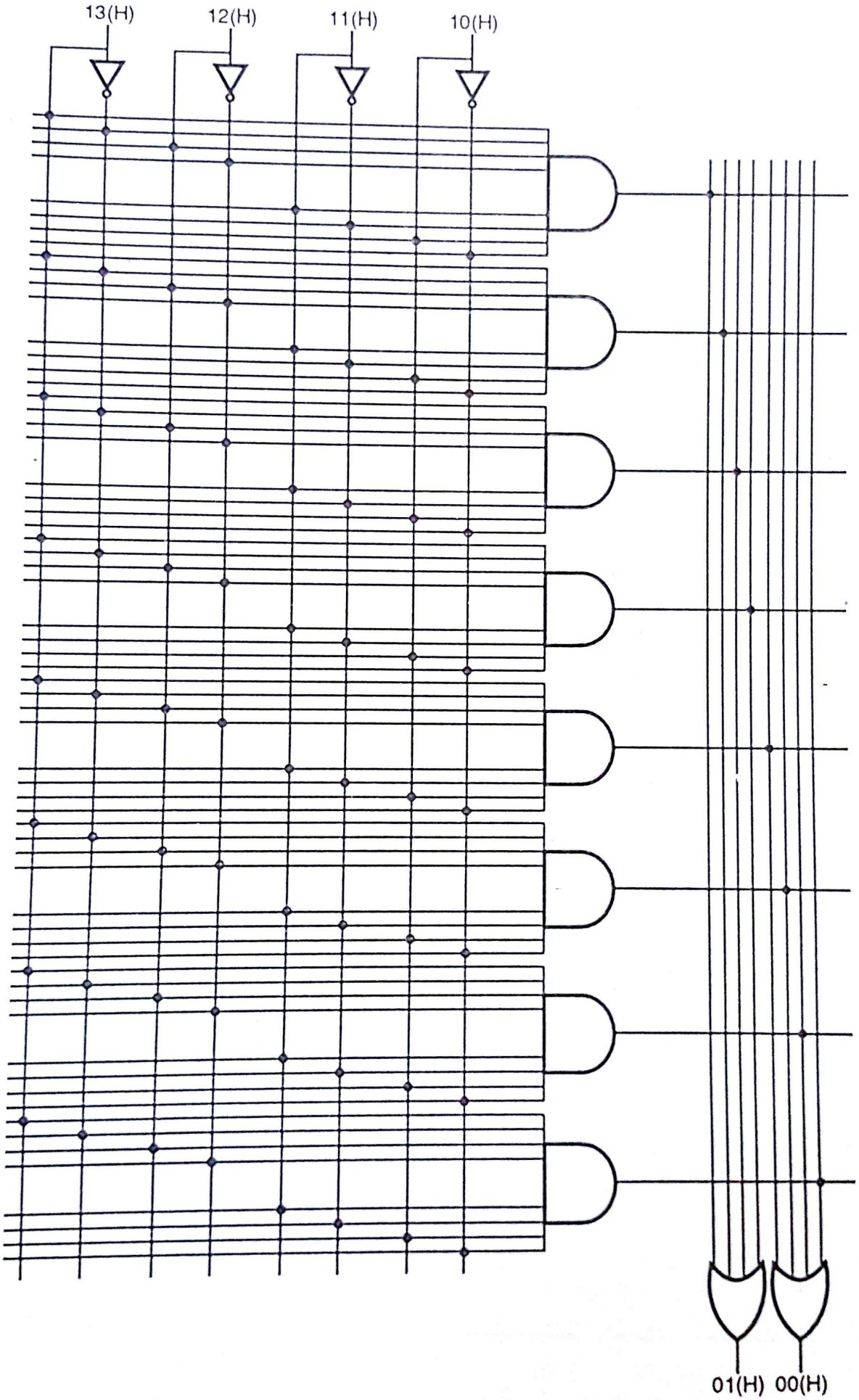



**Fig. 12.8 : Example of PAL**

Scanned by CamScanner

Scanned by CamScanner



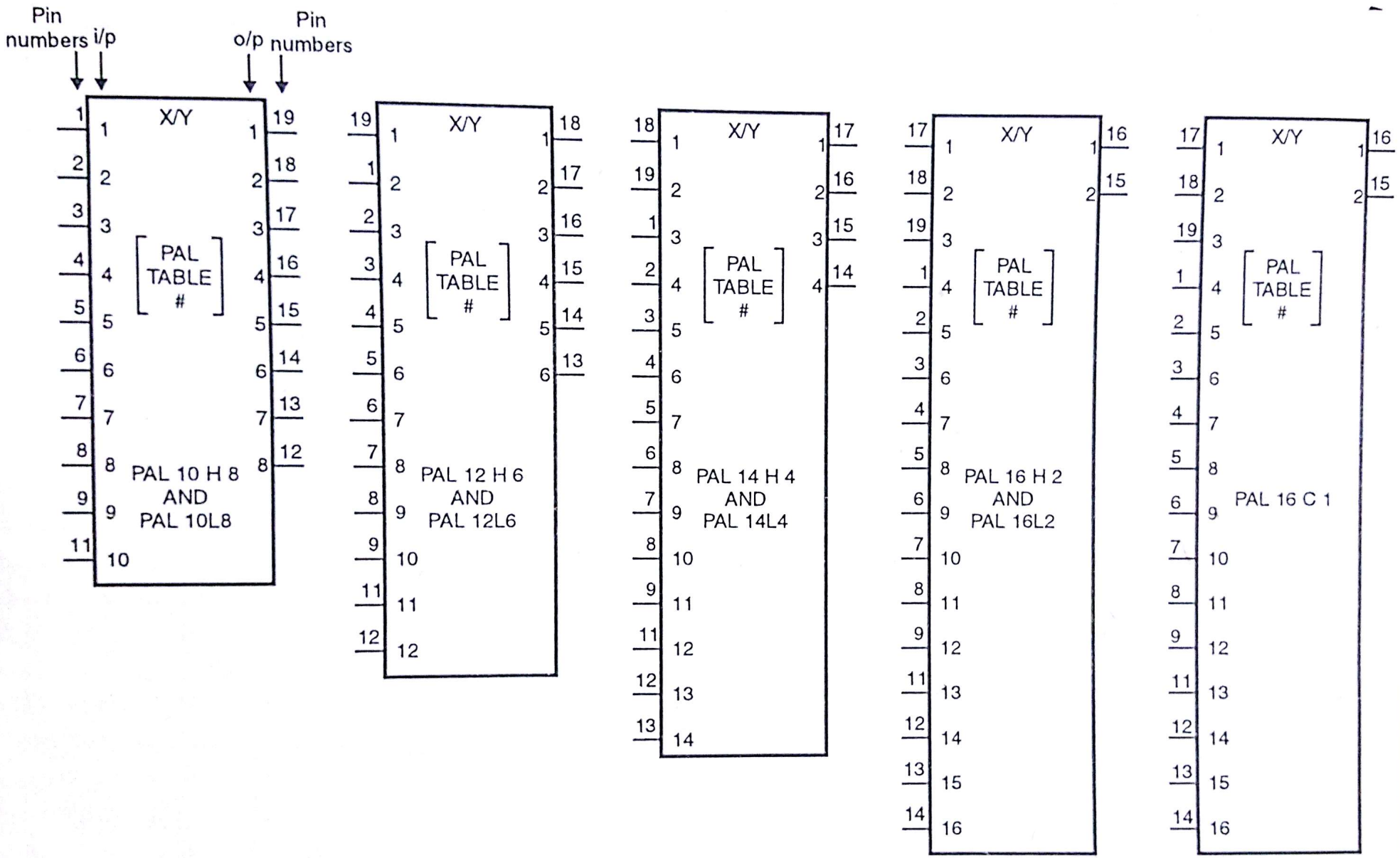



**Fig. 12.9 : PAL from monolithic memories**

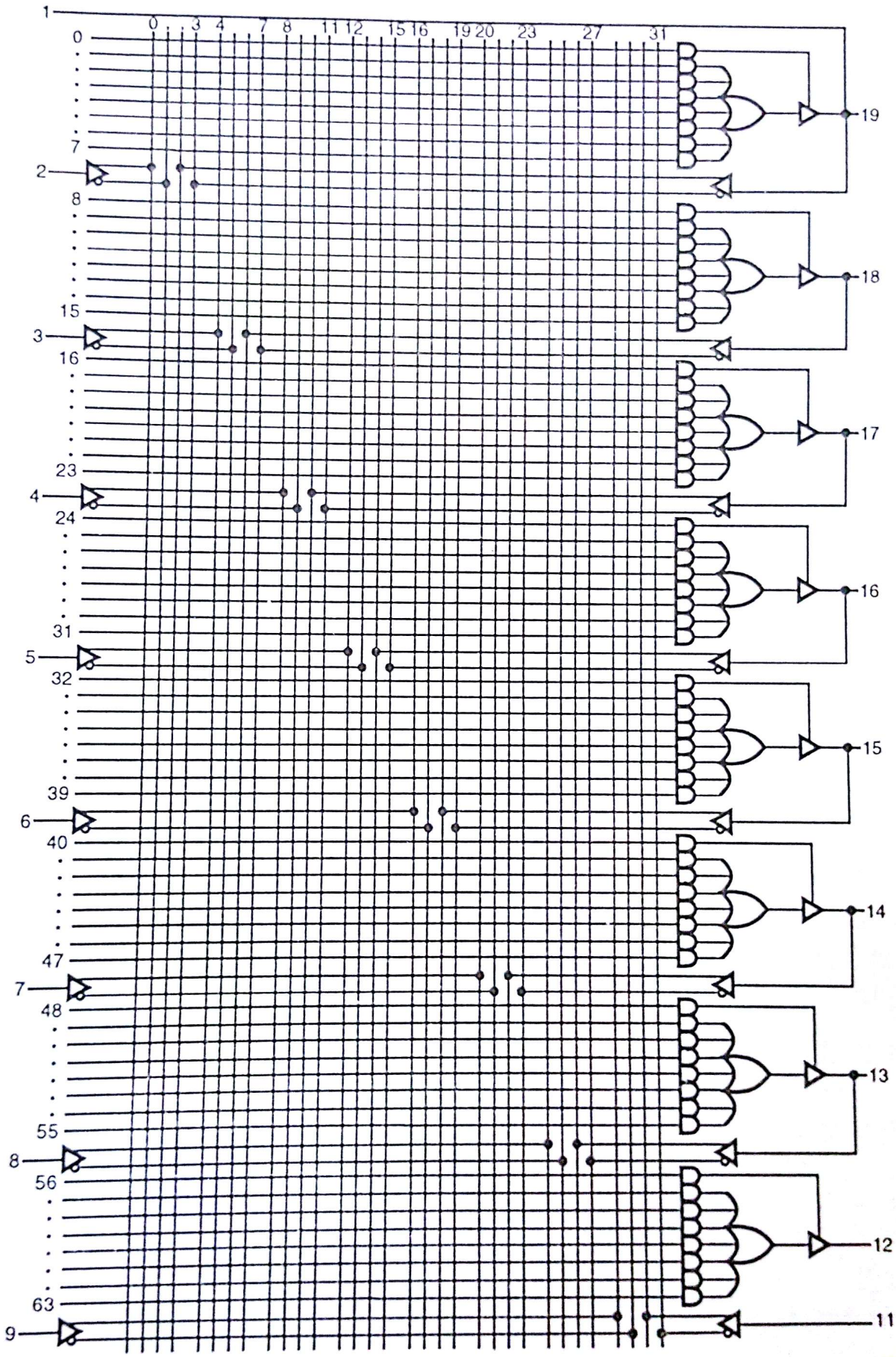



**Fig. 12.10 : Example of PAL**

Scanned by CamScanner

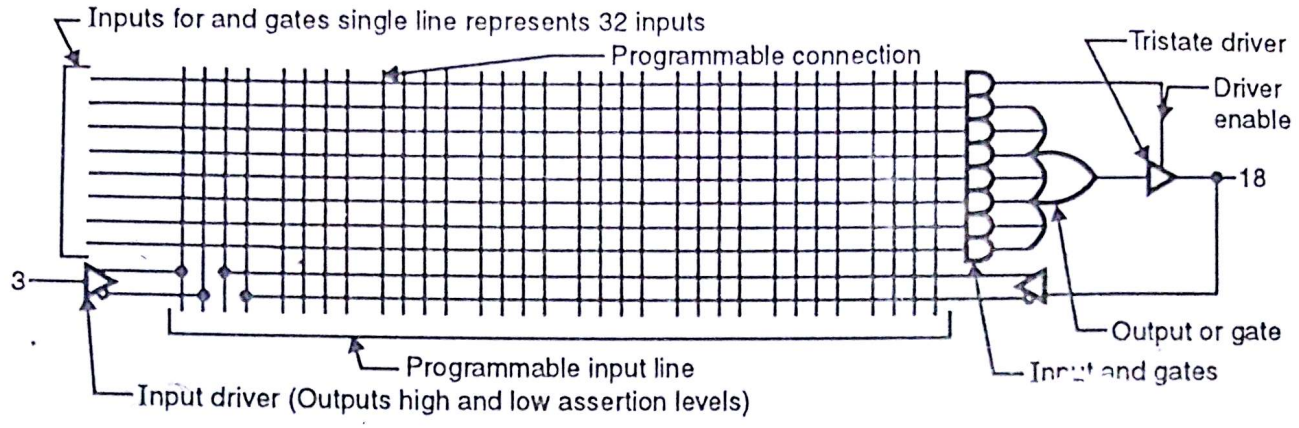



**Fig. 12.11 : Part of PAL Circuit**

Fig. 12.11 shows all the parts labelled.

(1) It has two level logic circuit.

(2) Firstly seven 32 input AND gates.

(3) Secondly 7 input OR gate.

(4) Programmable connections are between 32 vertical input lines and horizontal input lines to AND gates.

(5) Each horizontal input line to AND gate shows 32 actual inputs to the gate.

(6) Seven AND gates are permanently connected to an OR gate.

(7) External connection to circuit are indicated by pin numbers on the circuit diagram. The circuit shows only of the input and its driver.

(8) Driver is capable enough to provide high as well as low assertion level.

(9) O/p i.e. Pin 18 can serve as an i/p and also as an output, because of tristated o/p buffer.

(10) Enable of tristate driver is controlled by top AND gate.

(11) If top AND gate enables driver pin 18 will be an output line. But if AND gate disables driver pin 18 can function as an input line.

(12) If you observe diagram it is possible to have internal connection of an output as an input.

(13) Refer Fig. 12.10.

Pin number 1 to 9 and 11 are strictly input lines (10 inputs).

Pin number 12 and 19 are strictly output lines (2 outputs).

Pin number from 13 to 18 can be programmed as an input or an output (6 programmable).

∴ This PAL is

$$\text{PAL}\ \underset{\text{Inputs}}{\underset{\uparrow}{\overline{16}}}\ \text{L}\ \underset{\text{Output}}{\underset{\downarrow}{\overline{8}}}$$

## 12.3.1 Pal Programming :

The way we have truth table for FPLA, we have truth table for PAL. Fig. 12.12 shows one example of truth table for PAL 10H8 and PAL 10L8.

Scanned by CamScanner

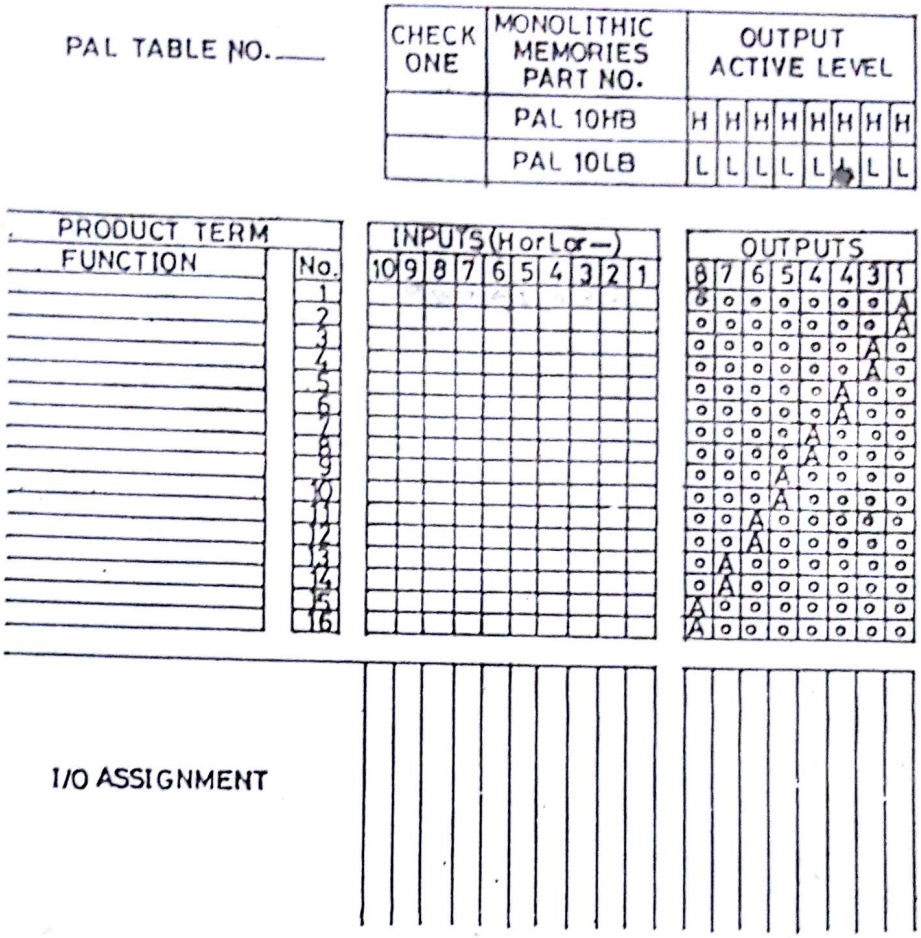



**Fig. 12.12 : Truth Table for pal 10L8 and PAL 10H8**

As seen, outputs and output levels have already been programmed. We have to simply define 16 product term. Normally, PALs are programmed with PAL programmer. One can have two types of PAL.

(1) In some PALs all programmable connections are connected with fusible links, and PAL programmer burns away the connections not desired. Here programming is permanent.

(2) Some PAL also supports erasable programming, so PAL can be reused.

For programming PAL we have to simply write boolean functions. The boolean expressions then converted to ***fusible link map*** by software like PALASM or PAL 20. EXE. We have to write boolean expression in any editor (NE, SK, TC etc.). To understand let's take one example of 3 line to 8 line decoder (74LS138) using PAL 16L8.

---

**Ex. 1 :** Design 3 to 8 line decoder using PAL 16L8.

**Soln. :** Let's say inputs are A, B and C, where A = MSB and C is LSB. Output lines are $Y_0$, $Y_1$, $Y_2$, $Y_3$, $Y_4$, $Y_5$, $Y_6$ $Y_7$.

Truth table is,

Scanned by CamScanner

| Inputs | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ | $Y_7$ |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

```
; PALASM                Design Description
  TITLE                 3 line to 8 line decoder
  PATTERN               3–8 decoder
  Revision              A
  AUTHOR                PALAN N. G.
  COMPANY               CUMMINS COLLEGE
  DATE                  7/7/2002
  CHIP                  PAL16L8
```

| ; Pins | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | NC | NC | NC | NC | NC | NC | GND |
| ; Pins | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| | NC | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ | $Y_7$ | $V_{CC}$ |

**Equations :**

```
/Y0  =  / A * / B * /C
;
/Y1  =  / A * / B * C
;
/ Y2  =  / A * B * / C
;
/Y3  =  / A * B * C
;
/Y4  =  A * /B * /C
;
/Y5  =  A * /B * C
;
/Y6  =  A * B * /C
;
/Y7  =  A * B * C
```

Scanned by CamScanner

As seen declaration section contains information about document design, device identification, pin assignment and string substitution. Anything after semicolon is comment. Initial lines are called ***Design Header***, because each line starts from a keyword.

After this, one has to assign signal names to the pins of the device. Something must be listed for each pin. If not used specify NC. Polarity i.e. active high or active low can be done by '/'. If '/' is present means active low, not present active high. We have selected all outputs active low.

Now when this file is given to EXE. program it produces or generates ***Jedec*** file. Thus file is normally given to PAL programmer. The jedec format is as shown in Fig. 12.13.

```
PAL16L8

CUMMINS COLLEGE

*D2217*
G0*F0*
L0000 11111111111111111111111111111111*
L0032 01010111111111111111111111111111*
L0256 11111111111111111111111111111111*
L0288 01011011111111111111111111111111*
L0512 11111111111111111111111111111111*
L0544 10010111111111111111111111111111*
L0768 11111111111111111111111111111111*
L0800 10011011111111111111111111111111*
L1024 11111111111111111111111111111111*
L1056 01100111111111111111111111111111*
L1280 11111111111111111111111111111111*
L1312 01101011111111111111111111111111*
L1536 11111111111111111111111111111111*
L1568 10100111111111111111111111111111*
L1792 11111111111111111111111111111111*
L1824 10101011111111111111111111111111*
C3EC4*
•A8F5
• • • • • • • • •
```

**Fig. 12.13 : Jedec format**

As seen all programmable connections numbered starting from 0000. Closed connections are shown as 1's and open connections as 0's.

Lastly I would like to add that in PAL they also provide security fuse. If you blow the fuse one cannot copy the PAL contents. This is very useful from secrecy point of view; because of this property PAL IC is most widely used in ***Hardware Lock***, so that nobody can copy licenced software copy. Software is only executable if and only if hardware lock is present.

□□□

DOLOND COLLEGE OF COMMERCE LIBRARY BSc IT 743

Scanned by CamScanner

# Logic Families

13

**Syllabus :**

Digital Circuits logic levels, Propagation delay times, power dissipation, Fan-out and Fan-in, Noise margin for popular logic families, TTL, LSTTL, CMOS and ECL integrated circuits and their performance comparison, open collector and Tri-state gates and buffers.

| Name of the Topic | Section number | Theory | Problems |
|---|---|---|---|
| • Digital IC Technology. | 13.2 | ✓ | – |
| • Terms related to digital ICs. | 13.3 | ✓ | – |
| • Bipolar families. | 13.4 | ✓ | – |
| • Transistor - Transistor logic (TTL). | 13.5 | ✓ | – |
| • TTL families. | 13.6 | ✓ | – |
| • Schottky TTL. | 13.7 | ✓ | – |
| • Comparison of TTL families. | 13.8 | ✓ | ✓ |
| • Handling TTL ICs. | 13.9 | ✓ | – |
| • Open collector. | 13.11 | ✓ | – |
| • Wire ANDing. | 13.12 | ✓ | – |
| • Tri-state TTL devices. | 13.13 | ✓ | – |
| • ECL. | 13.14 | ✓ | – |
| • MOS logic (CMOS). | 13.15-13.22 | ✓ | – |
| • Comparing TTL and CMOS. | 13.23-13.24 | ✓ | – |
| • Interfacing | 13.25 | ✓ | – |
| • BICMOS logic. | 13.26 | ✓ | – |



**Comment**

- Fantastic chapter from the print of view of knowledge and concept.
- Concentrate more on Sec. 13.3 to Sec. 13.5 and Sec. 13.15, Sec. 13.22, Sec. 13.26

Scanned by CamScanner

## 13.1 Introduction :

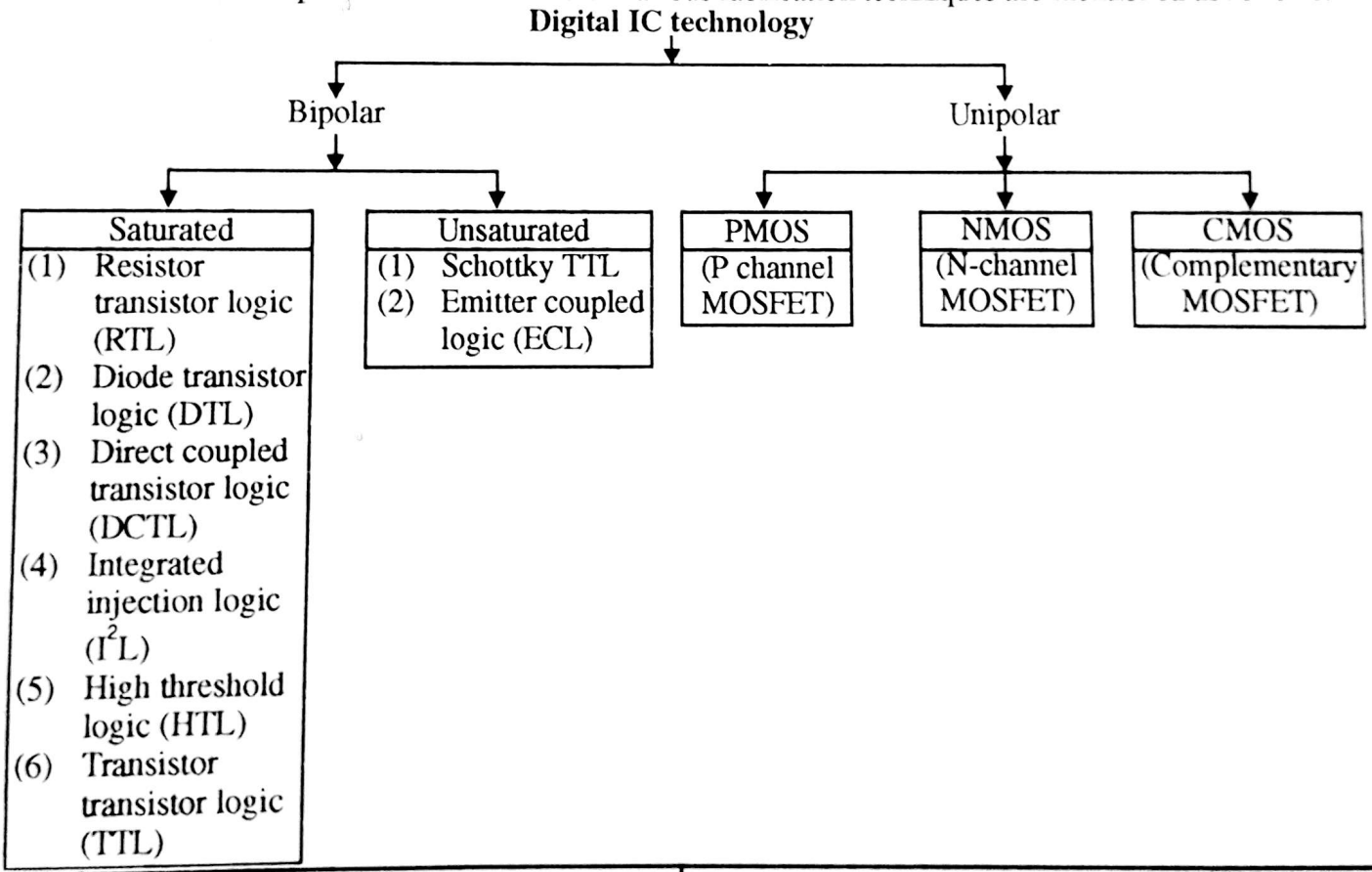
We have started with basic binary system and discussed complex digital ICs. Throughout our discussion we talked about logic '0' and '1'. We treated logic '0' = GND, logic '1' = $V_{CC}$ (+5V). But practically logic levels are same ? or it varies from family to family. ***What will be maximum current given by gate? What is effect of noise, what is speed limitation for each family? How much power dissipated per gate?*** These all questions will be answered in this chapter. From practical implementation point of view, this chapter is very important.

## 13.2 Digital IC Technology :

One can built AND – OR – NAND – NOR using transistor. Driving transistor in saturation and to cutoff, is obviously going to take some time in nano sec. Thus depending upon speed, power consumption, transistor is operated in saturation or unsaturation. Because of this, we have different varieties of logic families.

Transistor is bipolar device, one can think of unipolar device MOSFET also. This unipolar device also provides different families. The chart for the family is given as follows.

Logic families are fabricated on silicon chip. There are different technologies for fabrication of Integrated circuit. Digital IC's are classified according to complexity of circuit i.e. relative no. of individual basic gates it would require to build the circuit. Various fabrication techniques are mentioned as follows.




| Fabrication Technique | Individual basic logic gates required |
|---|---|
| Small Scale Integration (SSI) | < 12 |
| Medium Scale Integration (MSI) | 12 or more, but < 100 |
| Large Scale Integration (LSI) | > 100 and < 1000 |
| Very Large Scale Integration (VLSI) | 1000 or more. |

**Table 1**

Scanned by CamScanner

## 13.3 Terms Related to Digital IC :

### 13.3.1 General Definition :

(a) **Current (I) :** Current is the flow of electric charge from one potential to another through a conductor. The unit of measure is ampere or Amp. abbreviated A. Common unit found in semiconductor industry is μ A. Negative current is defined as current flowing out of device terminal. Positive current is defined as current flowing into device terminal.

(b) **Voltage (V) :** Voltage or electromotive force which causes current to flow through a conductor. Unit of measure is 'volt', abbreviated 'V'.

### 13.3.2 Voltage Parameter :

**Output Voltage Parameter**

Ideally, 'LOW' output is 0 V and 'HIGH' output is +5V. But due to internal circuit drop, output will not be 5V or 0V. When 'LOW' output is expected instead of 0V, it will have some value (typically 0.2 to 0.3 V). When 'HIGH' output is expected instead of +5V it will be less than +5V (typically 3.6 to 4V). But for this we have defined some limit i.e. at HIGH output, minimum voltage and at 'LOW' output, maximum voltage, $V_{OH}$ and $V_{OL}$ respectively.

(i) **$V_{OH}$ (HIGH level output voltage OR Worst case HIGH output) :**
The minimum voltage which will be available at an output terminal of gate, when required output should be at HIGH level logic. Normally it is above 2.4 V.

(ii) **$V_{OL}$ (LOW level output voltage OR Worst case LOW output) :**
The maximum voltage which will be available at output terminal of gate, when required output should be at LOW level. Normally it is below 0.4 V.

Let's draw the graphical representation for $V_{OH}$ and $V_{OL}$.

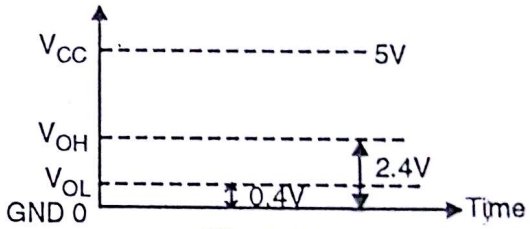



Fig. 13.1

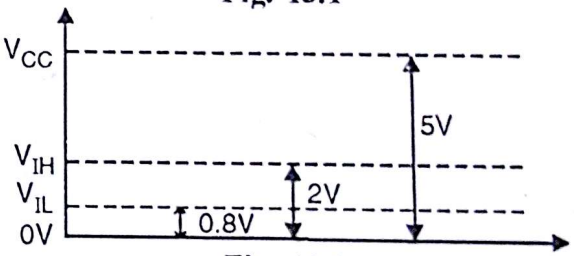



Fig. 13.2

**Input Voltage Parameter**

The way we have defined minimum output voltage treated as HIGH and maximum output voltage treated as LOW, same way for input side there will be minimum input voltage which will be treated as HIGH and maximum input voltage which will be treated as LOW, i.e. $V_{IH}$ and $V_{IL}$ respectively.

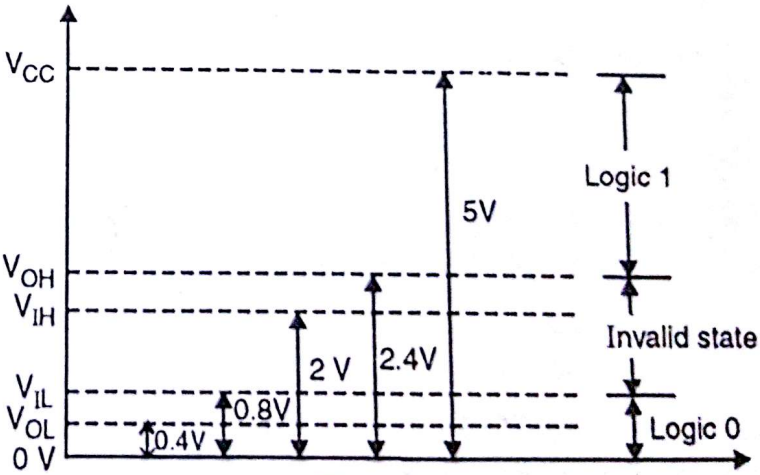



Fig. 13.3

(i) **$V_{IH}$ (HIGH level input voltage OR Worst case HIGH input voltage) :**
The minimum positive input voltage level that can be applied to an input terminal of device and be recognized as logic HIGH level. Normally $V_{IH(minimum)} = 2V$.

Scanned by CamScanner

(ii) **$V_{IL}$ (LOW level input voltage OR Worst case LOW input voltage) :**

The maximum positive input voltage level that can be applied to the input terminal of a device and be recognised as logic LOW level. Normally $V_{IL\ (max)} = 0.8$ V.

Let's combine Figs. 13.1 and Fig. 13.2

(iii) **$V_R$ (Input Reverse Voltage) :** The voltage applied to an input of a device that causes the input structure to become reverse biased. Usually equal to minimum high level output voltage ($V_{OH}$).

(iv) **$V_F$ (Input Forward Voltage) :** The voltage applied to input of a device that causes the input structure to become forward biased; usually equals to maximum output low level voltage ($V_{OL}$).

(v) **$B_{Vin}$ (Input Breakdown Voltage) :** The maximum voltage that the device is guaranteed to be able to withstand without exceeding more, input current.

### 13.3.3 Current Parameters :

On the similar line of voltage, we will define current also i.e. $I_{OH}$, $I_{OL}$, $I_{IH}$ and $I_{IL}$.

**Output Current Specification**

(i) **$I_{OH}$ (HIGH level output current) :** The current flowing out of an output terminal, when output is supposed to be at HIGH level logic.

(ii) **$I_{OL}$ (LOW level output current) :** The current flowing into an output terminal, when output is supposed to at LOW level logic.

(iii) **$I_{OS}$ (Output Short circuit current) :** The current out of an output terminal, when output is shorted to ground, the output is supposed to be at HIGH level logic.

**Input Current Specification**

(i) **$I_J$ (Maximum HIGH input current) :** Current flowing into an input when maximum voltage applied to it.

(ii) **$I_{IH}$ (HIGH level input current) :** The current flowing into an input when input has high level logic voltage equal to minimum high level output voltage ($V_{OH}$). Typically = 40 μ A (1 TTL load).

(iii) **$I_{IL}$ (LOW level input current) :** The current flowing out of an input terminal,. when a low level voltage is equal to the max. low level output voltage ($V_{OL}$), applied to it. Typically = – 1.6 μ A (one TTL load).

**Supply Current Specification**

(i) **ICCH – Supply current (outputs in the HIGH state) :** The current flowing into the VCC terminal of a device when logic HIGH level established at the output.

(ii) **ICCL – Supply Current (outputs in LOW state) :** The current flowing into the VCC terminal of a device, when logic LOW level established at the output.

### 13.3.4 Noise :

Noise is unwanted random signal. The normal conductor or tracks of PCB acts as antenna and catches noise. If noise gets introduced in the signal, it may add or subtract with the signal.

Let's see how it is going to create problem by providing false triggering. Consider a circuit, in which NAND 1 drives NAND 2 (Refer Fig. 13.4).

Scanned by CamScanner

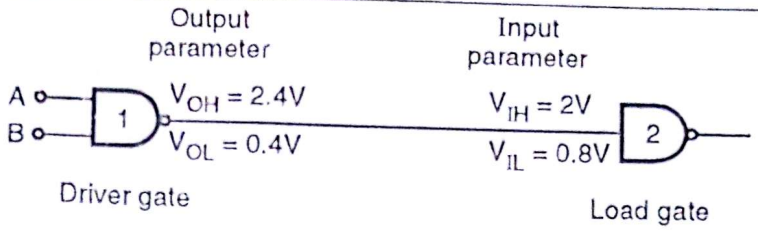



**Fig. 13.4**

Let's consider noise gets introduced in the line of 500 mV, when output from driver gate is at low level, with the polarity shown in Fig. 13.5.

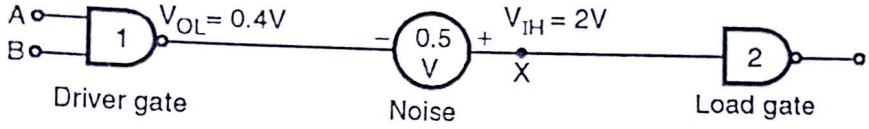



**Fig. 13.5 : Pulse triggering into high state**

Noise gets added with output $V_{OL}$, therefore voltage at point X = $V_X$ = 0.4 + 0.5V = 0.9V. As defined max. low level input ($V_{IL}$) is 0.8 V, and minimum high level input is 2 V. Therefore $V_X$ = 0.9 falls in invalid input refer Fig. 13.3. Therefore output is unpredictable, but it may give false change in output stage of load gate even though output of driver gate is at LOW state.

Same way, when $V_{OH}$ = 2.4 v noise gets introduced as shown in Fig. 13.6.

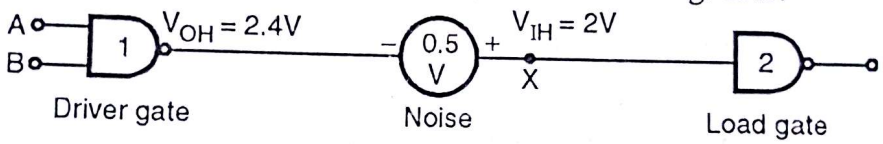
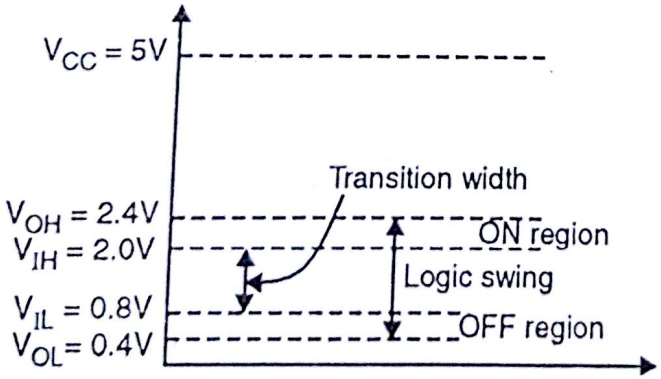



**Fig. 13.6 : False triggering into low state**

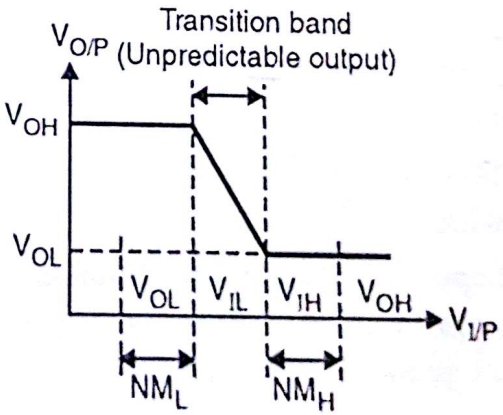
$V_X = V_{OH} - V_{noise}$ = 2.4 – 0.5 V = 1.9V. Therefore as $V_X$ is lower than minimum voltage required at input to treat it as high i.e. $V_{IH}$, load gate may change state or can provide unpredictable output. How much noise gets introduced that depends upon environment. But as far as we are concern, we want that logic family should not change output because of noise i.e. family should be immune to noise.

Noise immunity can be defined as how much insensitivity a logic gate has to triggering or reaction to noise.

To improve noise immunity what should be done ? To explain it, we will redraw Fig. 13.3.




**(a) Logic level diagram**




**(b) Input, output characteristics**

**Fig. 13.7**

Scanned by CamScanner

Let's understand first new terms.

(i) **Transition width :** Transition width is $V_{IH} - V_{IL}$ i.e. amount of voltage require to go from LOW to HIGH state or V. If the input voltage level falls between transition width output is unpredictable.

(ii) **Logic Swing :** The difference between two output voltage levels is known as logic swing of gate. Logic swing = $V_{OH} - V_{OL}$.

(iii) **Noise Margin (NM) :** The amount of extraneous signal which can be tolerated (margin of noise tolerated), before an output voltage of gate deviates from allowable logic voltage level.

**Low level noise margin ($NM_L$) :** The difference between $V_{IL} - V_{OL}$ is known as low level noise margin.

$$NM_L = V_{IL} - V_{OL} \cong 0.4V \cong 400mV.$$

It means that if output of driver gate is low logic (0.4V), and if noise voltage $\geq (V_{IL} - V_{OH})$ superimposes on signal, then it will lead you to invalid region (Refer Fig. 13.9 (a)). Difference between $V_{IL} - V_{OL}$ should be as large as possible.

**High level noise margin (NMH) :** The difference between $V_{OH} - V_{IH}$ is known as high level noise margin.

$$NM_H = V_{OH} - V_{IH} \cong 400 \text{ mV}.$$

This difference should be also more, to have high noise immunity. In short, one should have very low transition width and very high logic swing, to have high noise immunity.

### DC Noise Margin and AC Noise Margin :

DC noise margin is nothing but what we studied just now. In general, DC noise margin can be defined as difference between, output voltage limits of driver gate and voltage requirement of load gate i.e. $V_{IL} - V_{OL}$ and $V_{OH} - V_{IH}$.

### AC Noise Margin

The term dc margin seems little inappropriate when applied to noise, which by general conception is an ac factor. While dc noise is not impossible, the user is much more concerned with transient type of noise. In circuit, with speed of integrated circuit logic, microsecond wide pulses are extremely long and may be treated as dc, so far as operating margins are concerned.

As the noise pulse width shortens into low nanosecond region, a limit is reached where an input pulse can be shorter than the time required for signal to propogate through the device. Therefore higher amplitudes are required to change the output. As pulse width is more or high amplitude pulse will change the output of the gate. Because of pulse amplitude, some effects are there which can be broken into two separate reactions.

*(1) Effect of input being driven into opposite state marginally or completely. This affect propogation delay of the device.*

*(2) Capacitive storage and coupling effect, which causes high amplitude pulse to enter and be partially stored. This behaviour allows gate to continue to respond to pulse even after it has disappeared at the input.*

## 13.3.5 Fan Out :

Number of load gates connected at the output of a gate is called fanout of the gate. It is driving capability of a particular gate to drive 'N' no. of gates.

Scanned by CamScanner

As shown in Fig. 13.8, driver gate drives 'N' no. of load gates. N = fanout. If we connect more no. of gates, we are loading the gate, therefore output voltage will reduce and because of more current taken out from output terminal, gate may get damaged.

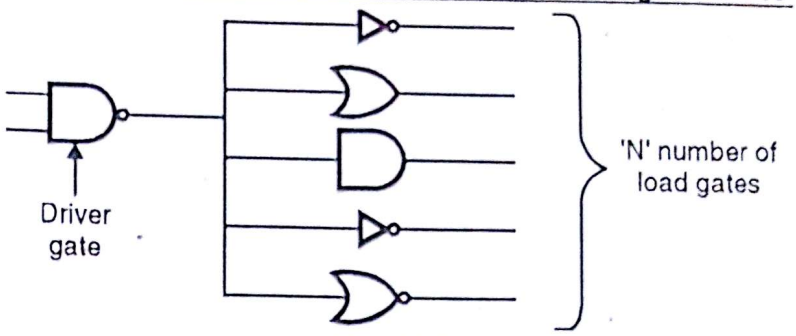



Fig. 13.8

### 13.3.6 Fan In :

Number of inputs connected to gate is called fan in of the gate.

For 2 input NAND, fan in is 2.

4 input AND, fan in is 4.

In short, no. of input terminals = fan in.

### 13.3.7 Speed of Operation or Propagation Delay :

As we have seen propagation dalay is nothing but time taken by particular gate to produce output, when input is applied.

The cause of propagation delay is :

(1) Circuit capacitance.

(2) Finite switching speed of transistor. Transistor takes some time to switch from saturation to cutoff and cutoff to saturation.

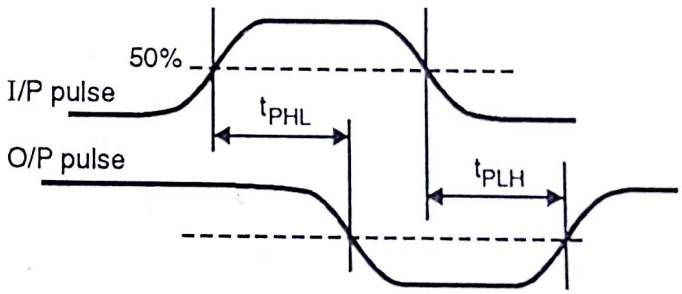



Fig. 13.9

The $t_{pd}$ is measured at 50 % voltage level of input and output wave form. Let's say a pulse is given to inverter input. Then input and output wavefroms as shown in Fig. 13.9.

$t_{PLH}$ = Propagation delay measured when output switches from low to high.

$t_{PHL}$ = Propagation delay measured when output switches from high to low.

$$\therefore t_{pd} = \text{Propagation delay} = \frac{t_{PLH} + t_{PHL}}{2}$$

### 13.3.8 Current Source and Current Sink :

Current Source and Sink are named depending upon flow of current from output terminal. Let's consider driver gate and load gate.

**Current Source**

Fig. 13.10 shows output of driver gate connected to input of load gate. Output of driver gate is 'HIGH', therefore driver acts as a source and supplies current to load gate. Source current is same as $I_{IH}$.

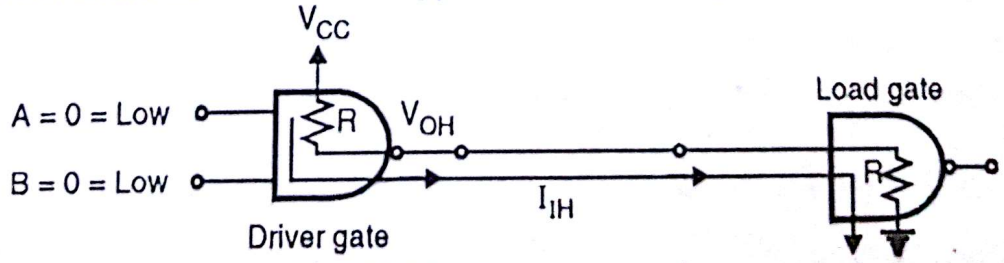



Fig. 13.10 : Current sourcing

Scanned by CamScanner

**Current Sink**

As shown in Fig. 13.11, when input to NAND (driver gate) is high, output produced is low ($V_{OL}$). Therefore current ($I_{IL}$) flows from load gate to driver gate. With respect to driver gate, current is coming in means gate ***Sinks*** the current, therefore called sinking current.

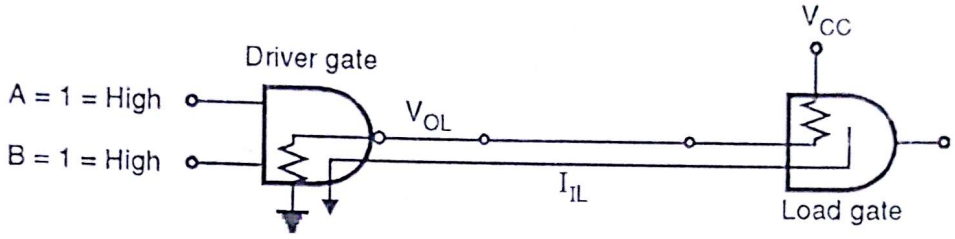



**Fig. 13.11 : Current sink**

## 13.3.9 Power Dissipation :

The important parameter of the logic family is power dissipation. Normally it should be as low as possible. The power dissipated by gate HIGH and LOW state is different. Therefore normally average is taken.

$$P_{D(avg)} = \frac{P_{DH} + P_{DL}}{2}$$

$P_{DH} \Rightarrow$ Power dissipated when input was HIGH

$P_{DL} \Rightarrow$ Power dissipated when input was LOW.

## 13.3.10 Figure of Merit or Speed Power Product or Power Delay Product :

The product of propagation time and power dissipation ($t_{pd} \times P_D$) is known as figure of merit or performance of gate. Normally minimum value of product is desired.

These is always compromise between speed ($t_{pd}$) and dissipation. To reduce dissipation, Internal resistances are increased, then propagation delay is more because stray capacitance and junction capacitance will take more time to charge and discharge, therefore speed is less. If resistance are decreased to increase speed, power dissipation increases. Therefore figure of merit is taken while selecting family.

$$PD = P_{D(avg)} \cdot t_{pd}$$

PD = Power delay product

$P_{D(avg)}$ = Average power dissipation

$t_{pd}$ = Propagation delay

## 13.3.11 Operating Temperature :

The temperature range in which IC will function properly is called operating temperature for the IC.

For consumer and industry grade IC, temp, range is $0^0$ to $70^0$C.

For military, aerospace, temperature range is $-55^0$C to $+125^0$ C.

## 13.3.12 Package Density :

Package density means no. of gates fabricated per unit square area. Some of the families have higher package density, because of ***merging of components***.

Scanned by CamScanner

Before we start studying various logic families it is important to understand all the parameters. Because each family is compared taking above permeters into consideration.

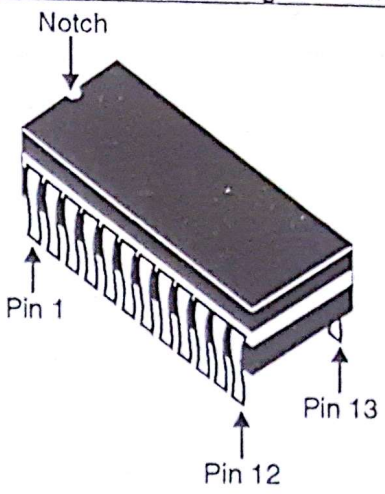
### 13.3.13 Physical Packaging of Logic Circuits :

Fig. 13.12 shows a common physical package used to house IC logic circuits. This package is made of either plastic or ceramic and is called dual–in–line (DIP) package. Pins of the IC may be 14, 16, 18, 20, 40 and so on.

Fig. 13.12 shows physical dimension 24 pin Chip.

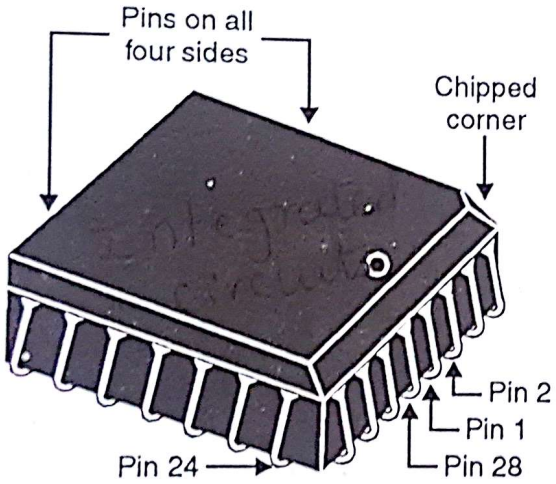
Presence of the notch on one end, is used to locate pin no. 1. Some DIPs use small dot on the top surface of the package to locate pin 1. The spacing between pins (lead pitch) is typically 100 mils.

Due to advancement in digital integrated circuit, we have few more IC packages.



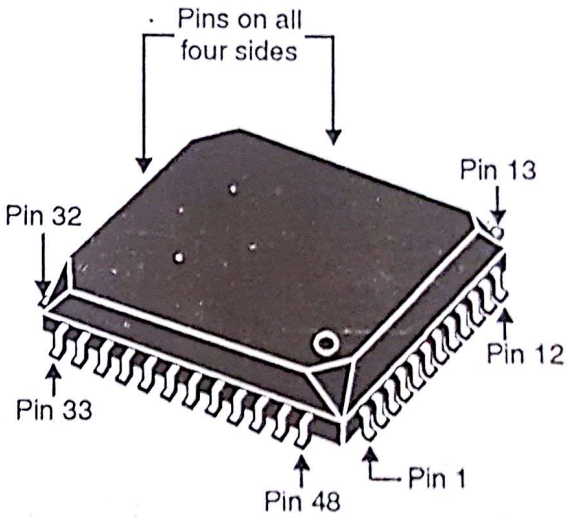

24-pin DIP

**Fig. 13.12**




28-pin PLCC
(J-lead)
socket or surface-mount

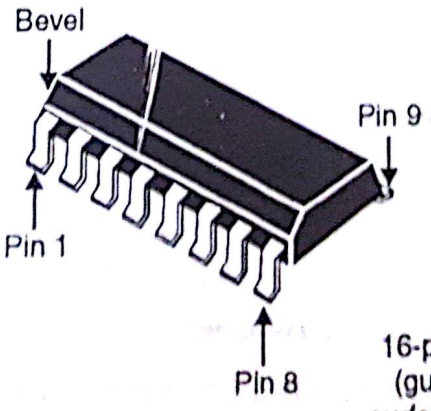
(a)




48-pin QFP
(gull-wing)
surface-mount

(b)

**Fig. 13.13 : Common IC packages**




16-pin SOIC
(gull-wing)
surface-mount

**Fig. 13.14 : Common IC packages**

Scanned by CamScanner

| Acronym | Package Name | Height | Lead Pitch |
|---|---|---|---|
| DIP | Dual-In-line Package | 200 mils (5.1 mm) | 100 mils (2.54 mm) |
| SOIC | Small Outline Integrated Circuit | 2.65 mm | 50 mils (1.27 mm) |
| SSOP | Shrink Small Outline Package | 2.0 mm | 0.65 mm |
| TSSOP | Thin Shrink Small Outline Package | 1.1 mm | 0.65 mm |
| TVSOP | Thin Very Small Outline Package | 1.2 mm | 0.4 mm |
| PLCC | Plastic Leaded Chip Carrier | 4.5 mm | 1.27 mm |
| QFP | Quad Flat Pack | 4.5 mm | 0.635 mm |
| TQFP | Thin Quad Flat Pack | 1.6 mm | 0.5 mm |

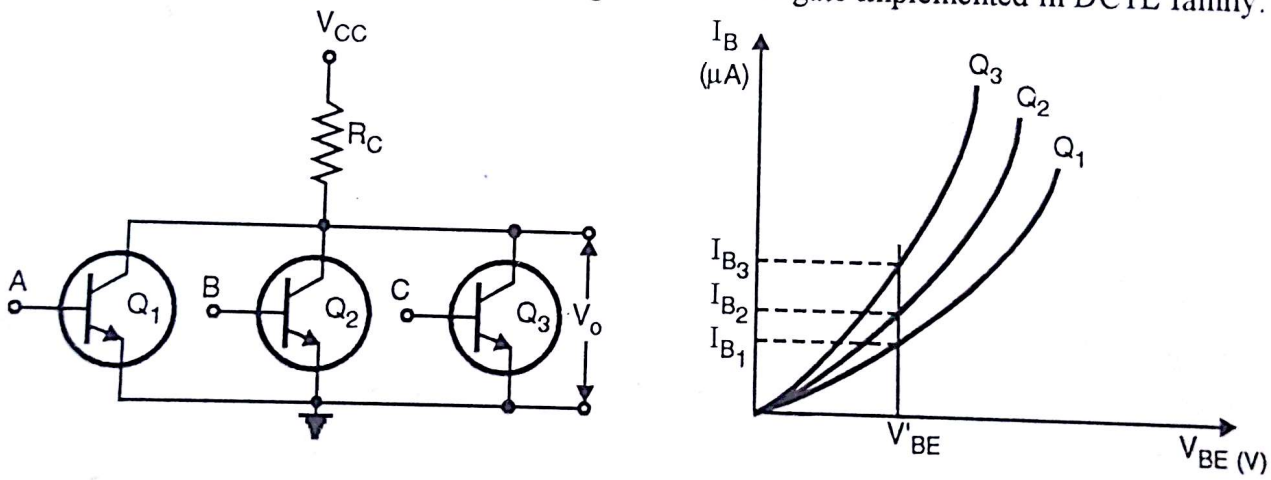
**Table 2 : IC packages**

## 13.4 Bipolar Families :

We are going to start with bipolar families.

### 13.4.1 Direct Coupled Transistor Logic (DCTL) :

DTCL is the most simplest family. Refer Fig. 13.15. NOR gate implemented in DCTL family.




**(a) NOT gate in DCTL family** **(b) Input characteristic of transistor**

**Fig. 13.15**

(1) Transistor $Q_1$, $Q_2$ and $Q_3$ are used, connected parallely.

(2) INPUT A, B and C are directly given (No base R used).

(3) Load resistance $R_C$ is common.

(4) OUTPUT $V_o$ taken from collector of transistor with respect to ground.

**Working :**

(1) When A = B = C = 0, $Q_1 = Q_2 = Q_3$ = OFF, $\therefore V_o \cong V_{CC}$ (HIGH logic).

(2) When either of the A, B, and C is 1, transistor $Q_1$ or $Q_2$, or, $Q_3$ will conduct and will operate in saturation. $\therefore V_o = V_{CE\,(SAT)} \cong 0.2V$ (LOW logic).

(3) When A = B = C = 1, because of matched characteristics, same base current will allow same collector current to flow. $V_o = V_{CE\,(SAT)} \cong 0.2V$.

(4) The above operation is of NOR gate.

(5) Thus $V_{OH} = V_{CC} = 3.6V$ and $V_{OL} = 0.2V$.

Scanned by CamScanner

**Problem :**

Ideally input characteristics of all three transistors should be identical. But it is not possible. It won't match exactly. Refer Fig. 13.15 (b), as shown, at same $V_{BE}$, base current of each transistor will be different . $I_{B1} < I_{B2} < I_{B3}$. Therefore transistors $Q_3$ takes more current than $Q_1$ and $Q_2$. $Q_3$ ***hogs*** much of the current. Transistor $Q_1$, $Q_2$ will not get much base current thus overall circuit performance may affect. This phenomena is called ***current hogging***.

**Advantages :**

(1) Simplest

(2) Cheap.

**Disadvantages :**

(1) Poor noise margin

(2) Low logic swing, typical 0.7V.

## 13.4.2 Resistor Transistor Logic (RTL) :

RTL was popular form of logic. As name indicates, we use resistors to adjust base current of transistor and will remove current hogging phenomena of DCTL. The circuit for simple RTL NOR gate is shown in Fig. 13.16.

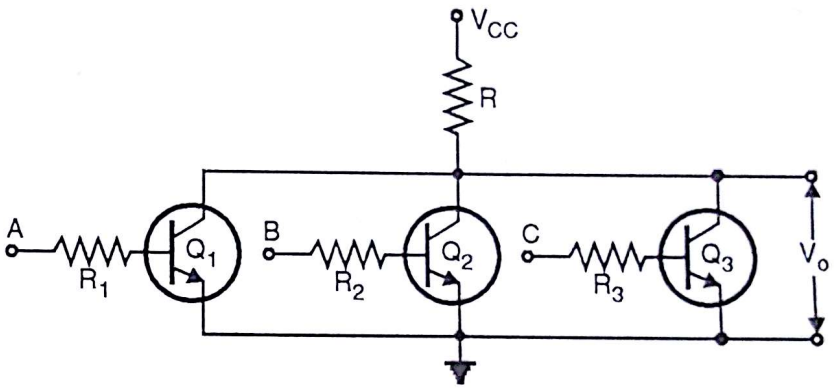



**Fig. 13.16 : RTL NOR gate**

(1) Transistor $Q_1$, $Q_2$ and $Q_3$ are used in parallel configuration.

(2) A, B and C are inputs of transistor $Q_1$, $Q_2$ and $Q_3$ i.e. inputs of gate.

(3) Load resistance $R_C$ is common.

(4) Output taken from collector of transistors.

(5) Inputs A, B and C are through base resistor $R_1$, $R_2$ and $R_3$.

**Working :**

(1) When A = B = C = 0, $Q_1$, $Q_2$ and $Q_3$ transistors are OFF. $\therefore$ $V_O = V_{CC}$ = logic 1.

(2) When either of the input A, B or C or all inputs are high, one of the transistor or all transistor gets saturated and $V_O \cong V_{CE\ (Sat)} \cong 0.2V$.

(3) Thus $V_{OH} = 3.6$ V and $V_{OL} = 0.2$ V.

(4) Supply voltage typically 3.6 V.

**Advantages :**

(1) Cheap

(2) Simplest.

**Disadvantages :**

(1) Poor noise margin.

(2) Poor fanout capability (typically 5).

(3) Low speed (Relatively fast when $t_{pd}$ = 12 nsec. compared with other saturated logic families).

(4) High power dissipation 16 mw.

(5) Small logic swing. 1V.

Scanned by CamScanner

### 13.4.3 Resistor Capacitor Transistor Logic (RCTL) :

In RTL, switching speed can be improved by adding capacitor in parallel with base resistance as shown in Fig. 13.17. This variation is called RCTL.

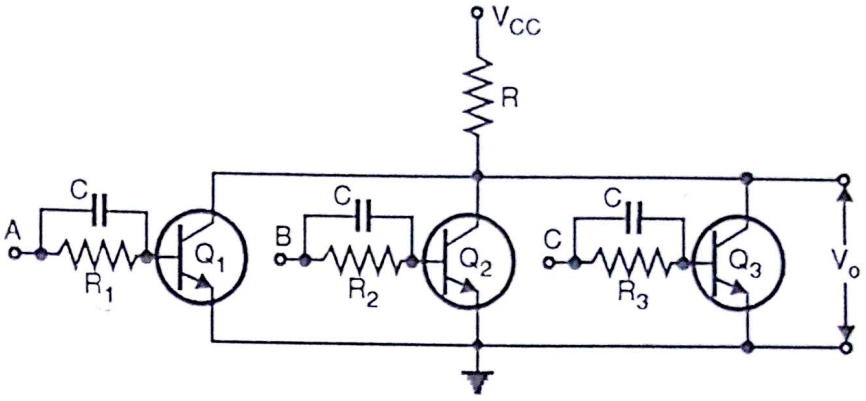



Fig. 13.17 : RCTL NOR gate

(1) Working is same as RTL and DCTL.

(2) When input transition is from 0 → 1 (rising edge), capacitor will by pass resistance so that transistor input capacitance charges more quickly.

(3) When input transition is from 1 → 0 (falling edge), capacitor will by pass resistance. So internal capacitor discharges faster.

(4) The use of capacitor allows higher value of resistors. Therefore power dissipation can be reduced.

**Problem :**

From the fabrication point of view, the circuit is not ideal, because it includes high proportions of resistor and capacitors. Capacitors and high value resistors are relatively expensive in monolithic IC form because of large area requirement.

### 13.4.4 Diode Transistor Logic (DTL) :

DTL was developed to increase logic swing, so that noise immunity can be increased. The following circuit shows simple AND gate using diode.

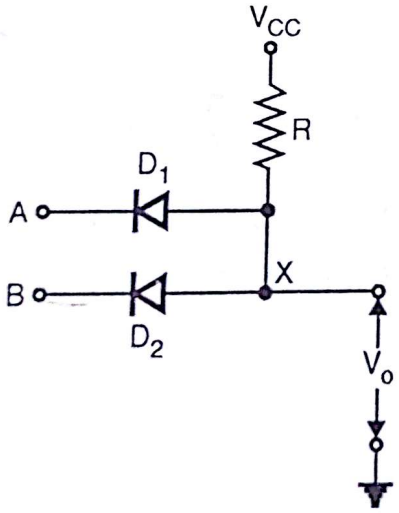



Fig. 13.18 (a) : AND gate

**AND gate Truth Table**

| A | B | $V_o$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(b) Truth table

**Working :**

(1) When A = B = 1, diodes $D_1$ and $D_2$ are reverse biased, therefore acts as open switch,
$\therefore V_o = V_{CC}$ = Logic 1.

(2) When A = B = 0 or either of the inputs A or B is '0', diodes $D_1$ and $D_2$ will get forward biased.
$\therefore$ Potential at X point is $V_X = V_F \cong 0.7V \therefore V_o = V_X \cong 0.7V$ = logic 0'.

$\therefore$ Circuit is simple AND gate.

Scanned by CamScanner

Let's see NAND in DTL logic.

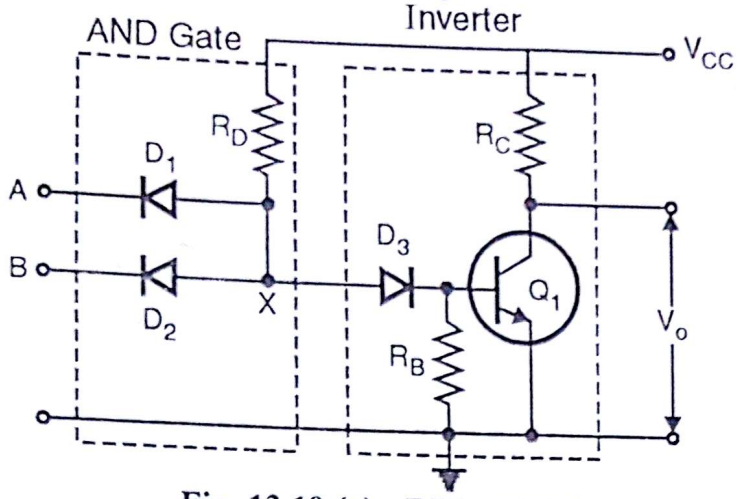



Fig. 13.19 (a) : DTL NAND

AND gate Truth Table

| A | B | $V_o$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(b) Truth table

## Working :

(1) A = B = 0, therefore $D_1$ and $D_2$ will get forward biased, therefore potential at X is 0.7V i.e. $V_X = V_F \cong 0.7$ V. To drive transistor $Q_1$ in conduction we require 1.4 V (diode $D_3$ drop + Base to emitter junction drop of $Q_1$), therefore $Q_1$ is cutoff and Output $V_o \cong V_{CC}$ = logic 1.

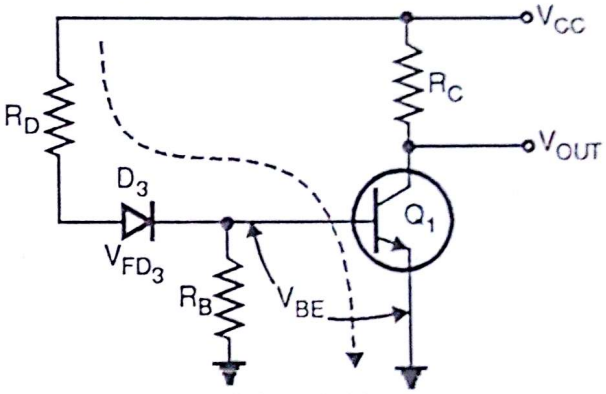



Fig. 13.20

(2) When either A or B is equal to 0, either $D_1$ or $D_2$ will conduct. $\therefore V_X = V_F \cong 0.7V$. Therefore transistor is cutoff giving $V_o \cong V_{CC}$ = logic 1.

(3) When A = B = 1, diodes $D_1$ and $D_2$ are reverse biased. Base current to transistor $Q_1$ will flow through $V_{CC}$, $R_D$, $D_3$ and base of transistor $Q_1$. Transistor $Q_1$ is in saturation and provides $V_O \cong V_{CE\,(sat)} \cong 0.2V$. when transistor conducts circuit can be shown as follows :

(4) **Use of diode $D_3$.**

Consider diode $D_3$ is not present. Refer point 1 and 2, when $D_1$, $D_2$ both or either of the diode conducts, $V_X = 0.7$. Therefore this voltage may be sufficient to drive transistor $Q_1$ and circuit will malfunction. Therefore $D_3$ is used. Now we will require 1.4V to drive $Q_1$ in saturation.

(5) **Use of $R_B$.**

Normally there is capacitance at base and emitter junction, it stores the charge. Assume that initially transistor $Q_1$ is in saturation. When A = B = 0 input is given, $V_X = 0.7$ therefore transistor $Q_1$ will try to come out of saturation. It is obvious that internal capacitor would like to discharge stored charge. If $R_B$ is not present, then capacitor won't get path to

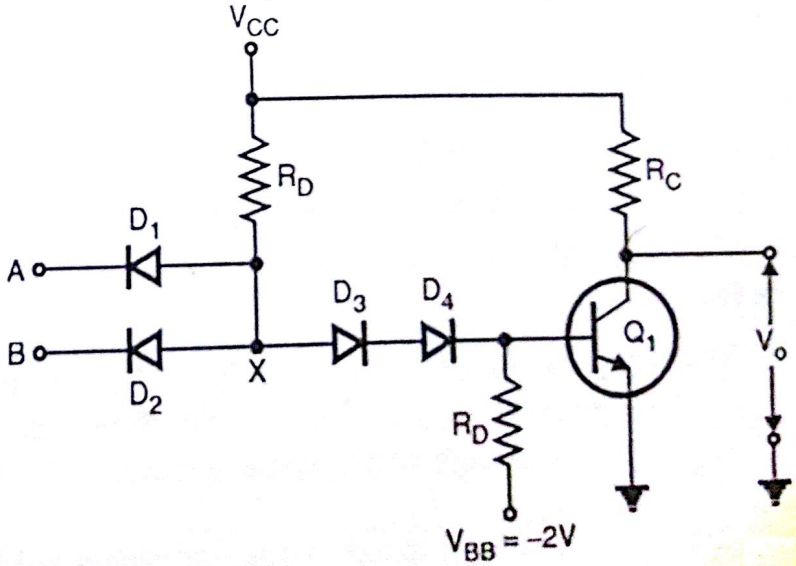



Fig. 13.21

Scanned by CamScanner

discharge because diode $D_3$ is reverse biased. Therefore it will take more time to come out of saturation. Here $R_B$ provides leakage path for internal capacitor to get discharge.

To improve noise immunity and to get fast discharge slight modification is done in DTL gate.

(1) Diode $D_4$ is added for more noise immunity, as diode is added more voltage at point X will be required to drive transistor $Q_1$ in saturation.

$$V_X \text{ should be } \geq V_{F_{D_3}} + V_{F_{D_4}} + V_{BE\,(Q1)}.$$

(2) Negative supply $V_{BB}$ at base will provide fast discharge path for internal capacitor to get discharge. Secondly forcing negative voltage across base will built fast depletion region inside base of transistor $Q_1$, so $Q_1$ cutoff at fast rate.

Modified DTL gate, is most widely used form of DTL gate available in IC form. Refer Fig. 13.20.

Two changes have been implemented.

(1) Diode $D_3$ is replaced by emitter follower circuit (transistor $Q_2$). Purpose of providing $Q_2$ is extra turn on current for $Q_1$, there by increasing fanout capability and also switching speed. Base current of $Q_1$ is amplified by $h_{fe}$ of $Q_2$ times. Thus driving current for $Q_1$ increased.

(2) $R_B$ is returned to ground, instead of negative supply.

This is done for single power supply operation. Minimum voltage at point Y required to drive $Q_1$ in saturation is,

$$V_Y = V_{BE2} + V_{F_{D_4}} + V_{BE1} = 0.7 + 0.7 + 0.7 = 2.1V \quad \ldots (1).$$

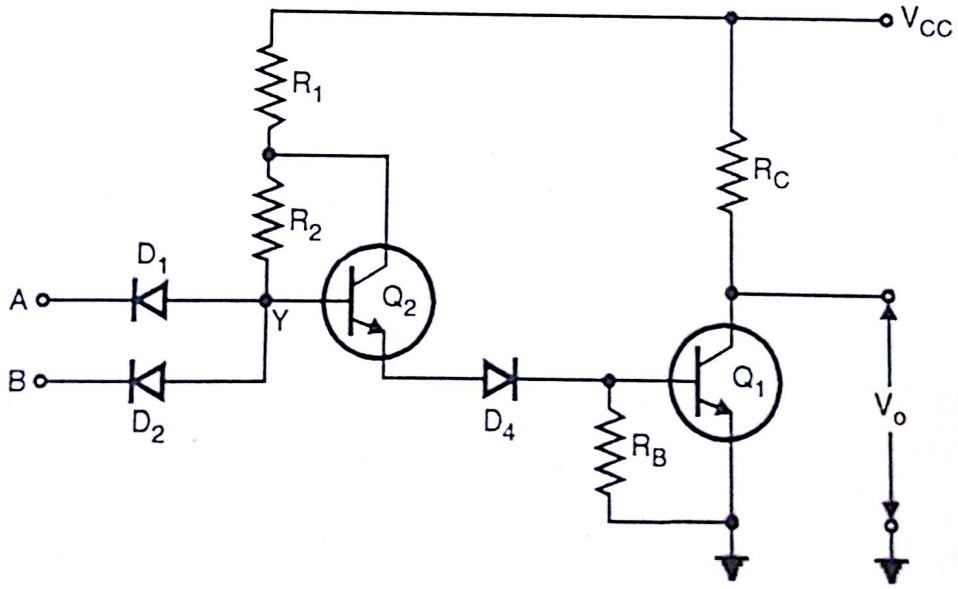



**Fig. 13.22 : Modified DTL NAND gate**

## Features :

(1) DTL circuit switches faster then RTL circuit, because signal passes through low forward resistance of diodes to the transistor. Typical delay is 25 η sec.

(2) Fan out, as high as 8 is possible because of high input impedance of the subsequent gates in logic 1 state.

Use of diodes rather than resistors and capacitors makes the DTL circuit more economical in IC form.

Scanned by CamScanner

### 13.4.5 High Threshold Logic Gate (HTL) :

When environment is very noisy the above logic families will not work satisfactorily. We require logic family having very high noise immunity. As name indicates HTL family uses supply voltage 15 V instead of 5V. The Fig. 13.23 shows NAND in HTL family.

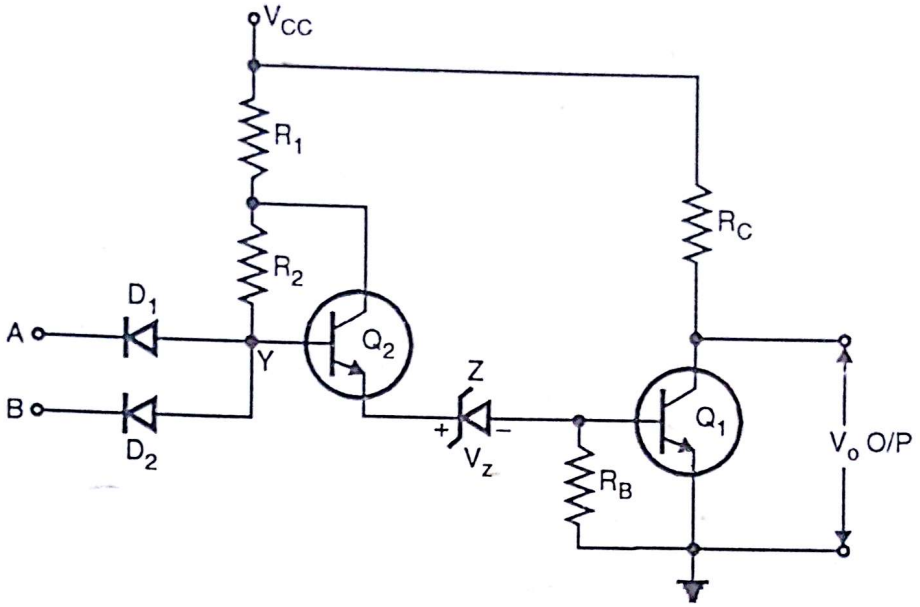



**Fig. 13.23 : HTL NAND gate**

**Working :**

(1) When A = B = 0, diodes $D_1$, $D_2$ gets forward biased, therefore potential at point Y with respect to ground is $V_Y = V_F \cong 0.7V$. This voltage is not sufficient to forward biased transistor $Q_2$ and breakdown zener, therefore $Q_1$ is OFF providing $V_O \cong 15$ V (logic 1).

(2) When either of the input A or B is 0, $V_Y = 0.7V$, $\therefore$ $Q_1$ = OFF. Voltage at point Y require to forward biased $Q_2$, $Q_1$ and breakdown zener requires

$$V_Y = V_{BE2} + V_t + V_{BE1} = 0.7 + 6.8 + 0.7 = 8.2 \text{ V} \quad \ldots(2).$$

(zener breakdown voltage = 6.8 V).

Therefore minimum voltage at A and B input required is

$$V_{A/B} = V_Y - V_{FD1/D2} = 8.2 - 0.7 = 7.5 \text{ V} \quad \ldots(3)$$

Compare Equations (1) and (2) the difference is 6.1, therefore obviously noise immunity is more in case of HTL.

(3) When A = B = 1 = high logic = 15 V. $D_1$ and $D_2$ will get reverse biased and $Q_2$ will be ON, breaking zener and switching on $Q_1$. Therefore $V_o = V_{CE(sat)}$.

**Advantages Over DTL :**

(1) Higher noise immunity.

(2) INPUT – OUTPUT characteristic is independent of temperature variable.

**Disadvantages :**

(1) Slow speed because as supply increased resistor values were increased, therefore stray and device capacitance need charge / discharge through resistance, thus increasing time constant.

Scanned by CamScanner

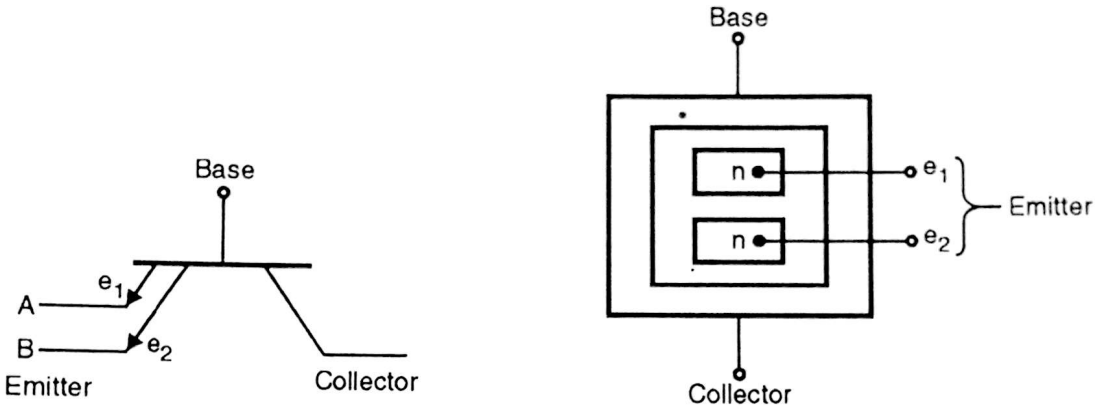
## 13.5 Transistor – Transistor Logic (TTL) ($T^2$ L Logic Family) :

**Problem in DTL :**

In DTL, when transistor $Q_1$ was coming out of saturation, diode $D_3$ becomes reverse biased and internal stored charge discharges through $R_B$. This is slow process of discharging, DTL is having speed limitation. To increase the speed, DTL has been modified. Modification is diodes $D_1$, $D_2$ and $D_3$ are replaced by multi emitter transistor.

***What is multiemitter transistor ?***

Fig. 13.24 shows multi–emitter transistor.




**(a) Symbol** **(b) Lateral geometry of multi emitter transistor**

**Fig. 13.24**

For the fabrication of multi–emitter transistor, single isolated collector region is diffused, a single base region is diffused and formed in the collector region. The several emitter regions are diffused as separate areas in the base region.

***Why one should use multi emitter configuration ?***

The answer is

(1) Saving in silicon area and onward package density i.e. no. of gates will be more for given space.

(2) The switching speed is improved because transistor at the input speeds up process of coming out of saturation, of transistor $Q_1$.

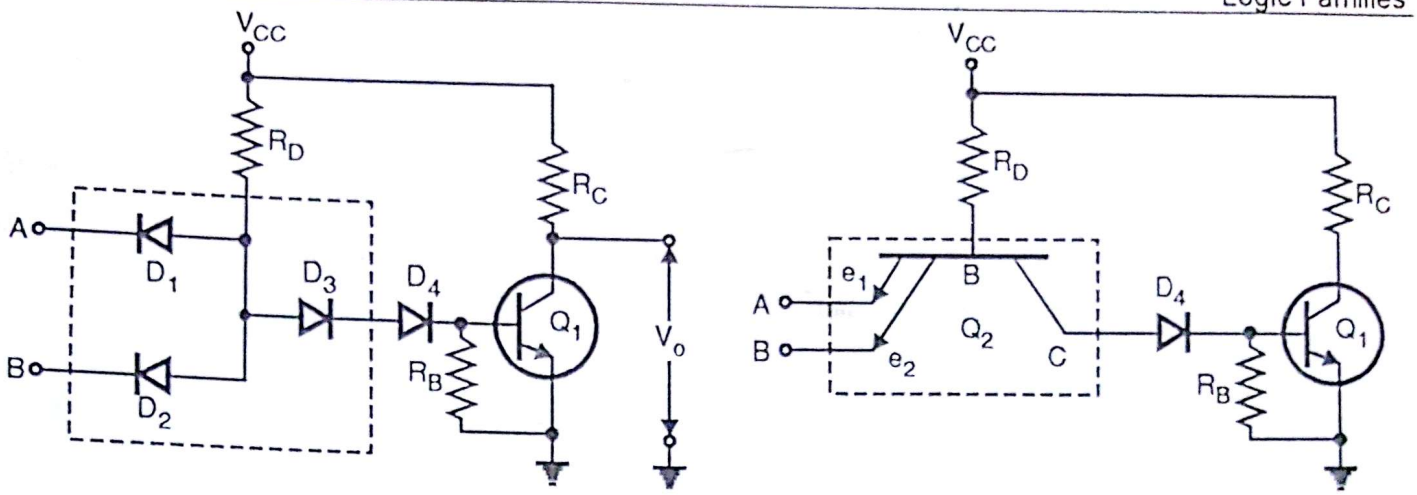
Multi–emitter transistor operates in two modes.

(1) Normal (2) Inverse.

In normal mode; emitter, base and collector functions as emitter, base and collector only. But in inverse mode emitter acts as collector and collector acts as emitter. For more detail let's see basic TTL NAND gate.

As shown in Fig. 13.25, diodes $D_1$, $D_2$ and $D_3$ are replaced by multi emitter transistor. The circuit is still modified further by removing $R_B$. This resistance is now not necessary because discharge will be provided by $Q_2$ i.e. multi emitter transistor. Secondly diode $D_4$ is also removed. The diode was introduced to increase noise immunity.

Scanned by CamScanner




(a) DTL NAND gate (b) TTL NAND

**Fig. 13.25**

Therefore final circuit is as shown in Fig. 13.26

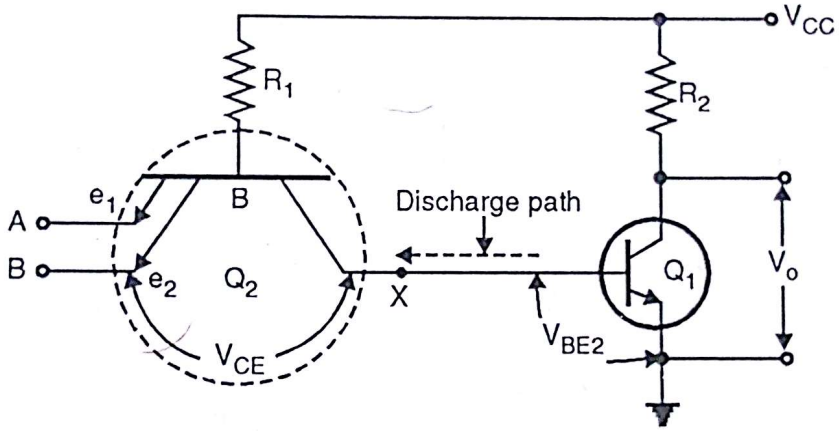



**Fig. 13.26 : TTL NAND gate**

**Working :**

(1) When A = B = 1, $Q_2$ will be OFF. In this case multi emitter transistor $Q_2$ act in ***Inverse mode*** i.e. collector acts as emitter and emitter acts as collector. Therefore $Q_1$ gets base drive from $V_{CC}$, $R_1$ and forward biased collector junction (collector acts as emitter) to the base. ∴ $Q_1$ is ON. $Q_1$ is now in saturation. ∴ $V_o = V_{CE(sat)} \cong 0.2V$ (logic 0).

(2) When A = B = 0 or acts either of the terminal is at logic 0 (0.2V) potential, $Q_2$ will be forward biased. In this, transistor $Q_2$ acting in *Normal* mode. Current will be flowing through $V_{CC}$, $R_1$ forward biased base emitter junction ($Be_1$, $Be_2$) to ground. Therefore $V_{BE1} = 0.7V$. Ground potential is 0.2V.

Potential at base B = 0.7 + 0.2V = 0.9V. To forward biased $Q_1$ we require voltage 0.7 (base to collector junction) + 0.7 ($V_{BE1}$ ($Q_1$)) = 1.4 V. ∴ transistor $Q_1$ comes out of saturation and it will be OFF.

***How stored charge of $Q_1$ gets discharged ?***

The stored charge in $Q_1$ will leaks off through collector of transistor $Q_2$ to ground. As ON resistance of transistor is very low discharge is very fast. Therefore $Q_1$ comes out of saturation very fast. Discharge path is shown by dotted arrow in Fig. 13.26. In comparison with DTL, TTL is ten times faster.

Scanned by CamScanner

## 13.5.1 Totem Pole Output Stage or Active Pull–up :

**Problem in TTL Circuit :**

Normally, output impedance of any electronic circuit should be as low as possible. Now let's analyze the same for TTL gate.

Refer Fig. 13.26 (b). When $Q_1$ is ON i.e. $V_o$ = 0 level = $V_{CE\ (sat)}$, impedance looking into the output side is very low (approx. 8 – 10 Ω). So when output is at LOW level impedance is low.

But if $Q_1$ is OFF i.e. $V_o \cong$ logic 1, output impedance looking into the network is equal to load resistance $R_C$. Normally $R_C$ is in kΩ.

Therefore output impedance is high, when $V_o$ = logic 1. Therefore charging time of external stray and input junction capacitance increases, thus speed of operation is affected.

Solution of above problem is one can reduce $R_C$ but then current $I_C$ will increase, increasing power dissipation. Secondly $Q_1$ will go into hard saturation, so coming out of saturation will take more time.

Second problem of decreasing $R_C$, is as shown in Fig. 13.27.

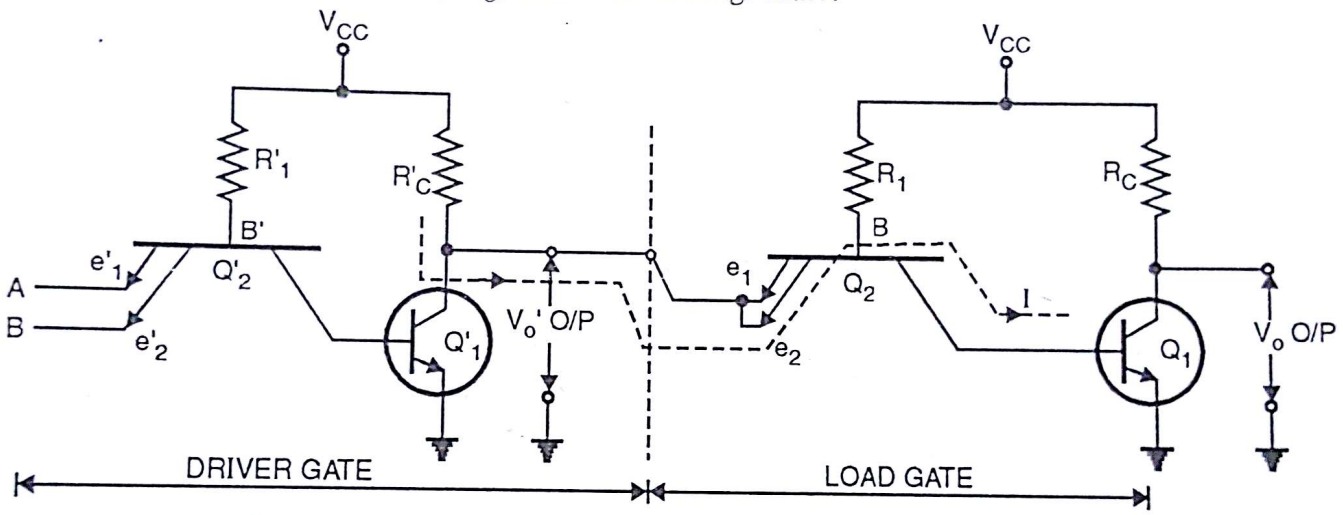



**Fig. 13.27(a)**

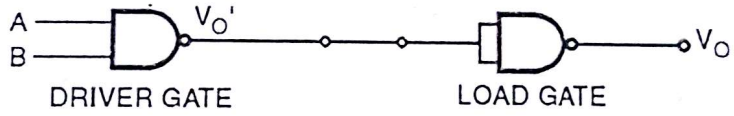



**Fig. 13.27(b)**

Fig. 13.27 (a) shows one NAND gate drives another. Output driver gate connected to input of load gate. Equivalent circuit is shown in Fig. 13.27 (b).

When A = B = 0, of driver gate, $Q_2'$ works in normal mode, therefore $Q_1'$ = OFF and produces HIGH output. $V_o'$ = High and it is given to input of load gate.

Therefore $Q_2$ works in inverse mode (i.e. emitter as collector and collector as emitter). In this mode gain is high, therefore $Q_2$ will fetch current from $V_{CC}$, $R_C'$, $e_1/e_2$and collector of $Q_2$ to base of $Q_1$. (Current I path is shown by dotted line Fig. 13.27 (a)). As current drawn is more, drop across $R_C'$ will be more, therefore $V_o'$ of driver gate with respect to ground reduce i.e. instead of $V_o' = V_{CC}$, it will be lesser than $V_{CC}$ and therefore noise immunity decreases.

So to reduce output impedance, best solution is to use ***emitter follower*** circuit. The basic characteristic of emitter follower is output impedance is very low. The output stage using emitter follower is called ***totem pole*** output stage or ***Active pull up stage***. The circuit of totem pole output stage is shown in Fig. 13.28.

Scanned by CamScanner

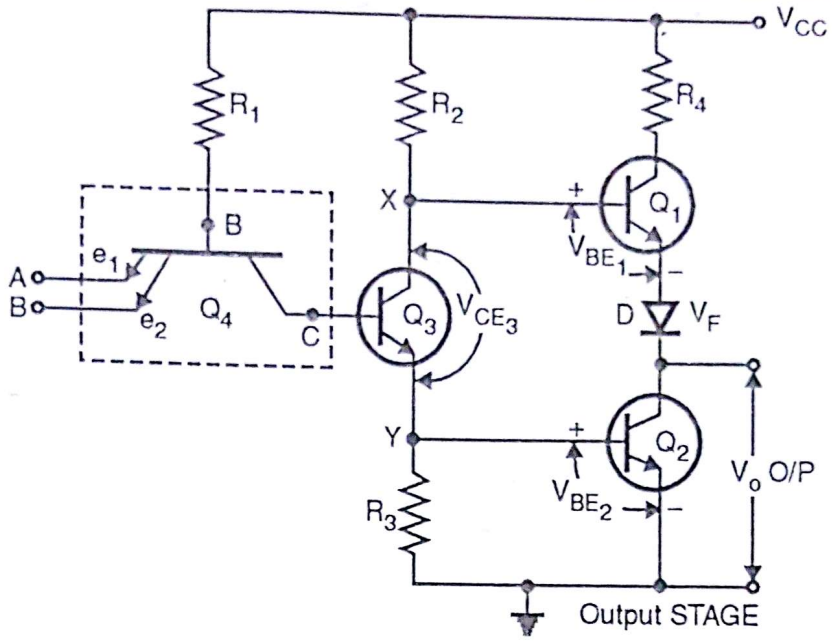



**Fig. 13.28**

$Q_4$ is multi-emitter transistor. $Q_1$, $Q_2$, diode D and $R_4$ forms output stage.

**Working :**

(1) When A = B = 0 or either of the terminal is '0' , $Q_4$ is ON. $Q_4$ works in normal mode. Therefore transistor $Q_3$ is OFF. Therefore $V_Y \cong 0$ and $V_X \cong V_{CC}$. Therefore $Q_2$ is OFF. As the potential at X point is VCC it will drive $Q_1$ ON. The output is taken from emitter of $Q_1$ after diode. Therefore impedance is low.

$$V_o = V_{CC} - V_{BE_1} - V_F$$

$$\therefore \quad V_o = 5 - 0.7 - 0.7 = 3.6 \text{ V}$$

How the circuit will look like when $Q_2$ is off is shown in Fig. 13.29.

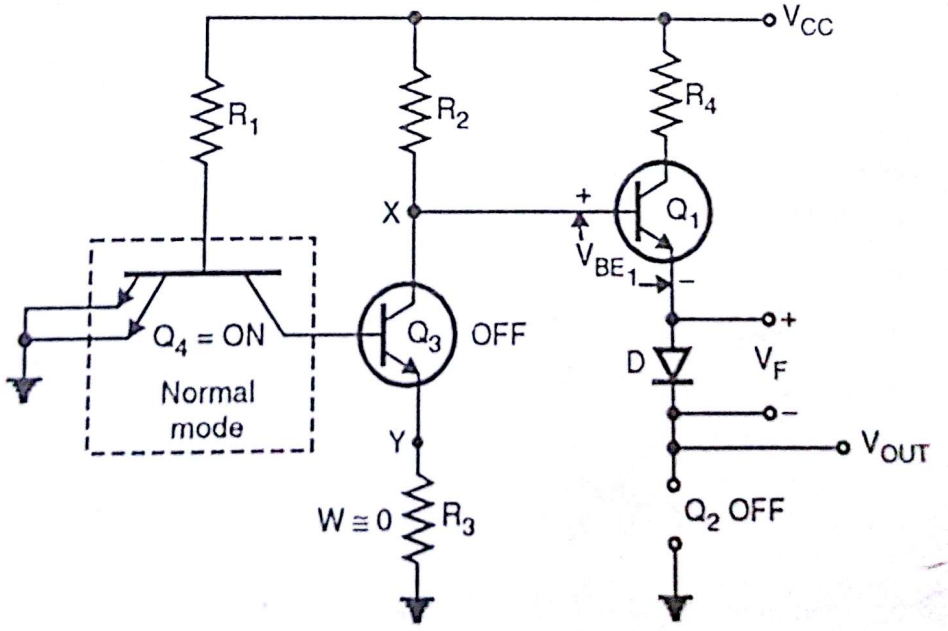



**Fig. 13.29**

Eventhough output is at high logic output impedance is low.

Scanned by CamScanner

(2) When A = B = 1 transistor $Q_4$ will work in Inverse mode. Therefore $Q_3$ conducts as it will receive base drive from $Q_4$. As $Q_3$ is ON, $Q_2$ will also be ON, 'cause at point Y sufficient voltage will be produced.

$\therefore \quad V_Y = 0.7$ V.

$V_X = V_{CE_3} + V_Y = V_{CE_{3\,(Sat)}} + V_Y = 0.2 + 0.7 = 0.9$ V.

But to switch on transistor $Q_1$ we require $V_X = V_{BE1} + V_F = 0.7 + 0.7 = 1.4$ V.

But $V_X$ is 0.9V, therefore $Q_1$ and diode D is Cut-off. $V_o = V_{CE_{2\,(Sat)}} \cong 0.2$ V = logic. The circuit will be as shown in Fig. 13.30.

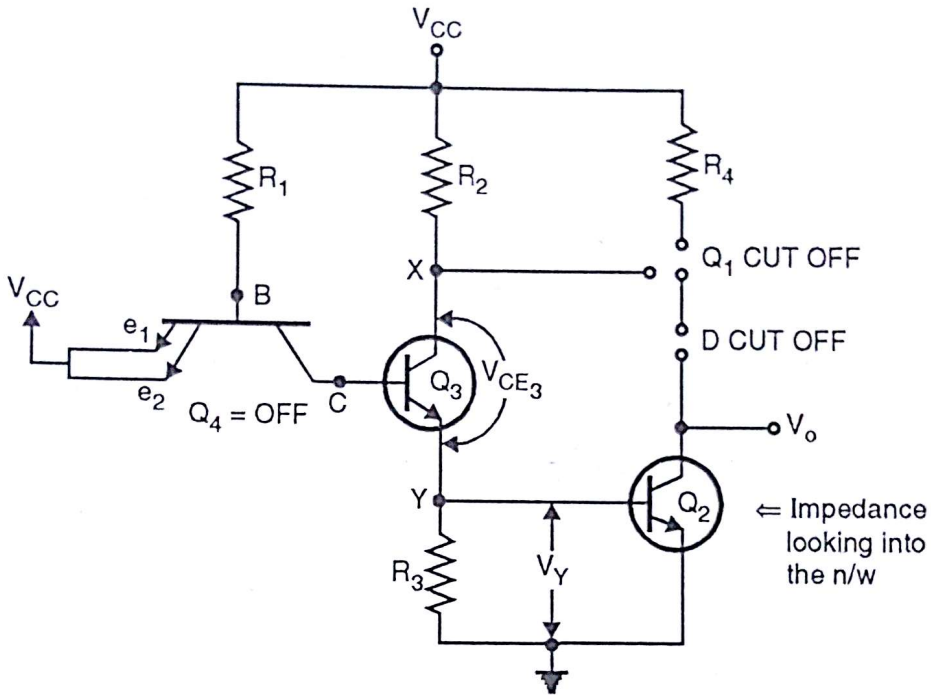



**Fig. 13.30**

If you look into output network, impedance will be low i.e. impedance of ON transistor $Q_2$ (8 to 10 Ω), therefore at LOW output impedance is also low.

So from point 1 and 2, we conclude that irrespective of output level impedance is low.

(3) **Keep A and B floating :**

Floating means left open. When A = B = Float, emitter $e_1$ and $e_2$ will be open. Therefore current will not pass through emitter, means $Q_4$ will operate in inverse mode (collector base junction gets forward biased.)

As $Q_4$ is OFF, $Q_3$ is ON. $Q_3$ will produce sufficient potential at Y point so that $Q_2$ also conducts. $\therefore V_Y = 0.7$ V, $V_X = V_{CE_{3\,(Sat)}} + V_Y = 0.9$, not sufficient to drive $Q_1$ and diode D to turn it on.

$$V_o = V_{CE_{2(Sat)}} \cong 0.2 \text{ V} = \text{logic 0.}$$

If you observe point (2) and (3), you will find the circuit behavior is same and output is also same.

**Conclusion :** Floating input terminal(s) in TTL is treated as ***HIGH level*** input.

It is specifically mentioned ***input terminal(s)***, it doesn't mean that you should keep $V_{CC}$ terminal floating and expect circuit should function. Sorry, it won't work.

Looking to totem pole circuit you may ask why $R_4$ and D is connected. What is the function of $R_4$ and D. To understand function, let's see operation without these components.

Scanned by CamScanner

### Function of Diode D

As we have seen in point (2) and (3) of previous discussion, when A = B = 1 or float, $Q_4$ is OFF, therefore $Q_3$ conducts forcing $Q_2$ to conduct. As $Q_3$ conducts, $V_X = V_{CE_{3(sat)}} + V_Y = 0.2 + 0.7 = 0.9$. Now let's consider diode D not present. Therefore $V_X = 0.9$, is sufficient voltage to drive $Q_1$ ON. So at output stage $Q_1$ and $Q_2$ both will conduct and in that if $R_4$ is also not present then it will be simply short circuit and IC will get damage. Therefore diode D is introduced, so $V_X = 0.9$, will not be sufficient to drive $Q_1$ and diode D. Therefore $Q_1$ and D are cutoff.

The circuit condition without D and $R_4$ is shown in Fig. 13.31.

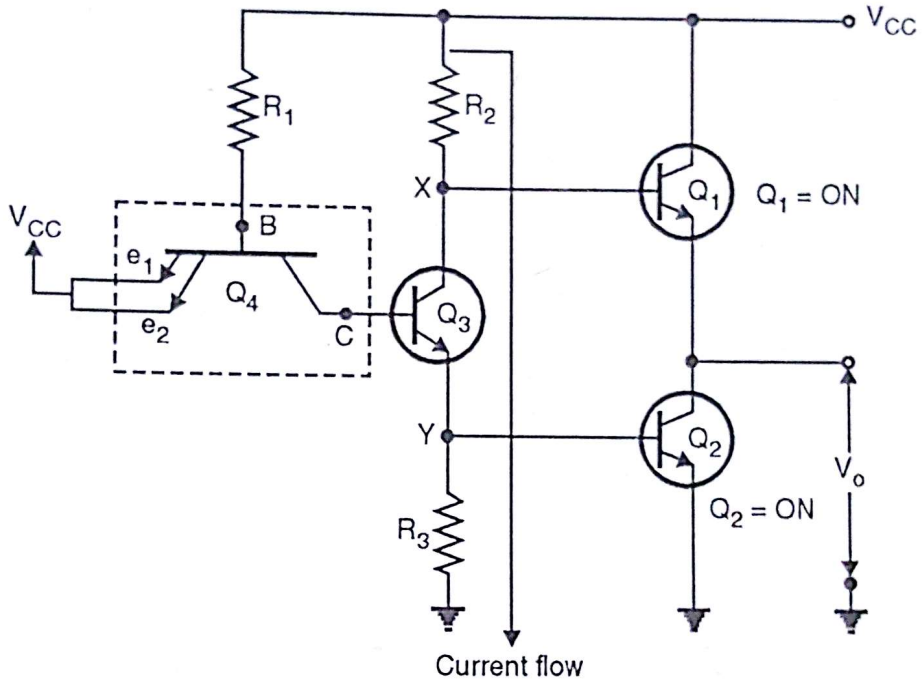



**Fig. 13.31**

### Function of Resistor $R_4$

Remember basic point that even though all transistors are fabricated at same time, one cannot guarantee about characteristics of transistor that it will be identical.

So let's consider $Q_2$ is slow and $Q_1$ is fast.

When input to A = B = 1 transistor $Q_4$ OFF operates in inverse mode, $Q_3$ is ON and therefore $Q_2$ is ON. $Q_1$ is OFF. Now let's say inputs are changed from 1 to 0. Therefore A = B = 0. As inputs are 0. $Q_2$ will try to come out of saturation and $Q_1$ will try to go into saturation. As $Q_1$ is faster it will saturate fast. But as $Q_2$ is slow it will take sometime to come out of saturation, therefore for sometime $Q_1$ and $Q_2$ both are ON. This may force large current and damage transistor $Q_2$ and $Q_3$, therefore as a current limit $R_4$ is introduced.

Secondly $R_4$ provides current limit if output is grounded accidentally.

## 13.5.2 Transfer Characteristic of TTL Gate :

Transfer characteristic give us relationship between input given and output appeared. It shows valid region for operating TTL gates. Fig. 13.32 shows x' fer (Transfer) characteristic.

Scanned by CamScanner

Refer both the figures for explanation.

(1) $V_I < V_a$ :

If $V_I < V_a$ transistor $Q_2$ and $Q_3$ are turned off. This causes transistor $Q_1$ to conduct and output is logic 1.

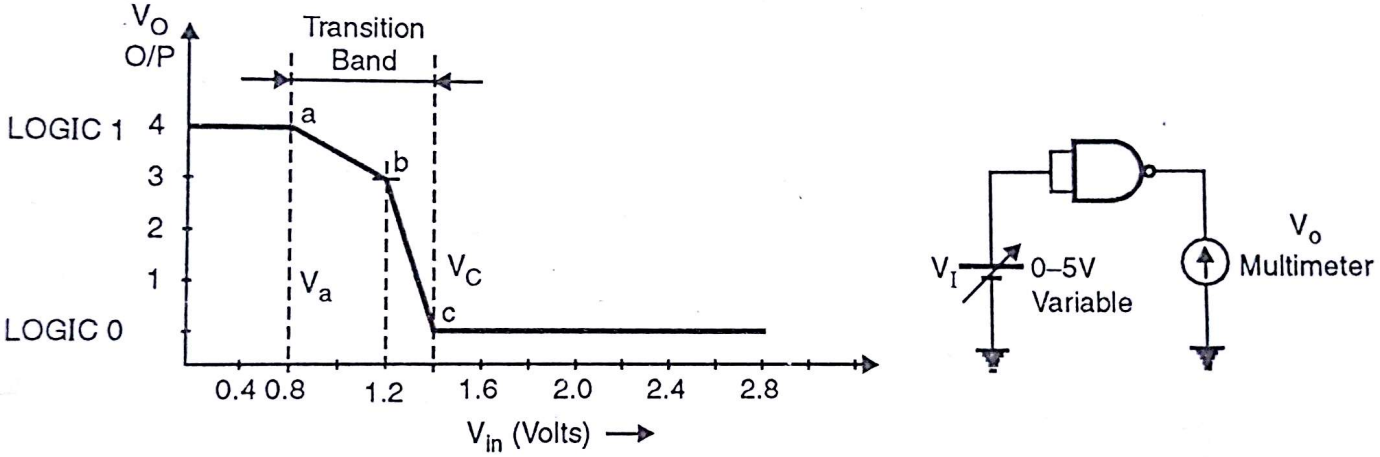
$$V_o = V_{OH} = V_{CC} - V_{BE1} - V_F = 5 - 0.7 - 0.7 = 3.6 \text{ V}.$$

(2) $V_I > V_C$ :

Transistor $Q_2$ and $Q_3$ conducts, $Q_1$ is OFF and output is at logic 0. Output logic 0 is determined by saturation resistance of $Q_2$ and current sink.

(3) $V_a < V_I < V_C$ :

This region is transition region. At point a in Fig. 13.32(a), $Q_4$ base current is gradually diverted from emitter of $Q_4$ to collector of $Q_4$. This conduction occurs at approx. 0.7 V. $Q_3$ is now operating in linear region with gain from input to collector determined by $R_2 / R_3$. As $Q_1$ is ON the output follows characteristic of $Q_3$ and therefore decreases at slope of 1.6 (at a to b). At point b, input is high enough to cause $Q_2$ to conduct, therefore gain of $Q_3$ increases hence steep slope region between points b and c. $Q_2$ turns off at point C and output is at logic 0.




(a) Transfer characteristic (b) Circuit diagram

**Fig. 13.32**

## 13.5.3 Current Sink and Source in TTL :

To understand current sink and source let's connect load NAND gate to driver NAND gate.

**Current Sink :**

Fig. 13.33 shows output of driver gate connected toload gate. Inputs to driver gate are A = B = 1 (not shown), therefore $Q_3$ and $Q_2$ are ON, $Q_1$ and diode D are OFF, thus produce LOW logic at the output.

Because of LOW logic input to load gate $Q_4$ load gate works in normal mode and current flows from $V_{CC}$, $R_1$ (load gate), has emitter junction (load gate) to link and to $Q_2$ (driver gate). So with respect to driver current, entering into output terminal, therefore current is called ***sink*** current. Driver gate 'sinks' current from load gate.

Scanned by CamScanner

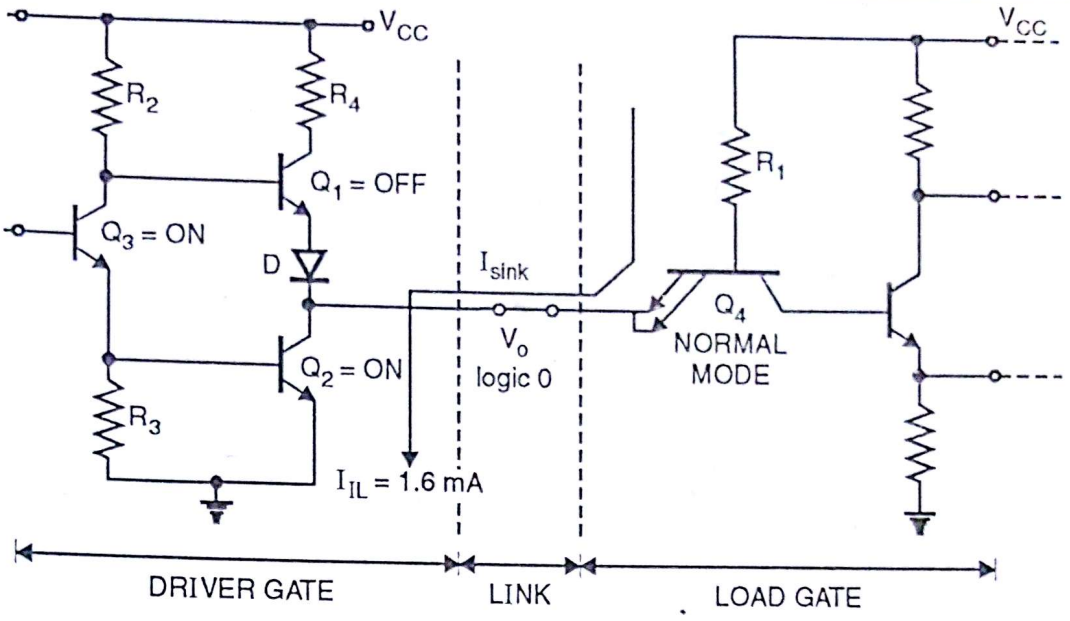



**Fig. 13.33**

**Current Source :**

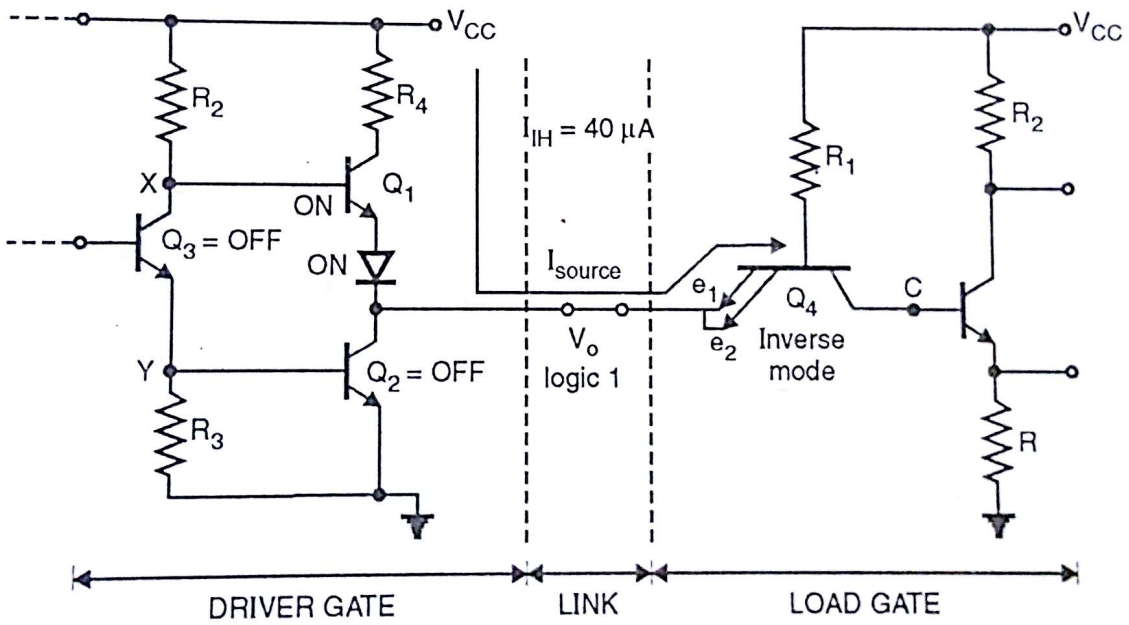



**Fig. 13.34**

When input A = B = 0 to driver gate input, $Q_3$ (driver gate) is OFF therefore $Q_2$ (driver gate) is also OFF. $V_X \cong V_{CC}$, therefore drives $Q_1$ and diode D in ON state. Thus $V_o$ of driver gate is at logic 1.

As input to load gate is logic 1, $Q_4$ works in inverse mode. The flow of current (I source) is from $V_{CC}$, $R_4$ (driver gate), $Q_1$ (driver gate), diode d (driver gate) to $e_1e_2$ (load gate). With respect to driver gate current is flowing out of the output terminal, therefore it is called ***source*** current. Driver gate sourcing load gate.

## 13.5.4 Clamping Diode :

Fig. 13.35 shows clamp diode connected to emitters of transistor $Q_4$.

Scanned by CamScanner

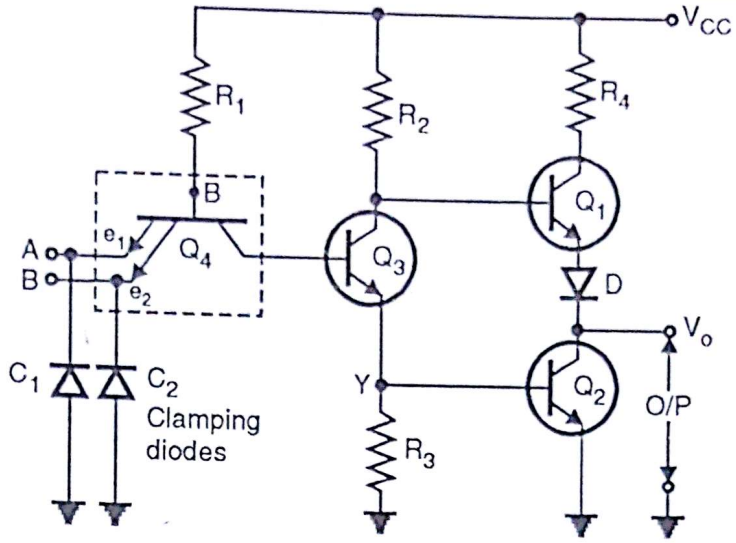
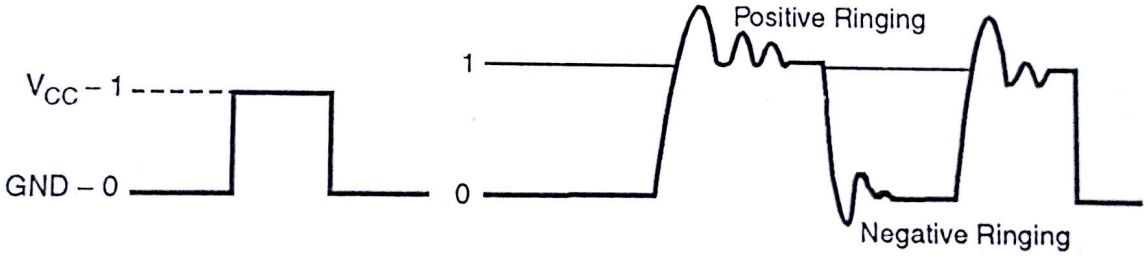



**Fig. 13.35**

## Working :

**Case I :** At normal ***HIGH*** or ***LOW*** level inputs–A and B, diodes $C_1$ and $C_2$ are reverse biased and will not be in picture.

**Problem :** But when transition occurs (transition is made up of high frequency harmonic components), any inductance associated with load causes ***ringing*** (high frequency oscillations) as shown in Fig. 13.36.




**(a) Expected wave** **(b) Wave with ringing**

**Fig. 13.36**

**Case II :** When ringing is on positive side voltage is greater than 5. More positive voltage will reverse bias emitter base junction of input transistor $Q_4$. Base resistance $R_1$ limits the current and thus protects input circuit.

**Case III :** But when ringing is on negative side, going below ground level, emitter of $Q_4$ is pulled more negative, this will forward bias $Q_4$ more, therefore it goes into hard saturation. This will lead undesired voltage spike at other node and can damage the chip, ∴ diodes (clamp) at the input 'clamps' (cuts) the ringing pulses and limits to maximum – 0.7V on negative side.

Scanned by CamScanner

### 13.5.5 Maximum Characteristic for Standard 54/74 TTL at $T_A$.= 25° C :

| Sr.No. | Parameter | Typical values |
|---|---|---|
| (1) | $V_{OH}$ /$V_{OL}$ | 3.5/0.2V |
| (2) | $V_{IH}$/$V_{IL}$ | 1.5/0.5V |
| (3) | $NM_H$ / $NM_L$ | 2/0.3V |
| (4) | Logic swing | 3.3V |
| (5) | Supply | +5V |
| (6) | Fan-out | 10 |
| (7) | $t_{pd}$ | 10 nsec |
| (8) | P.D. | 10 mw |

**Table 3**

## 13.6 TTL Families :

Since the introduction of the first bipolar TTL family, system designers have wanted more speed, less power consumption or a combination of the two attributes. With this, there were many development in the process and manufacturing technologies. Circuit design techniques were also advanced. These all development led to new subfamilies of TTL.

Presently following bipolar families are available.

| | |
|---|---|
| (1) TTL | DM 54/DM 74. |
| (2) Low power TTL | DM 54 L |
| (3) Low power Schottky | DM 54 LS/DM 74 LS |
| (4) Advanced low power Schottky | DM 54 ALS/ DM 74 ALS |
| (5) Schottky | DM 54 S / DM 74 S |
| (6) Advanced Schottky | DM 54 AS/DM 74 AS |
| (7) Fast | DM 54 F/ DM 74 F. |

**Note :** 54 series is for military grade
74 series is for commercial industrial grade.

The main difference between two series is temperature range and power supply fluctuation.

***Temperature range for military / Aerospace 54 series is – 55° C to + 125° C***

***Temperature range for commercial / industrial 74 series is 0° C to 70° C.***

***Power supply fluctuation valid for 54 series is 10%.***

***Power supply fluctuation valid for 74 series is 5 %***

### 13.6.1 TTL Logic (DM 54/DM 74) :

We have studied about the TTL logic just now. It was first saturated logic integrated circuit. It offers combination of speed, power consumption, output sources and sink capabilities. The basic gate features multi – emitter configuration for fast switching speed, active pullup output to provide low driving source impedance which also improves noise margin and device speed. Typical power dissipation is 10 mw per gate and typical propagation delay 10 ηsec. When driving 10 PF/400Ω load.

Scanned by CamScanner

### 13.6.2 Low Power TTL (54 L) :

(1) In this family resistor values are increased by nearly tenfold. Therefore P.D. (power dissipation) is less than 1/10 of TTL family.

(2) Speed is sacrificed at the cost of power, $t_{pd}$ increased three fold.

Typical P.D. = 1 mw per gate.

$t_{pd}$ = 33 η sec.

### 13.6.3 High Speed TTL :

We know that increasing the power dissipation will increase the speed. To improve transient response of TTL, we can decrease resistance values and increase power dissipation. This thinking prompted the development of high speed TTL (HTTL).

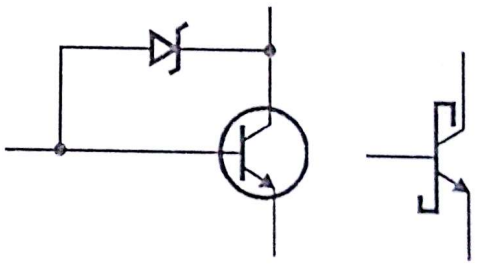
In HTTL you will find that in addition to decreased resistances, the output high driver section includes, Darlington pair configuration. Due to this more current will be sourced for charging load capacitance.

## 13.7 Schottky TTL :

A Schottky-barrier diode (SBD) is a metal semiconductor (usually MN) rectifying junction.

### 13.7.1 Schottky-Clamped BJT :

As shown, SBD is connected between base and collector of a BJT. SBD is incorporated as a clamp to divert excess base current and to prevent transistor from reaching deep saturation. The advantage of the decreased switching time of the SBJT is essentially accomplished with no additional fabrication. The only difference between the SBJT and the regular BJT is that, the base region metal contact is extended to overlap part of the collector N-region. Hence, the MN junction appearing between the base and collector terminals is simply the base metal in contact with collector N-region.



(a) Transistor and schottky barrier diode clamp (b) Symbol for transistor with clamp

Fig. 13.37

The schottky gate input and internal circuit resembles saturated standard TTL gate except resistor values are half of TTL value. Output has darlington pair for pull up, as shown in Fig. 13.38.

Typical P.D. = 20 mw per gate.

$t_{pd}$ = Three times as first as TTL, 3 η sec with load 15 PF/ 280 Ω.

$V_{IH}$ = 2.0 V $V_{IL}$ = 0.8 V

$V_{OH}$ = 2.7 V $V_{OL}$ = 0.5 V

Note that all the BJTs except $Q_5$, are SBJTs. Input clamp diodes are replaced by SBDs. Diode D in totempole stage has been replaced by $Q_5$. Transistor $Q_4$ and $Q_5$ make up a darlington pair operating as an emitter follower. This arrangement provides more source current to charge the load capacitance when the driver output is switched LOW-to-HIGH, thus reducing the transition time.

Scanned by CamScanner

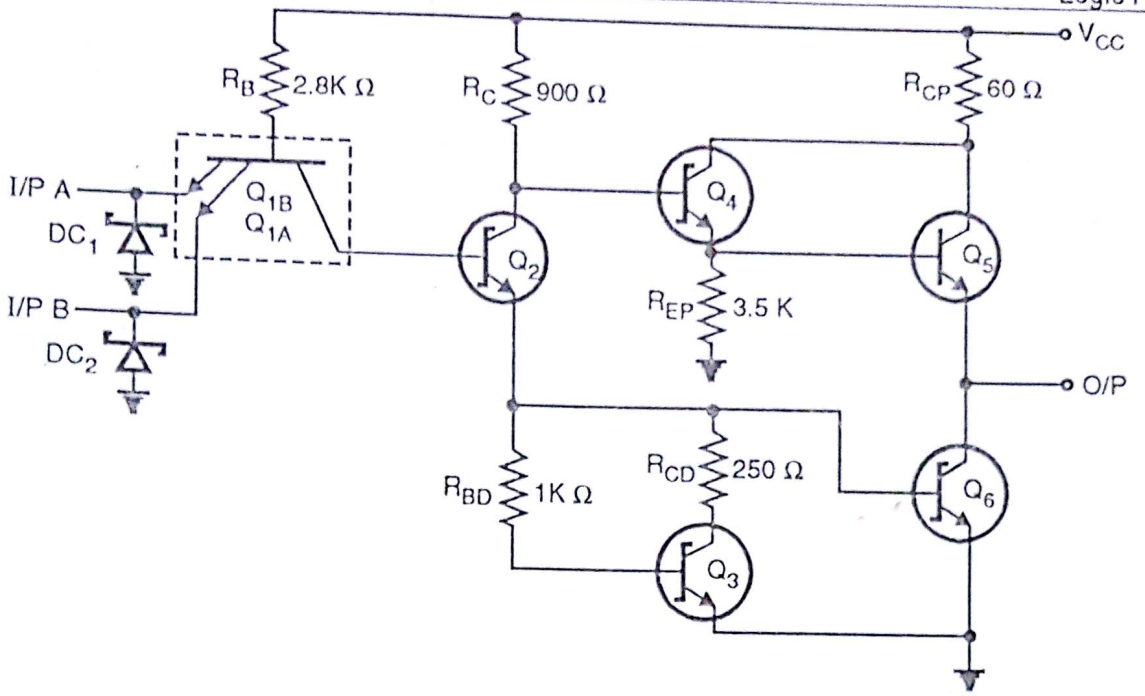



**Fig. 13.38 : Schottky TTL 74S00 series NAND gate**

## 13.7.2 Low Power STTL (LSTTL) :

Power dissipation of STTL was twice that of standard TTL. This problem prompted the development of Low power STTL.

### Large Resistors :

LSTTL resistor values are approximately ten times of STTL values. This leads to 1/10 of current and approximately 1/10 of power dissipation ; compared to TTL. However, fan-out is reduced and propagtion delay time is increased.

### Diode Input Section :

Main intention behind using multiemitter input transistor was to aid in stored charge removal from the base of $Q_1$. Since the SJBTs being utilized do not saturate, this is no longer necessary. Thus returning back to a Schottky diode input.

Advantage of using diode is :

The Schottky input diodes require about 1/3 of the Silicon surface area of multiemitter resistor. This reduces input parasitic capacitance.

## 13.7.3 Advanced LSTTL (ALSTTL) :

**Descriptions :** The advanced low power schottky family is one of the most advanced TTL family. It delivers twice the data handling efficiency and still provides upto 50% reduction in power consumption compared to LS family. This is possible because of new process where components are isolated by a selectively grown thick oxide layer rather than N junction used in conventional process. the refined process, coupled with improved circuit design technique yields smaller component geometry's, shallower diffusion and lower junction capacitance. This enables device to have increased $f_T$ in excess of 5 GHz switching speed

Scanned by CamScanner

by factor of two, while offering much lower operating currents. But at the same time ALS devices are more susceptible to damage from electrostatic discharges. Additional protection via base emitter shorted transistor is included at the input for rapid discharge of high voltage static electricity.

Inputs and outputs are clamped to suppress excessive ringing. A buried $N^+$ guard ring around all inputs and output structure prevents crosstalk.

### 13.7.4 Fairchild Advanced Schottky TTL (Fast) :

FAST (Fairchild advanced schottky TTL) circuits are made with the advanced isoplanar II process, which produces transistors with very high, well controlled switching speeds, extremely small parasitic capacitance's, and $f_T$ in excess of 5 MHz.

In the isoplanar process, components are isolated by a selectively grown thick oxide rather than $p^t$ isolation region used in the planar process. Since the oxide needs no separation from the base – collector regions. component and chip sizes are substantially reduced. The base and emitter ends terminate in the oxide wall ; masks can thus overlap device area into isolation oxide. This overlap feature eliminates extremely close tolerances normally required for base and emitter marking.

### 13.7.5 Advanced Schottky TTL (ASTTL) :

The final and most complex TTL logic family is ASTTL. This was introduced to meet need of systems designer who requires ultimate speed.

## 13.8 Comparison of TTL Families :

We have studied subfamilies of TTL. Let's compare all the families.

**Table 4 : Typical TTL series characteristics**

| | 74 | 74S | 74LS | 74AS | 74ALS | 74F |
|---|---|---|---|---|---|---|
| **Performance ratings :** | | | | | | |
| Propagation delay (ns) | 9 | 3 | 9.5 | 1.7 | 4 | 3 |
| Power dissipation (mW) | 10 | 20 | 2 | 8 | 1.2 | 6 |
| Speed-power product (pJ) | 90 | 60 | 19 | 13.6 | 4.8 | 18 |
| Max.clock rate (MHz) | 35 | 125 | 45 | 200 | 70 | 100 |
| Fan-out (same series) | 10 | 20 | 20 | 40 | 20 | 33 |
| **Voltage parameters :** | | | | | | |
| $V_{OH\ (min)}$ | 2.4 | 2.7 | 2.7 | 2.5 | 2.5 | 2.5 |
| $V_{OL\ (max)}$ | 0.4 | 0.5 | 0.5 | 0.5 | 0.4 | 0.5 |
| $V_{IH\ (min)}$ | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| $V_{IL\ (max)}$ | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 |
| **Current parameters :** | | | | | | |
| $I_{OH\ (min)}$ mA | –0.4 | – 1 | – 0.4 | – 2 | – 0.4 | – 1 |
| $I_{OL\ (min)}$ mA | 16 | 20 | 8 | 20 | 8 | 20 |
| $I_{IH\ (max)}$ μA | 40 | 50 | 20 | 20 | 20 | 20 |
| $I_{IL\ (max)}$ mA | – 1.6 | –2 | – 0.4 | – 0.5 | – 0.1 | –0.6 |

Scanned by CamScanner

The most common way of comparing logic families is by using their speed power product. Fig. 13.39 provides graphical representation of the same.

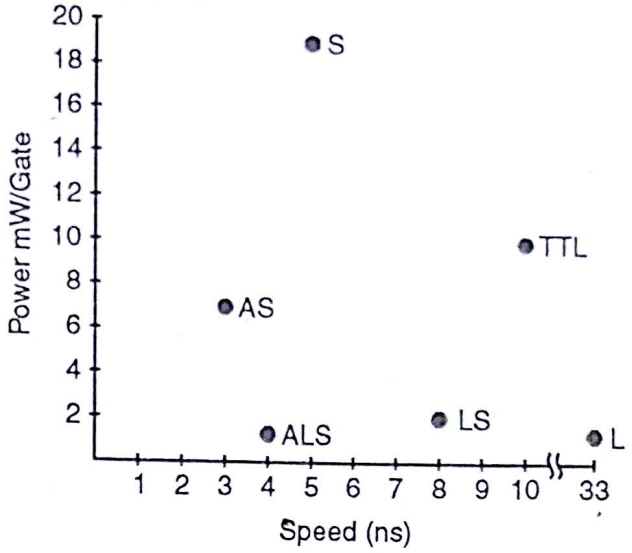



**Fig. 13.39 : Speed power product comparison**

Each of the logic families is a compromise between speed and power consumption the speed / power product is approximately a constant, a decrease in the power consumption must be trade off in slowing down the device and vice-versa. The same is depicted graphically as follows :

| | Speed | | Power Consumption |
|---|---|---|---|
| Fastest | AS/F | Low | L |
| ↑ | S | ↓ | ALS |
| | ALS | | LS |
| | LS | | F |
| | TTL | | AS |
| | L | | TTL |
| Slowest | | High | S |

**Ex. 1 :** Calcualte high level noise margin and low level noise margin for all TTL series.

**Soln. :** High level noise margin = $V_{NH} = V_{OH\ (min)} - V_{IH\ (min)}$

Low level noise margin = $V_{NL} = V_{IL\ (max)} - V_{OL\ (max)}$

**Table 5**

| Parameter | 74 | 74S | 74LS | 74AS | 74ALS | 74F |
|---|---|---|---|---|---|---|
| $V_{NH}$ | 0.4 | 0.7 | 0.7 | 0.5 | 0.5 | 0.5 |
| $V_{NL}$ | 0.4 | 0.3 | 0.3 | 0.3 | 0.4 | 0.3 |

**Ex. 2 :** Calculate FAN-OUT capability of the different TTL series. Note that driver gate and driven gate (load gate) are from that family only ;

**Soln. :** If the driver and load gate are from the same family we have two equations

$$\text{Fan-out (LOW)} = \frac{I_{OL\ (min)}}{I_{IL\ (max)}}$$

$$\text{Fan-out (HIGH)} = \frac{I_{OH\ (min)}}{I_{IH\ (max)}}$$

Scanned by CamScanner

**Table 6**

| Parameter | 74 | 74S | 74LS | 74AS | 74ALS | 74F |
|---|---|---|---|---|---|---|
| Fan-out (LOW) | 10 | 10 | 20 | 40 | 80 | ≡33 |
| Fan-out (HIGH) | 10 | 20 | 20 | 100 | 20 | 50 |

**Note :** To find out final fan-out, just consider the lowest value from fan-out (LOW) and fan-out (HIGH).

**Ex. 3 :** How many 74S gates will be driven by gate of 74ALS ?

**Soln. :** The example specifically asks for surfacing two different logic families. If this is the case. Follow two standard rules.

***Rule 1 :*** ***Add up the $I_{IH}$ for all inputs connected to an output. The sum must be less than the output's $I_{OH}$ specification.***

***Rule 2 :*** ***Add up the $I_{IL}$ for all inputs connected to an output. This sum must be less than the output's $I_{OL}$ specification.***

For our case, we have all 74S gates,

$$\therefore \quad \text{fan-out (HIGH)} = \frac{I_{OH\,(min)} \text{ of 74 ALS}}{I_{IH\,(max)} \text{ of 74 S}}$$

$$= \frac{0.4 \text{ mA}}{50\ \mu\text{A}} = \frac{400}{50}$$

$$= 20$$

$$\text{fan-out (LOW)} = \frac{I_{OL\,(min)} \text{ of 74 ALS}}{I_{IL\,(max)} \text{ of 74S}}$$

$$= \frac{8 \text{ mA}}{2\text{mA}} = 4$$

$\therefore$ Final fan-out is 4.

## 13.9 Important Points while Handling TTL :

While working with TTL family, you have to consider few practical aspects. Let's make this session, question-answer type.

***Q.1 If input terminal is left unconnected what will happen ?***

**Ans. :** If in TTL circuit input is left unconnected or open, it acts like a logical 1 applied to that input. Reason is, emitter-base junction or diode at the input will not be forward-biased, when you apply logical 1 or you keep terminal floating or open.

***Q.2 What about unused inputs of TTL gate ?***

**Ans. :** Many times it may happen that, we will not require all the input terminals of the gate. For example, for 4 input AND gate, we may use only 3 inputs. Fourth we don't require. In such case you have following options :

1. *Either keep that terminal floating (Norrmally, this option is not most widely used).*

Scanned by CamScanner

2. *Connect that input to appropriate logic level, i.e. HIGH or LOW, permanently.*
3. *Short that terminal with any input terminal of the same gate. Refer Fig. 13.40(a).*

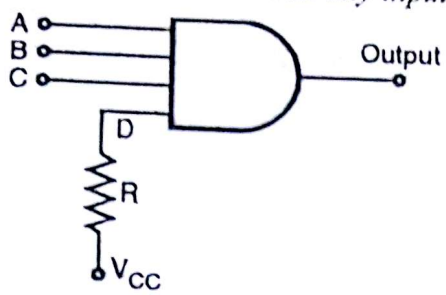



**Fig. 13.40 (a) : Connected to proper logic-level**

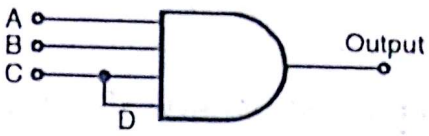



**Fig. 13.41 : Shared with any input terminal (Tied-together inputs)**

In third option, we are tieing two (or more) TTL inputs on the same gate to form common input. The common input will generally represent a load that is the sum of the load current rating of each individual input.

*Q.3* ***Do we come across problem of current transients in TTL logic ?***

**Ans. :** Yes, we do have problem of current transients in TTL due to TOTEM POLE structure. Refer Fig. 13.42.

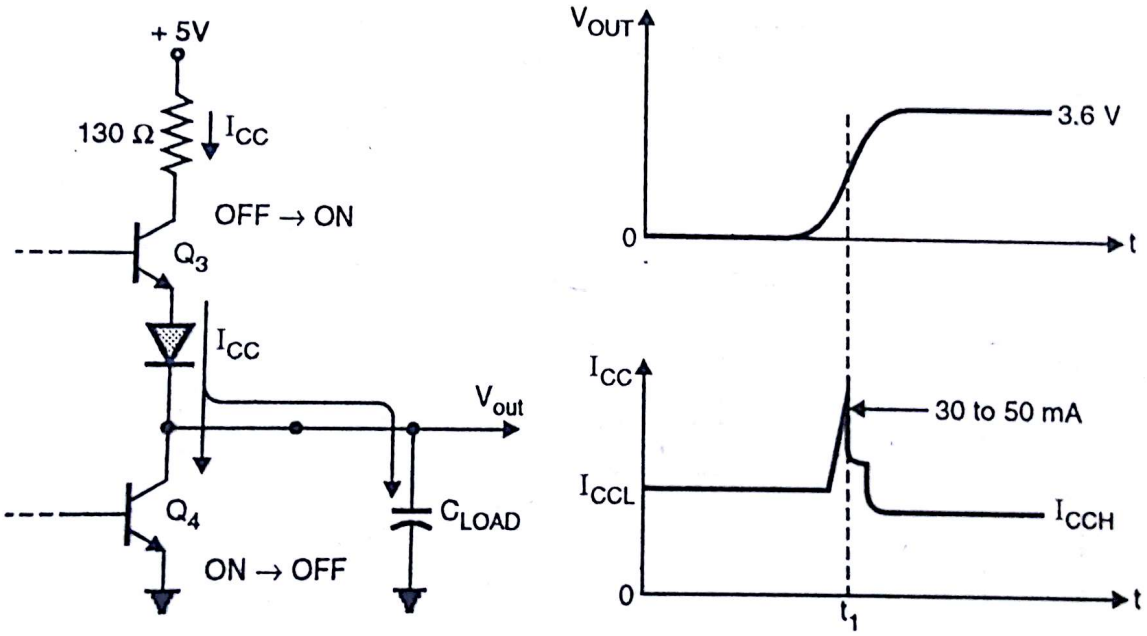



**Fig. 13.42 : A large current spike is drawn from $V_{CC}$ when a totem-pole output switches from LOW to HIGH**

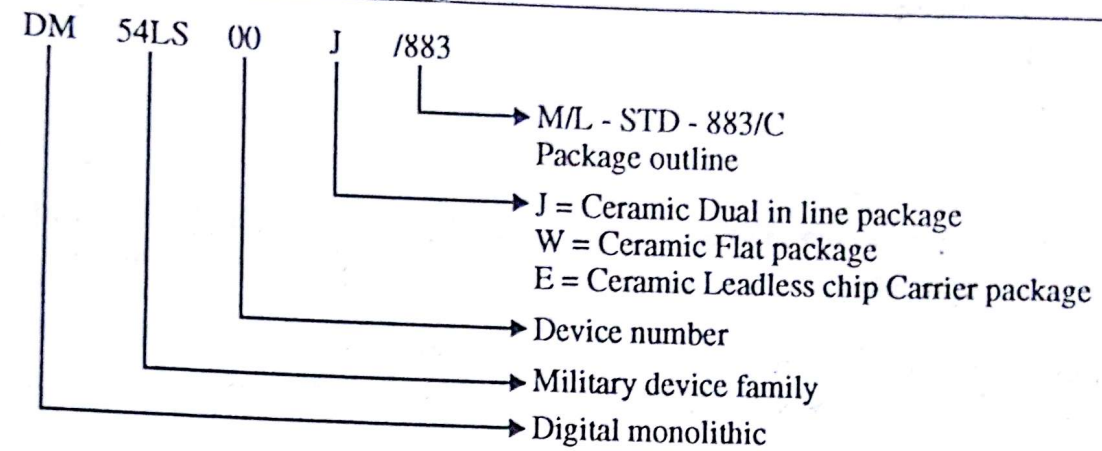
## Operation :

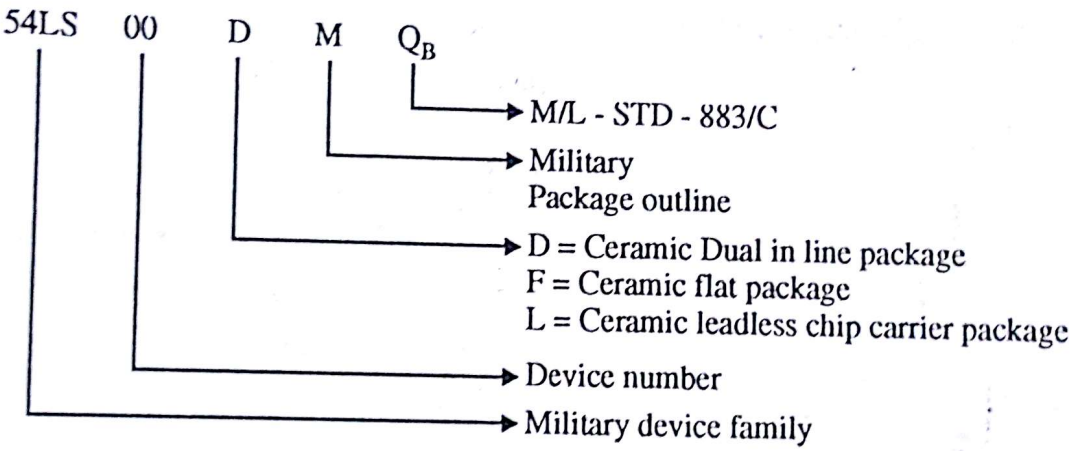
When output is switching from the LOW state to the HIGH state ; $Q_3$ changes state from OFF to ON and $Q_4$ from ON to OFF. Previously, $Q_4$ is in saturation condition, therefore it will take more time to change the state compared to $Q_3$. Therefore, for a very short duration (typ 2 η sec) ; both the transistors are ON and draws large current at time $t_1$, from the power supply. The effect is more severe if load capacitance C is more. These spikes can cause serious malfunctions during switching transition.

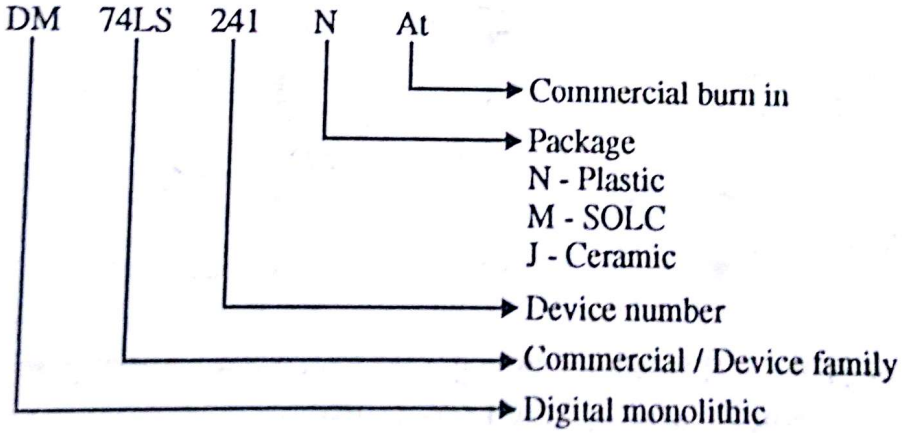
Scanned by CamScanner

**Remedy :** One has to use small radio-frequency, capacitors connected from $V_{CC}$ to ground. These capacitors will essentially ***short out*** these high frequency spikes. This is called ***Power supply decoupling*** The typical value of capacitor is 0.1 to 0.001 μF disc type.

## 13.10 Chip Number and Meaning :




## 13.11 Open Collector :

In this type of gates totem pole output stage is removed. Only transistor $Q_2$ is present and its collector is left open as shown in Fig. 13.43.

Transistor $Q_1$, diode D and resistor $R_4$ is removed from the standard TTL circuit. Normally one should connect external resistance at collector. This resistance is called ***pull up resistance***.

Functionally it is same as TTL.

Scanned by CamScanner

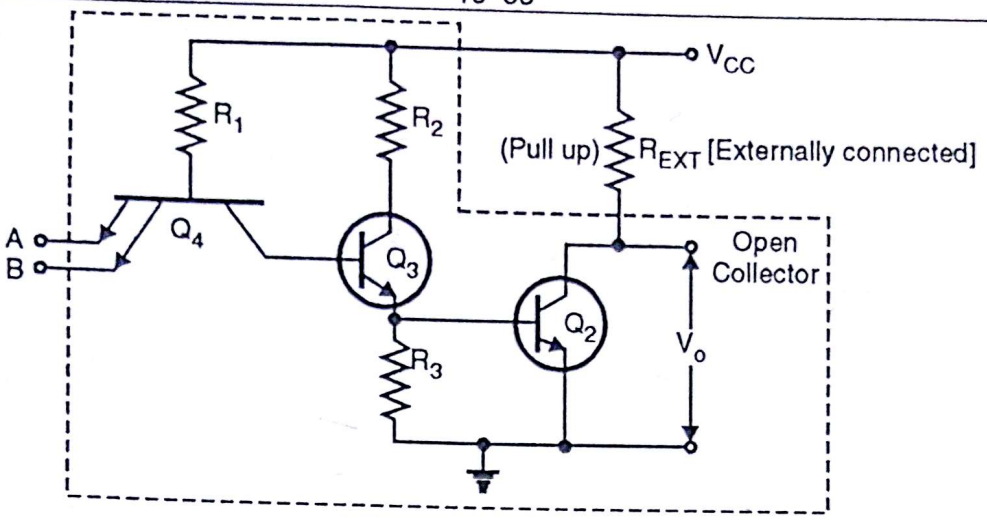



**Fig. 13.43**

When A = B = 1, $Q_4$ works in inverse mode, therefore $Q_3$ is ON, therefore $Q_2$ is ON. $Q_2$ is in saturation and pulls output voltage to ground.

When A = B = 0 or either the input A or B is 0, $Q_4$ works in normal mode. Therefore $Q_3$ is OFF therefore $Q_2$ is OFF and acts as open circuit.

Choice of pull up resistance depends upon choice of designer, but one should not cross $I_{c(max.)}$ of transistor $Q_2$. Normally resistance value selected in the range of 1 kΩ to 5 kΩ for fanout of approx. 7.

But if we want to calculate value of R, then we have to perform some calculation. Suppose this device is suppose to drive 5 TTL gates (Standard TTL) then what will be the value of R ?

5 TTL load we have to drive, therefore fan-out = 5.

$$\text{fan-out} = \frac{I_{OL}}{I_{IL}}$$

$I_{IL}$ for standard TTL logic is 1.6 mAmp.

$$\therefore \quad \text{fan-out} = \frac{I_{OL}}{I_{IL}}$$

$$5 = \frac{I_{OL}}{1.6\ \text{mA}} \; ; \; \therefore \; I_{OL} = 8\ \text{mA}.$$

Now apply KVL $\quad V_{CC} = R_{EXT} \times I_{OL} + 0.2\ V$

$$\frac{5 - 0.2\ V}{8\ \text{mA}} = R_{EXT} = \frac{4.8}{8\ \text{mA}} = 600\ \Omega$$

The disadvantages of open collector gate is their slow switching speed, because normally pull up resistance has value in kΩ, which results in relatively long time constant when it is multiplied by stray output capacitance.

But the main advantage of open collector gate is for ***Wire ANDing***. Before we discuss about wire ANDing, let's see current parameter for open collector gate.

**$I_{CEX}$ (output leakage current) :** The current flowing into an open collector output when output is at logic high state. This test checks reverse breakdown of the output transistor.

Scanned by CamScanner

**$I_o$ (OFF) (OFF state Output current)** : The current flowing in an output with input conditions such that output switching element is in off state.

> **Note :** This parameter is usually specified for open collector outputs intended to drive device other than logic circuit, such as displays.

## 13.12 Wire ANDing :

It means tieing output of gates to obtain AND function. This should be used to decrease two levels of logic and their associated delay. Here you may ask question that totem pole output has following advantages such as :

*(1) Low output impedance in both states.*

*(2) Drives capacitive load.*

*(3) Low noise susceptibility (high noise immunity).*

*(4) High speed.*

Then why TTL is not preferred for wire ANDing ? Before coming to the answer, let's first perform wire ANDing in DTL gate.

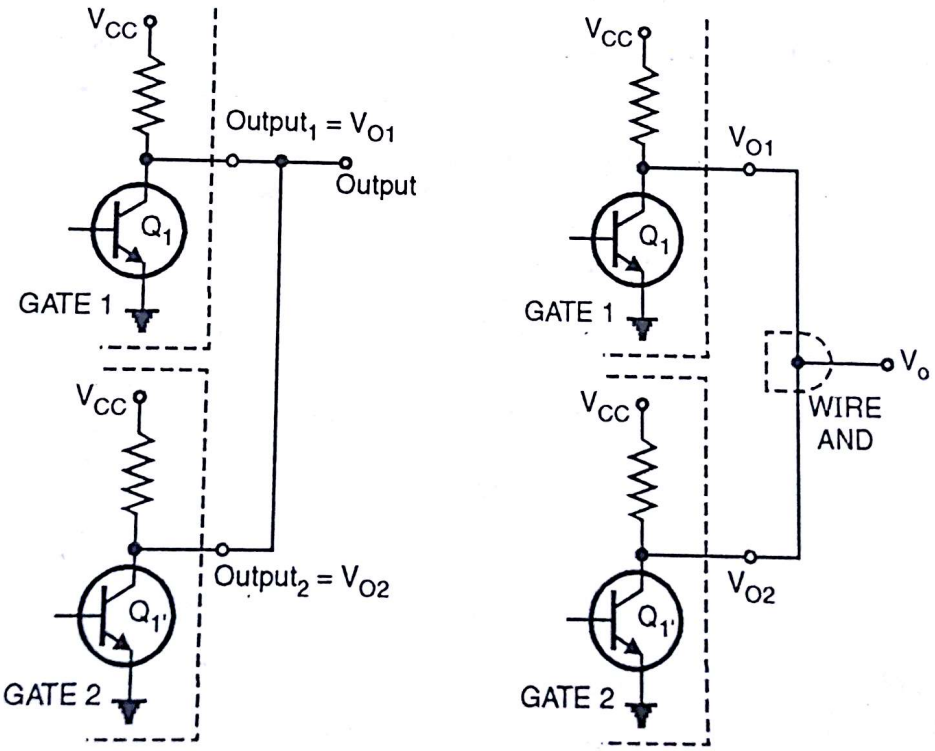



**Fig. 13.44 : Wire ANDing using DTL gate**

Fig. 13.44, Wire ANDing using DTL gates. Output of gate 1 and gate 2, i.e., $V_{O1}$ and $V_{O2}$ is tied together to obtain $V_{O(final)}$.

Actually if we observe, by tieing output of DTL gate, output stage transistors $Q_1$ and $Q_1'$ are coming in parallel as shown in Fig. 13.45.

Scanned by CamScanner

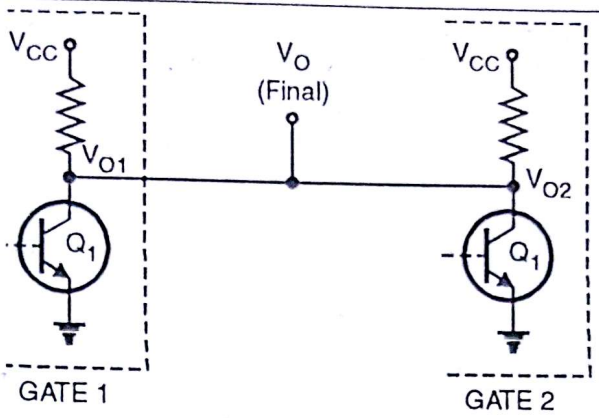



**Fig. 13.45**

## Working :

**Case I :** $V_{o1} = 0 \quad V_{o2} = 0$

means $Q_1$ is ON as $V_{o1}$ is 0, $Q_1'$ is also ON as $V_{o2} = 0$.

$\therefore V_{o\,(final)} = 0$

$$\boxed{V_{O1} = 0\ V_{o2} = 0 \Rightarrow V_{o(final)} = 0}$$

**Case II :** $V_{o1} = 0 \quad V_{o2} = 1$

Means $Q_1$ is ON as $V_{o1}$ is 0, $Q_1'$ is OFF ($Q'_1$ acts as open switch as) $V_{o2} = 1$. But as $Q_1$ and $Q_1'$ are in parallel, $V_{o(final)}$ with respect to ground will be 0 i.e. $V_{CE\,(sat)} \cong 0.2V$ = logic 0.

$$\boxed{V_{o1} = 0\ V_{o2} = 1 \Rightarrow V_{o(final)} = 0}$$

**Case III :** $V_{o1} = 1 \qquad V_{o2} = 0$

Means $Q_1$ is OFF (acts as open switch) as $V_{o1}$ is 1, $Q'_1$ is ON as $V_{o2} = 0$. But both transistors are in parallel so that $V_{o(final)}$ with respect to ground is '0' or $V_{CE(sat)} \cong 0.2V$. $\boxed{V_{o1} = 1\ V_{o2} = 0 \Rightarrow V_{O(final)} = 0}$

**Case IV :** $V_{o1} = 1 \qquad V_{o2} = 1$

Means $Q_1$ and $Q'_1$ both are OFF, $\therefore$ acts as open circuit.

$\therefore V_{o\,(final)} = 1$. $\boxed{V_{o1} = 1\ V_{o2} = 1 \Rightarrow V_{O(final)} = 1}$

Now let's combine all results

| $V_{o1}$ | $V_{o2}$ | $V_{o(final)}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Conclusion :** $V_{o(final)}$ output matches with AND gate truth table.

It means that we get the same output if we give $V_{O1}$ and $V_{O2}$

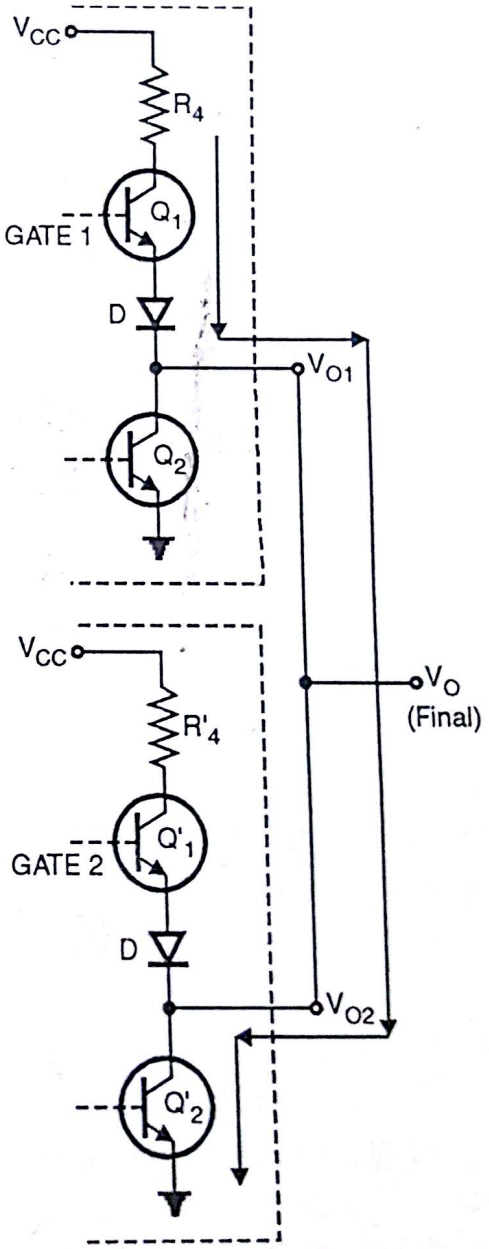



**Fig. 13.46**

Scanned by CamScanner

to an AND gate. Just tieing output together can give us output which is equal to truth table of AND, eventhough AND gate is not present is called wire ANDing. Therefore we are assuming that virtually AND gate is present and this is shown in Fig. 13.44 (dotted AND gate.)

Now let's try to implement wire ANDing in TTL family gate.

Let's analyse the working.

When $V_{O1}$ and $V_{O2}$ are at 0 logic,transistor $Q_2$ and $Q'_2$ are ON. $\therefore V_{O(final)} = 0$.

When $V_{O1} = V_{O2} = 1$, transistor $Q_2$ and $Q'_2$ are OFF and transistor $Q_1$ and $Q'_1$ conducts

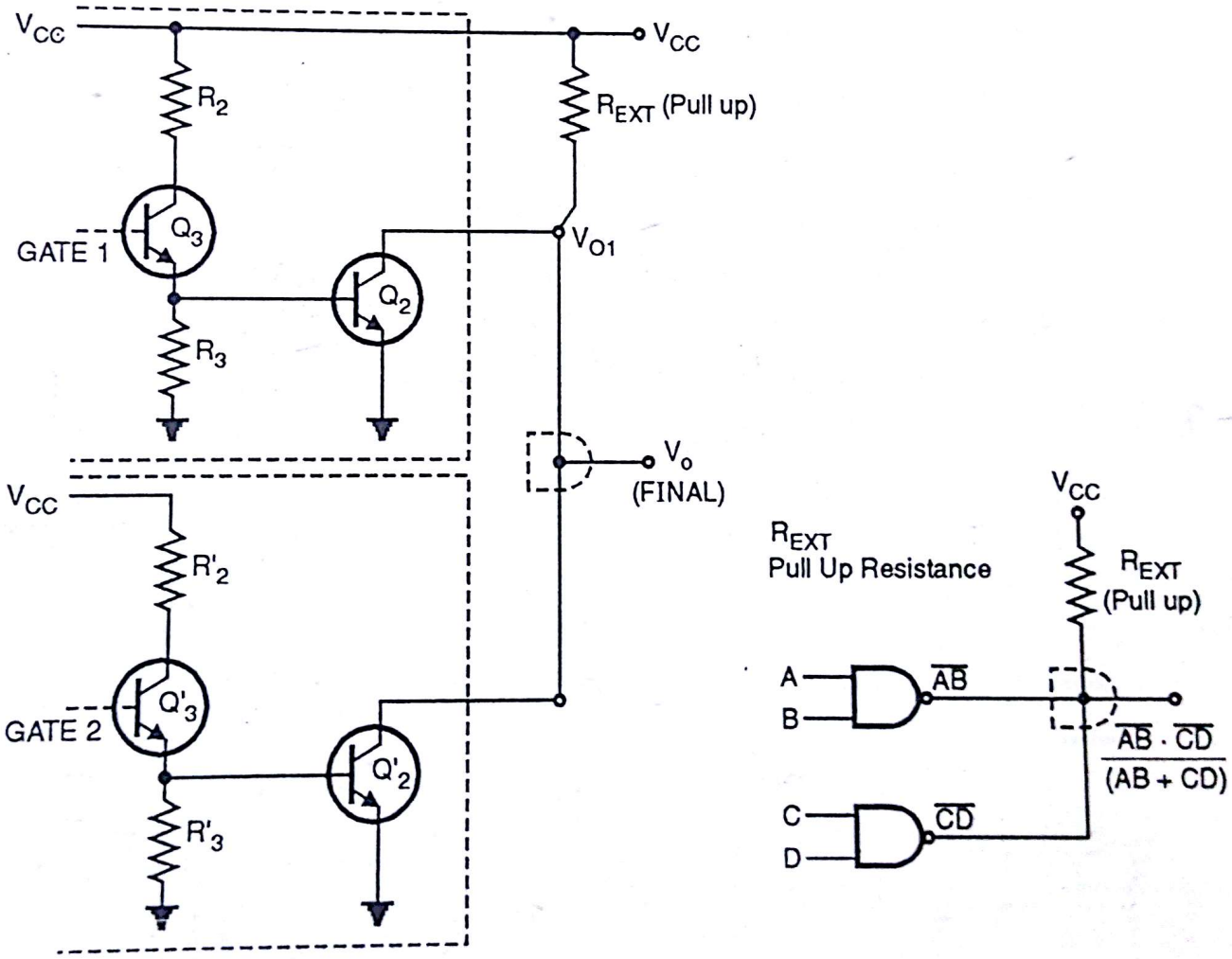
$\therefore V_{O(final)} = 1$. Let's consider $V_{O1} = 1$ and $V_{O2} = 0$.

Therefore transistor $Q_1$ and diode D is conducting as $V_{O1} = 1$

Transistor $Q'_2$ conducts in gate 2 as $V_{O2} = 0$. Because of this, current I will flow from $V_{CC}$, $R_4$, $Q_1$, D and $Q_2$. This current I is large and it may damage the gate. If the gate is not damaged then output is unpredictable. Therefore this type of wire ANDing should be prohibited.

**Remember :** In TTL logic output should not be short. Otherwise gate may get damaged.

But now in TTL if we have open collector output we can use wire ANDing function let's see the Fig. 13.47 for the same.




**(a) Wire ANDing using open collector** **(b) Wire ANDing using logical symbol**

**Fig. 13.47**

As shown in Fig. 13.47 (a), TTL NAND gate output is tied together with external pull up resistance. Choice of the resistance depends upon designer.

Observing diagram carefully we will come to know that transistor $Q_2$ and $Q'_2$ are coming in parallel as shown in Fig. 13.48, like DTL gate.

Scanned by CamScanner

Therefore working is same as what we have seen in DTL. Thus provides wired ANDing as shown in Fig. 13.47 (b). $V_{O1} = \overline{AB}$, $V_{O2} = \overline{CD}$, then $V_{O(final)} = \overline{AB} \cdot \overline{CD} = \overline{AB + CD}$

↑ AND

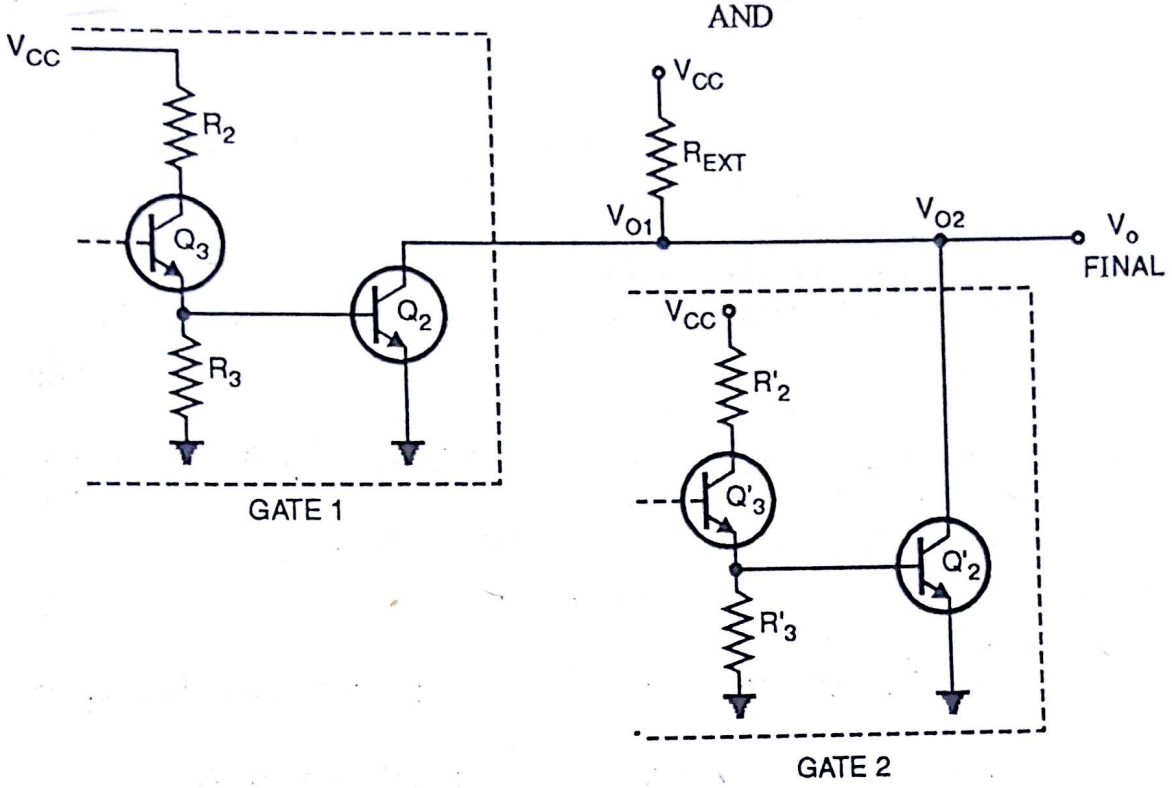
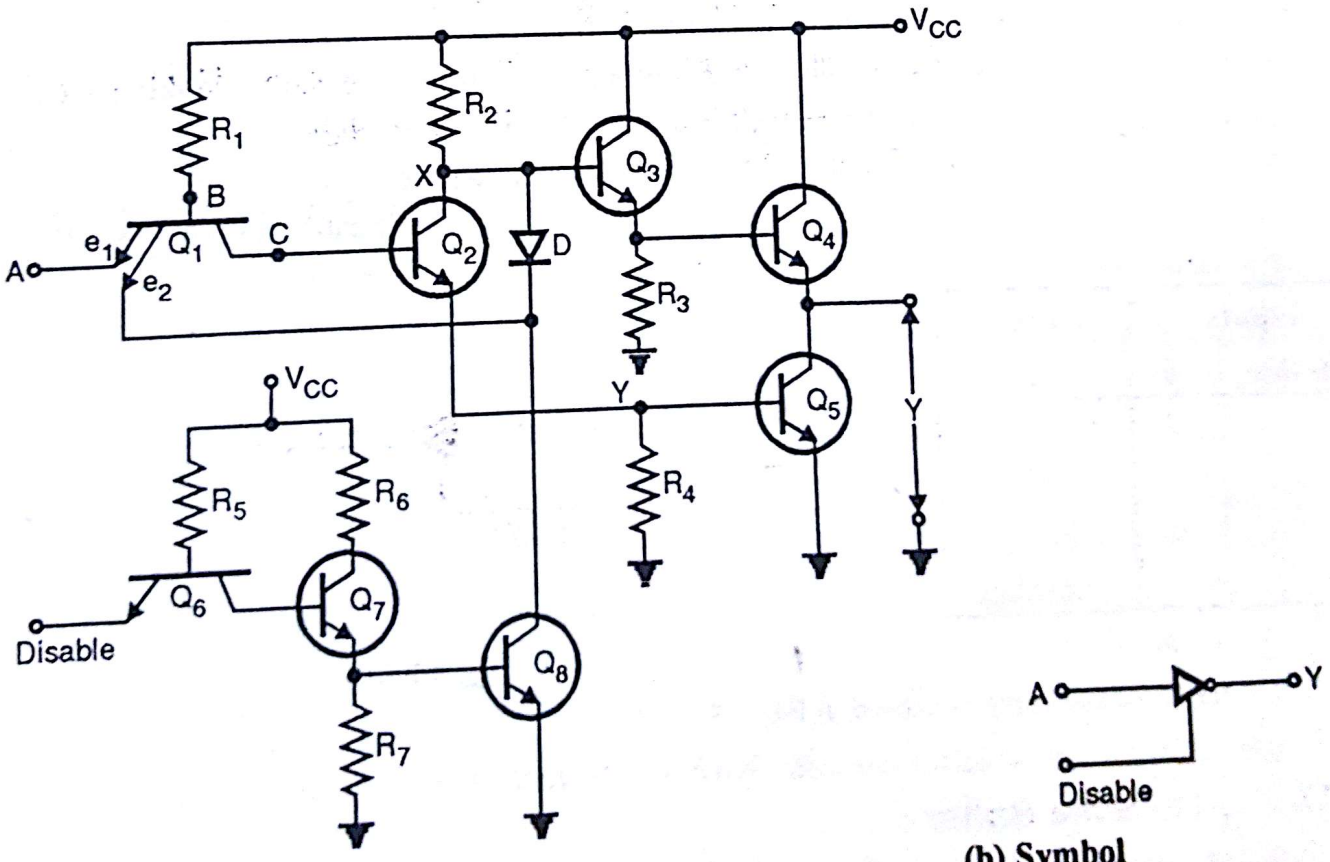



Fig. 13.48

## 13.13 Tri-state TTL Device :




(a) Tri-state inverter

(b) Symbol

Fig. 13.49

Scanned by CamScanner

We have seen that we cannot wire AND outputs of TTL gate, because this will lead us to high current flow in the output stage and gate may destroy. This has led us to Tri–state (three state) TTL logic. Using tri–state gates one can connect totem pole output directly without destroying any device. Basic reason behind not using open collector gate is the low speed. Fig. 13.49 shows circuit of tri–state inverter.

**Working :**

**Case I :** LOW disable input

When disable = 0, $Q_6$ conducts and therefore $Q_7$ is OFF. As $Q_7$ is OFF. $Q_8$ is also OFF. This forces second emitter $e_2$ of $Q_1$ transistor and cathode of diode D to float.

(a) If A = 0.

As A = 0, transistor $Q_1$ works in normal mode,

Therefore $Q_2$ is OFF, giving high logic at point X. $VX \cong V_{CC} - V_{BE3} - V_{BE4}$ = Logic 1.

A = 0 , Y = 1

(b) If A = 1

Transistor $Q_1$ works in inverse mode, therefore $Q_2$ conducts. This forces potential at point Y sufficient to drive $Q_5$ transistor in saturation. Therefore $V_X = V_Y + V_{CE2\ (Sat)} = 0.2 + 0.7 = 0.9V$. This voltage is not sufficient to drive transistor $Q_3$ and $Q_4$ in conduction.

Therefore $Q_3 = Q_4$ = OFF. As $Q_5$ is in saturation, Output Y = 0.

A = 1, Y = 0

**Case II :** HIGH disable input.

When disable = 1, $Q_6$ is OFF. This turns ON $Q_7$. $Q_7$ in turn switches ON $Q_8$, now $Q_8$ is in saturation.

Therefore potential at X point is $V_X = V_D + V_{CE8\ (sat)}$, as diode D is forward biased.

$V_D$ = Forward drop of diode D

$V_{CE8\ (Sat)} = 0.2V$ as $Q_8$ is ON.

$V_X = 0.7 + 0.2 = 0.9V$. This is not sufficient to drive $Q_3$ and $Q_4$.

Potential at $V_Y$ is also less, therefore $Q_5$ is also OFF.

With $Q_4$ and $Q_5$ cutoff, the output Y is ***Floating***. Therefore, impedance looking back into Y approaches infinite. This state is called as ***High Impedance*** or ***Hi – Z*** condition.

This is also third state of TTL device. Therefore also called tri–state.

Disable = 1, A = X, Y = Hi – Z

Let's summarize the result.

| Inputs | | Output |
|---|---|---|
| Disable | A | Y |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | x | Hi – Z (floating) |

Table 7

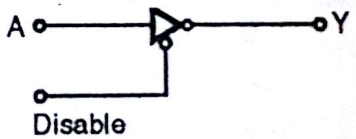



Fig. 13.50

The symbol for the same is shown in Fig. 13.49(b).

Disable terminal can be active low also. Symbol is as shown in Fig. 13.50.

## 13.13.1 Tri–State Buffer :

Buffer is a device used to increase driving capacity of gate. Boolean expression for output is Y = A. means whatever is the input that is the output. But with more output current.

Scanned by CamScanner

**Truth table**

| Inputs | | Output |
|---|---|---|
| Disable | A | Y |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | x | Hi – Z |

Fig. 13.51 (a)

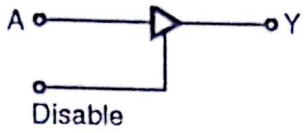



Fig. 13.51 (b)

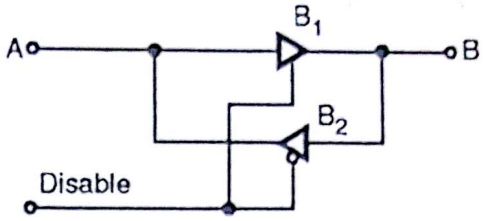



Fig. 13.51 (c)

The buffer we have seen just now is uni–directional buffer. We can also have bi–direction buffer. Bi-directional means data flow can be from either of the direction depending upon disable pin status.

As shown two tri–state buffers are connected parallely opposite.

**Working :**

(1) Disable = 0, $B_1$ buffer will be enable and $B_2$ is disabled so acts as open switch, therefore data flow is from A to B. Refer Fig. 13.52(a).

(2) Disable = 1, $B_2$ buffer is enabled and $B_1$ is disabled, so $B_1$ acts a open switch, therefore data flow is from B to A. Refer Fig. 13.52(b).

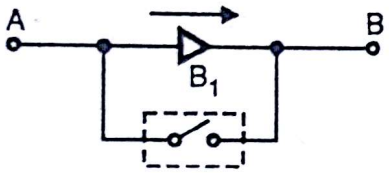



Fig. 13.52 (a)

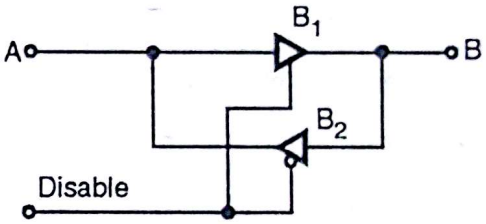



Fig. 13.52 (b)

**Application :**

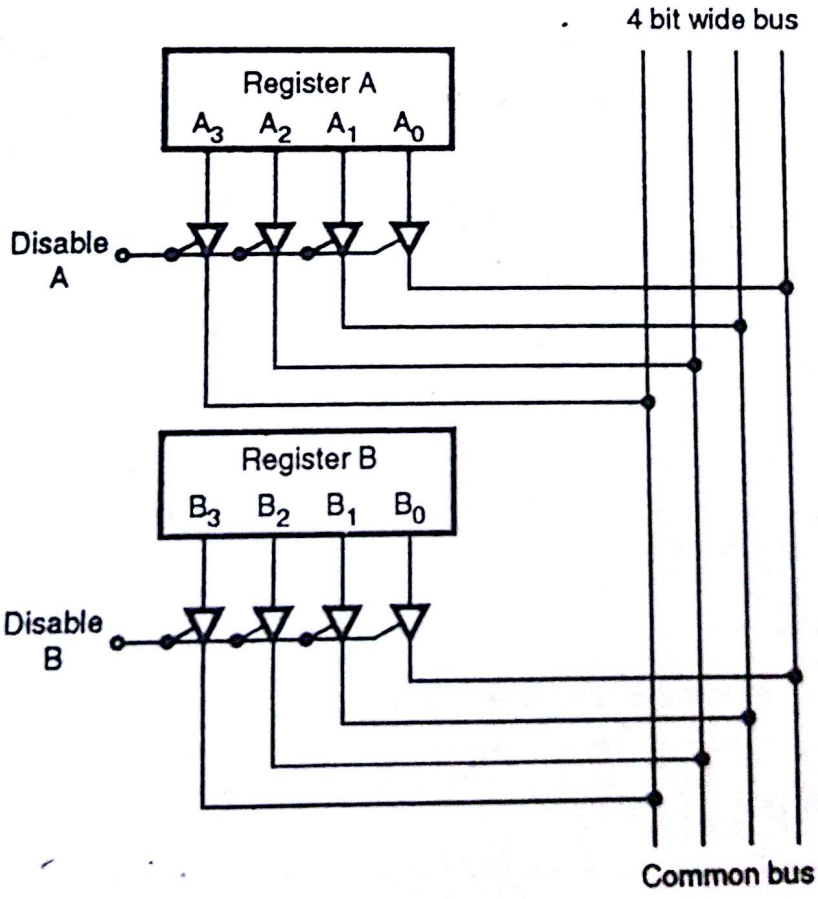



**Fig. 13.53 : Bus organised structure using tri-state buffer**

Scanned by CamScanner

Tri–state buffers are most widely used in computer or microprocessor based system. In microprocessor system there is single bus (8 bit or 10 bit wide) for carrying data from one chip to other chip. Outputs of all this chips are connected to a single bus only as shown in Fig. 13.53.

(1) Outputs $A_3$ $A_2$ $A_1$ $A_0$ of register A is connected to Bus through tri–state buffer. Outputs $B_3$ $B_2$ $B_1$ $B_0$ of register B is also connected to same bus.

(2) Initially Disable, A = Disable B = 1. Therefore all buffers are deactivated and acts as open switch. Therefore no data is present on bus.

(3) If we want output of register A, make disable A = 0 disable B = 1. Therefore tri–state buffers of A register will be enabled and for B it will be disabled. Therefore contents $A_3$ $A_2$ $A_1A_0$ will appear on Bus.

(4) If we want output of register B, make disable A = 1 disable B = 0. Therefore Tri – state buffers of B will get enabled and that of A are disabled. Contents $B_3$ $B_2$ $B_1$ $B_0$ will appear on bus.

(5) Precaution should be taken that disable A and disable B should not be low at same time.

The structure shown in Fig. 13.53 is called ***Bus organized structure.***

**Current parameter for tri–state buffer.**

**(1) $I_{oZ}$ – High impedance state output current :**

These tests guarantee that device will not excessively load a bus line when the device output is put into TRI – STATE mode.

**(2) $I_{oZH}$ (or $I_{sink}$) :**

The current flowing out of an output terminal when output is in tri–state condition, and the outputs of other chip interfaced to buffer output, produces LOW output.

**(3) $I_{oZL}$ (or $I_{source}$) :**

The current flowing out of an output terminal when output is in tri–state condition, And the outputs of other chip interfaced to buffer output, produces HIGH output.

## 13.14 Emitter Coupled Logic (ECL) :

In previously seen families majority was having drawback of slow speed. The basic reason was transistors are driven into saturation, resulting more propagation delay. ECL gates are having non saturating current switch. ECL is the fastest commercially available form of logic gate.

### Reasons for high speed :

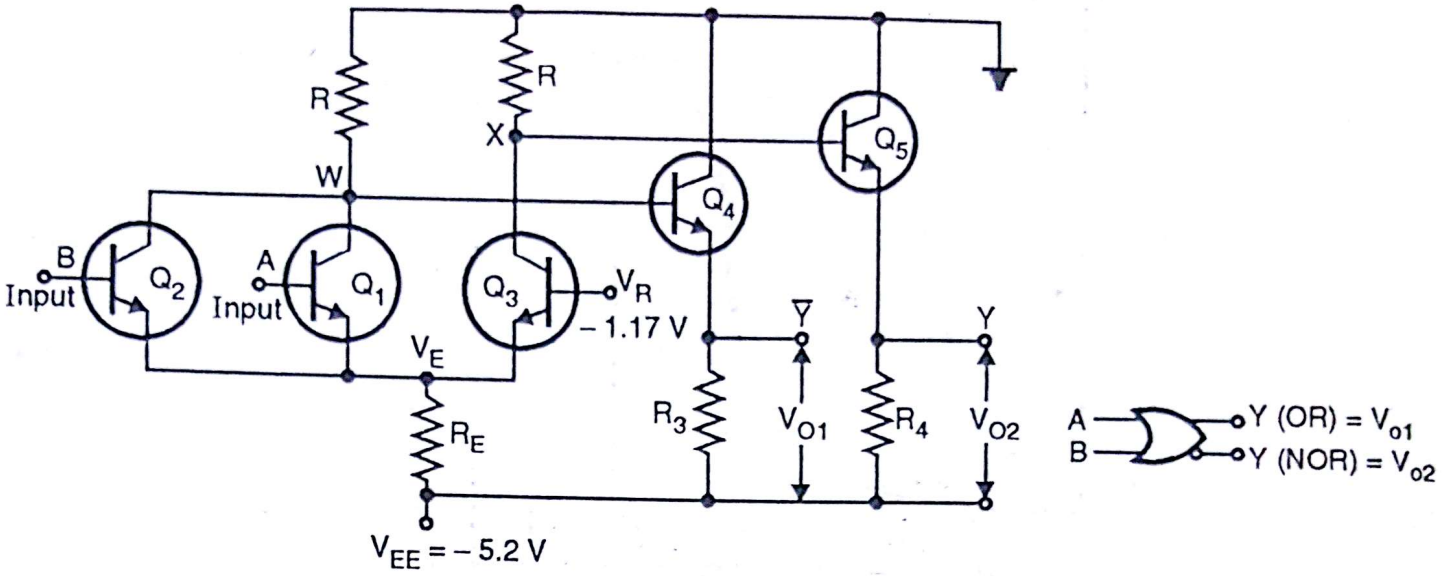
(1) Transistors are not driven into saturation, thus avoiding storage time delays, and by keeping the logic signal swings relatively small (about 0.8 V), thus reducing the time required to charge and discharge load and parasitic capacitances.

(2) Secondly in ECL, saturation is avoided by using BJT differential pair as a current switch.

Presently there are two popular forms of ECL. (1) ECL 10 K. (2) ECL 1000 K.

Scanned by CamScanner

The basic gate circuit of ECL 10 K family is as shown in Fig. 13.54 (a).




**(a) Basic ECL gate** **(b) Symbol**

**Fig. 13.54**

ECL gate is having differential amplifier stage. Output of differential amplifier given to emitter follower circuit. This provides low output impedance and also buffering. Two outputs are taken $V_{O1}$ and $V_{O2}$ matches with NOR and OR gate respectively.

## Working :

(1) When either of the input A or B is high, $Q_1$ or $Q_2$ conducts and $Q_3$ is cutoff. Hence there is high level at X pt, which will drive $Q_5$ and output Y = 1. Therefore it is OR operation $Y = A + B$

Correspondingly potential at w point will be low, therefore $Q_4$ will not conduct therefore output $\bar{Y} = 0$. Therefore NOR operation $Y = \overline{A + B}$

(2) When A = B = 0, $Q_1$ $Q_2$ are cutoff and $Q_3$ conducts. Potential at x point is low, therefore $Q_5$ is OFF and therefore Y = 0. OR operation Y = A + B.

Potential at point w is high, therefore $Q_4$ conducts and output $\bar{Y} = 1$, NOR operation $\bar{Y} = \overline{A + B}$

At the output stage we have emitter follower.

Purpose of using emitter follower :

(1) They shift the level of the output signals by one $V_{BE}$ drop.

(2) It provides low output impedance and with the large output currents required for charging load capacitances.

## OR Transfer Curve :

Fig. 13.55 shows OR transfer characteristic.

Scanned by CamScanner

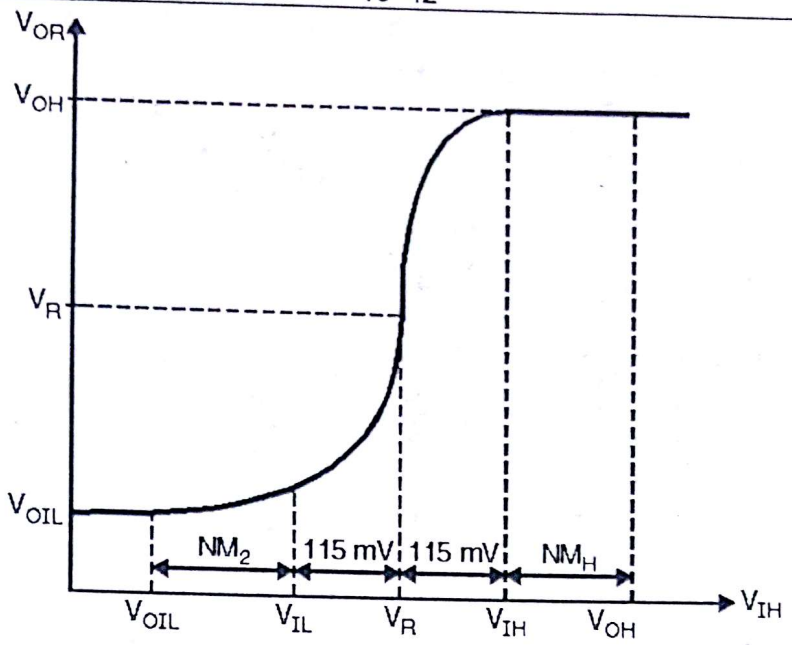



**Fig. 13.55**

**NOR Transfer Curve :**

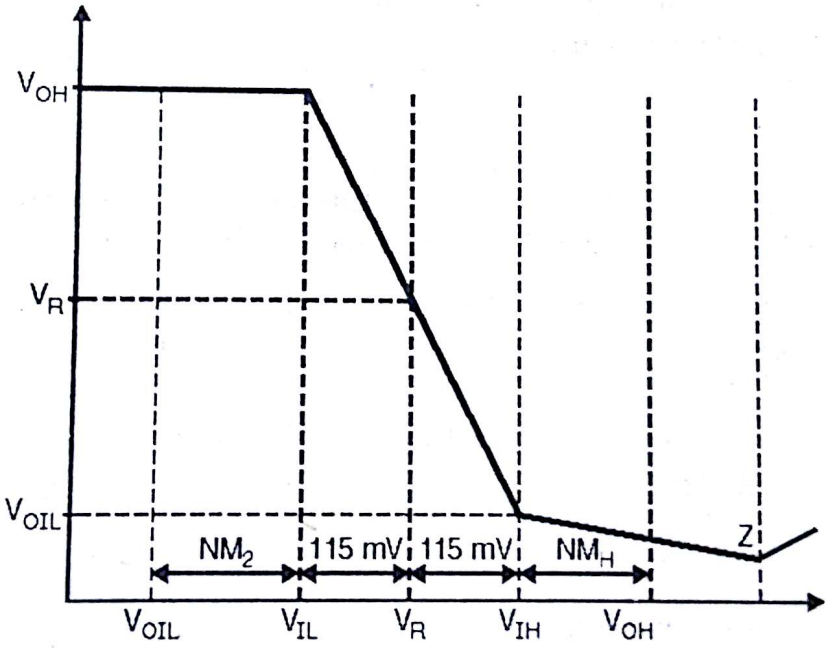



**Fig. 13.56**

ECL 10 K series is slightly slower; it features.

- Gate propagation delay of 2 ns.
- Power dissipation 25 mw for delay – power product of 50 pJ.
- Easier to use, because rise and fall times of the pulse are made longer, thus reducing signal coupling and crosstalk, between adjacent signal lines.
- Speed is of 3.5 n sec.

Scanned by CamScanner

## ECL 100 K Series Features :

- Gate delay of 0.75 n sec.
- Power dissipation 40 mw/gate, for delay–power product of 30 pJ.
- Speed is of 1 n sec.

## Specifications :

For MCEL 10,000 at 25° C following are the worst case values for $V_{IL}$, $V_{IH}$, $V_{OL}$ and $V_{OH}$.

$V_{IL\ (max)} = -1.475$ V  $V_{IH\ (min)} = -1.105$ V

$V_{OL\ (max)} = -1.630$ V  $V_{OH\ (max)} = -0.980$ V

$\therefore$ $NM_L = 0.155$ V  $NM_H = 0.125$ V

## Fan Out :

Output resistance of ECL gate is very small. DC from out is as high as 90 and ac fanout is limited to 10.

## Power Dissipation :

Supply current and gate power dissipation of unterminated ECL remain relatively constant independent of logic state of the gate. Therefore there are no voltage spikes introduced on the supply line.

**Table 8**

| Families | | Propagation delay (ns) | Power Per gate (mw) | Speed–power Product (pJ) |
|---|---|---|---|---|
| | MECL 10 K H | 1 | 25 | 25 |
| MECL 10 K | 10100, 10500 series | 2 | 25 | 50 |
| | 10200, 10600 series | 1.5 | 25 | 37 |
| | MECL VI | 1 | 60 | 60 |
| | AS TTL | 1.7 | 8 | 13.6 |
| | ALS TTL | 4 | 1.2 | 4.8 |
| | HCMOS | 8 | 0.17 | 1.4 |

The input impedance of circuit is very high, because of differential amplifier at input side.

## Advantages :

(1) Fastest Logic.
(2) High fan out due to high $Z_{in}$ and low $Z_{out}$, typically 25.
(3) Excellent speed power product.
(4) Logic flexibility by providing 2 outputs, OR and NOR.

## Disadvantages :

(1) Dissipates large stand by power.
(2) High speed generates voltage and current transients.
(3) Limited logic swing. More sensitive to environmental noise.

Scanned by CamScanner

## 13.15 MOS Logic :

MOSFET (Metal oxide semiconductor field effect transistor) can be subgroup into two types.

*(1) Depletion MOSFET.*

*(2) Enhancement MOSFET.*

Enhancement MOSFET is most widely used.

In that we have PMOS and NMOs. PMOS technology was dominating one, but because NMOS devices can be made smaller and thus operate faster, and because, NMOS requires lower supply voltages than PMOS, NMOS is most widely used. But PMOS is not outdated because it is used in complementary MOS or CMOS circuit.

## 13.16 N–MOS Inverter :

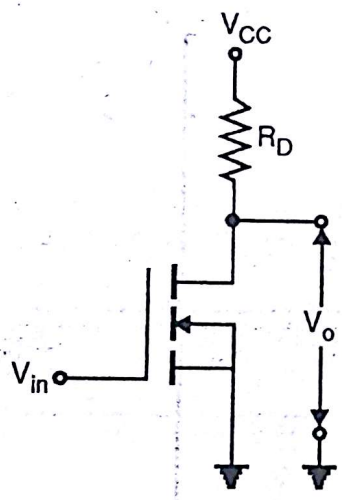



Fig. 13.57 (a)

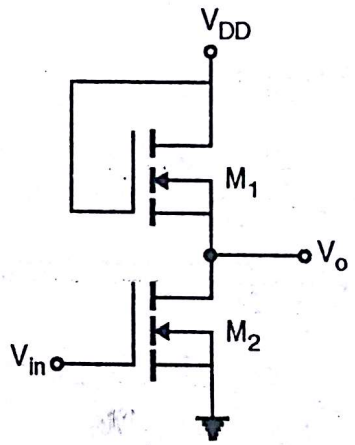



Fig. 13.57 (b)

The Fig. 13.57 (a) shows MOSFET inverter with passive load. $R_D$ is adjusted in such a way that when

$V_{in} = 0 \rightarrow$ MOSFET OFF.

$V_{in} = 1 \rightarrow$ MOSFET ON, providing

Sufficient $I_{DS}$ (drain to source current).

Thus MOSFET switches between saturation and cutoff (Refer Fig. 13.57 (b) ).

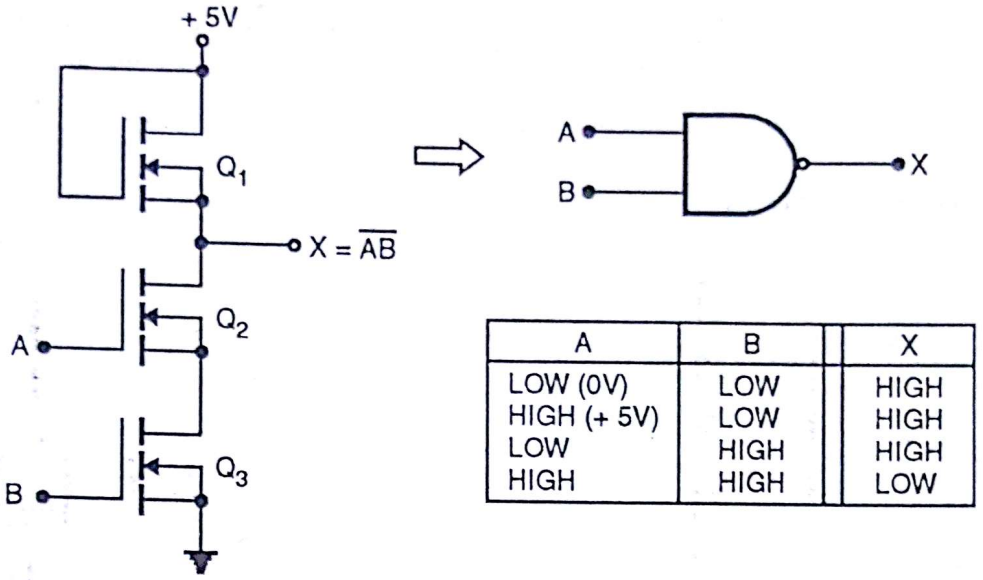
We can also connect ***active load***, shown in Fig. 13.57 (b).

With MOSFET technology, it is easier to fabricate a MOSFET than a resistor. For this, another MOSFET $M_1$ is used as load for $M_2$. Gate of $M_1$ is connected to supply, therefore it is always ON and behaves as resistor. $M_1$ is designed in such a way that it should act as resistor. $M_2$ should act as switch.

The advantage of using MOSFET switch and load is, MOFET requires less space in comparison with TTL (bipolar device.) therefore higher package density.

Scanned by CamScanner

## 13.17 N–MOS NAND Gate :




| A | B | X |
|---|---|---|
| LOW (0V) | LOW | HIGH |
| HIGH (+ 5V) | LOW | HIGH |
| LOW | HIGH | HIGH |
| HIGH | HIGH | LOW |

**Fig. 13.58 : N–MOS NAND gate**

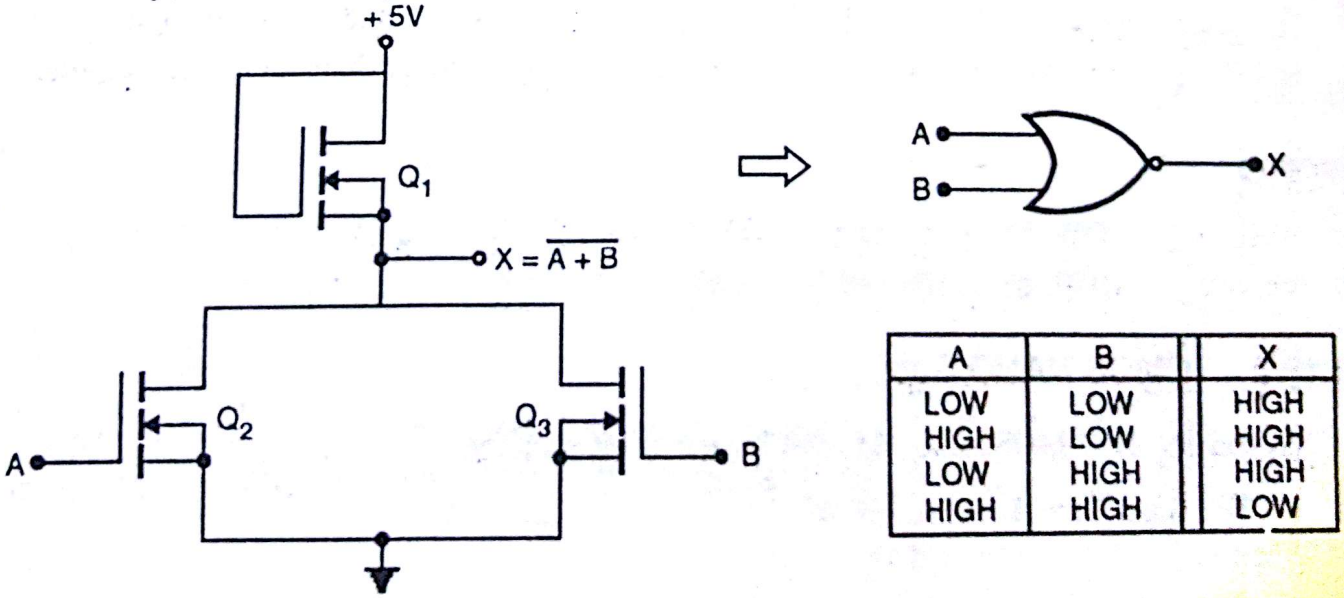
$Q_1$ is acting as load resistance. $Q_2$ and $Q_3$ are used as switches.

1. A = B = 0 or either A = 0 or B = 0.
   In above cases, respective transistors, i.e. $Q_2$ for A and $Q_3$ for B, will be OFF. Thereby presenting high resistance from the output terminal to ground, so that output X is HIGH.
2. A = B = 1 (or + 5V)
   $Q_2$ and $Q_3$, both will be ON ; offering low impedance. The output X is LOW.

## 13.18 N–MOS NOR Gate :

$Q_1$ acts as load resistance.$Q_2$ and $Q_3$ will act as simple switch ; connected in parallel.

1. A = B = 1 or either A = 1 or B = 1.
   In above cases, respective transistor will be ON, offering LOW impedance and forcing output X = 0.
2. A = B = 0.
   $Q_2$ and $Q_3$, both will be OFF. Therefore output X goes HIGH.
   Clearly, this is NOR operation.




| A | B | X |
|---|---|---|
| LOW | LOW | HIGH |
| HIGH | LOW | HIGH |
| LOW | HIGH | HIGH |
| HIGH | HIGH | LOW |

**Fig. 13.59 : N-MOS NOR gate**

Scanned by CamScanner

## 13.19 Complementary MOS Logic (CMOS) :

CMOS family uses both; P and N channel MOSFET. This leads to several advantages over the P-MOS and N-MOS families. Generally, CMOS is faster and consumes less power than other MOS families. CMOS has lower package density and slightly complex IC fabrication process.

### 13.19.1 CMOS Inverter :

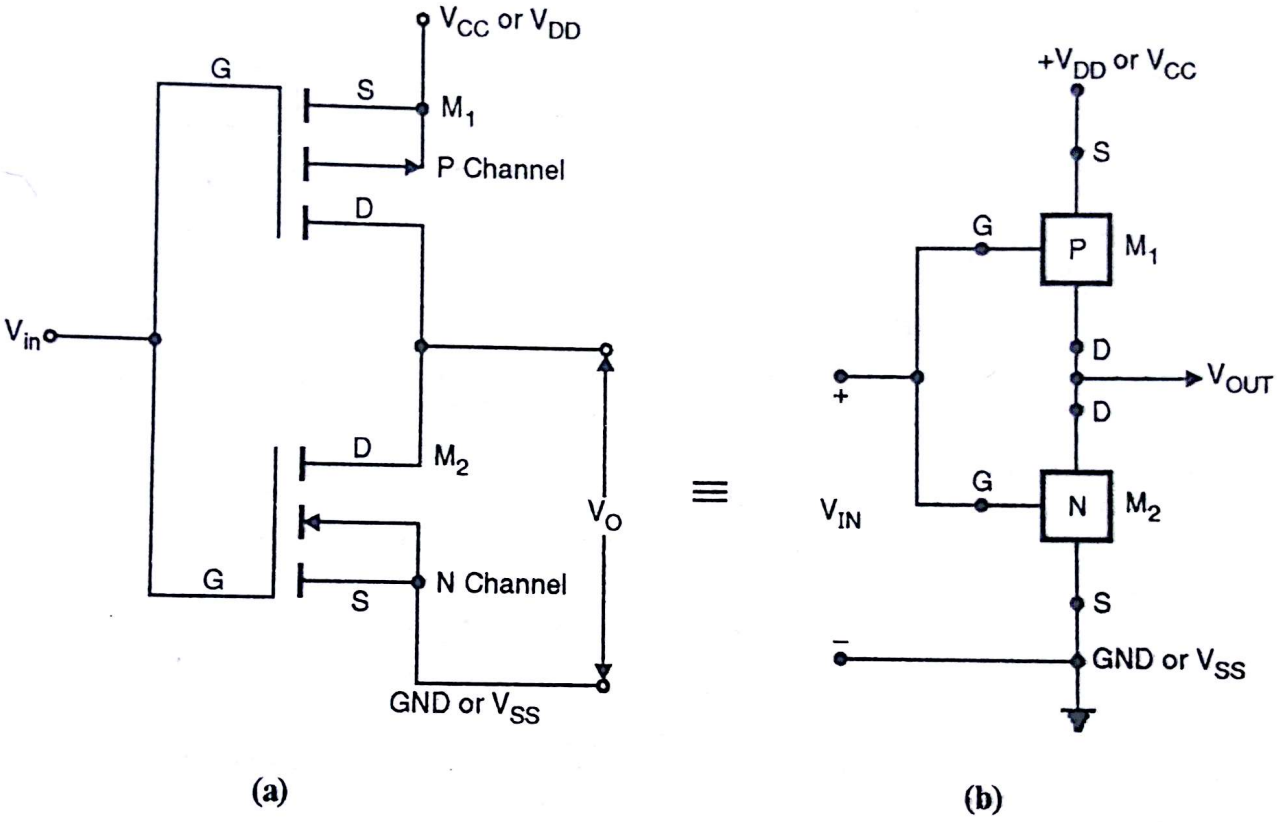



**Fig. 13.60 : CMOS inverter**

The basic CMOS circuit is the inverter, shown in Fig. 13.60. It consists of two MOS enhancement mode transistors, the upper is P channel type, the lower N channel type.

The power supplies for CMOS are call $V_{DD}$ and $V_{SS}$ or $V_{CC}$ and ground depending upon manufacturer. $V_{DD}$ and $V_{SS}$ are carryover from conventional MOS circuits and stand for drain and source supplies.

**Working :**

(1) When $V_{in}$ = LOW, $M_2$ is off but $M_1$ is ON. It means that $V_o$ is equal to $V_{CC}$ or HIGH logic.
(2) When $V_{in}$ = HIGH, $M_2$ is ON and $M_1$ is OFF. In this case $V_o \equiv V_{SS} \equiv$ GND.

### 13.19.2 CMOS NAND Gate :

$M_1$ and $M_2$ are P-MOSFETS. $M_3$ and $M_4$ are N-MOSFETS.

1. A = B = 0 or either A = 0 or B = 0.
   When A = 0, $M_1$ = ON and $M_3$ = OFF
   when B = 0, $M_2$ = ON and $M_4$ = OFF.

Scanned by CamScanner

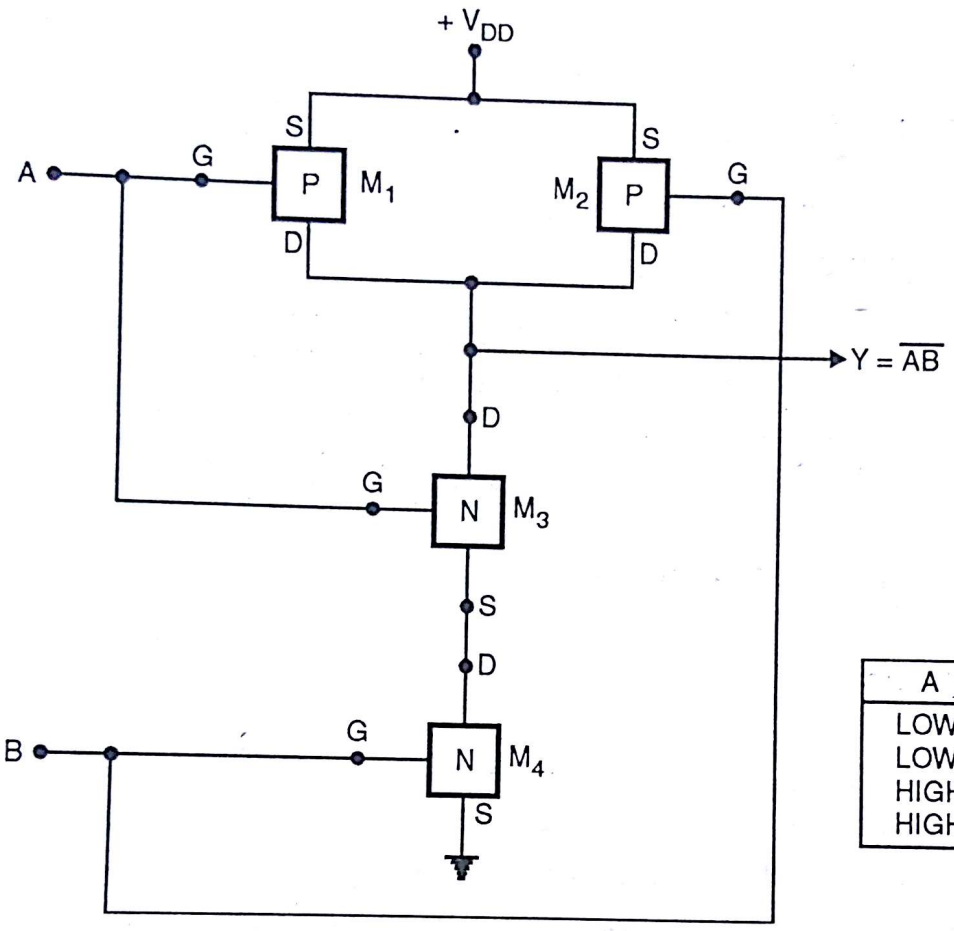
Therefore, in all the cases either $M_3$ or $M_4$ or both are OFF ; offering HIGH impedance.

Therefore output Y = HIGH.

2. A = B = 1.

$M_1$ and $M_2$ are OFF. $M_3$ and $M_4$ are ON; offering low resistance from output terminal to ground.

Output y = LOW.




| A | B | Y |
|---|---|---|
| LOW | LOW | HIGH |
| LOW | HIGH | HIGH |
| HIGH | LOW | HIGH |
| HIGH | HIGH | LOW |

**Fig. 13.61 : CMOS NAND gate**

## 13.19.3 CMOS NOR Gate :

$M_1$ and $M_2$ are P-channel MOSFET in series. $M_3$ and $M_4$ are N channel MOSFETS in parallel.

When A = 0 → $M_1$ = ON and $M_3$ = OFF

A = 1 → $M_1$ = OFF and $M_3$ = ON

When B = 0 → $M_2$ = ON and $M_4$ = OFF

B = 1 → $M_2$ = OFF and $M_4$ = ON.

1. A = B = 0

$M_1 = M_2$ = ON and $M_3 = M_4$ = OFF. So we have HIGH impedance from output terminal to ground.

Therefore output Y = HIGH.

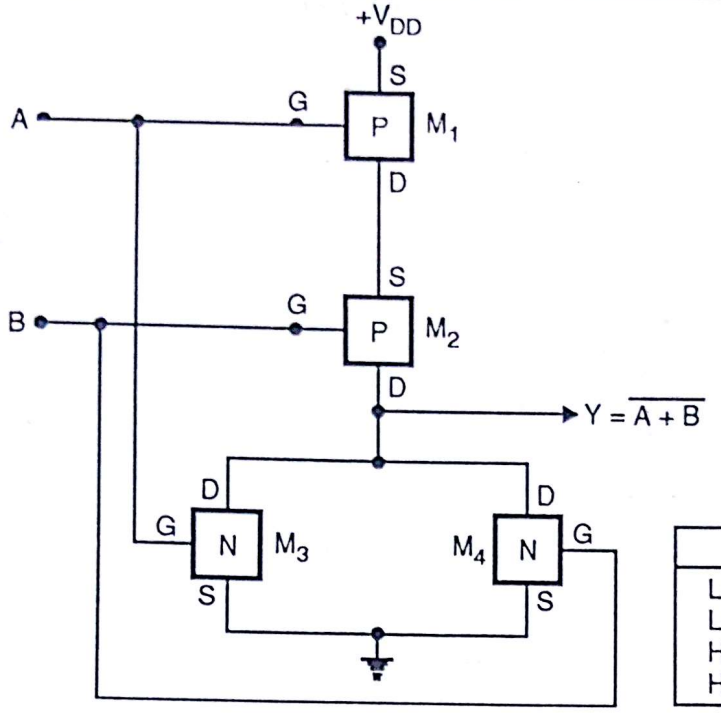
2. A = B = 1 OR either A = 1 or B = 1.

If A = 1, $M_3$ = ON and if B = 1, $M_4$ = ON. Due to this we get LOW impedance between output terminal and ground. Therefore Output Y = LOW.

Till this point we have studied CMOS INVERTER, NAND and NOR gates. Here we should ask ***Why one should go for CMOS ?***

This question led us to *CMOS characteristics.*

Scanned by CamScanner




| A | B | Y |
|---|---|---|
| LOW | LOW | HIGH |
| LOW | HIGH | LOW |
| HIGH | LOW | LOW |
| HIGH | HIGH | LOW |

**Fig. 13.62 : CMOS NOR gate**

## 13.20 CMOS Characteristics :

Let's talk about characteristic of ideal logic family. It should dissipate no power, have zero propagation delay, controlled rise and fall times, and have noise immunity equal to 50 % of the logic swing.

The properties of CMOS begin to approach these ideal characteristic.

First, CMOS dissipates low power. Typically static power dissipated is 10 η ω per gate which is due to flow of leakage currents. The active power depends upon power supply voltage, frequency, output load and input rise time, but typically gate dissipation at 1 MHz with 50 PF load is less than 10 mw.

Second, the propagation delays through CMOS are short through not quite zero. Depending upon power supply voltage, the delay through a typical gate is on the order of 25 η sec to 50 η sec.

Third, rise time and fall times are controlled, tending to ramp rather than step functions. Typically, rise time and fall times tend to be 20 to 40 % longer than propagation delay.

Last, but not least, the noise immunity approaches 50 %, being typically 45 % of the full logic swing.

Because of low power requirement, normally used for battery operated systems, like digital wrist watches and portable microcomputer as noise immunity is more normally used where environment is noisy.

Only problem is in comparison with TTL, CMOS is slower.

### 13.20.1 Transfer Characteristics :

As shown in Fig. 13.63 when input is low output voltage is in high state. As input voltage increases, output remains in high state untill a threshold is reached. As 0.45 $V_{cc}$ output switches to LOW state. Then any input greater than 0.45 $V_{cc}$, the output is at LOW state.

Scanned by CamScanner

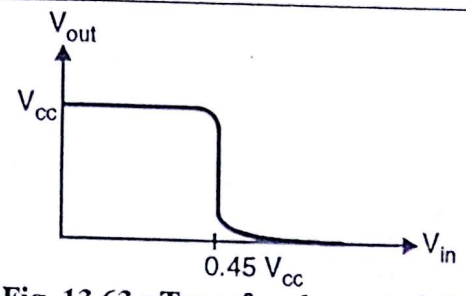



Fig. 13.63 : Transfer characteristic

MULUND COLLEGE OF COMMERCE LIBRARY BSc. IT 743 Mulund

### 13.20.2 Noise Immunity :

CMOS circuits have a typical noise immunity of 0.45 $V_{CC}$. This means that spurious input which is 0.45 $V_{CC}$ or less away from VCC or ground typically will not propogate through the system as an erroneous logic level. This doesn't mean that no signal at all will appear at output of first circuit. In fact these will be an output signal as a result of spurious input, but it will be reduced in amplitude. As this signal propogates through the system it will be attenuated even more by each circuit it passes through untill it finally disappears. Typically it will not change any signal to opposite logic level.

### 13.20.3 Unused Inputs OR Floating Inputs :

When TTL input is floating, it is equivalent to high input. But it is better to connect input either to ground or $V_{CC}$, to prevent floating leads to pickup stray noise in the environment.

But in CMOS unused inputs should not be left open. because of very high impedance ($\sim 10^{12}$ Ω), a floating input may drift back and forth between a '0' and '1' creating some very intriguing system problems. Therefore all unused inputs should be tied to $V_{CC}$ ground or another used input.

### 13.20.4 Power Supply Filtering :

Since CMOS can operate over a large range of power supply voltage (3V to 15 V), the filtering necessity is minimal. The minimum power supply voltage required will be determined by Maximum frequency of operation of the fastest element in the system. The filtering should be designed to keep the power supply somewhere between this minimum voltage and maximum rated voltage the parts can tolerate.

### 13.20.5 Electrostatic Discharge Prevention :

All CMOS devices are suceptible to damage by the discharge of electrostatic energy between the two pins. This sensitivity to static charge is due to the fact that gate input capacitance (5 PF typical) in parallel with very high resistance ($10^{12}\Omega$) leads itself to high input impedance and hence readily builds up electrostatic charge. If precaution is not taken then higher voltage, resulting from static discharge, is applied to the device, permanent damage like short to subtract, $V_{DD}$ pin, $V_{SS}$ pin, or output can occur.

Even if you remove or put CMOS device when power is ON, transient voltage caused by inductive kickback and other effects may exceed the gate voltage rating ; and can destroy internal insulating layer of MOSFET.

### 13.20.6 Protection Circuits :

(1) The series resistance of 200 Ω using $p^+$ diffusion helps limit the current when the input is subjected to a high voltage zap.

Scanned by CamScanner

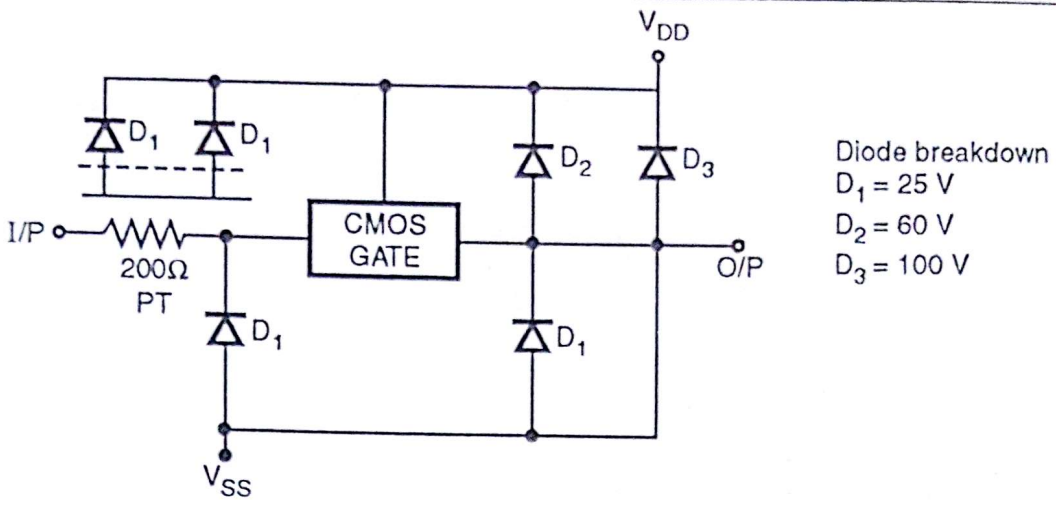



**Fig. 13.64**

Associated with this resistance is distributed diode network to $V_{DD}$ which protects against positive transients.

(2) Transmission gate with intrinsic diodes to protect against static discharge. Refer Fig. 13.65.

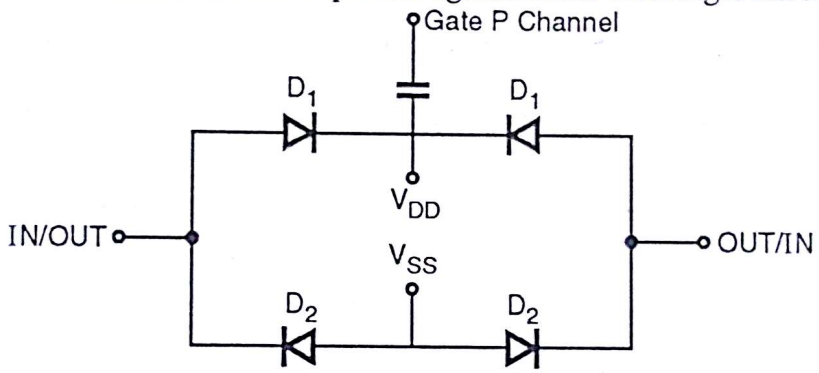



**Fig. 13.65**

No additional series resistors are used so that ON resistance of transmission gate is not affected.

### 13.20.7 Sourcing and Sinking :

When a standard CMOS driver output is low, the input current to CMOS load is only 1 micro amp i.e. sink current. Same way when driver output is high, CMOS driver sources 1 μA.

$I_{IL(max)} = -1\ \mu A$ $I_{IH(max.)} = 1\ \mu A$.

## 13.21 Handling Guide for CMOS Devices :

Extremely high static voltages generated in manufacturing environment can destroy even the optimally protected devices by reaching their threshold failure energy levels. For preventing such catastrophies. Simple precaution should be taken.

1. In handling unmounted chips, care should be taken to avoid differences in voltage potential between pins. Conductive carriers such as conductive foams or conductive rails should be used in transporting devices.
2. Soldering iron tips, metal parts of features and tools, and handling facilities should be grounded.
3. Devices should not be removed or plugged in when supply is ON.
4. Table tops should be covered with grounded conductive tops. Also test tables should be covered.
5. Do not exceed the maximum ratings specified by data sheet.

Scanned by CamScanner

6. All unused device inputs should be connected to $V_{CC}$ or GND.
7. All low impedance equipment (pulse generator etc.) should be connected to CMOS inputs only after the CMOS device is powered up. Similarly, this type of equipment should be disconnected before power is turned off.
8. All CMOS devices should be stored or transported in material that are anti static or conductive. CMOS device must not be inserted into conventional plastic *snow* Styrofoam, or plastic trays, but should be left in their original container untill ready for use.
9. Nylon or other static generating material should not come in contact with CMOS devices.
10. Double check test equipment setup for proper polarity of $V_{CC}$ and GND before conducting parametric or functional testing.
11. When lead straightening or hand soldering is necessary, provide ground straps for apparatus used and be sure that soldering iron tips are grounded.
12. If automatic handlers are being used, high level of static electricity may be generated by movement of the devices, the belts or printed circuit boards. Reduce static buildup by using ionized air blowers, anti static sprays and room humidifier.

## 13.22 CMOS Sub-Families :

The way we had TTL subfamilies, we do have subfamilies of CMOS.

### 13.22.1 4000/14000 CMOS :

The oldest CMOS series is the 4000 series first introduced by RCA. It has low power dissipation. If supports wide range of power supply voltage (3 to 15V). Slowest among all the series. It has low output current capabilities. This series is not compatible with TTL series.

### 13.22.2 74C Series :

This CMOS series is pin to pin compatible and functionally equivalent TTL device. Performance characteristics of this series is about the same as those of 4000 series.

### 13.22.3 74HC / HCT (High-Speed CMOS) :

This series is improved version of 74C. Ten times faster than 74C. This series is also pin compatible and functionally equivalent to TTL. 74HC devices are not electrically compatible with TTL, but 74HCT devices are. This series is most widely used.

### 13.22.4 74AC / ACT (Advanced CMOS) :

This series is functionally equivalent to TTL series, but not *pin compatible* with TTL series. Series is also referred as ACL i.e. advanced CMOS logic, because it has high noise immunity, less propagation delay and high speed.

### 13.22.5 74AHC (Advanced High Speed CMOS) :

The latest series of CMOS. Series offers natural migration path from the HC series to faster, lower-power, low drive applications. Series is three times faster than HC series.

Scanned by CamScanner

## 13.23 Logic Voltage Levels of CMOS and TTL :

Till this point we have studied CMOS and TTL families. Let's observe voltage logic levels of each series to conclude about noise immunity of each series.

**Table 9 : Input / output voltage levels (in volts) with $V_{DD} = V_{CC} = +5V$**

| Parameter | CMOS | | | | | | | TTL | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 4000B | 74HC | 74HCT | 74AC | 74ACT | 74AHC | 74AHCT | 74 | 74LS | 74AS | 74ALS |
| $V_{IH(min)}$ | 3.5 | 3.5 | 2.0 | 3.5 | 2.0 | 3.85 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| $V_{IL(max)}$ | 1.5 | 1.0 | 0.8 | 1.5 | 0.8 | 1.65 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 |
| $V_{OH(min)}$ | 4.95 | 4.9 | 4.9 | 4.9 | 4.9 | 4.4 | 3.15 | 2.4 | 2.7 | 2.7 | 2.7 |
| $V_{OL(max)}$ | 0.05 | 0.1 | 0.1 | 0.1 | 0.1 | 0.44 | 0.1 | 0.4 | 0.5 | 0.5 | 0.4 |
| $V_{NH}$ | 1.45 | 1.4 | 2.9 | 1.4 | 2.9 | 0.55 | 1.15 | 0.4 | 0.7 | 0.7 | 0.7 |
| $V_{NL}$ | 1.45 | 0.9 | 0.7 | 1.4 | 0.7 | 1.21 | 0.7 | 0.4 | 0.3 | 0.3 | 0.4 |

## 13.24 Comparing TTL AND CMOS Series :

**Table 10 : Digital IC Series Comparison***

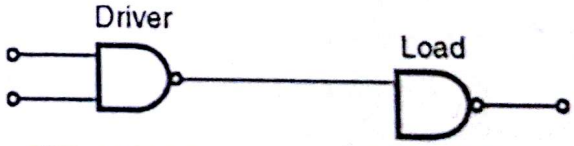
| Parameter | 4000B | 74HC / HCT | 74AC / ACT | 74AHC / T | 74 | 74LS | 74AS | 74ALS |
|---|---|---|---|---|---|---|---|---|
| Power dissipation per gate (mW) (a) Static | $1.0 \times 10^{-3}$ | $2.5 \times 10^{-3}$ | $5.0 \times 10^{-3}$ | $9.0 \times 10^{-5}$ | 10 | 2 | 8 | 1.2 |
| (b) At 100 kHz | 0.1 | 0.17 | 0.08 | 0.006 | 10 | 2 | 8 | 1.2 |
| Propagation delay(ns) | 50 | 8 | 4.7 | 3.7 | 9 | 9.5 | 1.7 | 4 |
| Speed-power (at 100 kHz) (pJ) | 5 | 1.4 | 0.37 | 0.02 | 90 | 19 | 13.6 | 4.8 |
| Maximum clock rate (MHz) | 12 | 40 | 100 | 130 | 35 | 45 | 200 | 70 |
| Worst-case noise margin (V) | 1.5 | 0.9 | 0.7 | 0.55 | 0.4 | 0.3 | 0.3 | 0.4 |

* All CMOS values are for $V_{DD} = 5V$. Parameters for specific devices may vary.

## 13.25 Interface :

Interface means the way driver device is connected to load device. Fig. 13.66 shows driver gate interfaced to load gate.

When driver and load gate both are of same family i.e. TTL or CMOS or PMOS, then one need not bother. But one has to consider voltage and current parameter when one family drieves another. TTL drives CMOS or vice versa, PMOS drives CMOS or vice versa and so on.




TTL / CMOS
PMOS / CMO
NMOS / CMOS
TTL / PMOS

CMOS / TTL
CMOS / PMOS
CMOS / NMOS
PMOS / TTL

**Fig. 13.66**

Then question comes what are the important points to be considered while interfacing two families.

Scanned by CamScanner

*(1) Output of devices of one family should satisfy current and voltage requirements of the inputs of devices of other family.*

*(2) One has to consider power supply requirement of the both the family. Here one can have different configuration i.e.*

*(a) Both the families operated at same power supply voltage.*

*(b) Both the families operated at different power supply voltage.*

*(3) The other family's outputs should swing as near as possible to the full voltage range of the first family's power supply.*

### 13.25.1 CMOS Interfaced To PMOS :

Points to be considered while Interfacing PMOS and CMOS.

(1) **Power Supply Set :** Most of the popular PMOS parts are specified with 17 V to 24 V power supplies, while maximum power supply voltage for CMOS is 15 V.

(2) The output swing of push pull PMOS output is significantly less than the power supply voltage across it. PMOS swings from very close to its $V_{SS}$ supply (positive supply) to quite a few volts above its $V_{DD}$ supply (negative supply).

So if we reduce supply voltage of PMOS, we expect output swing will go low to match requirement of CMOS. But the case is not so.

**Single Power Supply Operation :**

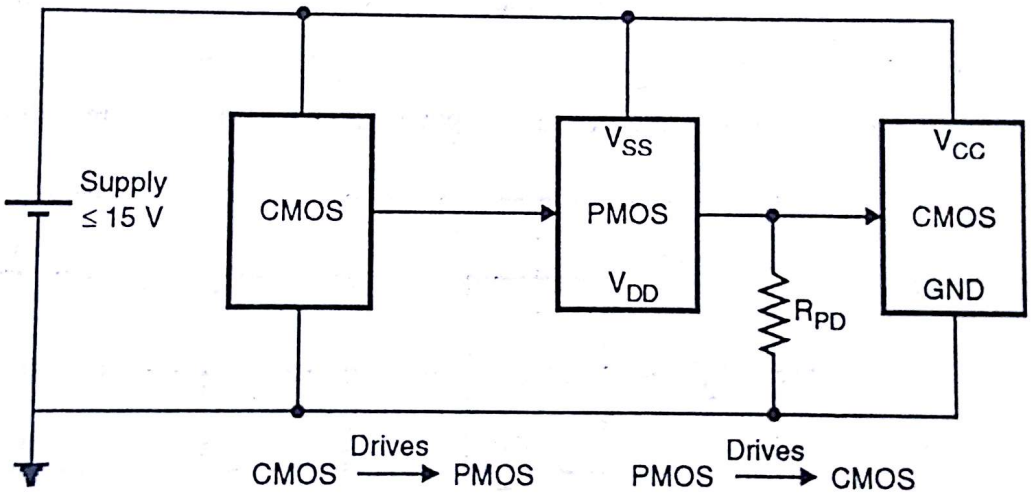



**Fig. 13.67 : Single power supply for CMOS/PMOS**

(1) CMOS drives PMOS directly.

(2) PMOS cannot drive CMOS directly because of its output will not pull down close enough to lower power supply voltage. [Refer point 2, under heading of ***Interface PMOS and CMOS***] Because of this one has to add $R_{PD}$ (R pull down) to each PMOS output to pull it low to lower power supply voltage level.

(3) Selecting $R_{PD}$ : The value should be selected such that it is small enough to give desired RC time constant. But it should not be so small that the PMOS output cannot pull it to upper power supply voltage when it needs to.

(4) The approach works with push pull as well as open drain PMOS outputs.

Scanned by CamScanner

**Using BIAS :**

To match logic swing of the PMOS with CMOS, reduce CMOS power supply voltage. To achieve this zener supply is used to bias up lower power supply voltage of CMOS. This is shown in Fig. 13.68.

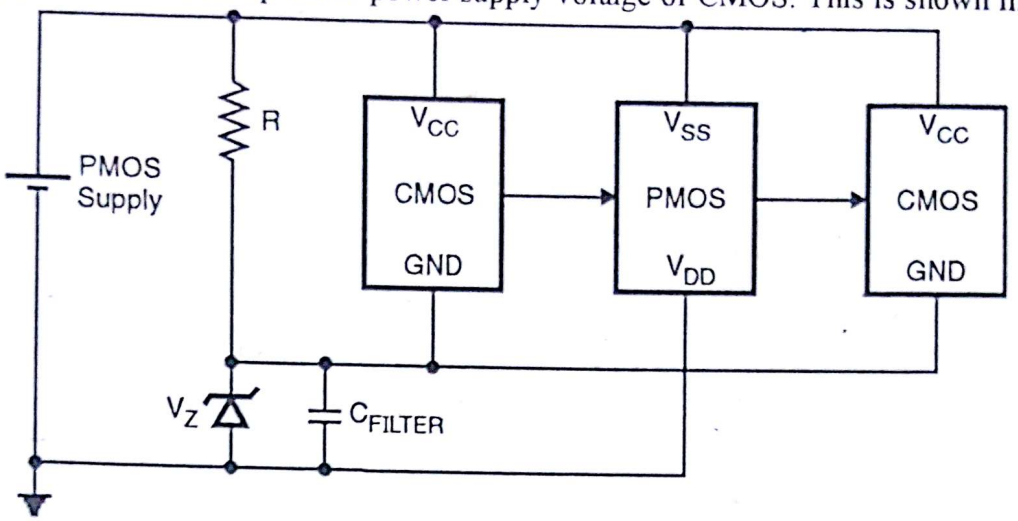



**Fig. 13.68**

GND of CMOS chip is not connected to the GND. Instead it is connected to cathode of zener. In short ground pin of CMOS is lifted by providing bias voltage (by zener) to reduce total voltage across CMOS. This is done to match minimum swing of PMOS output. The CMOS still can drive PMOS directly.

**Using bipolar supply :**

If the system has bipolar logic, it will have two power supplies. The configuration is as shown in Fig. 13.69.

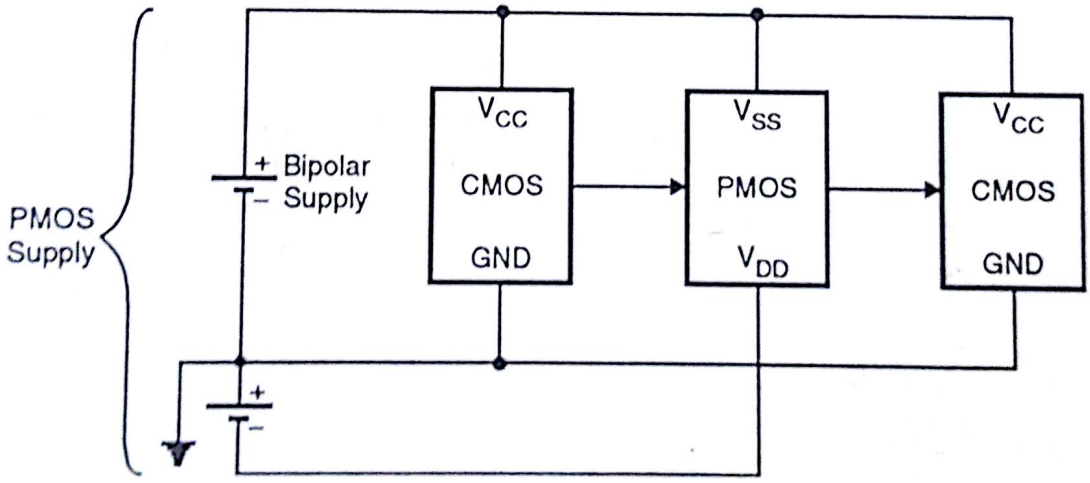



**Fig. 13.69 : Dual power supply operation**

## 13.25.2 CMOS Interfaced to NMOS :

Interfacing CMOS to NMOS is simpler. Supply requirement for NMOS is 5V to 12V. This is directly compatible with CMOS.

Important Point to note is ,

(1) At LOW level, output of NMOS is 1V to 2V above GND level (or low supply voltage).

(2) At HIGH level, output of NMOS is 1 to 2V below $V_{CC}$ level (or upper supply voltage).

Scanned by CamScanner

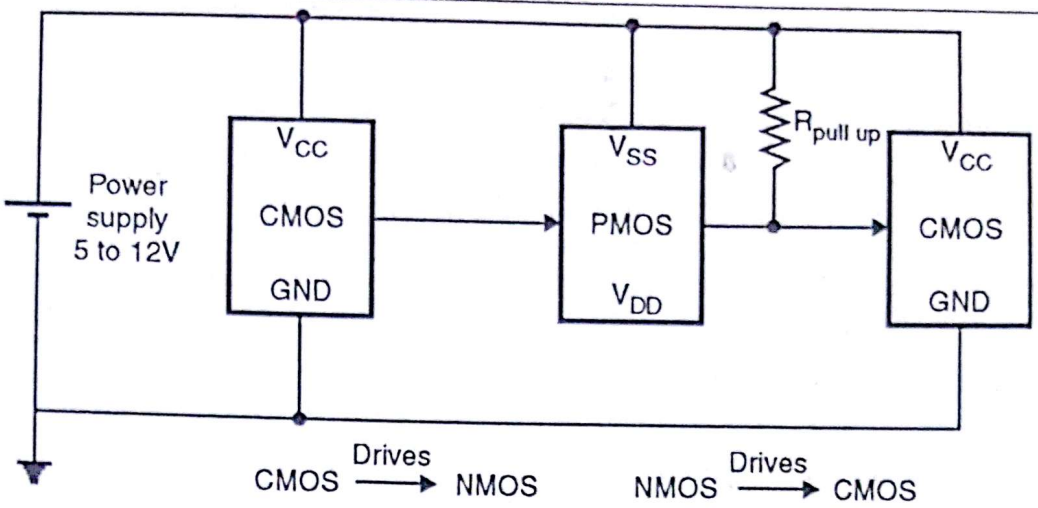



**Fig. 13.70**

Therefore for higher supply voltage NMOS and CMOS can be directly Interfaced, because HIGH logic level output voltage will be 1 to 2V below upper supply voltage. But for lower supply voltage NMOS output for HIGH logic level will be 2 to 4V below upper supply voltage, therefore one has to ***pull*** the output voltage ***up***. This can be done using pull up resistance.

### 13.25.3 TTL to CMOS Interface :

It means TTL gate will drive CMOS gate, therefore one has to know $V_{OH}$, $V_{OL}$ of TTL and $V_{IL}$, $V_{IH}$ of CMOS. The Fig. 13.71 shows the typical values.

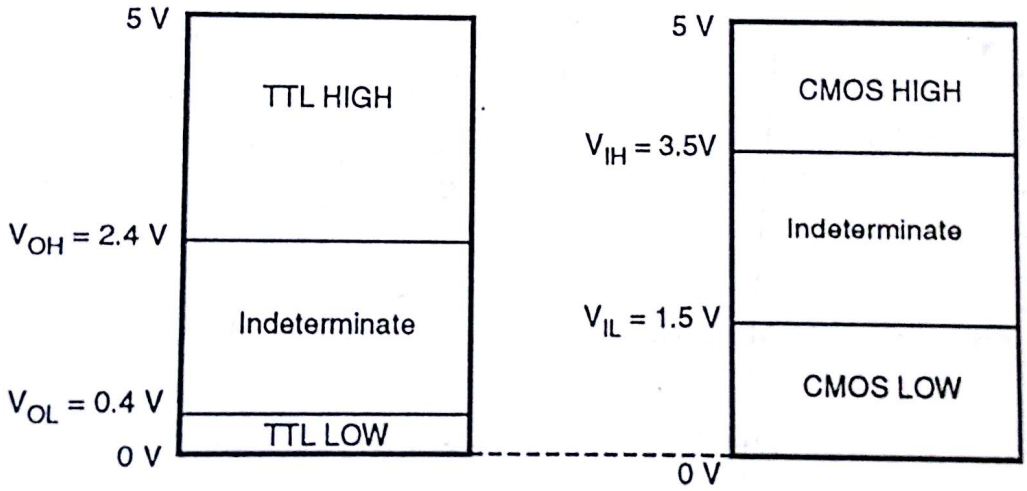



**Fig. 13.71**

If you observe Fig. 13.71, you will find that $V_{OL}$ (TTL) < $V_{IL}$ (CMOS), therefore no problem. But $V_{OH}$ (TTL) falls in ***indeterminate*** region of CMOS because $V_{IH}$ (CMOS) = 3.5V where as $V_{OH}$ (TTL) = 2.4 V. The remedy over this problem is simple, just pull the output voltage of TTL to higher level. ***How you do this*** ? Just by putting $R_{pull\ up}$(Pull up resistance). The same is shown in Figs. 7.70 and Fig. 13.72.

The Fig. 13.72 shows TTL and CMOS operating on single power supply i.e. +5V.

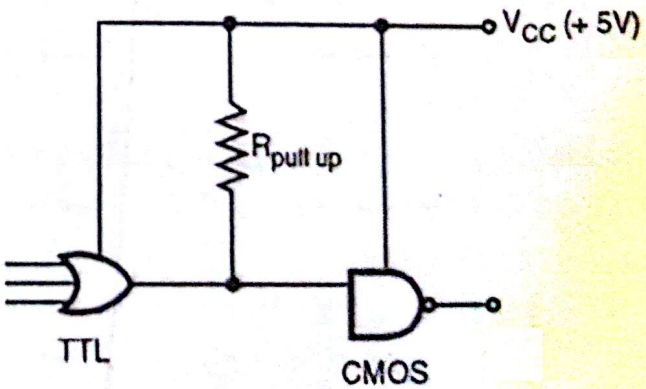



**Fig. 13.72 : TTL to CMOS with pull up resistance**

Scanned by CamScanner

But in some of the cases you may use different power supply for CMOS because CMOS works upto 15V. So the case is TTL operating at +5V and CMOS operating at power supply voltage greater than +5V (typically 9V, 12V or 15V).

Here again the problem will arise of matching $V_{OL}$ with $V_{IL}$ and $V_{OH}$ with $V_{IH}$. Again the remedy is simple, use pull up R touring $V_{OL}$ and $V_{OH}$ of TTL gate up to CMOS level. Interface is shown in Fig. 13.73.

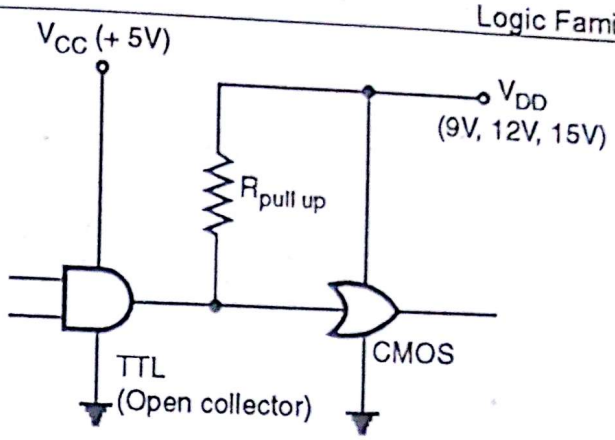



Fig. 13.73

***What should be value of pull up R ?***

For same supply voltage the thumb rule is one can select $R_{pull\ up}$ between 1 K to 4 K 7 Ω.

***How pull up works ?***

**Case I :**

When output of TTL is LOW, it will sink current. Sinking current for 1 TTL load is ≅ 1.6 mA.

Therefore value of R will be $= \frac{5V}{1.6\ mA} = 3125\ \Omega$ OR 3.1 kΩ.

Pull up doesn't change LOW state output voltage drastically.

**Case II :**

When TTL output is in HIGH state [Refer Fig. 13.28.Totem pole output stage]. In this case transistor provides output upto 2.4V, but because of the pull resistance output rises above 2.4 V, therefore transistor will be OFF and final output will be +5V through pull up to input of CMOS.

For different power supply operation.

$$R_{pull\ up} = \frac{12\ V\ (or\ 9V\ or\ 15V)}{1.6\ mA} = 7.5\ k\ \Omega \rightarrow \text{for } 12\ V.$$

## Level Shifter

Sometimes when TTL and CMOS operated at different power supply voltage, level shifter circuit is used.

As name suggests level shifter IC will shift or pull the output voltage level to CMOS input requirement chips available are 40109, MC 14504B.

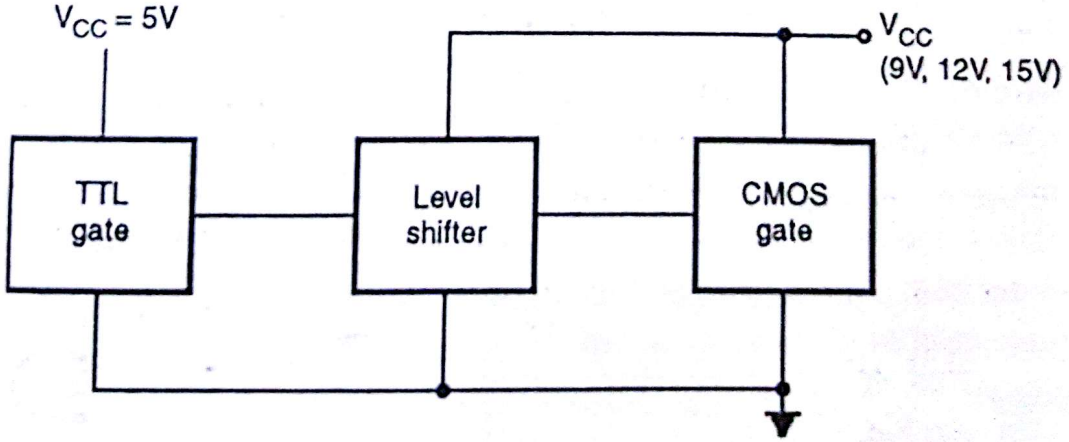



Fig. 13.74

Scanned by CamScanner

### 13.25.4 CMOS to TTL Interface :

**(a) CMOS to Standard TTL :**

CMOS directly cannot drive standard TTL because current required will be – 1.6 m A, where CMOS requires current in μAmp. Therefore one has to increase current sinking capability of CMOS. For that we normally use CMOS buffer. Buffer is a device which will increase current driving capacity.

**(b) CMOS to Low Power TTL :**

For low power TTL worst case low state input current is 180 μ A. CMOS can sink 360 μA, therefore it can drive two low power TTL device.

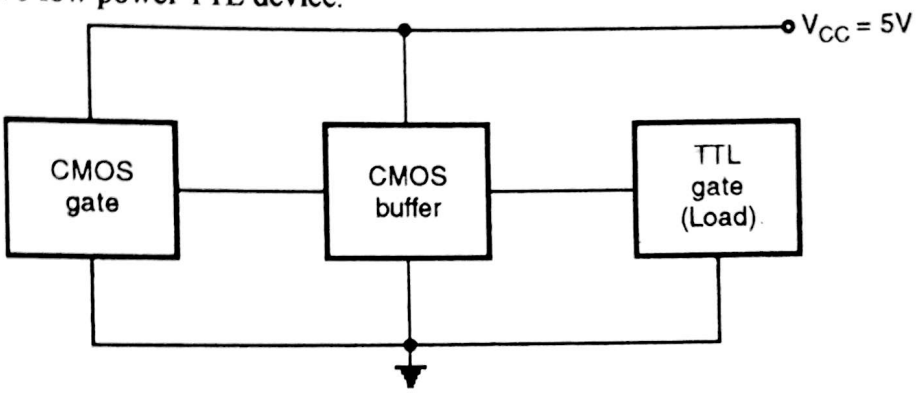



**Fig. 13.75**

## 13.26 BICMOS Digital Circuit :

BICMOS technology combines Bipolar and CMOS circuit on same IC chip. Because of this we get advantage of both technology such as.

(1) Low power
(2) High input impedance } CMOS characteristic
(3) Wide noise margin

(4) High current driving capacity
(5) High speed of operation } TTL characteristic
(6) High package density.

Let's see basic BICMOS logic Inverter.

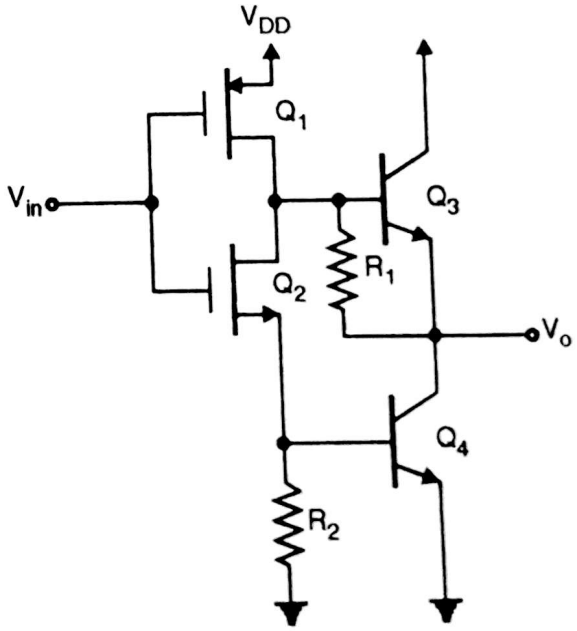



**Fig. 13.76 : Basic inverter using BICMOS**

**Working :**

**Case I :** $V_i = 0$ and $V_o$ floating.

(1) Transister $Q_2$ will be OFF and no current will flow into base of $Q_4$.

(2) As $Q_2$ is OFF, $V_{R_2} = 0$, therefore transistor $Q_4$ = OFF.

(3) PMOS transistor $Q_1$ is ON. This establishes low impedance conducting channel between base of $Q_3$ and $V_{DD}$.

Scanned by CamScanner

(4) As $V_o$ is floating no current will flow through $Q_3$ or $R_1$. Therefore no current flows through $Q_1$. Therefore voltage drop across $Q_1$ will be nearly zero.

(5) Therefore resistor $R_1$ will cause voltage at gate output to be equal to $V_{DD}$. Resistor $R_1$ will work as pull up and $V_{OH} = V_{DD}$.

**Case II :** $v_i = 0$ and $v_o$ connected to load.

(1) Refer step (1), (2) and (3) of case I.

(2) As $v_o$ is connected to load, it has to source current to load.

(3) To source current transister $Q_3$ is ON and acts as emitter follower, providing gate with low output impedance and high current driving capacity.

(4) Current supplied by $Q_1$ to $Q_3$ will be amplified by $\beta$ of $Q_3$.

**Case III :** $v_i = 1$

(1) Transistor $Q_1$ = OFF and $Q_2$ turns ON.

(2) Transistor $Q_2$ supplies current to $Q_4$, $\therefore$ $Q_4$ = ON.

(3) If load is not present, current through $Q_4$ and $Q_2$ will be nearly zero.

(4) Path of $R_1$, conducting channel of $Q_2$ and $R_2$ will give LOW output level.

**Drawback of BICMOS :**

(1) Complex fabrication process.

(2) Because of complexity cost per gate increase than that of CMOS.

**BICMOS NAND Gate :**

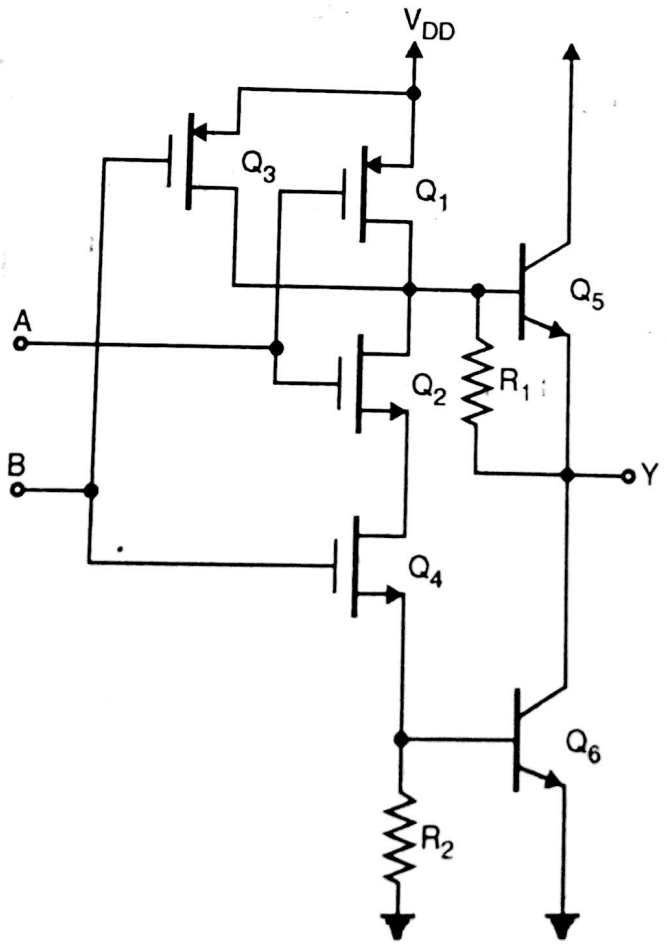



**Fig. 13.77 : BICMOS NAND gate**

Scanned by CamScanner

**Latest series available :**

| Series | Technology |
|---|---|
| 40xx | CMOS |
| 41xx | CMOS |
| 43xx | CMOS |
| 44xx | CMOS |
| 45xx | CMOS |
| 47xx | CMOS |
| 4xxx | CMOS |
| 72xx | CMOS |
| 7xxx | CMOS |
| 74ABT | QUBIC BICMOS technology |
| 74AC xx | advanced CMOS |
| 74ACT xx | advanced CMOS/TTL Interface |
| 74ACQ xx | advanced CMOS Quiet series |
| 74ACTQ xx | advanced CMOS/TTL, Quiet series |
| 74LVC xx | advanced low voltage CMOS logic |
| 74BC xx | high speed BICMOS |
| 74BCT xx | high speed BICMOS/TTL Interface |
| 74C xx | high speed CMOS, 74's pin assignment |
| 74FCT xx | high speed BICMOS/TTL Interface |
| 74HC xx | high speed CMOS, 74's assignment |
| 74HC 40xx | high speed CMOS, 4000's pin assignment |
| 74HC 70xx | high speed CMOS, 7000's pin assignment |
| 74HCT xx | high speed CMOS/TTL Interface, 74's pin assignment |
| 74HCT 40xx | high speed CMOS/TTL Interface, 4000's pin assignment |
| 74HCT 70xx | high speed CMOS/TTL Interface, 7000's pin assignment |
| 74HCU xx | high speed CMOS, backupless |
| 74xx | TTL normal |
| 74FB xx | high speed BICMOS |
| 74GTL xx | Gunning transistor logic |
| 74HLL xx | high speed low voltage CMOS logic |
| 74LCX xx | low voltage high speed CMOS |
| 74LV xx | low voltage CMOS |
| 74LVC xx | low voltage CMOS |
| 74LVQ xx | low voltage Quiet CMOS logic |
| 74LVT xx | Advanced BICMOS Low Voltage technology |
| 74ALS xx | advanced low power schottky TTL |
| 74Fxx | Fast TTL |
| 74ASxx | advanced schottley TLL |
| 74H xx | high speed TTL |
| 74L xx | low power TTL |
| 74LS xx | low power schottky TTL |
| 74S xx | Schottky TTL |
| 74VHC xx | Very high speed CMOS |
| 74VHCT xx | Very high speed CMOS, TTL compatible. |

Scanned by CamScanner

# Dec. 2002

## Section – I

**1. (a) Convert the following into binary, octal and hexadecimal numbers**
**(i) $(2019)_{10}$ (ii) $(345)_{10}$** **(3)**

**Ans. : (i)**

| | | |
|---|---|---|
| 2 | 2019 | |
| | Quotient | Remainder |
| 2 | 1009 | 1 |
| 2 | 504 | 1 |
| 2 | 252 | 0 |
| 2 | 126 | 0 |
| 2 | 63 | 0 |
| 2 | 31 | 1 |
| 2 | 15 | 1 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |
| | ↓ | 1 |

$$(2019)_{10} = [0111\ 1110\ 0011]_2$$

$$= [\ 7\quad E\quad 3\ ]_{16}$$

$$(2019)_{10} = [0111\ 1110\ 0011]_2$$

$$= [\ 3\quad 7\quad 4\quad 3\ ]_8$$

**(ii)**

| | | |
|---|---|---|
| 2 | 345 | |
| | Quotient | Remainder |
| 2 | 172 | 1 |
| 2 | 86 | 0 |
| 2 | 43 | 0 |
| 2 | 21 | 1 |
| 2 | 10 | 1 |
| 2 | 5 | 0 |
| 2 | 2 | 1 |
| 2 | 1 | 0 |
| | ↓ | 1 |

$$(345)_{10} = [101011001]_2$$

$$= [\ 1\quad 5\quad 9\ ]_{16}$$

$$(345)_{10} = [101\ 011\ 001]_2$$

$$= [\ 5\quad 3\quad 1\ ]_8$$

Scanned by CamScanner

**(b) Convert the decimal number $(135)_{10}$ to Gray Code. Explain the uses of Gray Code.** (2)

**Ans. :**

| | Quotient | Remainder |
|---|---|---|
| 2 | 135 | |
| 2 | 67 | 1 |
| 2 | 33 | 1 |
| 2 | 16 | 1 |
| 2 | 8 | 0 |
| 2 | 4 | 0 |
| 2 | 2 | 0 |
| 2 | 1 | 0 |
| | 1 | |

$(135)_{10} = [1\ 0\ 0\ 0\ 0\ 1\ 1\ 1]_2$

$[1\ 0\ 0\ 0\ 0\ 1\ 1\ 1]_2$ → binary

$[1\ 1\ 0\ 0\ 0\ 1\ 0\ 0]_{gray}$

Gray code is MOST WIDELY used in optical encoder, mainly CNC machines.

**(c) Subtract the following using 1's complement and 2's complement method and express the result as hexadecimal number.** (3)

**Ans. :** $(62)_8 - (29)_{10}$

$[62]_8 = [50]_{10} = [110010]_2$

$[29]_{10} = [011101]_2$

(i) 1's complement method

1's complement of $(011101)_2$ is $(100010)_2$

$(50)_{10} = 1\ 1\ 0\ 0\ 1\ 0$

$+$

$(29)_{10} = 1\ 0\ 0\ 0\ 1\ 0$

$1\ 0\ 1\ 0\ 0$

$1$

$0\ 1\ 0\ 1\ 0\ 1$ → Answer

Scanned by CamScanner

(ii) Two's complement = 1's complement + 1

$= (100010)_2 + 1$

$= (100011)_2$

$$
\begin{array}{rcl}
(50)_{10} & = & 1\ 1\ 0\ 0\ 1\ 0 \\
 & + & \\
(29)_{10} & = & 1\ 0\ 0\ 0\ 1\ 1 \\
\hline
 & & \boxed{1}\ \underbrace{0\ 1\ 0\ 1\ 0\ 1}_{\text{Answer}}
\end{array}
$$

Discard

**(d) Perform the following operations directly without converting the number to any other base**

**(i) $(ABC)_{16} \times (25)_{16}$ (ii) $(33)_8 - (77)_8$** **(4)**

**Ans. :** (i)

$$
\begin{array}{ccccccl}
 & & A & B & C & \longrightarrow & (2748)_{10} \\
 & & \times & 2 & 5 & \longrightarrow & (37)_{10} \\
\hline
1 & 5 & 7 & 8 & X & & \\
 & 3 & 5 & A & C & & \\
\hline
1 & 8 & D & 2 & C & \longrightarrow & (1,01,676)_{10}
\end{array}
$$

(ii)

$$
\begin{array}{rcc}
 & \overset{2}{\cancel{3}} & \overset{1}{3} \\
- & & \\
 & 7 & 7 \\
\hline
\boxed{1} & 5 & 6
\end{array}
$$

' borrow

$\therefore$ $(56)_8$ is negative number as there is borrow.

**(e) Obtain an even parity Hamming Code for data code 1010111.** **(6)**

**Why is Hamming Code called error correcting code?**

**Ans.: Hamming Code :**

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $P_2$ | $P_1$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |

| Parity | Error bit |
|---|---|
| (1) 1, 3, 5, 7, should form even parity | |
| 1 1 1 1 ⟶ yes, ∴ reset error bit. | 0 |
| (2) 2, 3, 6, 7 should form even parity | |
| 1 1 1 1 ⟶ No, ∴ set error bit | 1 |
| (3) 4, 5, 6, 7 should form even parity | |
| 0 1 0 1 ⟶ yes, ∴ reset error bit. | 0 |

Scanned by CamScanner

$\therefore$ error bit is $(010)_2$, $\therefore P_2$

Corrected code is $\longrightarrow (1010101)_2$.

Hamming code is called as error correcting code, because after receiving the code, it doesn't only check for error bit, but also corrects it.

**2. (a) Simplify the following :** (6)

**(i)** $A + \bar{A}B + ABC + \bar{A}\bar{B}CD$ **(ii)** $\bar{A}C\,[\,\overline{\bar{A}BD}\,] + \bar{A}\,B\,\bar{C}\,\bar{D} + A\bar{B}C$

**(iii)** $A\bar{B} + ABC + A(B + A\bar{B})$

**Ans. :** (i) $A + \bar{A}B + ABC + \bar{A}\bar{B}CD$

$= A + B + ABC + \bar{A}\bar{B}CD$

$= A\,[1 + BC] + B + \bar{A}\bar{B}CD$

$= A + B + \bar{A}CD$

$= A + \bar{A}CD + B = A + CD + B$

$= A + B + CD$

(ii) $\bar{A}C\left[\overline{\bar{A} \cdot BD}\right] + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$

$= \bar{A}C\,[\,A + \overline{B\bar{D}}] + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$

$= \bar{A}C\,[A + [\bar{B} + \bar{D}] + \bar{A}\,B\,\bar{C}\,\bar{D} + A\bar{B}C$

$= 0 + \bar{A}C\,[\bar{B} + \bar{D}] + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$

$= \bar{A}\bar{B}C + \bar{A}C\bar{D} + \bar{A}B\bar{C}\bar{D} + A\bar{B}C$

$= \bar{B}C\,[A + \bar{A}] + \bar{A}\bar{D}\,[\,C + B\bar{C}]$

$= \bar{B}C + \bar{A}\bar{D}\,[B + C]$ [As $A + \bar{A}B = A + B$]

(iii) $A\bar{B} + ABC + A(B + A\bar{B})$

$= \overline{A[\bar{B} + BC]} + A\,(B + A\bar{B})$

$= \overline{A[\bar{B} + BC]} + A(B + A)$

$= \overline{A \cdot (\bar{B} + C)} + AB + A$

$= \overline{\bar{A} \cdot (\,\bar{B} + C\,)} + A(B + 1) = \overline{\bar{A} \cdot [B\bar{C}]} + A$

$= \overline{A + \bar{A}B\bar{C}} = A + B\bar{C} = \bar{A} \cdot (\,B\bar{C}\,) = \bar{A}[\bar{B} + C]$

Scanned by CamScanner

**(b) A corporation having 100 shares entitles the owner of each share to cast one vote at the share holders meet. Assume that A has 40 shares, B has 30 shares, C has 20 shares and D has 10 shares. A two third majority is required to pass a resolution in a share holders meeting Each of these four men has a switch which he closes to vote YES and opens to vote NO for his percentage of shares. When the resolution is passed, the output LED must be ON Derive a truth table for the output function and give the sum of product equation for it Also implement using only NAND gates.** (10)

**Ans. :** (i) A = 40 shares

B = 30 shares

C = 20 shares

D = 10 shares

(ii) 2/3 majority is required to pass a resolution, means 66.66 or more number of shares will be required.

(iii) closed ⟶ yes ⟶ logic 1

open ⟶ No ⟶ logic 0.

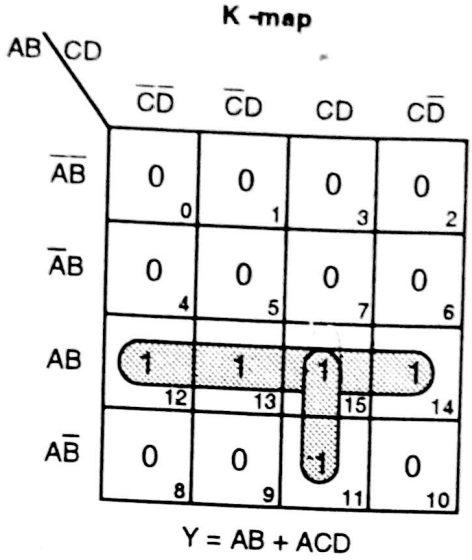
(iv) When resolution is passed, output LED will glow, means output expected is ACTIVE HIGH. Let's label y as an outputs.

(v) Truth table

| Inputs | | | | Output |
|---|---|---|---|---|
| **A** | **B** | **C** | **D** | **Y** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 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 |

Scanned by CamScanner

(vi)




$Y = AB + ACD$

(vii)

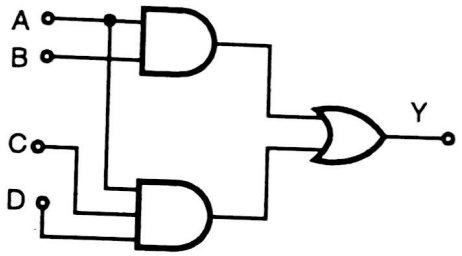



**Fig. 1**

(viii)

$$Y = \overline{\overline{AB + ACD}}$$

$$= \overline{\overline{AB} \cdot \overline{ACD}}$$

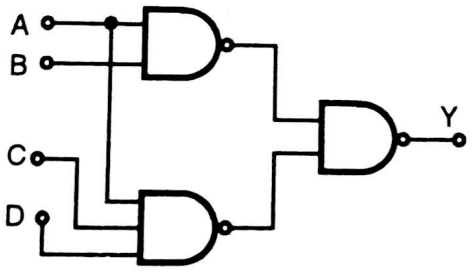



**Fig. 2**

**3. (a) Explain the terms : (i) prime implicant (ii) input variable (iii) min term (iv) max term.** **(4)**

**Ans. :**

1. Prime Implicant : Refer Section 4.9 page no. 4-61.
2. Input variable : It is input to a system and attain any value (logic 0 or logic 1) at any instant of time.
3. Minterm : Refer Section 3.5 (ii) page no. 3-18.
4. Maxterm : Refer Section 3.5 (iii) page no. 3-18.

**(b) Use Quine–McCluskey method to find minimum POS expression for**

$$f(w, x, y, z) = \pi M(1, 3, 4, 5, 9, 10, 11) + \pi d(6, 8)$$ **(8)**

Scanned by CamScanner

**Ans. :** Quine –McCluskey method

**Step 1 :** $f(w, x, y, z) = \pi M(1, 3, 4, 5, 9, 10, 11) \cdot \pi d(6, 8)$

| Decimal No. | w | x | y | z |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 |

**Step 2 :** Group maxterms according to number of zeros.

| | Decimal | w | x | y | z | |
|---|---|---|---|---|---|---|
| √ | $M_{11}$ | 1 | 0 | 1 | 1 | ⟶ Single zero |
| √ | $M_3$ | 0 | 0 | 1 | 1 | Two zeros |
| √ | $M_5$ | 0 | 1 | 0 | 1 | |
| √ | $M_6$ | 0 | 1 | 1 | 0 | |
| √ | $M_9$ | 1 | 0 | 0 | 1 | |
| √ | $M_{10}$ | 1 | 0 | 1 | 0 | |
| √ | $M_1$ | 0 | 0 | 0 | 1 | Three zeros |
| √ | $M_4$ | 0 | 1 | 0 | 0 | |
| √ | $M_8$ | 1 | 0 | 0 | 0 | |

**Step 3 :**

| Group | Maxterms | w | x | y | z | |
|---|---|---|---|---|---|---|
| 1 | $M_{11}, M_3$ | X | 0 | 1 | 1 | √ |
| | $M_{11}, M_9$ | 1 | 0 | X | 1 | √ |
| | $M_{11}, M_{10}$ | 1 | 0 | 1s | X | √ |
| 2 | $M_3, M_1$ | 0 | 0 | X | 1 | √ |
| | $M_5, M_1$ | 0 | X | 0 | 1 | $\rightarrow P_A$ |
| | $M_5, M_4$ | 0 | 1 | 0 | X | $\rightarrow P_B$ |
| | $M_6, M_4$ | 0 | 1 | X | 0 | $\rightarrow P_C$ |
| | $M_9, M_1$ | X | 0 | 0 | 1 | √ |
| | $M_9, M_8$ | 1 | 0 | 0 | X | √ |
| | $M_{10}, M_8$ | 1 | 0 | X | 0 | √ |

Scanned by CamScanner

**Step 4 :**

| Group | Maxterms | w | x | y | z | |
|---|---|---|---|---|---|---|
| 1 | $M_{11}, M_3, M_9, M_1$ | X | 0 | X | 1 | $\rightarrow P_D$ |
| | $M_{11}, M_9, M_3, M_1$ | X | 0 | X | 1 | |
| | $M_{11}, M_{10}, M_9, M_8$ | 1 | 0 | X | X | $\rightarrow P_E$ |
| | $M_{11}, M_9, M_{10}, M_8$ | 1 | 0 | X | X | |

**Step 5 :**

| | Prime Implicants | Maxterms | | | | | | | Don't Care | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $M_1$ | $M_3$ | $M_4$ | $M_5$ | $M_9$ | $M_{10}$ | $M_{11}$ | $M_6$ | $M_8$ |
| $P_A$ | 0 X 0 1 | X | | | X | | | | | |
| $P_B$ | 0 1 0 X | | | X | X | | | | | |
| $P_C$ | 0 1 X 0 | | | X | | | | | X | |
| $P_D$ | X 0 X 1 | X | X | | | X | | X | | |
| $P_E$ | 1 0 X X | | | | | X | X | X | | X |

Select, $P_B$, $P_D$ and $P_E$; it will cover all the maxterms. $M_6$ and $M_8$ are don't care, ∴ optional.

$$f(w, x, y, z) = \{0\ 1\ 0\ X,\ X\ 0\ X\ 1,\ 1\ 0\ X\ X\}$$

$$f(w, x, y, z) = (w + \bar{x} + y)\,(x + \bar{z})\,(\bar{w} + x)$$

**(c) Implement the above using 2 i/p NOR gate only.** (4)

**Ans.:**

$$f(y) = \overline{(w + \bar{x} + y)\,(x + \bar{z}) + (\bar{w} + x)}$$

$$= \overline{(w + \bar{x} + y)} + \overline{(x + \bar{z})} + \overline{(\bar{w} + x)}$$

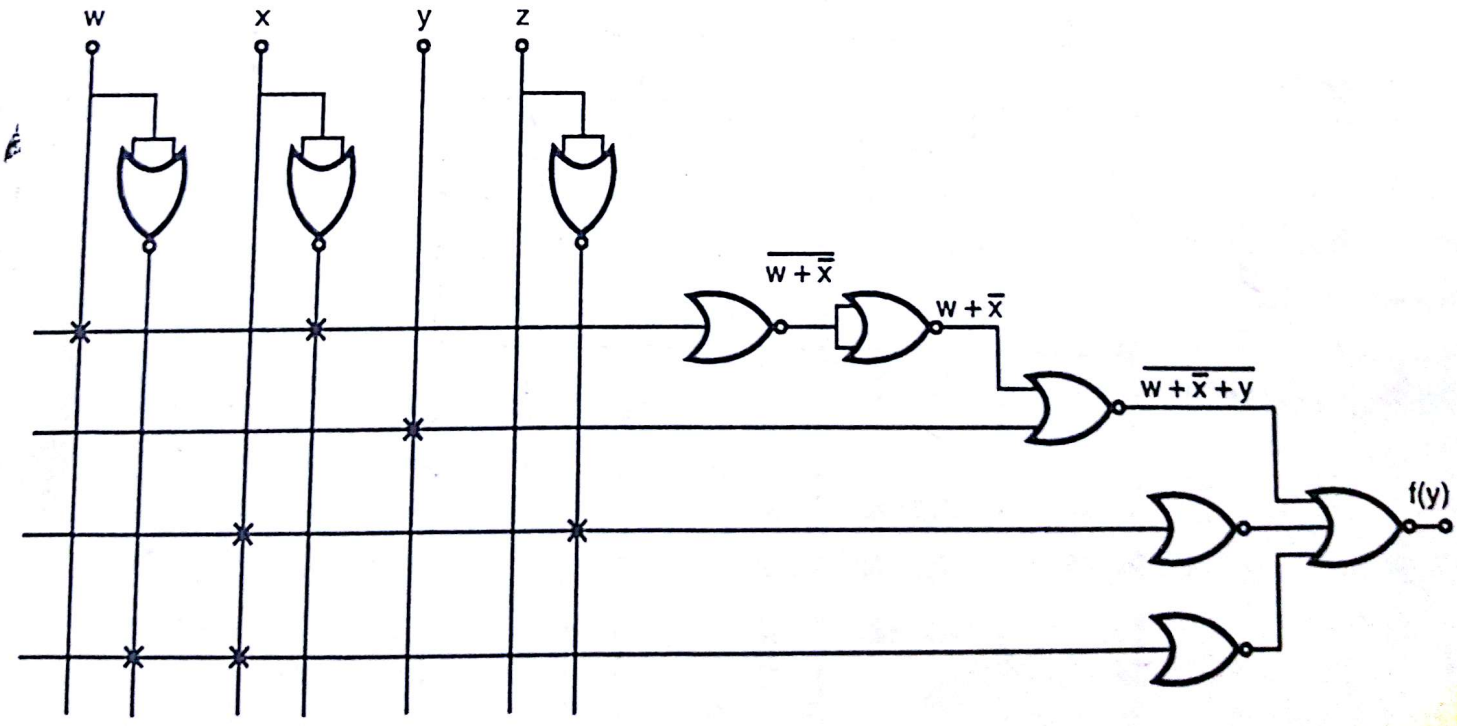



**Fig. 3**

Scanned by CamScanner

**4. (a) Design an XS–3 adder circuit using IC 7483 and required gates.** **(10)**

**Ans.** : Refer Section 6.4

**(b) Why are NAND and NOR gates called universal gates?** **(3)**

**Ans. :** Refer Section 3.10

**(c) Prove that a two level AND–OR circuit can be replaced by NAND–NAND circuit and a two level OR–AND circuit can be replaced by a NOR–NOR circuit.** **(3)**

**Ans. :** To prove this we will take one example

(i)

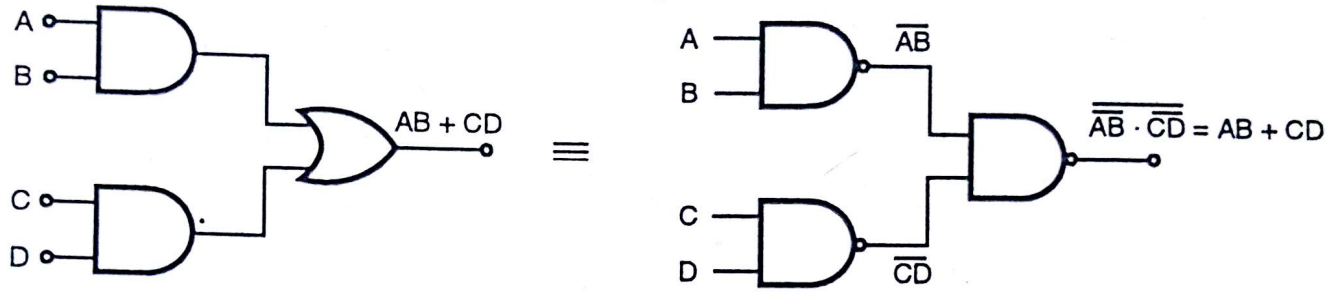



**Fig. 4**

(ii)

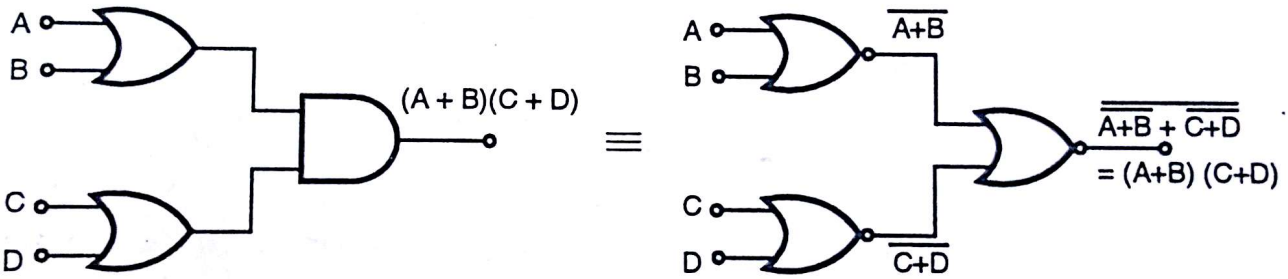



**Fig. 5**

**5. Write short notes on :** **(16)**

**(i) Carry Look Ahead carry Generator (ii) Arithmetic Logic Unit (ALU).**

**(iii) Parity Generator/checker (iv) Gray Code.**

**Ans. :** (i) Refer Section 6.2.3

(ii) Refer Section 7.5

**(iii) Parity Generator/checker :** Now we will see the circuit which will generate parity and also checks it. Normally EX–OR gate is used to generate even parity. Refer turth table of EX-OR gate.

| A | B | Y Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Scanned by CamScanner

As shown when odd no. of 1's are present in the input, output of EX OR gate (i.e. Y) is '1'. This will set total numbers of '1's to even number.

The circuit in Fig. 5A shows even parity generator, for 4 bit data input.

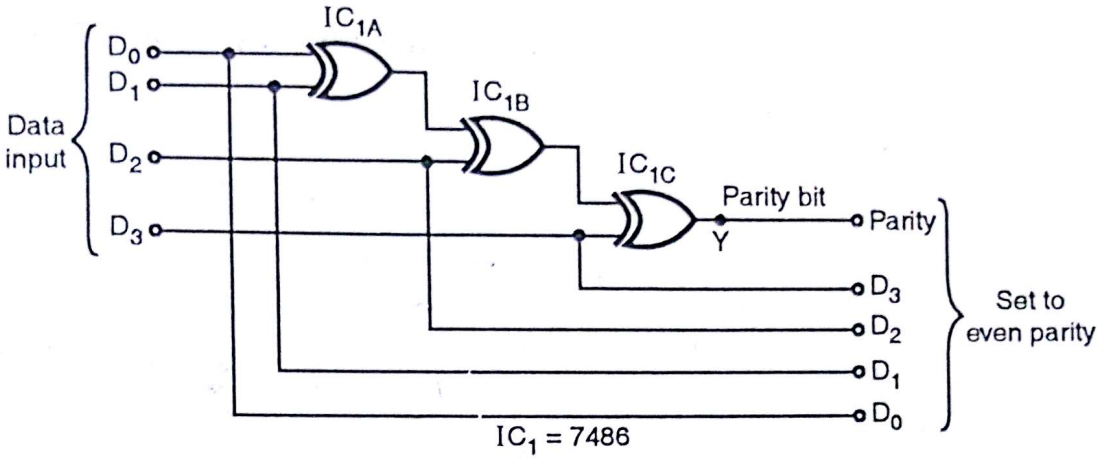



**Fig. 5A**

The circuit is implemented using 2 input EX–OR gate. Input to the circuit is data bits $D_0$ –$D_3$. Output of the circuit is $D_3 - D_0$ with parity bit. If $D_3 - D_0$ = 0 1 1 1, then output Y = 1.

| ∴ | $D_3$ | $D_1$ | $D_2$ | $D_0$ | Parity | |
|---|---|---|---|---|---|---|
| | 0 | 1 | 1 | 1 | 1 | → even number of '1's |

But if $D_3 - D_0$ = 1 0 0 1, then output Y = 0

| ∴ | $D_3$ | $D_1$ | $D_2$ | $D_0$ | Parity | |
|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | 0 | → even number of '1's |

Now we will see parity checker circuit in Fig. 5B. The parity checker circuit also employ EXOR gate.

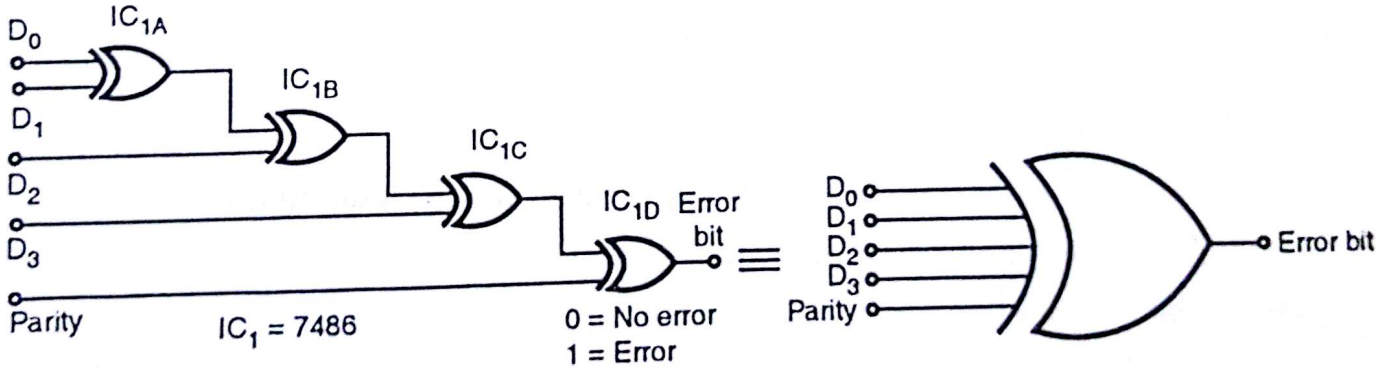



**Fig. 5B**

The parity checker circuit, checks for even parity. If input bits ($D_3 - D_0$, parity) has even parity, error bit = 0, otherwise for odd parity, error bit = 1. The circuit is shown using 2 input EXOR gate. The same can be implemented using 5 input EXOR gate also.

(iv) Refer Section 1.11.2.

Scanned by CamScanner

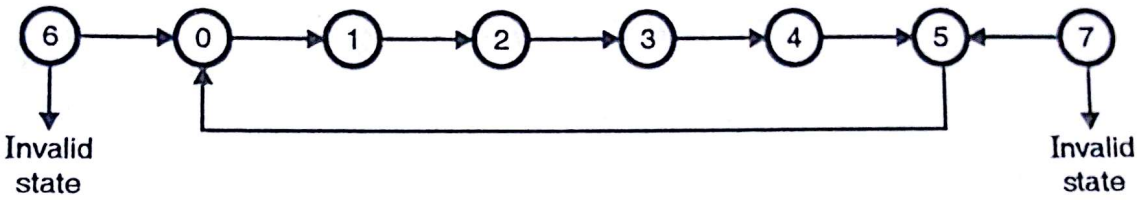
## Section – II

**6. Design a mod–6 synchronous up/down counter using SR flip–flops. If the counter enters any of the unused states what will happen? Ensure that the counter does not enter an unused state. Clearly specify value of mode control for up counting and down counting. Also draw state diagram.** **(16)**

**Ans.:** A mod - 6 counter has 6 states, 0 to 5. The number of flip–flops required would be 3. The invalid states will be 110 and 111.

If the counter enters an invalid state, it might never return to a valid state and may encounter a LOCK OUT condition.

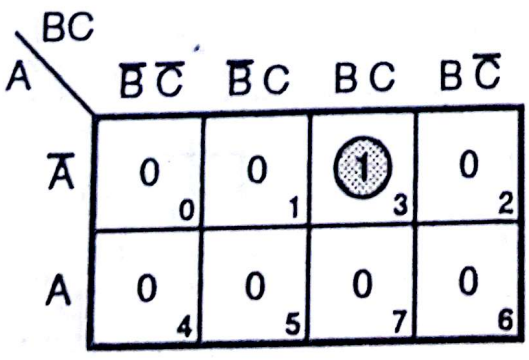
To ensure that a counter does not enter an unused state, we shall design the counter to return to valid state after one clock tick if it enters an unused state.



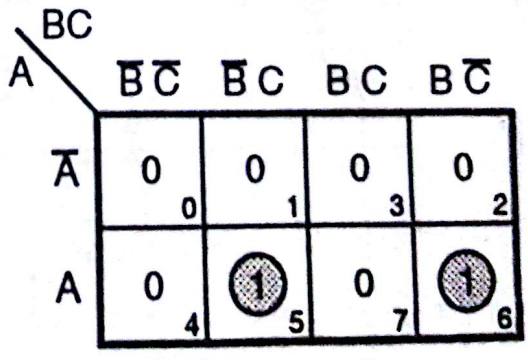

**Truth Table :**

| A | B | C | $S_A$ | $R_A$ | $S_B$ | $R_B$ | $S_C$ | $R_C$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

**K-maps :**

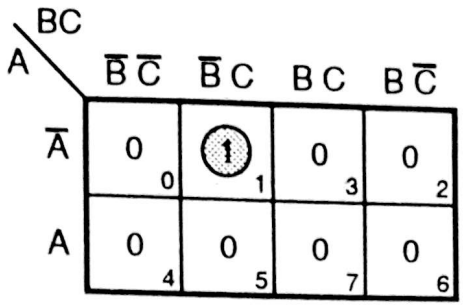



$\therefore S_A = \overline{A}BC$

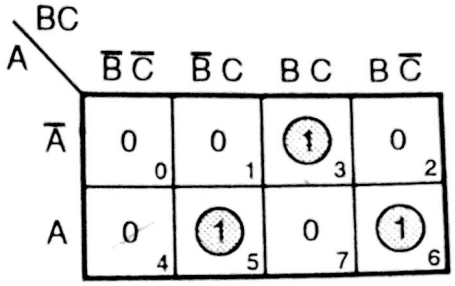



$R_A = B \oplus C$

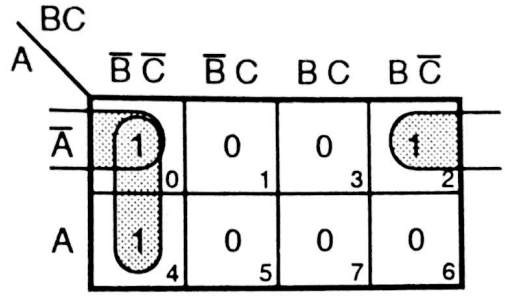
Scanned by CamScanner



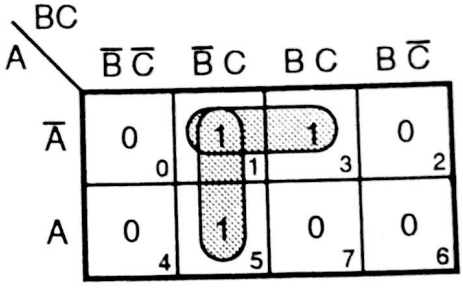

$S_B = \overline{A}\,\overline{B}\,C$




$R_B = \overline{A}BC + A\overline{B}C + AB\overline{C}$
$= \overline{A}BC + A(B \oplus C)$




$S_C = \overline{B}\,\overline{C} + \overline{A}\,\overline{C}$
$= \overline{C}(\overline{A} + \overline{B})$




$R_C = \overline{B}C + \overline{A}C$
$= C(\overline{A} + \overline{B})$

**Implementation :**

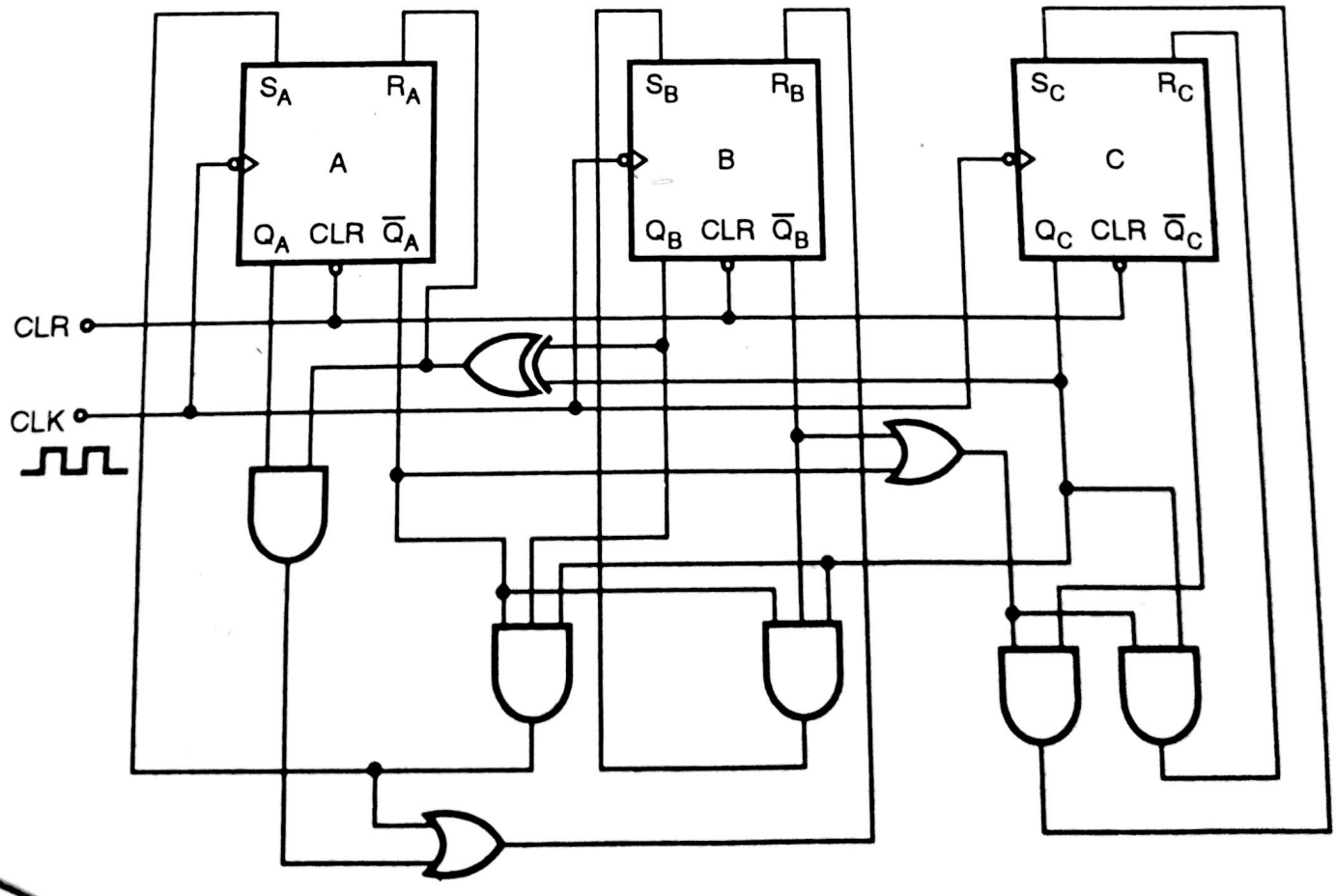



**Fig. 6**

Scanned by CamScanner

**7. (a) Design a clocked J–K flip–flip using a basic NAND cell using NAND gates only.** **(8)**

**Ans. :** Refer Section 8.5.

**(b) What is race condition ? Why does it occur? What modification is required if race condition is to be avoided?** **(4)**

**Ans. :** Refer Section 8.5.1.

**(c) Convert a J–K flip–flop to D flip–flop and a T flip–flop.** **(4)**

**Ans. :** Refer Ex. 7(i) for JK to D FF

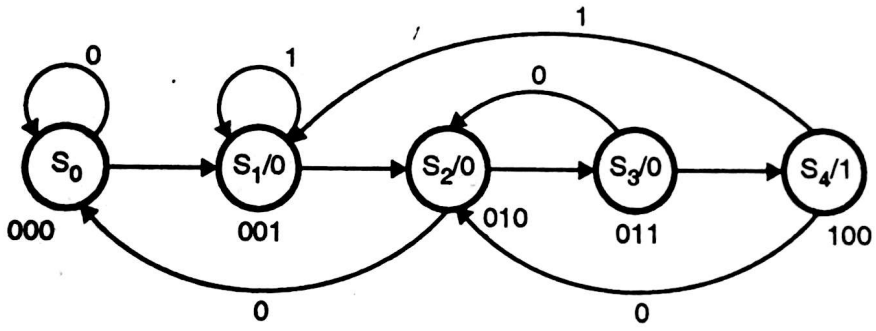
Refer Ex. 6(i) for JK to T FF

**8. (a) Draw the diagram of a 4 bit universal shift register and explain its operation.** **(10)**

**Ans. :** Refer Section 10-10 Page no. 10-16. Diagram 10.20 (b) is important.

**(b) Draw the state diagram of a sequence detector 1011 in an input string 101101100010110. The correct pattern can overlap with another pattern. When the input pattern is detected, it causes an o/p z to be asserted high.** **(6)**

**Ans. :**




**State table :**

| Input | Previous state | | | Next state | | | Output |
|---|---|---|---|---|---|---|---|
| | $C_0$ | $B_0$ | $A_0$ | $C_1$ | $B_1$ | $A_1$ | Z |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

Scanned by CamScanner

| $D_0C_0$ \ $B_0A_0$ | $\overline{B}_0\overline{A}_0$ | $\overline{B}_0A_0$ | $B_0A_0$ | $B_0\overline{A}_0$ |
|---|---|---|---|---|
| $\overline{D}_0\overline{C}_0$ | 0 0 | 0 1 | 0 3 | 0 2 |
| $\overline{D}_0C_0$ | 0 4 | X 5 | X 7 | X 6 |
| $D_0C_0$ | 0 12 | X 13 | X 15 | X 14 |
| $D_0\overline{C}_0$ | 0 8 | 0 9 | 1 11 | 0 10 |

$$Z = D_0B_0A_0$$

| Previous state | | | | Next state | | | $J_A$ | $K_A$ | $J_B$ | $K_B$ | $J_C$ | $K_C$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $D_0$ | $C_0$ | $B_0$ | $A_0$ | $C_1$ | $B_1$ | $A_1$ | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X | 0 | X |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X | 0 | X | 0 | X |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | X | 1 | 1 | X | 0 | X |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | X | 0 | 0 | X | 0 | X |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | X | X | 1 | 0 | X |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | X | X | 0 | 0 | X |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | X | 1 | X | 0 | 0 | X |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | X | 0 | X | 0 | 1 | X |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | X | 1 | X | X | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | X | 0 | X | X | 1 |

**K-maps :**

| $D_0C_0$ \ $B_0A_0$ | $\overline{B}_0\overline{A}_0$ | $\overline{B}_0A_0$ | $B_0A_0$ | $B_0\overline{A}_0$ |
|---|---|---|---|---|
| $\overline{D}_0\overline{C}_0$ | 0 0 | X 1 | X 3 | 0 2 |
| $\overline{D}_0C_0$ | 0 4 | X 5 | X 7 | X 6 |
| $D_0C_0$ | 1 12 | X 13 | X 15 | X 14 |
| $D_0\overline{C}_0$ | 1 8 | X 9 | X 11 | 1 10 |

$$J_A = D_0$$

| $D_0C_0$ \ $B_0A_0$ | $\overline{B}_0\overline{A}_0$ | $\overline{B}_0A_0$ | $B_0A_0$ | $B_0\overline{A}_0$ |
|---|---|---|---|---|
| $\overline{D}_0\overline{C}_0$ | X 0 | 1 1 | 1 3 | X 2 |
| $\overline{D}_0C_0$ | X 4 | X 5 | X 7 | X 6 |
| $D_0C_0$ | X 12 | X 13 | X 15 | X 14 |
| $D_0\overline{C}_0$ | X 8 | 0 9 | 0 11 | X 10 |

$$K_A = \overline{D}_0$$

743

Scanned by CamScanner

| $D_0C_0$ \ $B_0A_0$ | $\overline{B}_0\overline{A}_0$ | $\overline{B}_0A_0$ | $B_0A_0$ | $B_0\overline{A}_0$ |
|---|---|---|---|---|
| $\overline{D}_0\overline{C}_0$ | 0 (0) | 1 (1) | X (3) | X (2) |
| $\overline{D}_0C_0$ | 1 (4) | X (5) | X (7) | X (6) |
| $D_0C_0$ | 0 (12) | X (13) | X (15) | X (14) |
| $D_0\overline{C}_0$ | 0 (8) | 0 (9) | X (11) | X (10) |

$J_B = \overline{D}_0$

| $D_0C_0$ \ $B_0A_0$ | $\overline{B}_0\overline{A}_0$ | $\overline{B}_0A_0$ | $B_0A_0$ | $B_0\overline{A}_0$ |
|---|---|---|---|---|
| $\overline{D}_0\overline{C}_0$ | X (0) | X (1) | 0 (3) | 1 (2) |
| $\overline{D}_0C_0$ | X (4) | X (5) | X (7) | X (6) |
| $D_0C_0$ | X (12) | X (13) | X (15) | X (14) |
| $D_0\overline{C}_0$ | X (8) | X (9) | 0 (11) | 0 (10) |

$K_B = \overline{A}_0\overline{D}_0$

| $D_0C_0$ \ $B_0A_0$ | $\overline{B}_0\overline{A}_0$ | $\overline{B}_0A_0$ | $B_0A_0$ | $B_0\overline{A}_0$ |
|---|---|---|---|---|
| $\overline{D}_0\overline{C}_0$ | 0 (0) | 0 (1) | 0 (3) | 0 (2) |
| $\overline{D}_0C_0$ | X (4) | X (5) | X (7) | X (6) |
| $D_0C_0$ | X (12) | X (13) | X (15) | X (14) |
| $D_0\overline{C}_0$ | 0 (8) | 0 (9) | 1 (11) | 0 (10) |

$J_C = D_0B_0A_0$

| $D_0C_0$ \ $B_0A_0$ | $\overline{B}_0\overline{A}_0$ | $\overline{B}_0A_0$ | $B_0A_0$ | $B_0\overline{A}_0$ |
|---|---|---|---|---|
| $\overline{D}_0\overline{C}_0$ | X (0) | X (1) | X (3) | X (2) |
| $\overline{D}_0C_0$ | 1 (4) | X (5) | X (7) | X (6) |
| $D_0C_0$ | 1 (12) | X (13) | X (15) | X (14) |
| $D_0\overline{C}_0$ | X (8) | X (9) | X (11) | X (10) |

$K_C = 1$

**Implementation :**

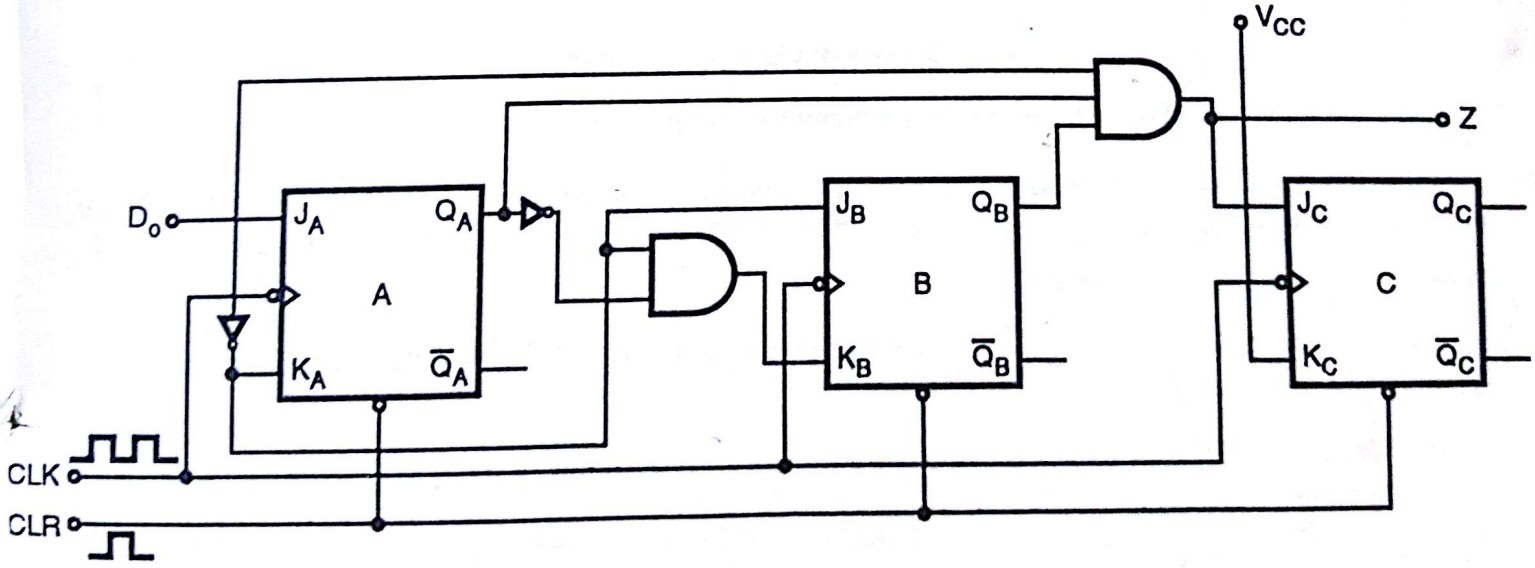



**Fig. 7**

Scanned by CamScanner

**9. (a) Design a mod 5 asynchronous up counter using J–K flip–flop. Explain what modifications are necessary to make it work as down counter. What is glitch problem? (6)**

**Ans. :** Refer Page No. 9-82, Ex. 15. (Procedure is same for mod 5 asynchronous up counter)

**(b) Define the various typical characteristics of a TTL gate. (4)**

**Ans. : (i) ($V_{OH}$) High level output voltage :** It is the minimum voltage which will available at an output terminal of gate, when required output should be at high level logic.

**$V_{OL}$ (Low level output voltage) :** It is the maximum voltage which will be available at output terminal of gate.

**(ii) Input voltage parameter :**

(a) $V_{IH}$ – It is the minimum positive input voltage level that can be applied to an input terminal of device and be recognised as logic high level.

(b) $V_{IL}$ – It is the maximum positive input voltage level that can be applied to the input terminal of a device and be recognised as logic low level.

On the similar line of voltage, we can define current also i.e. $I_{OH}$, $I_{OL}$, $I_{IH}$ and $I_{IL}$

**(iii) Noise margin :** The amount of extraneous signal which can be tolerated, before an output voltage of gate deviates from allowable logic voltage level.

**(iv) Fan out :** Number of load gates connected at the output of gate is called fan out of the gate. It is driving capability of a particular gate to drive 'N' no. of gates.

**(v) Fan in :** Number of inputs connected to gate is called fan in of the gate.

**(vi) Speed of operation or propagation delay :** It is nothing but time taken by particular gate to produce output, when input is applied.

**(vii) Power dissipation :** It is very important parameter of the logic family. Normally it should be as low as possible. The power dissipated by gate HIGH and LOW state is different.

$$P_{D(avg)} = \frac{P_{DH} + P_{DL}}{2}$$

$P_{DH} \rightarrow$ Power dissipated when input is high.

$P_{DL} \rightarrow$ Power dissipated when input is low.

**(viii) Figure of merit or speed power product or power delay product :** The product of propagation time and power dissipation ($t_{dp} \times P_D$) is known as figure of merit or performance of gate. Normally minimum value of product is desired.

**(ix) Operating temperature :** The temperature range in which IC will function properly is called operating temperature for the IC.

**(c) Draw the diagram of a 4 bit twisted ring counter, explaining its operation with the help of a timing diagram. (6)**

**Ans. :** Refer Section 10.7

743 LIBRARY

Scanned by CamScanner

**10. Write short notes on any three :** **(18)**

**(a) PAL and PLA**

**(b) Tristate registers**

**(c) Master Slave JK FF**

**(d) Transition table, State table and State diagrams.**

**Ans. :** (a) The comparision of PAL and PLA is as follows :

| S. No. | PAL | PLA |
|---|---|---|
| 1. | PAL is called as Programmable Array Logic. | PLA is called as programmable Logic Array. |
| 2. | It is in the form of OR-AND matrix. | It is in the form of AND-OR matrix. |
| 3. | PAL is used in CPLD's (Complex Programmable Logic Devices). | Next version of PLA is FPGA's (Field Programmable Gate Array) and FPLA's (Field Programmable Logic Array). |
| 3. | It is moderately complex. | It is highly complex. |
| 4. | It is typically used for small circuits. | It is used for complex circuits. |

**(b) Tristate registers :** Tristate registers is same as tristate buffer. So refer to Chapter 13, Section 13.13.1.

**(c) Master Slave JK FF :** Refer to Section 8.7.

**(d) Transition table OR State transition table :** Transition table specifies required state behavior in binary form. This is constructed from state assignment and state table or diagram.

For state table and state diagram, Refer to chapter 8, Section 8.2.3 and 8.2.4.

❑❑❑

Scanned by CamScanner